# Fixing the Semantics for Dynamic Controllability and Providing a More Practical Characterization of Dynamic Execution Strategies

Luke Hunsberger
Vassar College
Poughkeepsie, NY 12604-0444 USA
hunsberg@cs.vassar.edu

## Abstract

*Morris, Muscettola and Vidal (MMV) presented an algorithm for checking the dynamic controllability (DC) of temporal networks in which certain temporal durations are beyond the control of the planning agent. Their DC-checking algorithm is based on rules for inferring new constraints based on the real-time context within which execution decisions must be made. This paper presents a counter-example to demonstrate that some of the inference rules are, in fact, not sound. The paper fixes the problem by strengthening the definition of dynamic execution strategies to correctly capture the central prohibition against advance knowledge of future events. The new definition enables MMV's soundness proof to go through with minimal changes. It then uses the stronger definition to derive an equivalent, alternative characterization of dynamic execution strategies that highlights the real-time execution decisions that a planning agent must make. The procedural strategy used by MMV in their completeness proof is shown to satisfy the stronger definition, thus ensuring that the DC-checking algorithm is also complete with respect to the stronger definition. As a result, the paper puts MMV's DC-checking algorithm on a more solid theoretical foundation, while also providing a more practical characterization of dynamic execution strategies.*

## 1. Introduction

Dechter et al. [1] introduced Simple Temporal Networks (STNs), a practical formalism for representing and managing time-points and temporal constraints. Vidal and Ghallab [8] argued that certain processes are only initiated by a planning agent, but their durations are beyond the agent's control. They augmented STNs to include *contingent constraints* and defined the *controllability* of such networks. Vidal and Fargier [6, 7], and later Vidal [5], presented more concise definitions of networks with contingent constraints—called *Simple Temporal Networks with Uncertainty* (STNUs)—and three kinds of controllability: *weak, strong* and *dynamic*. Of these, dynamic controllability has the most practical use. Loosely speaking, a network is dynamically controllable (DC) if there exists a *dynamic execution strategy* (DES) that an agent can use to execute the time-points under its control that will guarantee the consistency of the network no matter how the contingent durations turn out. Crucially, a DES is not allowed to depend on advance knowledge of future events. In particular, real-time execution decisions cannot depend on the values of contingent durations that have not yet completed.

Morris, Muscettola and Vidal [3]—henceforth MMV—further refined the semantics of dynamic controllability and presented a pseudo-polynomial-time algorithm for checking the dynamic controllability of arbitrary STNUs. Their proof of the soundness of their DC-checking algorithm is based on rules for inferring new constraints that reflect the real-time context within which execution decisions must be made. Their completeness proof demonstrates that any network accepted by the algorithm has a viable DES.

Later, Morris and Muscettola [4] presented a more concise, $O(n^5)$-time DC-checking algorithm, and Morris [2] presented an even faster, $O(n^4)$-time algorithm.[1] The correctness of the later algorithms is based on the MMV semantics and the correctness of the MMV algorithm; thus, this paper focuses on the MMV semantics and algorithm.

### 1.1. This paper

This paper begins by presenting a counter-example to the soundness of MMV's DC-checking algorithm. The problem is two-fold. First, MMV's definition of a DES—and hence of dynamic controllability—does not adequately capture the prohibition against advance knowledge of future events that lies at the heart of our pre-theoretic notion of dynamic controllability. Second, MMV's soundness proof relies on a

[1] $n$ is the number of time-points in the network.

property of DESs that is not entailed by their definition.

The paper fixes the problem by augmenting MMV's definition of DES to include a stronger property that correctly captures the above-mentioned prohibition. It then proves that MMV's soundness proof, with minimal changes, goes through using the new definitions. *No changes to the DC-checking algorithm are required; it is sound with respect to the new definitions.*

Finally, the paper uses the new definition of dynamic execution strategies to derive a more practical, alternative characterization of such strategies. The new characterization is based directly on the kinds of real-time execution decisions that a planning agent must make. Each real-time decision has one of two forms. They can be glossed as "wait until something happens" or "if nothing happens before time $\tau$, then execute the time-points in the set $\chi$". Such strategies can be derived using the sorts of incremental computations that are reflected in the *procedural* strategy used by MMV in their completeness proof. This paper demonstrates that their procedural strategy satisfies the new definition of a dynamic execution strategy, thereby ensuring that the *existing* DC-checking algorithm is also complete with respect to the new definition of dynamic controllability.

Although some of the elements of the alternative characterization of strategies developed in this paper bear some similarity to notions defined by Vidal and Fargier [6, 7], they are quite distinct. In addition, the strategies based on real-time execution decisions are novel, their properties are rigorously analyzed, and they are proven to be equivalent to the (revised) MMV definitions.

## 1.2. Summary of MMV's approach

This section summarizes the definitions used by MMV, but also draws from Vidal [5] and Vidal and Fargier [7].

**Simple Temporal Network (STN).** An STN is a 4-tuple, $\langle N, E, l, u\rangle$, where $N$ is a *finite* set of nodes (or time-points), $E$ is a set of directed edges, and $l : E \to \mathbb{R} \cup \{-\infty\}$ and $u : E \to \mathbb{R} \cup \{\infty\}$ are functions that map edges to lower and upper bounds, respectively. An edge, $e \in E$, from the time-point $X$ to the time-point $Y$, represents the constraint, $Y - X \in [l(e), u(e)]$, indicated as follows:

$$\begin{tikzcd} X \bullet \arrow[r, "{[l(e), u(e)]}"] & \bullet Y \end{tikzcd}$$

The edges are also called *links*. One of the time-points, called the *zero time-point* (or $Z$), is fixed at the value 0. All other time-points are constrained to occur after $Z$.

**Simple Temporal Network with Uncertainty (STNU).** An STNU is a 5-tuple, $\langle N, E, l, u, \mathcal{C}\rangle$, where $N, E, l$ and $u$ are as in an STN, and $\mathcal{C} \subseteq E$ is a subset of the edges: the *contingent links.* (The rest of the edges are called *requirement links.*) For each contingent link, $e \in \mathcal{C}$, the bounds are assumed to satisfy $0 < l(e) < u(e) < \infty$. If $e$ is a contingent link from $A$ to $C$, then $C$ is called a *contingent time-point,* and $A$ is called the *activation time-point* for $C$. In this paper, contingent links are indicated as follows:

$$\begin{tikzcd} A \bullet \arrow[r, dashed, "{[l(e), u(e)]}"] & \bullet C \end{tikzcd}$$

Let $N^c \subseteq N$ denote the set of contingent time-points and $N^x = N - N^c - \{Z\}$ the set of *executable* time-points.

