Agile Controllability of Simple Temporal Networks with Uncertainty and Oracles

Johann Eder[1] Roberto Posenato[2] Carlo Combi[2] Marco Franceschetti[3] Franziska S. Hollauf[1]

1. University of Klagenfurt, Austria
2. University of Verona, Italy
3. University of St. Gallen, Switzerland

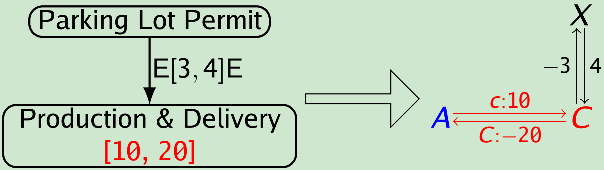
TIME 2024

Introduction

- Simple temporal networks with uncertainty (STNUs) [Morris and Muscettola, 2005, Cairo and Rizzi, 2019] have gained wide recognition for modeling temporal constraints. They extend Simple Temporal Networks (STNs) [Dechter et al., 1991].
- Dynamic controllability [Morris and Muscettola, 2005] is currently the most studied notion for the temporal correctness of STNUs.
- For checking dynamic controllability, effective and efficient methods with polynomial complexity have been proposed [Morris, 2014, Cairo and Rizzi, 2019, Hunsberger and Posenato, 2022].

A motivating example

An STNU not dynamically controllable!



A, C represents the starting/ending event of “P. & D.” task.
X represents the ending event of “Parking Lot Permit” task.
Contingent link: $C - A \in [10, 20]$

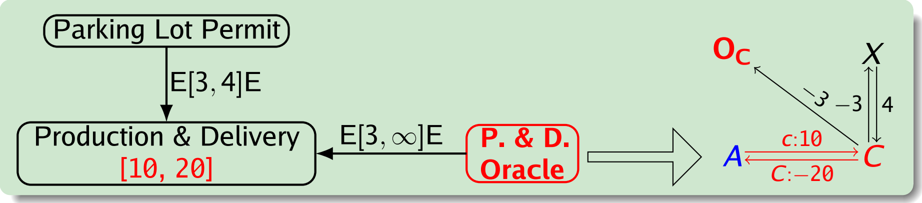
- A scheduler *executes* the network assigning values to timepoints.
- A scheduler only decides execution of A and X.
- A scheduler only assigns a value to C when it occurs and, thus, the duration $d = C - A$ is revealed.

Focus of the Talk

- Dynamic controllability assumes that the duration of contingent activities, hence the values of contingent timepoints, are only known when they happen.
- How the notion of dynamic controllability can be generalized such that a timepoint may depend not only on timepoints which are earlier, but also which are known earlier?

The Motivating Example revisited

Extending STNUs with oracles



- We propose to add the **oracle O_C** as a new special timepoint.
- If a contingent timepoint is associated with an oracle, then **the oracle** must be executed before the contingent timepoint, and at its execution, it reveals the duration of the contingent link in the current execution.

Focus of the Talk

- The main novelties:
  1. The formal definition of STNUO (Simple Temporal Network with Uncertainty and Oracles).
  2. A formal definition of Agile Controllability.
  3. ORUL, a set of rules for propagating constraints in STNUOs.
  4. An algorithm for checking Agile Controllability of STNUOs.
  5. A proof–of–concept implementation of the checking algorithm.

STNU Definition

Definition (STNU)

An STNU is a triple $(\mathcal{T}, \mathcal{C}, \mathcal{L})$, where:

- $\mathcal{T}$ is a finite, non–empty set of real–valued variables called **timepoints**. $\mathcal{T}$ is partitioned into $\mathcal{T}_X$, the set of executable timepoints, and $\mathcal{T}_C$, the set of contingent timepoints.
- $\mathcal{C}$ is a set of **binary (ordinary) constraints**, each of the form $Y - X \leq \delta$ for some $X, Y \in \mathcal{T}$ and $\delta \in \mathbf{R}$.
- $\mathcal{L}$ is a set of contingent links, each of the form (A, x, y, C), where $A \in \mathcal{T}_X, C \in \mathcal{T}_C$ and $0 < x < y < \infty$. A is called the activation timepoint; C contingent timepoint.

