On the Complexity of Temporal Controllabilities for Workflow Schemata

Carlo Combi
carlo.combi@univr.it

Roberto Posenato
roberto.posenato@univr.it

Department of Computer Science
University of Verona
strada le Grazie, 15 - 37134 Verona, Italy

ABSTRACT

Recently, different kinds of *controllability* have been proposed for workflow schemata modeling real world processes made of tasks and coordination activities. Temporal controllability is the capability of executing a workflow for all possible durations of all tasks satisfying all temporal constraints. Three different types of controllability are possible – *strong controllability*, *history-dependent controllability*, and *weak controllability* – and a general exponential-time algorithm to determine the kind of controllability has been proposed. In this paper we analyze the computational complexity of the temporal controllability problem to verify the quality of proposed algorithms. We show that the *weak controllability* problem is coNP-complete, while *strong controllability* problem $\in \Sigma_2^{\mathsf{P}}$ and it is coNP-hard. Regarding the *history-dependent controllability* problem, we are able to show that it is a PSPACE problem but further research is required to determine its hardness characterization.

Keywords

temporal workflow analysis; computational complexity; controllability; temporal conceptual workflow design; temporal constraint networks.

1. INTRODUCTION

Workflow management systems (*WfMSs*) allow organizations to streamline, automate, and manage business processes that depend on information systems and human resources [13]. In general, business processes have different kinds of temporal restrictions such as a limited duration of single tasks or activity deadlines w.r.t. either the beginning of the workflow or a specific time point in the control flow: to this regard, time violations lead to some form of exception handling, thus increasing the complexity of business process management. Thus, a *WfMS* should manage the necessary information about a process, its time restrictions, and its actual time requirements, both at design time and at run time. Focusing on the design step, a workflow schema can be viewed as a graphical specification of admitted coordinate execution of a set of tasks in order to reach a given goal. Nodes represent tasks, generally executed by external agents, and connectors, executed by the workflow engine to coordinate the overall task execution. Edges represent the order of execution of nodes. Each node/edge has a temporal constraint consisting of the range of admitted durations for its execution. Further temporal constraints may be specified for non consecutive nodes. Controllability is the capability of executing a workflow for all possible durations of all tasks and satisfying all temporal constraints. Recently, three different kinds of *controllability* have been proposed for workflow schemata [5]: i) *strong controllability* refers to the fact that it is possible to derive a single duration range for each connector/edge ensuring the controllability of each possible execution of the workflow schema; ii) *history-dependent controllability* refers to the capability of deriving a duration range for each connector/edge ensuring the controllability of each possible execution of the workflow schema but such range could depend on the specific set of tasks preceding the considered connector/edge; iii) *weak controllability* refers to the capability of deriving a duration range for each connector/edge ensuring the controllability of each execution once it is known which execution path has to be followed. Moreover, in [5], a general exponential-time algorithm to determine the controllability of a workflow schema is shown.

Since in [5] the authors do not deal with the computational complexity of the controllability problem, in this paper we consider it in order to verify the quality of proposed algorithms. This study is important, even from an application point of view, because the controllability checking has to be executed both at design time and, more important, at run time repeatedly. Thus, a fine tuning of algorithms can be reached only if we are aware of the problem complexity.

2. RELATED WORK

Regarding the concept of controllability of workflow schemata, two main research directions may be considered: i) the extension of methodologies for business-process modeling and of workflow management systems to consider different kinds of temporalities; ii) in the AI-related area of temporal constraints, the studies about the constraint satisfiability when some temporal constraints are related to time distances between point-based events that are not under control.

In [3], Combi et al. propose a temporal conceptual workflow model that enhances the expressiveness of previous proposals in representing temporal constraints, such as those related to tasks and connectors. One innovative aspect of the proposal is the standardization in expressing temporal constraints among tasks or connectors; temporal constraints are expressed by ranges representing lower and upper bound of the constraints. Furthermore, the authors propose different kinds of temporal constraints consistency and exhibit some algorithms to check the consistency of a workflow schema both at design time and at run time.

In [2], Bettini et al. try to merge the research directions on temporal workflow models and on temporal constraint networks. The authors introduce the concept of *free schedule*: a schedule is free when it is possible to statically fix the start times of all tasks without constraining their durations and satisfying all the given constraints, before the beginning of the execution. A polynomial algorithm ($O(n^4)$ where n is the number of nodes) is then provided to check the existence of a free schedule.

Free schedules resemble the concept of controllability, that has been mainly investigated by Morris, Muscettola, and Vidal [16, 12], in the AI area of temporal constraint networks for planning. Assuming that a temporal planner has to manage the likely uncertainty about the duration of processes, Vidal et al. propose an extension of the Simple Temporal Network, the Simple Temporal Network With Uncertainty (*STNU*), where edges, i.e., constraints, are divided in two classes: *contingent links* and *requirement links*. Contingent links represent processes of uncertain duration, where finish timepoints (i.e., *STNU* nodes) are decided by Nature within the limits imposed by the bounds defined on the contingent links. Requirement links represent all the other processes whose finish timepoints are controlled by the agents that execute processes. Informally, *Controllability* refers to the capability of specifying all the timepoints controlled by agents, satisfying all the requirement and contingent links. In particular, *dynamic controllability* ensures that it is possible to specify at run time the timepoints controlled by agents only by knowing the duration of the already happened contingent links, without preventing any possible duration of the future contingent links [16]. Several algorithms have been proposed to check the dynamic controllability of a constraint network [12]; eventually, Morris showed that in the framework of *STNU* the checking controllability algorithm is polynomial, i.e., $O(n^4)$, w.r.t. the number of *STNU* nodes [11]. In [8, 9], Hunsberger highlights some issues in the approach proposed by Morris and Muscettola and proposes a stronger definition of dynamic execution strategies that fixes these problems and puts the checking algorithm on a more solid theoretical foundation.