To *execute* a time-point means to fix its value to the current time. The planning agent directly controls the execution of only the executable time-points. Nature is presumed to control the duration of each contingent link. A contingent time-point, $C$, is said to be *activated* if its activation time-point, $A$, has been executed.

**Situations and Projections.** Let $\mathcal{N}$ be an STNU whose contingent links are $e_1, \ldots, e_q$ and whose corresponding labels are $[x_1, y_1], \ldots, [x_q, y_q]$. The *space of situations* for $\mathcal{N}$ is the cross-product, $\Omega_{\mathcal{N}} = [x_1, y_1] \times \ldots \times [x_q, y_q]$. Each $\omega \in \Omega_{\mathcal{N}}$ is called a *situation.* Each situation specifies durations for *all* of the contingent links in $\mathcal{N}$.

For a situation, $\omega = (\omega_1, \omega_2, \ldots, \omega_q)$, the *projection*, $\mathcal{N}_\omega$, is the STN (not STNU) derived from $\mathcal{N}$ by replacing each contingent link, $e_i$, with a requirement link labeled by $[\omega_i, \omega_i]$, thereby fixing its duration to the value $\omega_i$.

**Schedules.** Given an STNU, $\mathcal{N} = \langle N, E, l, u, \mathcal{C}\rangle$, a *schedule* is a mapping $T : N \to \mathbb{R}$ (i.e., a complete set of variable assignments for *all* of the time-points in $N$). For convenience, the shorthand $T_t$ is used instead of $T(t)$ to denote the *time* at which $t$ is executed according to the schedule $T$. The set of schedules for $\mathcal{N}$ is denoted by $\mathcal{T}_{\mathcal{N}}$. A schedule $T$ is *consistent* with $\mathcal{N}$ (resp., a projection $\mathcal{N}_\omega$) if the assignments in $T$ satisfy all of the constraints in $\mathcal{N}$ (resp., $\mathcal{N}_\omega$). Henceforth, we restrict attention to schedules in which $T_{A_i} < T_{C_i}$ for each contingent link, $A_iC_i$.

**Pre-histories.** Given a schedule $T$ for an STNU $\mathcal{N}$, and a time-point $x \in N$, the *pre-history of $x$ with respect to $T$* is:

$$T_{<x} = \{(C_i, T_{C_i} - T_{A_i}) \ : \ A_iC_i \in \mathcal{C},\ T_{C_i} < T_x\}$$

Thus, $T_{<x}$ specifies the durations of all contingent links that execute *before* $x$ according to the schedule $T$.

**Execution Strategies.** An *execution strategy* for an STNU $\mathcal{N}$ is a mapping, $S : \Omega_{\mathcal{N}} \to \mathcal{T}_{\mathcal{N}}$, from situations to schedules.[2] $S$ is called *viable* if for each situation $\omega \in \Omega_{\mathcal{N}}$,

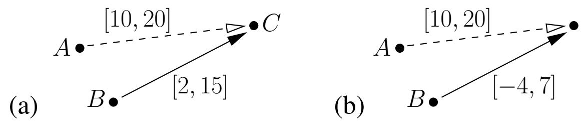

**Figure 1. Sample triangular networks**

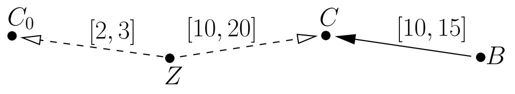

**Figure 2. A counter-example, $\mathcal{N}^!$**

the schedule $S(\omega)$ is consistent with the projection $\mathcal{N}_\omega$. $S$ is called a *dynamic execution strategy* (DES) for $\mathcal{N}$ if

$$[S(\omega')]_{<x} = [S(\omega'')]_{<x} \quad \Longrightarrow \quad [S(\omega')]_x = [S(\omega'')]_x$$

for all situations, $\omega', \omega''$, and *executable* time-points, $x$. Thus, if two schedules generated by a DES have identical pre-histories for $x$, they must assign the same value to $x$.

**Dynamic Controllability.** An STNU, $\mathcal{N}$, is *dynamically controllable* (DC) if there exists a *viable* DES for $\mathcal{N}$.

**The DC-Checking Algorithm.** MMV's DC-checking algorithm infers constraints that any viable DES must satisfy. Some of the constraints are ternary constraints, called *waits*. For example, the networks in Fig. 1 each contain a contingent link, $AC$, and a requirement link, $BC$. For (a), the DC-checking algorithm infers the constraint, $B-A \in [5, 8]$. For (b) it infers a *wait*: *while $C$ is unexecuted, $B$ must wait until at least* 13 *units after $A$* (i.e., $B - A \geq 13$). The algorithm also propagates waits. MMV prove that the rules for generating and propagating waits are *sound* (i.e., the resulting waits must be satisfied by any viable DES). They also prove that any STNU accepted by the algorithm has a viable DES (i.e., that the algorithm *complete*).

## 2 Counter-example

The STNU, $\mathcal{N}^!$, shown in Fig. 2, contains two contingent links—$ZC_0$ and $ZC$—and one requirement link: $BC$. Since $Z$ is fixed at 0, the duration of $ZC$ is the same as the value of $C$; and the duration of $ZC_0$ equals the value of $C_0$.

$\mathcal{N}^!$ does not satisfy our intuitive notion of dynamic controllability since there is no safe time to execute $B$. For example, executing $B$ at time 0 risks violating the constraint, $C - B \leq 15$, should $C$ happen to execute after time 15. But executing $B$ sometime after 0 risks violating the constraint, $C - B \geq 10$, should $C$ happen to execute at time 10.

For $\mathcal{N}^!$, the DC-checking algorithm infers the (impossible) constraint, $C - B \in [5, 0]$, and thus rejects $\mathcal{N}^!$. But the strategy, $S^!$, defined below, is a viable DES for $\mathcal{N}^!$.

> For any situation, $\omega$, if $ZC \leq 15$ in $\omega$,
> let $[S^!(\omega)]_B = 0$; otherwise, let $[S^!(\omega)]_B = 5$.

$S^!$ clearly violates the important prohibition against decisions that depend on advance knowledge of future events. In particular, the execution time for $B$, which must be chosen before $C$ executes, depends on the execution time of $C$. However, this strategy is viable, since all of the schedules it generates are consistent with the corresponding projections. Furthermore, it is a DES. To see this, notice that the pre-history for any schedule that assigns $B$ the value 0 is necessarily empty, but the pre-history for any schedule that assigns $B$ the value 5 necessarily contains information about $C_0$—since $ZC_0$ is constrained to occur within $[2, 3]$. Thus, any schedules having the same pre-histories for $B$ assign the same value to $B$, which is what the DES definition requires. Thus, according to MMV's definition, $\mathcal{N}^!$ is DC, although the DC-checking algorithm rejects it. Thus, $\mathcal{N}^!$ contradicts the soundness of the DC-checking algorithm.

