




Dynamic Controllability of Processes without Surprises

Johann Eder ^{*,†} , Marco Franceschetti [†]  and Josef Lubas [†] 

Department of Informatics-Systems, Universität Klagenfurt, 9020 Klagenfurt, Austria; marco.franceschetti@aau.at (M.F.); josef.lubas@aau.at (J.L.)

* Correspondence: johann.eder@aau.at

† These authors contributed equally to this work.

Abstract: Dynamic controllability guarantees that a process control can steer the execution of a business process without violating any temporal constraints although some tasks have uncontrollable durations. However, it has been shown that dynamic controllability may lead to process executions with the undesirable property that tasks have to be started or ended on extremely short notice. Sudden termination forces the agent to immediately terminate the execution of a process task without any prior notice in order to meet some temporal constraints. Semi-dynamic controllability guarantees dynamic controllability and the absence of sudden termination. Here, we show that dynamic controllability may also lead to the problem of sudden start, which forces the immediate start of a process task without prior notice. We formalize all constellations of temporal constraints causing sudden start and sudden termination in a process. We propose a technique to design processes in which activities can be dynamically dispatched without these surprises, i.e., with advance notice, and extend the notion of semi-dynamic controllability by also considering the sudden start. This leads to a sound and complete algorithm for checking the semi-dynamic controllability of time-constrained processes.

Keywords: process scheduling; contingent durations; sudden start; sudden termination; dynamic controllability



Citation: Eder, J.; Franceschetti, M.; Lubas, J. Dynamic Controllability of Processes without Surprises. *Appl. Sci.* **2022**, *12*, 1461. <https://doi.org/10.3390/app12031461>

Academic Editor: Ricardo Colomo-Palacios

Received: 23 December 2021

Accepted: 27 January 2022

Published: 29 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The progress of business process technology in the last decades led to a wide-spread adoption of process management across enterprises and organizations [1]. A business process consists of a set of tasks that are executed in coordination to achieve a goal, provide a service or produce a product (cf. [2,3]). For effective process management, modeling and verification of the temporal aspects of a business process are crucial [4].

Temporal aspects which may be included as part of a business process model are deadlines, task durations and other temporal constraints such as minimum and maximum time span between events [5]. Given a process model including temporal aspects, its verification has the goal of checking whether the model meets certain quality criteria. In particular, it is essential to know beforehand whether time failures, i.e., violations of temporal constraints, can be avoided with effective scheduling strategies for the dispatching of activities. Or in other terms, whether the controller of a business process (e.g., the process enactment service) can follow a strategy to execute the process without violating any temporal constraints [6]. In such a strategy, however, controllers have to cope with uncertainty and have to react to observations of external events.

The source of uncertainty that controllers have to cope with is the duration of process activities. In particular, researchers in the field of time management agree on distinguishing between two types of activities, depending on their duration [7]. On the one hand, there are activities whose duration is under the control of the execution environment (a process manager, a dispatching agent, an information system, ...). On the other hand, there are activities whose duration cannot be controlled, but merely observed at run time. These uncontrollable durations (and the corresponding activities) are called *contingent*. Frequent

causes of uncertain durations are the invocation of external services or the dependencies on other controllers in cross-organizational processes (see e.g., [8]) where service level agreements provide duration intervals.

For processes with contingent activities, *dynamic controllability* [6,7] is the notion for temporal correctness that is preferred to, e.g., the satisfiability of temporal constraints. Satisfiability requires the existence of one possible instance of a process that satisfies all temporal constraints. On the contrary, dynamic controllability requires that there exists a dynamic execution strategy for the process controller to set the start and end of activities in response to all possible durations of contingent activities revealed at run time. We refer to [9,10] as examples of techniques that can be adopted to efficiently check dynamic controllability of processes.

Despite being a much desirable property for a time-constrained process, dynamic controllability can present an undesirable side effect. Indeed, a dynamically controllable process might admit only dynamic execution strategies that force (1) some activities to start, without knowing yet when they need to finish (*sudden termination*), and (2) some activities to start immediately without prior notice (*sudden start*). We use the term *sudden* to describe such phenomena since they require, at run time, to schedule the termination, resp. start of a process activity immediately, without giving the executor any time to prepare for such an event.

In our prior work [11], we have addressed the sudden termination problem. We characterized *semi-contingent* activities, which require to set their duration, resp. their deadline, when they start, i.e., their duration can be chosen by the controller, but only until they start. In particular we have shown in detail one sudden termination pattern, and only sketched other patterns that may lead to such a sudden termination.

This paper is an extension of the work in [11]. As such, it covers all sudden termination patterns shown in [11] in more detail, and addresses the additional problem of sudden start. So the research question driving this extended paper is: *how to determine whether the start and end events of process tasks can be scheduled with a given advance notice, meeting all temporal constraints, despite uncertainties?*

Here, we propose a technique to support process designers in checking whether it is possible to steer a process with the guarantee that no sudden (start or end) event needs be dispatched at run time. The additional contributions of this paper with respect to [11] are the following:

- The characterization of the sudden start problem in addition to the sudden termination problem, which might arise in dynamically controllable processes.
- The procedure for the transformation of sudden start problems to sudden termination problems and the proof of its correctness.
- The detailed formal analysis of all the patterns potentially causing a sudden termination.
- A procedure to check whether a process model is semi-dynamically controllable, i.e., whether its events can be dynamically scheduled with a warning time, such that all temporal constraints are met despite uncertainties.

The remainder of this paper is structured as follows: in Section 2, we illustrate the problem and introduce the necessary background with the help of examples. In Section 3, we introduce a lean process model that allows the formulation of the problem. We formalize the sudden termination and sudden start problems, and semi-dynamic controllability in Section 4. In Section 5, we formalize the patterns that can induce the sudden termination problem, and show how to solve it. In Section 6, we show how to transform the sudden start problem into a sudden termination problem. In Section 7, we show how to check semi-dynamic controllability. In Section 8, we provide an implementation of a checking procedure. In Section 9, we discuss related work, and in Section 10 we draw conclusions.

2. Background and Motivation

2.1. Dynamic Execution Strategies

The goal of all this research into correctness of temporally constrained process models is to check at *design time*, whether it is possible to avoid temporal problems at *run time*. Temporal problems are primarily time failures, i.e., the violation of a temporal constraint.

So let us first have a look at run time. A process controller dispatches tasks according to the process model and the evolving execution history. The process model defines a partial order on tasks expressed typically by sequence, split and join nodes in the process model [4]. For all tasks the minimum and the maximum duration are known. The process controller has to observe temporal constraints, which are defined as maximum or minimum time gaps between the start and end event of tasks.

The controller assigns a time-point when a task has to be started. For each task the controller can do this assignment up to this very start-time. When does a task end? For some tasks the duration is determined by the environment, by “nature”, by other parties. The controller only knows that the end-time of such a task is between minimum and maximum duration after its start-time. These *contingent* durations [10] can only be observed by the controller, but the controller cannot influence them. For an example, the (regular) transfer of some funds between accounts at different banks in the EU takes between a minimum of 0 and maximum of 4 days, without the possibility for sender or receiver to influence the duration. The controller has to take this imponderability into account when controlling the execution of a process. For *non-contingent* durations, however, the controller might set not only the start-time but also the end-time (to any value between minimum and maximum duration after start).

For some process models, the controller is able to assign time-points for the events (start and end of tasks) such that the execution does not violate any temporal constraint, before the whole process even starts. We call such process models *strongly controllable* [6].

However, not all process models are strongly controllable. For an example, a process transfers some amount from one account to another at another bank. A constraint requires that within two days after the transfer is finished (the amount is credited at the recipient account) a notification should be sent out. It is easy to see that it is not possible to set a start date for the send-notification task before it is known how long the transfer took (between 0 and 4 days). Nevertheless, if the controller reacts dynamically after observing the actual duration, it is always possible to assign a start date for the send-notification task such that the constraint is satisfied. The controller can have a dynamic strategy for assigning execution time-points.

In general, at a time-point τ the controller knows about all events (start and end of tasks) that happened before τ . According to this information the controller assigns time points greater than τ to some start- and end-events of tasks, which did not yet take place. The controller applies an *execution strategy* determining at a given time point τ which values to assign to which event as a function of the process model and the history of the process execution up to τ .

Such an execution strategy has to consider, in particular, the observed contingent duration of tasks. If such a dynamic execution strategy exists, which guarantees that for all possible contingent durations the controller can assign time-points dynamically in a way that no constraint is violated, we call the process *dynamically controllable* [6].

Of course, not all process models are dynamically controllable. For an example, there is no dynamic execution strategy to avoid violating the constraint “the notification has to be sent 2 days before the amount is credited at the receiving account”. The process with this constraint is not dynamically controllable. Any assignment strategy for the “send-notification” task could lead to a violation of the constraint, due to the uncertainty in the time required for the money transfer.

Dynamic controllability is the most refined notion for temporal correctness of process models [6], as it is the most relaxed notion that still has the property that execution without

time failures is possible for all observable uncertainties. In other words, if a process is not dynamically controllable, the controller cannot avoid that time failures might occur.

The question, whether it is possible to check at design time whether a process model is dynamically controllable, attracted a lot of attention in research (see, e.g., [6,9,12–14]). Formal checks are necessary, as testing typically suffers from the combinatorial explosion of considering all possible combinations of durations of contingent activities. Recently rather efficient checking procedures were proposed and proven to be sound and complete [15]. The quest for efficient procedures for checking process models with additional control structures, constraint types, etc., however, still continues.

Nevertheless, also execution strategies for dynamically controllable process models can have problems or inconveniences as we showed in [11]. The controller might set the end-time of a non-contingent task at any time before that end-time. In particular, in the time frame between minimum and maximum duration after the start-time, the controller might even order to stop the task immediately. All viable dynamic execution strategies might make it necessary that the execution of a task with a non-contingent duration has to start without knowing its deadline yet. The controller has to suddenly force such a task to stop: the problem of *sudden termination*. We elaborate this phenomenon below in detail and complement it with the problem of *sudden start*: the execution of a task has to start immediately, without advance notice.

Of course, it depends on the type of activity and the context whether sudden start or sudden termination is acceptable. For some activities a sudden termination poses no problem, e.g., for waiting tasks: it may be fine to start a waiting task without knowing when it must end. However, for other activities (see examples below), a sudden termination is highly undesirable, unacceptable or even impossible. This is the case, e.g., for process tasks involving human actors, or invoking external services: a human actor may not accept being abruptly forced to stop performing a calculation but would prefer to know in advance when the calculation results are due in order to properly perform the task. Similarly, for an external service invocation it may be necessary to receive a result before being able to proceed with the process. Similar examples can be found for sudden start: for instance, it would be impossible for a lab technician to immediately start executing a task of analysis of hazardous material without a prior notice that would allow her to wear protective equipment before. In such a case, a prior notice of 30 min may be required for the technician to get ready to perform the analysis. Therefore, we argue that we have to represent in the process model, whether sudden start and sudden termination are acceptable for a task. The rest of the paper describes procedures for checking, at design time, whether these *surprises* could happen and whether it is in the power of the controller to prevent such surprises.

2.2. Sudden Termination

To be self-contained, we briefly revise the small example from [11] to explain the sudden termination problem. Consider the simplified procurement process in Figure 1. The controller may choose the duration of semi-contingent task *A* for arranging shipment of goods between 5 and 7 days. *A* is bound to a contingent task *R* of receiving payment that lasts between 3 and 5 days. A *lower-bound constraint (lbc)* demands that the end of *A* is at least 3 days after the end of *R* to allow a customer to cancel the order. An *upper-bound constraint (ubc)* states that *A* has to finish within 4 days after *R* to guarantee timely delivery. Now, let's assume *R* should start at time point 10, and hence ends anytime between 13 and 15. Is there a choice for the start time and duration of *A* satisfying the constraints? After some trials you see: no! *A* might, e.g., start at 12 and end between 17 and 19, but 19 is too late, if *R* ends at 13, and 17 is too early, if *R* ends at 15. At 12 the end of *R* is unknown.

