Dynamic Controllability of Temporal Plans in Uncertain and Partially Observable Environments

Arthur Bit-Monnot, Paul Morris

Dynamic Controllability of Temporal Plans in Uncertain and Partially Observable Environments

Arthur Bit-Monnot ABITMONNOT@LAAS.FR
LAAS-CNRS, Université de Toulouse, CNRS, INSA,
Toulouse, France

Paul Morris
Formerly at NASA Ames Research Center,
Moffett Field, CA 94035 USA

Abstract

The formalism of Simple Temporal Networks (STNs) provides methods for evaluating the feasibility of temporal plans. The basic formalism deals with the consistency of quantitative temporal requirements on scheduled events. This implicitly assumes a single agent has full control over the timing of events. The extension of Simple Temporal Networks with Uncertainty (STNU) introduces uncertainty into the timing of some events. Two main approaches to the feasibility of STNUs involve (1) where a single schedule works irrespective of the duration outcomes, called Strong Controllability, and (2) whether a strategy exists to schedule future events based on the outcomes of past events, called Dynamic Controllability. Case (1) essentially assumes the timing of uncertain events cannot be observed by the agent while case (2) assumes full observability.

The formalism of Partially Observable Simple Temporal Networks with Uncertainty (POSTNU) provides an intermediate stance between these two extremes, where a known subset of the uncertain events can be observed when they occur. A sound and complete polynomial algorithm to determining the Dynamic Controllability of POSTNUs has not previously been known; we present one in this paper. This answers an open problem that has been posed in the literature.

The approach we take factors the problem into Strong Controllability micro-problems in an overall Dynamic Controllability macro-problem framework. It generalizes the notion of labeled distance graph from STNUs. The generalized labels are expressed as max/min expressions involving the observables. The paper introduces sound generalized reduction rules that act on the generalized labels. These incorporate tightenings based on observability that preserve dynamic viable strategies. It is shown that if the generalized reduction rules reach quiescence without exposing an inconsistency, then the POSTNU is Dynamically Controllable (DC). The paper also presents algorithms that apply the reduction rules in an organized way and reach quiescence in a polynomial number of steps if the POSTNU is Dynamically Controllable.

Remarkably, the generalized perspective leads to a simpler and more uniform framework that applies also to the STNU special case. It helps illuminate the previous methods inasmuch as the max/min label representation is more semantically clear than the ad-hoc upper/lower case labels previously used.

1. Introduction

Many applications (for example, the Remote Agent Experiment (Muscettola, Nayak, et al., 1998), as an early one) have drawn attention to the importance of quantitative reasoning

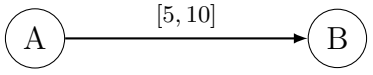
about time in practical planning systems. In particular, a need has been felt for temporal representations to specify scheduling requirements that an agent needs to satisfy. In general, these requirements could involve exogenous events, as well as the agent's own actions, and these events might or might not be observable. A number of formalisms have been established in response to the need to model these kinds of problems, and algorithms have been developed to solve them. The formalism of Simple Temporal Networks and its extensions has been particularly useful in this regard. Nevertheless, there remain many unsolved problems and under-developed theories in this area. This paper provides a greater understanding and solution for one such class of problems.

1.1 STNs and Extensions

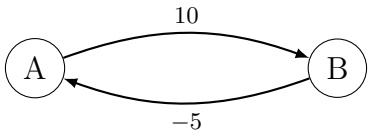
1.1.1 STN

A Simple Temporal Network (STN) (Dechter et al., 1991) is a graph in which the edges are annotated with upper and lower numerical bounds. The nodes in the graph represent temporal events or *timepoints*, while the edges correspond to constraints on the durations between the events. A link A $\xrightarrow{[l,u]}$ B in the STN specifies that timepoint B must be executed at least l and at most u time units after the execution of timepoint A. An STN is *consistent* if there exists an time assignment to all timepoints that satisfies all the constraints.

Each STN is associated with a *distance graph* derived from the upper and lower bound constraints. In the distance graph, an edge A $\xrightarrow{d}$ B specifies that delay from A to B must not be more than d ($B - A \leq d$). The link A $\xrightarrow{[l,u]}$ B of an STN is thus expressed as two edges A $\xrightarrow{u}$ B and B $\xrightarrow{-l}$ A in the equivalent distance graph. An STN is consistent if and only if the distance graph does not contain a negative cycle. To avoid confusion with the distance graph, we will refer to edges in the STN as *links* while the term *edges* will be reserved for edges in the distance graph.



(a) An STN with two timepoints A and B.



(b) Distance graph of the STN

Figure 1: An STN and its distance graph. The constraints specify that the timepoint B must be executed at least 5 time units and at most 10 time units after the timepoint A.

1.1.2 STNU

A Simple Temporal Network With Uncertainty (STNU), introduced by Vidal and Fargier (Vidal, 2000; Vidal & Fargier, 1999), is similar to an STN except the links are divided into two classes, *requirement links* and *contingent links*. Requirement links are temporal constraints that the agent must satisfy, like the links in an ordinary STN. Contingent links may be thought of as representing causal processes of uncertain duration, or periods from a reference time to exogenous events. Their finish timepoints, called here *contingent timepoints*, are controlled by Nature, subject to the limits imposed by the

bounds on the contingent links. We will refer to the start timepoint of a contingent link as its *activation* timepoint. This may itself be a contingent timepoint if it is the finish point of some other contingent link. All other timepoints, called *executable timepoints*, are controlled by the agent, whose goal is to satisfy the bounds on the requirement links. Each contingent link is required to have finite positive upper and lower bounds. An STNU may be thought of as determining a family of STNs where the contingent links take on each of their possible durations; the individual STNs in the family are called *projections*. An STNU is said to be *Weakly Controllable* if every projection is consistent. Weak Controllability was shown to be in co-NP, and later proved to be co-NP Hard (Morris & Muscettola, 1999). However, this property does not support a generally useful execution strategy.

The uncontrollable timepoints in STNUs are generally assumed to be either all unobservable, or all observable when they occur, giving rise to different execution strategies. An STNU is *Strongly Controllable* if there is a single schedule that satisfies the requirements in *all* of the projections, and thus does not depend on observations. An STNU is said to be *Dynamically Controllable* (Hunsberger, 2009; Morris et al., 2001; Morris, 2014; Vidal & Fargier, 1999) if there is a strategy for scheduling each executable timepoint that depends only on observations that are available in the past *or present* at the time it is scheduled.[1] Whether an STNU is Dynamically Controllable or not can be determined by algorithms that run in cubic time (Cairo et al., 2018; Cairo & Rizzi, 2017; Morris, 2014; Nilsson et al., 2015).

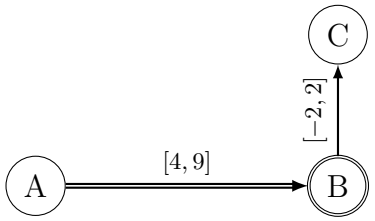
As mentioned, an STN has an alternative representation as a *distance graph* (Dechter et al., 1991). Similarly, there is a representation for an STNU called the *labeled distance graph* (Morris & Muscettola, 2005). Similar to an STN, an STNU is *dynamically controllable* if and only if its labeled distance graph does not contain a negative cycle of a particular form, called a *semi-reducible negative cycle* (Morris, 2014).

1.1.3 POSTNU

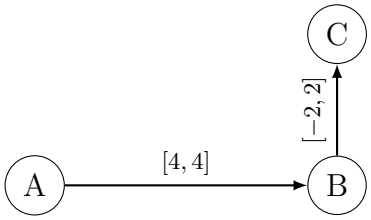
A Partially Observable STNU (POSTNU) (Moffitt, 2007) is an STNU in which the contingent timepoints are further subdivided into observable and unobservable (hidden) timepoints. Thus, the controllability problem for a POSTNU may be regarded as a combination of Strong and Dynamic Controllability. In the general POSTNU problem, a contingent link may be activated by a hidden timepoint. In that case, if the endpoint is observable, the POSTNU semantics specifies that when it is observed, we learn only the *time* of the endpoint, not the *duration* of the link that was activated by the hidden timepoint. Of course we do learn (or can easily calculate) the time difference between the observed endpoint and any previous known time.

The algorithm of Moffitt (2007) for checking the controllability of a POSTNU is complete but not sound in that it might incorrectly label a POSTNU as controllable. Another algorithm, also relying on the compilation to STNUs, is provided by Bit-Monnot et al. (2016) that is sound but only complete for a subclass of POSTNUs. Variable Delay prob-

1. The literature varies on whether observations can be reacted to instantaneously, or must be strictly in the past. The instantaneous variant was the original concept and has some technical advantages, including allowing an executable timepoint to be simultaneous with an observation.



(a) An STNU with a contingent timepoint B and two controllable timepoints A and C.



(b) A projection of the STNU into an STN where the AB contingent link takes its minimal duration 4.

Figure 2: An STNU and derived constructs. The STNU contains a contingent link AB specifying that the contingent timepoint B will occur sometime between 4 and 9 time units after the activation timepoint A. Furthermore, the BC requirement link requires that the controllable time point C be executed at most 2 time units before or after the occurrence of B. This STNU is dynamically controllable: the strategy of executing C immediately after observing B will satisfy all requirements. However it is not strongly controllable: if B is not observed, then there is no execution strategy that satisfies all constraints.

lems (Bhargava et al., 2018) may also be viewed as a restricted subclass of POSTNUs whose Dynamic Controllability problem can be reduced to STNU checking.

Another paper (Bhargava & Williams, 2019) considers a disjunctive class (PODTNUs) of partially observable networks, and shows this more general problem is PSPACE-complete. Their PSPACE algorithm could also be applied to POSTNUs as a special case, but it enumerates strategies and does not appear to be polynomial in time. Indeed, the authors acknowledge that known polynomial algorithms for POSTNUs and Multiagent STNUs (MaSTNUs) are not complete, and point out the importance of fully addressing the complexity of these problems.

Thus, to the best of our knowledge, for the general POSTNU problem, a polynomial sound and complete algorithm for assessing Dynamic Controllability, has not previously been known. We present one here, and in the process introduce new concepts that contribute to a greater understanding of these problems.

In the following sections, we first present formal definitions and basic concepts, including a distinction between *micro-projections* and *macro-projections*. Then we outline our general approach to the POSTNU problem, which factors it into separate micro and macro subproblems. For the former, the paper discusses the transfer of requirements from non-observable timepoints to equivalent constraints on new timepoints called *compound observables*. For the macro subproblem, the paper introduces *generalized labels* and associated *generalized reduction rules*. This is followed by a soundness and completeness proof with respect to these rules. A conceptually simple DC-checking procedure is then shown to have polynomial complexity and result in a dispatchable network. Finally, the paper introduces, as a more efficient procedure, an adaptation of the cubic algorithm of Morris (2014) to the POSTNU case and concludes with some closing remarks.

1.1.4 Other STN Extensions

Over the years, several other extensions to STNs and STNUs have been studied. For instance, Conrad and Williams (2011) and Zavatteri et al. (2019) respectively study STN with choices and with decisions, where the executor might have the choice between several set of requirement constraints. In the context of Conditional STNs, Hunsberger and Posenato (2020) instead focus on determining the dynamic controllability of networks where requirement constraints are activated as a result of contingent *observations* made at runtime. Combi et al. (2014) consider the introduction of temporal uncertainty in Conditional STNs, which is further extended with runtime decision points by Zavatteri and Viganò (2019). Probabilistic STN consider a different form of uncertainty where the duration of contingent links is not bounded but associated with a probability distribution leading to the study of adaptations of dynamic controllability (Gao et al., 2020) and strong controllability (Frank, 2019).

The number of extensions to the original STN framework highlight the spectrum of practical needs for the execution of temporal plans, the partial observability setting that is under study in this paper being only one extension. However it is important to note that partial observability is orthogonal to many other extensions. While beyond the scope of this paper, interesting future work could be done regarding the introduction of partial observability in, e.g., conditional STNUs or STNUs with decisions.

1.2 Motivations and Overview

Space-related problems are a potential application area for POSTNU algorithms of interest to the authors. The duration of uncontrolled or natural processes are a source of temporal uncertainty. Observations are accomplished by imperfect sensors. For example, a spacecraft process of uncertain duration may be triggered by a temperature rise, which occurs at an uncertain elapsed time from a reference point. A sensor measuring the temperature may have a latency of uncertain duration; it can be modeled by a process also triggered by the temperature rise.

However, for the general reader, we provide here a more everyday example where "Nature" is simulated by a second agent. It should be noted that we do not claim to address the general two-agent problem, which may not satisfy the restrictions of a POSTNU. The example we present takes care to satisfy these restrictions.

Consider the situation where my friend and I plan on having lunch in a restaurant and want to organize ourselves so that none of us waits more than 5 minutes. The restaurant is close to me (7 to 10 minutes drive, depending on traffic). However my friend wants to drop a package at the post office on his way, taking him 15 to 30 minutes depending on the queue and will need another 15 to 20 minutes from the post office to the restaurant. Thus, if we agree that my friend leaves his house at 11:30am, I can expect him to arrive at the restaurant between 12pm and 12:20pm. In this situation, the best I can do is to attempt to arrive at 12:10pm at the restaurant which might result in one of us waiting 10 minutes.

This unsatisfactory situation is caused by my incapacity to observe the departure of my friend from the post office. To avoid this difficulty, he could notify me of his departure with a text message (e.g. "I just left the post office." or "I'm on my way to the restaurant."). When receiving this message, I will know with little uncertainty his departure time. From it

I can infer a small time window for his arrival at the restaurant and plan my own departure accordingly.

This article is concerned with formalizing and proposing a sound and complete algorithm for such a problem. The problem can be formulated as the POSTNU of Figure 3. The controllable timepoints are the departure time of my friend (FD, controllable because we can agree on it) and myself (MD). My friend's departure from the post office is modeled as a *hidden* contingent timepoint (FP), which is necessary as it is not an event that I can observe. Three observable contingent timepoints model the message's availability on my phone (MN) and the arrivals at the restaurant of my friend (FA) and myself (MA).

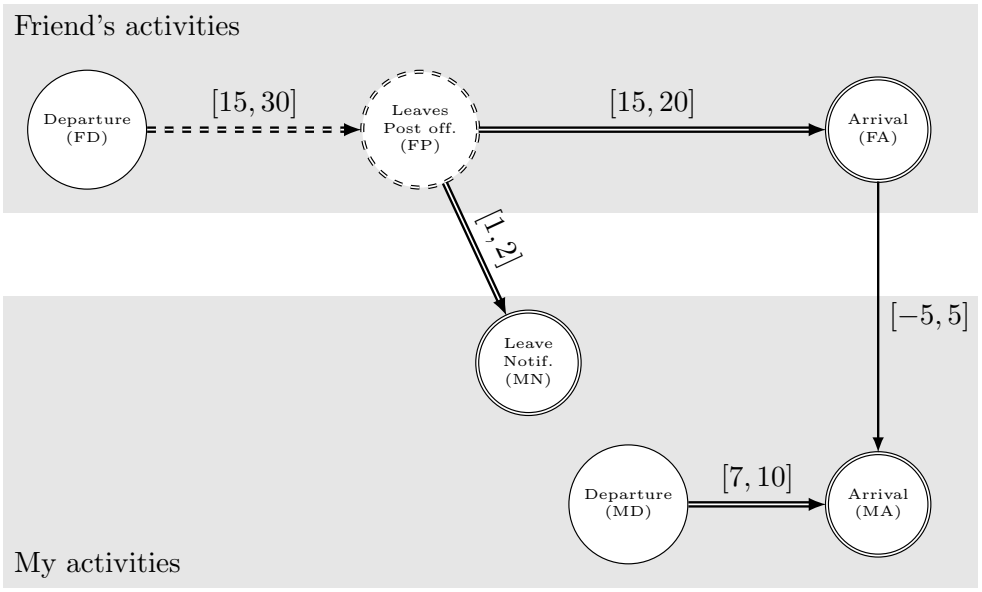


Figure 3: A POSTNU representing the post office example.

Solving this problem requires finding an execution strategy: a scheme to decide when to execute controllable events. The strategy can be dynamic: I can decide when to leave the house based on the occurrence time of past observable events. Thus my strategy might be to depart 7 minutes after the (observable) *Leave Notification* event but cannot refer to the (unobservable) *Leaves Post Office* event nor to a future *Arrival* event.

While humans routinely deal with such situations, no existing approach supports this albeit simple example. Indeed, the algorithm of Moffitt (2007) is unsound while the algorithms of Bit-Monnot et al. (2016) and Bhargava et al. (2018) are only complete for a subset of POSTNUs which in particular exclude this example. We propose one such algorithm that generalizes the key concepts required to reason on the dynamic controllability of STNUs to the partially observable setting.

2. Formal Preliminaries

Formally, an STN may be described as a 4-tuple $\langle N, E, l, u\rangle$ where N is a set of nodes called *timepoints*, E is a set of edges called *links*, and l and u are functions mapping each edge into the lower and upper bounds of the interval of possible durations.

STNU An STNU is a 5-tuple $\langle N, E, l, u, C\rangle$, where N, E, l, u are as in a STN, and C is a subset of the links: the *contingent* links, the others being called *requirement* links. Each contingent link e is required to satisfy $0 < l(e) \leq u(e) < \infty$.[2]

The durations of contingent links are assumed to vary independently; thus every combination that satisfies the upper and lower bounds of the contingent links gives rise to an STN called a *projection*. Contingent links are not allowed to share finish points. However, their finish points may be the start timepoints of other contingent links (thus potentially forming a branching tree structure). The finish timepoints of contingent links are called *contingent timepoints*. A contingent timepoint that does *not* start a new contingent link is called a *terminal* contingent timepoint.

A *schedule* is an assignment of times to all the timepoints. The *pre-history* of a specific time t with respect to a schedule T, denoted by $T\{\preceq t\}$, specifies the durations of all contingent links that have finished up to and including time t.

An *execution strategy* S is a mapping

$$S : \mathcal{P} \rightarrow \mathcal{T}$$

where $\mathcal{P}$ is the set of projections and $\mathcal{T}$ is the set of schedules. An execution strategy S is *viable* if $S(p)$, henceforth written S_p, is consistent with p for each projection p. An execution strategy S is *dynamic* if for projections p_1 and p_2, and executable timepoint x where $S_{p_1}(x) = t$, the strategy satisfies [3]

$$S_{p_1}\{\preceq t\} = S_{p_2}\{\preceq t\} \Rightarrow S_{p_1}(x) = S_{p_2}(x)$$

An STNU is *Dynamically Controllable* if there is a viable dynamic execution strategy.

POSTNU A POSTNU is a 6-tuple $\langle N, E, l, u, C, O\rangle$ where N, E, l, u, C are as in a STNU. Here O is a subset of the contingent timepoints C, called *observable* timepoints. A contingent timepoint that is not observable is called a *hidden* timepoint. For a POSTNU, we will use the terminology *micro-projection* to describe each STN determined by the possible combinations of contingent links, which again are assumed to vary independently within their bounds.[4]

In the POSTNU of Figure 4, observe that the duration of the contingent links will not be known because they involve the hidden timepoint E whose occurrence cannot be observed. However since X is controllable, the duration from X to Y (resp. from X to W) will be known when the occurrence time of Y (resp. W) is observed. Here we say that X is the *closest non-hidden ancestor* of Y: the first executable or observable timepoint reached when following **backward** the chain of contingent links ending in Y. Because an STNU, hence also a POSTNU, does not allow two contingent links to have the same finish timepoint, each observable timepoint is guaranteed to have a unique *closest non-hidden ancestor*.

Definition 1 (Macro-link)**.** *Given an contingent timepoint Y with a closest non-hidden ancestor X, we call a* macro-link *the chain of contingent links from X to Y. If Y is observable, we will refer to it as an* observable macro-link *as its duration will be* observed *on the occurrence of Y.*

2. We allow a contingent link with $l(e) = u(e)$ although for STNUs it essentially behaves the same as a requirement link with the same bounds. However, the behavior can be different in the POSTNU context.
3. This incorporates the flaw correction of Hunsberger (2013) and provides for an instantaneous reaction.
4. We will later also define macro-projection. The distinction between the two will be somewhat analogous to the distinction between microstate and macrostate in thermodynamics.

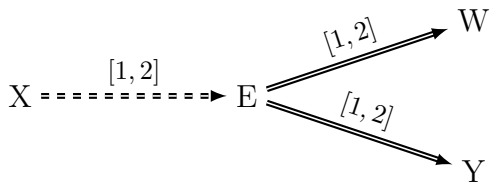


Figure 4: Example POSTNU where W and Y are observable contingent timepoints, E is a hidden contingent timepoint and X is an executable timepoint. The dashed representation of the XE contingent link is intended only to signal that its contingent timepoint E is hidden.