This counter-example employs what should be an irrelevant contingent time-point, $C_0$, to take advantage of a loophole in the DES definition. The presence of $C_0$ causes the different sets of schedules to have different pre-histories for $B$, thereby escaping the main DES requirement. In so doing, the execution time of $B$ is allowed to depend on advance knowledge of the future execution of $C$.

In view of this contradiction, MMV's proof of the soundness of the DC-checking algorithm must have an error. In particular, their proof depends on the following claim:[3]

> Let $S$ be a DES, $\omega$ a situation, $t \neq C$ a time-point, and $AC$ a contingent link where $C \notin [S(\omega)]_{<t}$. Let $\omega'$ be a situation that differs from $\omega$ only in its value for the duration of $AC$—but such that $C \notin [S(\omega')]_{<t}$. Then $[S(\omega')]_t = [S(\omega)]_t$.

Although this property holds in the case of triangular networks, it does not hold for $\mathcal{N}^!$. In particular, $S^!$ is a DES, $\omega = (2, 10)$ is a situation (i.e., $C_0 = 2, C = 10$), $[S^!(\omega)]_B = 0$, and $C \notin [S^!(\omega)]_{<B} = \emptyset$. However, $\omega' = (2, 20)$ is a situation that differs only in the value it assigns to $C$, $C \notin [S^!(\omega')]_{<B} = \{(C_0, 2)\}$, and yet $[S^!(\omega')]_B = 5 \neq 0 = [S^!(\omega)]_B$.

[2]MMV define strategies as mappings from projections to schedules, not situations to schedules. However, since situations correspond one-to-one to projections, this difference is purely superficial.

[3]In MMV's discussion of the *Precede* case: "Since $C$ is not in $T_{<B}$ or $T_{<A}$, $T_B$ cannot depend on $AC$. Therefore $T_A$ and $T_B$ are unchanged if the projection is mutated to a projection $p'$ where $AC$ equals $y$." A similar assumption is made in their Lemma 1, which covers propagation of waits.

## 3. Closing the loophole

The problem with the DES definition is that it compares pre-histories relative to time-point *variables.* The revised DES definition, given below, instead compares pre-histories relative to *fixed times,* $k \in \mathbb{R}$. To facilitate comparison, the revised definitions for DES and DC are marked by asterisks.

**Pre-histories relative to a number** $k$**.** Given an STNU $\mathcal{N}$, a schedule $T$, and some $k \in \mathbb{R}$, then $T^{<k}$ denotes the *pre-history of* $T$ *relative to* $k$, which specifies the durations of the contingent links in $\mathcal{N}$ that finish before $k$ in $T$:

$$T^{<k} = \{(C_i, T_{C_i} - T_{A_i}) : A_iC_i \in \mathcal{C},\ T_{C_i} < k\}$$

**Dynamic Execution Strategy**$^*$**.** A strategy, $S$, is called a *dynamic execution strategy*$^*$ (DES$^*$) if for any situations, $\omega', \omega'' \in \Omega_{\mathcal{N}}$, and executable time-point $x$ in $\mathcal{N}$: if $[S(\omega')]_x = k$ and $[S(\omega')]^{<k} = [S(\omega'')]^{<k}$, then $[S(\omega'')]_x = k$. In other words, if the strategy $S$ in the situation $\omega'$ assigns the value $k$ to the executable time-point $x$, then $S$ must also assign $k$ to $x$ for any other situation $\omega''$ whose pre-history *relative to* $k$ matches that of $\omega'$.

**Dynamic Controllability**$^*$**.** An STNU is called *dynamically controllable*$^*$ (DC$^*$) if there is a *viable* DES$^*$ for it.

Although the strategy, $S^!$, is a DES for the network, $\mathcal{N}^!$, it is not a DES$^*$. To see this, first note that $[S^!(\omega)]^{<0} = \emptyset$ for all situations $\omega$, since no contingent time-point can execute before 0. Next, note that for some $\omega'$, $[S^!(\omega')]_B = 0$. Thus, for $S^!$ to be a DES$^*$ would require $[S^!(\omega)]_B = 0$ for *all* situations $\omega$. However, for some $\omega''$, $[S^!(\omega'')]_B = 5$.

In general, being a DES$^*$ requires a strategy to make the same decisions about executable time-points in the same real-time contexts. As will be shown subsequently, the existing DC-checking algorithm is both sound and complete with respect to the new DES$^*$ and DC$^*$ definitions.

**Lemma 1.** Let $S$ be a DES$^*$, and $\omega'$ and $\omega''$ situations such that $S(\omega') \neq S(\omega'')$. Let $k_0 \in \mathbb{R}$ be the first time at which $S(\omega')$ and $S(\omega'')$ differ. Then $S(\omega')$ and $S(\omega'')$ agree about which *executable* time-points execute at $k_0$; however, for some contingent link, $A_iC_i$, one of $S(\omega')$ and $S(\omega'')$ says $C_i$ executes at $k_0$, while the other says $C_i$ executes *after* $k_0$. In any case, $[S(\omega')]_{A_i} = [S(\omega'')]_{A_i} < k_0$.

**Proof 1.** By construction, $[S(\omega')]^{<k_0} = [S(\omega'')]^{<k_0}$ which, given that $S$ is a DES$^*$, implies that $S(\omega')$ and $S(\omega'')$ agree about the executable time-points that execute at $k_0$. Thus, $S(\omega')$ and $S(\omega'')$ can only disagree about the execution of *contingent* time-points at $k_0$. Without loss of generality, let $C_i$ be a contingent time-point such that $[S(\omega')]_{C_i} = k_0$, but $[S(\omega'')]_{C_i} > k_0$. Let $A_i$ be the activation time-point for $C_i$. Then $[S(\omega')]_{A_i} < [S(\omega')]_{C_i} = k_0$ implies that $[S(\omega')]_{A_i} = [S(\omega'')]_{A_i}$, by the choice of $k_0$. ∎

**Lemma 2.** If $S$ is a DES$^*$ for $\mathcal{N}$, then it is a DES for $\mathcal{N}$.

**Proof 2.** Suppose $[S(\omega')]_{<x} = [S(\omega'')]_{<x}$ for some $\omega'$ and $\omega''$, and executable $x$. Let $k_0$, $A_i$ and $C_i$ be as in Lemma 1.[4] Since $S(\omega')$ and $S(\omega'')$ agree about $A_i$, but not $C_i$, it follows that $C_i \notin [S(\omega')]_{<x} = [S(\omega'')]_{<x}$. Thus, $C_i$ occurs at or after $x$ in both $S(\omega')$ and $S(\omega'')$. Thus, $x$ occurs before $k_0$ in both. Thus, $[S(\omega')]_x = [S(\omega'')]_x$. ∎

