Technical VDC

Cameron W. Pittman

June 24, 2023

Contents

1 Modeling Temporal Networks

Our aim is to architect a pipeline that takes a temporal network as input and produces temporally consistent actions as output. This Section lays the groundwork for the delay scheduler by first describing our chosen models for temporal constraints and scheduling algorithms. We then outline the architecture of the delay scheduler, before finally providing the necessary background on our choice for modeling observation delay.

TODO . . . take the approach of defining temporal networks, and updating our defns as we work towards STNUs

1.1 Temporal Networks

Temporal networks form the backbone of our architecture for temporal reasoning under observation delay. Simple Temporal Networks (STNs) offer the basic building blocks for most expressive temporal network formalisms (Dechter, Rina and Meiri, Itay and Pearl, Judea, 1991). An STN is composed of a set of variables and a set of binary constraints, each of which limits the difference between a pair of these variables; for example, $B-A \in [10, 20]$. Each variable denotes a distinguished point in time, called an *event*. Constraints over events are binary *temporal constraints* that limit their temporal difference; for example, the aforementioned constraint specifies that event A must happen between 10 and 20 minutes before event B.

STN (Dechter, Rina and Meiri, Itay and Pearl, Judea, 1991)

An *STN* is a pair $\langle X, R\rangle$, where:

- X is a set of variables, called events, each with a domain of the reals $\mathbb{R}$, and
- R is a set of simple temporal constraints. Each constraint $\langle x_r, y_r, l_r, u_r\rangle$ has scope $\{x_r, y_r\} \subseteq X$ and relation $x_r - y_r \in [l_r, u_r]$.

Schedule (,)

A *schedule*, ξ, is a mapping of events to times, $\xi : X \to \mathbb{R}$.

An STN is used to frame scheduling problems. A schedule is feasible if it satisfies each constraint in R. We use the notation $\xi(x)$ to represent a mapping from an event, x, to a time, $x \to \mathbb{R}$, in the schedule. A schedule is *complete* if all $x \in X$ are assigned times in ξ. An STN is *consistent* if it has at least one feasible schedule that assigns all events in X.

An STN is consistent if and only if there is no negative cycle in its equivalent distance graph (Dechter, Rina and Meiri, Itay and Pearl, Judea, 1991). Let n be the number of events in a temporal network and m to be the number of constraints. Then consistency of an STN can be checked in $O(mn)$ time using the Bellman-Ford Algorithm to check for negative cycles.

While an STN is useful for modeling problems in which an agent can control the exact time of all events, it does not let us model actions whose durations are uncertain. A Simple Temporal Network with Uncertainty (STNU) is an extension to an STN that allows us to model these types of uncertain actions (Vidal, Thierry and Fargier, Héléne, 1999).

STNU (Vidal, Thierry and Fargier, Héléne, 1999)

An *STNU* S is a quadruple $\langle X_e, X_c, R_r, R_c\rangle$, where:

- X_e is the set of executable events with domain $\mathbb{R}$,
- X_c is the set of contingent events with domain $\mathbb{R}$,
- R_r is the set of requirement constraints of the form $l_r \leq x_r - y_r \leq u_r$, where $x_r, y_r \in X_c \cup X_e$ and $l_r, u_r \in \mathbb{R}$, and
- R_c is the set of contingent constraints of the form $0 \leq l_r \leq c_r - e_r \leq u_r$, where $c_r \in X_c$, $e_r \in X_e$ and $l_r, u_r \in \mathbb{R}$.

An STNU divides its events into executable and contingent events and divides its constraints into requirement and contingent constraints. The times of executable events are under the control of an agent, and assigned by its scheduler. STNU executable events are equivalent to events in an STN. Contingent events are controlled by nature. Contingent constraints model the temporal consequences of uncertain actions and are enforced by nature. They relate a starting executable event and an ending contingent event. To ensure causality, the lower-bound of a contingent constraint is required to be non-negative; hence, the end event of the constraint follows its start event. Contingent constraints are not allowed to be immediately followed by additional contingent constraints. Requirement constraints specify constraints that the scheduler needs to satisfy and may relate any pair of events. An STNU requirement constraint is equivalent to an STN constraint.

To clarify terminology, we sometimes refer to contingent constraints as contingent links and requirement constraints as requirement links. When we discuss contingent constraint or contingent link duration, we refer to the amount of time that actually elapses between a contingent link's starting executable event and its ending contingent event. We sometimes refer to STNUs as defined in Defintion 1.1 as *vanilla* STNUs (in contrast to the many "flavors" of STNUs, namely the variants with fixed and variable observation delay functions as will be defined below).

With STNs, our goal is to construct a consistent schedule for all events such that all constraints are satisfied. In STNUs, however, contingent events cannot be scheduled directly. Instead, we are interested in determining whether there is a *controllable* policy that guarantees that a schedule can be constructed such that all constraints are satisfied despite how uncertainty is resolved.

Situations (Vidal, Thierry and Fargier, Héléne, 1999)

For an STNU S with k contingent constraints $\langle e_1, c_1, l_1, u_1\rangle, \cdots, \langle e_k, c_k, l_k, u_k\rangle$, each *situation*, ω, represents a possible set of values for all links in S, $\omega = (\omega_1, \cdots, \omega_k) \in \Omega$. The *space of situations* for S, Ω, is $\Omega = [e_1, c_1] \times \cdots \times [e_k, c_k]$.