In Figure 4, there are two observable macro-links X $\Rightarrow$ E $\Rightarrow$ Y and X $\Rightarrow$ E $\Rightarrow$ W. Unlike the original contingent links of the POSTNU, the duration of these macro-links are observable. In Figure 4, observing the occurrence time of Y and W will provide us with the duration of their respective macro-links. Finally, note that the durations of the macro-links XW and XY are correlated since they share the hidden contingent link XE. As a consequence, observing the duration of XW to be 4 (implying XE = EY = 2), means that the possible values of XY are either 3 or 4, depending on the value taken by EY.

To emphasize the distinction from macro-links and to be more consistent with the micro-projection terminology, we may often refer to the original contingent links as *micro-links*.

Still in Figure 4, consider the micro-projection p_1 where XE=1, EW=EY=2 and the micro-projection p_2 where XE=2 and EW=EY=1. Even though the micro-links have different durations, the two macro-links have a duration of 3 in both micro-projections. Having only access to the duration of macro-links, an observer will not be able to distinguish p_1 from p_2 due to the hidden nature of E.

Definition 2. *Two micro-projections p_1 and p_2 are* observationally equivalent *if the durations of all the observable macro-links are the same in p_1 and p_2.*

Clearly this relation is symmetric, reflexive, and transitive: thus, an equivalence relation. Its equivalence classes will be called *macro-projections*. Note that the observable macro-links have fixed durations in a macro-projection.

For example, in the POSTNU (where only E is hidden)

$$\text{X} \xRightarrow{[1,10]} \text{E} \xRightarrow{[1,10]} \text{Y}$$

the set of micro-projections where XE and EY sum to 15, such as 6+9, 10+5, etc., constitute a macro-projection where XY = 15. Observe that the "extremal" macro-projections where XY = 2 and 20 each contain only one micro-projection.

For a POSTNU, an execution strategy S is *viable* if S_p is consistent with p for each micro-projection p. In order to define Dynamic Controllability for POSTNUs, we can now simply modify the definition of pre-history.

Definition 3. *The* observational pre-history *of a specific time t with respect to a schedule T, denoted by $T\{\trianglelefteq t\}$, specifies the durations of all observable macro-links that have finished up to and including time t.*

A *dynamic strategy* S for a POSTNU is defined similarly as for an STNU except we require it to satisfy

$$S_{p1}\{\trianglelefteq t\} = S_{p2}\{\trianglelefteq t\} \Rightarrow S_{p1}(x) = S_{p2}(x)$$

instead, i.e., we replace $\preceq$ with $\trianglelefteq$ in the definition. Then, as for an STNU, a POSTNU is *Dynamically Controllable* if there is a viable dynamic execution strategy.

As noted by Vidal and Fargier (1999), if the start timepoint of a contingent link is non-hidden, we may assume without loss of generality that it is an executable timepoint. (If not, we may replace it by an executable timepoint that is constrained to be simultaneous with the original. Note that this does not change the Dynamic Controllability status of the network.) However, we may *not* make this assumption for hidden timepoints.

3. Analysis and General Approach

In an STNU, the durations of all contingent links are assumed to be independent. As we saw, this remains true for the micro-links of a POSTNU but does not hold for the duration of the macro-links. This introduces a fundamental difference as observing the duration a macro-link might provide indirect information on the duration of another, correlated, macro-link.

Definition 4 (Correlated macro-links)**.** *We say that the durations of two macro-links are correlated iff they share at least one micro-link.*

Since the POSTNU definition does not permit two contingent links to have the same endpoint, notice that two correlated macro-links necessarily have the same source (the *closest non-hidden ancestor*) and at least their first micro-link will be shared. This makes the correlation relation between two macro-links transitive. Since it is also obviously reflexive and symmetric, the correlation relation defines an equivalence class that we can use to partition the macro-links (and their underlying micro-links) into non-independent groups: *hidden groups*

Definition 5 (Hidden groups)**.** *Two contingent links (micro-links) are in the same* hidden group *iff they appear in two macro-links whose durations are correlated.*

In our post-office example, this definition would imply two hidden groups that can be seen in Figure 5. From this partition, one can state that, e.g., observing the MN event might provide indirect information one the occurrence time of FP and FA, since their macro-links are in the same hidden group and thus correlated. On the other hand, being in another hidden group implies that the MA event will no provide any information regarding the duration of macro-links in the other group.

Since the POSTNU definition does not permit two contingent links to have the same endpoint, the links in a hidden group G will form the edges of a tree rooted at some non-hidden timepoint A, which will be the closest non-hidden ancestor for all the timepoint nodes in the tree. The leaves of the tree will be either observable timepoints, or terminal hidden timepoints. A possible algorithm for partitioning contingent links into hidden groups is to start a group from a contingent link with a non-hidden timepoint and recursively expand the group with outgoing contingent links until an observable timepoint is met (Algorithm 1).

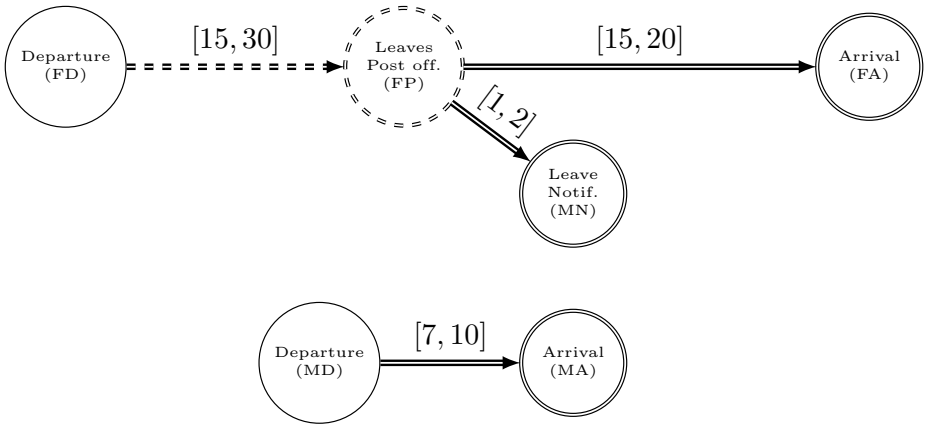


Figure 5: The two hidden groups of the post office example.

Algorithm 1 Partition contingent links of a POSTNU into hidden groups

```
1:  function HiddenGroups(⟨N, E, l, u, C, O⟩)
2:      HGroups = ∅
3:      for all contingent link cl ∈ C where activation(cl) is non-hidden do
4:          G ← {cl}
5:          if target(cl) is hidden then
6:              Q ← Q ∪ {cl}
7:          while Q non empty do
8:              cl ← pop(Q)
9:              for all cl′ ∈ C where target(cl) = activation(cl′) do
10:                 G ← G ∪ {cl′}
11:                 if target(cl) is hidden then
12:                     Q ← Q ∪ {cl′}
13:         HGroups ← HGroups ∪ {G}
        return HGroups
```

Each observable leaf will be called an *eye* of the hidden group. Notice that observing an eye of a hidden group will provide information limiting occurrence time of other contingent timepoints in the hidden group. In Figure 5 for instance, after observing the occurrence of MN at time 10 one can infer that the hidden FP event occurred in the [8,9] interval. This information can be further propagated to infer that the FA event will occur in the [23,29] range.

General approach Our approach for checking the Dynamic Controllability of a POSTNU essentially factors the problem into two parts:

(1) We transfer requirements constraints away from hidden timepoints, onto synthetic observable timepoints consistent with indirect knowledge brought by the eyes.

(2) We solve a dynamic controllability problem on the resulting network that is free of hidden timepoints but where knowledge on the duration of a macro-link may come from several observable timepoints.

For part (1), the occurrences of hidden timepoints will be restricted to values consistent with the observations of the eyes. In that case, satisfying the requirement links is tantamount to solving special cases of the temporal decoupling problem (Hunsberger, 2002) where the observations are treated as additional constraints on Nature, which is viewed as the second agent. Given a fixed macro-projection P, the relative time of a hidden timepoint E within its hidden group G may vary over the micro-projections of P. However, the earliest and latest occurrences of E relative to G will be fixed in P, and determined by the observations. We will see that the requirement links on E can be effectively transferred to a new pair of timepoints related to the earliest and latest possible occurrences of E.

For part (2), the resulting earliest and latest occurrences of contingent timepoints in the macro-projections can be expressed in terms of formulas with parameters that depend on the observations. These formulas can be interpreted as generalized versions of the labels used in the labeled distance graph of a standard STNU. We introduce corresponding generalizations of the STNU reduction rules. These are shown to be sound and complete for determining Dynamic Controllability.

We will prove the transformation steps and generalized reduction rules are sound by showing they leave the set of viable dynamic strategies unchanged. (This set may be empty if the POSTNU is not Dynamically Controllable.) Completeness will be a consequence of the fact that successful completion of the reduction process will make the projections dispatchable, and this implies Dynamic Controllability (Morris, 2014, 2016).

4. Transferring Requirement Constraints

The task of transferring requirement links away from hidden timepoints may be thought of as a type of multi-agent temporal decoupling problem (Hunsberger, 2002). Accordingly, we adopt a viewpoint where Nature is regarded as an agent scheduling the hidden groups subject to the observations, which are regarded as additional constraints on what Nature could have done. This leads to the following concept of a Nature STN.

Definition 6 (Nature STN)**.** *The* Nature STN *for a hidden group G with root timepoint X is an STN schema that contains an STN link for each micro-link in G, with the same bounds. In addition, for each eye Y of G, the network contains a rigid link from X to Y of length $\dot{Y}$, where $\dot{Y}$ is a variable representing the observed duration of the macro-link from X to Y.*

The links in the Nature STN that correspond to the micro-links will be called *concrete links*, while those corresponding to the observations will be called *observation links*.

Note that the Nature STN schema instantiates to a specific STN for each macro-projection (where the observed durations have definite values). By construction all such STNs are consistent as the existence of a macro-projection implies the existence of a micro-projection (and of the corresponding consistent assignment to the concrete links of the STN). Since this implies the absence of negative cycles, we can assume the existence of shortest paths that are well-defined and non-cyclic. We can also assume that the observed

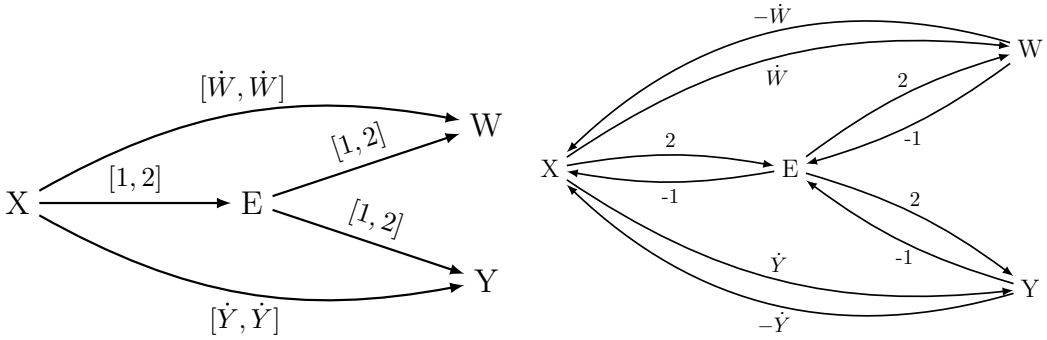


Figure 6: Nature STN schema (left) and corresponding distance graph (right) of the hidden group depicted in Figure 4.

durations are non-negative since they result from the combination of contingent links that are non-negative by definition.

We wish to transfer requirement constraints from hidden timepoints to equivalent constraints on observable timepoints. The following lemma supports an intermediate step in that direction. Suppose E is a hidden timepoint in some hidden group G rooted at X.

Lemma 1. *Consider some fixed macro-projection P where $[e^-, e^+]$ are the inferred* **tightest** *bounds of the XE temporal distance in the Nature STN. Let $E_{lo} = X + e^-$ and $E_{hi} = X + e^+$. Suppose $[z^-, z^+]$ are the bounds on a requirement link $\mathcal{Z}$ that has E as the source timepoint and some other timepoint Z (not part of the same hidden group) as the target timepoint. Then $\mathcal{Z}$ can be replaced by the constraint $E_{hi} + z^- \leq Z \leq E_{lo} + z^+$.*

Proof. Observe that the micro-projections within the P macro-projection will contain all the possible values for E in the solutions of the Nature STN relative to P, i.e., all E in the range $[\mathrm{E_{lo}}, \mathrm{E_{hi}}]$. Note that:

$$\begin{aligned} (\forall E \in [\mathrm{E_{lo}}, \mathrm{E_{hi}}] : E + z^- \leq Z) \quad &\equiv \quad \mathrm{E_{hi}} + z^- \leq Z \\ (\forall E \in [\mathrm{E_{lo}}, \mathrm{E_{hi}}] : Z \leq E + z^+) \quad &\equiv \quad Z \leq \mathrm{E_{lo}} + z^+. \end{aligned}$$

Thus, $\mathcal{Z}$ can be replaced by the equivalent $\mathrm{E_{hi}} + z^- \leq Z \leq \mathrm{E_{lo}} + z^+$. □

As illustrated in Figure 7, it is useful to think of $\mathrm{E_{lo}}$ and $\mathrm{E_{hi}}$ as *virtual* timepoints that have fixed offsets from X in each macro-projection (but the offsets may vary between macro-projections). The virtual timepoint $\mathrm{E_{hi}}$ would be executed at the latest possible occurrence time of E. To ensure the minimal delay z^- from E to Z, it follows that the Z must be scheduled at least z^- time units after $\mathrm{E_{hi}}$. Similarly, $\mathrm{E_{lo}}$ would be executed at the earliest possible occurrence time of E and any maximum delay constraint on E can be transferred to $\mathrm{E_{lo}}$.

Note that requirements between timepoints of the same hidden group are excluded from Lemma 1: their viability is independent of observation and any scheduling, and must be a logical consequence of the micro-links in the group.

Motivated by the lemma, we are interested in tight lower/upper bounds on the distance from X to E in the Nature STN that would hold in all macro-projections. We now develop

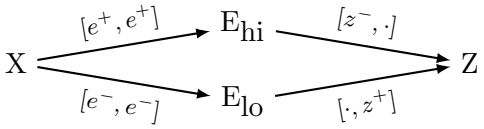


Figure 7: Result of applying Lemma 1 to transfer a requirement link E $\xrightarrow{[z^-,z^+]}$ Z, where e^- and e^+ are the tightest bounds on XE in the Nature STN.

symbolic formulas that express the bounds for each macro-projection in terms of the raw bounds and the observed durations.

Upper bound To exploit Lemma 1, we are interested in computing UB(X,E), the upper bound on the duration from X to E, where X is the root of the hidden group and E an hidden timepoint of the same group. Recall that the upper bound in an STN may be calculated as the shortest-path distance in the distance graph (Dechter et al., 1991). Denoting as $\mathrm{SD}(X, E)$ the length of the shortest path from X to E in the Nature STN, we thus have $\mathrm{UB}(X, E) = \mathrm{SD}(X, E)$.

Lemma 2. *In the Nature STN of a hidden group G, a minimal shortest path from the root X to any other timepoint E is either:*

- *a path composed only of concrete edges, or*
- *a path that starts with an observation edge X* $\xrightarrow{\dot{Y}}$ *Y and ends with the shortest path over concrete edges from Y to E (where Y* $\in$ *eyes(G)).*

Proof. A Nature STN being consistent by definition, it contains no negative cycles. Thus for any shortest path, there is a minimal shortest path of the same length that is non-cyclic and can be obtained by removing the zero-length cycle. Thus any minimal shortest path is non-cyclic.

The first edge of any path from X is either an observation edge X $\xrightarrow{\dot{Y}}$ Y or a concrete edge. Observe that in the distance graph of a Nature STN all observation edges start or end at the root X of the hidden group. Except for the first edge, there might not be another observation edge in the minimal shortest path as it would imply that the path goes again through X and is cyclic. □

Let us define CSD(A,B) as the shortest distance over only concrete edges of the Nature STN. Based on Lemma 2, the length of a shortest path from X to E is thus:

- $\mathrm{CSD}(X, E)$ if the path only involves concrete edges, and
- $\dot{Y} + \mathrm{CSD}(Y, E)$ if it passes through an eye Y of the hidden group.

We can now express $\mathrm{UB}(X, E)$ as the length of the shortest path among all candidates:

$$\mathrm{UB}(X, E) = \mathrm{SD}(X, E) = \min \{ \mathrm{CSD}(X, E), \min_{Y \in eyes(G)} (\dot{Y} + \mathrm{CSD}(Y, E)) \}$$

Recalling that $\dot{Y} = Y - X$, it is convenient to regard the root timepoint X as an additional special eye called the *root eye* such that $\dot{X} = 0$ always. (The other eyes will then be called

leaf eyes to distinguish them.) Then we can rewrite the formula (where the notation Eyes(G) includes the root eye) as

$$\mathrm{UB}(X, E) = \min_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CSD}(Y, E))$$

Lower bound We now turn our attention to $\mathrm{LB}(X, E)$, the lower bound on the duration from X to E. In an STN distance graph, this lower bound is equal to the negated shortest distance from E to X, i.e., $\mathrm{LB}(X, E) = -\mathrm{SD}(E, X)$ (Dechter et al., 1991).

Reapplying the reasoning behind Lemma 2, observe that a minimal shortest path from E to X is either exclusively composed of concrete edges or ends with a single $Y \xrightarrow{-\dot{Y}} X$ observation edge where Y is an eye of the hidden group. Again, the shortest path distance from E to X, can be expressed as a min expression over the eyes:

$$\mathrm{SD}(E, X) = \min_{Y \in \mathrm{Eyes}(G)} (\mathrm{CSD}(E, Y) - \dot{Y}).$$

Recalling that $\mathrm{LB}(X, E) = -\mathrm{SD}(E, X)$, we express the lower bound as

$$\mathrm{LB}(X, E) = \max_{Y \in \mathrm{Eyes}(G)} (\dot{Y} - \mathrm{CSD}(E, Y)).$$

Notice that min changes to max and the rigid link value $\dot{Y}$ ends up as an added term in the lower bound representation.

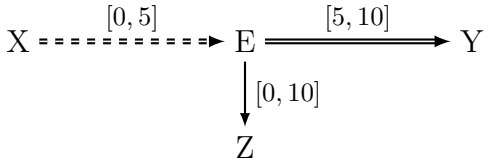
Synthesis Let us define the *concrete upper bound* $\mathrm{CUB}(Y, E) = \mathrm{CSD}(Y, E)$ and the *concrete lower bound* $\mathrm{CLB}(Y, E) = -\mathrm{CSD}(E, Y)$, the bounds on YE implied by the concrete edges. As they involve only concrete edges, these concrete bounds will boil down to a unique numeric constant in every macro-projection. Substituting them in the previous formulas, we obtain

$$\mathrm{UB}(X, E) = \min_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CUB}(Y, E))$$
$$\mathrm{LB}(X, E) = \max_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CLB}(Y, E))$$

as concise bound expressions.

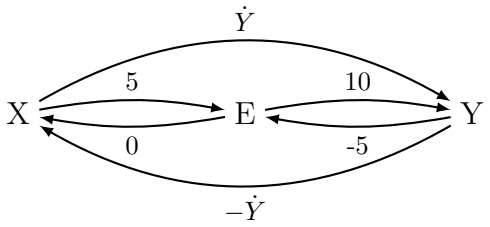
Since $\mathrm{E}_{\mathrm{hi}} = X + \mathrm{UB}(X, E)$ and $\mathrm{E}_{\mathrm{lo}} = X + \mathrm{LB}(X, E)$, we could now apply Lemma 1 to transfer any requirements on E to requirements on E_{hi} and E_{lo}, whose fixed bounds with respect to X can be expressed in terms of formulas involving the observations.

Example 1. *Consider the following POSTNU where X and Z are executable timepoints, E is hidden, and Y is observable.*



Even though E is not observable, the occurrence time of both X and Y provide indirect information on E. We are thus interested in reformulating the E $\xrightarrow{[0,10]}$ Z *requirement link in terms of symbolic expressions involving the occurrence times of X and Y.*

E is part of a hidden group composed of the micro-links XE and EY, for which the corresponding Nature STN distance graph is the following (note that Z is not part of the hidden group of E and hence not represented in the Nature STN).



It is easy to see that the shortest distances XE and EX depend on $\dot{Y}$, the duration of the macro-link XY, resulting in the following shortest distance expressions:

$$SD(X,E) = \min\{5, \dot{Y} - 5\}$$
$$SD(E,X) = \min\{0, 10 - \dot{Y}\}$$

and the corresponding bounds:

$$UB(X,E) = \min\{5, \dot{Y} - 5\}$$
$$LB(X,E) = \max\{0, \dot{Y} - 10\}$$

Interpreting these expression as the virtual timepoints E_{lo} and E_{hi}, we obtain the following definitions and corresponding bounds on Z.

$$E_{lo} = X + \max\{0, \dot{Y} - 10\} \qquad Z \leq E_{lo} + 10$$
$$E_{hi} = X + \min\{5, \dot{Y} - 5\} \qquad Z \geq E_{hi}$$

Let us give an intuitive interpretation of this result. We know that E occurs after X. Additionally, we know that E occurs at most 10 time units before Y. This is reflected in the definition of the virtual timepoint E_{lo} which should be interpreted as: it is not possible for E to occur before E_{lo}. The requirement of scheduling Z at most 10 time units after E_{lo} thus ensures that Z is scheduled at most 10 time units after E, which was our original constraint.