**Lemma 3.** Suppose $[S(\omega')]^{<k} = [S(\omega'')]^{<k}$ for a DES$^*$ $S$, and some $k, \omega'$ and $\omega''$. Then for *all* time-points $t$, if $[S(\omega')]_t < k$ or $[S(\omega'')]_t < k$, then $[S(\omega')]_t = [S(\omega'')]_t$. Furthermore, for all *executable* time-points $x$, if $[S(\omega')]_x \leq k$ or $[S(\omega'')]_x \leq k$, then $[S(\omega')]_x = [S(\omega'')]_x$. Thus, if $S(\omega')$ and $S(\omega'')$ have the same pre-histories relative to $k$, then they must agree about the execution times of *all* time-points *before* $k$, and all *executable* time-points *at* $k$.

**Proof 3.** Let $k_0$, $A_i$ and $C_i$ be as in Lemma 1. By choice of $k_0$, $S(\omega')$ and $S(\omega'')$ agree about all executions before $k_0$. If $k_0 < k$, then $C_i$ executes before $k$ in $S(\omega')$ or $S(\omega'')$. But then $C_i \in [S(\omega')]^{<k} = [S(\omega'')]^{<k}$, which implies $[S(\omega')]_{C_i} = [S(\omega'')]_{C_i}$, contradicting the choice of $C_i$. Thus, $k \leq k_0$, and $S(\omega')$ and $S(\omega'')$ agree about *all* time-points executing *before* $k$. Finally, suppose $[S(\omega')]_x = k$ or $[S(\omega'')]_x = k$ for some executable $x$. If $k < k_0$, then $[S(\omega')]_x = [S(\omega'')]_x$ by the choice of $k_0$; if $k = k_0$, then $[S(\omega')]_x = [S(\omega'')]_x$ by Lemma 1. ∎

Lemma 4 shows that the DES$^*$ definition correctly captures the prohibition against advance knowledge of future events, which is the property on which MMV based their soundness proof for their DC-checking algorithm.

**Lemma 4.** Let $S$ be a DES$^*$, $\omega$ a situation, $t \neq C$ a time-point, and $AC$ a contingent link such that $C \notin [S(\omega)]_{<t}$. If $\omega'$ is the same as $\omega$, except that the duration of $AC$ is changed, and $C \notin [S(\omega')]_{<t}$, then $[S(\omega')]_t = [S(\omega)]_t$.

**Proof 4.** Let $k_0$, $A_i$ and $C_i$ be as in Lemma 1. Thus, $S(\omega)$ and $S(\omega')$ disagree about the duration of $A_iC_i$. Thus, $AC$ must be the link $A_iC_i$. Without loss of generality, suppose $[S(\omega)]_C = k_0$ and $[S(\omega')]_C > k_0$. Since $C \notin [S(\omega)]_{<t}$, $C$ must occur at or after $t$ in $S(\omega)$ (i.e., $[S(\omega)]_t \leq k_0$). If $t$ is executable, or if $t$ is contingent with $[S(\omega)]_t < k_0$, then

[4] Proofs of Lemmas 2, 3, 4 and 8 ignore trivial case, $S(\omega') = S(\omega'')$.

$[S(\omega)]_t = [S(\omega')]_t$ by Lemma 1. If $t$ is contingent with $[S(\omega)]_t = k_0$, then $S(\omega)$ and $S(\omega')$ must agree about the execution of $t$'s activation time-point. Furthermore, since $t \neq C$, $S(\omega)$ and $S(\omega')$ must agree about the duration of the contingent link ending in $t$. Thus, $[S(\omega)]_t = [S(\omega')]_t$. ∎

**Corollary 4.1.** MMV's DC-checking algorithm is sound with respect to the definitions of DES$^*$ and DC$^*$.

## 4. Alternative Characterization of a DES$^*$

Defining strategies as mappings from (complete) situations to (complete) schedules obscures the real-time features of typical execution scenarios. For example, an agent typically becomes aware of the unfolding situation only incrementally, over time. As more contingent durations complete, the space of possible situations contracts. In addition, when making real-time execution decisions, an agent knows the execution times of only those time-points that have already executed. Finally, the DES$^*$ definition obscures the *kinds* of execution decisions an agent can make.

This section introduces *partial schedules* to represent not only the contexts within which an agent must make real-time execution decisions, but also the *outcomes* of those decisions. A partial schedule specifies the execution times of some, but not all of the time-points. However, since partial schedules represent what has actually happened so far, it is important to restrict attention to partial schedules that *respect* (i.e., are consistent with) at least one situation.

Two kinds of *real-time execution decisions* (RTEDs) are defined: `WAIT` and $(\tau, \chi)$. These can be glossed as: "Wait until some contingent duration completes" or "If nothing happens before $\tau$, then execute the (executable) time-points in $\chi$." The *outcome* of an RTED depends on the situation, and is represented by a partial schedule that specifies the execution of one or more additional time-points. The outcome of a `WAIT` decision involves the execution of only contingent time-points; the outcome of a $(\tau, \chi)$ decision can involve the execution of contingent or executable time-points.

An *RTED-based strategy* is defined as a mapping from partial schedules to real-time execution decisions. This section proves that RTED-based strategies correspond one-to-one to DES$^*$s. In addition, an RTED-based strategy is used to verify that MMV's DC-checking algorithm is complete with respect to the new DES$^*$ and DC$^*$ definitions.

**Schedules$^*$.** Given an STNU, $\mathcal{N} = \langle N, E, l, u, \mathcal{C}\rangle$, a *schedule*$^*$ is a (possibly partial) mapping $T : N \rightarrow \mathbb{R}$. Let $Dom(T) \subseteq N$ denote the domain of $T$. If $Dom(T) = N$, then $T$ is a (complete) schedule as previously defined; otherwise, $T$ is a *partial* schedule. If $t \in Dom(T)$, then $t$ is said to *appear* in $T$. The shorthand notation, $T_t$ instead of $T(t)$, is also used for partial schedules. For convenience, $T$ may be viewed as a set of elements of the form, $(t, T_t)$. Let $\mu(T) = \max\{T_t : t \in Dom(T)\}$ denote the *maximum execution (MaxEx) time* of time-points appearing in $T$.