Each *situation* in the *space of situations*, $\omega \in \Omega$, represents a different assignment of contingent links in the schedule (Vidal, Thierry and Fargier, Héléne, 1999). We may represent the situation for a specific constraint as ω_i for the i-th constraint in S, or $\omega(E)$ for contingent event E. Situations are sets of intervals. To examine spaces of situations, we can make the following comparisons.

Comparisons of Spaces of Situations

Given two spaces of situations, Ω_1 and Ω_2, with contingent link j, $1 \leq j \leq k$,

- $\Omega_1 = \Omega_2$ if and only if $\omega_1 = \omega_2 \forall \omega_1 \in \Omega_1 \forall \omega_2 \in \Omega_2$
- $\Omega_1 \subset \Omega_2$ if situation j in Ω_1 is a subset of situation j in Ω_2, $\omega_{1j} \subset \omega_{2j}$, and all other situations are equivalent
- $\Omega_1 \subset \Omega_2$ if Ω_1 omits contingent link j, e.g. $\Omega_1 = \prod_{\substack{i=1 \\ i \neq j}}^{k} [e_{1i}, c_{1i}]$, and all other situations are equal.

Situations may be applied to STNUs.

Projection (Vidal, Thierry and Fargier, Héléne, 1999, ,)

A *Projection* is an application of a situation, ω, on an STNU S, which collapses the durations of contingent links to specific durations resulting in an STN.

A *projection* is an STN that is the result of applying a situation to an STNU, and thus the contingent links have reduced from uncertain ranges to specific durations (Vidal, Thierry and Fargier, Héléne, 1999, ,).

Execution Strategy

An *execution strategy*, $\mathcal{S}$, is a mapping of situations to schedules, $\mathcal{S} : \Omega \to \Xi$.

An *execution strategy* then naturally maps a specific resolution of the uncertainty of the contingent constraints to a set of assignments for the events of an STNU. For an STNU, time monotonically increases and we only observe *activated* contingent events, or those contingent events at the tail of a contingent link whose free event predecessor has been executed. As such, we modify our definition of ξ.

Partial Schedule

A *partial schedule*, ξ, is a mapping from a proper subset of events in X, X', to times, $\xi : X' \to \mathbb{R}$.

As a proper subset, ξ represents an assignment of events *so far* during the execution of an STNU. From here on, ξ refers to a partial schedule.

In the world of STNU literature, there are many forms of controllability that represent the ability of a scheduler to construct execution strategies that satisfy constraints under different conditions (Vidal, Thierry and Fargier, Héléne, 1999). Three forms of controllability, *strong*, *weak*, and *dynamic* are studied most often, though in practice we omit weak controllability from our analysis. As we will see below, variable-delay controllability will unify strong and dynamic controllability into a single theory. A temporal network is *strongly controllable* (or exhibits strong controllability), if there exists a complete schedule that will satisfy all constraints for all projections of the

STNU. A temporal network exhibits dynamic controllability if an execution strategy, $\mathcal{S}$, exists for a given partial schedule, ξ.

1.2 Fixed-Delay Controllability

Under fixed-delay controllability (, a), we consider the problem of scheduling execution decisions when the assignment of values to contingent events is learned (if at all) after some time has passed from the initial assignment. We prefer this model because it is flexible enough to model most forms of event observation we would expect to see in a real-time execution context. Fixed-delay controllability uses a *fixed-delay function* to encode the delay between when an event occurs and when it is observed by a scheduling agent. We sometimes refer to an STNU with an associated fixed-delay function as a *fixed-delay STNU.*

Fixed-Delay Function (, a)

A *fixed-delay function*, $\gamma : X_c \rightarrow \mathbb{R}^+ \cup \{\infty\}$, maps a contingent event to the amount of time that passes between when the event occurs and when its value is observed.

As a matter of convention, we use $AB[l, u]$ to represent requirement links between events A and B and use $AE[l, u]$ to represent contingent links between A and E. When we refer to the fixed-delay function associated with a contingent event E of some contingent constraint $AE[l, u]$, we use the notation $\gamma(E)$, or equivalently, γ_E. Without instantaneous observation of contingent events, we must clarify the relationship between when an event occurs and when it is *observed.*

Contingent Event Observation

Observations, , are a mapping from contingent events to times when the agent receives events, $: E \rightarrow \mathbb{R}$, based on the relationship, $(E) = \xi(E) + \gamma(E)$.

Note that $\xi(E)$ is now indirectly learned through the relationship,

$$\xi(E) = (E) - \gamma(E)$$

We also present a revised definition of situations, Ω_f, to reflect the impact of the delay function on event observations.

Fixed-Delay Situations

For an STNU S with k contingent constraints $\langle e_1, c_1, l_1, u_1\rangle, \cdots, \langle e_k, c_k, l_k, u_k\rangle$ and fixed-delay function γ, each *fixed-delay situation*, ω_f, represents a possible set of *observed* values for all links in S, $\omega_f = (\omega_{f1}, \cdots, \omega_{fk})$. The $\{$space of situations$\}$ for S, Ω_f, is $\Omega_f = [e_1, c_1] + [\gamma_1, \gamma_1] \times \cdots \times [e_k, c_k] + [\gamma_k, \gamma_k]$.