Similarly, we know that E occurs at most 5 time units after X and at least 5 time units before Y. This is reflected in the definition of the virtual timepoint E_{hi} which should be interpreted as: it is not possible for E to occur after E_{hi}. The requirement of scheduling Z after E_{hi} ensure that Z is scheduled after E which was our original constraint.

Example 2 (running example)**.** *We present here the effect of computing the lower and upper bound expressions of the* FA *timepoint for the post office example of Figure 3:*

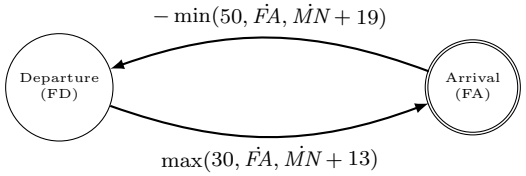
$$FD + LB(FD, FA) \leq FA \leq FD + UB(FD, FA)$$
$$FD + \max\{30, \dot{FA}, \dot{MN} + 13\} \leq FA \leq FD + \min\{50, \dot{FA}, \dot{MN} + 19\}$$

which should be interpreted as:

- *FA is known **not** to have occurred as long as: less that 30 minutes have elapsed since FD, FA has not occurred **and** less that 13 time units have elapsed since the occurrence of MN.*
- *FA is known to have occurred once: 50 minutes have elapsed since the the friend's departure (FD), FA has been observed **or** 19 minutes have elapsed since the receiving the notification (MN).*

Note that in any macro-projection, the two bounds converge to a single value $\mathrm{FD} + \dot{\mathrm{FA}}$.

The figure below gives an initial intuition of how this information can be represented in a distance graph by expressing the bounds as the corresponding edges. Building such a distance graph in a systematic way will be the purpose of the next two sections.



5. Enforcing Observability of Virtual Timepoints

In this section we are interested in making the virtual timepoints E_{lo} and E_{hi} observable, which is desirable to simplify the mechanisms of dynamic controllability checking. As a motivation for doing so, we first informally discuss how the full observability is leveraged in STNUs. Then we propose a way to transfer the constraints on virtual timepoints onto observable timepoints.

Exploiting the full observability of STNUs In an STNU, the occurrence time of a contingent (observable) timepoint is always known when it occurs. Consider a contingent link $X \Rightarrow Y$ where Y is observable and X is executable. The duration $\dot{Y}$ of the link will be known at time $X + \dot{Y}$ which is also the time at which Y will occur. Note that the contingent Y timepoint could be replaced by an executable timepoint Z subject to the constraint $X \xrightarrow{[\dot{Y},\dot{Y}]} Z$. Indeed, a dynamic execution strategy will need to wait until the $\dot{Y}$ duration is observed at time $X + \dot{Y}$ and immediately schedule Z. Both Y and Z would occur exactly at the same time, making them indistinguishable, regardless of whether their occurrence time is decided by Nature or by the controlling agent. In a nutshell, it means that an *observable* contingent timepoint Y can be treated as an executable subject to constraints on the $\dot{Y}$ duration. In STNUs, DC-checking algorithms such as the one of Morris (2014) exploit this property to avoid distinguishing between contingent and executable timepoints in their reduction rules.

Making virtual timepoints observable Lemma 1 permits us to transfer requirements on hidden timepoints to virtual timepoints E_{hi} and E_{lo} that are fixed within each macro-projection, and depend only on the observation durations. One drawback, however, is that E_{hi} and E_{lo} may not be themselves observable.

For example, given an eye Y, the $\mathrm{CUB}(Y, E)$ value could be either negative or non-negative. (The E timepoint could be an ancestor or cousin of Y with respect to the tree

structure of the hidden group.) If it is negative, then the value of $\dot{Y} + \mathrm{CUB}(Y, E)$ will not be known until later, when Y is observed. Consequently, we may not be able to tell whether

$$\mathrm{E_{hi}} = X + \min_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CUB}(Y, E))$$

has occurred or not, *at the time* it occurs. We now show how virtual timepoints can be made observable by artificially delaying them, while preserving viable strategies by compensating for this delay on the requirement links they are involved in. Our objective in the process is to build a network with homogeneous properties that can be exploited in the reduction phase.

Note that we can always choose some minimal fixed value $\delta_{\mathrm{hi}}(E) \geq 0$ such that $\mathrm{CUB}(Y, E) + \delta_{\mathrm{hi}}(E) \geq 0$ for every $Y \in \mathrm{Eyes}(G)$. The $\delta_{\mathrm{hi}}(E)$ value depends only on the weights of the concrete links and does not vary with the observations.[5]

In contrast to the $\mathrm{E_{hi}}$ value, $\mathrm{E_{hi}} + \delta_{\mathrm{hi}}(E)$ will be observable. This suggests introducing a new virtual timepoint $\mathrm{E_{hi+}} = \mathrm{E_{hi}} + \delta_{\mathrm{hi}}(E)$. Similarly, we can introduce a virtual timepoint $\mathrm{E_{lo+}} = \mathrm{E_{lo}} + \delta_{\mathrm{lo}}(E)$ so that the offsets in the $\mathrm{LB}(X, E) + \delta_{\mathrm{lo}}(E)$ formula will be non-negative.

The following lemmas justify the transfer of requirement constraints from $\mathrm{E_{hi}}$ and $\mathrm{E_{lo}}$ to $\mathrm{E_{hi+}}$ and $\mathrm{E_{lo+}}$, respectively, and establish their observability. The lemmas are stated for completeness. We omit the proofs, which are immediate.

Lemma 3. *If $E_{hi+} = E_{hi} + \delta_{\mathrm{hi}}(E)$, then $E_{hi} + z^- \leq Z$ is equivalent to $E_{hi+} + (z^- - \delta_{\mathrm{hi}}(E)) \leq Z$. If $E_{lo+} = E_{lo} + \delta_{\mathrm{lo}}(E)$ then $Z \leq E_{lo} + z^+$ is equivalent to $Z \leq E_{lo+} + (z^+ - \delta_{\mathrm{lo}}(E))$.*

Lemma 4. *If an event $\dot{Y}$ is observable, then the event $\dot{Y} + p$, where $p \geq 0$ is some fixed value, is also observable. If every event in some set S is observable, then $\min_{y \in S}(y)$ and $\max_{y \in S}(y)$ are also observable.*

We can paraphrase $\min_{y \in S}(y)$ as "whichever is earliest of the events in S, and $\max_{y \in S}(y)$ as "whichever is latest of the events in S." Applying this interpretation to the

$$\mathrm{E_{hi+}} - X = \min_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CUB}(Y, E) + \delta_{\mathrm{hi}}(E))$$
$$\mathrm{E_{lo+}} - X = \max_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CLB}(Y, E) + \delta_{\mathrm{lo}}(E))$$

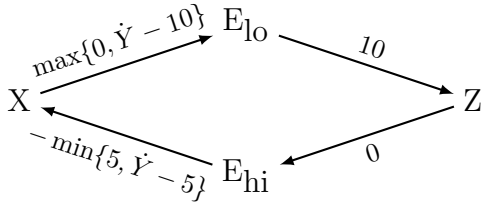
expressions, we can regard $\mathrm{E_{hi+}}$ and $\mathrm{E_{lo+}}$ as *compound observables*, derived from the eyes, with non-negative offsets calculated from the bounds of the contingent links. Repeated applications of the two lemmas have the effect of transferring the requirement constraints from hidden timepoints to equivalent constraints on compound observables.

Example 3. *Following on from Example 1, where we had*

$$E_{lo} = X + \max\{0, \dot{Y} - 10\} \qquad Z \leq E_{lo} + 10$$
$$E_{hi} = X + \min\{5, \dot{Y} - 5\} \qquad Z \geq E_{hi}$$

This could be interpreted as the following distance graph.

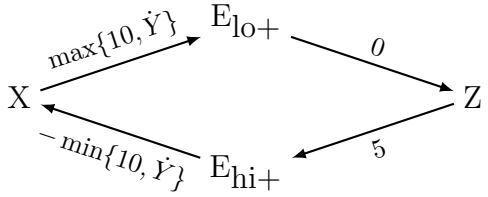
5. For definiteness, we could choose $\delta_{\mathrm{hi}}(E)$ to be the smallest value with this property. The same value, but with opposite sign, is added to the weight of the transferred requirement constraint. As we see below in a lemma, the transferred edge is logically equivalent to the original, so it actually would not matter if a larger value were chosen to be added and subtracted.



Consider the macro-projection where $\dot{Y} = 12$. Further assume that X is scheduled at the temporal origin ($X = 0$). In this macro-projection, E_{lo} would occur at time 2 while E_{hi} would occur at time 5. However this information is unknown until the duration $\dot{Y}$ is observed at time 12 (when the observable Y is observed).

We can now define the compound observables $E_{hi+} = E_{hi} + \delta_{\mathrm{hi}}(E)$ and $E_{lo+} = E_{lo} + \delta_{\mathrm{lo}}(E)$ by choosing the appropriate $\delta_{\mathrm{hi}}(E)$ and $\delta_{\mathrm{lo}}(E)$ terms such that all terms in the min/max expressions are non-negative. Choosing $\delta_{\mathrm{hi}}(E) = 5$ and $\delta_{\mathrm{lo}}(E) = 10$, we obtain the following expressions and corresponding distance graph.

$$E_{lo+} = X + \max\{10, \dot{Y}\} \qquad Z \leq E_{lo+}$$
$$E_{hi+} = X + \min\{10, \dot{Y}\} \qquad Z \geq E_{hi+} - 5$$



Example 4. *Suppose we have five events: X, E, A, Y, and Z, where E is hidden and Z is executable. We have a contingent link* $\mathrm{X} \xrightarrow{[0,5]} \mathrm{E}$*, a contingent link* $\mathrm{E} \xrightarrow{[1,1]} \mathrm{A}$*, and a contingent link* $\mathrm{E} \xrightarrow{[10^6,10^6]} \mathrm{Y}$*. We also have a requirement link* $\mathrm{E} \xrightarrow{[2,2]} \mathrm{Z}$*.*

It's clear from the example setup that Y is in some sense superfluous. Intuitively, we can execute Z exactly one unit of time after we see A. It may therefore seem odd that Y has an outsize influence on the encoding of E_{hi+} and E_{lo+} via the parameter δ, although the modification produces a logically equivalent network.

This is not a problem for correctness of the approach. The reformulated network contains these constraints:

$$\mathrm{X} \xleftarrow{[-\min(5+10^6, \dot{\mathrm{A}}-1+10^6, \dot{\mathrm{Y}})]} \mathrm{E_{hi+}} \xleftarrow{-2+10^6} \mathrm{Z}$$

$$\mathrm{X} \xrightarrow{[\max(0+10^6, \dot{\mathrm{A}}-1+10^6, \dot{\mathrm{Y}})]} \mathrm{E_{lo+}} \xrightarrow{+2-10^6} \mathrm{Z}$$

Since $0 \leq \dot{A} - 1 \leq 5$ and $\dot{Y} = \dot{A} - 1 + 10^6$ in this example, these reduce to

$$\mathrm{X} \xleftarrow{-((\dot{\mathrm{A}}+1)-2+10^6)} \mathrm{E_{hi+}} \xleftarrow{-2+10^6} \mathrm{Z}$$

$$\mathrm{X} \xrightarrow{(\dot{\mathrm{A}}+1)-2+10^6} \mathrm{E_{lo+}} \xrightarrow{+2-10^6} \mathrm{Z}$$

which can be rewritten as

$$(\mathrm{A}+1) \xleftarrow{-(-2+10^6)} \mathrm{E}_{\mathrm{hi}+} \xleftarrow{-2+10^6} \mathrm{Z}$$

$$(\mathrm{A}+1) \xrightarrow{-2+10^6} \mathrm{E}_{\mathrm{lo}+} \xrightarrow{+2-10^6} \mathrm{Z}$$

and it is now easy to see that executing Z at time $(A+1)$ *is consistent with both requirements. (It can be seen from the description that the* $E_{hi+}-(A+1) = -(-2+10^6)$ *and* $E_{lo+}-(A+1) = -2+10^6$ *values are fixed in this example.) Thus, the intuitive strategy still works for the reformulated network. This is not surprising since replacing requirements by equivalent ones preserve all viable strategies, including the dynamic ones. (It is of interest that, when we later consider plus/minus reductions, the added and subtracted* 10^6 *amounts will cancel in this example.)*

The rationale for the introduction of the E_{hi+} *and* E_{lo+} *observables is that it allows us to confine the initialization phase to operations that preserve all the viable strategies. We defer operations that may preserve only the dynamic viable strategies until the plus/minus reduction phase. (Other approaches are possible at the cost of complicating the initialization phase.)*

6. Construction of a Generalized Distance Graph

Having identified a way to transfer requirements from hidden timepoints onto observable ones, we now turn our attention to the construction of a *generalized distance graph* as a generalization of the *labeled distance graph* of conventional STNUs. This section's focus is on the construction and characterization of the initial graph. A later section will introduce the *reduction rules* whose role are to extend the graph with implied edges until a violation of dynamic controllability is identified or quiescence is reached.

6.1 Generalized Labels

Here we interpret the max/min bound expressions for the compound observables as generalized labels analogous to those in the labeled distance graph of a conventional STNU. (See for example (Morris, 2014)).

Since the requirements have been transferred to observables, our further discussion is mostly in terms of the macro-projections. Thus, in the following, we may sometimes use the unqualified term "projection" to mean macro-projection, when it is clear from the context what is intended. (However, we must remain mindful that the independence of macro-links is limited by possibly shared micro-links.)

Recall that a compound observable of the form $\mathrm{E}_{\mathrm{hi}+}$ provides a lower-bound

$$\min_{Y \in \mathrm{Eyes}(G)} \{ \dot{Y} + \mathrm{CUB}(Y, E) + \delta_{\mathrm{hi}}(E) \}$$

on the X to $\mathrm{E}_{\mathrm{hi}+}$ distance that could potentially combine with a $\mathrm{E}_{\mathrm{hi}+}$ to Z lower bound of some requirement. In our generalized distance graph, this lower bound on the X to $\mathrm{E}_{\mathrm{hi}+}$ distance will be represented as an edge $X \leftarrow \mathrm{E}_{\mathrm{hi}+}$ with weight

$$\ell_{\mathrm{hi}+}(E) = -\min_{Y \in \mathrm{Eyes}(G)} \{ \dot{Y} + \mathrm{CUB}(Y, E) + \delta_{\mathrm{hi}}(E) \} . \quad (1)$$

Notice that all the $\dot{Y} + \mathrm{CUB}(Y, E) + \delta_{\mathrm{hi}}(E)$ terms evaluate to non-negative numbers. Moreover, $\dot{Y}$ for the leaf eyes is always positive. Thus, the negated expression will evaluate to a negative number. The edge is analogous to an Upper-Case edge in an STNU distance graph, and for convenience we will sometimes call it an Upper-Case edge here.

Similarly, for a compound observable of the form $\mathrm{E}_{\mathrm{lo}+}$, there will be an edge $X \rightarrow \mathrm{E}_{\mathrm{lo}+}$ with weight

$$\ell_{\mathrm{lo}+}(E) = \max_{Y \in \mathrm{Eyes}(G)} \{ \dot{Y} + \mathrm{CLB}(Y, E) + \delta_{\mathrm{lo}}(E) \} . \quad (2)$$

The $\dot{Y} + \mathrm{CLB}(Y, E) + \delta_{\mathrm{lo}}(E)$ terms all evaluate to non-negative numbers in each macro-projection. The edge is analogous to a Lower-Case edge in an STNU distance graph, and we will sometimes use that terminology here also.

When there are multiple leaf eyes, it is also possible for observations of one eye to provide information that restricts the possible time of occurrence of another eye *before* it has occurred, and this may be useful for an early determination that a requirement has been satisfied. This may happen if the macro-links for the two eyes share micro-links. The generalized labels for each eye Y must therefore reflect inferences from observations of the other eyes. As it turns out, the appropriate labels for Y are the same as if Y were also a hidden timepoint. The $\dot{Y}$ coming from the direct observation of Y is simply included as another term in the min/max formulas. Note, however, that the $\mathrm{CLB}(Y, Y)$ and $\mathrm{CUB}(Y, Y)$ terms will be 0. As an example, in the POSTNU of Figure 4, $\mathrm{UB}(X, Y) = \max(2, \dot{Y}, \dot{W} - 1)$ and $\mathrm{LB}(X, Y) = \min(4, \dot{Y}, \dot{W} + 1)$.

Relationship with STNU It is instructive to regard an ordinary STNU as a special case of a POSTNU where there are no hidden timepoints. If we were to treat each contingent link $\mathrm{X} \xrightarrow{[y^-, y^+]} \mathrm{Y}$ as a degenerate "hidden group" with a single micro-link and a single leaf eye, the above analysis could still be applied, resulting in an upper-case label of the form $-\min(\dot{Y}, y^+)$ and a lower-case label of $\max(\dot{Y}, y^-)$. These may be compared to the conventional $Y{:}{-}y^+$ and $y : y^-$ labels of an STNU, and suggests a semantic interpretation of those labels. It is helpful to bear this comparison in mind when we consider reduction rules, in the next section.

6.2 Construction of the Labeled Distance Graph

Given a POSTNU Π, we now give the complete procedure for the construction of the labeled distance graph (V, Edges). The set of vertices V will be composed of the controllable and observable timepoints of Π as well as the two compound observables of each hidden timepoint of Π. The set of edges Edges is obtained by *(i)* converting macro-links to upper and lower case edges, and *(ii)* transferring requirement links involving hidden timepoints to the corresponding compound observables.

The algorithm is detailed in Algorithm 2. It works by first extracting all hidden groups. For each hidden group, lines 8-10 add the compound observable timepoints and the corresponding upper and lower case edges to the network. The $\delta_{\mathrm{lo}}(E)$ and $\delta_{\mathrm{hi}}(E)$ can be arbitrarily fixed to a constant number greater that the length of any shortest path in the Nature STN of E. Each observable eye is also added with the corresponding lower and upper bound expressions (being observable, those do not need to be delayed). Then lines 13-20 add the requirement constraints, transferring from the hidden timepoints to the new observables.

Algorithm 2 Construction of the labeled distance graph of a POSTNU Π

procedure DISTANCEGRAPH(Π)
  $V \leftarrow$ CONTROLLABLES(Π) $\cup$ OBSERVABLES(Π)
  $Edges \leftarrow \emptyset$
  $\{ G_1, \ldots, G_n \} \leftarrow$ HIDDENGROUPS(Π)
  for all G_i **do**
    $X \leftarrow root(G_i)$
    Construct nature STN of G_i
    for all hidden timepoint $E \in G_i$ **do**
      $V \leftarrow V \cup \{ \mathrm{E}_{\mathrm{lo}+}, \mathrm{E}_{\mathrm{hi}+} \}$
      $Edges \leftarrow Edges \cup \{ \mathrm{X} \xrightarrow{\ell_{\mathrm{lo}+}(\mathrm{E})} \mathrm{E}_{\mathrm{lo}+}, \mathrm{E}_{\mathrm{hi}+} \xrightarrow{\ell_{\mathrm{hi}+}(\mathrm{E})} \mathrm{X} \}$
    for all observable timepoint $Y \in eyes(G_i)$ **do**
      $Edges \leftarrow Edges \cup \{ \mathrm{X} \xrightarrow{\mathrm{UB(X,Y)}} \mathrm{Y}, \mathrm{Y} \xrightarrow{-\mathrm{LB(X,Y)}} \mathrm{X} \}$
  for all $\mathrm{A} \xrightarrow{\ell} \mathrm{B} \in$ REQUIREMENTS(Π) **do**
    if A is hidden **then**
      $A \leftarrow \mathrm{A}_{\mathrm{lo}+}$
      $\ell \leftarrow \ell - \delta_{\mathrm{lo}}(A)$
    if B is hidden **then**
      $B \leftarrow \mathrm{B}_{\mathrm{hi}+}$
      $\ell \leftarrow \ell + \delta_{\mathrm{hi}}(B)$
    $Edges \leftarrow Edges \cup \{ \mathrm{A} \xrightarrow{\ell} \mathrm{B} \}$
  return $(V, Edges)$

Complexity Observe that in a Nature STN with n vertices, the number of edges m is bounded above by $4 \times (n-1)$: each of the $n-1$ non-root vertex has exactly one incoming contingent link, which is transformed into at most 4 edges in the Nature STN distance graph. In the worst case, computing the labels of the generalized distance graph requires us to know the shortest distance between any two timepoints in a Nature STN. This can be done in a single pass of Johnson's algorithm with a complexity of $O(n^2 \times log(n) + n \times m)$. Given that $m \in O(n)$ this further reduces to $O(n^2 \times log(n))$. This process must be repeated for each of the Nature STNs.

