Simple Temporal Networks with Partially Shrinkable Uncertainty (Extended Version)

Andreas Lanz[1], Roberto Posenato[2], Carlo Combi[2], Manfred Reichert[1]
[1]*Institute of Databases and Information Systems, University of Ulm, Germany*
[2]*Department of Computer Science, University of Verona, Italy*

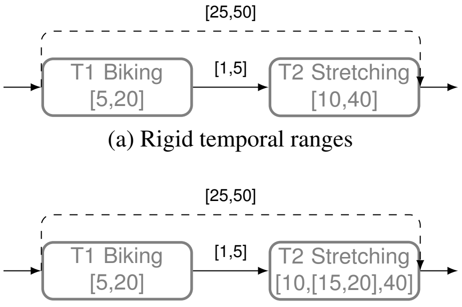
Keywords: Simple Temporal Constraint Network with Uncertainty, STNU, Controllability, Guarded Constraints

Abstract: The Simple Temporal Network with Uncertainty (STNU) model focuses on the representation and evaluation of temporal constraints on time-point variables (timepoints), of which some (i.e., contingent timepoints) cannot be assigned (i.e., executed by the system), but only be observed. Moreover, a temporal constraint is expressed as an admissible range of delays between two timepoints. Regarding the STNU model, it is interesting to determine whether it is possible to execute all the timepoints under the control of the system, while still satisfying all given constraints, no matter when the contingent timepoints happen within the given time ranges (controllability check). Existing approaches assume that the original contingent time range cannot be modified during execution. In real world, however, the allowed time range may change within certain boundaries, but cannot be completely shrunk. To represent such possibility more properly, we propose Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU) as an extension of STNU. In particular, STNPSUs allow representing a contingent range in a way that can be shrunk during run time as long as shrinking does not go beyond a given threshold. We further show that STNPSUs allow representing STNUs as a special case, while maintaining the same efficiency for both controllability checks and execution.

1 INTRODUCTION

For more than a decade, the temporal constraint community has focused on the concept of *controllability* (Morris et al., 2001). Given a set of temporal constraints, of which each is expressed as an admissible range of delays between two time-point variables (timepoints for short), we distinguish two types of constraints: *contingent* and *requirement constraints*. The latter represent the standard temporal constraints, where both timepoints are under control of the system that "executes" the timepoints according to the assigned constraints (i.e., the system fixes the timepoints on the time line). This means that, during execution, the range admissible for some timepoints could be restricted by the system as it depends on the execution of already executed timepoints. In turn, contingent constraints are related to pairs of timepoints of which one (i.e., the *contingent timepoint*) is not under control of the system. Contingent timepoints are either given by the environment (Morris et al., 2001), i.e., they are related to uncontrollable, but expected, events, or by an external agent (i.e., human or software) who may decide autonomously when to execute the contingent timepoint. Considering this scenario, the attention of the temporal constraint community has moved from the problem of *consistency*, which consists of determining whether there exists an execution of all timepoints satisfying all given constraints (Dechter et al., 1991), to the problem of *controllability*; i.e., to determine whether it is possible to execute all timepoints under the control of the system, while satisfying all given constraints, no matter when the contingent timepoints happen within their given time ranges (Morris et al., 2001).

Most contributions from literature assume that the original time range of a contingent constraint cannot be modified during execution. Thus there is no difference between contingent timepoints given by the environment and the ones executed by external agents. In the real world, however, it is quite common that during execution the allowed time range may change, although it cannot be completely shrunk. To represent the behavior of external agents more properly, we may assume that an agent accepts certain reductions (i.e., modifications) of the initial execution range, as long as these do not go beyond a given threshold. In other words, there is an unshrinkable range of execution time the agent can always use. Further, this range is included into a larger one, the system may shrink during execution. The basic idea of our approach is to represent the fact that both the agent and the system



(a) Rigid temporal ranges

(b) Flexible temporal ranges

Figure 1: A simple physiotherapy. Range $[x,y]$ represents the minimum and maximum allowed duration (in minutes) for the corresponding activity.

are aware that some timepoints of the larger time range may be removed before starting the agent's activity. For example, consider a physiotherapy (cf. Fig. 1) consisting of two subsequent activities, namely Biking and Stretching, with one overall temporal constraint. The first activity has an allowed duration range, while its actual duration is decided by the physiotherapist according to the patient's state. The second activity, i.e., the stretching exercise, is performed by the patient over a time period, which is decided by another therapist who considers both the state of the patient and the goal of the therapy. Let us assume that the given ranges are as depicted in Fig. 1 (a): activities are visualized as rounded boxes and subsequent activities are linked to their predecessor through a directed arc. Temporal constraints are represented through arcs together with their related ranges. Activities Biking and Stretching have possible durations within ranges $[5,20]$ and $[10,40]$, respectively, which are autonomously decided by therapists. However, the overall therapy must be within range $[25,50]$, assuming that it may take between 1 and 5 time units to start Stretching after ending Biking. Note that for this scenario it can be easily verified that the corresponding temporal network is not controllable, as there is no way to ask the second activity to have a duration depending on the actual duration of the first activity.

As more realistic representation of this scenario, the second therapist may accept that the allowed duration range may be shrunk during execution, while guaranteeing that the "core" range $[15,20]$ can be always applied when executing Stretching. This scenario is depicted in Fig. 1 (b) where the range is represented as $[10,[15,20],40]$, highlighting the non-shrinkable part. One can easily observe that in this case the network can be executed in a way satisfying all constraints, while still allowing the therapists to autonomously choose the durations of the involved activities.

This paper discusses how to represent and deal with the described extension of contingent constraints in simple temporal constraint networks with uncertainty (STNUs), i.e., temporal networks that allow representing both requirement and contingent constraints (Morris et al., 2001). In addition to dynamic controllability, we discuss that there are no alternative representations of such shrinkable contingent constraints based on compositions of standard requirement and contingent constraints. Moreover, we generalize shrinkable constraints to represent time ranges having certain "guards" on their possible lower and upper bounds.

The paper is organized as follows. Sect. 2 considers existing proposals relevant for our work and provides background information on STNUs. Sect. 3 introduces guarded constraints and the corresponding Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU). Sect. 4 discusses the expressiveness of STNPSUs. Finally, Sect. 5 concludes with a summary and outlook.

2 BACKGROUND AND RELATED WORK

A Simple Temporal Network (STN) (Dechter et al., 1991) is a directed weighted graph where a node represents a time-point variable (timepoint), usually corresponding to the start or end of activities, and an edge represents a lower and an upper bound constraint on the distance between the two timepoints it connects. Each STN is associated with a *distance graph*, derived from the upper and lower bound constraints, where a constraint between a pair of timepoints X and Y is represented as two edges: $X \xrightarrow{v} Y$, representing the constraint $Y \leq X + v$, and $X \xleftarrow{-u} Y$, which stands for $Y \geq X + u$, $u, v \in \mathbb{R}$. An STN is denoted as *consistent* if it is possible to execute each node, i.e., to assign a real value to each timepoint such that all temporal constraints are satisfied. The consistency property can be verified by searching for *negative loops* in the graph. It is well known that consistency checking as well as determining the earliest/latest value of each timepoint can be done in polynomial time (Dechter et al., 1991).

To properly represent events that cannot be executed, but only observed, (Morris et al., 2001) introduced Simple Temporal Networks with Uncertainty (STNUs). STNUs augment STNs with *contingent timepoints* representing timepoints whose value is decided by the environment. Each contingent timepoint has one incoming edge, called *contingent link*, which is labeled by a time range. Therefore, any contingent timepoint may assume a value from a bounded range, but the exact value is decided by the environment at run time. (Morris et al., 2001) provided a formal semantics for the *dynamic controllability*, which is discussed in detail in Sect. 2.1. Moreover, (Morris et al., 2001) presented a *pseudo-polynomial-time* algorithm, called *DC-checking algorithm*, that determines whether a given STNU is *dynamically controllable (DC)*. Further, (Morris and Muscettola, 2005) proposed the first polynomial DC-checking algorithm, which operates in $O(n^5)$ time, where n is the number of timepoints. In this paper, we denote this algorithm as MM5. In turn, (Morris, 2006) presented an $O(n^4)$-time and (Morris, 2014) an $O(n^3)$-time DC-checking algorithm for STNUs, i.e., there are two interesting optimizations of the MM5 algorithm not further discussed in this paper.