To emphasize that the *observed* value for an event is not the same as $\xi(E)$, we also use the term *observation space* as a synonym for the space of situations.

With the semantics of delayed observations in hand, we can define what it means for a fixed-delay STNU to be controllable.

Fixed-Delay Controllability (, a)

An STNU S is *fixed-delay controllable* with respect to a delay function, γ, if and only if for the space of situations, Ω_f, there exists an $\mathcal{S}$ that will construct a satisfying schedule for requirement constraints during execution, ξ.

Importantly, fixed-delay controllability (FDC) generalizes the two concepts of controllability that are central to STNUs, strong and dynamic controllability. In particular, by using a fixed-delay function where we observe all events instantaneously, checking fixed-delay controllability reduces to checking *dynamic controllability.* Similarly, a fixed-delay function that specifies we never observe any contingent event corresponds to checking *strong controllability* (Vidal, Thierry and Fargier, Héléne, 1999).

To determine whether an STNU is fixed-delay controllable, we determine whether there exists a *valid* execution strategy for it.

Valid Execution Strategy

A *valid* $\mathcal{S}$ is one that enforces that, for any $\omega_f \in \Omega_f$, while receiving information about the durations after a fixed delay, the outputted decision respects all existing temporal constraints and ensures the existence of a subsequent valid execution strategy following that action.

As is the case for a vanilla STNU, evaluating whether a valid $\mathcal{S}$ exists for a fixed-delay STNU reduces to checking for the presence of a *semi-reducible negative cycle* in a *labeled distance graph* derived from the fixed-delay STNU (Morris, Paul, 2006). The key insight for checking fixed-delay controllability is the inclusion of γ in the constraint generation rules for building the labeled distance graph [**?**].

The labeled distance graph corresponds to the constraints of the STNU with each unlabeled edge from A to B with weight w (denoted ABw) representing the inequality $x_B - x_A \leq w$. Labeled edges represent conditional constraints that apply depending on the realized value of contingent links in the graph. For example, a lower-case labeled edge from A to B with weight w and lower-case label c (denoted $ABc : w$) indicates that $x_B - x_A \leq w$ whenever the contingent link ending at C takes on its lowest possible value. An upper-case labeled edge from A to B with weight w and upper-case label C (denoted $ABC : w$) indicates that $x_B - x_A \leq w$ whenever the contingent link ending at C takes on its highest possible value. Given a labeled distance graph, there are several valid derivations we can apply to generate additional edges (see Table 1). If it is possible to derive a negative cycle that is free of lower-case edges, then the STNU has a *semi-reducible negative cycle* and the STNU is not controllable.

Note that with fixed-delay controllability, the lower-case and cross-case rules are modified from the Morris and Muscettola (Morris, Paul and Muscettola, Nicola, 2005), accounting for γ. More specifically, we address the case where observation delay makes it impossible to receive information about a contingent event before its immediate successor. More detail can be found in (Bhargava, Nikhil and Muise, Christian and Vaquero, Tiago and Williams, Brian, 2018).

Edge Generation Rules			
	Input edges	Conditions	Output edge
No-Case Rule	ABu, BCv	N/A	$ACu + v$
Upper-Case Rule	ADu, $DBC : v$	N/A	$ABC : u + v$
Lower-Case Rule	$ACc : x$, CDw	$w < \gamma(C)$, $C \neq D$	$ADx + w$
Cross-Case Rule	$ACc : x$, $CDB : w$	$w < \gamma(C)$, $B \neq C \neq D$	$ADB : x + w$
Label Removal Rule	$BAC : u$, $AC[x, y]$	$u > -x$	BAu

Table 1: Edge generation rules for a labeled distance graph from (Bhargava, Nikhil and Muise, Christian and Vaquero, Tiago and Williams, Brian, 2018).

1.3 Variable-Delay Controllability

While fixed-delay controllability is quite expressive, its fundamental limitation is that it cannot model variability in delay. If uncertainty in delay is added to the model, then the controllability problem requires the addition of reasoning over uncertainty in both the timing of contingent events and their subsequent observations. Notably, with variable observation delay, we no longer guaranteed to learn the true time when contingent events were assigned.

We now introduce this model in terms of definitions for a *variable-delay function* and *variable-delay controllability* (Bhargava, Nikhil and Muise, Christian and Williams, Brian C., 2018) as applied to *variable-delay STNUs*. Since variable-delay semantics generalizes the notion of fixed-delay, as a matter of convenience, we also use the simplified term *delay STNUs* to refer to STNUs with variable observation delay.

Variable-Delay Function

A *variable-delay function*, $: X_c \to (\mathbb{R}^+ \cup \{\infty\}) \times (\mathbb{R}^+ \cup \{\infty\})$, maps a contingent event to an interval $[a, b]$, where $a \leq b$. The interval bounds the time that passes after $\xi(x_c)$ before that value is observed to be assigned. No prior knowledge is assumed about the distribution associated with this interval.

Importantly, this model does not assume that an executing agent infers *when* a contingent event was executed, but instead infers *that* the event was executed. Like contingent constraints, the resolved value of will be selected by Nature during execution. Thus, when an agent receives an observation is a function of the independent resolutions of the contingent link and .