Dynamic Controllability: preliminary definitions

Definition (Situation)

Each situation $\omega = (\omega_1, \ldots, \omega_K) \in \Omega$ represents one possible complete set of values for the duration of the contingent links of $\mathcal{N}$ (chosen by nature).

Definition (Schedule)

A **schedule** for an STNU $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is a mapping $\xi \colon \mathcal{T} \cup \{\perp\} \to \mathbf{R}$, where we assume that $\xi(\perp) = +\infty$. Ξ denotes the set of all schedules for an STNU. For historical reasons, we represent $\xi(X)$ as $[X]_\xi$.

Definition (Execution Strategy)

An **execution strategy** S for an STNU $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is a mapping $S : \Omega \to \Xi$.

Definition (Viable Execution Strategy)

An **execution strategy** S for an STNU $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is **viable** if for each situation $\omega \in \Omega$ the schedule $S(\omega)$ is a solution for $\mathcal{N}$, i.e., an assignment that **satisfies all the constraints** in the network.

Dynamic Controllability

Definition (Dynamic Execution Strategy)

An **execution strategy for an STNU** $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is **dynamic** if, for any two situations ω', ω'' and any executable timepoint $X \in \mathcal{T}_X$, it holds that:

$$\textit{if } [X]_{S(\omega')} = k \textit{ and } S(\omega')^{\leq k} = S(\omega'')^{\leq k}, \textit{ then } [X]_{S(\omega'')} = k,$$

where $S(\omega')^{\leq k}$ is the set of contingent link durations observed up to and including time k, called *history until k*. Since history also considers contingent durations observed at instant k, we say that the dynamic execution strategy implements the *instantaneous reaction* semantics.

Definition (Dynamic Controllability)

An STNU is **dynamically controllable** if there exists a viable dynamic execution strategy for it, that is, an execution strategy that assigns the executable timepoints with the guarantee that all constraints will be satisfied, irrespectively of the duration values (within the specified bounds) of contingent links [Hunsberger, 2016].

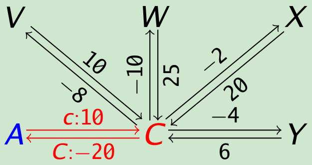
STNU graphical representation

Each STNU $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ has a corresponding graph $\mathcal{G} = (\mathcal{T}, \mathcal{E}_{\mathrm{o}} \cup \mathcal{E}_{\mathrm{lc}} \cup \mathcal{E}_{\mathrm{uc}})$, also called ***distance graph***, where the timepoints in $\mathcal{T}$ serve as the graph's nodes and the constraints in $\mathcal{C}$ and $\mathcal{L}$ correspond to labeled, directed edges. In particular:

- $\mathcal{E}_{\mathrm{o}} = \{X \xrightarrow{\delta} Y \mid (Y - X \leq \delta) \in \mathcal{C}\}$
- $\mathcal{E}_{\mathrm{lc}} = \{A \xrightarrow{c:x} C \mid (A, x, y, C) \in \mathcal{L}\}$, and
- $\mathcal{E}_{\mathrm{uc}} = \{C \xrightarrow{C:-y} A \mid (A, x, y, C) \in \mathcal{L}\}$.

STNU graphical representation

Example



Checking Dynamic Controllability

Rule	Pre-existing and generated edges	Applicability Conditions
(R)	$W \xrightarrow{v} Y \xrightarrow{u} X$, $W \xrightarrow{u+v} X$	none
(U)	$X \xrightarrow{v} C \xrightarrow{C:-y} A$, $A \xrightarrow{c:x} C$, $X \xrightarrow{max\{v-y,-x\}} A$	$(A, x, y, C) \in \mathcal{L}$
(L)	$X \xleftarrow{v} C \xleftarrow{c:x} A$, $A \xrightarrow{x+v} X$	$v \leq 0$ or $X \in \mathcal{T}_C$, $X \not\equiv C$, $\exists(B, s, t, X) \in \mathcal{L}$, $v \leq t$

Table: Edge-generation rules used by RUL [Cairo and Rizzi, 2019, Fig. 2]

- Constraint propagation algorithms have been proposed to check whether an STNU is dynamically controllable [Morris, 2014, Cairo et al., 2018, Hunsberger and Posenato, 2022, Hunsberger and Posenato, 2023].
- The algorithm terminates when either reaching network quiescence (the network is dynamically controllable), or a *negative cycle* is found (the network is not dynamically controllable).

