

Negotiating Temporal Commitments in Cross-Organizational Business Processes

Marco Franceschetti 

Department of Informatics Systems, Universität Klagenfurt, Austria

<https://www.aau.at/en/isys/ics/>

marco.franceschetti@aau.at

Johann Eder 

Department of Informatics Systems, Universität Klagenfurt, Austria

<https://www.aau.at/en/isys/ics/>

johann.eder@aau.at

Abstract

Cross-organizational business processes emerge from the cooperation of intra-organizational business processes through exchange of messages. The involved parties agree on communication protocols, which contain in particular temporal constraints: as obligations on one hand, and as guarantees on the other hand. These constraints form also requirements for the design of the hidden implementation of the processes and are the basis for control decisions for each party. We present a comprehensive methodology for modeling the temporal aspects of cross-organizational business processes, checking dynamic controllability of such processes, and supporting the negotiation of temporal commitments. We do so by computing the consequences of temporal constraints in choreographies, and by computing the weakest preconditions for the dynamic controllability of a participating process.

2012 ACM Subject Classification Applied computing → Cross-organizational business processes

Keywords and phrases Cross-organizational processes, Temporal parameters, Range negotiation

Digital Object Identifier 10.4230/LIPIcs.TIME.2020.4

1 Introduction

Cross-organizational business processes [13, 17, 20, 30] emerge from an ensemble of local processes communicating through process views [6]. Process views abstract from the internals of the process implementation, and describe the communication interfaces of the processes. The design of choreographies can thus be solely based on these process views. Such an architecture achieves the desired low coupling between the processes, and facilitates keeping business secrecy of process internals as far as possible. A widely adopted approach models cross-organizational processes by a protocol for exchanging messages governed by Service Level Agreements [28]. Formulating, checking, and enforcing temporal properties - characteristics, obligations, commitments, and objectives - for cross-organizational p2p-processes requires distributed algorithms for checking and/or computing temporal properties [14].

Temporal parameters [11] encode events, and can be exchanged via messages between the partners of a collaboration to communicate event timestamps, without having to expose process internals in the process views. The exchange of temporal parameters has been shown to enable an effective way for expressing cross-organizational temporal constraints, while preserving business secrets encoded in the local processes. Thus, temporal parameters foster lean interfaces caring for loose coupling of the involved processes [16].

Temporal constraints for the execution of processes can be formulated as relations between time-points of message exchanges and/or ranges for the temporal parameters. Such constraints (e.g. the time interval for the reaction to the receipt of a message) can be descriptive for one party (“when can I expect an answer from the other party”) and prescriptive for the other party (“when do I have to send an answer?”). These temporal constraints are part

of the negotiation of the communication protocols (choreography), respectively of Service Level Agreements, which are part of the specification for cross-organizational processes. For process designers it is, therefore, of paramount importance to verify at design time, whether at run time it is possible to avoid any violation of these temporal obligations in the contracts, independent of uncertainties, when other parties are sending messages within their agreed time frames. Technically, this means that process designers need to check the dynamic controllability [26] of their processes. Basically, a process is dynamically controllable, if the execution environment is able to dynamically schedule its tasks in response to all foreseeable circumstances, in such a way that all temporal constraints can be met.

These computations are, however, also a necessary part of the negotiation step for the design of cross-organizational processes. In particular, it is necessary, for the negotiators, to calculate the consequences of any temporal obligation, and to compute which uncertainties (e.g. duration intervals) in the constraints of other parties are acceptable. In a nutshell, there is a need to calculate the trade-offs between temporal constraints to support the negotiation of temporal commitments.

In [11] we have shown how to check, at design time, whether a single process with temporal parameters is dynamically controllable. In [15] we have shown how to check, whether a given set of constraints on the temporal parameters ensures dynamic controllability of a service composition. For cross-organizational processes this problem is more complex, due to the distributed design and execution, and the potentially different requirements of each partner. So now we ask: *how to achieve contracts between the parties of a cross-organizational business process, with adequate restrictions on temporal parameters, admitting schedules free from time failures?*

The main contributions of this paper are:

- A cross-organizational process model based on process views and message exchanges, featuring temporal parameters for temporal constraints.
- A procedure for checking the dynamic controllability of cross-organizational processes with temporal parameters at design time.
- An algorithm for negotiating restrictions for temporal parameters of a cross-organizational process in a fully distributed way, such that these restrictions yield dynamic controllability.

The remainder of this paper is structured as follows: in Section 2 we provide a motivating example. Sect. 3 introduces the process model for cross-organizational processes. In Sect. 4 we show a procedure for negotiating temporal parameter restrictions with dynamic controllability guarantees, and we evaluate the procedure under correctness and complexity dimensions. Sect. 5 (Related work) and Sect. 6 (Conclusions) conclude the paper.