By convention, we use $^-(x_c)$ and $^+(x_c)$ to represent the lower-bound and upper-bound, respectively, of the range representing the possible delay in observation. For any fixed-delay function γ, we can produce a corresponding variable-delay function where $^+(x_c) =^- (x_c) = \gamma(x_c)$.

During execution, the *observation projection*, Γ, represents the resolution of observation delay.

Observation Projection

The *observation projection* Γ is a mapping from a contingent event to a fixed observation delay, $\Gamma \ : \ X_c \to \mathbb{R} \in [^-(X_c),^+ (X_c)]$.

Much like how a projection collapses a vanilla STNU to an STN, the observed projection collapses a contingent link with variable-observation delay to one with fixed-observation delay. However, unlike the projection of an STNU, the observation projection is not guaranteed to be learned. We update our definitions of , ξ, and Ω accordingly.

Contingent Event Observation

Contingent event observations, , are a mapping from contingent events to times when the agent receives events, $\ : \ E \to \mathbb{R}$, based on the relationship, $(E) = \xi(E) + \Gamma(E)$.

Determining the precise schedule of a contingent event, ξ, is complicated by an interval bounded Γ. We must use interval-bounded contingent event assignments instead.

Schedule

A *schedule*, ξ, when applied to contingent events, is a mapping of events to interval-bounded times, $\xi : X_c \to (\mathbb{R}^+ \cup \{\infty\}) \times (\mathbb{R}^{++} \cup \{\infty\})$, where, for any contingent event x_c, $\xi(x_c) \in [e_{x_c} +^- (x_c), c_{x_c} +^+ (x_c)]$.

We sometimes use interval bounded schedules for requirement events as well, where $\xi(x_r) = t = [t, t]$ for some requirement event x_r assigned to time t.

We once again revise our definition of situations, Ω_v, to reflect the impact of the variable-delay function on the space of observations.

Variable-Delay Situations

For an STNU S with k contingent constraints $\langle e_1, c_1, l_1, u_1\rangle, \cdots, \langle e_k, c_k, l_k, u_k\rangle$ and variable-delay function , each *variable-delay situation*, ω_v, represents a possible set of *observed* values for all links in S, $\omega = (\omega_{v1}, \cdots, \omega_{vk})$. The $\{$space of situations$\}$ for S, Ω_v, is $\Omega_v = [e_1, c_1] + [^{-}_{1}, ^{+}_{1}] \times \cdots \times [e_k, c_k] + [^{-}_{k}, ^{+}_{k}]$.

We see that the space of observations has likewise grown in the transition to variable observation delay. If $^{-} <^{+}$, Ω_v for variable observation delay is strictly larger than Ω_f for fixed-observation delay and Ω for vanilla STNUs.

Like the fixed-delay function for fixed-delay controllability, the variable-delay function relates an observation delay to a contingent event, independent of other events. We take a similar approach to defining variable-delay controllability, relative to fixed-delay controllability.

Variable-Delay Controllability

An STNU S is *variable-delay controllable* with respect to a variable-delay function, , if and only if for the space of situations, Ω_v, there is an $\mathcal{S}$ that produces a satisfying schedule for requirement events during execution, ξ.

Determining whether a given variable-delay STNU, S, is variable-delay controllable has two components (Bhargava, Nikhil and Muise, Christian and Williams, Brian C., 2018). The first is to derive a fixed-delay STNU, S', with fixed-observation delay, γ, that is equivalent with respect to controllability. The second is to show that S' is fixed-delay controllable. Below, we reiterate the claims of (Bhargava, Nikhil and Muise, Christian and Williams, Brian C., 2018), demonstrating how to derive S' from S that is equivalent with respect to controllability. We first demonstrate how to transform the contingent links from S to S', and demonstrate their correctness with respect to observation spaces, before following up with transformations to the requirement links to maintain the same scheduling semantics in S'.

For the following lemmas, let x_c be a contingent link in S, where $x_c \in [l, u]$ and variable-delay function (x_c). Let x'_c be the transformed contingent link in S' with fixed-delay function, $\gamma(x'_c)$.

Note that we receive (x_c) from Nature, but make the assignment $\xi(x'_c)$ in the dispatchable form of S'. To be clear, while (x_c) is an interval, $(\mathbb{R} \cup \infty) \times (\mathbb{R} \cup \infty)$, (x'_c) is in $\mathbb{R}$. For a fixed interval, e.g. $(x_c) \in [t, t]$, we sometimes employ an equivalent representation, $(x_c) = t$.

Additionally, we sometimes apply $-$ and $+$ superscripts to l and u to denote the earliest and latest times respectively that an assignment at those bounds could be observed. For instance, the relationship in Definition 1.3 simplifies to, viz-eqn-obs-assign.png

Figure 1: We visualize the relationship between realized assignments across S and S'. In this example, each horizontal line is a timeline monotonically increasing from left to right. Dashed lines represent observation delays. We see how an assignment in S, (x$_\mathrm{c}$)\$, realized observation delay, $g(x_c)$, and an observation in S, (x_c), lead to an assignment in S', (x'$_\mathrm{c}$)\$.

$$(x_c) = [l +^- (x_c), u +^+ (x_c)] \tag{1}$$

$$(x_c) = [l^-(x_c), u^+(x_c)] \tag{2}$$

Lastly, we need a means to compare observation spaces if we are to transform variable-delay to fixed-delay STNUs.