One can observe that it is impossible to determine a duration for *A* such that for all possible durations of *R* the constraints are satisfied. The end of the ongoing activity *A* can only be scheduled when the end of *R* has been observed.

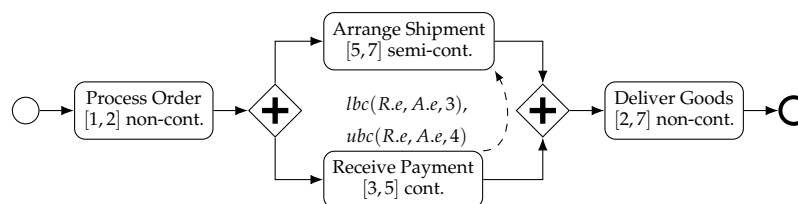


Figure 1. Example process with a potential sudden termination of an activity.

As shown in [11], treating semi-contingent activities for checking dynamic controllability is not obvious and requires special strategies. Indeed, if they are considered as non-contingent activities, they may be subject to a sudden termination. Instead, if they are treated as contingent activities, we are unnecessarily strict and reject processes that could be scheduled without sudden termination. Therefore, in [11], we introduced the notions of *semi-contingent activities* and *semi-dynamic controllability* to adequately deal with such activities and their durations.

There exist various semi-contingent activities for which the end time is decided when they start, and cannot further be changed. An example is executing a batch computation using a configurable cloud Virtual Machine service. A client can choose the VM configuration, setting the memory and number of CPU cores, before starting the computation. The execution time depends on the VM configuration. Once the computation starts, the client cannot force it to terminate to get the results faster, but needs to wait for completion. Another example is shipping a package with express delivery service or regular service. Once the package is handed in to the courier and shipment starts, the delivery time is set and cannot be changed, e.g., after 1 day for express and 3 days for regular service.

2.3. Sudden Start

We explain the sudden start problem with another small example. Figure 2 depicts a simplified manufacturing process. The process starts with the preprocessing of some raw materials, which takes 20 to 24 h, depending on the quality of the materials. Subsequently, assembly of component A and of component B start, requiring 3 to 5 h, and 1 to 2 h, respectively. The time required for assembling component A cannot be controlled, since it also heavily depends on the quality of the raw material. Afterwards, the two components are assembled (2 to 4 h), and finally the product can be packaged. The final assembly must start at least 1 h and at most 2 h after completion of component A assembly.

One can observe that, due to the uncontrollable duration of component A assembly, it is not known when final assembly has to start, until the task has to start, which means that it cannot be scheduled beforehand. Similar to the sudden termination, such a behavior may as well be unacceptable, or impossible. Especially for activities that require some preparation, it would be desirable, instead, that the agent receives an advance notice some time k before the activity has to start.

An example for an activity whose start time should be known in advance from the administrative workflows is, for an employee, being asked to write a report immediately, interrupting any other task that was being performed before. An example from the aviation industry is boarding a plane: the boarding time should be announced at least 30 min prior to its start, otherwise passengers might not make it in time to reach the gate. Similar and from the same domain, the air traffic control should communicate to the pilot of a plane, flying on hold due to high traffic, the landing time at least 5 min in advance, in order for the plane to perform the necessary maneuvers. Or, before telling the cabin crew to sit for (imminent) landing, the pilot gives them an advance notice 5 min before (“*prepare for landing*”), so that the crew will not be put in danger and forced to rush to their seats.

In Section 4 we will show how to address the sudden start problem, by transforming it into a sudden termination problem, such that the start time of an activity can be scheduled with an advance of k time units.

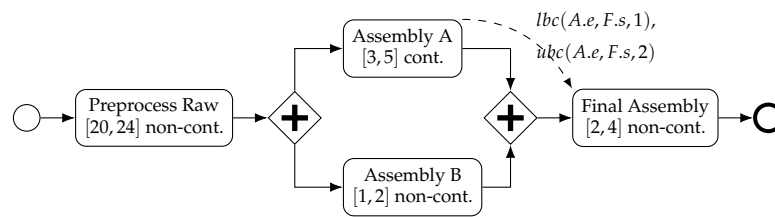


Figure 2. Example process with a potential sudden start of an activity.

3. Process Model

3.1. Process Model with Temporal Constraints

As we have shown in [11], a minimal process model is sufficient to capture the patterns for which sudden start and sudden termination may occur without loss of general applicability. The process model we consider supports the most common control flow patterns found in the practice, as shown in [16]: sequence, parallel splits and joins, and exclusive splits and joins.

The temporal aspect of the process model includes activity durations, process deadline, and upper- and lower-bound constraints between events (i.e., start and end of activities). We measure time in *chronons*, representing, e.g., hours, days, ..., which have domain the set of natural numbers, and are on an increasing time axis starting at *zero*. On such a time axis, we measure a duration as the distance between two points. A point on the time axis is defined by its distance from *zero*.

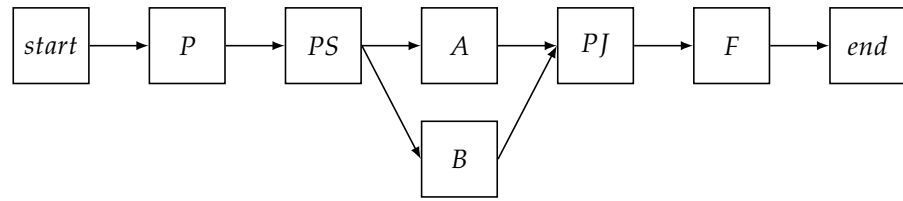
In order to be able to talk about the sudden start and sudden termination, durations (and the corresponding activities) can be contingent, semi-contingent and non-contingent. Contingent durations cannot be controlled, thus it cannot be known when the corresponding activities will actually terminate. Non-contingent durations, on the other hand, can be controlled by the process controller at any time. This means, in particular, that the corresponding activities are allowed to start without knowing the time when they have to end, i.e., they can be suddenly terminated. Semi-contingent durations require to know, at the start time of the corresponding activities, when they must terminate. So the process controller can set the duration (hence their end time) of a semi-contingent activity until the activity starts.

Definition 1 (Process Model). A process P is a tuple (N, E, C, k, Ω) , where:

- N is a set of nodes n with $n.type \in \{start, activity, xor - split, xor - join, par - split, par - join, end\}$. Each $n \in N$ is associated with $n.s$ and $n.e$, the start and end event of n . From N we derive the set of all node events $N^e = \cup\{n.s, n.e | n \in N\}$.
- E is a set of edges $e = (n1, n2)$ defining precedence constraints, which mean that node $n2$ is executed after the execution of $n1$ is completed.
- C is a set of temporal constraints:
 - duration constraints $d(n, n_{min}, n_{max}, durationtype) \forall n \in N$, where $n_{min}, n_{max} \in \mathbb{N}$, $durationtype \in \{c, sc, nc\}$, stating that n takes some time in $[n_{min}, n_{max}]$. n can be contingent ($durationtype = c$), semi-contingent ($durationtype = sc$), non-contingent ($durationtype = nc$);
 - upper-bound constraints $ubc(a, b, \delta)$, where $a, b \in N^e, \delta \in \mathbb{N}$, requiring that $b \leq a + \delta$;
 - lower-bound constraints $lbc(a, b, \delta)$, where $a, b \in N^e, \delta \in \mathbb{N}$, requiring that $b \geq a + \delta$.
- $k : N \rightarrow \mathbb{N}$ is a function that associates each node $n \in N$ with a required advance notice $k(n)$ for its start, meaning that if n has to start at time t , this has to be known at a time $k(n)$ before t . We call such a point in time the dispatcher event $n.d$ of n .
- $\Omega \in \mathbb{N}$ is the process deadline (the longest admissible duration between start.s and end.e).

Figure 3 shows the representation of the process in Figure 2 according to the process model of Definition 1.

We now define the temporal semantics of the process model.



$$\begin{aligned}
 N^e &= \{start.s, start.e, P.s, P.e, PS.s, PS.e, A.s, A.e, B.s, B.e, PJ.s, PJ.e, F.s, F.e, end.s, end.e\} \\
 C &= \{d(start, 0, 0, nc), d(P, 20, 24, nc), d(PS, 0, 0, nc), d(A, 3, 5, c), d(B, 1, 2, c), \\
 &\quad d(PJ, 0, 0, nc), d(F, 2, 4, sc), d(end, 0, 0, nc), ubc(A.s, F.s, 2), lbc(A.e, F.s, 1)\} \\
 k(F) &= 5, \quad k(n) = 0 \text{ for } n \neq F \\
 \Omega &= 42
 \end{aligned}$$

Figure 3. Process model for the process of Figure 2. Sets N (nodes) and E (edges) are represented as a graph.

3.2. Temporal Semantics

We define the temporal semantics of temporally constrained process definitions by defining which executions of a process definition are admissible [4], through the concept of *valid scenario*. A scenario represents a run of a process (i.e., a process instance) with assignments of timestamps specifying:

- when each event (starting and ending of process steps) occurred, and
- when each advance notice for the start of an activity is given.

Essentially, a scenario is a mapping of events to timestamps, independent of who assigns these timestamps. Hence, we define a scenario as follows:

Definition 2 (Scenario). Let $P(N, E, C, k, \Omega)$ be a process model. A scenario for P is a function $\sigma : N^e \cup \{n.d | n \in N\} \rightarrow \mathbb{N}$, assigning:

- to each start event $n.s$ of a node n a timestamp $\sigma(n.s)$, denoted as $\overline{n.s}$;
- to each end event $n.e$ of a node n a timestamp $\sigma(n.e)$, denoted as $\overline{n.e}$;
- to each dispatcher event $n.d$ a timestamp $\sigma(n.d)$, denoted as $\overline{n.d}$, at which the notice of start of n is given.

A scenario is *valid*, if it is such that all temporal constraints are satisfied:

Definition 3 (Valid Scenario). Let $P(N, E, C, k, \Omega)$ be a process model. Let σ be a scenario for P . σ is a valid scenario for P iff:

1. $\forall (n1, n2) \in E, \sigma(n1.e) \leq \sigma(n2.s)$;
2. $\forall d(n, n_{min}, n_{max}, [c|sc|nc]) \in C, \sigma(n.s) + n_{min} \leq \sigma(n.e) \leq \sigma(n.s) + n_{max}$;
3. $\forall ubc(a, b, \delta) \in C, \sigma(b) \leq \sigma(a) + \delta$;
4. $\forall lbc(a, b, \delta) \in C, \sigma(a) + \delta \leq \sigma(b)$;
5. $\sigma(end.e) \leq \sigma(start.s) + \Omega$;
6. $\forall k(n) > 0, \sigma(n.d) + k(n) \leq \sigma(n.s)$.

An example for a scenario for the example process of Figure 3 is given by the mapping $\sigma = \{start.s \rightarrow 0, start.e \rightarrow 0, P.s \rightarrow 1, P.e \rightarrow 22, PS.s \rightarrow 23, PS.e \rightarrow 23, A.s \rightarrow 23, A.e \rightarrow 27, B.s \rightarrow 24, B.e \rightarrow 26, PJ.s \rightarrow 27, PJ.e \rightarrow 27, F.s \rightarrow 29, F.e \rightarrow 33, end.s \rightarrow 33, end.e \rightarrow 33, k(F) \rightarrow 24\}$. One can verify that the scenario fulfills all requirements 1-6 from Definition 3, hence it is valid.