2 Motivating Example

Consider the BPMN (Business Process Modeling Notation) choreography diagram depicted in Fig. 1, which models a simplified procurement process. Three organizations are involved: a distributor D of medical equipment, a factory F manufacturing surgical masks, and a supplier S of raw materials. The process starts at D with the request to F for a batch of masks. For simplicity, let us focus on the interactions between D and F only.

The process model includes three cross-organizational temporal constraints, which restrict the times of events in the local processes of D and F . These constraints are realized through the exchange of messages between D and F , which include temporal data specifying requirements.

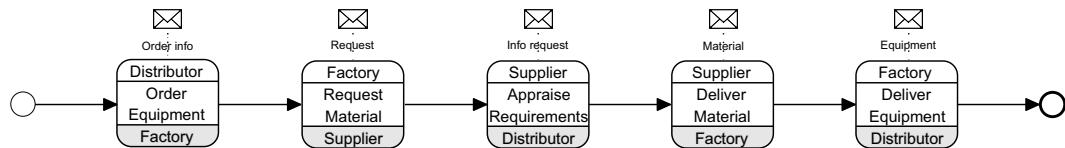


Figure 1 BPMN choreography diagram for a cross-organizational process.

A first constraint $c1$ is internal to the process of D , and it requires that the masks are produced between 6 and 10 days after the order is placed. A second constraint $c2$ in the process of F , however, requires the production to take place 9 to 11 days after the order is placed. A third constraint $c3$ requires F to complete the delivery of the masks at most 2 days after they have been produced.

Here $c1$, $c2$ and $c3$ are cross-organizational constraints, since they restrict events in processes of different organizations, to occur within specified bounds with respect to each other. These constraints can be enforced by contracts, which specify an earliest or latest date for completion of a task. However, to guarantee the fulfilment of the constraints, a negotiation needs to take place to ensure an agreement on the requirements. For example, $c1$ and $c2$ here are conflicting, since they require different time intervals for the same event, e.g., D would not accept that F produces the masks 11 days after the order is placed.

On the other hand, temporal constraints may as well influence each other. For instance, if F is convinced by D to produce the masks earlier than intended, the need to ship them within 2 days (from $c3$) may be unattended because shipment is already scheduled for a certain date.

So to make sure that the local processes can coordinate and execute meeting all constraints, D and F need to find an agreement on what they can expect from each other, as well as on what they can offer. Our aim is to provide the partners of a cross-organizational process a protocol for negotiating temporal parameter restrictions, which lead to agreed constraints with no risk of time failure.

3 Cross-Organizational Processes

Cross-organizational processes emerge from local processes interacting by exchanging messages according to some choreography. Here we do not focus on the definition of process views and the alignment of process views with choreographies or global processes, but we assume that a choreography or message exchange protocol has been defined already. We focus on the definition of temporal constraints for the process model. Temporal constraints may depend on each other and there may be trade-offs between different temporal constraints.

We consider a cross-organizational process from the viewpoint of one party: a process model in which some of the steps are used for receiving and sending messages. Temporal constraints from such a viewpoint are local constraints and choreography constraints. Choreography constraints can only be formulated using the elements (events, temporal variables) which are known by both parties, i.e. elements of the process view. Local constraints can also include local events and local temporal variables [16].

In such a cross-organizational process, the local processes need to agree on the choreography constraints such that the cross-organizational process can be executed without time failures for all foreseeable situations.

3.1 Local Process Model

We consider here a standard (local) process model, such as the model we introduced in previous work [16], to which we refer for details on the architecture and assumptions.

Basically, the model is a standard process model derived from workflow nets, consisting of steps connected by precedence relationships. It features the most commonly used control flow elements as identified in [33]: start and end events, tasks, sequence, exclusive split, parallel split, and their corresponding joins. In addition, it has 2 specialties: (i) *communication steps*: some of the steps send or receive messages, and (ii) *temporal parameters*: some of the parameters in the message payload may encode temporal information. In [16] we argued in detail about the use of temporal parameters for exchanging temporal information, and on the use of temporal parameters for formulating cross-organizational temporal constraints. According to the state-of-the-art and to avoid design flaws [8], we consider here acyclic block-structured processes, assume that variables are available when they ought to be sent, and that no deadlocks occur. Discussions about these assumptions for well-formedness are out of scope for this paper.