Observation Space Mapping

Let μ be a mapping from an assignment to a situation, $\mu : \xi \to \omega$. To say that $\mu(x'_c) \subseteq \omega_v(x_c)$ means that, for any assignment of x'_c in S', there is an equivalent situation in S for x_c.

For the transitions below, it is a *valid observation space mapping*, if we can show that $\mu(x'_c) \subseteq \omega_v(x_c)$. If so, it is guaranteed that any assignment in the observation space of x'_c also has a valid assignment in the observation space of x_c.

We now have the necessary vocabulary and notation to step through the transformations from S to S'. These lemmas were first presented in (Bhargava, Nikhil and Muise, Christian and Williams, Brian C., 2018).

Variable-Delay to Fixed-Delay Transformations

The *variable-delay to fixed-delay transformations* define a set of observation space mappings, where there are valid observation space mappings for all the contingent constraints in S' to S.

Thus, if there is a satisfying $\mathcal{S}$ for the fixed-delay observation space of S', it is guaranteed to simultaneously satisfy any situation in the variable-delay observation space, Ω_v, of S.

For any contingent event $x_c \in X_c$ in S, if $^-(x_c) =^+ (x_c)$, we emulate (x_c) in S' using $\gamma(x'_c) =^+ (x_c)$.

We translate an already fixed-bounded observation delay in the form of (x_c) to the equivalent fixed-delay function, $\gamma(x'_c)$, thus $\omega_f(x'_c) = \omega_v(x_c)$.

For any contingent event $x_c \in X_c$, $^+(x_c) = \infty$, we emulate (x_c) in S' as $\gamma(x'_c) = \infty$.

There are projections where we would not receive information about x_c, therefore we have to act as if we *never* receive an observation of x_c. Any $\mathcal{S}$ that works when we do not receive information about x_c would also work when do receive an observation if we choose to ignore the observation.

None of our decisions depend on $\xi(x'_c)$, thus no observation space mapping to S is necessary.

If $u - l \leq^+ (x_c) -^- (x_c)$, we emulate (x_c) in S' using $\gamma(x'_c) = \infty$.

We can ignore observations of x_c because they are not guaranteed to narrow where (x_c) was assigned in the range $[l, u]$.

Let α be the range of (x_c) when $(x_c) \in [l, l]$. Let β be the range of (x_c) when $(x_c) \in [u, u]$. By Equation 1,

$$\alpha = [l^-(x_c), l^+(x_c)]$$
$$\beta = [u^-(x_c), u^+(x_c)]$$

We can show that $u^-(x_c) \leq l^+(x_c)$.

$$u - l \leq^+ (x_c) -^- (x_c)$$
$$u +^- (x_c) \leq l +^+ (x_c)$$
$$u^-(x_c) \leq l^+(x_c)$$

The lower bound of β is less than the upper bound of α, thus $\alpha \cap \beta$. An observation $(x_c) \in [u^-(x_c), l^+(x_c)]$ could be the result of $(x_c) = [l, l]$, $(x_c) = [u, u]$, or any value $(x_c) \in [l, u]$. Observations provide no information about the underlying contingent constraint, therefore we ignore (x_c).

None of our decisions depend on $\xi(x'_c)$, thus no observation space mapping to S is necessary.

If $u - l \geq^+ (x_c) -^- (x_c)$, we can emulate (x_c) under minimal information by replacing the bounds of x_c with $x'_c \in [l^+(x_c), u^-(x_c)]$ and letting $\gamma(x'_c) = 0$.

Under Lemma 1.3, observations (x_c) are guaranteed to narrow the range of (x$_\mathrm{c}$)$.

We have the same ranges for α and β as in Lemma 1.3, however we can show that $u^-(x_c) \geq l^+(x_c)$ instead.

$$u - l \geq^+ (x_c) -^- (x_c)$$
$$u +^- (x_c) \geq l +^+ (x_c)$$
$$u^-(x_c) \geq l^+(x_c)$$

Thus, receiving an observation is guaranteed to narrow the derived range of (x_c). The transformation tightens the range of x'_c to one where there is maximum ambiguity of the assignment of x_c while guaranteeing an execution strategy for any assignment of $x_c \in [l, u]$.

Based on the derivations above, it is clear that $\mu(x'_c)$ maps to the observation space where there is ambiguity as to the projection of $(x_c) \in [l, u]$. We must also show that $\mu(x'_c)$ has mappings to the extrema of $\xi(x_c)$. We start with the earliest (x'_c).

$$(x_c') = l^+(x_c) = l +^+ (x_c)$$

We show that that this assignment of $\xi(x_c')$ can be modeled as the following observation in S.

$$(x_c) \in [l +^- (x_c), l +^+ (x_c)]$$
$$(x_c) \in [l, l] + \Gamma(x_c)$$

It is possible that $\xi(x_c) = [l, l]$. As such, all observations in (x_c) may share the same execution strategy because the underlying temporal constraints depend on $\xi(x_c)$, not (x_c') or $\Gamma(x_c)$. We may expand the range of the observation space when we map to S with $\mu(x_c')$.