In [15], Tsamardinos et al. address the challenge of conditional planning developing the Conditional Temporal Problem (*CTP*) formalism, an extension of standard temporal constraint-satisfaction processing models used in non-conditional temporal planning. Informally, a *CTP* is a temporal problem where some constraints hold only when specific conditions are verified. Therefore, a *CTP* may admit different executions (i.e., conditional plans). The framework allows for the construction of conditional plans that are guaranteed to satisfy complex temporal constraints, but it does not consider issues related to the design of (structured) workflow schemata. Finally, the authors propose a classification of *CTP* consistency (corresponding to the controllability in *STNU*) with an analysis of the computational complexity of checking different kinds of consistency.

3. THE WORKFLOW CONCEPTUAL MODEL

In this paper we adopt the workflow conceptual model by Combi et al. without considering temporal granularities [3, 4]: a structured workflow is represented as a *workflow schema*, a digraph where nodes correspond to *activities* and edges represent *control flows* that define activity dependencies on the order of execution. In the following we will briefly sketch the basic constructs of the model. A workflow schema (graph) $WG = (N, T, T_S, T_E, C, F)$ is a six-tuple such as: $N = T \cup C$ is a finite set of activities (nodes), $T \subseteq N$ is a finite set of tasks, $T_S \in T$ is the Start node, $T_E \in T$ is the End node, $C \subset N$ is a finite set of connectors, and $F \subset N \times N$ is the control flow relation (edges). *Tasks*, depicted as rounded boxes, represent elementary work units that will be executed by external agents. Each task has the mandatory attribute *duration*, an unmodifiable temporal range specifying the allowed temporal spans for its execution. *Connectors*, depicted as diamonds, represent internal activities executed by the *WfMS* to achieve a correct and coordinated execution of tasks. Each connector has the mandatory attribute *duration* specifying the temporal spans allowed to the *WfMS* for executing it. The value of a connector *duration* can be modified at run time to guarantee the right coordination, and the effective duration is decided by the *WfMS*. A connector is either a *split* or a *join*. *Split* connectors split the incoming control flow into two or more flows. The set of nodes that can start their execution is given by the kind of split connector: *Total*, *Alternative* or *Conditional*. *Join* connectors realize the complement operation of *Split*: two or more incoming control flows are joined into outgoing one. A *join* connector can be either *And* or *Or*. Every edge of the graph has a temporal property, *delay*, to denote the allowed times that can be spent by the *WfMS* for possibly delaying the enactment of the second activity according to the given temporal constraints. The model is structured, i.e., there are suitable correspondences and proper nesting between splits and joins: for example, an *Alternative* split must have a corresponding *Or* join and between these two connectors there cannot be other splits without corresponding joins. The Start T_S node and the End T_E one represent the start and the end nodes of the workflow, respectively; they are graphically represented by a circle with one ingoing/outgoing edge, respectively, without any temporal attribute.

Allowed durations/delays are expressed by ranges like [MinD, MaxD] where $0 \leq \mathsf{MinD} \leq \mathsf{MaxD} \leq \infty$. Besides the basic temporal constraints, it is possible to define several other kinds of temporal constraints as the *relative* constraints, depicted as dashed oriented edges. A *relative constraint* limits the time distance (duration) between the starting/ending instants of two non-consecutive workflow activities expressed according to the following pattern: $\langle \mathsf{I_F} \rangle [\mathsf{MinD}, \mathsf{MaxD}] \langle \mathsf{I_S} \rangle$, where (i) $\langle \mathsf{I_F} \rangle$ marks which instant of the First activity to use ($\langle \mathsf{I_F} \rangle = \mathsf{S}_{\langle \mathsf{activity} \rangle} \mid \mathsf{E}_{\langle \mathsf{activity} \rangle}$ as the starting/ending execution instant, respectively; the subscript can be omitted if it is clear from the context.); (ii) $\langle \mathsf{I_S} \rangle$ marks the instant for the Second activity in the same way; (iii) [MinD, MaxD] represents the allowed range for the time distance between the two instants $\langle \mathsf{I_F} \rangle$ and $\langle \mathsf{I_S} \rangle$. It is assumed that $-\infty \leq \mathsf{MinD} \leq \mathsf{MaxD} \leq \infty$.

To *perform* a workflow according to a given schema, a *WfMS* has to assign tasks to agents and execute connectors/edges in the order fixed by the control flow starting from the Start and observing all temporal constraints. If the schema does not contain *Split* connectors, then the flow is a simple path and, therefore, at each instant only one activity is running (i.e., connector/task/edge execution). If the schema contains *Split* connectors, then it is possible that in some instants either multiple tasks are ran by external agents or the *WfMS* has to execute multiple connectors/edges in parallel. More formally, let S_T and S_C be the set of all starting instants of tasks and connectors, respectively and E_T, E_C the similar set considering the ending instants. We define a *schedule* of a schema as:

DEFINITION 1 (SCHEDULE). *A schedule s of a workflow schema is a mapping $S_T \cup S_C \cup E_C \to \mathcal{T}$ (being $\mathcal{T}$ the time domain), i.e., the time assignment to the starting instant of tasks and to the starting and ending instants of connectors.*

In general, it is not possible to determine a full-defined schedule of a schema in advance because it is possible that starting instant of some activities may depend on the ending instants of the previously executed tasks. Therefore, it is important to determine if the designed ranges of all activities and delays allow the *WfMS* to determine a schedule at run time that satisfies all temporal constraints. Hence, it is useful to consider the concept of *range schedule*:

DEFINITION 2 (RANGE SCHEDULE). *A range schedule r_s of a workflow schema is a mapping $S_T \cup S_C \cup E_C \rightarrow \mathcal{T}^2$ that, given a starting/ending instant x, return the range $[l,u]$, where $l,u \in \mathcal{T}$, such that for each value t in $[l,u]$ there exists at least one schedule s having $s(x) = t$.*

If a workflow schema contains one or more *Conditional* or *Alternative* connectors, not all the cases (i.e., executions) perform exactly the same set of tasks. Hence, *workflow path* (*wf-path*) denotes a workflow subgraph in which all alternative or conditional connectors have exactly one successor, i.e., a simple path representing one possible execution of the workflow. A *wf-path* can be briefly represented by a string containing the task labels of the *wf-path* sorted w.r.t. their execution order and separated by a dash if the order is sequential or by a vertical bar if the order is parallel: e.g., the *wf-path* T1-T2-T3 represents a *wf-path* where tasks are executed in the sequential order T1, T2, T3 while T1-(T2|T3) represents a *wf-path* where task T1 is executed before concurrent tasks T2 and T3. It is worthy noting that the set *WfP* of all *wf-paths* of a workflow schema may have an exponential cardinality w.r.t. the number of conditional or alternative connectors present in the schema. As regards the relationship between schedules and *wf-paths*, since different *wf-paths* may require different schedules, it is useful to define the following concept:

DEFINITION 3 (EXECUTION STRATEGY). *An execution strategy St is a mapping $St : WfP \rightarrow Rs$, where Rs is the set of all possible range schedules. For each $p \in WfP$, it always returns a range schedule $St(p)$ such that it contains all the time values for each starting/ending point belonging to schedules satisfying all the temporal constraints of p.*

Following *wf-path*, another important concept is that of *prefix*. A *prefix* of an activity/edge *y* is defined as the set P_y of all the *wf-paths* that have the same successor for each alternative or conditional connector that *precedes y*. An activity *x* precedes an activity *y* if either it belongs to the predecessors of *y* or precedes an activity of this set. The *prefix* of an activity/edge is useful to consider all possible *wf-paths* that can be followed after the execution of the considered activity/edge. Hereinafter a prefix of a given activity/edge *y* is represented by the *wf-path* notation specifying the common part that *wf-paths* of the prefix share.

4. THE CONTROLLABILITY OF WORKFLOW SCHEMATA

A workflow schema is *controllable* if the *WfMS* is able to perform any *wf-path* satisfying all relative constraints, all delays, all connector durations without setting (allowed) task durations involved in the *wf-path*, i.e., if an *execution strategy* exists. The problem of controllability checking arises when there is at least one relative constraint that involves two or more tasks.

In [4, 5], the authors discussed how to check the controllability of all patterns that can be present in sequential paths or in parallel ones and, then, they extended the pattern analysis to workflow schemata. Considering workflow schemata, they showed that there are different kinds of controllabilities for a schema and, therefore, they proposed a general algorithm to determine the kind of controllability of any workflow schema without facing the issue of the computational complexity of the problem. Here, we summarize the results before introducing the complexity analysis.

Regarding pattern controllability, Fig. 1 and Fig. 2 depict common patterns in sequential paths and in parallel ones, respectively.

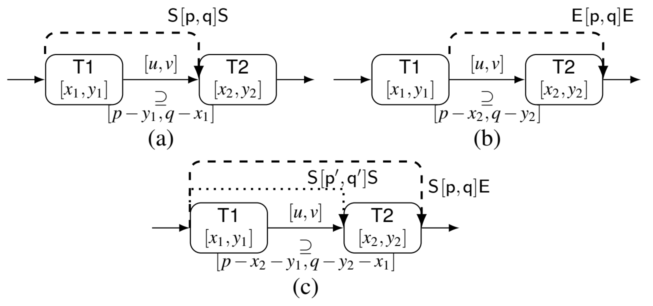

Figure 1: Three sequential patterns with a relative constraint. Patterns are significant if $0 \leq p \leq q$. The relative constraint and task ranges yield a range adjustment of the delay between the two tasks. Hence, a pattern is controllable if the designed range $[u,v]$ contains the derived range w.r.t. task durations and relative constraint range. For example, pattern (a) is controllable if $[u,v] \supseteq [p-y_1,q-x_1]$. In (c) the relative constraint S[p′,q′]S (dotted) is induced by S[p,q]E and it is determined by applying pattern (b).

Fig. 1 shows two fundamental sequential patterns and the most frequently derived one. Fig. 2 shows four basic parallel patterns, each containing a relative constraint. Pattern (e) is the most interesting one. Due to lack of space, we cannot report all the analysis made in [4, 5]. Here we note only that: (i) if $q < 0$, then it is sufficient to choose a suitable value in the range $[x_1-q, y_1-p]$ (depending on the T1 duration) as delay of the edge T1-A to control the pattern, (ii) if $p \geq 0$, it is sufficient to fix the delay of T1-A to be $[y_1-q, x_1-p]$ to have the controllability, (iii) if $p < 0$ and $q \geq 0$, it is not possible to set a single range to guarantee the controllability but it is necessary to set a new constraint (the augmented *wait* constraint) between the Start of T1 and A that is conditioned by the End of T1 (E_{T1}). The augmented *wait* constraint has the special label $\langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle$ that means: A could occur either when (1) "T1 has ended and within $|p|$ time units" or when (2) "y_1-q time units have elapsed since the start of T1 and T1 has not yet finished" (if A does not occur when condition (2) holds, the following end of T1 will trigger condition (1)). Sometimes the *wait* constraint can be simplified: if $(y_1-q) \leq x_1$, then a lower bound can be set because condition (2) is always verified before the end of T1: so the constraint can be represented as $[y_1-q, E_{T1}-p]$. Since a *wait* constraint $\langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle$ is meaningful only at run time, in order to evaluate its effect on controllability it is necessary consider its equivalent temporal range during controllability check; it has been shown that the equivalent range is one of the following: $[y_1-q, y_1-p]$ if $y_1-q \leq x_1$, $[x_1, y_1-p]$ otherwise.
Moreover, in [4, 5] the authors also discussed how to propagate wait constraints to other links in the context of workflow patterns as summarized in Fig. 2-(f)-(g) for the regression through a connector and through a task, respectively.