If $t \notin Dom(T)$, then $t$ is *unexecuted* in $T$. The set of time-points that are unexecuted in $T$ is denoted by $U(T)$. Similarly, $U^x(T), U^c(T)$ and $U^a(T)$ respectively denote the sets of *executable*, *contingent* and *activated* time-points that are unexecuted in $T$. Note that $U^a(T) \subseteq U^c(T)$, since only contingent time-points can be *activated.*

**Respect.** Let $T$ be a (possibly partial) schedule$^*$ for an STNU $\mathcal{N}$. Let $\omega = (\omega_1, \ldots, \omega_q) \in \Omega_{\mathcal{N}}$. *$T$ respects $\omega$* if for each contingent link, $A_iC_i$, one of the following holds:

(1) neither $A_i$ nor $C_i$ appear in $T$;

(2) only $A_i$ appears in $T$, and $T_{A_i} + \omega_i > \mu(T)$; or

(3) both $A_i$ and $C_i$ appear in $T$, and $T_{A_i} + \omega_i = T_{C_i}$.

For each $T$, the *set of situations respected by* $T$ is denoted by $\Omega(T)$. $T$ is called *respectful* if it respects at least one situation in $\Omega_{\mathcal{N}}$ (i.e., if $\Omega(T) \neq \emptyset$). If $T$ is both respectful and partial, it is called a *respectful, partial schedule* (RPS). A *strategy* $S$ is *respectful* if for each $\omega$, $S(\omega)$ respects $\omega$.[5]

**The `WAIT` Decision.** Let $T$ be some RPS for $\mathcal{N}$ such that $U^a(T)$ is non-empty (i.e., there is at least one contingent time-point that is activated, but not yet executed in $T$). Then `WAIT` is an allowable RTED for $T$.

**The Outcome of a `WAIT` Decision.** If $U^a(T) \neq \emptyset$ and $\omega \in \Omega(T)$ is a situation respected by $T$, then the *time* at which the *next contingent* time-point will execute (according to $T$ and $\omega$) is defined by:

$$tnc(T, \omega) = \min\{\, T_{A_i} + \omega_i : C_i \in U^a(T)\}$$

Since $U^a(T) \neq \emptyset$, $tnc(T, \omega)$ is well defined; and since $T$ respects $\omega$, $tnc(T, \omega) > \mu(T)$.

Next, let $\chi^a(T, \omega) \subseteq U^a(T)$ denote the *non-empty* set of activated contingent time-points that, according to $T$ and $\omega$, will execute next, at the time $tnc(T, \omega)$:

$$\chi^a(T, \omega) = \{C_i \in U^a(T) : T_{A_i} + \omega_i = tnc(T, \omega)\}$$

Then $\mathcal{O}(T, \omega, \texttt{WAIT})$ denotes the *outcome* of the `WAIT` decision for $T$ in the situation $\omega$, which is defined to be:

$$T \,\cup\, \{(C_i, tnc(T, \omega)) : C_i \in \chi^a(T, \omega)\}$$

Note that $\mathcal{O}(T, \omega, \texttt{WAIT})$ is a schedule$^*$ that augments $T$ to include all of the contingent time-points that execute at the time, $tnc(T, \omega)$. Thus, the MaxEx time for this outcome is $tnc(T, \omega)$. In addition, $T \subset \mathcal{O}(T, \omega, \texttt{WAIT})$.

[5]A viable strategy is necessarily respectful since $S(\omega)$ being consistent with $\mathcal{N}_\omega$ requires $S(\omega)$ to respect all durations in $\omega$; however, a respectful strategy need not be viable, since it need not satisfy all constraints in $\mathcal{N}$.

**A $(\tau, \chi)$ Decision.** Let $T$ be some RPS for $\mathcal{N}$ such that $U^x(T)$ is non-empty (i.e., at least one *executable* time-point is unexecuted in $T$). If $\tau > \mu(T)$ and $\chi$ is a *non-empty* subset of $U^x(T)$, then $(\tau, \chi)$ is an allowable RTED for $T$.

**The Outcome of a $(\tau, \chi)$ Decision.** Let $\omega \in \Omega(T)$ be a situation respected by $T$. Then $\mathcal{O}(T, \omega, (\tau, \chi))$ denotes the *outcome* of the decision, $(\tau, \chi)$, for $T$ in the situation $\omega$. The outcome depends on the relationship between the numbers $tnc(T, \omega)$ and $\tau$. For simplicity, let $\tau^c = tnc(T, \omega)$, and let $\chi^a = \chi^a(T, \omega)$. (If $U^a(T) = \emptyset$, let $\tau^c = \infty$.) If $\tau^c < \tau$, the outcome involves the execution of only the (contingent) time-points in $\chi^a$; if $\tau < \tau^c$, the outcome involves the execution of only the (executable) time-points in $\chi$; if $\tau^c = \tau$, the outcome involves the execution of the time-points in *both* $\chi^a$ and $\chi$. In particular, $\mathcal{O}(T, \omega, (\tau, \chi))$ is defined by:

$$\begin{cases} T \cup \{(C, \tau^c) : C \in \chi^a\}, & \text{if } \tau^c < \tau \\ T \cup \{(x, \tau) : x \in \chi\}, & \text{if } \tau < \tau^c \\ T \cup \{(C, \tau^c) : C \in \chi^a\} \cup \{(x, \tau) : x \in \chi\}, & \text{if } \tau^c = \tau \end{cases}$$

Note that the MaxEx time of the outcome is $\min\{\tau^c, \tau\}$. In addition, $T \subset \mathcal{O}(T, \omega, (\tau, \chi))$.

**Lemma 5.** If $T$ is a partial schedule that respects the situation $\omega$, and $\delta$ is an RTED that is allowed for $T$, then the outcome $\mathcal{O} = \mathcal{O}(T, \omega, \delta)$ also respects $\omega$.

**Proof 5.** Let $A_iC_i$ be some contingent link.
**Case 1:** Neither $A_i$ nor $C_i$ appear in $T$. Thus, $C_i \notin U^a(T)$ and $C_i$ is unexecuted in $\mathcal{O}$. If $A_i$ is still unexecuted in $\mathcal{O}$, then condition (1) of the definition of respect is satisfied; otherwise, condition (2) is satisfied, since $\omega_i > 0$.
**Case 2:** $A_i$ appears in $T$, but not $C_i$ (i.e., $C_i \in U^a(T)$). Suppose $C_i \in \chi^a(T, \omega)$. If $\delta$ is the `WAIT` decision or $\delta = (\tau, \chi^c)$ and $tnc(T, \omega) \leq \tau$, then $C_i$ is executed in $\mathcal{O}$ and condition (3) in the definition of respect is satisfied, since $T_{A_i} + \omega_i = tnc(T, \omega)$. Otherwise, $C_i$ is not executed in $\mathcal{O}$ and condition (2) is satisfied, since $\tau < tnc(T, \omega)$. On the other hand, if $C_i \notin \chi^a(T, \omega)$, then $C_i$ will not execute until after $tnc(T, \omega)$. Thus, $C_i$ is not executed in $\mathcal{O}$ and condition (2) is satisfied.
**Case 3:** $A_iC_i$ is finished and respected by $T$. Thus, $A_iC_i$ is finished and respected by the outcome, $\mathcal{O} \supset T$. ■

**RTED-based Strategy (RTEDS).** An *RTED-based strategy* for an STNU, $\mathcal{N}$, is a mapping, $R$, from respectful partial schedules to real-time execution decisions. Thus, if $T$ is a partial schedule that respects at least one situation, then $R(T)$ is a real-time execution decision for $T$.