(Tsamardinos et al., 2003) introduced the Conditional Temporal Problem (CTP) that augments an STN with *observation timepoints*. Thereby, each observation timepoint is associated with a boolean proposition. When executing the observation timepoint, the truth value of its proposition becomes known. Moreover, each timepoint of a CTP has a propositional label, which corresponds to a conjunction of (positive or negative) literals. A timepoint may only be executed in *scenarios* for which its propositional label evaluates to true. (Tsamardinos et al., 2003) proposed a formal semantics for the *dynamic consistency* property. The latter holds if there exists a strategy for executing the timepoints of the network in a way such that all constraints will be satisfied no matter how the observations turn out.

(Moffitt and Pollack, 2007) proposed three extensions of STNUs. One of them is the *partial shrinkage* in which *observation events* may trigger the reduction of contingent time ranges. In more detail, at design time it is possible to assign different time ranges and a special timepoint, called *observation*, to each contingent link; then, at run time, the respective observation timepoint chooses the time range to be considered for the contingent link before enabling the contingent timepoint. This way it becomes possible to customize the temporal range in which a contingent timepoint may occur. Further, it is shown how the concept of dynamic controllability can be expanded to capture such extensions. An analysis regarding the costs of respective DC-checking algorithm was not provided. The STNPSU model presented in this paper is similar in the sense that it allows one to choose the time range to be considered for a contingent link during run time. However, its execution semantics is quite different.

(Hunsberger et al., 2012) combined the features of STNUs and CTPs in order to create a Conditional Simple Temporal Network with Uncertainty (CSTNU). For CSTNUs, dynamic controllability was defined in a way generalizing the respective notions for STNUs and CTPs. Since existing DC-checking algorithms for STNUs and CTPs work differently, they cannot simply be combined to yield a DC-checking algorithm for CSTNUs. Instead, (Combi et al., 2013) proposed a novel algorithm (even if not complete) that incorporates new rules taking into account the propositional truth values generated by the observation timepoints.

(Lanz et al., 2013) showed how CSTNUs can be applied in the context of time-aware business processes in order to verify their controllability at both design and run time. Concerning temporal aspects of a business process, it is emphasized that activity durations usually represent worst case estimates, which are either based on the experience of a domain expert or extracted from process logs; further, the execution times of most activities can be shortened if required. Accordingly, one may assume that an activity has a *flexible* maximum duration $MaxD_F$ that may be restricted up to a *contingent* minimum and maximum duration range $[MinD_C, MaxD_C]$. In other words, they proposed and analyzed a mapping of time-aware business processes to CSTNU in which activity durations are expressed in terms of shrinkable time intervals $[[MinD_C, MaxD_C]MaxD_F]$.

2.1 Dynamic Controllability of STNUs

As proposed by (Morris et al., 2001), an STNU is a set of time-point variables (timepoints) and temporal constraints together with a set of contingent links. Each contingent link has the form (A,x,y,C), where A and C are timepoints and $0 < x < y < \infty$ holds. A is called the *activation timepoint* and C the *contingent timepoint*. Once A is executed, C is guaranteed to be executed such that $C - A \in [x,y]$ holds. However, the particular time at which C is executed is uncontrollable since it is decided by the environment; i.e., it can be only observed when it happens.

Let $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ be an STNU, with $\mathcal{T}$ being a set of timepoints, $\mathcal{C}$ a set of constraints, and $\mathcal{L}$ a set of contingent links. The corresponding graph for $\mathcal{S}$ has the form $(\mathcal{T}, \mathcal{E}, \mathcal{E}_\ell, \mathcal{E}_u)$. Thereby, each timepoint in $\mathcal{T}$ serves as a node in the graph; $\mathcal{E}$ is a set of *ordinary* edges; $\mathcal{E}_\ell$ is a set of *lower-case* and $\mathcal{E}_u$ 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.

An STNU is *dynamically controllable* if there exists a strategy for executing its timepoints, in a way guaranteeing that all constraints in the network can be satisfied, no matter how the durations of the contingent links actually turn out. The strategy is dynamic since its execution decisions can react to observations of contingent links that have already been completed, while excluding those not completed yet.

This section presents preliminary notions and introduces the dynamic controllability of an STNU as defined in (Morris et al., 2001) and subsequently fixed in (Hunsberger, 2009).

For an STNU, a *situation* specifies fixed durations for all contingent links.

Definition 1 (Situations)**.** *Let $\mathcal{S}$ be an STNU comprising k contingent links, $(A_1,x_1,y_1,C_1),\ldots,(A_k,x_k,y_k,C_k)$, with corresponding duration ranges $[x_1,y_1],\ldots,[x_k,y_k]$. Then: $\Omega_{\mathcal{S}} = [x_1,y_1] \times \ldots \times [x_k,y_k]$ is called the space of situations for $\mathcal{S}$. Any $\omega = (d_1,\ldots,d_k) \in \Omega_{\mathcal{S}}$ is called a situation. Where possible, we may write Ω instead of $\Omega_{\mathcal{S}}$.*

The concept of *schedule* formalizes the execution of timepoints.

Definition 2 (Schedule)**.** *A* schedule *for an STNU is a mapping $\psi : \mathcal{T} \to \mathbb{R}$ that assigns a real number to each timepoint in $\mathcal{T}$.*

Given a situation ω for an STNU, the replacement of its contingent links by the durations specified in ω determines a projection of the STNU onto situation ω.

Definition 3 (Situation Projection for an STNU)**.** *Suppose $\mathcal{S} = (\mathcal{T},\mathcal{C},\mathcal{L})$ is an STNU and $\omega = (d_1,\ldots,d_k)$ a situation. The* projection *of $\mathcal{S}$ onto ω—denoted as sitPrj$(\mathcal{S},\omega)$—is the STN $(\mathcal{T},\mathcal{C}')$ with:*

$$\mathcal{C}' = \mathcal{C} \cup \{(d_i \le C_i - A_i \le d_i) \mid 1 \le i \le k\}$$

Given an STNU, multiple schedules may exist. We are interested in finding a strategy that determines schedules that satisfy all constraints in any situation.

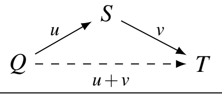
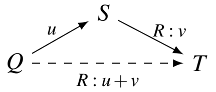
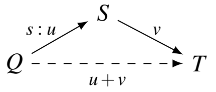
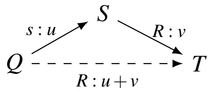
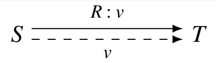
Definition 4 (Execution Strategy for an STNU)**.** *Let $\mathcal{S} = (\mathcal{T},\mathcal{C},\mathcal{L})$ be an STNU. An* execution strategy *for $\mathcal{S}$ is a mapping $\sigma : \Omega \to \psi$ such that for each situation $\omega \in \Omega$, $\sigma(\omega)$ is a (complete) schedule for the timepoints in $\mathcal{T}$. Furthermore, if for each situation ω schedule $\sigma(\omega)$ is a solution for the situation projection sitPrj$(\mathcal{S},\omega)$, then σ is called* viable. *In any case, the execution time of timepoint X in schedule $\sigma(\omega)$ is denoted as $[\sigma(\omega)]_X$.*

A situation history for an STNU specifies the durations of all contingent links that have finished their execution prior to a time t in schedule $\sigma(\omega)$.

Definition 5 (Situation History for an STNU)**.** *Let $\mathcal{S} = (\mathcal{T},\mathcal{C},\mathcal{L})$ be any STNU, σ any execution strategy for $\mathcal{S}$, ω any situation, and t any real number. The* history *of t in situation ω for strategy σ—denoted as sitHst(t,ω,σ)—is defined as follows:*

$$\begin{aligned}\mathit{sitHst}(t,\omega,\sigma) = \{&(A,C,[\sigma(\omega)]_C - [\sigma(\omega)]_A) \mid \\ &\exists x,y \text{ such that } (A,x,y,C) \in \mathcal{L} \land [\sigma(\omega)]_C < t\}\end{aligned}$$

Table 1: Edge-generation rules of the MM5 algorithm. Dashed edges are the generated ones.