Extending STNUs with Oracles

Definition (STNU with **Oracles** (STNUO))

An STNUO is a tuple $(\mathcal{T}, \mathcal{C}, \mathcal{L}, \mathcal{O})$, where:

- $\mathcal{T}$ is a finite, non-empty set of real-valued variables called **timepoints**. $\mathcal{T}$ is partitioned into $\mathcal{T}_X$, the set of executable timepoints and $\mathcal{T}_C$, the set of contingent timepoints. $\mathcal{T}_O \subseteq \mathcal{T}_X$, is the set of **oracle** timepoints.
- $\mathcal{C}$ is a set of binary (ordinary) constraints, each of the form $Y - X \leq \delta$ for some $X, Y \in \mathcal{T}$ and $\delta \in \mathbf{R}$.
- $\mathcal{L}$ is a set of contingent links, each of the form (A, x, y, C), where $A \in \mathcal{T}_X$, $C \in \mathcal{T}_C$ and $0 < x < y < \infty$. A is called the *activation* timepoint; C *contingent* timepoint. If (A_1, x_1, y_1, C_1) and (A_2, x_2, y_2, C_2) are distinct contingent links, then $C_1 \neq C_2$.
- $\mathcal{O}\colon \mathcal{T}_C \to \mathcal{T}_O \cup \{\perp\}$ is a function that associates a contingent timepoint with its corresponding **oracle**, if any. For the sake of simplicity and without loss of generality, we assume that each **oracle** is associated with a single contingent timepoint.

Extending Execution Strategies

Definition (Agile Execution Strategy with Oracles)

Let $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L}, \mathcal{O})$ be an STNUO. An *execution strategy with oracles* S_O for $\mathcal{N}$ is *agile* if, for any two situations ω', ω'' and any executable timepoint $X \in \mathcal{T}$, it holds that:

$$\text{if } [X]_{S_O(\omega')} = k \text{ and } S_O(\omega')^{\leq k} = S_O(\omega'')^{\leq k}, \text{ then } [X]_{S_O(\omega'')} = k$$

where $S_O(\omega)$ is a schedule determined by the execution strategy with oracles S_O given the situation ω, and $S_O(\omega)^{\leq k}$ is the **oracle history** until k.

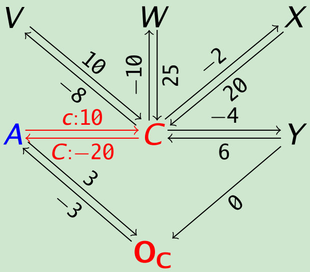
Definition (Agile Controllability (AC))

An STNUO $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L}, \mathcal{O})$ is *agilely controllable* if it admits a viable agile execution strategy with oracles.

Controllability of STNUs with Oracles

Example

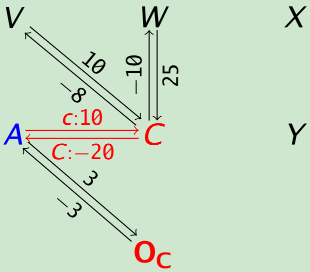
Let us consider an STNUO $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L}, \mathcal{O})$ where $\mathbf{O_C}$ is the oracle for C that must be executed 3 time units after the activation timepoint A. Let $d \in [10, 20]$ be the duration revealed by the oracle $\mathbf{O_C}$.



Scheduling STNUs with Oracles – I

Example

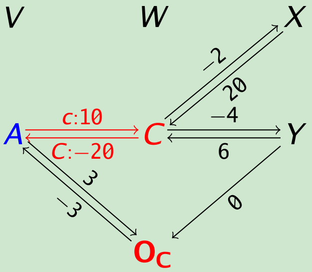
1. V can neither be scheduled with nor without an oracle because if the contingent link lasts 10, the oracle is executed too late to allow V to be executed, satisfying the constraint with C.
2. W must be scheduled before the oracle to satisfy the constraint with C. Therefore, the oracle is not relevant for scheduling W.