A scenario is an assignment that results from following an execution strategy (also called schedule), which states when each controllable event should occur and when each advance notice should be given. For a process to be dynamically controllable, it is necessary that there exists a dynamic execution strategy, in which the decision about starting and ending activities is made based on the timestamp of all earlier events. In particular, a

process is dynamically controllable if a dynamic execution strategy for it exists, which leads to a valid scenario.

Several approaches have been developed in prior work for checking the dynamic controllability of processes. Here, we consider the approach based on mapping a process model to a Simple Temporal Network with Uncertainty (STNU), and applying constraint propagation methods that are proven to be sound and complete for checking STNU dynamic controllability, such as [9]. We present our approach in detail in Section 7.

In Section 2 we gave examples of dynamically controllable processes that are affected by sudden termination and sudden start. In the next section, we give a formalization of the sudden termination and sudden start problems. We then define semi-dynamic controllability as the possibility to have a dynamically controllable process, which can be scheduled without any sudden termination or sudden start.

4. Sudden Termination and Sudden Start Problems, and Semi-Dynamic Controllability

In the following, we assume that a given process model P is dynamically controllable. With the terms *condition*, resp. *constraint* we indicate that a constraint is either explicitly stated in P or that it can be derived from the explicit constraints in P . Techniques for deriving implicit constraints from explicit ones are presented in Section 7. Finally, with the symbol Φ we designate the set of all implicit and explicit constraints valid in P .

4.1. Sudden Termination Problem

We briefly recap here the characterization of the sudden termination problem for a semi-contingent activity, and the conditions that have to be satisfied, such that a sudden termination problem might occur [11]. First, we highlight the distinction between sudden termination constellation (STC) and sudden termination pattern (STP). In an STP, constraints can only be satisfied with the sudden termination of an activity, while in the more general STC, it might depend on the controller, whether a sudden termination actually occurs. In an STP, sudden termination needs to necessarily happen, while in an STC that is not an STP it is possible to avoid sudden termination.

We have shown in prior work that an STC requires at least two constraints between two activities. Given two activities C (contingent) and S (semi-contingent), a sudden termination means that the admissible times for ending S depend on the observation of the end of C , and this is not known when S starts. This requires the existence of a constraint c_1 between the end events of C and S .

A sudden termination only occurs if C and S have to be executed concurrently, i.e., it is not possible that they are executed sequentially. This requires an additional constraint c_2 . When the end event $S.e$ of S can always be executed before the end event $C.e$ of C , it cannot depend on the observed duration of C . In a similar way, if S can always start after C has ended, the duration of C is already known at the start of S .

In an STC, for all possible start times of C , *there exist* certain start times for S , such that, no matter how long S is enforced to take, some (uncontrollable) durations of C exist, which lead to the violation of some temporal constraint in the process. Instead in an STP, for all possible start times of C , *for all* start times for S , no matter how long S is enforced to take, some (uncontrollable) durations of C exist, which lead to the violation of some temporal constraint in the process.

Definition 4 (Sudden Termination Constellation and Pattern). *Let P be a dynamically controllable process. Let S and C be activities in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let Φ be the set of constraints in P .*

*S and C are in a **sudden termination constellation (STC)** iff*

$$\forall C.s \exists S.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg \Phi$$

*S and C are in a **sudden termination pattern (STP)** iff*

$$\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg \Phi$$

We use the notation $STC_{S,C}$ to indicate that S and C are in a sudden termination constellation, and $STP_{S,C}$ for a sudden termination pattern.

If there is an STP in a process, sudden termination cannot be avoided without changing the process model. If there is an STC but not an STP, the execution of a process can be dynamically scheduled so that all constraints are met and sudden termination is avoided. We are interested in verifying in which cases it is possible to prevent an STC to become an STP, i.e., whether the process controller is able to schedule the process without sudden termination.

4.2. Sudden Start Problem

In which cases is an activity required to suddenly start? Similar to the sudden termination, a required sudden start is due to the existence of some temporal constraint, which binds a start event to some other uncontrollable (contingent) event. Thus, it is not possible to schedule the start event in advance, but it is required to react to the observation of the occurrence of a contingent event.

In an SSC, for all possible start times of C , there exist certain start times for S and some advance notice w at least equal to $k(S)$, such that some (uncontrollable) durations of C exist, which lead to the violation of some temporal constraint in the process. Instead in an SSP, for all possible start times of S and for all possible advance notice w at least equal to $k(S)$, for all start times for C , some (uncontrollable) durations of C exist, which lead to the violation of some temporal constraint in the process.

Definition 5 (Sudden Start Constellation and Pattern). Let P be a dynamically controllable process. Let S and C be activities in P , with C having duration constraint $d(C, C.d_{min}, C.d_{max}, c)$. Let Φ be the set of constraints in P .

S and C are in a **sudden start constellation (SSC)** iff
 $\forall C.s \exists S.s \exists w \in \mathbb{N} : S.s - w \geq k(S) \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg\Phi$

S and C are in a **sudden start pattern (SSP)** iff
 $\forall S.s, w \in \mathbb{N} : S.s - w \geq k(S), \forall C.s \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg\Phi$

We use the notation $SSC_{S,C}$ to indicate that S and C are in a sudden start constellation, and $SSP_{S,C}$ for a sudden start pattern.

Similar to the STP, if there is an SSP in a process, sudden start cannot be avoided, without changing the process. If there is an SSC but not an SSP, the execution of a process can be dynamically scheduled in a way to both observe all constraints and avoid sudden start.

4.3. Semi-Dynamic Controllability

The constellation of constraints leading to an STP or to an SSP does not violate dynamic controllability of processes. Indeed, we have shown that there exist dynamically controllable processes exhibiting an STP or in an SSP. Therefore, the notion of dynamic controllability turns out to be not sufficiently adequate: it allows semi-contingent activities to start without knowing yet when they must end, or it allows activities to start without prior notice. In [11], we defined semi-dynamic controllability as a specialization of dynamic controllability that recognizes the need to know the required end time for a semi-contingent activity at its start time. Here, we extend that definition, including the requirement that no activity may be forced to suddenly start, if it demands prior notice of start.

Definition 6 (Semi-Dynamic Controllability). Let P be a process. P is semi-dynamically controllable (sdc) iff P is dynamically controllable (dc), and $\nexists S, C \in P.N : STP_{S,C}$ or $SSP_{S,C}$.

The notion of semi-dynamic controllability introduced here is stricter than the notion of dynamic controllability, since it requires (1) that a process is dynamically controllable, (2) that no semi-contingent activity is involved in a sudden termination, (3) that no activity is involved in a sudden start pattern.

5. Sudden Termination Constellations and Patterns

Here, we show patterns of mutually constrained semi-contingent and contingent activities, which may lead to the sudden termination of the semi-contingent activity. These patterns cover *all* the meaningful configurations of temporal constraints between pairs of events. For each of these patterns, we show additional necessary and sufficient conditions that need to hold, to guarantee that sudden termination cannot occur. With these results, we will develop a procedure to check semi-dynamic controllability of process definitions.

5.1. Sudden Termination Constellation and Pattern STP-1

We start by presenting the most fundamental constraint constellation for an STP. In this constellation, the end events of a contingent activity C and a semi-contingent activity S are connected by one upper- and one lower-bound constraint: $ubc(C.e, S.e, w)$, and $lbc(C.e, S.e, v)$.

The only uncertainty in the constellation is the non-controllable actual duration of C . The constraint $ubc(C.e, S.e, w)$ requires that $S.e \leq C.e + w$. For actual durations $S.d$, resp. $C.d$ of activities S , resp. C , this requires that $S.s + S.d \leq C.s + C.d + w$. The constraint $lbc(C.e, S.e, v)$ requires that $C.e + v \leq S.e$.

The *Sudden Termination Pattern 1 (STP-1)* (see Figure 4) is defined as follows: a contingent activity C and a semi-contingent activity S , with constraints $ubc(C.e, S.e, w)$ and $lbc(C.e, S.e, v)$ as above are in an STP-1, if there is no way to schedule the start of S and the start of C , such that the value for the end of S can be fixed, when S starts.

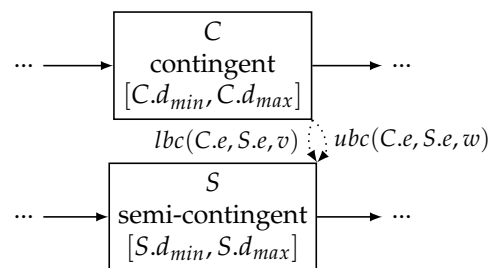


Figure 4. Sudden Termination Pattern 1 (STP-1).

Definition 7 (Sudden Termination Pattern: STP-1). Let P be a dynamically controllable process. Let S and C be activities in P , with the duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.e, S.e, v)$ be constraints in P .

S and C are in STP-1, iff $\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : S.s + S.d > C.s + C.d + w$, or $S.s + S.d < C.s + C.d + v$.

We now present the sufficient and necessary conditions for checking whether an STP might occur for a given pair of activities S and C .

We observe that if the constraints $S.s > C.e$ and $C.s > S.e$ do not hold in P , then the general precondition for the existence of a sudden termination problem is satisfied.

Theorem 1. Let P be a dynamically controllable process. Let S be a semi-contingent activity, and C be a contingent activity in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.e, S.e, v)$ be constraints in P . A sudden termination of S cannot occur, iff $C.d_{max} + v \leq S.e - C.s \leq C.d_{min} + w$ holds in P .

Proof. We show that $C.d_{max} + v < S.e - C.s \leq C.d_{min} + w$ is a necessary and sufficient condition that the activities S and C are not in an STP.

Necessary condition: we show that if the condition does not hold, a sudden termination might occur. If the condition does not hold then $\nexists S.s, S.d, C.s$ with $C.d_{max} + v < S.e - C.s \leq C.d_{min} + w$. This is only possible if $C.d_{max} + v > C.d_{min} + w$.

We now assume that C and S are in an STP. This means that $\forall C.s, S.s$ there is no $S.d$ such that $\forall C.d$ the constraints hold: $S.s + S.d \leq C.s + C.d + w$ and $S.s + S.d \geq C.s + C.d + v$.

Hence $\nexists S.d$ such that $\forall C.d: C.d + v \leq S.s + S.d - C.s \leq C.d + w$. Which requires in particular, that $\nexists S.d$ to satisfy $C.d_{max} + v \leq S.e - C.s \leq C.d_{min} + w$.

Sufficient condition: We show that if the inequality holds, sudden termination does not occur. We show that $\exists C.s, S.s, S.d$ such that $\forall C.d_{min} \leq C.d \leq C.d_{max}$ the constraints are satisfied, i.e., $C.s + C.d + v \leq S.s + S.d \leq C.s + C.d + w$, which holds since $\forall C.d_{min} \leq C.d \leq C.d_{max}: C.d + v \leq C.d_{max} + v \leq S.s + S.d - C.s \leq C.d_{min} + w \leq C.d + w$. \square

With Theorem 1 we can derive conditions for a process model to be dynamically controllable and such that an STP-1 cannot occur in it. In particular, the conditions require the process model to include a particular pair of a lower- and an upper-bound constraint between the start of S and the start of C .

Theorem 2. Let P be a process. Let S be a semi-contingent activity, and C be a contingent activity in P with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ be a constraint in P .

A sudden termination of S cannot occur, iff constraints $lbc(C.s, S.e, C.d_{max} + v)$ and $ubc(C.s, S.e, C.d_{min} + w)$ hold in P .