<table>
<tr><td><b>No Case:</b></td><td> </td></tr>
<tr><td><b>Upper Case:</b></td><td> </td></tr>
<tr><td><b>Lower Case:</b></td><td>
Applicable if: $v < 0 \lor (v = 0 \land S \not\equiv T)$</td></tr>
<tr><td><b>Cross Case:</b></td><td>
Applicable if: $R \not\equiv S \land (v < 0 \lor (v = 0 \land S \not\equiv T))$</td></tr>
<tr><td><b>Label Removal:</b></td><td>
Applicable if: $v \ge -x$, x is the lower bound for the contingent link from T to R</td></tr>
</table>

Definition 6 (Dynamic Execution Strategy for an STNU)**.** *An execution strategy σ for an STNU is called* dynamic *if for any situations, ω_1 and ω_2, and any non-contingent timepoint X, it holds:*

$$\begin{aligned}&\mathit{sitHst}([\sigma(\omega_1)]_X,\omega_1,\sigma) = \mathit{sitHst}([\sigma(\omega_1)]_X,\omega_2,\sigma)\\ &\quad\Rightarrow [\sigma(\omega_1)]_X = [\sigma(\omega_2)]_X.\end{aligned}$$

Definition 7 (Dynamic Controllability of an STNU)**.** *An STNU $\mathcal{S}$ is called* dynamically controllable *(DC) if there exists an execution strategy for $\mathcal{S}$ that is both viable and dynamic.*

In order to determine whether an STNU is dynamically controllable, (Morris and Muscettola, 2005) proposed a polynomial-time checking algorithm, MM5, which works by recursively generating new edges in the STNU graph according to the rules from Table 1 and checking whether newly added edges determine negative loops in the graph. For each rule, existing edges are represented as solid arrows and newly ones as dashed arrows. Each of the first four rules takes two existing edges as input and generates a single edge as output. Finally, notation $R \not\equiv S$ expresses that R and S must be distinct time-point variables, and does not represent a constraint on the *values* of those variables.

We observe that the edge-generation rules from Table 1 only generate ordinary or upper-case edges. The

```
Procedure MM5-DC-Check($G$)
Input: $G = (\mathcal{T}, \mathcal{C}, \mathcal{L})$: STNU graph instance to analyze.
Output: the controllability of $G$.
for 1 to $|\mathcal{T}|^2 + |\mathcal{T}||\mathcal{L}| + |\mathcal{L}|$ do
    if (AllMax matrix inconsistent) then return false;
    Generate new edges using rules from Table 1;
    if (no edges generated) then return true;
return false;
```

upper-case edges generated by respective rules represent conditional constraints, called *waits* (Morris et al., 2001). In particular, an upper-case edge $B\xrightarrow{C:-v}A$ represents the following constraint: as long as contingent timepoint C remains unexecuted, timepoint B must wait at least v units after the execution of A, the activation timepoint for C.

Procedure MM5-DC-Check shows the pseudocode of the MM5 DC-checking algorithm. Its time complexity is $O(n^5)$ (Morris and Muscettola, 2005).

2.2 Alternative Characterization of an Execution Strategy

As observed in (Hunsberger, 2009), the original definition of *dynamic execution strategy* (DES) obscures the real-time features of typical execution scenarios and the kinds of execution decisions an execution system may make. Therefore, (Hunsberger, 2009) proposed an alternative characterization of a DES to not only represent the conditions under which a system must make real-time execution decisions, but also the outcomes of those decisions. Two kinds of real-time execution decisions (RTEDs) are defined: WAIT and (τ, χ), which can be described as: "Wait until some contingent duration completes" or "If nothing happens before τ, then execute the (executable) timepoints in χ." The outcome of a RTED depends on the situation and is represented by a partial schedule that specifies the execution of one or more additional timepoints. The outcome of a WAIT decision solely involves the execution of contingent timepoints, whereas the outcome of a (τ, χ) decision may involve the execution of contingent as well as non-contingent timepoints. An RTED-based strategy is defined as a mapping from partial schedules to real-time execution decisions. (Hunsberger, 2009) proved that RTED-based strategies correspond one-to-one to DESs.

In more detail, given an STNU and a *partial* schedule $\psi : \mathcal{T} \to \mathbb{R}$ (i.e., the domain of ψ may be a subset of $\mathcal{T}$), we denote by $\mu(\psi) = \max\{\psi(t) \mid t \in Dom(\psi)\}$ the maximum execution time of timepoints appearing in ψ, by $U(\psi) = \{x \mid x \notin Dom(\psi)\}$ the set of unexecuted timepoints in ψ, by $U^x(\psi) \subseteq U(\psi)$ the set of non-contingent unexecuted timepoints, by $U^c(\psi) \subseteq U(\psi)$ the set of contingent unexecuted timepoints, and by $U^a(\psi) \subseteq U^c(\psi)$ the set of contingent activated unexecuted timepoints, respectively.

Let ψ be a partial schedule for an STNU $\mathcal{S}$ and $\omega = (\omega_1, \dots, \omega_q)$ a situation for $\mathcal{S}$. ψ respects ω if for each contingent link (A_i, x, y, C_i) one of the following conditions holds: (1) neither A_i nor C_i appear in ψ; (2) only A_i appears in ψ, and $\psi(A_i) + \omega_i > \mu(\psi)$; or (3) both A_i and C_i appear in ψ, and $\psi(A_i) + \omega_i = \psi(C_i)$. ψ is called *respectful* if it respects at least one situation. If ψ is both respectful and partial, it is called a respectful, partial schedule (RPS). A strategy σ is respectful if for each ω, $\sigma(\omega)$ respects ω.

Let us recall the definition of WAIT and (τ, χ) decisions.

WAIT Decision. Let ψ be some RPS for $\mathcal{S}$ such that $U^a(\psi)$ is non-empty. Then WAIT is an admissible RTED.

Outcome of a WAIT Decision. If $U^a(\psi) \neq \emptyset$ and ω is a situation respected by ψ, then the *time* at which the *next contingent* timepoint will execute is defined as $tnc(\psi, \omega) = \min\{\psi(A_i) + \omega_i \mid C_i \in U^a(\psi)\}$. With $\chi^a(\psi, \omega) = \{C_i \in U^a(\psi) \mid \psi(A_i) + \omega_i = tnc(\psi, \omega)\}$, we denote the non-empty set of contingent timepoints that will be executed at time $tnc(\psi, \omega)$. Then, the outcome of the WAIT decision for ψ in situation ω is defined to be the execution of contingent timepoints at time $tnc(\psi, \omega)$: $\psi \cup \{(C_i, tnc(\psi, \omega)) \mid C_i \in \chi^a(\psi, \omega)\}$.

(τ, χ) **Decision**. Let ψ be some RPS for $\mathcal{S}$ such that $U^x(\psi) \neq \emptyset$. If $\tau > \mu(\psi)$ and χ is a non-empty subset of $U^x(\psi)$, then (τ, χ) is an admissible RTED for ψ.

Outcome of a (τ, χ) **Decision**. Let ω be a situation respected by ψ. The outcome of a (τ, χ) decision depends on the relationship between $tnc(\psi, \omega)$ and instant τ. For the sake of simplicity, let $\tau^c = tnc(\psi, \omega)$ and $\chi^a = \chi^a(\psi, \omega)$ if $U^a(\psi) \neq \emptyset$; otherwise, let $\tau^c = \infty$. If $\tau^c < \tau$, the outcome solely involves the execution of the contingent timepoints in χ^a. In turn, if $\tau < \tau^c$, the outcome solely involves the execution of the non-contingent timepoints in χ. Finally, for $\tau^c = \tau$, the outcome involves the execution of the timepoints in both χ^a and χ.

(Hunsberger, 2009) proved that the original dynamic execution strategy can be described in terms of RTEDs as shown in procedure RTEDExecutionStrategy. Thereby, function RTExecutionDecision is used to determine the next RTED. For the sake of brevity, the RTED WAIT is represented as (τ, χ) decision with $\tau := \infty$ and $\chi := \emptyset$ in the given context.