We formally define the local process model as follows:

► **Definition 1 (Local Process Model).** A local process model P is a tuple $(\text{proc_id}, N, E, V, C, \Omega)$, where:

- proc_id is a unique process id.
- N is a set of nodes. A node n has type $n.\text{type} \in \{\text{start}, \text{activity}, \text{xor-split}, \text{par-split}, \text{xor-join}, \text{par-join}, \text{send}, \text{receive}, \text{end}\}$, and start and end events $n.s$, resp. $n.e$. For each node n :
 - $n.I$ is the set of input variables;
 - $n.O$ is the set of output variables;
 - $n.t$ (to) is the id of the receiver for send nodes;
 - $n.f$ (from) is the id of the sender for receive nodes.
- $E \subseteq N \times N$ is a set of edges defining precedence constraints.
- V is a set of temporal variables, partitioned in disjoint sets V^I, V^O, N^e :
 - V^I is the set of input parameters of P :

$$V^I = \bigcup_{n \in N} \{v \in n.I | n.\text{type} = \text{receive}\};$$
 - V^O is the set of output parameters of P :

$$V^O = \bigcup_{n \in N} \{v \in n.O | n.\text{type} = \text{send}\};$$
 - N^e is the set of variables for start and end events of nodes: $N^e = \bigcup_{n \in N} \{n.s, n.e\}$.
- C is a set of temporal constraints consisting of
 - duration constraints for each $n \in N$: $d(n, d_{\min}, d_{\max})$, with $n.\text{type} = \text{activity}$, where $d_{\min}, d_{\max} \in \mathbb{N}$ with $d_{\min} \leq d_{\max}$; for all other nodes, $d_{\min} = d_{\max} = 0$;
 - range constraints for temporal variables $v \in V^I \cup V^O$: $r(v, v_{\min}, v_{\max})$, where $v_{\min}, v_{\max} \in \mathbb{N}$ with $v_{\min} \leq v_{\max}$;
 - upper-bound constraints: $ubc(a, b, \delta)$, where $a, b \in V, \delta \in \mathbb{N}$, requiring that $b \leq a + \delta$;
 - lower-bound constraints: $lbc(a, b, \delta)$, where $a, b \in V, \delta \in \mathbb{N}$, requiring that $b \geq a + \delta$.
- Ω is the maximum process duration.

Given the definition of the local process model, we can now illustrate in detail its temporal aspect, and define its temporal semantics.

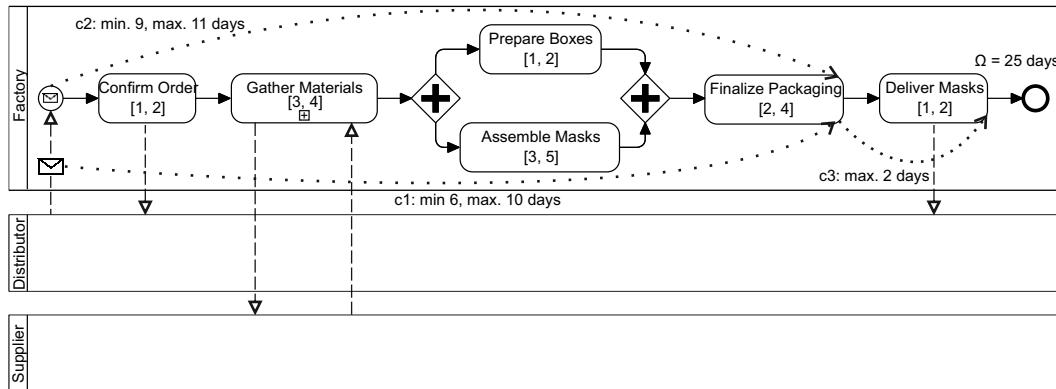


Figure 2 BPMN collaboration diagram for the process of Fig. 1 from the viewpoint of the Factory.

3.2 Temporal Aspect of the Local Process Model

The temporal aspect of the process model includes time points for events, durations of nodes, temporal parameters, and temporal constraints as relations between timepoints and/or durations. Each process node n is associated with two events, $n.s$ and $n.e$, representing the start and end of the respective node.

Duration constraints specify minimum and maximum bounds for the actual durations of activities. Without loss of generality [12] we consider all activities as contingent, i.e. their actual duration cannot be controlled, but only be observed at run time, but it can be assumed to be within the interval specified by the minimum and maximum duration.

Temporal parameters are data elements containing a timestamp value. They can be sent to some other process via a communication channel. Input parameters are received from another process by a receiving node; output parameters are sent to another process by a sending node.

Temporal constraints restrict the points in time when events may occur. By imposing temporal constraints between parameters and events, it is possible to constrain the occurrence of local events, or events at other local processes.