**Lemma 6.** Let $R$ be an RTEDS, and $\omega$ some situation. Then $R$ and $\omega$ determine a *unique* sequence of schedules$^*$,

$$T^0 = \{(Z, 0)\} \subset T^1 \subset T^2 \subset \ldots \subset T^\alpha$$

where for each $i < \alpha$, $T^i$ is partial and respects $\omega$, $T^{i+1} = \mathcal{O}(T^i, \omega, R(T^i))$, and $\mu(T^i) < \mu(T^{i+1})$; and where $T^\alpha$ is a complete schedule that respects $\omega$.

**Proof 6.** Given some $T^i$ and $\omega$, the outcome $T^{i+1} = \mathcal{O}(T^i, \omega, R(T^i))$ deterministically augments $T^i$ to include the execution of at least one more time-point. Thus, the sequence is unique and terminates in a complete schedule, $T^\alpha$. Furthermore, $\mu(T^i) < \mu(T^{i+1})$. Finally, since $T^0$ respects *all* situations, including $\omega$, Lemma 5 inductively ensures that each $T^i$, including $T^\alpha$, respects $\omega$.

**Lemma 7.** Any DES$^*$ $S$ and situation $\omega$ together determine a *unique* sequence of schedules$^*$,

$$\sigma_0 = \{(Z, 0)\} \subset \sigma_1 \subset \sigma_2 \subset \ldots \subset \sigma_\beta = S(\omega)$$

where $\mu(\sigma_0) < \mu(\sigma_1) < \ldots < \mu(\sigma_\beta)$, and for each $i < \beta$, the time-points that appear in $\sigma_{i+1} - \sigma_i$ are all executed at the time $\mu(\sigma_{i+1})$. This sequence of schedules$^*$ is henceforth called the *signature sequence of schedules*$^*$ for $S(\omega)$. If $S(\omega)$ respects $\omega$, then each $\sigma_i$ respects $\omega$.

**Proof 7.** The schedule $S(\omega)$ involves $|N|$ execution events, some of which may occur simultaneously. Let $0 = \tau_0 < \ldots < \tau_\beta$ be the distinct times of those events, where $\beta \leq |N|$. For each $i$, let $\sigma_{i+1} = \sigma_i \cup \{(t, [S(\omega)]_t) : [S(\omega)]_t = \tau_{i+1}\}$. Then $\mu(\sigma_i) = \tau_i$ for each $i \leq \beta$. If $S(\omega)$ respects $\omega$, then it satisfies condition (3) of *respect* for each contingent link, implying that each $\sigma_i \subseteq S(\omega)$ respects $\omega$. ■

**Reachable schedules.** Let $S$ be a respectful DES$^*$ and $\omega$ a situation. Let $\sigma_0 \subset \ldots \subset \sigma_\beta$ be the signature sequence of schedules for $S(\omega)$. Each $\sigma_i$ is called *reachable* by $S(\omega)$. $\mathcal{R}(S(\omega)) = \{\sigma_i : i < \beta\}$ denotes the set of *reachable partial schedules* for $S(\omega)$. $\mathcal{R}(S) = \bigcup_{\omega \in \Omega} \mathcal{R}(S, \omega)$ denotes the set of *reachable partial schedules* for the *strategy* $S$. By Lemma 7, each $\sigma \in \mathcal{R}(S)$ respects at least one situation.

**Lemma 8.** Let $S$ be a respectful DES$^*$. If $T$ is reachable by $S(\omega')$, and $T$ respects $\omega''$, then $T$ is reachable by $S(\omega'')$.

**Proof 8.** Let $\{\sigma'_i\}$ and $\{\sigma''_i\}$ be the signature sequences of schedules$^*$ for $S(\omega')$ and $S(\omega'')$, respectively. Let $j$ be the smallest index for which $\sigma'_j \neq \sigma''_j$.

Since $T$ is reachable by $S(\omega')$, $T = \sigma'_i$ for some $i$.

**Case 1:** $T = \sigma'_i$ for some $i < j$. But then $T = \sigma''_i$ and, hence, is reachable by $S(\omega'')$.

**Case 2:** $T = \sigma'_i$ for some $i \geq j$. Thus, $\sigma'_j \subset T$. Since $T$ respects $\omega''$, so does $\sigma'_j$. And $\sigma''_j$ respects $\omega''$ by Lemma 7, since $S$ is respectful. Let $\hat{\mu} = \min\{\mu(\sigma'_j), \mu(\sigma''_j)\}$. By construction, $\hat{\mu}$ is the first time at which $\sigma'_j$ and $\sigma''_j$ differ. Thus, $[S(\omega')]^{<\hat{\mu}} = [\sigma'_j]^{<\hat{\mu}} = [\sigma''_j]^{<\hat{\mu}} = [S(\omega'')]^{<\hat{\mu}}$. Since $S$ is a DES$^*$, this implies that $S(\omega')$ and $S(\omega'')$ agree about which executable time-points execute at $\hat{\mu}$. But since both $\sigma'_j$ and $\sigma''_j$ respect $\omega''$, they cannot disagree about the execution times for any contingent time-points at $\hat{\mu}$. Thus, $\sigma'_j = \sigma''_j$, which contradicts the choice of $j$. ∎

**Corollary 8.1.** Let $T \in \mathcal{R}(S)$ be a reachable partial schedule for a respectful DES$^*$ $S$. For any $\omega', \omega'' \in \Omega(T)$, $T$ is reachable for $S(\omega')$ and $S(\omega'')$. Thus, the signature sequences for $S(\omega')$ and $S(\omega'')$ are the same up to $T$, and hence for each $t \in Dom(T)$, $[S(\omega')]_t = T_t = [S(\omega'')]_t$.