Proof. The constraints express exactly the condition in Theorem 1. Hence in a process model P including these constraints S and C cannot be in an STP-1. \square

5.2. Sudden Termination Constellation and Pattern STP-2

We define STP constellation STP-2 (see Figure 5) as follows: a contingent activity C and a semi-contingent activity S with constraints $ubc(C.e, S.e, w)$ and $lbc(C.s, S.e, v)$ as above are in an STP, if there is no way to schedule the start of S and the start of C , such that the value for the end of S can be fixed, when S starts.

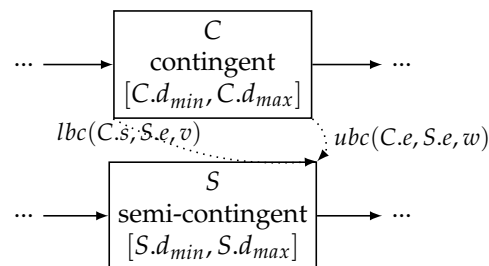


Figure 5. Sudden Termination Pattern 2 (STP-2).

Definition 8 (Sudden Termination Pattern 2). Let P be a dynamically controllable process. Let S and C be activities in P , with the duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.e, v)$ be constraints in P .

S and C are in a sudden termination pattern STP-2 iff $\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : S.s + S.d > C.s + C.d + w$, or $S.s + S.d < C.s + v$.

In the following we derive conditions to check whether an STP-2 might occur for a given pair of activities S and C .

Theorem 3. Let P be a dynamically controllable process. Let S be a semi-contingent activity, and C be a contingent activity in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.e, v)$ be constraints in P .

A sudden termination of S cannot occur, iff $v < S.e - C.s \leq C.d_{min} + w$ holds in P .

Proof. Please refer to Appendix A.1. \square

Theorem 4. Let P be a process. Let S be a semi-contingent activity, and C be a contingent activity in P with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.e, v)$ be a constraint in P .

A sudden termination of S cannot occur, iff both the constraints $lbc(C.s, S.e, v)$ and $ubc(C.s, S.e, C.d_{min} + w)$ hold in P .

Proof. The constraints express exactly the condition described in Theorem 3. Hence in a process model P including these constraints S and C cannot be in an STP-2. □

5.3. Sudden Termination Constellation and Pattern STP-3

We define STP constellation STP-3 (see Figure 6) as follows: a contingent activity C and a semi-contingent activity S with constraints $ubc(C.s, S.e, w)$ and $lbc(C.e, S.e, v)$ as above are in an STP, iff there is no way to schedule the start of S and the start of C , such that the value for the end of S can be fixed, when S starts.

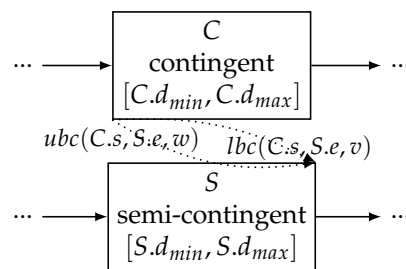


Figure 6. Sudden Termination Pattern 3 (STP-3).

Definition 9 (Sudden Termination Pattern 3). Let P be a dynamically controllable process. Let S and C be activities in P , with the duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.s, S.e, w)$ and $lbc(C.e, S.e, v)$ be constraints in P .

S and C are in a sudden termination pattern STP-3 iff $\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : S.s + S.d > C.s + w$, or $S.s + S.d < C.s + C.d + v$.

In the following we derive conditions to check whether an STP-3 might occur for a given pair of activities S and C .

Theorem 5. Let P be a dynamically controllable process. Let S be a semi-contingent activity, and C be a contingent activity in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.s, S.e, w)$ and $lbc(C.e, S.e, v)$ be constraints in P .

A sudden termination of S cannot occur, iff $C.d_{max} + v < S.e - C.s \leq w$ holds in P .

Proof. Please refer to Appendix A.2. □

Theorem 6. Let P be a process. Let S be a semi-contingent activity, and C be a contingent activity in P with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.s, S.e, w)$ and $lbc(C.e, S.e, C.d_{max} + v)$ be constraints in P .

A sudden termination of S cannot occur, if the constraints $lbc(C.s, S.e, C.d_{max} + v)$ holds in P .

Proof. The constraints express exactly the condition described in Theorem 5. Hence in a process model P including these constraints S and C cannot be in an STP-3. □

5.4. Sudden Termination Constellation and Pattern STP-4

A Sudden Termination Pattern STP-4 (see Figure 7) is defined as follows: a contingent activity C and a semi-contingent activity S with constraints $ubc(C.e, S.e, w)$ and $lbc(C.e, S.e, v)$ as above are in an STP, if there is no way to schedule the start of S and the start of C , such that the value for the end of S can be fixed, when S starts. This means that for all admissible

start times of S and C and all possible durations of S there is a possible duration of C such that one of the constraints is violated.

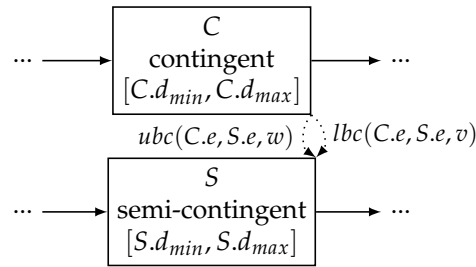


Figure 7. Sudden Termination Pattern 4 (STP-4).

Definition 10 (Sudden Termination Pattern 4). Let P be a dynamically controllable process. Let S and C be activities in P , with the duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.s, v)$ be constraints in P .

S and C are in a sudden termination pattern STP-4 iff $\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : S.s + S.d > C.s + C.d + w$, or $S.s < C.s + v$.

In the following we derive conditions to check whether an STP-4 might occur for a given pair of activities S and C .

Theorem 7. Let P be a dynamically controllable process. Let S be a semi-contingent activity, and C be a contingent activity in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.s, v)$ be constraints in P .

A sudden termination of S cannot occur, iff $S.d_{min} + v < S.e - C.s \leq C.d_{min} + w$ holds in P .

Proof. Please refer to Appendix A.3. □

Theorem 8. Let P be a process. Let S be a semi-contingent, and C be a contingent activity in P with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.s, v)$ be constraints in P .

A sudden termination of S cannot occur, iff the constraints $lbc(C.s, S.e, S.d_{min} + v)$ and $ubc(C.s, S.e, C.d_{min} + w)$ hold in P .

Proof. The constraints express exactly the condition described in Theorem 7. Hence in a process model P including these constraints S and C cannot be in an STP-4. □

5.5. Sudden Termination Constellation and Pattern STP-5

A Sudden Termination Pattern STP-5 (see Figure 8) is defined as follows: A contingent activity C and a semi-contingent activity S with constraints $ubc(C.e, S.e, w)$ and $ubc(S.s, C.e, v)$ as above are in an STP, iff there is no way to schedule the start of S and the start of C , such that the value for the end of S can be fixed, when S starts. This means that for all admissible start times of S and C and all possible durations of S there is a possible duration of C such that one of the constraints is violated.

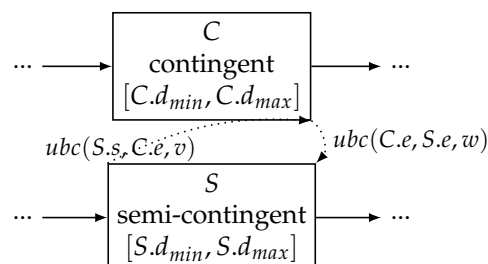


Figure 8. Sudden Termination Pattern 5 (STP-5).

Definition 11 (Sudden Termination Pattern 5). Let P be a dynamically controllable process. Let S and C be activities in P , with the duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $ubc(S.s, C.e, v)$ be constraints in P .

S and C are in a sudden termination pattern STP-5 iff $\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : S.s + S.d > C.s + C.d + w$, or $C.s + C.d > S.s + v$

In the following we derive conditions to check whether an STP-5 might occur for a given pair of activities S and C .

Theorem 9. Let P be a dynamically controllable process. Let S be a semi-contingent activity, and C be a contingent activity in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $ubc(S.s, C.e, v)$ be constraints in P .

A sudden termination of S cannot occur, iff $C.d_{max} + S.d_{min} - v < S.e - C.s \leq C.d_{min} + w$ holds in P .

Proof. Please refer to Appendix A.4. □

Theorem 10. Let P be a process. Let S be a semi-contingent, and C be a contingent activity in P with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.s, v)$ be constraints in P .

A sudden termination of S cannot occur, iff the constraints $lbc(C.s, S.e, C.d_{max} + S.d_{min} - v)$ and $ubc(C.s, S.e, C.d_{min} + w)$ hold in P .

Proof. The constraints express the condition described in Theorem 9. Thus in a process model including these constraints S and C cannot be in an STP-5. □

5.6. Sudden Termination Constellation and Pattern STP-6

A Sudden Termination Pattern STP-6 (see Figure 9) is defined as follows: A contingent activity C and a semi-contingent activity S with constraints $ubc(C.e, S.e, w)$ and $ubc(S.s, C.s, v)$ as above are in an STP, if there is no way to schedule the start of S and the start of C , such that the value for the end of S can be fixed, when S starts. This means that for all admissible start times of S and C and all possible durations of S there is a possible duration of C such that one of the constraints is violated.

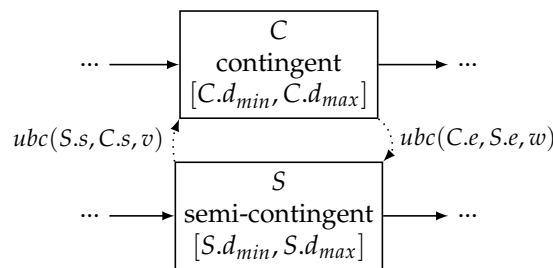


Figure 9. Sudden Termination Pattern 6 (STP-6).

Definition 12 (Sudden Termination Pattern 6). Let P be a dynamically controllable process. Let S and C be activities in P , with the duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $ubc(S.s, C.s, v)$ be constraints in P .

S and C are in a sudden termination pattern STP-6 iff $\forall S.s, C.s \forall S.d_{min} \leq S.d \leq S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : S.s + S.d > C.s + C.d + w$, or $C.s > S.s + v$.

In the following we derive conditions to check whether an STP-6 might occur for a given pair of activities S and C .

Theorem 11. Let P be a dynamically controllable process. Let S be a semi-contingent activity, and C be a contingent activity in P , with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $ubc(S.s, C.e, v)$ be constraints in P .

A sudden termination of S cannot occur, iff $S.d_{min} - v < S.e - C.s \leq C.d_{min} + w$ holds in P .

Proof. Please refer to Appendix A.5. \square

Theorem 12. Let P be a process. Let S be a semi-contingent, and C be a contingent activity in P with duration constraints $d(S, S.d_{min}, S.d_{max}, sc)$ and $d(C, C.d_{min}, C.d_{max}, c)$. Let $ubc(C.e, S.e, w)$ and $lbc(C.s, S.s, v)$ be constraints in P .

A sudden termination of S cannot occur, iff the constraints $lbc(C.s, S.e, S.d_{min} - v)$ and $ubc(C.s, S.e, C.d_{min} + w)$ hold in P .

Proof. The constraints express exactly the condition described in Theorem 11. Hence in a process model P including these constraints S and C cannot be in an STP-6. \square

5.7. Recap of the STP Constellations

Table 1 summarizes the key information of the patterns presented in this section. In the first column of the table, we report all possible constellations between a contingent and a semi-contingent activity that could cause a sudden termination problem. In the second column, we report the corresponding condition whose fulfillment guarantees that sudden termination cannot occur. The third column translates in form of temporal constraints the condition of the second column. According to Theorems 2–12, these are the constraints that have to be added in a process model, to ensure that sudden termination is avoided.