A special temporal constraint is defined by Ω , which requires that the overall process duration is less than the value specified by Ω .

Constraints can only be formulated between the elements in the process model. Hence constraints between events in different processes can only be formulated, if information about event is sent to the other process in form of temporal parameters.

In Fig. 2 we show the BPMN collaboration diagram for the choreography diagram from Fig. 1, enriched with an explicit representation of the temporal constraints as dotted arrows. We stress the problem of not being able to access the internals of other local processes, by showing the diagram from the viewpoint of the factory F , with the other processes modeled as black boxes. In the example, D must include the date she requires for mask production, as part of the message sent with the first choreography task (see Fig. 1). In its local process definition, F would use such a date, which is an input parameter, in a constraint binding the time for completing mask production.

3.3 Negotiation of Temporal Commitments

Parties in a cross-organizational process have to agree on temporal commitments, i.e. on constraints defined on admissible timepoints of communication events (sending and receiving messages) and on the ranges for the temporal parameters. The goal is to agree on a set

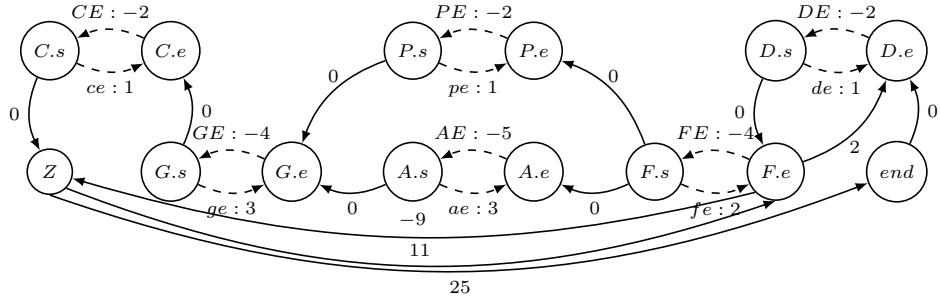


Figure 3 STNU derived from the local process of F in Fig. 1.

of constraints, such that time failures can be avoided for all possible process instances, i.e. if all local processes with temporal parameters are dynamically controllable, then the cross-organizational process is dynamically controllable.

For the receiving process, a range of an input parameter is a guarantee that the values of the parameter will be within the range. On the other hand, it is an obligation for the sending process to send a value within the agreed range. If the range of an input variable is wider, then the uncertainty for the receiving process is higher, and it is more difficult to avoid time failures. However, if the range of an output parameter is smaller, it is more difficult to send a valid parameter. Every parameter is both output for some process and input for some other process.

Moreover, the ranges of the input parameters influence the possible ranges of the output parameters, and dynamic controllability also depends on these ranges. Therefore, these ranges have to be negotiated between the different parties. Additionally, there might be inter-dependencies between parameters, leading to trade-offs between the ranges, i.e. a broader range for one parameter may reduce the ranges of other parameters.

In negotiating parameter restrictions, each party has to understand the dependencies and trade-offs between different ranges and other constraints. Since these dependencies are distributed over all the participating processes, these dependencies and their consequences can only be calculated with a distributed procedure.

For the negotiations, the following computations are necessary:

1. which ranges of the input parameters can be accepted,
2. which ranges of the output parameters can be guaranteed,
3. which constraints have to hold between parameters.

For the input parameters, we aim to compute the widest possible ranges; for the output parameters, the most narrow ranges (cf. contra-variant subtyping [5]). In addition, the computations need to determine the constraints between parameters, representing the above mentioned trade-offs.

For checking dynamic controllability, and for computing admissible restrictions for temporal parameters, we map process models into temporal constraint networks, and resort to the temporal reasoning capabilities of temporal constraint networks. In particular, we currently consider Simple Temporal Networks with Uncertainty (STNUs) as the formal apparatus, instead of more expressive networks (e.g., CSTNUs): this may lead to stricter results, but the reasoning algorithms have lower complexity. In Fig. 3 we show the STNU equivalent to the process of F from Fig. 2, with abbreviated activity names for better readability. For details on STNUs, and how to map process models into STNUs, we refer to Appendix A.

4 Constraint Negotiation

Process models and the problem of negotiating constraints are mapped into terms of temporal constraint networks. Thus, here we present a framework for the negotiation of temporal constraints, which is based on temporal networks as the formal apparatus for inferring temporal constraints. For solving the negotiation problem outlined above, we developed new inference mechanisms for STNUs. Finally, we map the solution back into process model terms.

In the following we consider only STNUs which are verified to be dynamically controllable.