Further observe that the Nature STNs are built by partitioning the set of contingent edges and that there is exactly 1 contingent link per contingent timepoint. The total number of nodes across all Nature STNs is thus bounded above by $2 \times |C|$ where $|C|$ is the number of contingent timepoints. The complexity of computing the generalized labeled graph in Algorithm 2 is thus $O(|C|^2 \times log|C|)$.

This dominates the complexity of extracting the hidden groups which is linear in the number of contingent links (Algorithm 1). For completeness we must also incorporate the cost of transforming the requirement edges, each of which is treated exactly once in constant time. The overall complexity of Algorithm 2 is thus $O(|C|^2 \times log|C| + |E|)$, where $|E|$ is the number of requirement links in the original POSTNU. As we will see, this is dominated by the overall complexity of checking the Dynamic Controllability of the resulting network.

Example 5 (running example)**.** *Following on from Example 2, we present in Figure 8 a directly relevant subset of the generalized distance graph that would be built by our procedure for the post office example of Figure 3.*

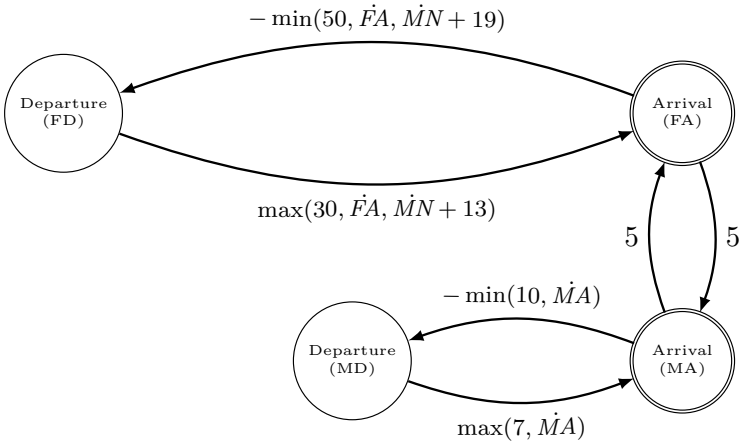


Figure 8: Generalized distance graph of the post-office example (Figure 3). Note that the MN, $\text{FP}_{\text{hi}+}$ and $\text{FP}_{\text{lo}+}$ timepoints are not subject to any requirement and thus omitted from the graph.

6.3 Initial Labels: Properties and Notation

In this section, we note some properties of the generalized labels in the constructed labeled distance graph. These will be useful later when we introduce derived labels, and establish invariants that are preserved by the derivations. Here, we show the properties hold for the labels on the initial edges between root timepoints and observables. Recall the formulas

$$\text{UB}(X, E) = \min_{Y \in \text{Eyes}(G)} (\dot{Y} + \text{CUB}(Y, E))$$
$$\text{LB}(X, E) = \max_{Y \in \text{Eyes}(G)} (\dot{Y} + \text{CLB}(Y, E)).$$

We have the following lemmas that will be important for the derivations.

Lemma 5. *The following inequalities hold in general.*

$$\mathit{CUB}(X, E) \geq \mathit{CUB}(Y, E) \text{ for all } Y \in \mathit{eyes}(G)$$
$$\mathit{CLB}(X, E) \geq \mathit{CLB}(Y, E) \text{ for all } Y \in \mathit{eyes}(G)$$

Proof. Recall that $\text{CLB}(P, Q) = -\text{CUB}(Q, P)$ for any P and Q.

Consider any $Y \in \text{eyes}(G)$. Then

$$\text{CUB}(Y, E) \leq \text{CUB}(Y, X) + \text{CUB}(X, E) \leq \text{CUB}(X, E)$$

by the triangle inequality and $\text{CUB}(Y, X) = -\text{CLB}(X, Y) \leq 0$. Also, by the triangle inequality, we have

$$\text{CUB}(E, X) \leq \text{CUB}(E, Y) + \text{CUB}(Y, X)$$

and so

$$\mathrm{CLB}(Y,E) + \mathrm{CLB}(X,Y) \leq \mathrm{CLB}(X,E).$$

Then

$$\mathrm{CLB}(Y,E) \leq \mathrm{CLB}(Y,E) + \mathrm{CLB}(X,Y) \leq \mathrm{CLB}(X,E).$$

□

Consider a fixed macro-projection P. With respect to P and the above formulas, we will say an eye $W \in \mathrm{Eyes}(G)$ is *UB active* for E if it sets the tightest UB of E:

$$\dot{W} + \mathrm{CUB}(W,E) = \min_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CUB}(Y,E)).$$

We will also say a subset $\mathcal{W}$ of $\mathrm{Eyes}(G)$ is *UB active* for E if it contains an eye that is UB active for E. Note that this implies

$$\min_{Y \in \mathcal{W}} (\dot{Y} + \mathrm{CUB}(Y,E)) = \min_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CUB}(Y,E)).$$

Analogously, $W \in \mathrm{Eyes}(G)$ is *LB active* for E if

$$\dot{W} + \mathrm{CLB}(W,E) = \max_{Y \in \mathrm{Eyes}(G)} (\dot{Y} + \mathrm{CLB}(Y,E))$$

and we similarly extend this concept to subsets of $\mathrm{Eyes}(G)$.

The following lemma identifies special macro-projections $P_{\max}$, where the root eye is UB active, and $P_{\min}$, where the root eye is LB active.

Lemma 6. *There is a macro-projection P_{max} where $UB(X,E) = CUB(X,E)$, and a macro-projection P_{min} where $LB(X,E) = CLB(X,E)$.*

Proof. Let us consider the All-Max projection $P_{\max}$ where all micro-links take on their maximum bounds. Given an observable Y, note that in the $P_{\max}$ projection, $\dot{Y}$ takes its maximum value $\dot{Y}_{\max}$ and there is a *concrete* path XY of the length $\dot{Y}_{\max}$ in the Nature STN. Thus the XY concrete shortest path can not be greater that $\dot{Y}_{\max}$:

$$\mathrm{CUB}(X,Y) \leq \dot{Y}_{\max}.$$

Adding the YE distance to both sides of the inequality we obtain

$$\mathrm{CUB}(X,Y) + \mathrm{CUB}(Y,E) \leq \dot{Y}_{\max} + \mathrm{CUB}(Y,E).$$

With the triangle inequality we can state $\mathrm{CUB}(X,E) \leq \mathrm{CUB}(X,Y) + \mathrm{CUB}(Y,E)$ and thus

$$\mathrm{CUB}(X,E) \leq \dot{Y}_{\max} + \mathrm{CUB}(Y,E).$$

Hence in the All-Max projection $P_{\max}$, $\mathrm{UB}(X,E) = \mathrm{CUB}(X,E)$, the smallest term of the min expression.

Similarly, the All-Min projection, where each micro-links takes on its minimum bound, satisfies the conditions for $\mathrm{LB}(X,E) = \mathrm{CLB}(X,E)$. □

Recall that macro-link observations have only limited independence from each other—their durations may be correlated if they involve shared micro-links. Nevertheless, the following lemma establishes some flexibility with respect to which observations are active in the max/min computations. This will be useful for showing the existence of projections with different properties that have related past histories.

Lemma 7. *Suppose: $\mathcal{E}$ is a subset of Eyes(G) that includes the root eye; E is a hidden timepoint of G; and P is any macro-projection. Then there exists macro-projections P' and P'', in which $\dot{Y}$ is unchanged for $Y \in \mathcal{E}$, such that (a) $\mathcal{E}$ is UB active for E in P', and (b) $\mathcal{E}$ is LB active for E in P''.*

Proof. Consider the Nature STN instance Γ corresponding to P. Delete the rigid constraints corresponding to the W observations for W not in $\mathcal{E}$. This forms a new STN Γ'.

For part (a), by (Dechter et al., 1991), there is a solution of Γ' where $\dot{W} = \mathrm{UB}(X, W)$ for W not in $\mathcal{E}$. Consequently, we can add to Γ' rigid constraints of the form $\dot{W} = \mathrm{UB}(X, W)$, for W not in $\mathcal{E}$, without creating an inconsistency, and we form P' accordingly.

Note that in Γ', for W not in $\mathcal{E}$,

$$\mathrm{UB}(X, E) \leq \mathrm{UB}(X, W) + \mathrm{UB}(W, E) \leq \mathrm{UB}(X, W) + \mathrm{CUB}(W, E)$$

using the triangle inequality. It follows that, in P',

$$\min_{Y \in \mathcal{E}}(\dot{Y} + \mathrm{CUB}(Y, E)) \leq \dot{W} + \mathrm{CUB}(W, E)$$

for W not in $\mathcal{E}$. Thus, $\mathcal{E}$ is UB active for E in P'.

For part (b), we analogously use the (Dechter et al., 1991) solution where $\dot{W} = \mathrm{LB}(X, W)$ to form P''. In Γ' we have

$$\mathrm{LB}(X, E) \geq \mathrm{LB}(X, W) + \mathrm{LB}(W, E) \geq \mathrm{LB}(X, W) + \mathrm{CLB}(W, E)$$

so in P'' we have

$$\max_{Y \in \mathcal{E}}(\dot{Y} + \mathrm{CLB}(Y, E)) \geq \dot{W} + \mathrm{CLB}(W, E)$$

and thus $\mathcal{E}$ is LB active for E in P''. □

Label Notation For the rest of the paper, it is convenient to introduce a more compact notation for generalized labels, as follows. Suppose I is an index set for eyes(G). We can rewrite the initial labels as

$$\ell_{\mathrm{hi}+}(E) = -\min(v_0, \min_{i \in I}(\dot{Y}_i + v_i))$$

where $v_0 = \mathrm{CUB}(X, E) + \delta_{\mathrm{hi}}(E)$ and $v_i = \mathrm{CUB}(Y_i, E) + \delta_{\mathrm{hi}}(E)$. Similarly, we can rewrite

$$\ell_{\mathrm{lo}+}(E) = \max(u_0, \max_{j \in I}(\dot{Y}_j + u_j))$$

where $u_0 = \mathrm{CLB}(X, E) + \delta_{\mathrm{lo}}(E)$ and $u_j = \mathrm{CLB}(Y_j, E) + \delta_{\mathrm{lo}}(E)$.

This notation makes it easier to treat the contribution from the root eye specially which will be useful in the study of both the initial labels (from Algorithm 2) and the derived labels. Notice that it follows from Lemma 6 that

$$u_0 = \min_P(\max(u_0, \max_{j \in I}(\dot{Y}_j + u_j))) \text{ and } -v_0 = \min_P(-\min(v_0, \min_{i \in I}(\dot{Y}_i + v_i)))$$

where P ranges over all the projections. Thus, u_0 and $-v_0$, respectively, are the tightest values the labels can have over the projections.

7. Properties and Invariants of the Distance Graph

Having an initial distance graph, the next step for building a DC-checking procedure is the definition of *reduction rules* that reduce a pair $A \xrightarrow{\ell_1} B \xrightarrow{\ell_2} C$ into a single edge $A \xrightarrow{\ell_1 \circ \ell_2} C$ whose label results from the composition of the two edges. In this section, we establish two preliminary results for the definition of the reduction rules:

- We introduce an *observability tightening* process that simplifies a generalized label while leaving intact the set of viable dynamic strategies.
- We establish invariants of the edges in the initial graph as well as the sufficient conditions for these invariants to hold for any derived edge.

Instead of limiting the analysis to labels initially derived by Algorithm 2, we will typically analyse labels of the form $\ell \pm q$ where ℓ is an initial generalized label in the max or min form and q is a scalar term. This will allow the results of this section to also apply to any derived edge produced from the combination of an initial label ℓ with a scalar term q (with few restrictions on q).

7.1 Observability Tightening

The following results exploit the sensitivity of dynamic strategies to observability. They emphasize the asymmetry of dynamic controllability with respect to the direction of time (in contrast to weak and strong controllability). Here we establish the results for the initial labels (before applying any reduction rules). We will later extend the reasoning to derived labels.

The intuition behind observability tightening is as follows: if a constraint involves an event that cannot be observed when it is needed during execution, then a dynamic strategy must instead satisfy a suitably tightened constraint that is observable. For example, suppose there is a deadline of $\max(u, B - 5)$ for some timepoint Z, where u is the earliest time that the observable B can occur. If the time of occurrence of B is not known until it is observed, then it would be unsafe to wait until after u to execute Z, since by the time B is observed, the $B - 5$ bound has already passed. Thus, the deadline should be tightened to u. On the other hand, suppose there is a release time (lower bound) of $\min(v, B - 5)$ for Z, where v is the latest time that the observable B can occur. In this case, we should tighten the release time to $\min(v, B)$ to take advantage of an earlier occurrence of B since this satisfies the lower bound. These intuitions formed the basis for the reductions in the earlier dynamic controllability work (Morris et al., 2001).

7.1.1 Tightening of Edges to Enforce Observability

The following lemmas exploit this type of reasoning and make it precise. Although the basic idea is clear, a rigorous proof of the results in this setting is nontrivial, as we see below. This is due to the possibility of shared micro-links between eyes, which prevents us from simply reasoning about each eye separately.

As context for the lemmas, we suppose Z is an executable timepoint and X is the root of some hidden group for which $\{Y_i\}$ is the indexed set of leaf eyes.

Lemma 8 (Min observability tightening)**.** *Suppose a viable dynamic strategy satisfies* $Z - X \geq (-\ell_{\text{hi+}}(E)) - q$ *for all projections, where*

$$\ell_{\text{hi+}}(E) = -\min(v_0, \min_j(\dot{Y}_j + v_j))$$

is an initial label and $0 \leq q < v_0$*. Then the strategy must also satisfy*

$$Z - X \geq \min(v_0 - q, \min_j(\dot{Y}_j + \max(0, v_j - q)))$$

Proof. We are assuming $Z - X \geq (-\ell_{\text{hi+}}(E)) - q$, that is, $Z - X \geq \min(v_0 - q, \max_i(\dot{Y}_i + v_i - q))$, for all projections.

To simplify notation, we set $q_i = v_i - q$. Then $q_0 \geq 0$, and $Z - X \geq \min(q_0, \min_i(\dot{Y}_i + q_i))$ for all projections. We need to show this implies

$$Z - X \geq \min(q_0, \min_i(\dot{Y}_i + q_i^+))$$

for all projections, where $q_i^+ = \max(0, q_i)$.

Our approach will be to show that if there is a projection that does not satisfy the condition, then there is a dynamically indistinguishable projection with a contradictory property. This establishes that the condition is satisfied for all projections.

Suppose P is a projection that does not satisfy the condition. Let Z^P be the time assigned to Z in P by the viable dynamic strategy. Then

$$\min(q_0, \min_i(\dot{Y}_i + q_i)) \leq Z^P - X < \min(q_0, \min_i(\dot{Y}_i + q_i^+))$$

in P.[6]

Note that dropping terms from a min expression will not decrease its value. Thus,

$$Z^P - X < \min(q_0, \min_{q_i \geq 0}(\dot{Y}_i + q_i^+)) = \min(q_0, \min_{q_i \geq 0}(\dot{Y}_i + q_i))$$

and

$$Z^P - X < \min(q_0, \min_{q_i < 0}(\dot{Y}_i + q_i^+)) = \min_{q_i < 0}(\dot{Y}_i).$$

Let $\mathcal{E}$ be the subset of Eyes(G) defined by $\mathcal{E} = \{Y_i : Z^P - X < \dot{Y}_i + q_i \text{ in } P\}$. Thus, $\mathcal{E}$ includes the root eye $X = Y_0$ and is a superset of $\{Y_i : q_i \geq 0\}$ by our assumption. Since $Z^P - X < \min_{q_i<0}(\dot{Y}_i)$, the Z^P value is assigned before the Y_i not in $\mathcal{E}$ have been observed.

6. To minimize clutter, only Z^P, which is referenced in a wider context, is superscripted with P.

By lemma 7 part (a), there is another projection P' where $\dot{Y}_i$ is unchanged for $Y_i \in \mathcal{E}$ and $\mathcal{E}$ is UB active. Let $M = \min_{Y_i \in \mathcal{E}}(\dot{Y}_i + q_i)$. (Note that the value of M is the same in P' and P.) Thus, $Z^P - X < M \leq \dot{Y}_i + q_i$ in P' for all i, including the Y_i not in $\mathcal{E}$ (since $\mathcal{E}$ is UB active). However, $\dot{Y}_i + q_i \leq Z^P - X$ in P for the Y_i not in $\mathcal{E}$ (by the definition of $\mathcal{E}$). Thus, $\dot{Y}_i$ is increased in the transition from P to P' for $Y_i \notin \mathcal{E}$ (and is unchanged otherwise). Consequently, at time Z^P, the observables that are in the future in P are also in the future in P', and those in the past are unchanged.

It follows that the pre-history at time Z^P is the same in P' as it is in P. Since the strategy is dynamic, therefore $Z^{P'} = Z^P$. Thus, $Z^{P'} - X < \dot{Y}_i + q_i$ in P' for all i, but that violates the constraint that $Z - X \geq \min(q_0, \min_i(\dot{Y}_i + q_i))$ for all projections. This contradiction establishes the result. □

Lemma 9 (Max observability tightening)**.** *Suppose a viable dynamic strategy satisfies* $Z - X \leq \ell_{\text{lo+}}(E) - q$ *for all projections, where*

$$\ell_{\text{lo+}}(E)) = \max(u_0, \max_i(\dot{Y}_i + u_i))$$

is an initial label and $0 \leq q \leq u_0$*. Then it must also satisfy*

$$Z - X \leq \max(u_0 - q, \max_{u_i \geq q}(\dot{Y}_i + u_i - q))$$

(i.e., the terms where $(u_i - q)$ *is negative may be dropped).*

Proof. We are assuming $Z - X \leq \ell_{\text{lo+}}(E) - q$, that is, $Z - X \leq \max(u_0 - q, \max_i(\dot{Y}_i + u_i - q))$, for all projections. The proof has similar steps to lemma 8 but there is an added complication, as we will see.

To simplify notation, we set $q_i = u_i - q$. Then $q_0 \geq 0$, and $Z - X \leq \max(q_0, \max_i(\dot{Y}_i + q_i))$ for all projections. We need to show this implies

$$Z - X \leq \max(q_0, \max_{q_i \geq 0}(\dot{Y}_i + q_i))$$

for all projections.

Our approach will be to show that if there is a projection that does not satisfy the consequent, then there is a dynamically indistinguishable projection with a contradictory property. This establishes that the condition is satisfied for all projections.

Suppose P is a projection that does not satisfy the condition. Let Z^P be the time assigned to Z in P by the viable dynamic strategy. Then

$$\max(q_0, \max_{q_i \geq 0}(\dot{Y}_i + q_i)) < Z^P - X \leq \max(q_0, \max_i(\dot{Y}_i + q_i))$$

in P. (Again, to minimize clutter, only Z^P is superscripted.)

Let $\mathcal{E}$ be the subset of Eyes(G) defined by $\mathcal{E} = \{Y_i : \dot{Y}_i + q_i < Z^P - X \text{ in } P\}$. Thus, $\mathcal{E}$ includes the root eye $X = Y_0$ and is a superset of $\{Y_i : q_i \geq 0\}$ by our assumption. Note also that $Z^P - X \leq \dot{Y}_i + q_i < \dot{Y}_i$ in P for any Y_i not in $\mathcal{E}$. Thus, Z^P is assigned before these Y_i have been observed. Without loss of generality, we may assume that $\mathcal{E}$ has a maximum size among projections P that do not satisfy the condition.

By lemma 7 part (b), there is another projection P' where $\dot{Y}_i$ is unchanged for $Y_i \in \mathcal{E}$ and $\mathcal{E}$ is LB active for E. Let $M = \max_{Y_i \in \mathcal{E}}(\dot{Y}_i + q_i)$. (Note that the value of M is the same in P' and P.) Thus, $\dot{Y}_i + q_i \leq M < Z^P - X$ in P' for all Y_i, including the Y_i not in $\mathcal{E}$. However, $Z^P - X \leq \dot{Y}_i + q_i$ in P for the Y_i not in $\mathcal{E}$. Thus, $\dot{Y}_i$ is reduced in the transition from P to P' for $Y_i \notin \mathcal{E}$ (and is unchanged otherwise).

Since the $\dot{Y}_i$ are reduced, rather than increased as in the proof of lemma 8, we require additional work to obtain a suitable projection indistinguishable from P.