**The sets $\Omega^c(T)$ and $\Omega^x(T)$.** For each $T \in \mathcal{R}(S)$, let $\Omega^c(T)$ denote the set of situations, $\omega \in \Omega(T)$, such that the next execution event in $S(\omega)$ after $\mu(T)$ involves *only* contingent time-points; and let $\Omega^x(T)$ denote the set of situations, $\omega \in \Omega(T)$, such that the next execution event in $S(\omega)$ after $\mu(T)$ involves *at least one* executable time-point. Note that $\Omega(T) = \Omega^c(T) \cup \Omega^x(T)$.

**Lemma 9.** Suppose $T \in \mathcal{R}(S)$ for a respectful DES$^*$ $S$. If $\omega', \omega'' \in \Omega^x(T)$, then the time of the next execution event after $\mu(T)$ is the same for $S(\omega')$ and $S(\omega'')$. Call that time $\tau^x$. Also, the sets of executable time-points that execute at $\tau^x$ in $S(\omega')$ and $S(\omega'')$ are the same. Finally, if $\omega^c \in \Omega^c(T)$, then the time, $\tau^c$, of the next *contingent* execution after $\mu(T)$ according to $S(\omega^c)$ is *less than* $\tau^x$.

**Proof 9.** Let $\tau'$ and $\tau''$ be the times of the next execution events after $\mu(T)$ in the schedules $S(\omega')$ and $S(\omega'')$, respectively. Suppose $\tau' \leq \tau''$. By Corollary 8.1, $S(\omega')$ and $S(\omega'')$ are the same up to the time $\mu(T)$. Thus, by construction, $[S(\omega')]^{<\tau'} = [S(\omega'')]^{<\tau'}$. Thus, by Lemma 3, $S(\omega')$ and $S(\omega'')$ must agree on all executable time-points that execute at $\tau'$, which implies that $\tau' = \tau''$. Let $\tau^x = \tau' = \tau''$.

Suppose $\tau^c \geq \tau^x$. Then $[S(\omega')]^{<\tau^x} = [S(\omega^c)]^{<\tau^x}$ which, by Lemma 3, implies that $S(\omega')$ and $S(\omega^c)$ must agree on all executable time-points that execute at or before time $\tau^x$, contradicting the choice of $\omega^c \in \Omega^c(T)$. ∎

**Theorem A.** Let $S$ be a respectful DES$^*$ for $\mathcal{N}$. Then there exists an RTED-based strategy, $R$, that is equivalent to $S$ in the sense that for each situation $\omega \in \Omega_{\mathcal{N}}$, the signature sequence of schedules$^*$ for $S(\omega)$ is identical to the unique sequence of outcomes determined by $R$ and $\omega$.

**Proof A.** Let $S$ be a respectful DES$^*$. Define a mapping $R$ from RPSs to RTEDs, as follows. First, if $T$ is *reachable* for $S$ (i.e., $T \in \mathcal{R}(S)$):

- If $\Omega^x(T) = \emptyset$, then let $R(T) = \texttt{WAIT}$.
- If $\Omega^x(T) \neq \emptyset$, then let $R(T) = (\tau^x, \chi)$, where $\tau^x$ is the time of the next execution event for any $\omega \in \Omega^x(T)$, and $\chi$ is the set of executable time-points that execute at the time $\tau^x$ in any such $\omega$.

The uniqueness of $\tau^x$ and $\chi$ is guaranteed by Lemma 9; thus, $R(T)$ is well-defined for $T \in \mathcal{R}(S)$.

For any other respectful partial schedule, $T$, if $U^a(T)$ is non-empty, let $R(T) = \texttt{WAIT}$; otherwise, let $R(T) = (\mu(T) + 1, \{x\})$, where $x$ is some time-point in $U^x(T)$.

Finally, for any situation $\omega$, let $\sigma_0 \subset \ldots \subset \sigma_\beta = S(\omega)$ be the signature sequence of schedules$^*$ for $S(\omega)$; and let $T^0 \subset \ldots \subset T^\alpha$ be the unique sequence of outcomes determined by $R$ and $\omega$. It suffices to show that $\alpha = \beta$ and $T^i = \sigma_i$ for each $i$—and hence that $T^\alpha = \sigma_\beta = S(\omega)$.

**Base Case:** $\sigma_0 = \{(Z, 0)\} = T^0$.

**Recursive Case:** $\sigma_i = T^i$ for some $i < \alpha$. Then $T^i$ is reachable for $S$, and $[T^i]_t = [S(\omega)]_t$ for all $t \in Dom(T^i)$. Thus, for any $C_j \in U^a(T^i)$, $T^i_{A_j} + \omega_j = [S(\omega)]_{A_j} + \omega_j = [S(\omega)]_{C_j}$. If $\Omega^x(T^i) = \emptyset$, then $R(T^i) = \texttt{WAIT}$ and $\omega \in \Omega^c(T^i)$, which implies that $U^a(T^i) \neq \emptyset$. Thus, $tnc(T^i, \omega) = \min\{[S(\omega)]_{C_j} : C_j \in U^a(T^i)\}$ equals $\tau^c$, the time of the next contingent execution event in $S(\omega)$, and $\chi^a(T^i, \omega)$ is the set of contingent time-points executing at that time in $S(\omega)$. Thus, the outcome, $T^{i+1}$, equals $\sigma_{i+1}$.

However, if $\Omega^x(T^i) \neq \emptyset$, then $R(T^i) = (\tau^x, \chi)$. If $\omega \in \Omega^x(T^i)$, then the outcome includes the execution of the time-points in $\chi$. Otherwise, $\omega \in \Omega^c(T^i)$, and $\tau^c < \tau^x$ implies that the outcome involves the execution of contingent time-points as in the `WAIT` case. In either case, the outcome agrees with $S(\omega)$, implying that $T^{i+1} = \sigma_{i+1}$. ∎

**Theorem B.** Each RTEDS is a respectful DES$^*$.

**Proof B.** Let $R$ be any RTED-based strategy for $\mathcal{N}$. Given any $\omega$, define $S_R(\omega)$ to be the terminal schedule in the unique sequence of outcomes determined by $R$ and $\omega$ which, by Lemma 6, is guaranteed to respect $\omega$.

Let $\omega', \omega''$ be a pair of situations for which the DES$^*$ property fails. Let $k$ be the earliest time of such a failure. Thus, for some executable $x$, $k = [S_R(\omega')]_x$ and $[S_R(\omega')]^{<k} = [S_R(\omega'')]^{<k}$, but $[S_R(\omega'')]_x \neq k$.