Scheduling STNUs with Oracles – II

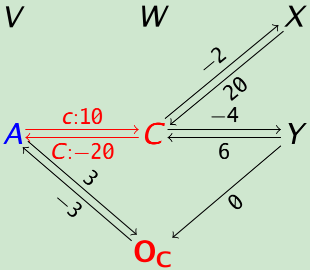
Example ($\mathbf{O_C}$ reveals duration d)

3. X can be scheduled without oracle (for example, $X = A + 2$) or with oracle (for example, $X = \mathbf{O_C} - 3 + d - 4 = A + d - 4$, where 4 is one of the possible values to choose.)
4. Y can be scheduled only with oracle: $Y = \mathbf{O_C} - 3 + d - 5 = A + d - 5$. Thus, Y can be executed only after $\mathbf{O_C}$.



Scheduling STNUs with Oracles – III

Example



- Node X can be scheduled since the contingency of C is 10 smaller than 18, the range of possible distances between C and X.
- For node Y, these values are 10 and 2. Applying the rules U and L to Y, C, and A leads to a negative cycle.

Rule propagation and oracles

For the propagation of constraints, this has the following consequences:

1. If the oracle is not used, then the L and U rules must be applied.
2. If the oracle is used, then the L and U rules must not be applied on X and C, but the additional constraint $O_c \leq X$ must be added to the STNUO.

Example

- Using the oracle avoids the negative cycle resulting from propagation in case (1), such as Y.
- Using the oracle as in case (2) might lead to a conflict that was not there, such as W.
- For timepoint X, there is the choice to either apply the rules L and U or consider the oracle O_C and introduce constraint $O_c \leq X$.

Using Oracles or not?

Definition (Oracle Candidate)

We call $\mathcal{U} = \{(X, C) \mid X \in \mathcal{T}_X, C \in \mathcal{T}_C, \mathcal{O}(C) \neq \perp\}$ the set of all **potential oracle candidates**. If there is constraint $X \leq C + \delta$, or a constraint $C \leq X + \delta'$ with $C \in \mathcal{T}_C$, $\mathcal{O}(C) \neq \perp$, $X \in \mathcal{T}_X$, $\delta < 0$, and $0 \leq \delta'$, then we call (X, C) an **oracle candidate** since a viable execution strategy *could* require the usage of the oracle.

Example

All pairs $(V, C), (W, C), (X, C), (Y, C)$ are oracle candidates.

Example

(Y, C) is named oracle dependent as there is no viable execution strategy without using the oracle O_C.

ORUL: Propagation rules for STNUOs

Rule	Pre-existing and generated edges	Conditions
Relax (REL)	$\begin{tikzcd} W \arrow[r, "v"] \arrow[rr, bend right, "u+v"'] & Y \arrow[r, "u"] & X \end{tikzcd}$	none
Upper (UPP)	$\begin{tikzcd} X \arrow[r, "v"] \arrow[rr, bend right, "max\{v-y,-x\}"'] & C \arrow[r, shift left, "C:-y"] & A \arrow[l, shift left, "c{:}x"] \end{tikzcd}$	$(X,C) \in \mathcal{U}^-$ or $\mathcal{O}(C) = \perp$ or $X \in \mathcal{T}_C$
Lower (LOW)	$\begin{tikzcd} X & C \arrow[l, "v"'] & A \arrow[l, "c{:}x"'] \arrow[ll, bend left, "x+v"] \end{tikzcd}$	$X \in \mathcal{T}_{\mathcal{X}}$, $((X,C) \in \mathcal{U}^-$ or $\mathcal{O}(C) = \perp)$, $v \leq 0$, or $X \in \mathcal{T}_C$, $X \not\equiv C$, $\exists(B,w,y,X) \in \mathcal{L}$, $v \leq y$
Oracle (ORC)	$\begin{tikzcd} X \arrow[d, bend right, "0"'] & C \arrow[l, "v"'] \arrow[r, shift left, "C:-y"] & A \arrow[l, shift left, "c{:}x"] \\ O_C & & \end{tikzcd}$	$X \in \mathcal{T}_X$, $(X,C) \in \mathcal{U}^+$

- $\mathcal{U}^+$ is the set of all oracle candidates for which the oracle has to be used, and
- $\mathcal{U}^-$ is the set of all oracle candidates for which the oracle has not to be used.