In general, given a *wf-path*, the controllability analysis requires checking the controllability of all temporal constraints (i.e., between any pair of nodes). To determine *all* temporal constraints, it is sufficient to transform the *wf-path* into the equivalent instance of Simple Temporal Problem (*STP*) and apply an *all-pairs shortest path* algorithm, as Floyd-Warshall [6]: if the original network has all consistent constraints, the algorithm determines the minimal satisfiable temporal constraint between any pair of nodes of the network, otherwise it signals the inconsistency state. Hence, after the constraints propagation on all possible pairs of nodes and a further consistency check, there are four possible outcomes: (i) the consistency check fails, (ii) the new constraint ranges are equal to

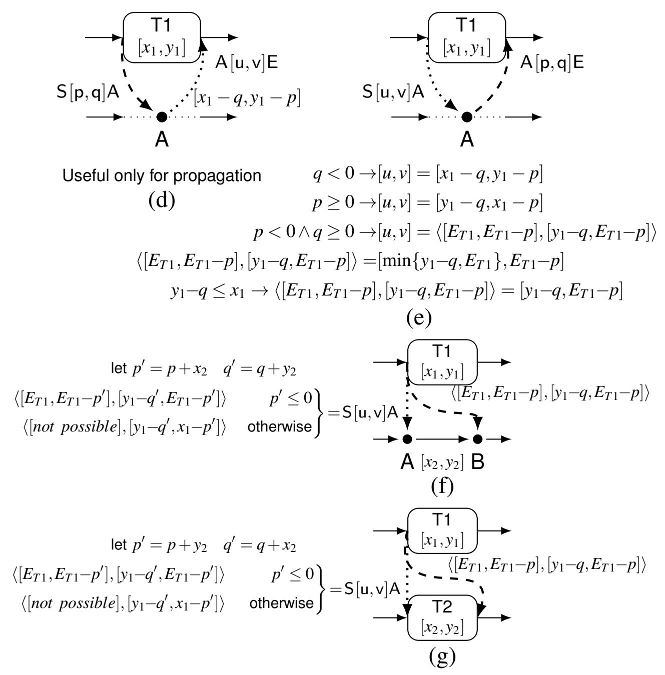
T1
$[x_1, y_1]$
A[u,v]E
S[p,q]A
$[x_1-q, y_1-p]$
A
Useful only for propagation
(d)
T1
$[x_1, y_1]$
A[p,q]E
S[u,v]A
A
$q<0 \rightarrow [u,v] = [x_1-q, y_1-p]$
$p \geq 0 \rightarrow [u,v] = [y_1-q, x_1-p]$
$p<0 \wedge q \geq 0 \rightarrow [u,v] = \langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle$
$\langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle = [\min\{y_1-q, E_{T1}\}, E_{T1}-p]$
$y_1-q \leq x_1 \rightarrow \langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle = [y_1-q, E_{T1}-p]$
(e)
let $p' = p + x_2$ $q' = q + y_2$
$\langle [E_{T1}, E_{T1}-p'], [y_1-q', E_{T1}-p'] \rangle$ $p' \leq 0$
$\langle [not\ possible], [y_1-q', x_1-p'] \rangle$ otherwise
= S[u,v]A
T1
$[x_1, y_1]$
$\langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle$
A $[x_2, y_2]$ B
(f)
let $p' = p + y_2$ $q' = q + x_2$
$\langle [E_{T1}, E_{T1}-p'], [y_1-q', E_{T1}-p'] \rangle$ $p' \leq 0$
$\langle [not\ possible], [y_1-q', x_1-p'] \rangle$ otherwise
= S[u,v]A
T1
$[x_1, y_1]$
$\langle [E_{T1}, E_{T1}-p], [y_1-q, E_{T1}-p] \rangle$
T2
$[x_2, y_2]$
(g)

Figure 2: Four parallel patterns with a relative constraint. We remember here that $p \leq q$. The dotted edges are induced relative constraints by the composition of the given relative constraint and the T1 duration. For sake of simplicity, we put A in the labels of relative constraints, as A could represent either a starting or an ending instant of an activity. In (e) the designed range $[u, v]$ has to be set to one of derived ranges according to p and q values. When $p < 0 \wedge q \geq 0$ holds, the constraint between the start of T1 and the event A becomes a *wait* constraint, that can be turned to a simple range only at run time. In (f) and (g) the relative constraints are *wait* constraints and they are propagated through an edge/connector (f) and through a task (g).

the corresponding old ones, (iii) at least one task range has been *squeezed* (the new range has a strictly tighter lower bound or upper bound), and (iv) only non-task range(s) has (have) been squeezed. In (i) and (iii) cases the *wf-path* is not controllable, in (ii) case the controllability analysis is completed and the *wf-path* is called *controllable*, and in (iv) case it is necessary to check the controllability of new ranges and, then, to apply the *all-pairs shortest path* algorithm again, in order to verify if the stable state has been reached. The algorithm just described is a pseudo-polynomial one since the time to reach the stable state could depend on some duration range values. In [11], Morris describes an advanced polynomial-time algorithm for the controllability of Simple Temporal Problem with Uncertainty (*STNU*) that may be used to check the controllability of a *wf-path* in a $O(n^4)$ time, where n is the order of the *wf-path*.