$$\mu : l^+(x_c) \to \omega_v(x_c)$$
$$\omega_v(x_c) = [l +^- (x_c), l +^+ (x_c)]$$

We see that μ has a valid observation space mapping to the minimum of the range of $\omega_v(x_c)$. We use the same argument for the maximum.

$$(x_c') = u +^- (x_c)$$

Observations anywhere in $[u +^- (x_c), u +^+ (x_c)]$ may share execution strategies because, it is possible that in all cases, $\xi(x_c) = [u, u]$. We may then expand the range of the observation space when we map to S.

$$\mu : u^-(x_c) \to \omega_v(x_c)$$
$$\omega_v(x_c) = [u +^- (x_c), u +^+ (x_c)]$$

Thus, $\mu(x_c')$ maps to the maximum of the range of $\omega_v(x_c)$. The transition creates assignments in S' that map to the entire $\omega_v(x_c)$ in S.

This concludes the modifications required to transform a contingent event $x_c \in X_c$ in S to its equivalent $x_c' \in X_c$ in S'. What remains is to address the transformation of requirement links, $x_r \in X_r$, in S such that their transformed equivalents, $x_r' \in X_r$ in S', express the same execution semantics in S' as they did in S. We will demonstrate the correctness of the transformations after Lemma 1.3.

If we have contingent link XC with duration $[l, u]$, outgoing requirement link CZ with duration $[u, v]$ with an unobservable C, and contingent link CY

with range $[^{-}(x_c),^{+}(x_c)]$, we can emulate the role of the original requirement link during execution with a new link YZ with bounds $[u-max(^{-}(x_c), XY-u), v-min(^{+}(x_c), XY-l)]$, where XY is the true duration of XY.

See Figure **??**c for reference. From an execution perspective, X and Y are the only events that can give us any information that we can use to reason about when to execute Z (since C is wholly unobservable).

If we execute Z based on what we learn from Y, then we use our information from Y to make inferences about the true durations of XC and CY based on XY. We know that the lower-bound of CY is at least $XY - b$ and that its upper-bound is at most $XY - a$. But we also have the a priori bounds on the contingent link that limit its range to $[^{-},^{+}]$. Taken together, during execution we can infer that the true bounds of CY are $[max(^{-}, XY-b), min(^{+}, XY-a)]$. Since we have bounds only on Z's execution in relation to C, we can then infer a requirement link YZ with bounds $[u - max(^{-}, XY-b), v - min(^{-}, XY-a)]$.

If we try to execute Z based on information we have about X, we must be robust to any possible value assigned to XC. This means that we would be forced to draw a requirement link XZ with bounds $[u+b, v+a]$. But we know that $u-max(^{-}, XY-b) \leq u+b-XY$ and $v-min(^{-}, XY-a) \geq v+a-XY$, which means that the bounds we derived from Y are at least as expressive as the bounds that we would derive from X.

Since we have a local execution strategy that depends on the real value of XY, we can try to apply this strategy to the contingent link that we restricted in Lemma 1.3, in order to repair the remaining requirement links.

If we have an outgoing requirement link CZ with duration $[u, v]$, where C is a contingent event, we can emulate the role of the original requirement link by replacing its bounds with $[u -^{-}(x_c), v -^{+}(x_c)]$.

See Figure **??**d for reference. If we directly apply the transformation from Lemma 1.3 and Figure **??**c to our original STNU, we introduce complexity through the need to reason over *min* and *max* operations in our link bounds. However, from Lemma 1.3, we know that in a controllability evaluation context, it is acceptable for us to simplify the XY link to a stricter range of $[a+^{+}, b+^{-}]$, instead of $[a+^{-}, b+^{+}]$. This means that for the purpose of evaluating controllability, we can assume $a+^{+} \leq XY \leq b+^{-}$. When we evaluate the requirement link YZ, we see $max(^{-}, XY-b) =^{-}$ and $min(^{+}, XY-a) =^{+}$. This gives us bounds of $[u-^{-}, v-^{+}]$ for the YZ requirement link as seen in Figure **??**d.

Lemma 1.3 handles outgoing requirement edges connected to contingent events. In addition, we must handle incoming edges.

If we have an incoming requirement link ZC with duration $[u, v]$, where C

is a contingent event, we can replace the bounds of the original requirement link with $[u +^{+} (x_c), v +^{-} (x_c)]$.

A requirement link ZC with bounds $[u, v]$ can be immediately rewritten as its reverse CZ with bounds $[-v, -u]$. After reversing the edge, we can apply Lemma 1.3 to get YZ with bounds $[-v-^{-}, -u-^{+}]$, which we can reverse again to get ZY with bounds $[u+^{+}, v+^{-}]$.

After applying Lemma 1.3, despite the limited expected range of assignments in x'_c in S' compared to x_c in S, we can show that Lemma 1.3 guarantees a satisfying schedule for any $(x_c) \in [l^{-}(x_c), u^{+}(x_c)]$ using an $\mathcal{S}$ that employs *buffering* and *imagining* contingent events.