Agile Controllability

Theorem

Let

- $\mathcal{N} = (\mathcal{T}, \mathcal{C}, \mathcal{L}, \mathcal{O})$ *be an STNUO;*
- $\mathcal{U} = \{(X, C) \mid X \in \mathcal{T}_X, C \in \mathcal{T}_C, \mathcal{O}(C) \neq \perp\}$ *be the set of all possible pairs (ordinary node, contingent node) where the contingent node has its corresponding oracle.*

$\mathcal{N}$ *is Agilely Controllable if* $\exists\, \mathcal{U}^+, \mathcal{U}^-$ *such that* $\mathcal{U} = \mathcal{U}^+ \cup \mathcal{U}^-$, $\mathcal{U}^+ \cap \mathcal{U}^- = \emptyset$, *and the closure of* $\mathcal{N}$ *considering* $\mathcal{U}^+, \mathcal{U}^-$ *for the propagation rules of ORUL does not include a negative cycle.*

The Checking Algorithm CheckAC

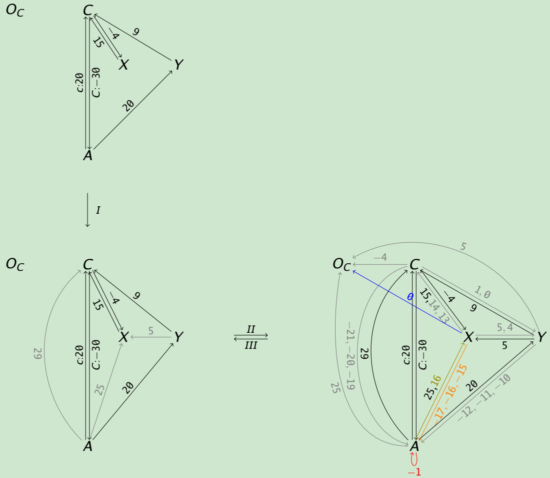
Input: $\mathcal{N}$ an STNUO, $\mathcal{U}^-$, $\mathcal{U}^+$
Output: Agile controllability status

$\texttt{ok} \leftarrow \texttt{applyRules}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;
if *ok* **then**
  $\texttt{save}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;
  $\mathcal{U}^0 \leftarrow \texttt{getOpenOracles}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;
  if $\mathcal{U}^0 \neq \emptyset$ **then**
    $(X, C) \leftarrow \texttt{select}(\mathcal{U}^0)$;
    if $\texttt{interval}(X, C) > \texttt{contingencyInterval}(C)$ **then**
      $\mathcal{U}^- \leftarrow \mathcal{U}^- \cup \{(X, C)\}$;
      $\texttt{ok} \leftarrow \texttt{checkAC}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;
      if *not ok* **then**
        $\texttt{restore}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;
    if $(X, C) \notin \mathcal{U}^-$ **then**
      $\mathcal{U}^+ \leftarrow \mathcal{U}^+ \cup \{(X, C)\}$;
      $\texttt{ok} \leftarrow \texttt{checkAC}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;
      if *not ok* **then**
        $\texttt{restore}(\mathcal{N}, \mathcal{U}^-, \mathcal{U}^+)$;

return *ok*

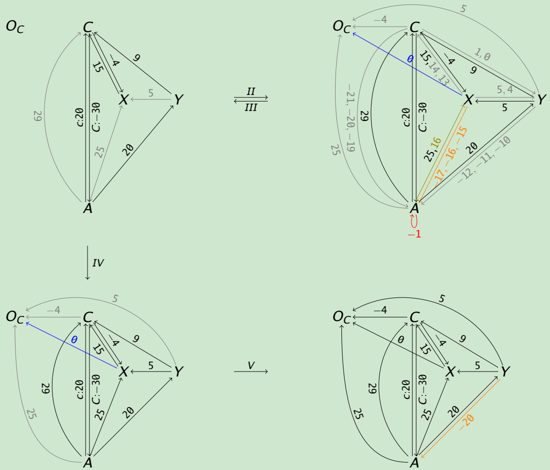
Applying CheckAC on an STNUO I

Example



Applying CheckAC on an STNUO II

Example



Proof–of–concept prototype[1]