In Section 7 we apply these results for checking whether a process can be scheduled without any sudden termination. The idea is that, for each STC in a process P , we include the additional constraints that guarantee the absence of a sudden termination. Then we check the resulting process for dynamic controllability. If it is dynamically controllable, then (with Theorems 2–12) it is also semi-dynamically controllable.

Table 1. Sudden termination constellations, with conditions to avoid sudden termination, and constraints to add to check for semi-dynamic controllability.

Constraints	Condition to Avoid Sudden Termination	Constraints to Add
$ubc(C.e, S.e, w)$ $lbc(C.e, S.e, v)$	$C.d_{max} + v \leq S.e - C.s \leq C.d_{min} + w$	$lbc(C.s, S.e, C.d_{max} + v)$ $ubc(C.s, S.e, C.d_{min} + w)$
$ubc(C.e, S.e, w)$ $lbc(C.s, S.e, v)$	$v < S.e - C.s \leq C.d_{min} + w$	$lbc(C.s, S.e, v)$ $ubc(C.s, S.e, C.d_{min} + w)$
$ubc(C.s, S.e, w)$ $lbc(C.e, S.e, v)$	$C.d_{max} + v < S.e - C.s \leq w$	$lbc(C.s, S.e, C.d_{max} + v)$
$ubc(C.e, S.e, w)$ $lbc(C.s, S.s, v)$	$S.d_{min} + v < S.e - C.s \leq C.d_{min} + w$	$lbc(C.s, S.e, S.d_{min} + v)$ $ubc(C.s, S.e, C.d_{min} + w)$
$ubc(C.e, S.e, w)$ $ubc(S.s, C.e, v)$	$C.d_{max} + S.d_{min} - v < S.e - C.s \leq C.d_{min} + w$	$lbc(C.s, S.e, C.d_{max} + S.d_{min} - v)$ $ubc(C.s, S.e, C.d_{min} + w)$
$ubc(C.e, S.e, w)$ $ubc(S.s, C.s, v)$	$S.d_{min} - v < S.e - C.s \leq C.d_{min} + w$	$lbc(C.s, S.e, S.d_{min} - v)$ $ubc(C.s, S.e, C.d_{min} + w)$

6. Transforming Sudden Start into Sudden Termination

We now show that we can treat the problem of the sudden start as a sudden termination problem. Hence, the same techniques for verifying whether a process can be scheduled without sudden termination can also be used to verify whether a process can be scheduled without sudden start, having a k -advance notice for the start events. The approach is based on transforming a process definition with tasks with k -warning into a process definition with explicit waiting tasks, which bears the same temporal semantics.

Definition 13 (Process Model Transformation). Let $P(N, E, C, k, \Omega)$ be a process as in Definition 1. Let N_w be a set of new (waiting) nodes: $\forall n \in N, k(n) > 0 \exists \text{wait}_n \in N_w, \text{wait}_n.\text{type} = \text{activity}$. Let $xbc(a, b, \delta) = \text{ubc}(a, b, \delta)$ or $\text{lbc}(a, b, \delta)$. Then $\Psi(P)$ (P extended with waiting tasks) is a new process $P'(N', E, C', k', \Omega)$ where:

- $N' = N \cup N_w$;
- $C' = C \cup C_w \cup C_e \setminus C_s$, where:
 - $C_w = \{d(\text{wait}_n, k(n), h, sc) | \text{wait}_n \in N_w, h \in \mathbb{N}, h > \Omega\} \cup \{\text{lbc}(\text{start}.s, \text{wait}_n.s, 0) | \text{wait}_n \in N_w\} \cup \{\text{ubc}(\text{wait}_n.e, n.s, 0) | \text{wait}_n \in N_w\} \cup \{\text{lbc}(\text{wait}_n.e, n.s, 0) | \text{wait}_n \in N_w\}$;
 - $C_s = \{xbc(t, n.s, \delta) | k(n) > 0\} \cup \{xbc(n.s, t, \delta) | k(n) > 0\}$;
 - $C_e = \{xbc(t, \text{wait}_n.e, \delta) | xbc(t, n.s, \delta) \in C_s\} \cup \{xbc(\text{wait}_n.e, t, \delta) | xbc(n.s, t, \delta) \in C_s\}$;
- $k'(n) = 0 \forall n \in N'$.

Essentially, the transformed process model is the original process model augmented with a waiting task wait_n for each activity n whose start requires a k -advance notice. The minimum duration of this waiting task is equal to $k(n)$, and the maximum duration is a large enough horizon value. In addition, all temporal constraints binding the start event of n are remapped to refer to the end event of wait_n , and the start event of n is required to take the same timestamp of the end event of wait_n .

We show that the transformation is sound and complete, i.e., the transformed process model is such that any valid scenario for it is also a valid scenario for the original process model, and vice versa. This follows from the observation that the end time of a waiting task wait_n coincides with the start time of the associated activity n : $\overline{\text{wait}_n.e} = \overline{n.s}$.

Lemma 1 (Soundness). Let $P(N, E, C, k, \Omega)$ be a process model; let $\Psi(P) = P'(N', E, C', k', \Omega)$ be a process model obtained by applying the transformation of Definition 13 to P . Then all valid scenarios σ' for P' there is a valid scenario σ for P .

Proof. Let S be any activity requiring advance notice $k(S)$ in P . Let wait_S be the waiting task associated to S in P' . Let Γ be the set of constraints on $S.s$ in P ; let Γ' be the set of corresponding constraints on $\text{wait}_S.e$ in P' . Let σ' be a valid scenario for P' . Let σ be the projection of σ' over the elements of P .

From $\text{ubc}(\text{wait}_S.e, S.s, 0), \text{lbc}(\text{wait}_S.e, S.s, 0) \in C'$, and σ' valid, then it follows that $\overline{\text{wait}_S.e} = \overline{S.s}$.

Since $\overline{\text{wait}_S.e} = \overline{S.s}$, and σ' is valid, it follows that:

- $\forall \text{ubc}(a, \text{wait}_S.e, \delta): \overline{S.s} \leq \overline{a} + \delta$;
- $\forall \text{ubc}(\text{wait}_S.e, b, \delta): \overline{b} \leq \overline{S.s} + \delta$;
- $\forall \text{lbc}(a, \text{wait}_S.e, \delta): \overline{S.s} \geq \overline{a} + \delta$;
- $\forall \text{lbc}(\text{wait}_S.e, b, \delta): \overline{b} \geq \overline{S.s} + \delta$.

which means that Γ' satisfied by σ' in $P' \Rightarrow \Gamma$ satisfied by σ in P .

From $d(\text{wait}_S, k(S), h, sc), \text{ubc}(\text{wait}_S.e, S.s, 0), \text{lbc}(\text{wait}_S.e, S.s, 0) \in C'$, and σ' valid, it follows that $\overline{S.s} \geq \overline{\text{wait}_S} + k(S)$ in P' . So $\exists \overline{k_S} \in \sigma$ with $\overline{k_S} = \overline{\text{wait}_S} + k(S) \leq \overline{S.s}$ in P .

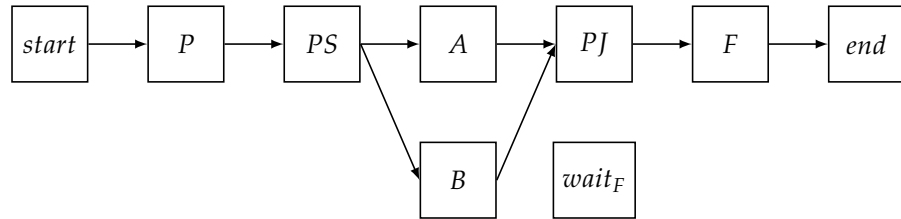
All other process elements of P are the same in P' . Thus, all other time point assignments of σ' that meet constraints in P' , correspond to time point assignments of σ that meet constraints in P . So σ is a valid scenario for P . \square

Lemma 2 (Completeness). Let $P(N, E, C, k, \Omega)$ be a process model; let $\Psi(P) = P'(N', E, C', k', \Omega)$ be a process model obtained by applying the transformation of Definition 13 to P . Then for all valid scenarios σ for P there exists a valid scenario σ' for P' .

Proof. The proof can be constructed like the proof of soundness, with the difference that the timestamp for each advance notice k_S is assigned to the start event of the waiting task $wait_S$. \square

Figure 10 shows the process model resulting from the application of the transformation Definition 13 to the process model of Figure 3.

We can now show that we can treat the sudden start of an activity in a process model P as the sudden termination of its associated waiting task in the transformed process model P' .



$$\begin{aligned}
 N^e &= \{start.s, start.e, P.s, P.e, PS.s, PS.e, A.s, A.e, B.s, B.e, PJ.s, PJ.e, wait_F.s, wait_F.e, \\
 &\quad F.s, F.e, end.s, end.e\} \\
 C &= \{d(start, 0, 0, nc), d(P, 20, 24, nc), d(PS, 0, 0, nc), d(A, 3, 5, c), d(B, 1, 2, c), \\
 &\quad d(PJ, 0, 0, nc), d(wait_F, 5, 43, sc), d(F, 2, 4, sc), d(end, 0, 0, nc), \\
 &\quad lbc(start.s, wait_F.s, 0), ubc(wait_F.e, F.s, 0), lbc(wait_F.e, F.s, 0), ubc(A.s, wait_F.e, 2), \\
 &\quad lbc(A.e, wait_F.e, 1)\} \\
 k(n) &= 0 \quad \forall n \\
 \Omega &= 42
 \end{aligned}$$

Figure 10. Process model resulting from the transformation of the process of Figure 3.

Theorem 13 (Sudden Start as Sudden Termination of a Waiting Task). *Let $P(N, E, C, k, \Omega)$ be a process model; let $P'(N', E, C', k', \Omega)$ be a process model obtained by applying the transformation of Definition 13 to P . Let S and C be activities in P , with $d(C, C.d_{min}, C.d_{max}, c)$ and $k(S) > 0$. Let $wait_S$ be the waiting task associated to S in P' . Then $SSP_{S,C}$ in $P \Rightarrow STP_{wait_S,C}$ in P' .*

Proof. The waiting task is a semi-contingent activity in an STP and corresponds to the precedence constraint of S .

From $SSP_{S,C}$ and Definition 5 it follows that $\forall S.s, w : S.s - w \geq k(S), \forall C.s \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg \Phi_P$, where Φ_P is the set of constraints in P .

From the transformation, $wait_S$ is the semi-contingent waiting task associated to S in P' . All temporal constraints on $S.s$ are bound to $wait_S.e$ in P' .

So $\forall wait_S.e, w : wait_S.e - w \geq k(S), \forall C.s \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg \Phi_{P'}$, where $\Phi_{P'}$ is the set of constraints in P' .

In particular, since $wait_S.d_{min} = k(S) \leq wait_S.d \leq wait_S.d_{max}$, we can say that $\forall wait_S.e, w : wait_S.e - w \geq wait_S.d_{min}, \forall C.s \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg \Phi_{P'}$.

In other words, $\forall wait_S.s, C.s \forall wait_S.d_{min} \leq wait_S.d \leq wait_S.d_{max} \exists C.d_{min} \leq C.d \leq C.d_{max} : \neg \Phi_{P'}$ that is $STP_{wait_S,C}$ in P' , which corresponds to a sudden termination pattern for $wait_S$ in P' ($STP_{wait_S,C}$). \square

From Theorem 13 it follows that, if the waiting task in a transformed process P' is not in a sudden termination pattern, then the associated activity is not in a sudden start pattern in the original process P :

Corollary 1 (Absence of Sudden Start). *Let P be a process model; let S, C be activities in P such that $SSP_{S,C}$. Let P' be the process model obtained by applying the transformation of Definition 13 to P , with $wait_S$ the waiting task associated to S . Then $\neg STP_{wait_S,C}$ in $P' \Rightarrow \neg SSP(S, C)$ in P .*

Proof. Follows directly from the proof of Theorem 13. \square

From Corollary 1 we conclude that for treating a process with sudden start, we can transform the process with the transformation of Definition 13, and apply sudden termination avoidance techniques to the new process model.