Starting with a partial schedule $\psi = \{(Z, 0)\}$, which only fixes the initial timepoint Z, procedure RTEDExecutionStrategy iteratively determines an RTED $\delta(\psi)$, considering two possibilities (cf. function RTExecutionDecision). If all executable time-

```
Function RTExecutionDecision($\mathcal{S}, \psi$)
Input: $\mathcal{S}$: STNU. $\psi$; partial schedule
Output: $\delta(\psi)$: real-time execution decision
if ($U^x(\psi) = \emptyset$) then                                   // WAIT RTED!
    $\delta(\psi) := (\tau^x, \chi)$, where $\tau^x := \infty$ and $\chi := \emptyset$;
else                                                                 // $(\tau,\chi)$ RTED!
    foreach ($x \in U^x(\psi)$) do
        $[m(x), M(x)]$ = current time-window for $x$;
        $W(x) := -\infty$;
        foreach ($(A_i, C_i) \mid C_i \in U^a(\psi) \wedge x \xrightarrow{C_i:-w_i} A_i$) do
            $W(x) := \max\{W(x), \psi(A_i) + w_i\}$;
        $floor(x) := \max\{m(x), W(x)\}$;
        $go(x) := \min\{floor(x), M(x)\}$;
    $\delta(\psi) := (\tau^x, \chi)$, where $\tau^x := \min\{go(x) \mid x \in U^x(\psi)\}$
    and $\chi := \{x \in U^x(\psi) \mid \tau^x = go(x)\}$;
return $\delta(\psi)$;
```

```
Procedure RTEDExecutionStrategy($\mathcal{S}$)
Input: $\mathcal{S}$: STNU.
$\psi = \{(Z, 0)\}$; // initial partial schedule
while ($U(\psi) \neq \emptyset$) do
    $(\tau^x, \chi) = RTExecutionDecision(\mathcal{S}, \psi)$;
    if (nothing happens before time $\tau^x$) then
        Execute the timepoints in $\chi$;
    else
        Observe the contingent timepoints executed at
        some $\tau^c < \tau^x$;
    Update $\psi$ to include the executed events;
    Update $\mathcal{S}$ to include the corresponding constraints;
```

points have already been executed, $\delta(\psi) = (\infty, \emptyset)$ holds (i.e., RTED WAIT); otherwise, $\delta(\psi) = (\tau^x, \chi)$ with the values of τ^x and χ being computed by considering all unexecuted timepoints and using an all-pairs, shortest-path algorithm. $floor(x)$ corresponds to the earliest time, timepoint x may be executed without violating its lower bound $m(x)$ or any of its relevant waits. $go(x)$ is the same, except that it enforces the constraint that x does not violate its upper bound $M(x)$. It is noteworthy that Morris and Muscettola showed that a conflict between $floor(x)$ and $M(x)$ is not possible for an STNU accepted by their algorithm. After determining the RTED $\delta(\psi) = (\tau^x, \chi)$, procedure RTEDExecutionStrategy waits for the outcome of $\delta(\psi)$ and then updates ψ and $\mathcal{S}$ accordingly. If there are still unexecuted timepoints, the procedure iterates, otherwise it terminates.

3 GUARDED TEMPORAL CONSTRAINTS

Regarding an STNU, the execution of a contingent timepoint can be thought of as being completely out of the control of the system that executes the network. Typically, a system activates a contingent link (A, x, y, C) by executing its activation timepoint A. Afterwards, the execution of C is out of the system's control. However, the contingent timepoint C is guaranteed to execute such that $C - A \in [x, y]$ holds.

As motivated in Sect. 1, for real-world problems this is often too strict. In many cases, the system may exercise some control over the execution of the contingent timepoint. As example consider a case where, at an activation timepoint, the system transfers control to an external agent. The agent is then responsible for executing the corresponding contingent timepoint. In turn, the system waits for the agent to complete its task (i.e., to execute the contingent timepoint). When transferring the control to the agent, the system may inform the agent about the temporal constraints to be met. The agent then adapts its plan in order to comply with the additional constraints. At the same time, the system must guarantee that it is able to meet the commitment made, i.e., it needs to ensure that it can deal with any decision the agent makes for executing timepoint C based on the given constraints.

In many cases, the agent responsible for executing timepoint C cannot completely control the execution of C either (e.g., in case the agent is executing a network itself). Particularly, he might only be able to provide a preferred duration range $[x, y]$ as well as bounds x^{max} and y^{min} to which x may be increased or y may be decreased (i.e., $x \leq x^{max}$ and $y \geq y^{min}$). In turn, the system executing the network must ensure that, when executing timepoint A (i.e., when activating the constraint between A and C), the agent responsible for executing timepoint C has at least y^{min} time units and is not required to take more than x^{max} time units to execute C. We denote x^{max} (y^{min}) as the *guard* of x (y).

Note that this example addresses a common scenario, i.e., to transfer execution control at run time to another agent, which is responsible for executing a complex task (e.g., another network).

The need to model constraints of this type requires an extension of the STNU formalism, we denote as *Simple Temporal Network with Partially Shrinkable Uncertainty* (STNPSU). In particular, STNPSU extends contingent links of STNU to guarded links.

Definition 8 (STNPSU)**.** *A* Simple Temporal Network with Partially Shrinkable Uncertainty *(STNPSU) is a triple $(\mathcal{T}, \mathcal{C}, \mathcal{G})$, where:*

- *$\mathcal{T}$ is a set of* timepoints*;*
- *$\mathcal{C}$ is a set of* requirement constraints *$X \xrightarrow{[u,v]} Y$ (i.e., STN constraints); and*
- *$\mathcal{G}$ is a set of* guarded links *each having the form $(A, [x, x^{max}], [y^{min}, y], C)$ where A and C are timepoints, and $0 \leq x \leq y < \infty$, $x \leq x^{max}$, $0 \leq y^{min} \leq y$.*
- *If $(A_1, [x_1, {x_1}^{max}], [{y_1}^{min}, y_1], C_1)$ as well as $(A_2,$*

$[x_2, x_2{}^{max}], [y_2{}^{min}, y_2], C_2)$ *are distinct guarded links in $\mathcal{G}$, then C_1 and C_2 are distinct timepoints.*

Informally, we denote an STNPSU as *dynamically controllable* if it is possible to execute it such that, no matter how the execution of any guarded link turns out, for any other guarded link $(A, [x, x^{max}], [y^{min}, y], C)$ the lower bound x never must be increased beyond its guard x^{max} and the upper bound y never must be decreased below its guard y^{min} in order to ensure controllability of the network.

The execution semantics of STNPSU can be summarized as follows: The basic execution semantics is the same as for an STNU. However, when executing an STNPSU, the outer bounds $[x, y]$ of a guarded link $(A, [x, x^{max}], [y^{min}, y], C)$ may be restricted to $[x', y']$ with $x \le x' \le x^{max}$, $y^{min} \le y' \le y$, and $x' \le y'$ in order to ensure controllability of the remaining network. In turn, when executing the activation timepoint A of a guarded link $(A, [x, x^{max}], [y^{min}, y], C)$, the latter is activated and its current bounds $[x', y']$ are fixed. Particularly, the guarded link $(A, [x, x^{max}], [y^{min}, y], C)$ is replaced by the *strict guarded link* $(A, [x', x'], [y', y'], C)$. The latter is equivalent to a contingent link of STNU. As we will show in Sect. 3.3, this change does not affect controllability of the network.

It is noteworthy that guarded links of STNPSU may be used to represent two different types of constraints:

Type 1: If $x^{max} < y^{min}$ holds, a guarded link represents a *partially contingent constraint*. Particularly, the guarded link represents a temporal constraint $x \le C - A \le y$ with a *contingent* (i.e., unshrinkable) *core* $[x^{max}, y^{min}] \subseteq [x, y]$. This represents an extension of the classical contingent links of STNU. Moreover, if $x = x^{max} \wedge y = y^{min}$ hold, the guarded link is equivalent to a contingent link of STNU. We call this a *strict guarded link*.

Type 2: If $x^{max} \ge y^{min}$ holds, a guarded link represents a *partially shrinkable constraint* with a *guarded core* $[y^{min}, x^{max}]$. In detail, this represents a temporal constraint $x \le C - A \le y$ whose bounds cannot be shrunk beyond a certain point (i.e., x^{max} and y^{min}, respectively). As opposed to a contingent link, x may be restricted to be greater than y^{min} and y to be lower than x^{max}. This represents an extension of the classical requirement constraints.