When a whole workflow schema is considered to check the controllability, it is necessary to consider all the possible *wf-paths* together because the controllability of each *wf-path* in isolation does not guarantee the controllability of the overall schema [5]. It has been shown that there are three kinds of controllability for a workflow schema: *strong, weak*, and *history-dependent controllability*. *Strong Controllability* (SC) refers to the fact that it is possible to derive a single duration range for each connector/edge ensuring the controllability of each *wf-path* of the workflow schema, i.e., the execution strategy *St* is such that, for every pair of *wf-paths* p_1 and p_2, and starting/ending instant x executed in both *wf-paths*, $[St(p_1)](x) = [St(p_2)](x)$. As regards the prefix concept, the duration range of each connector/edge is common to all the different prefixes of the considered connector/edge and it guarantees the controllability of all the *wf-paths*. Since SC requires the independence of controllability from the past execution flow (i.e., prefix), allowing a temporal range to depend on the past execution results in the concept of *History–Dependent Controllability* (HDC). A workflow schema is *history-dependently controllable* if each connector/edge has (possibly) different duration ranges for different prefixes: for each prefix, the given duration range is common to all the *wf-paths* of that prefix. In other words, the execution strategy *St* guarantees that, for any starting/ending instant x and for every pair of *wf-paths* p_1 and p_2 belonging to x prefix, $[St(p_1)](x) = [St(p_2)](x)$. In this case the duration range of a connector/edge could depend on the executed tasks (i.e., the history), but does not prevent the *WfMS* to execute any possible future task. If the property of being able to follow any possible future execution of the workflow schema is not guaranteed, it could be that a workflow schema is composed by several *wf-paths* controllable in isolation, i.e., an execution strategy exists, but some controllable execution could be prevented at run time: this property is called *Weak Controllability* (WeC).

Regarding the check of a workflow schema, in [5] a simple exponential algorithm is proposed, *controllabilityCheck(G)*, that determines whether a given workflow schema G is controllable, its kind of controllability, and duration ranges for connectors/edges (duration ranges of tasks must be leaved unchanged). We refer the reader to [5] for the complete algorithm and its analysis. *controllabilityCheck(G)* verifies that each single *wf-path* is controllable in isolation and then verifies whether there is a single duration range for each connector/edge for all the *wf-paths* containing it. If it is the case, then the workflow schema is strongly controllable and the derived ranges have to be used and possibly refined at run time. Otherwise, the algorithm verifies whether the workflow schema is history-dependently controllable: it considers the (possibly several if there are parallel flows) last Or-*join*s of the workflow schema. Or-*join*s are considered as they are responsible of varying the number of prefixes (i.e., histories) of all activities following them. Starting from the last Or-*join*s, it verifies that for each connector/edge of each Or-*join* prefix there is a suitable range not preventing any possible future execution path. This range could be different according to the considered connector/edge prefix and ranges related to different prefixes could even be disjoint. If this check phase is positive, then the workflow schema is history-dependently controllable, otherwise it is weakly controllable. In the worst case, the time complexity of *controllabilityCheck(G)* is $O(|WfP|^2\ n^3\ MaxRange^2)$, where n is the order of the graph and *MaxRange* is the maximum integer value present among the temporal durations/delays. It's worth noting that $|WfP| \leq 2^c(a+1)^j$, where c is the total number of conditional connectors, $a = \max\{|\text{outDegree}(a_l)|\}$ is the maximum among the out degrees of the alternative connectors and j is the number of the alternative connectors.

5. THE COMPLEXITY OF CONTROLLABILITIES

In the previous section, after the introduction of the possible kinds of controllability of a workflow schema, we mentioned the proposed exponential time algorithm that, given a workflow schema as input, determines whether the schema is controllable and, if it is, the final temporal ranges to use at run time. In order to evaluate the quality of the algorithm, in this section we investigate on the computational complexity of the controllability problem.

First of all, we observe the following implication chain among the three kinds of controllability: SC $\Rightarrow$ HDC $\Rightarrow$ WeC. Considering a workflow schema x, we define x to be co*Weakly Controllable* if at least one of its *wf-path* is not controllable, co*Strongly Controllable* if it is not *Weakly Controllable* or at least one connector/delay does not admit a single temporal range for all the *wf-paths*.

5.1 Weak Controllability Case

There is no possibility other than checking the controllability of each possible *wf-path* of a workflow schema separately to state if the schema is weakly controllable. The proof is given as corollary of the following theorem about co*Weak Controllability* (coWeC).