Note that the projections of a POSTNU form a convex set. Thus, we can form a convex linear combination $P'' = \mu * P' + (1 - \mu) * P$ for any value μ such that $0 \leq \mu \leq 1$, where any contingent link that has duration d' in P' and duration d in P would have duration $d'' = \mu * d' + (1 - \mu) * d$ in P''. (Note that $d' = d$ implies $d'' = d' = d$.)

As μ transitions from 0 to 1, the intermediate projection P'' will transition between the properties of P and those of P'. Set $y_q = \min_{Y_i \notin \mathcal{E}}(\dot{Y}_i + q_i)$ and $y = \min_{Y_i \notin \mathcal{E}}(\dot{Y}_i)$. Note that $y_q < y$ (since the q_i are negative). Then, as μ increases, both y_q and y increase towards $Z^P - X$, and y_q eventually just passes it while y remains smaller. Thus, we can choose μ and hence P'' such that: (1) $Z^P - X < \dot{Y}_i$ for all Y_i not in $\mathcal{E}$; and (2) $\dot{Y}_i + q_i < Z^P - X$ for some Y_i not in $\mathcal{E}$.

By (1) it follows that the pre-history at time Z^P is the same in P'' as it is in P. Since the strategy is dynamic, therefore $Z^{P''} = Z^P$. With respect to (2), note that $\dot{Y}_i + q_i < Z^P - X$ cannot be true for all Y_i not in $\mathcal{E}$, since that would violate the constraint that $Z - X \leq \max(u_0 - q, \max_i(\dot{Y}_i + u_i - q))$ for all projections. It follows that P'' would provide a counterexample to the lemma where $\mathcal{E}$ has a larger size than in P, which violates the maximum assumption. This contradiction establishes the result. □

Example 6. *Consider once again Example 3. Composing the $X \leftarrow E_{hi+}$ and $E_{hi+} \leftarrow Z$ edges, we can infer for the $X \leftarrow Z$ edge a label of $-\min(5, \dot{Y} - 5)$. Observability tightening would then allow us to rewrite that as $-\min(5, \dot{Y})$ without loss of any dynamic viable strategies. Observability tightening is utilized in the reduction rule phase, discussed below, where edges are composed.*

Remark The observability tightening lemmas may also be applied where Z is an observable timepoint. To see this, note that Z could be paired with an executable Z' constrained to be simultaneous with it, in which case observability tightening of Z' transfers to Z.[7]

7.1.2 Observability Tightening Notation

We may use the terms "min observability tightening" (MINOT) and "max observability tightening," (MAXOT) respectively, for the types of observability tightening sanctioned by lemmas 8 and 9, respectively.

Given max and min labels $\ell_1 = \max(u_0, \max_i(\dot{Y}_i + u_i))$ and $\ell_2 = -\min(v_0, \min_j(\dot{W}_j + v_j))$ respectively, define

$$\text{MAXOT}(\ell_1) = \max(u_0, \max_{u_i \geq 0}(\dot{Y}_i + u_i))$$
$$\text{MINOT}(\ell_2) = -\min(v_0, \min_j(\dot{W}_j + \max(v_j, 0))).$$

7. This relies on the "instantaneous reaction" aspect of the POSTNU definition.

Notice that the MAXOT and MINOT representations of derived labels have the following elementary properties:

$$\begin{aligned}
&\text{MAXOT}[\ \text{MAXOT}(\ell - q_1) - q_2\ ] = \text{MAXOT}(\ell - q_1 - q_2)\\
&\text{MINOT}[\ \text{MINOT}(\ell + q_1) + q_2\ ] = \text{MINOT}(\ell + q_1 + q_2)
\end{aligned}$$

$$\begin{aligned}
q_1 \geq q_2 &\implies \text{MAXOT}(\ell - q_1) \leq \text{MAXOT}(\ell - q_2)\\
q_1 \leq q_2 &\implies \text{MINOT}(\ell + q_1) \leq \text{MINOT}(\ell + q_2)
\end{aligned}$$

for labels ℓ of the relevant max/min form.

7.2 Edge Invariants

In this subsection, we study some of the invariants of the edges in the generalized distance graph. For both upper and lower case edges, we state two lemmas that are trivially correct in the original graph. As corollaries of each lemma, we deduce a set of invariants that hold for each kind of edge. We will see that the lemmas, and thus their corollaries, still hold for any edge derived by the reduction rules of the next section.

7.2.1 Upper Case Edges

Lemma 10. *For any upper case edge* $\mathrm{B} \to \mathrm{A}$ *in the generalized distance graph, there is a lower case edge* $\mathrm{C} \xrightarrow{\mathrm{l_{orig}}} \mathrm{A}$ *in the original graph constructed by Algorithm 2 such that the label of* $\mathrm{B} \to \mathrm{A}$ *can be expressed as*

$$\mathrm{B} \xrightarrow{\mathrm{MINOT(l_{orig}+q)}} \mathrm{A}$$

where $l_{orig} = -\min(v'_0, \min_j(\dot{W}_j + v'_j))$, *and* $v'_0 > q \geq 0$.

This is originally true as the upper case label l_{orig} of an edge $\mathrm{C} \to \mathrm{A}$ in the original graph can be expressed as $\text{MINOT}(l_{orig})$ (Lemma 8). We will see that any derived upper case edge also fulfills this property.

As a corollary of Lemma 10 we can state that the label of an upper case edge $\mathrm{B} \to \mathrm{A}$ has the form

$$-\min(v_0, \min_j(\dot{W}_j + v_j))$$

with invariants:

1. The label can be expressed as

   $$-\min(v'_0 - q, \min_j(\dot{W}_j + \max(0, v'_j - q)))$$

   where $-\min(v'_0, \min_j(\dot{W}_j + v'_j))$ is the label of one of the original upper case edges in the labeled distance graph, as constructed by algorithm 2, and $v'_0 > q \geq 0$. This can be verified by expanding the $\text{MINOT}(l_{orig} + q)$ expression from Lemma 10.

2. $v_0 > 0$ and $\forall j \neq 0 : v_j \geq 0$. From previous invariant. Note that $v'_0 = 0$ is impossible, otherwise the original edge would be an ordinary requirement edge with label 0. Also implies $\dot{W}_j + v_j > 0$

3. The edge is negative (in all projections). Implied by the previous invariant.

4. $v_0 \geq v_j$. Initially true (Lemma 5) and then implied by the first invariant.

5. The target timepoint A of this edge is also the root observable timepoint for each $\dot{W}_j$ in the label. Initially true (since the W_j are in the same hidden group), and maintained by Lemma 10.

7.2.2 Lower Case Edges

Lemma 11. *For any lower case edge* $\mathrm{A} \to \mathrm{B}$ *in the generalized distance graph, there is a lower case edge* $\mathrm{A} \xrightarrow{\mathrm{l_{orig}}} \mathrm{C}$ *in the original graph constructed by Algorithm 2 such that the label of* $\mathrm{A} \to \mathrm{B}$ *can be expressed as*

$$\mathrm{A} \xrightarrow{\mathrm{MAXOT(l_{orig} - q)}} \mathrm{B}$$

where $l_{orig} = \max(u'_0, \max_i(\dot{Y}_i + u'_i)$*, and* $u'_0 \geq q \geq 0$*.*

This is originally true as the lower case label l_{orig} of an edge $\mathrm{A} \to \mathrm{C}$ in the original graph can be expressed as $\textsc{maxot}(l_{orig})$ (Lemma 9). We will see that any derived lower case edge also fulfills this property.

As a corollary of Lemma 11, we can state that the label of a lower case edge $\mathrm{A} \to \mathrm{B}$ has the form

$$\max(u_0, \max_i(\dot{Y}_i + u_i))$$

with invariants:

1. The label can be expressed as

$$\max(u'_0 - q, \max_{u'_i \geq q}(\dot{Y}_i + u'_i - q))$$

   where $\max(u'_0, \max_i(\dot{Y}_i + u'_i))$ is the label of one of the original lower case edges, as constructed by algorithm 2, and $u'_0 \geq q \geq 0$. This can be verified by expanding the $\textsc{maxot}(l_{orig} - q)$ expression of Lemma 11.

2. $\forall i : u_i \geq 0$ (includes $i = 0$). From previous invariant. Also implies $\dot{Y}_i + u_i > 0$.

3. The edge is non-negative (in all projections). Implied by the previous invariant.

4. $u_0 \geq u_j$. Initially true (Lemma 5) and then implied by the first invariant.

5. The source timepoint A of this edge is also the root observable timepoint for each $\dot{Y}_i$ in the label. Initially true (since the Y_i are in the same hidden group) and maintained by Lemma 11.

7.2.3 Scalar Terms in Generalized Labels

It is convenient to refer to the u_0 and v_0 values as the *scalar* terms in the labels, noted $scalar(\ell)$ for any label ℓ. Notice that MAXOT and MINOT do not involve any modification of the scalar terms. We may regard the scalar as preserving a record of the q offset to the initial label.

Assuming the invariants hold, we have an extension of Lemma 6 and its $P_{\max}$ and $P_{\min}$ special projections to derived labels.

Lemma 12. *The derived label* $-\min(v_0, \min_j(\dot{W}_j + v_j))$ *reduces to* $-v_0$ *in the* P_{max} *special projection. The derived label* $\max(u_0, \max_i(\dot{Y}_i + u_i))$ *reduces to* u_0 *in the* P_{min} *special projection.*

Proof. First, note that the q modification applied uniformly to all terms does not disturb the special projections property, which holds for the initial labels. Second, if $v_0 = \min(v_0, \min_j(\dot{W}_j + v_j))$ then

$$v_0 \leq \min_j(\dot{W}_j + v_j) \leq \min_j(\dot{W}_j + \max(v_j, 0))$$

so the MINOT operation does not affect the property. Similarly, if $u_0 = \max(u_0, \max_i(\dot{Y}_i + u_i))$ then

$$u_0 \geq \max_i(\dot{Y}_i + u_i) \geq \max_{u_i \geq 0}(\dot{Y}_i + u_i)$$

so the MAXOT operation also preserves the property. □

8. Generalized Reduction Rules

The reduction rules that we will introduce can be interpreted as generalizations of the "classical" STNU reduction rules. However, it is useful to view them in a slightly different way that makes more sense in the generalized context. In particular, each reduction rule application incorporates two separate types of transformation:

(1) the addition of a logically implied constraint that leaves the entire set of viable strategies unchanged; and

(2) an *observability tightening* step, as discussed previously, that may filter out some viable strategies, but not any dynamic viable strategies.

The second step is essentially what validates the removal of lower-case labels that is inherent in the classical Lower-Case and Cross-Case Reductions, and is also implicit in the derivation of upper-case labels. Its effect is to replace derived constraints that are not observable by minimally strengthened constraints that are observable.

The overall purpose of STNU reduction rules is to try to make every projection dispatchable, since this implies Dynamic Controllability (Morris et al., 2001; Morris, 2014; Shah et al., 2007). In an STN, dispatchability can be achieved by applying "Plus/Minus" operations that compose non-negative edges with following negative edges. The aim is to ensure the path constraints are enforced by "V-paths" where any non-negative edges follow any negative edges. If this process terminates without producing a negative cycle, then the

resulting network is dispatchable (Morris, 2016). For an STNU, the negative edges include upper-case edges not subject to Label Removal, and the non-negative edges include lower-case edges. In effect, the reductions constitute the operations needed to create V-paths, and their action is global across all of the projections.

The same approach is applicable to the generalized labels considered here. Thus, we have analogues of the Upper-Case, Lower-Case, and Cross-Case reductions that incorporate observability tightening steps. Actually, edges with ordinary numeric weights may be regarded as special cases of the generalized labels where the sets of leaf eyes are empty. (For non-negative edges, the root timepoint is the source; for negative edges, it is the target.) Thus, we only need to consider the generalized Cross-Case reduction.

The Cross-Case reduction involves a composition of a negative labeled edge (aka Upper Case) and a non-negative labeled edge (aka Lower Case), where the timepoints are either executable or observable, as follows:

$$\max(u_0, \max_i(\dot{Y}_i + u_i)) - \min(v_0, \min_j(\dot{W}_j + v_j))$$

where i and j range over suitable index sets for leaf eyes.

A special case of the Cross-Case reduction occurs when the source timepoint of the non-negative edge coincides with the destination timepoint of the negative edge so that the combination forms a self-loop. In this case we are only concerned with whether the combination is negative or non-negative. If negative, the network in not Dynamically Controllable; if non-negative, the combination is ineffectual and may be discarded. Thus, the self-loop combination, does not add a new edge to the network (but it does need to be checked, as discussed later).

Otherwise, if the source and destination are different timepoints, the derived edge will be given a derived label and added to the network. The following discussion applies to the derived labels.

The input edges to the reduction will be called the *parent* edges, and the resulting edge is the *child* edge. The u_i and v_i values for the initial edges are determined by the construction of the labeled distance graph. The derived edges will have derived u_i and v_i values computed according to the Cross-Case reduction rules. Since each reduction involves a non-negative edge and a negative edge and results in either a non-negative or negative edge, exactly one of the parents will have the same sign as the derived edge. If we iterate this relationship, we can trace back a line of same-sign ancestors that leads back to a unique edge in the initial labeled distance graph.

Certain properties of the initial labels will be maintained for the derived labels, as we will see. Thus, they will constitute invariants that are preserved by the derivation process. Since their validity essentially depends on an inductive argument, we may assume they hold for the inputs of the derivations, and it will be shown that they hold for the outputted derived labels.

Cross Case Reduction The cross case reduction requires combining two edges with max and min labels:

$$\mathrm{A} \xrightarrow{\max(u_0, \max_i(\dot{Y}_i + u_i))} \mathrm{B} \xrightarrow{-\min(v_0, \min_j(\dot{W}_j + v_j))} \mathrm{C} \tag{3}$$

into a single upper case or lower case edge A $\rightarrow$ C whose label should derive from:

$$\max(u_0, \max_i(\dot{Y}_i + u_i)) - \min(v_0, \min_j(\dot{W}_j + v_j)) \tag{4}$$

There are two cases to be considered: (1) where the root timepoints of the combining edges are different (i.e. $A \neq C$); and (2) where they are the same (i.e. $A = C$). Both cases are important for consistency checking. However, in our approach, only case (1) may add an edge to the network.

In the following, we first consider case (1) where the root timepoints are different.

8.1 Different Root Case

Here we deal with the non-same-root case (i.e. $A \neq C$ in Eq. (3)). This implies that the observables in the max label come from a different hidden group than those in the min label. Thus, there are no cross-correlations, durations of macro-links vary independently between the two groups, and we may speak of a *local projection* of a hidden group. In particular, we will utilize local versions of the $P_{\max}$ and $P_{\min}$ special projections for one of the hidden groups while the local projection for the other hidden group is unrestricted. Note that the proofs of lemma 6 and lemma 12 depend only on the properties of the local projection.

The cross-independence simplifies the analysis in one way compared to the same-root case. In particular, under certain circumstances, observability tightening can be applied for an observable in one group provided only that it is applicable in *any* local projection of the other group.

We now specify the form of the derived labels, and note how they preserve invariants. This will followed by a formal proof of soundness of the derivation rules.

The result of the

$$\max(u_0, \max_i(\dot{Y}_i + u_i)) - \min(v_0, \min_j(\dot{W}_j + v_j))$$

combination in the non-same-root case is defined as follows.

Negative Result (Upper Case): If $u_0 < v_0$, the result is

$$-\min(v_0 - u_0, \min_j(\dot{W}_j + \max(v_j - u_0, 0))).$$

Note the observation variables are all inherited from the negative parent edge. Since the target of the edge is also inherited from the negative parent, this preserves the invariant that it coincides with the root timepoint of the observables. Also note the result may be expressed as MINOT(ℓ+u_0) where ℓ is the negative parent label, preserving the representation invariant (Lemma 10).

Non-negative Result (Lower Case): If $u_0 \geq v_0$, the result is

$$\max(u_0 - v_0, \max_{u_i \geq v_0}(\dot{Y}_i + u_i - v_0))$$

Note the observation variables are now inherited from the non-negative parent edge. Since the source of the edge is also inherited from the non-negative parent, this preserves

the invariant that it coincides with the root timepoint of the observables. Also note the result may be expressed as $\textsc{maxot}(\ell - v_0)$ where ℓ is the non-negative parent label, preserving the representation invariant (Lemma 11).

The non-negative case may be viewed as a generalization of the STNU reduction cases that reduce away a lower-case edge. Here, the result is a lower-case edge that has fewer observation variables and/or smaller offsets.

Theorem 1. *The Cross Case reduction rules are sound in both the negative and non-negative result cases.*

Proof. Recall that we have $0 \leq u_i \leq u_0$ and $0 \leq v_j \leq v_0$ as invariants. In the following, we will use the symbol '$\approx$' to indicate a non-equivalent transformation step that nevertheless preserves the set of viable dynamic strategies because of observability tightening.

We are composing the max/min labels

$$\max(u_0, \max_{i>0}(\dot{Y}_i + u_i)) - \min(v_0, \min_{j>0}(\dot{W}_j + v_j))$$

corresponding to a $\mathrm{X}_1 \to \mathrm{Z} \to \mathrm{X}_2$ combination.

First suppose $u_0 < v_0$. Then the W_0 root timepoint must be scheduled before the Y_0 root timepoint. (Otherwise, the combination would not be consistent with all projections.) Consequently, W_0 must be scheduled before any of the $Y_i : i \geq 1$ are observed, and a viable dynamic strategy must satisfy the composed edge in the $P_{\min}$ local projection for the Y_i group where the max label reduces to u_0 (lemma 12). Thus, it must satisfy

$$\begin{aligned} X_2 - X_1 &\leq u_0 - \min(v_0, \min_{j>0}(\dot{W}_j + v_j)) \\ &= -\min(v_0 - u_0, \min_{j>0}(\dot{W}_j + v_j - u_0)) \\ &\approx -\min(v_0 - u_0, \min_{j>0}(\dot{W}_j + \max(v_j - u_0, 0))). \end{aligned}$$

Next suppose $u_0 \geq v_0$. Then W_0 cannot be scheduled before Y_0, and the latter must be scheduled before the $W_j : j \geq 1$ are observed. Consequently, a viable dynamic strategy must satisfy the composed edge in the $P_{\max}$ local projection for the W_j group where the min label reduces to v_0 (lemma 12). Thus, it must satisfy

$$\begin{aligned} X_2 - X_1 &\leq \max(u_0, \max_{i>0}(\dot{Y}_i + u_i)) - v_0 \\ &= \max(u_0 - v_0, \max_{i>0}(\dot{Y}_i + u_i - v_0)) \\ &\approx \max(u_0 - v_0, \max_{u_i \geq v_0}(\dot{Y}_i + u_i - v_0)) \end{aligned}$$

This establishes the soundness of the reductions in both the negative and non-negative result cases. □

The following result will be useful in a later section.

Lemma 13. *The edges derived by the reductions are at least as tight as what would be obtained by composing the input edges.*

Proof. In the non-negative result case, the reduction is equivalent to applying observability tightening to $\max(u_0, \max_i(\dot{Y}_i)) - v_0$. Since $-v_0 \leq -\min(v_0, \min_j(\dot{W}_j + v_j))$, this is at least as tight as composing the edges.

In the negative result case, the reduction is equivalent to applying observability tightening to $u_0 - \min(v_0, \min_j(\dot{W}_j + v_j))$. Since $u_0 \leq \max(u_0, \max_i(\dot{Y}_i + u_i))$, this is at least as tight as composing the edges. □

Synthetic rules Having performed this analysis, we can now reformulate the two reductions rules in terms of max/min observability tightening and scalar terms of the labels (Table 1). Recalling that an edge with a purely scalar label q can be interpreted as a lower-case label if $q \geq 0$ or as an upper-case label if $q < 0$, these two rules allow the combination of any non-negative edge with a following negative edge.

$$A \xrightarrow{\ell^+} B \xrightarrow{\ell^-} C \quad \text{is reduced to}$$

$A \xrightarrow{\text{MAXOT}[\ell^+ - scalar(\ell^-)]} \text{C}$	if $scalar(\ell^+) - scalar(\ell^-) \geq 0$
$A \xrightarrow{\text{MINOT}[\ell^- + scalar(\ell^+)]} \text{C}$	if $scalar(\ell^+) - scalar(\ell^-) < 0$

Table 1: Reduction rules, applicable where ℓ^+ is a non-negative label, ℓ^- is a negative label and $A \neq C$.

8.2 Same Root Case

Our next task in composing the max and min labels

$$\max(u_0, \max_i(\dot{Y}_i + u_i)) - \min(v_0, \min_j(\dot{W}_j + v_j))$$

is to deal with the case where the source timepoint of the edge with the max label coincides with the target of the edge with the min label (i.e. $A = C$ in Eq. (3)). We will call this the *same-root* case, since the root eye is the same for both labels. According to the invariants, this will also be the root timepoint for each of the $\dot{Y}_i$ and $\dot{W}_j$ observables.