As example of a Type 1 guarded link consider guarded link $(A, [10, 15], [20, 40], C)$, which represents the duration of activity Stretching (cf. Fig. 1 (b)). During execution, the outer bounds $[10, 40]$ of this guarded link may be shrunk in order to ensure controllability of the remaining network. In the given case, for example, they may be shrunk to $(A, [7, 15], [20, 23], C)$ or $(A, [5, 15], [20, 20], C)$. However, the outer bounds may at most be shrunk to the *contingent core* of the guarded link, i.e., the above guarded link may at most be shrunk to $(A, [15, 15], [20, 20], C)$.

In turn, an example of a Type 2 guarded link is given by $(A, [5, 20], [10, 25], C)$. In this case, the lower bound of the guarded link may at most be increased to 20 and the upper bound may at most be decreased to 10. Thus, $(A, [15, 20], [10, 20], C)$, $(A, [20, 20], [10, 23], C)$, and $(A, [5, 20], [10, 10], C)$ are possible values this guarded link may be shrunk to. Note that a Type 2 guarded link may also be shrunk to a single value, e.g., $(A, [15, 20], [10, 15], C)$. However, a Type 2 guarded link must always allow for at least one value within its *guarded core* $[y^{min}, x^{max}]$ (i.e., $[10, 20]$).

During execution, when activating a guarded link of Type 1 or 2 (i.e., when executing its activation timepoint), the current outer bounds of the guarded link are fixed. This is to ensure that the outer bounds of the guarded link cannot be modified while it is active. Therefore, the current outer bounds of the guarded link are set to be strict. For example, when executing timepoint A, the Type 2 guarded link $(A, [15, 20], [10, 20], C)$ is replaced by a strict guarded link $(A, [15, 15], [20, 20], C)$. The latter is equivalent to a contingent link $(A, 15, 20, C)$ of STNU and ensures that the agent responsible for executing timepoint C may now choose any time in range $[15, 20]$ to execute timepoint C.

3.1 Dynamic Controllability of STNPSU

This section presents preliminary definitions of basic concepts required for the definition of dynamic controllability of a STNPSU.

The set of *core situations* specifies the contingent core of all guarded links of Type 1 (partially contingent guarded links), while the set of *core settings* specifies the guarded core of all guarded links of Type 2 (partially shrinkable guarded links).

Definition 9 (Core Situations and Core Settings). *Suppose $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{G})$ is an STNPSU. Let $\mathcal{G}^c = \{g \in \mathcal{G} | g = (A, [x, x^{max}], [y^{min}, y], C) \wedge x^{max} < y^{min}\}$ be the set of guarded links for which the guard x^{max} of the lower bound is lower than the guard y^{min} of the upper bound (i.e., Type 1). Further, let $\mathcal{G}^r = \mathcal{G} \setminus \mathcal{G}^c$ be the set of guarded links for which $y^{min} \le x^{max}$ holds (i.e., Type 2).*

If $\mathcal{G}^c$ contains k guarded links, $(A_1, [x_1, x_1{}^{max}], [y_1{}^{min}, y_1], C_1), \ldots, (A_k, [x_k, x_k{}^{max}], [y_k{}^{min}, y_k], C_k)$, then $\Omega^c_{\mathcal{S}} = [x_1{}^{max}, y_1{}^{min}] \times \ldots \times [x_k{}^{max}, y_k{}^{min}]$ is called the space of core situations *for $\mathcal{S}$. Any $\omega^c = (d_1, \ldots d_k) \in \Omega^c_{\mathcal{S}}$ is called a* core situation.

Further, if $\mathcal{G}^r$ contains m guarded links, $(A_1, [x_1, x_1{}^{max}], [y_1{}^{min}, y_1], C_1), \ldots, (A_m, [x_m, x_m{}^{max}], [y_m{}^{min}, y_m], C_m)$, then $\Omega^r_{\mathcal{S}} = [y_1{}^{min}, x_1{}^{max}] \times \ldots \times$

$[{y_m}^{min}, {x_m}^{max}]$ *is called the* space of core settings *for* $\mathcal{S}$.

Given the space of core situations Ω^c and the space of core settings Ω^r of an STNPSU, a projection of the STNPSU onto an STNU can be obtained as follows: First, each guarded link in $\mathcal{G}^c$ is replaced by a contingent link for the range specified in Ω^c. Second, each guarded link in $\mathcal{G}^r$ is replaced by a requirement constraint for the range in Ω^r.

Definition 10 (Core STNU of an STNPSU)**.** *Let* $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{G})$ *be an STNPSU.*

Then: The projection *of* $\mathcal{S}$ *onto its space of core situations* Ω^c *and its space of core settings* Ω^r*—denoted as* $stnuPrj(\mathcal{S}, \Omega^c, \Omega^r)$*—corresponds to an STNU* $(\mathcal{T}, \mathcal{C}'', \mathcal{L}'')$ *with:*

$$\mathcal{C}'' = \mathcal{C} \cup \{({y_i}^{min} \leq C_i - A_i \leq {x_i}^{max}) \mid 1 \leq i \leq m, \Omega^r = [{y_1}^{min}, {x_1}^{max}] \times \ldots \times [{y_m}^{min}, {x_m}^{max}]\}$$

$$\mathcal{L}'' = \{(A_i, {x_i}^{max}, {y_i}^{min}, C_i) \mid 1 \leq i \leq k, \Omega^c = [{x_1}^{max}, {y_1}^{min}] \times \ldots \times [{x_k}^{max}, {y_k}^{min}]\}$$

We denote the respective STNU as the core STNU *of STNPSU* $\mathcal{S}$.

Finally, this leads us to the dynamic controllability of an STNPSU. We provide a formalization of the dynamic controllability of an STNPSU based on the dynamic controllability of an STNU. We choose this approach since the formalization of dynamic controllability of STNU is robust and verified in literature.

Theorem 1 (Dynamic Controllability of STNPSU)**.** *An STNPSU* $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{G})$ *is* dynamically controllable *(DC), if the core STNU that results from the STNU Projection* $stnuPrj(\mathcal{S}, \Omega^c, \Omega^r)$ *of the STNPSU is dynamically controllable.*

Proof. ⇒ It is a matter of definitions to show that, if the core STNU is DC (cf. Sect. 2.1), the corresponding STNPSU is DC as well: each schedule being a solution of the core STNU is also a solution of the STNPSU. Indeed, it is always possible to restrict the STNPSU to its core situations. Thus, for each core situation of the STNPSU, a dynamic execution strategy (DES), which is a viable DES for the STNU, is also a viable DES for the STNPSU regarding its core situations. Hence, if the core STNU is DC, the STNPSU will be DC as well.

⇐ If the core STNU is not DC (i.e., no viable DES exists), at least one core situation ω^c of the STNPSU exists for which no DES exists within the core settings. Hence, for core situation ω^c, one of the partially shrinkable guarded links must be restricted beyond its guards to find a DES which returns a solution. As this is not possible, the STNPSU is not DC either.

□

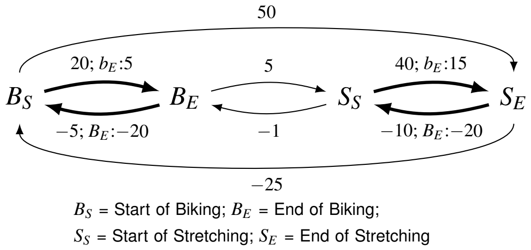

50
20; b_E:5
5
40; b_E:15
B_S
B_E
S_S
S_E
−5; B_E:−20
−1
−10; B_E:−20
−25
B_S = Start of Biking; B_E = End of Biking;
S_S = Start of Stretching; S_E = End of Stretching

Figure 2: STNPSU corresponding to the activity sequences from Fig. 1 (b)

3.2 DC-Checking for Guarded Constraints

This section shows how the dynamic controllability of an STNPSU may be checked without need to restrict the respective STNPSU to its core STNU. First, we emphasize the close relationship between dynamic controllability of STNU and the one of STNPSU (cf. Theorem 1). In turn, this fosters the following graph-based representation of an STNPSU, which is similar to the one of an STNU.