THEOREM 5.1. coWeC *is* NP*-complete.*

PROOF. It is sufficient to show the two following properties: 1) coWeC is in NP and 2) a NP-complete problem is polynomial-time reducible to it.
1) coWeC $\in$ NP: as previously discussed, it is possible to check the controllability of a *wf-path* in time $O(n^4)$ using the Morris algorithm [11]. Hence, given a workflow schema and one of its *wf-paths* as certificate it is possible to verify in polynomial time if the *wf-path* is not controllable and, therefore, to state that coWeC $\in$ NP.
2) SUBSETSUM $\prec_m$ coWeC: given a set of integers I and an integer s, the SUBSETSUM problem requires to verify if a non-empty subset $I' \subseteq I$ exists such that the sum of its elements is equal to s. SUBSETSUM is NP-complete [7].
To show a polynomial-time many-to-one reduction $\prec_m$ between SUBSETSUM and coWeC, it is sufficient to set up a polynomial-time function f between the set of instances of SUBSETSUM and the set of instances of coWeC such that each instance x is a positive instance of SUBSETSUM if and only if $f(x)$ is a positive instance of coWeC [7]. One possible $\prec_m$ between SUBSETSUM and coWeC is the following. Given the set $I = i_1, i_2, \ldots, i_n$ and the integer s, set up a schema with n sequential *Choice* blocks, each of them associated to an element of I, as depicted in Fig. 3. Each *Choice* blocks is made by a Conditional split connector, a task on *true* branch and the closing Or join connector. Considering the jth *Choice* block, the task on the *true* branch adds the associated element i_j to a global variable S spending i_j time units exactly. Each Conditional split randomly choices which branch to follow. After the last *Choice* block, there is another Conditional split where the equality between the global variable S and the input parameter s is tested. If the S value is equal to s, the task T_s is executed, otherwise nothing occurs. T_s is useful only as milestone to set up the relative constraint with range $[1, s-1]$ from Start. For sake of simplicity and without loss of generality, all connectors and delays have temporal range $[0,0]$ unit. If the instance of SUBSETSUM is positive, then any *wf-path* executing Ts (surely, at least one exists) is not controllable because the time required to calculate S is s and the relative constraint requires to start Ts within $(s-1)$ time units after Start. On the opposite direction, given an instance of the schema, if one *wf-path* is not controllable, then it contains the *true* branch of the last Conditional split. Hence, the corresponding SUBSETSUM instance is positive.

It is simple to verify that the construction of the workflow schema can be done in linear order time w.r.t. the size of the SUBSETSUM instance. □

COROLLARY 5.2. WeC *problem is* coNP*-complete.*

5.2 Strong Controllability Case

The SC definition requires that, for each connector/delay, the duration range has to be common to all *wf-paths*.

THEOREM 5.3. *Strong Controllability* $\in \Sigma_2^{\mathsf{P}}$.

PROOF. SC problem cannot be in coNP because it would mean that co*Strong Controllability* $\in$ NP and this is impossible (unless NP = coNP) since co*Strong Controllability* (coSC) would require to solve WeC, already shown to be coNP-complete. Let us consider the computational class $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$. It can be described as the class of languages L s.t. $L = \{\mathbf{x} \mid \exists \mathbf{y_1} \forall \mathbf{y_2} \text{ such that } (\mathbf{x}, \mathbf{y_1}, \mathbf{y_2}) \in R\}$, where R is a polynomially balanced, polynomial-time decidable relation and $\mathbf{x}, \mathbf{y_1}, \mathbf{y_2}$ are strings. A relation $R \subseteq (\Sigma^*)^3$ is polynomially balanced if, whenever $(\mathbf{x}, \mathbf{y_1}, \mathbf{y_2}) \in R$, it holds that $|\mathbf{y_1}|, |\mathbf{y_2}| \leq |\mathbf{x}|^k$ for some k [14].

Given a workflow schema, let us fix a total order among tasks and another total order among connectors/delays. Then, let $\mathbf{y_1}$ be the string built considering the temporal ranges of connectors/ranges allowing also empty ranges: in more details, let y_{1_i} be the temporal range of i-th connector/delay or the empty string. Let $\mathbf{y_2}$ be the similar string built considering the temporal ranges of tasks; note that it is possible that some values of $\mathbf{y_2}$ does not correspond to any *wf-path* because some combinations of temporal ranges and empty strings could be meaningless w.r.t. the given workflow schema. Setting $\mathbf{x}$ to be the string representation of the workflow schema, if a string $\mathbf{y_2}$ corresponds to a *wf-path* of $\mathbf{x}$, then, using Morris algorithm [11], it is possible to verify in polynomial time if a string $\mathbf{y_1}$ represents a set of temporal ranges that guarantees the controllability of *wf-path* $\mathbf{y_2}$ in $\mathbf{x}$. In other words, it is possible to build a polynomially balanced relation $R(\mathbf{x}, \mathbf{y_1}, \mathbf{y_2})$ decidable in polynomial time. A string $\mathbf{y_2}$ does not correspond to a *wf-path* when either one or more task temporal ranges are missing or there are too much temporal ranges (and therefore $\mathbf{y_2}$ corresponds to a mix of two or more *wf-paths*). Since the length of a *wf-path* is equal to the order of the schema graph at most and all *wf-paths* start from Start, checking the correctness of a given $\mathbf{y_2}$ can be done in linear order time. □

To our knowledge, SC seems not to be Σ_2^{P}-complete. We are investigating on the possibility that coSC belongs to AM(2), the class of problems decided by a 2-rounds interactive proof with public coins [1]. So far, we show that SC is coNP-hard.

THEOREM 5.4. SC *is* coNP*-hard.*

PROOF. Given a boolean expression ϕ with n variables in normal conjunctive form, the VALIDITY problem asks if ϕ is always true. VALIDITY is coNP-complete [14]. To prove the theorem, it is sufficient to show VALIDITY $\prec_m$ *Strong Controllability* and $\prec_m$ is computable in polynomial time.