In the same-root case, we are only really interested in determining whether the sign of the combination is negative or non-negative. If negative, then the POSTNU is not Dynamically Controllable, since a timepoint cannot precede itself. Otherwise the result derives a non-negative distance from the root timepoint to itself, which can be discarded as redundant, since a timepoint will never follow itself.

The essential issue with same-root checking is to determine whether the self-loop arising from the combined label will have a negative value in any macro-projection. However, since the observables may be correlated because of shared micro-links, this determination requires consideration of the underlying Nature STN.

The combination can be conveniently written as

$$\max_{i \geq 0, j \geq 0} (\dot{Y}_i + (u_i - v_j) - \dot{W}_j) \tag{5}$$

where $\dot{Y}_0 = \dot{W}_0 = 0$. If any of the terms in the maximization is non-negative, then the combination is also. More precisely, the condition for discarding the combination can be expressed as: for every projection, there is some i and j such that $\dot{Y}_i - \dot{W}_j \geq v_j - u_i$. This sub-problem can be stated negatively as determining whether there is no projection, which can be limited to the observables in the hidden group, such that the terms are all negative, i.e., such that:

$$\forall i, j : \dot{Y}_i - \dot{W}_j < v_j - u_i.$$

Given that $\dot{Y}$ is the shorthand for $Y - X$ where X is the root observable for Y and that all Y_i and W_j have the same root observable, the above inequalities can be rewritten as $Y_i - W_j < v_j - u_i$ which are essentially Simple Temporal constraints except that the inequalities are strict. The bounds on the micro-links may also be viewed as Simple Temporal constraints, and the problem reduces to determining whether this partially strict STN has a solution. A standard STN algorithm such as Bellman-Ford (Cormen et al., 1990) can be used to look for either an explicit negative cycle, or an implied negative cycle where the bounds sum to zero and the cycle passes through one or more of the strict edges.[8] Note that if this derived STN is consistent, then the POSTNU is NOT Dynamically Controllable. If it is not consistent, then the same-root combination is discarded.

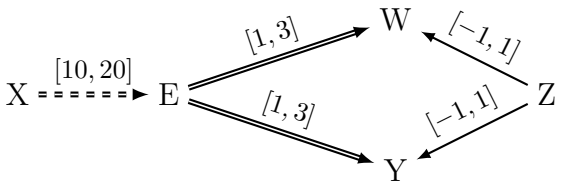
A shortcut that avoids the need for the negative restatement in many cases is if there is a *fixed* choice of i and j for which $\dot{Y}_i - \dot{W}_j \geq v_j - u_i$ for *all* projections. For example, this might be true if $Y_i = W_j$ and $u_i = v_j = 0$. In that case, we can immediately discard the same-root combination. However, if $v_j > u_i$, then the shortcut does not work for this Y_i and W_j pair.

Example 7. *In the classic STNU theory, there is an ad-hoc restriction that the Cross-Case reduction cannot be applied when the upper and lower case labels involve the same contingent timepoint. However, with the generalized labels, an ad-hoc restriction is not needed. Consider* A $\xRightarrow{[u,v]}$ B. *We have*

$$\max(u, \dot{B}) - \min(v, \dot{B}) = \max(\dot{B} - \dot{B}, \ldots) \geq 0.$$

Thus, the combination is non-negative and is discarded. The negatively stated analysis discussed above would produce constraints including $B - B < 0$ so the derived STN is inconsistent and the combination is also discarded with the more general analysis.

Example 8. *Consider the following POSTNU with an executable timepoint Z that is constrained relative to contingent timepoints Y and W in the same hidden group rooted at X, where the two macro-links share an XE microlink to the hidden timepoint E.*



8. In practice, one could simply decrement the strict bounds by a tiny amount and use standard STN methods.

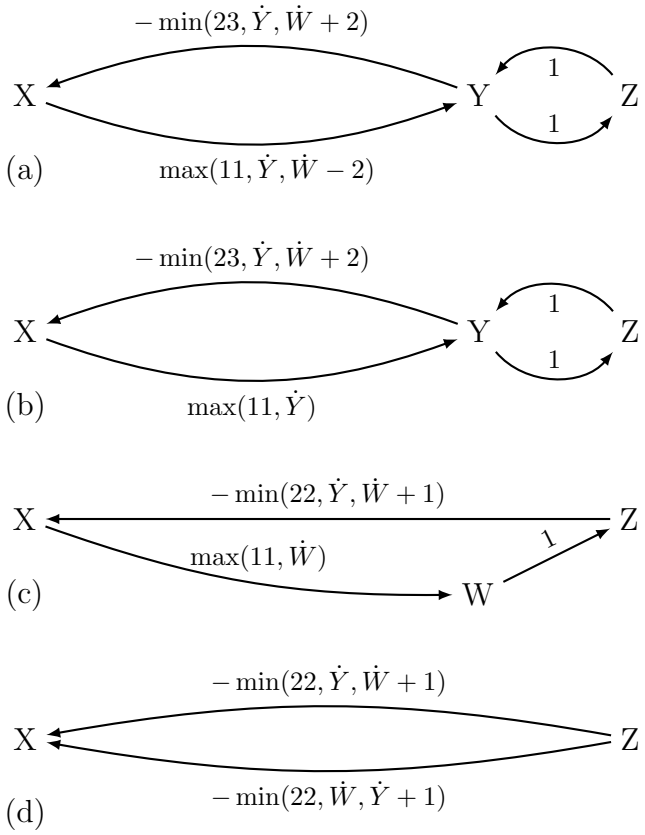


Figure 9: Labeled Edge Processing for Example 8.

Note the symmetry between Y and W in the example, which simplifies the exposition.

In Figure 9, we show the stages of processing for the labeled edges. The portion of the graph related to the Y timepoint is shown in diagrams (a) and (b). The remaining portion of the graph involving W is not shown, but is symmetrically similar with Y and W interchanged. In diagrams (c) and (d), we have combined relevant elements from the two symmetrical portions.

The initial graph produced by Algorithm 2 is shown in (a) (before observability tightening) and in (b) (after observability tightening). The XYX same-root cycle in (b) is similar to the STNU case in that it reduces to $\max(\dot{Y} - \dot{Y}, \ldots)$, which is non-negative. The $XYZYX$ path does not produce a same-root cycle because the $YZYX$ path does not lead to a tighter YX edge.

Now if we perform a reduction of the ZY and YX edges and consider the additional symmetric graph portion through W that was hidden in the (a),(b) diagrams, we see in diagram (c) that there is a further same-root cycle that passes through Z. A WZX reduction will now produce a $-\min(21, \dot{Y}, \dot{W})$ label. (Note the application of min observability tightening to the $\dot{Y} - 1$ term.) This same-root case then reduces to $\max(\dot{W} - \dot{W}, \ldots)$, which is also non-negative. Thus, the reductions do not produce a negative cycle.

It is also interesting to see (diagram (d)) that XZ obtains a $\min(22, \dot{Y}, \dot{W}+1)$ *lower bound constraint and (symetrically)* $\min(22, \dot{W}, \dot{Y}+1)$*, both of which must hold. Combining these, we get*

$$\max(\min(22, \dot{Y}, \dot{W}+1), \min(22, \dot{W}, \dot{Y}+1))$$

which simplifies to $\min(22, \dot{Y}+1, \dot{W}+1, \max(\dot{Y}, \dot{W}))$*. This is analogous to a* wait *condition in the classical STNU theory. Thus, Z must wait for execution until at least one of the following four events has occurred: 22 after X, or 1 after Y, or 1 after W, or* both *Y and W have been observed.*

9. Completeness and Algorithms

Previous sections established the soundness of the generalized reduction rules. This section considers the issue of whether they can support a complete procedure for determining dynamic controllability for POSTNU networks, and whether it has polynomial complexity. In particular, we will consider an algorithm that performs the reductions in an organized way, show it terminates, and has a polynomial complexity. Furthermore we will show that if the algorithm completes with success (i.e., without revealing an inconsistency), this implies the existence of a viable dynamic strategy for the problem, and hence it is dynamically controllable.

Given the awkwardness of the "non-same-root reduction" terminology, henceforth, we will refer to non-same-root reductions as *binary reductions* (involving two different roots). Since the "same-root" reductions only result in checks and do not produce new edges, we will continue to refer to them as same-root *checks* (rather than unary reductions).

Our approach will be to repurpose ideas and methods from previous STNU work, adapting them where necessary. Some issues arise in this regard:

1. What is the equivalent of "reduced distance" for generalized labels?
2. Is it necessary to consider "normal form" (Morris, 2014)?
3. How do we handle the same-root checking?

We deal with these issues now.

In an STNU labeled distance graph, the *reduced distance* of a path was defined to be the sum of the lengths of the edges, ignoring any labels. (Morris & Muscettola, 2005). The reductions preserved the reduced distance. For generalized labels, we will define the *scalar length* of a non-negative edge to be the scalar component of the label, while for a negative edge, it is the negation of the scalar component. For example, the scalar length of an edge with label $-\min(v_0, \min_i(\dot{Y}_i + v_i))$ is $-v_0$. We then define the *scalar distance* of a path to be the sum of the scalar lengths of the edges in the path. Note that the generalized reductions preserve the scalar distance.

In (Morris, 2006, 2014), STNU problems were converted to *normal form*, where a contingent link was broken into a "certain" portion (the minimum duration part) and the remaining "fully uncertain" portion. This was done as an intended simplification that had the effect of making a Label Removal rule assume a simpler form. However, it was later realized that this was unnecessary because the simpler form, though weaker, was nevertheless sound in the general case, and was sufficient for determining Dynamic Controllability.

Thus, the stronger, more complex, version of Label Removal, though also sound, was not useful in this context.

Interestingly, our generalized cross-case reduction implicitly incorporates a form of label removal, in that it discards the non-scalar components from one of the parent eyes. There is no separate label removal reduction, and none is needed. Thus, normal form has no relevance to the approach considered here.

In the STNU work, there is no same-root checking because reductions between upper and lower case labels from the same contingent link are not allowed. In our approach, as we have seen, STNU-like combinations (i.e., with no hidden variables) always result in non-negative self-loops, which are discarded. However, the same is not true for general POSTNU combinations, where negative self-loops may be derived. Thus, the checking described earlier is required. It is important to note, however, that the same-root combinations do not add any new edges to the network. Thus, they have no effect on the binary reduction process, which can proceed independently.

This means we can *defer* the same-root checking until after the binary reductions have completed. This may forego a possible early determination that the network is not Dynamically Controllable, but it allows the binary reduction process to more closely resemble the STNU reduction process, and any negative self-loops will still be discovered eventually when the same-root checking is done (as a post-process). However, the binary reduction process may produce a cycle of all-negative edges. If a negative self-loop or cycle of all-negative edges is produced, the POSTNU cannot be dynamically controllable. Since the reductions have been shown to preserve the set of viable dynamic strategies, such a derivation implies that this set is empty.

In this context, the hidden timepoints are only relevant to the same-root checking. For the binary reductions, we can confine our attention to the non-hidden timepoints and their edges, including the labeled edges that have been added either by the pre-processing step, or by the reductions. The graph formed by these edges will be referred to as the generalized *labeled distance graph*. Each macro-projection (hereafter called simply a projection) instantiates this labeled graph to the distance graph of an STN.

Next we address the question of how many edges can occur at any stage in the reduction process. The labeled distance graph is actually a *multigraph*, which can have multiple edges between two timepoints. For our POSTNU analysis, this can involve multiple edges with different labels. Clearly, there is no need for the reduction process to add a new labeled edge if it is implied by (i.e., strictly weaker than) an existing edge. Furthermore, an existing edge may be removed if it is implied by a newly added edge. However, the max/min labels, which involve variables and resemble vectors, may not be directly comparable. We will refer to these as *compound labels* and their edges as *compound edges*.

The following discussion provides rules for determining whether some edges are clearly implied by other edges. (The intent here is to limit the number of edges, not to provide an absolute determination of edge implication.) We will say an edge e_1 *subsumes* an edge e_2 if it implies it according to these rules. We will also say e_1 is *tighter* than e_2 if it subsumes it, but is not subsumed by it.

The key to limiting the number of non-subsumed edges arises from the invariant properties of edges, in particular, the property that a negative label can be expressed as $\text{MINOT}(\ell_1 + q)$ for some q where ℓ_1 is the label of an initial min edge, while a non-negative label can

be represented as MAXOT($\ell_2 - q$) for some initial max edge ℓ_2. Moreover, ℓ_1 and ℓ_2 are the respective same-sign ancestor edges in both cases. This means that we can use the q values to compare two compound labels that are derived from the same initial edge, and thus limit their number. For example, in Figure 9, diagram (e), while there are two compound labeled edges from Z to X, they are derived from two different initial edges.

Observe that initial edges with compound labels have distinct end-points (the compound observables). Thus, their number is limited by the number of timepoints in the initial labeled distance graph. Consequently, the number of tightest compound labels at any point in the reduction process may be limited to the number of pairs of an initial compound edge, and some other timepoint, i.e., it is $O(N^2)$, where N is the number of timepoints in the labeled distance graph (including the compound observables).

Also, it is clear that labels with ordinary numeric weights can be compared directly. Thus, the number of edges with such labels can be limited to the usual $O(N^2)$. Consequently, the total number of incomparable edges in the labeled distance graph can be limited to an $O(N^2)$ bound.

To summarize, if we follow a policy where binary reductions only add edges when they are not subsumed by existing edges, and existing edges are removed if they are subsumed by newly added edges, then the number of edges in the labeled distance graph will have an $O(N^2)$ bound.

Definition 7. *A binary reduction is* ineffective *if it would produce an edge that is subsumed by an existing edge in the labeled distance graph; otherwise it is* effective*. An existing edge is* superseded *if it is subsumed by a newly added edge.*

Thus the proposed policy is to ignore ineffective reductions and remove superseded edges.

Remark A binary reduction is effective if it produces an edge that is not subsumed by an existing *single* edge. However, the edge produced need not be the shortest path between its endpoints. For example, addition of a non-negative edge could be effective even if there is a path between the endpoints of two or more negative edges. (This ensures that determination of effectiveness is a low complexity operation.)

The following concepts will be generally useful.

Definition 8. *The labeled distance graph is* quiescent *if any further binary reductions would be ineffective.*

Note that quiescence does not necessarily imply consistency. For example, a graph consisting of a cycle of all-negative edges would be quiescent (since no binary reductions are applicable) but would not be consistent.

Definition 9. *In the labeled distance graph (possibly after some reductions), the* generation number *of a negative edge is defined inductively as follows. For an initial negative edge, the generation number is 0. Otherwise, the generation number of a negative edge equals the generation number of its negative parent incremented by 1.*

Lemma 14. *If only effective reductions add edges, the generation number of a negative edge can never exceed N, where N is the number of timepoints in the network.*

Proof. Suppose a negative edge e has generation number greater than N. Then there is some sequence of negative edges $e_1^-, \ldots, e_k^- = e$ such that e_1^- is an initial edge and, for each i, e_{i+1}^- is derived from e_i^- by a reduction with some"partner" non-negative edge e_i^+.

Now consider the "partner" sequence $e_1^+, \ldots, e_k^+$. This must form a path of non-negative edges of length greater than N. Thus, it must involve a repeated timepoint, so the path contains a loop of non-negative scalar length from some e_{i1}^+ to e_{i2}^+. However, this implies e_{i2}^- is subsumed by e_{i1}^-, so it would not have been added to the network, which is a contradiction. □

Lemma 15. *If the POSTNU labeled distance graph has a cycle of all-negative edges, or a cycle that is reducible to a cycle of all-negative edges, then the POSTNU is not Dynamic Controllable (DC).*

Proof. Suppose first there is a cycle of all-negative edges. Then there must be a simple cycle (i.e., without repeated timepoints) of all-negative edges. It follows that the negative edges in the simple cycle must have distinct target timepoints, hence distinct root timepoints, and any compound edges must belong to different hidden groups. Consequently, there will be a projection in which the edges have their scalar values. Thus, the projection will have a negative cycle, and so the POSTNU cannot be DC.

Since the reductions are sound, we have a similar conclusion if there is a cycle that is reducible to an all-negative cycle. □

In light of the importance of reducibility to an all-negative cycle, we will give this property a name.

Definition 10. *A POSTNU labeled distance graph has an* essential negative cycle *if it has a cycle that is reducible to a cycle of all negative edges.*

Remark In contrast to an essential negative cycle, a cycle simply with a negative scalar distance does not in general exclude Dynamic Controllability for a POSTNU. For example, the same-root cycle A $\xrightarrow{\max(5,\dot{Y})}$ B $\xrightarrow{-\min(10,\dot{Y})}$ A has negative scalar distance but the same-root check succeeds.

9.1 Naïve Algorithm

We now consider an algorithm that performs the reductions in an organized way. One of our main goals in this paper is to show the POSTNU problem is polynomial, which answers a question posed by (Bhargava & Williams, 2019). Accordingly, the algorithm is chosen for conceptual simplicity rather than efficiency. In a nutshell, the algorithm applies the binary reductions repeatedly in a breadth-first manner. The process terminates if quiescence is reached. We will show this happens after a polynomially bounded number of steps if the POSTNU is Dynamically Controllable.

At that point, if the network has no all-negative cycles, and if the same-root checking succeeds, then the POSTNU is classified as Dynamically Controllable (DC). Otherwise, it is classified as not DC.

The following procedure extends the breadth-first expansion by one level, that is, it applies the binary reductions in all possible effective ways to the edges in the input graph. It also removes superseded edges. We call this a *round of reductions.*

Algorithm 3 A single round of reduction in the POSTNU generalized labeled graph.

```
procedure Do-Round-Of-Reductions(Γ)
    Γ' ← Γ
    for all edge-pairs e1, e2 ∈ Γ do
        if e1, e2 are binary reducible to e then
            if e is not subsumed in Γ' then
                Γ' ← (Add e to Γ')
                if e' ≠ e ∈ Γ' is superseded by e then
                    Γ' ← (Remove e' from Γ')
    Γ ← Γ'
```

9.1.1 Termination

In the following, N is the number of timepoints in the initial labeled distance graph. Recall that *quiescent* means that any further binary reductions would produce edges with labels that are subsumed by the labels of existing edges. The following lemma limits the number of reduction rounds needed to reach quiescence. Recall that an essential negative cycle is a cycle that is reducible to a cycle of all negative edges.

Lemma 16. *(a) If a POSTNU is Dynamically Controllable (DC), then quiescence is achieved in at most $N^2 + N$ rounds of reductions. (b) If a POSTNU distance graph does not contain an essential negative cycle, then quiescence is achieved in at most $N^2 + N$ rounds of reductions.*

Proof. Suppose e is a non-negative edge that is produced after any number of rounds of reductions. We will show that e will be produced in at most N^2 rounds of reductions. The following reasoning has a lot in common with that used in (Morris, 2006) to prove the Nesting Lemma. We prove a similar Nesting property here, but it is applied to negative edges whereas (Morris, 2006) applies it to non-negative edges.

If we unroll (play in reverse) the binary reductions that gave rise to e, the result corresponds to some ancestral path $e_1, \ldots, e_k$ in the initial labelled distance graph. Consider any negative e_i in this path. Since e_i must be eventually reduced away in order for the path to reduce to the non-negative e, there must be some e_j preceding e_i in the path such that the scalar distance from e_j to e_i, inclusive, is non-negative. Without loss of generality, we assume e_j is the closest such edge, i.e., such that the path from e_j to e_i has the fewest number of edges.

Using terminology analogous to "moat edge" in (Morris, 2006), we will call e_j a *wall edge* for e_i. We can pair each negative edge in the unrolled path with a corresponding wall edge. We will call the inclusive path between the two edges the *cancel subpath* for the negative edge. This has the *prefix/postfix* property (Morris, 2006) that every proper prefix is non-negative, and every proper postfix is negative. It follows that if two cancel subpaths intersect, one must be contained in the other. (Otherwise, the intersected subpath would be both negative and non-negative, which is a contradiction.) Thus, the cancel subpaths are either nested or disjoint. It may be helpful to visualize the nested subpaths as being

surrounded by parentheses, as in this example

$$(\mathrm{A} \xrightarrow{+2} (\mathrm{B} \xrightarrow{+5} \mathrm{C} \xrightarrow{-3} \mathrm{D}) \xrightarrow{+7} \mathrm{E} \xrightarrow{-10} \mathrm{F})$$

where the reductions in the inner nested subpaths occur before those of the outer subpaths can be completed.

Now consider an innermost cancel subpath within the nesting structure. Observe that it must consist of one or more non-negative edges and a single negative edge. (Otherwise, it would not be innermost.) More generally, each cancel subpath has one or more non-negative edges, or nested cancel subpaths that reduce to non-negative edges, and a single negative edge.

It now follows that in an innermost cancel subpath, the subpath of non-negative edges cannot have length greater than N. Otherwise the reductions would have produced a negative edge with generation number greater than N, contrary to lemma 14. This means that N rounds of reductions will be sufficient to reduce away the innermost subpaths, i.e., to effectively decrement the depth of nesting by one level. In that case, the next to innermost level becomes the innermost level and similar reasoning can be applied.