Definition 11 (Graph of a STNPSU)**.** *The graph for an STNPSU* $\mathcal{S}$ *has the form* $(\mathcal{T}, \mathcal{E}, \mathcal{E}_\ell, \mathcal{E}_u)$*, where each timepoint in* $\mathcal{T}$ *corresponds to 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*:*

- *Each requirement constraint* $X \xrightarrow{[u,v]} Y$ *is represented by two ordinary edges* $X \xrightarrow{v} Y$ *and* $Y \xrightarrow{-u} X$*.*
- *Each guarded link* $(A, [x, x^{max}], [y^{min}, y], C)$ *is represented by*
  - *two ordinary edges* $A \xrightarrow{y} C$ *and* $C \xrightarrow{-x} A$*,*
  - *one lower-case edge* $A \xrightarrow{c:x^{max}} C$*, and*
  - *one upper-case edge* $C \xrightarrow{C:-y^{min}} A$*.*

As example of this graph-based representation of an STNPSU, consider the graph depicted in Fig. 2. It shows the STNPSU corresponding to the activity sequence depicted in Fig. 1 (b). If multiple edges exist between two nodes (e.g., an ordinary and an upper-case edge), for the sake of readability, we draw only one arrow between the nodes and annotate it with the values of the respective edges. Further, we use bold arrows to highlight edges representing a guarded link.

At this point, we want to emphasize important differences between the graph of an STNU and the one of an STNPSU:

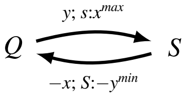


Figure 3: Guarded Link

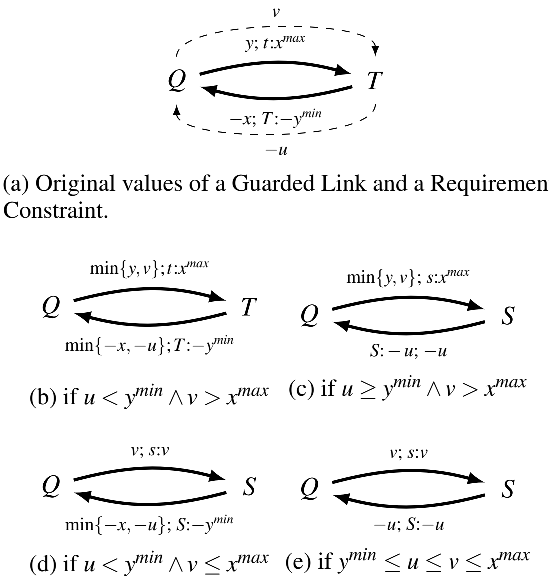
Table 2: Updated Label Removal Rule

Label Removal*:	$S \xrightarrow[v]{R:v} T$ Applicable if: $\mathbf{R} \not\equiv \mathbf{S}$ and $v \geq -x$, where x is the lower bound for the guarded link from T to R.

- In an STNU, the value of any lower-case edge $A \xrightarrow{c:x^{max}} C$ always corresponds to the negative value of the ordinary edge $C \xrightarrow{-x} A$ pointing in the opposite direction (i.e., $x = x^{max}$). Similarly, the value of any upper-case edge $C \xrightarrow{C:-y^{min}} A$ always corresponds to the negative value of ordinary edge $A \xrightarrow{y} C$ (i.e., $y = y^{min}$). For an STNPSU, this does not apply. Particularly, we only require $x \leq x^{max}$ and $y \geq y^{min}$.
- In an STNU, the value of a lower-case edge $A \xrightarrow{c:x^{max}} C$ always is lower than the negative value of the upper-case edge $C \xrightarrow{C:-y^{min}} A$ pointing in the opposite direction (i.e., $x^{max} < y^{min}$). Note that for an STNPSU this is not required.

We show that, except one minor change regarding one of the edge generation rules (cf. Table 1), procedure MM5-DC-Check may be reused in order to check dynamic controllability of a STNPSU. Particularly, we analyze all possible combinations of edges between three nodes of an STNPSU graph (i.e., all possible triangles). Based on this, it can be shown that the resulting distance graph of the STNPSU has no negative loops if and only if the distance graph of the core STNU has no negative loops as well. For the sake of brevity we do not provide a full analysis here, but focus on some of the more interesting cases instead.

First, consider the single guarded link depicted in Fig. 3. It comprises two triangles *S*-*Q*-*S* and *Q*-*S*-*Q*. Note that it is a matter of applying the edge-generation rules to these two triangles (i.e., the No Case rule to *S*-*Q*-*S* and the No Case, Upper Case, Lower Case, and Label Removal rules to *Q*-*S*-*Q*) to ascertain that a valid guarded link does not contain a negative loop. In case of a partially shrinkable guarded link (Type 2), in addition, the Label Removal (cf. Table 1) rule may be applied to the upper-case edge between S and Q, replacing it with a requirement edge. This poses no problem for checking dynamic controllability, but it is undesired as it obscures some of the properties of the guarded link. Thus, we restrict the Label Removal rule to $R \not\equiv S$ to prevent this (cf. Table 2). Note that this change does not influence the applicability of the rule to an STNU. Regarding an STNU, for $R \equiv S$ it holds that $v < -x$ (i.e., $x^{max} < y^{min}$; cf. Table 1), i.e., for an STNU, the rule will never be applied if $R \equiv S$ holds.



(a) Original values of a Guarded Link and a Requirement Constraint.

(b) if $u < y^{min} \land v > x^{max}$

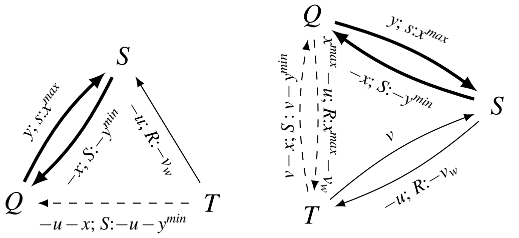
(c) if $u \geq y^{min} \land v > x^{max}$

(d) if $u < y^{min} \land v \leq x^{max}$

(e) if $y^{min} \leq u \leq v \leq x^{max}$

Figure 4: Combining a Guarded Link with a derived Requirement Constraint

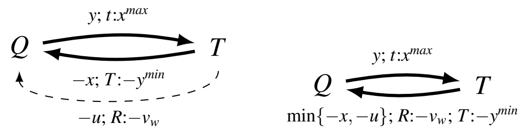
Consider the guarded link depicted in Fig. 4 (a). Due to some edge-generation rule, a pair of requirement constraints $Q \xrightarrow{v} T$ and $T \xrightarrow{-u} Q$ is added next to the guarded link. In this case only the most restrictive constraints need to be kept. The other ones are redundant and may be removed. If controllable, this scenario results in one of the four cases depicted in Fig. 4 (b)–(e). Note that if either $v < u$, $u > x^{max}$, or $v < y^{min}$ holds, the scenario will be not controllable (i.e., there exists a negative loop). If $u < y^{min}$ and $v > x^{max}$ holds, the outer bounds x and y of the guarded link may have to be restricted to u and v, respectively, as shown in Fig. 4 (b). Note that for partially contingent guarded links (i.e., Type 1), this is the only controllable case. In turn, if $u \geq y^{min}$ holds (cf. Fig. 4 (c) and (e)), guard y^{min} of the upper bound y is no longer required as it can be already ensured by the new lower bound u of the guarded link, i.e., the upper-case edge is redundant and may be removed. Likewise, if $v \leq x^{max}$ holds the lower-case edge (i.e., the guard x^{max} for the lower bound x) is redundant and may be removed. To preserve the difference between a guarded link and a requirement/wait constraint in the STNPSU graph, instead of removing the guards, we adapt their value according to the restriction given by the requirement edges (cf. Fig. 4 (c)–(e)). Particularly, instead of removing the lower-case or upper-case edge, we allow their values to be decreased (i.e., the negative value of the upper-case edge to be increased). Depending



(a) Guarded Link followed by Wait Constraint

(b) Guarded Link in parallel with Wait Constraint ($v_w \geq 0 \wedge u \geq 0$)

Figure 5: Combining a Guarded Link with a Wait Constraint



(a) Original values of Guarded Link and Wait Constraint.

(b) if $x < v_w \leq x^{max} \wedge u < y^{min}$

Figure 6: Combining a Guarded Link with a derived Wait Constraint

on the type of guarded link (i.e., Type 1 or Type 2), it is a matter of replacing x by x^{max} and y by y^{min} (or x by y^{min} and y by x^{max}, respectively) to see that this is similar to the edges generated when checking dynamic controllability of the core STNU.

Fig. 5 (a) shows how a guarded link from Q to S may be combined with a wait constraint from T to S. The dashed arrow denotes the edge generated by applying the No Case and Upper Case rules. Again it can be easily shown that the generated edge is similar to the one that would be generated for the core STNU. Note that we assume $x^{max} \leq y$, which is always true when applying the previously described restriction of the guards (cf. Fig. 4 (a)–(e)).