Buffering

Buffering a contingent event x_c is an execution strategy where, if x_c is observed earlier than the lower bound of the observation space $(x_c) < \omega_f^{-}(x'_c)$, we assign $\xi(x'_c)$ to the lower bound of the observation space, $\xi(x'_c) = \omega_f^{-}(x'_c)$.

Imagining

Imagining a contingent event x_c is an execution strategy where, if x_c is observed later than the upper bound of the observation space, $(x_c) > \omega_f^{+}(x'_c)$, we assign $\xi(x'_c)$ to the upper bound of the observation space, $\xi(x'_c) = \omega_f^{+}(x'_c)$.

If S' is fixed-delay controllable after applying Lemmas 1.3, 1.3, and 1.3 to contingent event Y with following requirement event Z, there is a valid $\mathcal{S}$ for any observation in the observation space of S, $\omega_v(Y) = [a^{-}(Y), b^{+}(Y)]$.

We first note the observation space of S' is a subinterval of the original observation space of S, $\omega_f(Y') \subset \omega_v(Y)$, and there are two distinct ranges of observations that are not in $\omega_f(Y')$.

$$\begin{aligned}
&\omega_f(Y') = [a +^{+} (Y), b +^{-} (Y)];\ \omega_v(Y) = [a +^{-} (Y), b +^{+} (Y)]\\
&\omega_f(Y') \not\supset [a +^{-} (Y), a +^{+} (Y))\quad (\textit{"Early" observations})\\
&\omega_f(Y') \not\supset (b +^{+} (Y), b +^{+} (Y)]\quad (\textit{"Late" observations})
\end{aligned}$$

We address the early observations first. The range of early assignments of $\xi(Y)$ in S that we care about are the ones that could produce an observation $(Y) \leq a +^{+} (Y)$, which is $\xi(Y) = [a, a + (^{+}(Y) -^{-} (Y))]$. We rewrite the range of early assignments as $\xi(Y) = a + (^{+}(Y) -^{-} (Y)) - \epsilon$, where $0 \leq \epsilon \leq (^{+}(Y) -^{-} (Y))$. By the semantics of S, the range of assignments of $\xi(Z)$ is then,

$$\xi(Z) = [a + (^+(Y) -^- (Y)) - \epsilon, a + (^+(Y) -^- (Y)) - \epsilon] + [u, v]$$
$$\xi(Z) = [a + u + (^+(Y) -^- (Y)) - \epsilon, a + v + (^+(Y) -^- (Y)) - \epsilon]$$

The earliest assignment of Y' in S' is $\xi(Y') = a +^+ (Y)$. By the semantics of S', the range of assignments of $\xi(Z')$ is then,

$$\xi(Z') = [a +^+ (Y), a +^+ (Y)] + [u -^- (Y), v -^+ (Y)]$$
$$\xi(Z') = [a + u + (^+(Y) -^- (Y)), a + v]$$

We see that $\xi(Z') \subseteq \xi(Z)$ for any ϵ, meaning the execution strategy when $\xi(Y') = a +^+ (Y)$ results in a valid assignment of $\xi(Z)$ for all early observations of $\xi(Y)$. We are safe to buffer early observations to $\xi(Y') = a +^+ (Y)$.

We use the same argument for imagining late observations. The range of late assignments of $\xi(Y)$ in S that we care about are the ones that could produce an observation $(Y) \geq b +^- (Y)$, which is $\xi(Y) = b - (^+(Y) -^- (Y)) + \epsilon$. By the semantics of S, the range of assignments of $\xi(Z)$ is then,

$$\xi(Z) = [b - (^+(Y) -^- (Y)) + \epsilon, b - (^+(Y) -^- (Y)) + \epsilon] + [u, v]$$
$$\xi(Z) = [b + u - (^+(Y) -^- (Y)) + \epsilon, b + v - (^+(Y) -^- (Y)) + \epsilon]$$

The last assignment of Y' in S' is $\xi(Y') = b +^- (Y)$. By the semantics of S', the range of assignments of $\xi(Z')$ is then,

$$\xi(Z') = [b +^- (Y), b +^+ (Y)] + [u -^- (Y), v -^+ (Y)]$$
$$\xi(Z') = [b + u, b + v - (^+(Y) -^- (Y))]$$

We see that $\xi(Z') \subseteq \xi(Z)$ for any ϵ, meaning the execution strategy when $\xi(Y') = b +^- (Y)$ results in a valid assignment of $\xi(Z)$ for all late observations of $\xi(Y)$. In practice, there is no reason to wait until after $(Y) = b +^- (Y)$ to receive a late observation. As soon as we see the clock has reached $b +^- (Y)$, we are safe to imagine that (Y) has been received.

We can examine a concrete example of Lemmas 1.3, 1.3, and 1.3 to show equivalence in the transformation from Figure **??**a to **??**d. We start by building an example of **??**a. Let $XC[2, 5]$ with $(C) \in [1, 2]$ and $CZ[11, 20]$. If we learn of event C at time 4, then one possibility is that the realized duration of C could have been 2 with an observation delay of 2. In this case, event Z must be executed in $[13, 22]$. However, if the realized duration of C were 3 with an observation delay of 1, then Z would fall in $[14, 23]$. Given we cannot distinguish between the possibilities, we take the intersection of the intervals, yielding $Z \in [14, 22]$. Likewise, if we learn of C at time 6, then C could have been realized at time 5 with an observation delay of 1 or it could have been realized at time 4 with an observation delay of 2. In the first case, Z must then fall in $[16, 25]$, while in the second, Z would fall in $[15, 24]$. The intersection yields $[16, 24]$.