Next we claim that the initial depth of nesting cannot be greater than N. If the depth of nesting were greater than N, an end timepoint would repeat, as illustrated here, where the numbers over the edges should be interpreted as scalar lengths rather than numeric weights.

$$(\ldots(\ldots(\ldots(\ldots \xrightarrow{-5} \mathrm{A})\ldots(\ldots)\ldots \xrightarrow{-5} \mathrm{B})\ldots \xrightarrow{-5} \mathrm{A})\ldots).$$

Note that the scalar distance from the end timepoint of each nested group to the end timepoint of the next outer nested group must be negative (by the prefix/postfix property). Thus, the subpath between the repetitions constitutes a cycle with negative scalar distance. Note that a subset of the reductions used to derive e will suffice to reduce this to an all-negative cycle; hence it is an essential negative cycle. Thus the POSTNU is not DC by lemma 15, contrary to our assumption. It follows that the depth of nesting is not greater than N, and consequently all of the nested structure can be reduced to e within N^2 rounds of reductions.

It follows that any non-negative edge that will ever be added will have been added at that point, and any further edges that might be added will all be negative. From then on, the reductions will involve a fixed set of non-negative "partners," and will proceed independently of each other. That is, the negative edges added in subsequent rounds, except for the first such round, will each have a negative parent that was itself added in the previous round. Thus, the generation numbers of added edges will steadily increase in the subsequent rounds. Since the generation numbers cannot exceed N (lemma 14), an additional N rounds of reduction will suffice to complete the addition of all the remaining negative edges. Thus, all the edges that will ever be added will be added within $N^2 + N$ rounds of reductions.

For the (b) part, note that the proof of (a) only needs the weaker condition of no essential negative cycle. □

The following is then the full simple algorithm that we propose to classify the Dynamic Controllability (DC) status of a POSTNU.

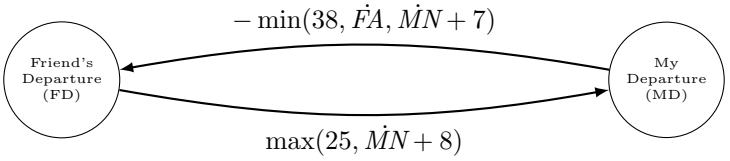
Algorithm 4 Naïve implementation of Dynamic Controllability checking of the generalized labeled graph of a POSTNU.

```
procedure SIMPLE-POSTNU-DC(Γ)
    i ← 0
    while i < N² + N and Γ not quiescent do
        DO-ROUND-OF-REDUCTIONS(Γ)
        i ← i + 1
    if Γ not quiescent then
        return false
    if Γ has all-negative cycle then
        return false
    if same-root checks fail in Γ then
        return false
    return true
```

Note that the algorithm only checks for cycles of all-negative edges, not for arbitrary negative cycles. We will see that this is the correct check for inconsistency (in combination with same-root checking) when the network is quiescent.

The termination bound suggests the algorithm is polynomial. However, we defer a detailed analysis of the complexity until we finish dealing with the completeness issue.

Example 9 (running example)**.** *Below is the result of applying the algorithm on our running example where we should meet a friend at a restaurant (Figure 3). The algorithm is run on the generalized distance graph of Figure 8 and we display the edges that are constraining for the MD node (my own departure), as derived by the algorithm.*



The graph shows the constraints that were derived between the MD controllable timepoint that depicts my departure time and the hidden group of my friend's activities rooted at the FD timepoint.

In particular the derived MD $\longrightarrow$ FD edge specifies the minimal delay constraint that MD should satisfy with respect to the observable events of my friend's hidden group: it should be executed either 38 minutes after the friend's departure (FD), after its arrival (FA) or 7 minutes after getting the notification that he leaved the post office (MN).

Likewise, the derived FD $\longrightarrow$ MD edge specifies a deadline on the MD event: it should be executed either at most 25 minutes after the friend's departure (FD) or at most 8 minutes after getting the notification (MN).

These two edges constitute a same-root case *that would pass the controllability check, resulting in the network being classified as dynamically controllable.*

9.2 Viable Dynamic Strategy

Our main result in this section will be to show that reaching quiescence without detecting an inconsistency (i.e., an all-negative cycle or a same-root checking failure) implies that the POSTNU is Dynamically Controllable. This will establish the completeness of the reduction algorithm.

We will do this by proving the existence of a viable dynamic strategy. Recall that a strategy is a mapping from projections to schedules. Each of the projections has an earliest-time schedule. We will show that, after reaching quiescence without detecting an inconsistency, the mapping from projections to their earliest-time schedules constitutes a viable dynamic strategy.

The following definitions and lemmas will be helpful for this purpose. Loosely speaking, an STN distance-graph is plus/minus closed if it is closed under an operation that composes consecutive plus/minus edge pairs. More formally, we have the following.

Definition 11. *An STN distance-graph is* plus/minus closed *if for every pair of consecutive edges* $\mathrm{A} \xrightarrow{x} \mathrm{B} \xrightarrow{-y} \mathrm{C}$ *such that x is non-negative, $-y$ is negative, and $A \neq C$, there is an edge* $\mathrm{A} \xrightarrow{z} \mathrm{C}$ *such that $z \leq x - y$.*

The following lemma shows that the binary reduction process, in effect, performs this closure operation simultaneously on all of the projections.

Lemma 17. *When quiescence is reached for the POSTNU distance graph, all its projections are plus/minus closed.*

Proof. Consider plus/minus consecutive edges $\mathrm{A} \xrightarrow{x} \mathrm{B} \xrightarrow{-y} \mathrm{C}$ in the distance graph of a projection, where $A \neq C$. There will be a corresponding plus/minus edge pair in the POSTNU distance graph, with labels (possibly compound) that instantiate to the x and $-y$ weights in the projection. Then, lemma 13 implies the quiescent POSTNU distance graph will contain an AC edge whose label is at least as tight as that resulting from composing the plus/minus edge pair. Thus, the projection will have an instantiated $\mathrm{A} \xrightarrow{z} \mathrm{C}$ edge such that $z \leq x - y$. □

The following result shows the DC algorithm also effectively applies consistency checking simultaneously to all the projections.

Lemma 18. *If quiescence is reached for the POSTNU distance graph without detecting an all-negative cycle or a same-root checking failure, then all its projections are consistent as STNs.*

Proof. Suppose a projection STN is inconsistent. Then it has a negative cycle and hence has a negative cycle with a minimal number of edges. Note that in an STN such a minimal cycle must be simple (no repeated timepoints).

First suppose the cycle has two edges. If they are both negative, then the corresponding edges in the POSTNU distance graph will also be negative, and this will constitute an all-negative cycle that will be detected by the DC checking algorithm.

Otherwise one edge is negative and the other non-negative. Consider the corresponding edges in the POSTNU distance graph, which will also be negative and non-negative, respectively. Since the root for a non-negative edge is at its start timepoint, while the root for a

negative edge is at its end timepoint, this will be a same-root case. The same-root checking is designed to determine whether there is a projection such that the combination self-loop is negative. Since that is the case here, the same-root checking will fail.

Now suppose the minimal cycle has three or more edges. If they are all negative, then the inconsistency will be detected. Otherwise, there must be a non-negative edge that is followed by a negative edge. But then plus/minus closure implies there is a negative cycle with fewer edges, which contradicts the minimality assumption.

Thus, the assumption of inconsistency implies one of the DC tests will fail, which establishes the result. □

To summarize, if the algorithm successfully completes (without detecting an inconsistency), every projection will be both consistent and plus/minus closed. This has further consequences using concepts analogous to those in (Morris, 2014, 2016) that relate dynamic controllability of STNUs to dispatchability of their projections. We now consider these consequences.

Definition 12. *Given an STN distance graph, a V-path is a path that consists of zero or more negative edges followed by zero or more non-negative edges.*

It is immediate that every subpath of a V-path is itself a V-path.

In the dispatchability work (Muscettola, Morris, et al., 1998), it is noted that negative edges propagate lower bounds in their reverse direction, while non-negative edges propagate upper bounds in the forward direction. Thus, negative edges may be thought of as pointing backward in time, and non-negative edges as pointing forward in time. If time is visualized as advancing away from the viewer, then a V-path may be visualized as having a shape like the letter V, hence the name. Intuitively, a V-path corresponds to a path through the past, and a V-path constraint between two timepoints is satisfied if it is satisfied with respect to events in the past. This is the intuition behind the connection to a dynamic strategy. The following formal results capture this intuition.

Definition 13. *A consistent STN has the* V-path property *with respect to its distance graph if for every path between two timepoints A and B there is a shortest path between A and B that is a V-path.*

Loosely speaking, the V-path property means that every path constraint is "enforced" by a V-path. Thus, a partial solution that locally satisfies V-path constraints will locally satisfy the minimal graph of (Dechter et al., 1991) and will have an extension to a complete solution, which is the essence of dispatchability. It is shown in (Morris, 2016) that the V-path property is, in fact, equivalent to dispatchability for a consistent STN. However, for our results, it suffices to work with the V-path property directly, using the following result.

Lemma 19. *If a consistent STN is plus/minus closed then it has the V-path property.*

Proof. Among all the shortest paths between two timepoints A and B, consider a path p with a minimal number of edges. Since the STN is consistent, this must be a simple path (no repeated timepoints). We claim it will also be a V-path. Suppose otherwise. Then there must be a non-negative edge e_1 in p that comes before a negative edge e_2. Without loss of generality, we may assume that e_2 immediately follows e_1 like this: $\mathrm{C} \xrightarrow{e_1} \mathrm{D} \xrightarrow{e_2} \mathrm{E}$. By

plus/minus closure, there will then be an edge from C to E that is at least as tight as the composition of e_1 and e_2. But this implies a shortest path between A and B with fewer edges than p, which contradicts the minimality of p. This establishes the result. □

It follows from the lemma that successful completion of the POSTNU Dynamic Controllability checking algorithm results in a labeled distance graph such that every projection has the V-path property.

Next we consider the concept of an earliest time schedule for a consistent STN. This has particular significance in the case of STNs that have the V-path property.

Definition 14. *(Dechter et al., 1991) The* earliest-time schedule *S for an STN is obtained as follows. An initial node $\odot$ is added, and for each node X in the original STN we add an edge from X to $\odot$ of weight 0. Then $S(X)$ is defined as $-D(X)$ where $D(X)$ is the shortest-path distance from X to $\odot$.*

The node $\odot$ and its edges are merely used for convenience in defining S; they should not be considered part of the STN (although we may mention them in reasoning about the properties of S).

Note that since every node has a direct edge to $\odot$ of length 0, $S(X) \geq 0$ for all X. The following result has independent interest as well as being useful here.

Theorem 2. *For a consistent STN that has the V-path property, and the earliest-time schedule S, every timepoint X either satisfies $S(X) = 0$ or there is a shortest path of all negative edges from X to some node Y such that $S(Y) = 0$.*

Proof. Suppose $S(X) > 0$. Then $D(X) < 0$ and there is a shortest path from X to $\odot$ that is negative. Since the STN has the V-path property, we can assume that the subpath up to (but not including) the final edge to $\odot$ is a V-path. Note that the final edge to $\odot$ has a zero weight. Thus, the start timepoint Y of the final edge must satisfy $S(Y) = 0$. Without loss of generality, we may assume that Y is the node with fewest edges between X and Y such that $S(Y) = 0$. Then $D(Z) < 0$ for every intermediate node Z on the path. Since the path from X to Y is a V-path (and recall that every subpath of a V-path is itself a V-path), it follows that every edge on that path has a negative weight. □

This remarkable result implies that for a consistent STN that has the V-path property, the non-negative edges are irrelevant as far as the earliest time schedule is concerned. It is only if the execution deviates from the earliest time schedule that the non-negative edges (and deadlines) matter.

Returning our attention to the POSTNU problem, we are now in a position to define our candidate strategy. We will say a POSTNU distance graph is *verified* if it is quiescent with respect to the binary reductions, and if it has no all-negative cycles or same-root checking failures.

Definition 15. *Given a verified distance graph for a POSTNU, the* earliest time *execution strategy $\breve{\mathcal{S}}$ (pronounced "S-breve") is defined for any projection p by $\breve{\mathcal{S}}(p) = \breve{S}_p$ where $\breve{S}_p$ is the earliest time schedule for p.*

This leads us to our main theorem.

Theorem 3 (Completeness)**.** *If the generalized reductions produce a verified distance graph, then the POSTNU is dynamically controllable.*

Proof. We will show that the earliest time execution strategy $\breve{\mathcal{S}}$ is both viable and dynamic. Viability is immediate since (1) having reached quiescence, all projections are consistent STNs (Lemma 18) and (2) the earliest time schedule for a consistent STN is known to be a solution (Dechter et al., 1991).

To establish that $\breve{\mathcal{S}}$ is dynamic, consider a projection p and an executable timepoint X. Suppose $\breve{S}_p(X) = t$. Let p' be any other projection such that $\breve{S}_p(\trianglelefteq t) = \breve{S}_{p'}(\trianglelefteq t)$. We must show that $\breve{S}_p(X) = \breve{S}_{p'}(X)$.

By theorem 2 the earliest-times schedules for p and p' (and thus the values of $\breve{S}_p(X)$ and $\breve{S}_{p'}(X)$) depend only on the negative edges in their distance graphs. Furthermore, the negative paths emanating from X necessarily lead to timepoints that are in the past (in any schedule). Since $\breve{S}_p(\trianglelefteq t) = \breve{S}_{p'}(\trianglelefteq t)$, any negative edges in those paths that arise from macro-links must have the same weights in p and p'. Also, any edges that arise from negative labeled edges have their weights determined by macro-links that have already finished by time t. Thus, the negative paths emanating from X in p are also there in p'.

The only possible confounding issue might be if an outgoing edge from X could be non-negative in p but negative in p'. However, we see from the invariants that while the magnitudes of the values of labeled edges may vary between projections, their signs do not change. We conclude that $\breve{S}_p(X) = \breve{S}_{p'}(X)$. □

For the following corollary, recall that an essential negative cycle is a cycle that is reducible to a cycle of all negative edges.

Corollary 3.1. *If a POSTNU is not Dynamically Controllable, then either the initial labeled distance graph contains an essential negative cycle, or N^2+N rounds of reductions will lead to a failed same-root check.*

Proof. Suppose the initial labeled distance graph does not contain an essential negative cycle. By lemma 16 (b), quiescence will be achieved within $N^2 + N$ rounds of reductions. If there are no failed same-root checks at that point, then the POSTNU will be Dynamically Controllable (DC) by the Completeness Theorem. Thus, if it is not DC, one of the conditions must be untrue. □

Remark The property for a POSTNU of having an essential negative cycle is analogous to the property for an STNU of having a semi-reducible negative cycle. However, for a POSTNU, that property is only *sufficient* to exclude Dynamically Controllability. For a necessary condition, it must be coupled with the same-root checks, as specified in the corollary.

9.3 Earliest Time Execution Design

It is of interest to consider a possible implementation design for the earliest time execution strategy.

We assume a separate thread is assigned to an executable timepoint X to manage its activation. The negative edges emanating from X constitute enabling conditions. The thread

keeps a count of how many conditions are still unsatisfied and executes X as soon as the count becomes zero. From the point of view of X, the target Y of a negative edge from X is something to be observed, even if Y is itself an executable timepoint. In general, therefore the edge will involve an enabling condition of the form

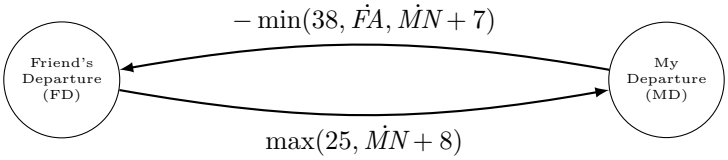
$$\min_{i \in I}(Y_i + u_i)$$

where the Y_i may be observable or executable timepoints (and where I may be a singleton).

The X thread will normally be sleeping. It will wake if any of the Y_i events occur. If that happens, the X thread will initiate a separate "alarm-clock" thread that will "ring" u_i units of time later. At any time there may be several such clocks that are active for a particular edge. The first one to ring will wake the X thread, at which point it will deactivate the other clocks for that edge, and will decrement the enablement count.

It is easy to see that the execution need only perform a bounded amount of active work for each $Y_i + u_i$ condition. Thus, the execution complexity is linear in the number of such conditions. For example, in a class of POSTNUs where the label sizes are bounded (such as STNUs), the complexity would be linear in the number of edges, and thus quadratic in the number of timepoints. This is similar to the STNU execution complexity for the latest time approach in (Cairo et al., 2018; Cairo & Rizzi, 2017).

Example 10 (running example)**.** *Given the final network of the post office example that was classified as dynamically controllable:*



the earliest time execution strategy would only consider the negative edge MD ⟶ FD.

*The table below gives the execution time that would be chosen for my departure (*MD*) based on the observed occurrence times of* FD*,* MN *and* FA*. Without loss of generality, the* FD *timepoint was set to execute at time 0, while the occurrence time of* FA *does not impact the schedule and is left unspecified.*

FD	*MN*	*FA*	*MD (execution time)*
0	*16*	-	*23* $(= MN + 7)$
0	*17*	-	*24* $(= MN + 7)$
0	*18*	-	*25* $(= MN + 7)$
	...		
0	*31*	-	*38* $(= MN + 7 = FD + 38)$
0	*32*	-	*38* $(= FD + 38)$

It should be noted that here the execution strategy switches from "7 minutes after the notification" to " 38 minutes after the friend's departure" once the notification arrives 31 minutes after the departure. This allows a more eager execution than a more ad-hoc strategy while still guaranteeing the satisfaction of all constraints.

9.4 Polynomial Complexity of the Naïve Algorithm

The precise complexity of the Dynamic Controllability algorithm depends on how cleverly the data structures and operations are represented. For a straightforward implementation, performing a round of reductions can be done by (1) iterating through pairs of edges, (2) reducing them if applicable and effective, and (3) removing any superseded edge. Since the number of incomparable edges is limited to $O(N^2)$, iterating through the pairs is bounded by $O(N^4)$. For each of these pairs, (2) and (3) can be done in $O(N)$ time. Thus, the combined time complexity is $O(N^5)$ for a single round of reductions. Since there are $N^2 + N = O(N^2)$ rounds, the total comes to $O(N^7)$ for the whole binary reduction portion.

It remains to consider the complexity of detecting all-negative cycles, and doing the same-root checking. Checking whether the directed subgraph of negative edges contains a cycle can be done in time proportional to the number of edges, which for our purposes has an $O(N^2)$ bound. This bound is dominated by the $O(N^7)$ bound of the binary reduction process.

Same-root checking The same-root checking requires a more careful analysis. First, consider same-root cases where at least one of the edges has an ordinary numeric weight. Since the numeric weight is independent of any compound label, this resembles the binary reduction case, and is easily resolved in $O(1)$ time by comparing the scalar components of the edges. Moreover, since only the tightest numeric weight is at issue, there are at most $O(N^2)$ such cases.

Note also that it is possible for two distinct hidden groups to share the same root timepoint. (This could occur if they do not share any microlinks.) In this case, labels from the two hidden groups will vary independently, and again these same-root cases can be resolved in $O(1)$ time by comparing the scalar components. The number of cases depends on the number of initial compound labels in each hidden group, which is bounded by $O(N)$. Thus, the number of cases is bounded by $O(N^2)$.

Otherwise, each same-root checking problem corresponds to a pair of initial compound labels within the same hidden group. Given a hidden group G, let G_l be the number of initial compound labels. Then G_l^2 is an upper bound on the number of pairs. If G_t is the number of timepoints in G, then the complexity of solving the derived STN for each pair can be expressed as $O(G_t^3)$. (This dominates the setup time for the STN.) The total complexity of same-root checking for G is then $O(G_l^2 * G_t^3)$. For the whole network, we get

$$O(\sum_G (G_l^2 * G_t^3)) \leq O(\sum_G (G_l * N * N^3)) = O(N^4 \sum_G (G_l)) \leq O(N^5)$$

as a generous overall complexity. This bound is also dominated by the $O(N^7)$ complexity derived earlier for the binary reduction process.

It is worth noting that if the individual hidden groups are bounded in the number of timepoints, as may be the case in practice, then both G_l and G_t are $O(1)$, and the complexity of the same-root checking phase is $O(N)$.

This confirms the overall problem is polynomial, answering the question posed by (Bhargava & Williams, 2019) for POSTNUs. However, this complexity is not the best possible. We now consider some improvements to the algorithm that reduce the order of the polynomial for individual portions and the overall algorithm.