Fig. 5 (b) depicts a similar case where a guarded link is placed in parallel with a wait constraint (and a requirement constraint). In this case two different wait constraints must be added between the activation timepoint Q of the guarded link and timepoint T. Note that the latter corresponds to the activation timepoint of the guarded link between T and R.

One of the most interesting cases is depicted in Fig. 6 (a): Due to an edge generation rule, a wait constraint $T \overset{-u;R:-v_w}{\longrightarrow} Q$ is added next to the guarded link. For the case $x < v_w \leq x^{min} \wedge u < y^{min}$ the result is depicted in Fig. 6 (b). Consider the wait constraint added next to the upper-case edge. Note that this is not possible for an STNU. Regarding an STNU, either $v_w > x^{max} = x$ (i.e., the STNU is not controllable) or $v_w \leq x$ (i.e., the wait constraint is redundant) holds. For an STNPSU, Fig. 6 (b) can be interpreted as follows: For Type 1 constraints (i.e., $v_w < x^{max} < y^{min}$) R:$-v_w$ is not relevant as T:$-y^{min}$ is more restrictive, i.e., we are only interested in dynamic controllability of the core STNU. Otherwise, R:$-v_w$ possibly represents a standard wait constraint for the core settings in $[y^{min}, x^{max}]$. In turn, when executing the STNPSU the wait constraint must be observed. Particularly, if the guarded link is activated prior to the execution of timepoint R, v_w must be used as lower bound when executing the guarded link. Otherwise, $\min\{-x,-u\}$ is used as lower bound.

3.3 Executing STNPSUs

This section shows how an STNPSU may be executed by means of an appropriate extension of procedure RTEDExecutionStrategy (cf. Sect. 2.2).

Consider procedure ExRTEDExecutionStrategy. The first part of the procedure executes the same actions as procedure RTEDExecutionStrategy (cf. Sect. 2.2). The second part activates all guarded links $(A_i, [x, x^{max}], [y^{min}, y], C_i)$ whose activation timepoint A_i has just been executed. The guarded link semantics requires to allow each of them, once it is activated, to use any possible value in the range defined by the current outer bounds, i.e., $[x, y]$. By construction and due to the fact that the network is DC, for a Type 1 guarded link the possibility of using any possible value in the range is guaranteed only for the core range $[x^{max}, y^{min}]$, while for a Type 2 guarded link only the possibility of using at least one value in the range $[y^{min}, x^{max}]$ is guaranteed. Particularly, the execution of some other timepoints before the occurrence of C_i may modify the bounds of these guarded links making the network not controllable. Therefore, the procedure has to suitably update the bounds of the guarded links (lines 14–20) before transforming them into strict ones (lines 21–27). Finally, the execution goes back to the first part until there are no more unexecuted timepoints.

The key point of the procedure consists in the execution of timepoints subjected to guarded links as contingent timepoints with suitable ranges; this allows for the exploitation of the correctness proof of RTEDExecutionStrategy (Hunsberger, 2009). In order to show that this transformation preserves the controllability of the network, it is sufficient to show that the transformation of any guarded link—during runtime—into a strict one with a suitable range is always possible and preserves the dynamic controllability of the rest of network (i.e., the unexecuted subnetwork).

Theorem 2. *Suppose $\mathcal{S}$ is a dynamically controllable STNPSU, ψ is a respectful, partial schedule, and $(A, [x, x^{max}], [y^{min}, y], C)$ is a guarded link of $\mathcal{S}$. Let us assume that A has just been executed and that the*

```
Procedure ExRTEDExecutionStrategy($\mathcal{S}$)
Input: $\mathcal{S}$: STNPSU.
1  $\psi = \{(Z,0)\}$; // initial partial schedule
2  while $(U(\psi) \neq \emptyset)$ do
3      $(\tau^x, \chi) = RTExecutionDecision(\mathcal{S}, \psi)$;
4      if (nothing happens before time $\tau^x$) then
5          execute the time-points in $\chi$;
6      else
7          observe the contingent timepoints executed at some
           $\tau^c < \tau^x$;
8          $\chi$ = set of executed contingent timepoints;
9      Update $\psi$ to include the execution events in $\chi$;
10     Update $\mathcal{S}$ to include the corresponding constraints;
11     foreach $(A_i \in \chi)$ do                    // Activate guarded links
12         foreach $((A_i, [x, x^{max}], [y^{min}, y], C_i) \in \mathcal{G})$ do
13             $[x', y']$ = current outer bounds of guarded $A_iC_i$;
14             repeat         // Prepare the guarded link for execution
                   // Determine its maximum controllable range
15                 $range(A_i, C_i) = \min\{v - u \mid a \in U(\psi)! \wedge$
                   $a \xrightarrow{v} C_i \wedge v \geq 0 \wedge (C_i \xrightarrow{-u} a \vee C_i \xrightarrow{C_j:-u} a)\}$;
                   // Update its bounds to observe max. controll. range
16                 $y' = \min\{y', \max\{y^{min}, x' + range(A_i, C_i)\}\}$;
                   // $x'$ is update only if the update $y'$ is not sufficient
17                 $x' = \max\{x', y' - range(A_i, C_i)\}$;
18                 if $x'$ or $y'$ is modified then
19                     Update $\mathcal{S}$ to include the modified
                       outer bounds of the guarded link;
20             until neither $x'$ nor $y'$ is modified;
               // Consider the max possible wait constraint for $C_i$
21             $W(A_i, C_i) := -\infty$;
22             foreach$((A_i, C_j) \mid C_j \in U^a(\psi) \wedge C_i \xrightarrow{C_j:-w_j} A_i)$do
23                 $W(A_i, C_i) := \max\{W(A_i, C_i), w_j\}$;
24             $floor(A_i, C_i) := \max\{x', W(A_i, C_i)\}$;
25             $x' := \min\{floor(A_i, C_i), y'\}$;
26             Transform the guarded link to the strict
               guarded link $(A_i, [x', x'], [y', y'], C_i)$;
27             Update $\mathcal{S}$ to include the new constraint;
```

outer bounds x and y of the guarded link AC have been updated as described in lines 13–25 of procedure RTEDExecutionStrategy *to the values x' and y'. Then: The new values x' and y' satisfy $x' \leq x^{max}$ and $y^{min} \leq y'$ and the STNPSU $\mathcal{S}'$ resulting after the transformation of the guarded link $(A, [x', x^{max}], [y^{min}, y'], C)$ into the strict one $(A, [x', x'], [y', y'], C)$ is dynamically controllable as well.*

Sketch of the proof. Let us assume that, before the execution of lines 13–25, the guarded link AC is given as $(A, [x, x^{max}], [y^{min}, y], C)$. Instructions of lines 13–25 update the outer bounds $[x, y]$ to $[x', y']$. Let us assume that the guarded link is of Type 1.[1] Since the network is DC before line 13 and the core range $[x^{max}, y^{min}]$ is a contingent range, the update made in lines 13–25 cannot reduce $[x, y]$ to $[x', y']$ such that $x' > x^{max}$ or $y' < y^{min}$ holds. Hence, the updated range $[x', y']$ contains (possibly in a weak way) the core range of $(A, [x, x^{max}], [y^{min}, y], C)$. Thus, there are 4 possible cases: (1) $x' = x^{max} \wedge y' = y^{min}$, (2) $x' = x^{max} \wedge y' > y^{min}$, (3) $x' < x^{max} \wedge y' = y^{min}$, and (4) $x' < x^{max} \wedge y' > y^{min}$. In case (1) the guarded link is already strict and, hence, it is not changed by the procedure. Case (4) is a combination of cases (2) and (3).