Let $\{T_1^j\}$ and $\{T_2^j\}$ be the sequences of outcomes determined by following $R$ in the situations $\omega'$ and $\omega''$, respectively. Let $T_1^{i+1}$ and $T_2^{i+1}$ be the first pair of outcomes in these sequences that differ. Thus, $T_1^i = T_2^i$. Hence, the governing decision at step $i$ was the same: $R(T_1^i) = R(T_2^i)$.

0. Let $T = \{(Z, 0)\}$ be the initial partial schedule.
1. If $U(T) = \emptyset$, then DONE!
2. If $U^x(T) = \emptyset$, let $\delta(T) = \texttt{WAIT}$. Go to Step 4.
3. For each $x \in U^x(T)$, let $[m(x), M(x)]$ be the current time-window for $x$ in $\mathcal{N}$. If every contingent link, $A_iC_i$, for which $x$ has a *wait*, $w(A_i, C_i, x)$, is activated in $T$, let $W(x) = \max\{T_{A_i} + w(A_i, C_i, x) : C_i \in U^a(T)\}$; otherwise, let $W(x) = \infty$. Let $\mathit{floor}(x) = \max\{m(x), W(x)\}$ and $go(x) = \min\{\mathit{floor}(x), M(x)\}$. Let $\delta(T) = (\tau^x, \chi)$, where $\tau^x = \min\{go(x) : x \in U^x(T)\}$ and $\chi = \{x \in U^x(T) : \tau^x = go(x)\}$.
4. If $\delta(T) = \texttt{WAIT}$, then wait until some contingent time-point executes. Otherwise, $\delta(T) = (\tau^x, \chi)$. If nothing happens before time $\tau^x$, then execute the time-points in $\chi$; otherwise, observe the contingent time-points executed at some $\tau^c < \tau^x$.
5. Update $T$ to include the execution events from Step 4. Update $\mathcal{N}$ to include the corresponding constraints. Go to Step 1.

**Figure 3. MMV's strategy as an RTEDS**

Let $\hat{\mu} = \min\{\mu(T_1^i), \mu(T_2^i)\}$. Note that $\hat{\mu}$ is the earliest time at which $S_R(\omega')$ and $S_R(\omega'')$ differ.

**Case 1:** $\hat{\mu} < k$. Then the DES$^*$ property holds at $\hat{\mu}$ and $[S_R(\omega')]^{<\hat{\mu}} = [S_R(\omega'')]^{<\hat{\mu}}$ implies, by Lemma 3, that $S_R(\omega')$ and $S_R(\omega'')$ agree about all *executable* time-points at $\hat{\mu}$, and *all* time-points before $\hat{\mu}$. But then $[S_R(\omega')]^{<k} = [S_R(\omega'')]^{<k}$ implies that $S_R(\omega')$ and $S_R(\omega'')$ agree on all contingent time-points at $\hat{\mu}$, too, contradicting choice of $\hat{\mu}$.

**Case 2:** $k \leq \hat{\mu}$. Then $S_R(\omega')$ and $S_R(\omega'')$ agree about all time-points before $k$. Since $x$ is executed at $k$ in $S_R(\omega')$, the decision $R(T_1^i) = R(T_2^i)$ must have been $(k, \chi)$ for some $\chi$ containing $x$. But then $T_2^{i+1}$ must be an outcome at time $k$, which implies that $x \in \chi$ is also executed in $T_2^{i+1}$ at time $k$, contradicting that $[S_R(\omega'')]_x \neq k$. ■

**RTED-based Version of MMV's Verification Strategy.** To prove that any STNU accepted by their DC-checking algorithm is dynamically controllable, MMV presented not a mapping from situations to schedules, but a *procedure* for incrementally generating a single schedule in response to the unfolding situation. Space limitations preclude duplicating that procedure here. Instead, Fig. 3 presents an equivalent procedure for generating real-time execution decisions. Although the computations are equivalent, they are structured around partial schedules and RTEDs.

The procedure in Fig. 3 starts with an initial partial schedule, $T = \{(Z, 0)\}$. It then computes an RTED, $\delta(T)$, in Step 2 or Step 3, as follows. If all executable time-points have already been executed, then $\delta(T) = \texttt{WAIT}$ (Step 2). Otherwise, $\delta(T) = (\tau^x, \chi)$ based on the values for $\tau^x$ and $\chi$ computed in Step 3. These values are based on the current time-windows for the as-yet-unexecuted executable time-points, which can be computed using an all-pairs, shortest-path algorithm. (No generation or propagation of waits is required.) $\mathit{floor}(x)$ is the earliest time $x$ can be executed without violating its lower bound, $m(x)$, or any of its relevant waits. $go(x)$ is the same except that it enforces the constraint that $x$ not violate its upper bound, $M(x)$. (MMV, in effect, prove that a conflict between $\mathit{floor}(x)$ and $M(x)$ is not possible for an STNU accepted by their algorithm.)

In Step 4, the agent waits to see what the outcome of the decision $\delta(T)$ will be. In Step 5, the agent updates $T$ and $\mathcal{N}$ to reflect that outcome, before returning to Step 1. Eventually, the procedure terminates when all of the time-points have been executed.

For any situation, $\omega$, the procedure in Fig. 3 leads the agent through the characteristic sequence of outcomes described in Lemma 6. Thus, it determines an RTED-based strategy which, by Theorem A, is necessarily a DES$^*$. Therefore, the strategy generated by MMV's procedure is necessarily a DES$^*$. Thus, their proof that that strategy is viable ensures that their DC-checking algorithm is complete with respect to the definitions of DES$^*$ and DC$^*$.

## 5. Conclusions

This paper fixed a technical flaw in MMV's semantics for dynamic controllability. It presented an alternative characterization of strategies based on real-time decisions and showed that MMV's existing DC-checking algorithm is sound and complete with respect to the revised semantics.

## References

[1] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[2] P. Morris. A structural characterization of temporal dynamic controllability. In *Principles and Practice of Constraint Programming (CP 2006)*, volume 4204 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 2006.

[3] P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In *17th Int'l. Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 494–499, 2001.

[4] P. H. Morris and N. Muscettola. Temporal dynamic controllability revisited. In *20th National Conference on Artificial Intelligence (AAAI-2005)*, pages 1193–1198, 2005.

[5] T. Vidal. A unified dynamic approach for dealing with temporal uncertainty and conditional planning. In *Fifth International Conference on Artificial Intelligence Planning Systems (AIPS-2000)*, pages 395–402, 2000.

[6] T. Vidal and H. Fargier. Contingent durations in temporal csps: from consistency to controllabilities. In *Proceedings of the TIME-97 Workshop*, 1997.

[7] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(1):23–45, 1999.

[8] T. Vidal and M. Ghallab. Temporal constraints in planning: Free or not free? In *CONSTRAINT '95 Workshop*, 1995.