Given a process P , if the process P' obtained with the transformation of Definition 13 is semi-dynamically controllable, then no activity in the process is required, at run time, to suddenly start or suddenly terminate.

Thus, we only need to focus on the sudden termination problems, to automatically solve also any sudden start problem.

7. Checking Semi-Dynamic Controllability with STNUs

Here, we present an approach based on mapping process models into temporally equivalent temporal constraint networks to operate on a process model, resulting in:

1. deriving all implicit constraints in the process model. The temporal constraints causing an STC or STP need not necessarily to be explicitly stated in the process model, but may be implicitly induced by the explicit temporal constraints in the process model. Therefore, we need to compute the set of all (implicit) constraints for identifying all possible STCs.
2. checking semi-dynamic controllability of the process model. Checking whether a process model is semi-dynamically controllable requires applying some dynamic controllability checking procedure, as per Theorems 2–12.

We start by showing how to map a process model into an STNU.

7.1. Mapping to STNUs

Several previous works, e.g., [17], have shown how to check whether a process model such as the process in Figure 1 is dynamically controllable by mapping it into a temporally equivalent STNU (Simple Temporal Network with Uncertainty) [10]. The idea is that a process model is dynamically controllable if and only if its equivalent STNU is dynamically controllable.

To be self-contained, we repeat here the most important aspects of STNUs, and refer the reader to [10] for details. Essentially, an STNU is a directed graph, with nodes representing time points and edges representing binary constraints between time points. A special time point Z , or *zero*, marks the reference in time, after which all other time points occur. Edges in an STNU can be non-contingent or contingent. Non-contingent edges represent constraints that must be enforced by the execution environment by assigning appropriate values to the time points. Contingent edges (also called links), instead, represent constraints that are guaranteed to hold, but the corresponding time point assignments can only be observed, not controlled by the execution environment.

In line with prior work, we consider the usual notation (A, B, δ) for non-contingent edges from A to B with weight δ , which require that $B \leq A + \delta$; and (A^C, l, u, C) for contingent links between A^C and C , which state that C occurs some time between l and u after A^C .

We also briefly recap the rules for mapping a process model P into a temporally equivalent STNU S here:

- Each $n.i \in P.N^e$ is mapped into a node in S ;
- Each $(n1, n2) \in P.E$ is mapped into a non-contingent edge $(n2.s, n1.e, 0)$ in S ; $(start.s, zero, 0)$ is added to S ;
- Each duration constraint $d(n, n_{min}, n_{max}, [sc|nc]) \in P.C$ is mapped into a pair of non-contingent edges $(n.s, n.e, n_{max})$ and $(n.e, n.s, -n_{min})$ in S ;
- Each duration constraint $d(n, n_{min}, n_{max}, c) \in P.C$ is mapped into a contingent link $(n.s, n_{min}, n_{max}, n.e)$ in S ;
- Each constraint $ubc(a, b, \delta) \in P.C$ is mapped into a non-contingent edge (a, b, δ) in S ;
- Each constraint $lbc(a, b, \delta) \in P.C$ is mapped into a non-contingent edge $(b, a, -\delta)$ in S ;
- Process deadline $P.\Omega$ is mapped into a non-contingent edge $(start.s, end.e, \Omega)$.

Figure 11 shows the STNU derived by applying the above mapping rules to the process model of Figure 1. In the figure, we adopted the usual graphical notation for STNUs, with contingent edges dashed and inverted with respect to non-contingent edges, and labeled with the name of the contingent time point. For a more compact presentation and without loss of temporal information, in the figure we did not include nodes resulting from the mapping of the par-split and par-join.

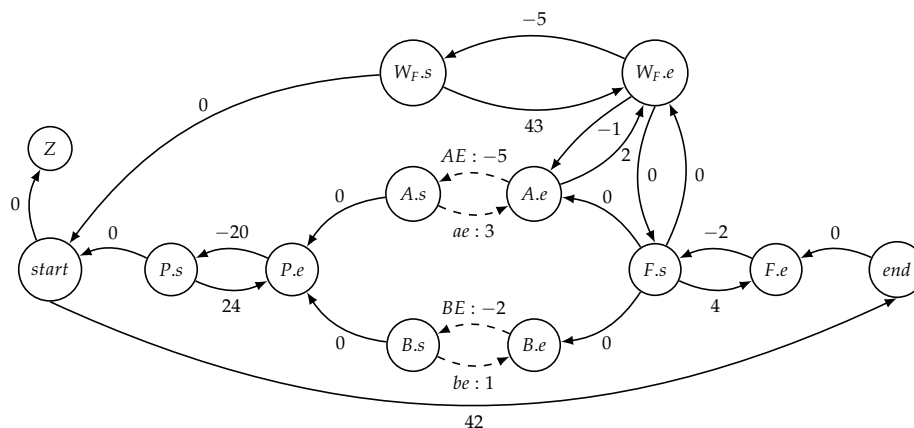


Figure 11. STNU derived from the example process shown in Figure 2, transformed as in Figure 10.

7.2. Checking Dynamic Controllability

Researchers have developed several techniques for checking the dynamic controllability of STNUs. The most efficient techniques, e.g., the one presented in [9], are based on deriving implicit constraints from the existing explicit constraints according to some constraint propagation rules.

Constraint propagation applies the propagation rules until either no new implicit edges can be derived (full propagation), or a negative cycle, in the usual sense of graph theory, is derived. In the first case, the STNU, hence its originating process, is dynamically controllable. Otherwise, a constellation of contradicting constraints generating a negative cycle exists, and the STNU is not dynamically controllable. Applying constraint propagation, it is possible to see, for instance, that no negative cycle can be derived in the STNU in Figure 11, thus it is dynamically controllable.

So applying a constraint propagation procedure to an STNU on the one hand derives all implicit constraints that hold if the given explicit constraints hold, and on the other hand determines whether the STNU is dynamically controllable. With this observation, and applying the results of Section 5, we show that it is possible to design a procedure for checking semi-dynamic controllability of process models.

7.3. Checking Semi-Dynamic Controllability

In Algorithm 1 we describe a possible procedure for checking semi-dynamic controllability. First, a process model P is mapped into an STNU T by applying the mapping rules. A data structure S_T is created for listing the STNU nodes representing semi-contingent activities, and for identifying sudden termination patterns.

The STNU T is checked for dynamic controllability by applying any constraint propagation procedure $check_dc(T)$. As discussed above, the procedure has the side effect of computing the closure of the set of explicit and implicit constraints. If the procedure returns True, the STNU, hence the process, is dynamically controllable (dc); otherwise, the algorithm terminates. If T is dc , procedure $find_stp(T, S_T)$ searches and returns all STPs in the STNU.

Algorithm 1 Check semi-dynamic controllability

```

1: Input: Process  $P$ 
2:  $T := \text{map\_to\_STNU}(P)$ 
3:  $S_T := \text{get\_semicontingent\_nodes}(P, T)$ 
4: if ( $\neg \text{check\_dc}(T)$ ) then
5:   return False
6: else
7:    $STP := \text{find\_stp}(T, S_T)$ 
8:   while ( $STP \neq \emptyset$ ) do
9:      $p := \text{extract\_first}(STP)$ 
10:     $T := T \cup \text{compute\_resolving\_constraints}(p)$ 
11:    if ( $\neg \text{check\_dc}(T)$ ) then
12:      return False
13:    else
14:       $STP := STP \cup \text{find\_stp}(T, S_T)$ 
15:    end if
16:  end while
17: end if
18: return True

```

Then, the following three steps are repeated as long as the network is *dc* and there are unresolved STPs:

1. For each STP p found, edges corresponding to the constraints to resolve p to avoid the STP are added to T .
2. Check for dynamic controllability is performed, deriving additional implicit constraints that may have been introduced with the new edges.
3. A new search for unresolved STPs is performed.

If, at the end of the repeated execution of the three steps, T remains *dc*, then it is also *sdc* and the algorithm returns *True*.

Applying the algorithm, one can verify that the process of the running example is not *sdc*, due to the negative cycle derived in Figure 12 between $W_{F.s}$ and $A.e$, formed by the edges with weights -6 and 5 .

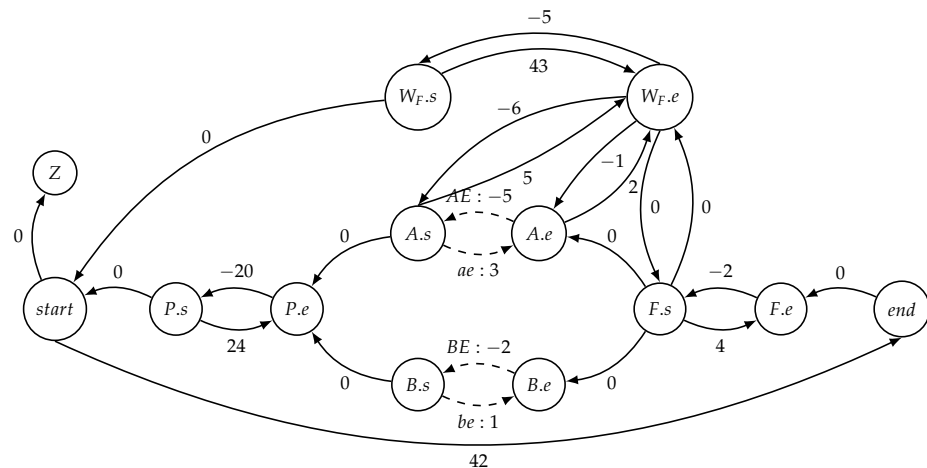


Figure 12. STNU resulting from the application of Algorithm 1 to the process of Figure 2.

Theorem 14. Algorithm 1 is a sound and complete procedure to check the semi-dynamic controllability.

Proof. In [9] it has been shown that the procedure we apply for *dc* check is sound and complete. In Section 5 we enumerated all constellations between a contingent and a semi-contingent activity that could lead to a sudden termination. For each of the constellations we derived additional constraints and we showed that these constraints are necessary and

sufficient to be able to avoid sudden termination, if the network with these additional constraints is dc. The procedure $find_stp(T, S_T)$ delivers all constellations described in Section 5. The procedure is monotonic as it only adds constraints but does not remove a constraint. \square

8. Implementation and Evaluation

8.1. Materials and Methods

To evaluate our proposed approach, we have implemented Algorithm 1 in Java, and measured the times for its execution on a number of randomly generated process models. We ran our experiments on a Windows 10 physical machine with an i7 CPU and 16 GB of RAM.

As test data, we randomly generated a set 50 processes of different sizes in terms of number of process steps and temporal constraints. We structured the test set into five subsets of processes having similar size: 10N for processes of size (around) 10 activities, 20N for 20, 30N for 30, 40N for 40 and 50N for 50. Each subset contains 10 processes; all the processes in a same subset have the same number of temporal constraints. Since Algorithm 1 maps processes into STNUs, and STNUs do distinguish between conditional (xor) and unconditional (and) splits, for the processes in the test set we consider the overall number of split nodes with no distinction between xor- and and-splits. The smallest process resulted in an STNU with 25 nodes; the largest process resulted in an STNU with 129 nodes. We report an overview on the test set in Table 2. We regard the range of process sizes used for the experiments as representative of most of the cases found in practical applications.

Table 2. Overview of the sets of processes used for the experimental evaluation. Each set contains 10 randomly generated processes.