Hence, let us consider case (2) and (3). In case (2), by contradiction, suppose that the changing of the guarded link $(A, [x^{max}, x^{max}], [y^{min}, y'], C)$ into the strict one $(A, [x^{max}, x^{max}], [y', y'], C)$ makes the remaining network not dynamically controllable. This means that there exists at least one negative loop involving timepoints A, C, and some B, where B has not yet executed (i.e., it has to occur after A), but has to be executed before C. All other timepoints, i.e., unexecuted ones that have to be executed after C, cannot contribute to form a negative loop since each of them—by definition of controllability—must have at least one possible execution time for each possible execution time of C in the range $[\psi(A) + x^{max}, \psi(A) + y']$.
Now, instead of considering the distance graph and negative loops in it, let us reason in term of ranges and their spans[2]. Given the dynamic controllability of the network before the transformation, it is a fact that B has at least one possible execution time for each possible execution time of C in the range $[\psi(A) + x^{max}, \psi(A) + y^{min}]$, i.e., with such range there is no negative loop involving A, B, and C. Equivalently, the span of the constraint between B and C is greater or equal to the span $y^{min} - x^{max} + 1$ of the contingent core of the guarded link between A and C. Therefore, a negative loop can only emerge when the new bound y' is considered, or, equivalent, when the span of the constraint between B and C is less than the span $y' - x^{max} + 1$ of the outer bounds of the guarded link between A and C.

However, when preparing the guarded link for execution, y' is determined such that the span of range $[x^{max}, y']$ of the link between A and C is lower than or equal to the span of the constraint between B and C. Thus, there cannot exist any negative loop involving A, B, and C.

Case (3) can be shown in a similar way taking also into account any possible wait constraint that may increase the value of x'. □

[1] The case of a guarded link of Type 2 can be discussed in a similar way.

[2] The span of a range $[a, b]$ is $b - a + 1$.

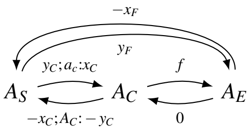


Figure 7: STNU pattern representing a shrinkable duration range $[x_F,[x_C+f,y_C],y_F]$ $(0 \leq x_C \leq x_F \leq x_C+f \wedge x_C \leq y_C \leq y_F \leq y_C+f)$.

4 ON THE EXPRESSIVENESS OF GUARDED CONSTRAINTS

This section informally discusses the expressiveness of STNPSUs. Further, we show that most guarded links cannot be represented in STNUs by other solutions.

Consider again the physiotherapy session scenario from Fig. 1 (b): Activity Stretching has a duration range of $[10,40]$, which may be shrunk to a core duration range of $[15,20]$ during run time according to the actual duration of activity Biking. Fig. 2 depicts the temporal aspects of the session in terms of an STNPSU: each of the two activities is represented through a pair of timepoints, of which one represents the starting instant of the activity and the other one the ending instant. The allowed duration of activity Stretching is represented as guarded link $(S_S,[10,15],[20,40],S_E)$ while the contingent duration of activity Biking is represented as strict guarded link $(B_S,[5,5],[20,20],B_E)$. Based on the results from Sect. 3 one can easily verify that the STNPSU is *dynamically controllable*, i.e., for each possible execution time of activity Biking, the system is able to determine a suitable duration range for activity Stretching containing the core range $[15,20]$ such that each possible execution time in this range satisfies the overall duration constraint $[25,50]$.

Let us discuss some of the limitations that arise when representing the temporal aspects of the physiotherapy session in terms of an STNU. The main problem is how to represent the temporal constraints of activity Stretching. One option to be considered is the pattern depicted in Fig. 7. It constitutes a generalization of the pattern proposed by (Lanz et al., 2013) (cf. Sect. 2). This pattern is composed of three timepoints A_S, A_C, and A_E connected by a contingent link and two requirement constraints. More precisely, timepoints A_S and A_E represent the starting and ending timepoint of the respective activity. In turn, A_C is an internal timepoint that is only used for checking dynamic controllability of the STNU, but is not considered when executing the activity. The values of the three constraints guarantee that the overall duration range of the pattern lies in range $[x_F,y_F]$ and the upper bound y_F can be shrunk to y_C at run time. Moreover, the lower bound x_F may be shrunk to x_C+f as well. Hence, the overall constraint represented by this pattern is similar to guarded link $(A_S,[x_F,x_C+f],[y_C,y_F],A_E)$ $(0 \leq x_C \leq x_F \leq x_C+f \wedge x_C \leq y_C \leq y_F \leq y_C+f)$. This pattern can be used to represent some settings of both types of guarded links. However, for example, it can not be used to represent the duration of activity Stretching (cf. Fig. 1 (b)).

Particularly note that, the pattern contains a strict dependency between the value of the guard for the lower bound, x_C+f, and the distance between the guard for the upper bound y_C and the upper bound y_F itself. In detail, for the contingent constraint between A_S and A_C (cf. Fig. 7) it holds $0 \leq x_C$. Thus, for the guard of the lower bound $x_C+f \geq f$ holds as well. Moreover, $y_F \leq y_C+f$ holds and thus $y_F-y_C \leq f$. As a result, $y_F-y_C \leq f \leq x_C+f$ must hold, i.e., the distance between the upper bound y_F and the guard for the upper bound y_C must be lower or equal to the value of the guard for the lower bound x_C+f. Note that, it is not possible to extend the pattern to cover arbitrary guarded constraints as it is not possible to resolve this dependency between the constraints comprising the pattern. Thus, STNPSU is more expressive than STNU.

5 CONCLUSION

The main contribution of this paper is to present an extension of STNU that allows for the definition and efficient management of a novel kind of constraints, i.e., *guarded links*. A guarded link represents an admissible range of delays between two timepoints, where each bound of the range can be shrunk during run time, but not beyond a given threshold. A guarded link constitutes a generalization of the two kinds of STNU constraints, i.e., requirement and contingent constraints, in the sense that a contingent link may be represented as a simple form of a guarded one and that a guarded link may represent a requirement constraint. An STNU where it is possible to define guarded links is denoted as Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU). In particular, the dynamic-controllability check and the execution of a STNPSU can be done in polynomial time.

Networks such as STNU can be used as temporal foundation for a broad class of Process-Aware Information Systems currently being developed (Reichert and Weber, 2012). In this context, the extension proposed in this paper may be used to better represent the temporal properties of sub processes (i.e., complex tasks). It is quite common to have sub processes whose allowed durations can be restricted in a limited way prior to their execution. In turn, once a sub process starts to execute, it is necessary to guarantee that the allowed duration range can be used without any further interference.

There are different avenues for future work. First, we want to study the applicability of our approach to CSTNU, for which the presence of labeled constraints and links requires to consider further and different propagation rules that have to be extended to take account of guarded constraints. Second, the application of STNPSU as temporal foundation of Process-Aware Information Systems could be interesting. Particularly, STNPSU might serve as a tool for an appropriate and scalable analysis of the temporal properties of the business processes.

REFERENCES

Combi, C., Hunsberger, L., and Posenato, R. (2013). An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In *Proc Intl Conf Agents & Art. Int. (ICAART'13)*, volume 2, pages 144–156. SciTePress.

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 *Intl Symp. on Temporal Repres. and Reasoning (TIME'09)*, pages 155–162. IEEE CPS.

Hunsberger, L., Posenato, R., and Combi, C. (2012). The Dynamic Controllability of Conditional STNs with Uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) @ ICAPS-2012*, pages 1–8, Atibaia.

Lanz, A., Posenato, R., Combi, C., and Reichert, M. (2013). Controllability of time-aware processes at run time. In *On the Move to Meaningful Internet Systems: Proc. CoopsIS'13*, pages 39–56. Springer.

Moffitt, M. D. and Pollack, M. E. (2007). Generalizing temporal controllability. In *Intl Joint Conf on Artificial Intelligence (IJCAI'07)*, pages 1985–1990. Morgan Kaufmann.

Morris, P. (2006). A structural characterization of temporal dynamic controllability. In Benhamou, F., editor, *Intl Conf on Principles and Practices of Constraint Programming (CP'06)*, pages 375–389. Springer.

Morris, P. (2014). Dynamic controllability and dispatchability relationships. In Simonis, H., editor, *Intl Conf on Integration of AI and OR Techniques in Constraint Programming (CPAIOR'14)*, volume 8451 of *LNCS*, pages 464–479. Springer.

Morris, P. H. and Muscettola, N. (2005). Temporal dynamic controllability revisited. In Veloso, M. M. and Kambhampati, S., editors, *National Conf 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 *Intl Joint Conf on Artificial Intelligence (IJCAI'01)*, pages 494–502. Morgan Kaufmann.

Reichert, M. and Weber, B. (2012). *Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies*. Springer.

Tsamardinos, I., Vidal, T., and Pollack, M. E. (2003). CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8:365–388.