- The algorithm has been implemented in Java.
- Experiments on an Ubuntu 22 machine having 16GB of RAM and an AMD EPYC–Rome (8) @ 2.6GHz CPU.
- Input data from the OSTNU benchmark, which is available online at `https://profs.scienze.univr.it/~posenato/software/benchmarks/OSTNUBenchmarks2024.tgz` [Posenato, 2022].
- 30 random STNUO instances with 30 nodes (5 contingent and 2 oracles).
- All execution average times are below 3 s.
- Comparable to the algorithm presented by Posenato *et al.* [Posenato et al., 2024].

[1]The source code of the prototype implementation, the parser of the data sets used for the experiments, and the complete results are publicly available at `https://git-isys.aau.at/ics/Papers/stnuo.git`

Concluding remarks

- We proposed agile controllability (AC) as a proper generalization of the well–established notion of dynamic controllability for STNUs,
- STUNOs, Simple Temporal Networks with Uncertainty and Oracles, can express *when* information about the timepoint of future contingent links is available,
- AC is expected to support a wide range of applications as it provides a less restrictive notion of temporal correctness of plans, processes, requirements, contracts, etc.
- Indeed, the algorithm `CheckAC` and its related proof–of–concept implementation, as it is presented here, is intended to demonstrate the existence of an effective backtracking algorithm to check the agile controllability of an STNUO. Optimizations are part of the of ongoing research.

References I

[Cairo et al., 2018] Cairo, M., Hunsberger, L., and Rizzi, R. (2018).
Faster dynamic controllability checking for simple temporal networks with uncertainty.
In *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume 120 of *LIPIcs*, pages 8:1–8:16.
ISBN: 9783959770897.

[Cairo and Rizzi, 2019] Cairo, M. and Rizzi, R. (2019).
Dynamic controllability of simple temporal networks with uncertainty: Simple rules and fast real–time execution.
Theoretical Computer Science, 797:2–16.

[Dechter et al., 1991] Dechter, R., Meiri, I., and Pearl, J. (1991).
Temporal constraint networks.
Artificial intelligence, 49(1–3):61–95.

[Hunsberger, 2016] Hunsberger, L. (2016).
Efficient execution of dynamically controllable simple temporal networks with uncertainty.
Acta Informatica, 53:89–147.

References II

[Hunsberger and Posenato, 2022] Hunsberger, L. and Posenato, R. (2022).
Speeding up the RUL^{-} Dynamic-Controllability-Checking Algorithm for Simple Temporal Networks with Uncertainty.
In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, volume 36, pages 9776–9785. AAAI Press.

[Hunsberger and Posenato, 2023] Hunsberger, L. and Posenato, R. (2023).
A Faster Algorithm for Converting Simple Temporal Networks with Uncertainty into Dispatchable Form.
Information and Computation, 293.

[Morris, 2014] Morris, P. (2014).
Dynamic controllability and dispatchability relationships.
In *CPAIOR 2014*, volume 8451 of *LNCS*.

[Morris and Muscettola, 2005] Morris, P. H. and Muscettola, N. (2005).
Temporal dynamic controllability revisited.
In *20th National Conf. on Artificial Intelligence (AAAI-2005)*.

[Posenato, 2022] Posenato, R. (2022).
CSTNU Tool: A Java library for checking temporal networks.
SoftwareX, 17:100905.

References III

[Posenato et al., 2024] Posenato, R., Franceschetti, M., Combi, C., and Eder, J. (2024).
Introducing agile controllability in temporal business processes.
In *Enterprise, Business-Process and Information Systems Modeling – 25th International Conference, BPMDS 2024, and 29th International Conference, EMMSAD 2024*, volume 511 of *Lecture Notes in Business Information Processing*, pages 87–99. Springer.