Subset	Min STNU Nodes	Max STNU Nodes	UBCs/LBCs	Min Splits	Max Splits
10N	25	29	2	1	3
20N	45	51	4	1	4
30N	67	79	6	2	7
40N	91	107	8	4	12
50N	113	129	10	5	13

8.2. Results and Discussion

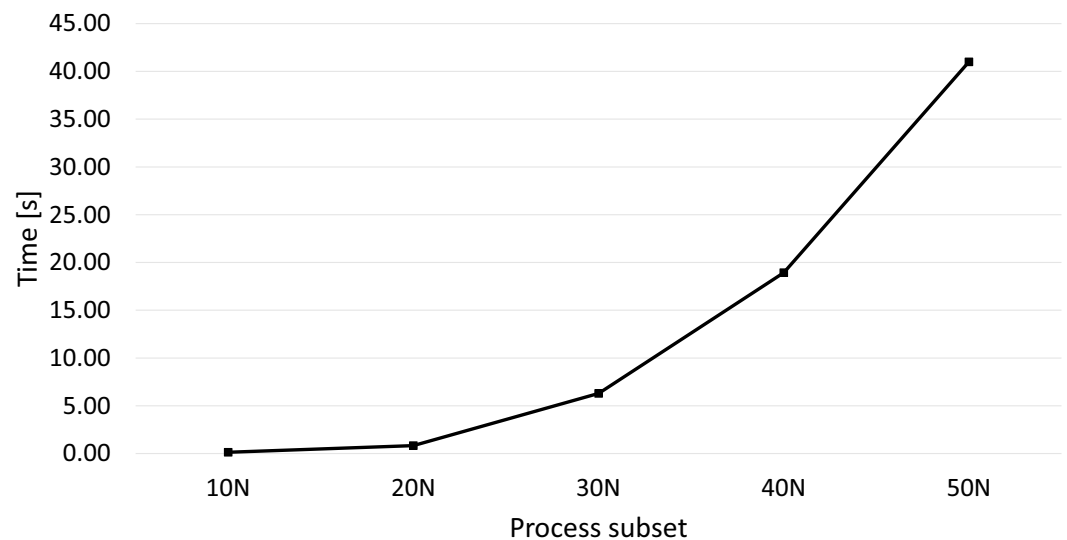
We report the minimum, maximum and average measured execution times for the various process sizes in Table 3, and the average only in Figure 13. On average, executing Algorithm 1 on a process of size (around) 10 in 10N required 0.13 s; in 20N, 0.83 s; in 30N, 6.29 s; in 40N, 18.94 s; in 50N, 41.00 s.

The nonlinear increase in the execution times is explained by the repeated execution of the dynamic controllability checking procedure, called at line 11 of Algorithm 1 for each STP in a process. It is known that checking dynamic controllability for STNUs is $O(N^4)$, where N is the number of STNU nodes [18]. Nevertheless, the proposed procedure is executed only during the design phase of a process, and has no impact on the run time performance of a process execution.

The results gathered from the experimental evaluation indicate that the required computation times are acceptable for a design time execution. Interestingly, these results are such despite the fact that Algorithm 1 introduces into the STNU new constraints to solve the STPs, and repeatedly executes a dynamic controllability checking procedure. We conclude that the proposed approach is applicable for most practical applications that require checking at design time for semi-dynamic controllability of process models.

Table 3. Overview of the measured minimum, maximum and average times for executing Algorithm 1 on the processes of the test set.

Subset	Min Time [ms]	Max Time [ms]	Average Time [ms]
10N	78	250	134
20N	484	1438	829
30N	4177	9340	6291
40N	11,865	26,558	18,936
50N	29,971	50,910	40,995

**Figure 13.** Results of the evaluation.

9. Related Work

Semi-contingent activities and semi-dynamic controllability have been introduced in our previous work [11], which, to the best of our knowledge, was the first to address them. Here, we extended the work in [11] with an extensive formalization of the sudden termination patterns, a formalization of the sudden start problem, and a mapping of sudden start into sudden termination. The most relevant related work, therefore, is in the areas of (a) formulation of temporal constraints in process models, (b) checking consistency and controllability of time-aware process models, and (c) scheduling and monitoring of process execution.

For general overviews of process time management, we refer to [5,12,19]. The work in [20] marks a milestone in the representation of temporal aspects in process models, as it presents a formalization of time patterns and their semantics, based on the analysis of recurring temporal constraints in a wide number of real world process models. Nevertheless, despite being a thorough investigation, it did not identify semi-contingent activities and the problems of sudden start and sudden termination. With this work, we established a number of (anti-)patterns of temporal constraints, which may lead to potentially undesirable behaviors during the execution of a process, and which designers should look for in their process models to prevent such behaviors.

The focus of this work is on business processes with temporal constraints. Several previous works focused on the verification of deadlines and temporal constraints in business processes, see, e.g., [21–24]. Similarly to our proposed approach based on STNUs, these approaches are based on network analysis, scheduling, and constraint networks. None of these works, however, addressed the problem of induced sudden termination of tasks. We integrate these verification approaches with the identification of a run-time behavior that can be performed at design time, without the need to develop new formalisms for temporal reasoning.

More recent works such as [25,26] combined the temporal aspects of a process definition with other constraining dimensions, e.g., resource availability. Our work, instead, currently focuses on the temporal dimension only. Looking into future research directions, we regard these recent works as interesting starting points for a holistic investigation of the problems of sudden termination and sudden start in processes.

Here, we concentrate on verifying process qualities at design time. We recognize that design time is only one stage of the process lifecycle and prior work was indeed conducted around the verification of temporal qualities at run time, e.g., pro-active monitoring of the compliance of process instances to their process model [27–29]. Nevertheless, to the best of our knowledge, all such approaches to monitoring and compliance checking address the notion of satisfiability rather than dynamic controllability and do not consider the problem of sudden termination. We consider it an interesting research opportunity to extend our current work also to the run time of processes, e.g., exploring the possibility of pro-actively giving warnings about the need to start or end tasks, even if they are not semi-contingent or necessarily require advance notice.

A popular technology for gaining insights about process definitions is process mining [30]. Process mining relies on the existence of a large number of cases (process logs) for discovering process definitions and being able to provide scheduling and monitoring information about them. Thus, it is not adequate for new or frequently changing processes, or processes with small number of instances. The approach we follow here, instead, relies on the existence and full knowledge of a process definition instead of logs in order to check for STPs and SSPs, and to try to avoid them. So our approach can be adopted by designers without the need to execute their processes and take the risk of incurring into run time problems.

The approach proposed here is based on mapping process definitions into Simple Temporal Networks with Uncertainty (STNUs) [10]. Many research contributions have been proposed in the last decades to the development of temporal constraint networks, in particular introducing different notions of controllability and introducing more expressive network models [7,10,14,31]. Here, while we do not develop any new type of temporal constraint network, we contribute to the field by defining semi-dynamic controllability as a specialization of dynamic controllability, which overcomes a class of undesirable behaviors introduced by dynamic controllability.

10. Conclusions

Temporal constraints are essential aspects in many applications of information technologies and have to be considered both in the elicitation of requirements and in the design of information systems. Research in temporal aspects of process modeling has two major aims: increasing the accuracy of representing time-related properties and temporal requirements of business processes, and developing algorithms to check relevant properties of processes at design time to avoid troubles at run time.

Here, we contribute to both aims. We found the current distinction between activities with temporally controllable and not-controllable durations too coarse grained for many practical applications, as it does not consider when the duration of an activity can be determined. Consequently, we introduced the notion of semi-contingent activities, whose duration can be controlled but not at all times. In particular, we found many application domains, where the duration of an activity might be determined by the controller (the process manager, scheduler, dispatcher, etc.) but only until the activity starts. When such an activity starts, its duration cannot be changed any more. In addition, we also found many applications where activities require an advanced notice before they may actually start, and consequently introduced *k*-warning to represent these requirements.

Ignoring these aspects at design time leads to surprises at run time, to the problems of sudden starts and sudden terminations. Modeling activities with semi-contingent durations, in contrast, paves the way for checking at design time whether sudden starts and sudden terminations can be avoided, as demonstrated with our proposed algorithm. The increased

expressiveness of temporal process model and the checking procedures require additional efforts, both for modeling temporal processes and for running the checking procedures.

The results presented here contribute to the engineering of temporal requirements. Checking the semi-dynamic controllability of temporal constraints also means to check whether requirements are in conflict. Therefore, the presented techniques support the establishment of conflict-free requirements in early phases of system development and help thus to avoid to build systems suffering from late discovery of conflicts between requirements. A current limitation of these techniques is that they only check whether a set of constraints has a conflict, but they do not identify the subsets of constraints which are in conflict and they not support designers or requirements engineers to derive a set of non-conflicting temporal constraints.

Taking a broader perspective, we regard this contribution as valuable also for process managers, who can benefit from the successful execution of business processes without surprises, which may otherwise affect the quality of process artifacts, outcomes and goals. Reasoning about the possibility of a sudden start or sudden termination our approach reduces the unpredictability of process execution and the necessity of taking precautions to handle sudden starts and sudden terminations without being overly restrictive and conservative in process planning and scheduling.

Author Contributions: Conceptualization, J.E., M.F. and J.L.; methodology, J.E. and M.F.; software, J.L.; validation, J.E., M.F. and J.L.; formal analysis, J.E., M.F. and J.L.; writing—original draft preparation, J.E., M.F. and J.L.; writing—review and editing, J.E., M.F. and J.L.; visualization, M.F. and J.L.; supervision, J.E.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/seluh/temporalvariables_prules (accessed on 23 December 2021). The code of the software used for the experiments reported in this study is publicly available. This code can be found here: https://github.com/seluh/temporalvariables_prules (accessed on 23 December 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proofs

We include here the proofs to Theorems 3–11 of Section 5.

Appendix A.1. Proof to Theorem 3

Proof. We show that $v < S.e - C.s \leq C.d_{min} + w$ is a necessary and sufficient condition that the activities S and C are not in an STP.

Necessary condition: we show that if the condition does not hold, a sudden termination might occur. If the condition does not hold then $\nexists S.s, S.d, C.s$ with $v < S.e - C.s \leq C.d_{min} + w$, which is only possible, if $v > C.d_{min} + w$.

We now assume that C and S are in an STP. This means that for all $C.s, S.s$ there is no $S.d$ such that for all $C.d$ the constraints holds: $S.s + S.d \leq C.s + C.d + w$ and $S.s + S.d > C.s + v$. Hence there is no $S.d$ such that for all $C.d$: $v < S.s + S.d - C.s < C.d + w$. which requires in particular, that there is no $S.d$ to satisfy $v < S.e - C.s \leq C.d_{min} + w$.

Sufficient condition: We show that if the inequality holds, sudden termination does not occur. We show that $\exists C.s, S.s, S.d$ such that $\forall C.d_{min} \leq C.d \leq C.d_{max}$ the constraints are satisfied, i.e., $C.s + v < S.s + S.d < C.s + C.d + w$, which holds since $\forall C.d_{min} \leq C.d \leq C.d_{max}$: $v \leq S.s + S.d - C.s \leq C.d_{min} + w \leq C.d + w$. \square

Appendix A.2. Proof to Theorem 5

Proof. We show that $C.d_{max} + v < S.e - C.s \leq w$ is a necessary and sufficient condition that the activities S and C are not in an STP.

Necessary condition: we show that if the condition does not hold, a sudden termination might occur. If the condition does not hold then $\nexists S.s, S.d, C.s$ with $C.d_{max} + v < S.e - C.s \leq w$, which is only possible, if $C.d_{max} + v > w$.