By the semantics represented in Figure **??**d, we can build an equivalent network with $\gamma(Y) = 0$ by setting $XY[4, 6]$ and $YZ[10, 18]$. If Y is observed at time 4, Z must be executed in $[14, 22]$. If Y is observed at time 6, Z then must be executed in $[16, 24]$. The execution semantics for both cases match the equivalent networks from **??**a described above.

1.4 Dynamic Scheduling through Real-Time Execution Decisions

An STNU, S, that exhibits dynamic controllability can be *scheduled* dynamically (or *online*). At a high-level, dynamic scheduling is the process of mapping the history of event assignments to the execution time of future free events. We follow the scheduling work by Hunsberger (Hunsberger, Luke, 2013, Hunsberger, Luke, 2016), which describes an $O(N^3)$ procedure, FAST-EX, for dynamic scheduling of STNUs. At its core is the notion of *Real-Time Execution Decisions* (RTEDs), which map a timepoint to a set of requirement events to be executed and are generated based on *partial schedules* of STNUs being executed. `WAIT` decisions may also be produced, reflecting the need to wait for the assignment of a contingent event before continuing. RTED-based scheduling applies a dynamic programming paradigm by first creating a dispatchable form of temporal constraints offline, updating the dispatchable form as the partial schedule is updated online, and querying the dispatchable form online to quickly find the next free event to schedule (Hunsberger, Luke, 2009).

Real-Time Execution Decisions (Hunsberger, Luke, 2009)
A *Real-Time Execution Decision* is a two-tuple $\langle t, \chi \rangle$, where:

- t is a time with domain $\mathbb{R}$,
- χ is a set of $x_r \in X_r$ to be executed at time t

Partial Schedule

A *Partial Schedule* for an STNU is a mapping $\xi : t \rightarrow \mathbb{R}$, where t is a proper subset of timepoints in X with domain $\mathbb{R}$.

A partial schedule is the set of assignments *so far* during execution. They can be represented by $\xi_t = \{(A, \xi(A)\}$ notation, meaning that by some time t, event A has been scheduled at time $\xi(A)$. By definition, all STNU executions begin with a partial schedule $\xi_0 = \emptyset$, meaning no timepoints have been assigned at time 0. As timepoints are executed, the partial schedule grows. For example, if event A is assigned to 1 and B to 2, then at time \$2, $\xi_2 = \{(A, 1), (B, 2)\}$.

The dispatchable form employed by FAST-EX is the *AllMax* distance graph, which is produced by the Morris $O(N^4)$ DC-checking procedure (Morris, Paul, 2006).

AllMax Distance Graph (Morris, Paul and Muscettola, Nicola, 2005)

The *AllMax* distance graph is a distance graph exclusively consisting of unlabeled and upper-case edges.

The key idea of FAST-EX is maintaining accurate distances from a zero point, Z, of the graph to all events. At the outset of execution, all events from S are present as nodes in *AllMax*. As contingent events are observed, *AllMax* performs update steps using Dijkstra Single Source/Sink Shortest Path (SSSP) to maintain distances to unexecuted events, while also removing executed events. We include pseudo-code of the update step in Figure 1.4.

Returnreturn InputInput OutputOutput AlgorithmFAST-EX UPDATE InitializeInitialization IfElseIfElseifthenelse ifelseendif Time t; Set of newly executed events `Exec` $\subseteq X_e \cup X_r$; AllMax Graph G; Distance matrix D, where $D(A, B)$ is the distance from A to B Updated D each continent event $C \in$ `Exec` Remove each upper-case edge, $YAC : -w$, labled by C Replace each edge from Y to Z with the strongest replacement edge each event $E \in$ `Exec` Add lower-bound edge $EZ{-}t$ For each event X, update $D(X, Z)$ using Dijkstra Single-Sink Shortest Paths each event $E \in$ `Exec` Add upper-bound edge ZEt For each event X, update $D(Z, X)$ using Dijkstra Single-Source Shortest Paths Algorithm for updating distances for all events in relation to Z upon the execution of an event. Adapated from 3[3], Fig. 19.

Given an accurate distance matrix, we find the next RTED as follows. Let U_x be the set of unexecuted free timepoints. If U_x is empty, then the RTED is to `WAIT`. Otherwise, we find the lower bound of the earliest executable time point and the set of executable events associated with it.

$$t = \min\{-D(X, Z) \mid X \in U_x\}$$
$$\chi = \{X \in U_x \mid -D(X, Z) = t\}$$

We cannot execute events in the past. Let `now` be the current time, i.e. the last timepoint captured in ξ. It is possible that $t \leq$ `now`, in which case we must reassign t to guarantee that $t >$ `now`. To do so, we update t as follows, where t_U is earliest *upper* bound of the executable timepoints,

$$t_U = \min\{D(Z, X) \mid X \in U_x\}$$
$$t = \frac{\texttt{now} + t_U}{2}$$

So long as $t_U >$ `now`, we know that the reassignment of t ensures $t >$ `now`.