The general idea is, given an expression ϕ with boolean variable $x_1, x_2, \ldots, x_n$, to build a workflow graph where the values of variables x_i are chosen by a sequence of n consecutive tasks, then the value of ϕ is evaluated and, finally, according to the ϕ value, the flow is divided into two branches: a strongly controllable branch (true branch) and a not strongly controllable one (false branch) as depicted in Fig. 4.

In more details, we assume that all variables are initially set to false. Each *Choice* block c_i is associated to variable x_i. The *Choice* connector chooses the branch to execute randomly and its execution can last 0 or 1 time unit. The true branch of c_i contains a task that sets the value of x_i to true and lasts one time unit exactly. For sake of simplicity and without loss of generality, all other connectors and delays have temporal range $[0,0]$ unit. After the last *Choice* block, there is a *Conditional* connector to evaluate ϕ. If ϕ results to be true, then task Tok is executed, otherwise task Tko is executed. Then, the workflow ends. The only relative constraint is between Start and End and it requires that the execution time has to be in the range $[n, 2n]$ time units.

If the VALIDITY instance is positive, then each possible *wf-path* contains the true branch of ϕ? connector. Setting the duration of

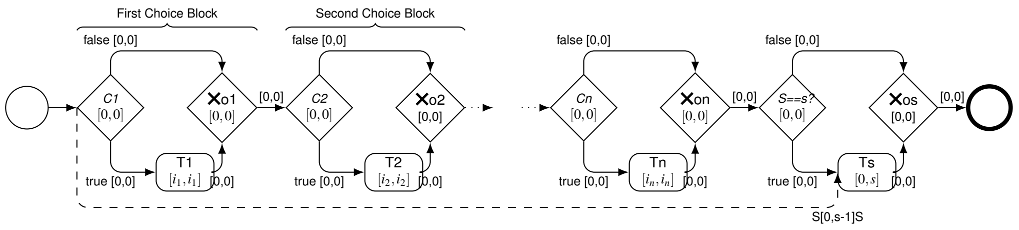

Figure 3: The workflow schema corresponding to an instance of SUBSETSUM.

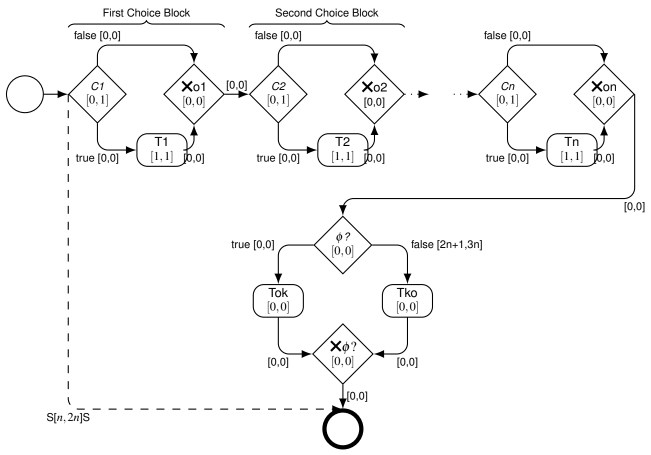

Figure 4: The workflow schema corresponding to an instance of VALIDITY.

all conditional connectors of choice blocks to one time unit, the duration of each possible *wf-path* is between n and $2n$ time units satisfying the intertask constraint and the SC condition. Hence, the schema is *Strongly Controllable*. If the VALIDITY instance is negative, then at least one *wf-path* executes the false branch of ϕ? connector spending at least $2n+1$ time units only for the execution of the branch. Since the upper bound of the overall relative constraint is $2n$, the *wf-path* is not controllable for any duration of conditional connectors, so the schema is not *Strongly Controllable*.

It is simple to show that the schema construction can be done in polynomial time. □

5.3 History-Dependent Controllability Case

The *History–Dependent Controllability* definition requires to determine, for each connector/delay, possible different temporal ranges, one for each prefix of the given connector/delay, such that the controllability of all *wf-paths* of the associated prefixes is guaranteed.

HDC problem cannot be in coNP class for the same reason explained for SC: it would mean that co*History–Dependent Controllability* (coHDC) $\in$ NP and this is impossible (unless NP = coNP) since coHDC could require to solve WeC, already shown to be coNP-complete.

Let us start with the simplified situation where the workflow schema does not contain any *Total* connector (i.e., no parallel flows). In a similar way as done for SC analysis, here we describe the HDC property representing the dependency of ranges w.r.t. the executed prefix in explicit form. Since a schema has different *wf-paths* (i.e., prefixes) only if it contains *Or* split or *Conditional* connectors (hereinafter, *split*), let us fix an order for all *splits* w.r.t. the distance from the Start. The first (and only one) *split* is at level 0 and it has only one prefix. Hence, the temporal ranges for the *split* and delays before it have to be the same for all *wf-paths* because there is only one prefix. After the first *split*, the number of prefixes increases and, therefore, it is possible to consider the specific prefix

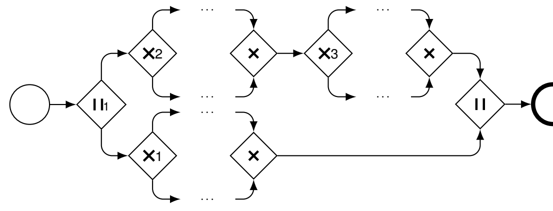

Figure 5: A workflow schema with X splits in parallel flows.