We now assume that C and S are in an STP. This means that for all C.s, S.s there is no S.d such that for all C.d the constraints holds: $S.s + S.d \leq C.s + w$ and $S.s + S.d > C.s + C.d + v$. Hence there is no S.d such that for all C.d: $C.d + v < S.s + S.d - C.s < w$. which requires in particular, that there is no S.d to satisfy $C.d_{max} + v < S.e - C.s \leq w$.

Sufficient condition: We show that if the inequality holds, sudden termination does not occur. We show that $\exists C.s, S.s, S.d$ such that $\forall C.d_{min} \leq C.d \leq C.d_{max}$ the constraints are satisfied, i.e., $C.s + C.d + v < S.s + S.d < C.s + w$, which holds since $\forall C.d_{min} \leq C.d \leq C.d_{max}$: $C.d + v \leq C.d_{max} + v \leq S.s + S.d - C.s \leq w$. \square

Appendix A.3. Proof to Theorem 7

Proof. We show that $S.d_{min} + v < S.e - C.s \leq C.d_{min} + w$ is a necessary and sufficient condition that the activities S and C are not in an STP.

Necessary condition: we show that if the condition does not hold, a sudden termination might occur. If the condition does not hold then $\nexists S.s, S.d, C.s$ with $S.d_{min} + v < S.e - C.s \leq C.d_{min} + w$, which is only possible, if $S.d_{min} + v > C.d_{min} + w$.

We now assume that C and S are in an STP. This means that for all C.s, S.s there is no S.d such that for all C.d the constraints holds: $S.s + S.d \leq C.s + C.d + w$ and $S.s + S.d > C.s + S.d + v$. Hence there is no S.d such that for all C.d: $S.d_{min} + v < S.s + S.d - C.s < C.d + w$. which requires, in particular, that there is no S.d to satisfy $S.d_{min} + v < S.e - C.s \leq C.d_{min} + w$.

Sufficient condition: We show that if the inequality holds, sudden termination does not occur. We show that $\exists C.s, S.s, S.d$ such that $\forall C.d_{min} \leq C.d \leq C.d_{max}$ the constraints are satisfied, i.e., $C.s + S.d_{min} + v < S.s + S.d < C.s + C.d + w$,

which holds since $\forall C.d_{min} \leq C.d \leq C.d_{max}$: $C.d + S.d_{min} + v \leq S.s + S.d - C.s \leq C.d_{min} + w \leq C.d + w$. \square

Appendix A.4. Proof to Theorem 9

Proof. We show that $C.d_{max} - v < S.e - C.s \leq C.d_{min} + w$ is a necessary and sufficient condition that the activities S and C are not in an STP.

Necessary condition: we show that if the condition does not hold, a sudden termination might occur. If the condition does not hold then $\nexists S.s, S.d, C.s$ with $C.d_{max} + S.d_{min} - v < S.s + S.d - C.s \leq C.d_{min} + w$, which is only possible, if $C.d_{max} + S.d_{min} - v > C.d_{min} + w$.

We now assume that C and S are in an STP. This means that for all C.s, S.s there is no S.d such that for all C.d the constraints hold: $S.s + S.d \leq C.s + C.d + w$ and $S.s + S.d > C.s + C.d + S.d - v$

Hence there is no S.d such that for all C.d: $C.d + S.d_{min} - v < S.s + S.d - C.s < C.d + w$. which requires, in particular, that there is no S.d to satisfy $C.d_{max} + S.d_{min} - v < S.s + S.d - C.s \leq C.d_{min} + w$.

Sufficient condition: We show that if the inequality holds, sudden termination does not occur. We show that $\exists C.s, S.s, S.d$ such that $\forall C.d_{min} \leq C.d \leq C.d_{max}$ the constraints are satisfied, i.e., $C.s + C.d - S.d - v < S.s + S.d \leq C.s + C.d + w$, which holds since $\forall C.d_{min} \leq C.d \leq C.d_{max}$: $C.d + S.d_{min} - v \leq C.d_{max} + S.d_{min} - v \leq S.s + S.d - C.s \leq C.d_{min} + w \leq C.d + w$. \square

Appendix A.5. Proof to Theorem 11

Proof. We show that $S.d_{min} - v < S.e - C.s \leq C.d_{min} + w$ is a necessary and sufficient condition that the activities S and C are not in an STP.

Necessary condition: we show that if the condition does not hold, a sudden termination might occur. If the condition does not hold then $\nexists S.s, S.d, C.s$ with $S.d_{min} - v < S.s + S.d - C.s \leq C.d_{min} + w$, which is only possible, if $S.d_{min} - v > C.d_{min} + w$.

We now assume that C and S are in an STP. This means that for all C.s, S.s there is no S.d such that for all C.d the constraints hold: $S.s + S.d \leq C.s + C.d + w$ and $S.s > C.s + S.d - v$

Hence there is no S.d such that for all C.d: $S.d_{min} - v < S.s + S.d - C.s < C.d + w$, which requires, in particular, that there is no S.d to satisfy $S.d_{min} - v < S.s + S.d - C.s \leq C.d_{min} + w$.

Sufficient condition: We show that if the inequality holds, sudden termination does not occur. We show that $\exists C.s, S.s, S.d$ such that $\forall C.d_{min} \leq C.d \leq C.d_{max}$ the constraints are satisfied, i.e., $C.s + S.d - v < S.s + S.d \leq C.s + C.d + w$, which holds since $\forall C.d_{min} \leq C.d \leq C.d_{max}$: $S.d_{min} - v \leq S.s + S.d - C.s \leq C.d_{min} + w \leq C.d + w$. \square

References

1. Van der Aalst, W.M. Business process management: A comprehensive survey. *Int. Sch. Res. Not.* **2013**, *2013*, 507984. [\[CrossRef\]](#)
2. Dumas, M.; Rosa, M.L.; Mendling, J.; Reijers, H.A. *Fundamentals of Business Process Management*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2018. [\[CrossRef\]](#)
3. Weske, M. *Business Process Management—Concepts, Languages, Architectures*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2019. [\[CrossRef\]](#)
4. Eder, J.; Panagos, E.; Rabinovich, M. Time constraints in workflow systems. In *Advanced Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 286–300.
5. Cheikhrouhou, S.; Kallel, S.; Guermouche, N.; Jmaiel, M. The temporal perspective in business process modeling: A survey and research challenges. *Serv. Oriented Comput. Appl.* **2015**, *9*, 75–85. [\[CrossRef\]](#)
6. Combi, C.; Posenato, R. Controllability in temporal conceptual workflow schemata. In *Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 64–79.
7. Vidal, T. Handling contingency in temporal constraint networks: From consistency to controllabilities. *J. Exp. Theor. Artif. Intell.* **1999**, *11*, 23–45. [\[CrossRef\]](#)
8. Eder, J.; Lehmann, M.; Tahamtan, A. Choreographies as federations of choreographies and orchestrations. In Proceedings of the International Conference on Conceptual Modeling, Tucson, AZ, USA, 6–9 November 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 183–192.
9. Cairo, M.; Rizzi, R. Dynamic Controllability Made Simple. In Proceedings of the 24th International Symposium on Temporal Representation and Reasoning (TIME 2017), LIPIcs, Mons, Belgium, 16–18 October 2017; Volume 90, pp. 8:1–8:16.
10. Morris, P.H.; Muscettola, N. Temporal dynamic controllability revisited. In Proceedings of the AAAI, Pittsburgh, PA, USA, 9–13 July 2005; pp. 1193–1198.
11. Eder, J.; Franceschetti, M.; Lubas, J. Scheduling Processes Without Sudden Termination. In *Enterprise, Business-Process and Information Systems Modeling*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 117–132.
12. Eder, J.; Panagos, E.; Rabinovich, M. Workflow Time Management Revisited. In *Seminal Contributions to Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 207–213.
13. Hunsberger, L.; Posenato, R. Dynamic Controllability Checking for Conditional Simple Temporal Networks with Uncertainty: New Sound-and-Complete Algorithms based on Constraint Propagation. In Proceedings of the 25th International Symposium on Temporal Representation and Reasoning (TIME 2018), Warsaw, Poland, 15–17 October 2018; Volume 120, p. 14.
14. Zavattoni, M.; Viganò, L. Conditional simple temporal networks with uncertainty and decisions. *Theor. Comput. Sci.* **2019**, *797*, 77–101. [\[CrossRef\]](#)
15. Cairo, M.; Rizzi, R. Dynamic controllability of simple temporal networks with uncertainty: Simple rules and fast real-time execution. *Theor. Comput. Sci.* **2019**, *797*, 2–16. [\[CrossRef\]](#)
16. Zur Muehlen, M.; Recker, J. How much language is enough? Theoretical and practical use of the business process modeling notation. In *Seminal Contributions to Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 429–443.
17. Eder, J.; Franceschetti, M.; Köpke, J. Controllability of business processes with temporal variables. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; ACM: New York, NY, USA, 2019; pp. 40–47.
18. Morris, P. A structural characterization of temporal dynamic controllability. In Proceedings of the International Conference on Principles and Practice of Constraint Programming, Nantes, France, 25–29 September 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 375–389.
19. Combi, C.; Pozzi, G. Temporal conceptual modelling of workflows. In *Conceptual Modeling-ER 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 59–76.
20. Lanz, A.; Reichert, M.; Weber, B. Process time patterns: A formal foundation. *Inf. Syst.* **2016**, *57*, 38–68. [\[CrossRef\]](#)
21. Bettini, C.; Wang, X.; Jajodia, S. Temporal reasoning in workflow systems. *Distrib. Parallel Databases* **2002**, *11*, 269–306. [\[CrossRef\]](#)

22. Cardoso, J.; Sheth, A.; Miller, J.; Arnold, J.; Kochut, K. Quality of service for workflows and web service processes. *J. Web Semant.* **2004**, *1*, 281–308. [[CrossRef](#)]
23. Guermouche, N.; Godart, C. Timed model checking based approach for web services analysis. In Proceedings of the ICWS 2009. IEEE International Conference on Web Services, Los Angeles, CA, USA, 6–10 July 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 213–221.
24. Marjanovic, O.; Orłowska, M.E. On modeling and verification of temporal constraints in production workflows. *Knowl. Inf. Syst.* **1999**, *1*, 157–192. [[CrossRef](#)]
25. Watahiki, K.; Ishikawa, F.; Hiraishi, K. Formal verification of business processes with temporal and resource constraints. In Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Anchorage, AK, USA, 9–12 October 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1173–1180.
26. Zavatleri, M.; Combi, C.; Viganò, L. Resource controllability of workflows under conditional uncertainty. In Proceedings of the International Conference on Business Process Management, Vienna, Austria, 1–6 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 68–80.
27. Hashmi, M.; Governatori, G.; Lam, H.P.; Wynn, M.T. Are we done with business process compliance: State of the art and challenges ahead. *Knowl. Inf. Syst.* **2018**, *57*, 79–133. [[CrossRef](#)]
28. Ly, L.T.; Maggi, F.M.; Montali, M.; Rinderle-Ma, S.; van der Aalst, W.M. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Inf. Syst.* **2015**, *54*, 209–234. [[CrossRef](#)]
29. Pichler, H.; Wenger, M.; Eder, J. Composing time-aware web service orchestrations. In Proceedings of the International Conference on Advanced Information Systems Engineering, Amsterdam, The Netherlands, 8–12 June 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 349–363.
30. van der Aalst, W.M.; Schonenberg, M.; Song, M. Time prediction based on process mining. *Inf. Syst.* **2011**, *36*, 450–475. [[CrossRef](#)]
31. Lanz, A.; Posenato, R.; Combi, C.; Reichert, M. Controlling time-awareness in modularized processes. In *Enterprise, Business-Process and Information Systems Modeling*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 157–172.