9.5 Improved Algorithm

The first thing to notice for the binary checking phase is that the invariant representations can be exploited to reduce the complexity of performing the reductions. The MINOT and MAXOT expressions do not actually need to be evaluated until the same-root checking post-process. During the binary reduction phase, the MINOT and MAXOT expressions can be retained as symbolic representations of the labels. Let scalar(ℓ) denote the scalar component of a label ℓ. Note that

$$\text{scalar}(\textsc{maxot}(\ell_1 - q_1)) = \text{scalar}(\ell_1) - q_1$$

$$\text{scalar}(\textsc{minot}(\ell_2 + q_2)) = \text{scalar}(\ell_2) + q_2.$$

Consequently, a binary reduction

$$\textsc{maxot}(\ell_1 - q_1) + \textsc{minot}(\ell_2 + q_2)$$

reduces to either $\textsc{maxot}(\ell_1 - q_1 - (\text{scalar}(\ell_2) + q_2))$ or $\textsc{minot}(\ell_2 + q_2 + (\text{scalar}(\ell_1) - q_1))$ depending on whether

$$(\text{scalar}(\ell_1) - q_1) - (\text{scalar}(\ell_2) + q_2)$$

is, respectively, non-negative or negative.

In this form, the binary reductions take $O(1)$ (constant) time, which shaves one factor of N from the overall complexity. Also observe that

$$q_1 = \text{scalar}(\ell_1) - \text{scalar}(\textsc{maxot}(\ell_1 - q_1)).$$

A similar remark applies to the MINOT labels. Thus, we can recover the invariant representation of a label from its scalar length and its initial edge. Consequently, a representation of the label as a pair of those items is simpler and has the same information.

Next we can notice, for a round of reductions, that we only need to iterate through pairs of *consecutive* edges, not through all pairs of edges. To exploit that, we introduce a more compact representation of edge values.

Consider an enumeration $\mathcal{E}$ consisting of (1) all timepoints, and (2) labels ℓ, where ℓ is an initial compound label. Since initial compound labels do not share their compound observable timepoints, the cardinality of $\mathcal{E}$ is $O(N)$. Let $\mathcal{E}_T$ denote the subset of timepoints in $\mathcal{E}$ while $\mathcal{E}_L$ denotes the subset of labels. For ℓ in $\mathcal{E}_L$, we will denote the root timepoint of ℓ by root(ℓ). We will also adopt the convention for x in $\mathcal{E}_T$ that root(x) = x.

Now consider a table $\mathcal{T}(\mathcal{E})$ that has a row and column for each element of $\mathcal{E}$. The cell corresponding to a row element x and a column element y may be used to store a representation of a label on an edge from root(x) to root(y). If x or y is a compound label the cell may be used to store derived scalar distances. If x and y are both timepoints, the cell may be used to store derived ordinary distances.

In more detail, the cells of the table are used to store values as follows:

1. Cells where the row and column are both timepoints are used to store the weights of derived edges that have ordinary numeric values.

2. Cells where the row is a non-negative label and the column is a timepoint are used to store non-negative scalar distances.

3. Cells where the row is a timepoint and the column is a negative label are used to store negative scalar distances.

4. The other cells are unused.

Thus, the table is used to maintain propagated distances as the binary reduction process proceeds. Separate cells are used to maintain separate distance calculations for the different initial labels and for the ordinary numeric values.

Using this table representation of the edges, the Do-Round-Of-Reductions procedure can iterate through triples x, y, z of elements in $\mathcal{E}$, where y is restricted to timepoints and $\text{root}(x) \neq \text{root}(z)$ (to avoid the same-root case). From x and y we can recover an edge and its invariant representation, and similarly for y and z. We may then consider those edges, which are consecutive, for a reduction. This reduces the complexity of a round of reductions to $O(N^3)$ and the whole binary reduction phase to $O(N^5)$.

A major remaining source of inefficiency is that edges are added that are later superseded by tighter edges. For quiescence, it is sufficient to derive edges that are never superseded, i.e., the tightest edges for a row/column pair in the table representation. Recall the example

$$(\text{A} \xrightarrow{+2} (\text{B} \xrightarrow{+5} \text{C} \xrightarrow{-3} \text{D}) \xrightarrow{+7} \text{E} \xrightarrow{-10} \text{F})$$

illustrating an unrolled initial path that reduces to a non-negative edge. After the innermost nested group is reduced, this becomes

$$(\text{A} \xrightarrow{+2} \text{B} \xrightarrow{+2} \text{D} \xrightarrow{+7} \text{E} \xrightarrow{-10} \text{F}).$$

Note that the non-negative subpath from A to E may not be the shortest non-negative subpath between those two timepoints. If not, then the derived edge from A to F will not be the tightest, and it will eventually be superseded by a tighter edge. We could avoid that fate by only considering shortest non-negative subpaths in our derivations, similar to the behavior of the recursive Dijkstra algorithm of Morris (2014).

Recursive Algorithm We now consider an adaption to POSTNUs of the (Morris, 2014) algorithm, which is cubic for STNU problems. Like in the STNU case, the key to an efficient binary reduction process is to perform the reductions in an organized way that tries to avoid producing non-tightest edges that would ultimately be superseded.

This is not a problem for the (Morris, 2014) algorithm because it computes tightest edges as it goes. That approach was based on some key ideas (originally suggested by Nicola Muscettola).

1. Compute reduced distance backward from negative nodes

2. Use Dijkstra's algorithm (D) for shortest paths along non-negative edges.

3. Invoke recursively at negative nodes before continuing D.

4. A recursive loop characterizes not DC.

In our adaption, essential negative cycles play a role similar to semi-reducible negative cycles in the original algorithm. However, a post-process is required for the same-root

checks since the essential negative cycles do not fully characterize the failure criteria. We also introduce a post-process to check for all-negative cycles. (This might not be strictly necessary, but is a low-complexity operation that simplifies the analysis.)

We will make use of the $\mathcal{T}(\mathcal{E})$ table discussed earlier to store derived distances. As in (Morris, 2014), a negative node is one that has incoming negative edges in the labeled distance graph. Note that the reductions do not alter the set of negative nodes (since the new negative edges will always have the same root node as the parent negative edge).

To support Dijkstra computations, we will utilize a priority queue Q_x for each negative node x in $\mathcal{E}_T$ and a priority queue Q_l for each negative label in $\mathcal{E}_L$. The Q_x priority queue will store edges incoming to x (original and derived) that have negative numerical weights. The queue is ordered by the weights (increasing order) as stored in the $\mathcal{T}$ table. (Note the comments in (Morris, 2014) regarding the validity of Dijkstra computations when the only negative edges are initial edges.) The Q_l priority queue stores edges incoming to root(l) with a restricted set of labels. This set contains only l itself or negative compound labels in a chain derived from l. The queue is ordered by the scalar distances (increasing order) as stored in the $\mathcal{T}$ table.

In (Morris, 2014), for expository simplicity, a preprocessing step divided the start nodes of contingent links into multiple nodes connected by simultaneity constraints. Each divided node either had only ordinary incoming negative edges, or was the start of only a single contingent link. The purpose was to separate distinct distance calculations. In the current work, distinct priority queues serve the same purpose, so we avoid dividing the root nodes. However, the algorithm has separate recursive calls corresponding to the separate priority queues.

Notation Given a node x, we will denote the set of original incoming negative compound edges by ONC(x), and the set of original incoming negative ordinary edges by ONO(x).

Detailed Recursive Algorithm The complete algorithm is shown in algorithms 5 and 6. It is very similar to the recursive backprop algorithm in (Morris, 2014) except we have divided the DCBackprop part into several subprocedures that initialize different priority queues for the different distance calculations. The processing of the queues is the same and is isolated in a separate subprocedure.

Like the naïve algorithm, the Recursive-POSTNU-DC algorithm starts by computing tightest derived edges necessary for DC checking (lines 2-4). It does so by running a backward propagation procedure DCBackprop exactly once on each *negative node*, i.e., node with incoming negative edges. This is to ensure that every negative node is eventually processed, but some of the calls may be trivial in the sense that the negative node has already been processed by a prior recursive call from DCBackProp.

DCBackprop($source, \Gamma$) maintains a queue of negative edges $B \rightarrow source$ ordered by increasing scalar value. Given such an edge, lines 51-56 select any non-negative edge $A \rightarrow B$ and applies *non-same-root cross-case* reduction to get a new edge $A \rightarrow source$ that is added to the network unless subsumed by an existing edge. If the derived edge is negative, it is enqueued for later processing. Prior to performing this cross-case reduction, we ensure that all tightest non-negative Edges incoming to B have been derived by a recursive call to DCBackprop(B, Γ) (lines 48-50). The reduction process continues until all (i.e.

Algorithm 5 Improved algorithm for DC checking of POSTNUs, based on the recursive derivation of tightest edges.

```
 1: procedure RECURSIVE-POSTNU-DC(Γ)
 2:     for all negative node n ∈ Γ do
 3:         if DCBACKPROP(n, Γ) = false then
 4:             return false
 5:     if all-negative cycle in Γ then
 6:         return false
 7:     if same-root check fails in Γ then
 8:         return false
 9:     return true
10:
11: procedure DCBACKPROP(source, Γ)
12:     if ancestor DCBackprop call with same source then
13:         return false
14:     if prior terminated call with same source then
15:         return true
16:     if DCBACKPROP_N(source, Γ) = false then
17:         return false
18:     for all edge l ∈ ONC(source) do
19:         if DCBACKPROP_L(l, Γ) = false then
20:             return false
21:     return true
```

existing and derived) negative edges targeting *source* have been extracted from the queue and processed.

The recursive structure enables two early exits of DCBACKPROP. First, a call to DCBACKPROP(X, Γ) that would result in a recursive invocation to DCBACKPROP(X, Γ) exhibits the presence of a cycle of all-negative edges and one can infer the uncontrollability of the network (line 13). Second, if a previous call to DCBACKPROP with the same source previously completed, the propagation exits immediately as it would not result in any new reduction (line 15).

As in the naïve algorithm, once the propagation process has completed, the algorithm proceed to assess the controllability of the network by looking for cycles of negative edges and performing the *same-root cross-case* checks (lines 5-8). If both checks succeed, then the network is classified as dynamically controllable (line 9).

Completeness and Correctness Before establishing its complexity, we now prove completeness and correctness of this algorithm.

Lemma 20. *If the recursive backprop algorithm completes successfully, the resulting generalized distance graph is quiescent.*

Proof. Suppose otherwise. Then after completion, there is a plus/minus pair A $\xrightarrow{+x}$ B $\xrightarrow{-y}$ C with an effective reduction, where $+x$ is non-negative, $-y$ is negative, and A $\neq$ C (thus not a same-root case).

Algorithm 6 Helper functions for the recursive algorithm. These adapt the Dijkstra algorithm for ordered edge derivation with a recursive invocation on encountered negative nodes.

```
33: procedure DCBackprop_N(source, Γ)
34:     PriorityQueue q ← empty
35:     for all edge e_0 ∈ ONO(source) do
36:         insert e_0 ∈ q
37:     return DCBackprop_Q(q, Γ)
38:
39: procedure DCBackprop_L(l, Γ)
40:     PriorityQueue q ← empty
41:     insert l ∈ q
42:     return DCBackprop_Q(q, Γ)
43:
44: procedure DCBackprop_Q(q, Γ)
45:     while q not empty do
46:         pop edge e from q
47:         node n ← start(e)
48:         if n is negative node then
49:             if DCBackprop(n, Γ) = false then
50:                 return false
51:         for all non-negative e′ in InEdges(n) do
52:             if e, e′ is not same-root case and e, e′ effectively derives e″ then
53:                 add e″ to Γ
54:                 remove any superseded e‴ from Γ
55:                 if e″ is negative edge then
56:                     insert e″ in q
57:     return true
```

Consider the precursor paths A $\xrightarrow{+x'}$ ...B and B... $\xrightarrow{-y'}$ C in the initial graph that eventually reduce to A $\xrightarrow{+x}$ B and B $\xrightarrow{-y}$ C, respectively.

Since all the negative nodes will eventually be selected for processing by DCBackprop (either from Recursive-POSTNU-DC directly or via a recursive call from DCBackprop_Q, at some point C will be selected, and the B... $\xrightarrow{-y'}$ C path will be reduced to B $\xrightarrow{-y}$ C. However, this reduced edge is negative, so the processing of C will not be complete until any edges preceding B have been considered.

There are now three possibilities. First, the A $\xrightarrow{+x}$ B edge may have been present in the initial graph, in which case the precursor path is the edge itself. The processing of C will then continue until the A $\xrightarrow{+x}$ B $\xrightarrow{-y}$ C plus/minus pair has been reduced. If the precursor path is not the edge itself, then in order to reduce to it, the edge entering B in the path must be negative; thus, B must be a negative node. The two remaining possibilities are that B is processed before C or after C. If B is processed before C, the outcome is the same as in the first possibility. Otherwise, during the processing of C, there will be a recursive call when

B is reached. This means the processing of B will be *finished* before that of C even though not *started* before C. Again, the outcome is the same. This contradicts the assumption that the reduction after completion is effective, and establishes the result. □

Theorem 4. *The recursive algorithm completes successfully if and only the POSTNU is Dynamically Controllable.*

Proof. Unsuccessful completion implies either a recursive loop, or an all-negative cycle, or a same-root checking failure. It is easy to see that a recursive loop also implies a derived all-negative cycle. Thus, all of these conditions rule out Dynamic Controllability.

If none of these conditions apply, then after completion, the generalized distance graph is quiescent by lemma 20 and does not contain an all-negative cycle or a same-root inconsistency. In that case, the proof of theorem 3 applies and shows the existence of a viable dynamic strategy. Thus, the POSTNU is Dynamically Controllable. □

Remark Our prior results also show that after application of the algorithm, the projections of the generalized distance graph are plus/minus closed, have the V-path property, and are dispatchable.

Complexity In the following, N and E are the number of nodes and edges, respectively, in the final labeled distance graph.

As discussed previously, the invariant representation allows binary reductions to be performed in $O(1)$ (bounded) time. The complexity of the binary reduction phase involves $O(N)$ calls to the backward Dijkstra algorithm (one for each element in $\mathcal{E}$). The cost of a single call to this algorithm is $O(E + N * \log(N))$. As discussed previously the number of non-redundant edges can be limited to $O(N^2)$. Thus the overall complexity of this phase has an $O(N^3)$ bound.

Also, as previously discussed, the checking for all-negative cycles is $O(N^2)$. The same-root checking is $O(N^5)$ in general, so this dominates the overall complexity of the whole algorithm. In the special case where the hidden groups have bounded size, the overall complexity is $O(N^3)$, which is the same theoretical complexity as the algorithm of Morris (2014) for checking the Dynamic Controllability of STNUs.

10. Closing Remarks

The focus of this paper has been theoretical with the goal of providing a sound and complete process for determining the Dynamic Controllability of partially observable STNUs. We have presented two algorithms for this determination that run in polynomial time, including an adaption of the STNU recursive backward Dijkstra approach of Morris (2014). Given a POSTNU with N timepoints, this second algorithm has a worst-case complexity in $O(N^5)$ in the general case. In the common case where the individual non-observable components of the network have a bounded size, its complexity reduces to $O(N^3)$.

The approach here answers the question posed by Bhargava and Williams (2019) for POSTNUs: the complete problem indeed has polynomial complexity. A more long-term objective would be to generalize the setting further, and replace Nature by a second agent, thus considering two-agent plans with limited communication. This would be related to the MaSTNU problem also discussed by Bhargava and Williams (2019). However, in that

extended problem, the second agent could have a more general STN than the Nature one, and this would affect many of the assumptions used in this paper for POSTNUs. Thus, the question of the complexity of the MaSTNU problem remains open.

An approach in another paper (Mountakis et al., 2017) considers dynamic updates during execution of a multi-agent problem using what may be regarded as a Strong Controllability form of multi-agent decoupling. However, this is not the same as Dynamic Controllability, which determines the existence of a dynamic execution strategy offline prior to execution.

Acknowledgments

Arthur Bit-Monnot wishes to thank Paul Morris, who passed away shortly after the submission of this paper. Since our first discussion on the subject in 2016, Paul has been the main driving force to bring this work to completion. Without his dedication and efforts, this paper would never have existed.

The work of Arthur Bit-Monnot has been partially supported by AIPlan4EU, a project funded by EU Horizon 2020 research and innovation program under GA n.101016442.

References

Bhargava, N., Muise, C., & Williams, B. (2018). Variable-Delay Controllability. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Bhargava, N., & Williams, B. (2019). Multiagent Disjunctive Temporal Networks. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Bit-Monnot, Ghallab, M., & Ingrand, F. (2016). Which contingent events to observe for the dynamic controllability of a plan. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Cairo, M., Hunsberger, L., & Rizzi, R. (2018). Faster Dynamic Controllability Checking for Simple Temporal Networks with Uncertainty. *International Symposium on Temporal Representation and Reasoning (TIME)*.

Cairo, M., & Rizzi, R. (2017). Dynamic Controllability Made Simple. *International Symposium on Temporal Representation and Reasoning (TIME)*.

Combi, C., Hunsberger, L., & Posenato, R. (2014). An Algorithm for Checking the Dynamic Controllability of a Conditional Simple Temporal Network with Uncertainty - Revisited. *Agents and Artificial Intelligence*.

Conrad, P. R., & Williams, B. C. (2011). Drake: An Efficient Executive for Temporal Plans with Choice. *Journal of Artificial Intelligence Research (JAIR)*, *42*(1).

Cormen, T., Leiserson, C., & Rivest, R. (1990). *Introduction to Algorithms*. MIT press.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal Constraint Networks. *Artificial Intelligence*, *49*.

Frank, J. (2019). On expected value strong controllability. *ICAPS Workshop on Integrating Planning, Acting, and Execution (IntEx)*.

Gao, M., Popowski, L., & Boerkoel, J. C. (2020). Dynamic Control of Probabilistic Simple Temporal Networks. *AAAI Conference on Artificial Intelligence (AAAI)*.

Hunsberger, L. (2002). Algorithms for a temporal decoupling problem in multi-agent planning. *AAAI Conference on Artificial Intelligence (AAAI)*.

Hunsberger, L. (2009). Fixing the Semantics for Dynamic Controllability and Providing a more Practical Characterization of Dynamic Execution Strategies. *International Symposium on Temporal Representation and Reasoning (TIME)*.

Hunsberger, L. (2013). Magic Loops in Simple Temporal Networks with Uncertainty - Exploiting Structure to Speed Up Dynamic Controllability Checking. *International Conference on Agents and Artificial Intelligence (ICAART)*.

Hunsberger, L., & Posenato, R. (2020). Faster Dynamic-Consistency Checking for Conditional Simple Temporal Networks. *International Conference on Automated Planning and Scheduling (ICAPS)*.

Moffitt, M. D. (2007). On the Partial Observability of Temporal Uncertainty. *AAAI Conference on Artificial Intelligence (AAAI)*.

Morris, P., & Muscettola, N. (1999). Managing Temporal Uncertainty Through Waypoint Controllability. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Morris, P., & Muscettola, N. (2005). Dynamic Controllability Revisited. *AAAI Conference on Artificial Intelligence (AAAI)*.

Morris, P., Muscettola, N., & Vidal, T. (2001). Dynamic Control of Plans with Temporal Uncertainty. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Morris, P. (2006). A Structural Characterization of Temporal Dynamic Controllability. *International Conference on Principles and Practice of Constraint Programming (CP)*.

Morris, P. (2014). Dynamic Controllability and Dispatchability Relationships. *Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*.

Morris, P. (2016). The Mathematics of Dispatchability Revisited. *International Conference on Automated Planning and Scheduling (ICAPS)*.

Mountakis, K. S., Klos, T., & Witteveen, C. (2017). Dynamic Temporal Decoupling. *Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*.

Muscettola, N., Morris, P., & Tsamardinos, I. (1998). Reformulating Temporal Plans For Efficient Execution. *International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Muscettola, N., Nayak, P., Pell, B., & Williams, B. (1998). Remote Agent: to boldly go where no AI system has gone before. *Artificial Intelligence*, *103*(1-2).

Nilsson, M., Kvarnström, J., & Doherty, P. (2015). Efficient Processing of Simple Temporal Networks with Uncertainty: Algorithms for Dynamic Controllability Verification. *Acta Informatica*, *53*.

Shah, J. A., Stedl, J., Williams, B. C., & Robertson, P. (2007). A Fast Incremental Algorithm for Maintaining Dispatchability of Partially Controllable Plans. *International Conference on Automated Planning and Scheduling (ICAPS)*.

Vidal, T. (2000). Controllability Characterization and Checking in Contingent Temporal Constraint Networks. *International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Vidal, T., & Fargier, H. (1999). Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, *11*.

Zavatteri, M., Combi, C., Rizzi, R., & Viganò, L. (2019). Hybrid SAT-Based Consistency Checking Algorithms for Simple Temporal Networks with Decisions. *International Symposium on Temporal Representation and Reasoning (TIME)*.

Zavatteri, M., & Viganò, L. (2019). Conditional simple temporal networks with uncertainty and decisions. *Theoretical Computer Science*, *797*.