in order to determine duration ranges for all delays following the split until the next split. More formally, let us recursively denote by $\mathbf{p}_{i,i+1,\dots,k}$ $i = 0,\dots,k-1$ is the sequence of tasks from one *split* of level $k-1$ to one of level k, according to the given (meaningful) piece of flow $\mathbf{p}_{i,i+1,\dots,k-1}$; $\mathbf{p}_{0,1}$ is a task sequence from the first *split* to a *split* at level 1; $\mathbf{p}_{k-1,k}$ is a task sequence from the $k-1$th *split* to End, viewed as the only *split* at level k; note that the number k of levels is less than the number n of nodes. Moreover, let us recursively denote by $\mathbf{x}_{i,i+1}$ the sequence of delays/join durations following a *split* at level i and before the following (w.r.t. the flow) *split* at level $i+1$; $\mathbf{x}_{i,i+1}$ contains also the range of the *split* at level $i+1$. $\mathbf{x}_{S,0}$ is the sequence of delays between Start and the first *split* including also the duration of the *split*. If a *wf-path* does not contain *splits* of level greater than j $(j<k)$, then $\mathbf{p}_{j,j+1,\dots,k}$ and $\mathbf{x}_{j,j+1},\dots,\mathbf{x}_{k-1,k}$ are empty. According to this notation, HDC may be expressed as

$$\exists \mathbf{x}_{S,0} \forall \mathbf{p}_{0,1} \exists \mathbf{x}_{0,1} \forall \mathbf{p}_{0,1,2} \cdots \exists \mathbf{x}_{k-2,k-1} \forall \mathbf{p}_{0,1,\dots,k} \exists \mathbf{x}_{k-1,k}$$
$$R(\mathbf{x}_{S,0}, \mathbf{x}_{0,1}, \dots, \mathbf{x}_{k-1,k}, \mathbf{p}_{0,1,\dots,k})$$

where $R(\mathbf{x}_{S,0}, \mathbf{x}_{0,1}, \dots, \mathbf{x}_{k-1,k}, \mathbf{p}_{0,1,\dots,k})$ is the polynomial-time checkable predicate representing the controllability of the *wf-path* $\mathbf{p}_{0,1,\dots,k}$ when the delays and connectors ranges are $\mathbf{x}_0, \mathbf{x}_{0,1}, \dots, \mathbf{x}_{k-1,k}$. Given a schema with n nodes, the problem belongs to class Σ_n^p. In general, the problem is in PSPACE.

The description of the HDC property is more difficult when the schema contains parallel flows with alternative flows inside. Indeed, the order of execution of *splits* on different parallel flows cannot be fixed in general and, therefore, requiring HDC for any possible execution order of *splits* could result in a too strict condition w.r.t. the effective possible executions. For example, as depicted in Fig. 5, the possible execution orders for *splits* are: either X1 X2 X3 or X2 X1 X3 or X2 X3 X1. In principle, any of these orders is acceptable and could depend on the previous execution history. Thus, at design time we need to verify HDC considering all these orders but, in this way, we could specify constraints more strict than those required by the schema. Even in this case, the above logical characterization need to be extensively revised to guarantee that all the possible schemata are captured.

6. DISCUSSION AND CONCLUSIONS

In this paper we discussed the computational complexity of controllability problem of workflow schemata; in particular, we considered the three different kinds of controllability and we showed that WeC problem is coNP-complete, while SC problem $\in \Sigma_2^{\mathsf{P}}$ and it is coNP-hard. We are investigating on the possibility that coSC belongs to the class AM(2): it would mean that SC cannot be Σ_2^{P}-complete unless the polynomial hierarchy collapses to the third level ($\mathsf{PH} = \Sigma_3^{\mathsf{P}}$). Regarding the *history-dependent controllability* problem, we showed that it is a PSPACE problem and we are investigating about its hardness characterization.

Even with incomplete complexity characterization, we are able to say that the computational complexity of algorithms proposed in [5] cannot be lowered significantly: thus, in the domain of workflow systems, ad-hoc (approximate) algorithms need to be studied for both design time and run time controllability checking.

7. REFERENCES

[1] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge Univ. Press, 2009.

[2] C. Bettini, X. S. Wang, and S. Jajodia. Temporal reasoning in workflow systems. *Dist. & Paral. Data.*, 11(3):269–306, 2002.

[3] C. Combi, M. Gozzi, J. M. Juárez, B. Oliboni, and G. Pozzi. Conceptual modeling of temporal clinical workflows. In *TIME*, pages 70–81. IEEE, 2007.

[4] C. Combi and R. Posenato. Controllability in temporal conceptual workflow schemata. In U. Dayal, J. Eder, J. Koehler, and H. A. Reijers, editors, *BPM*, volume 5701 of *LNCS*, pages 64–79. Springer, 2009.

[5] C. Combi and R. Posenato. Towards temporal controllabilities for workflow schemata. In Markey and Wijsen [10], pages 129–136.

[6] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[8] L. Hunsberger. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In C. Lutz and J.-F. Raskin, editors, *TIME*, pages 155–162. IEEE, 2009.

[9] L. Hunsberger. A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In Markey and Wijsen [10], pages 121–128.

[10] N. Markey and J. Wijsen, editors. *TIME 2010 - 17th Int. Symp. on Temporal Repres. and Reas., Paris*. IEEE, 2010.

[11] P. Morris. A structural characterization of temporal dynamic controllability. In F. Benhamou, editor, *CP*, volume 4204 of *LNCS*, pages 375–389. Springer, 2006.

[12] P. H. Morris and N. Muscettola. Temporal dynamic controllability revisited. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 1193–1198. AAAI Press, 2005.

[13] Object Management Group (OMG). Business process definition metamodel (bpdm), Beta 1. http://www.omg.org, 2007.

[14] C. M. Papadimitriou. *Computational complexity*. Addison, 1994.

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

[16] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Exp. Theor. AI*, 11(1):23–45, 1999.