4.1 Computing Constraints for Temporal Parameters

Recently, an efficient method for checking dynamic controllability of STNUs has been proposed in [3]. It applies three rules for inferring implicit constraints, and checks for contradictions in form of negative cycles in the STNU. For dynamic controllability, however, the possible values of a node might depend on all the nodes which have smaller values. Thus, the value of a parameter node might depend on the observed duration of contingent activities. This is unacceptable for parameter nodes, whose timestamps have to be fixed when they are communicated. We even require that possible ranges for these parameter nodes are defined at design time as part of the negotiations outlined above. We address this requirement by proposing three rules, which infer constraints on nodes such that they are independent of observed durations of contingent activities.

Another challenge is that there is no central STNU, but there is a set of communicating STNUs and an infrastructure for communicating constraints between these STNUs in the design phase. All possible algorithms have to take into account that local STNUs do not expose their internal structure and internal constraints, and only communicate the constraints on parameters to their peers.

We solve these problems by first introducing basic rules for inferring constraints, which make parameter nodes independent of observed contingent durations. Then we propose a procedure to compute restrictions on input and output parameters for a local STNU. Finally, we discuss a framework for the communication of constraints to come up with an agreement on parameter restrictions, which are compatible with the dynamic controllability of each participating STNU.

4.2 Basic Inference Rules

We propose three basic rules (*P-Relax*, *P-Upper*, and *P-Lower*, shown in Fig. 4) for the deduction of constraints in form of non-contingent STNU edges implicitly contained in an STNU, such that the derived constraints for parameters are independent of any contingent duration.

P-Relax: This rule is applied when three time points are connected by a sequence of two consecutive non-contingent edges with weights v and w , respectively. It introduces a new non-contingent edge with weight $v + w$ between the time point origin of the first edge and the time point destination of the second edge.

► **Lemma 2.** P-Relax derives the broadest non-contingent constraint between time point P and time point R .

Proof. Since *P-Relax* is defined as *Relax* in the RUL system, for the proof we refer to [3]. ◀

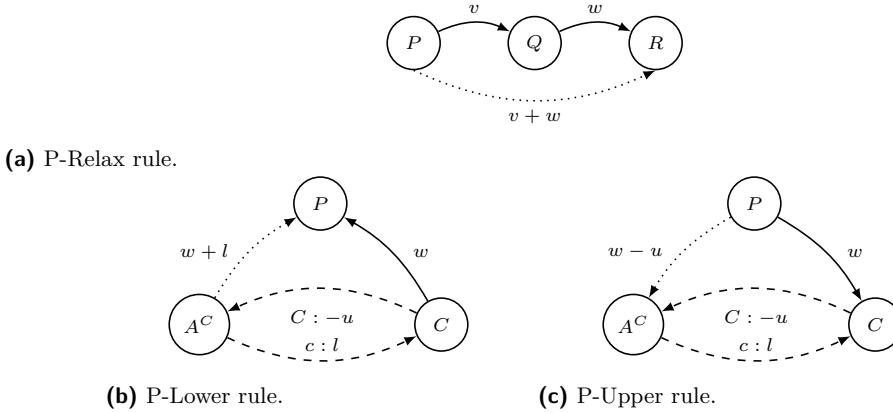


Figure 4 Constraint inference rules. The derived constraints are dotted.

P-Lower: This rule is applied when a time point P is the target of an upper-bound constraint with origin the contingent time point of a contingent link. Given the minimum contingent duration l , and the upper-bound constraint bound w , the rule derives an upper-bound constraint from the activation time point of the contingent link to P , with bound $w + l$.

► **Lemma 3.** P-Lower derives the broadest non-contingent constraint $c = (P \leq A^C + w + l)$ between time point A^C and time point P in a dynamically controllable STNU S , which makes the constraint $(P \leq C + w)$ between C and P redundant.

Proof. Redundancy: the introduction of c in S requires to satisfy $P \leq A^C + w + l$. $\forall d : l \leq d \leq u, A^C + d \leq C$. So $P \leq A^C + w + l \implies P \leq C + w$.

Broadness: suppose that the derived constraint between A^C and P is less strict, e.g. $P \leq A^C + w + l + 1$. Then in the scenario in which $P = A^C + w + l + 1$ and the contingent link takes its minimum duration ($C = A^C + l$), the requirement that $P \leq C + w$ from the original constraint is violated by 1 time unit. Therefore $P \leq A^C + w + l$ is the most lenient constraint which can be set between A^C and P without contradicting the one between C and P . ◀

P-Upper: This rule is applied when a time point P is the origin of an upper-bound constraint with target the contingent time point of a contingent link. Given the maximum contingent duration u , and the upper-bound constraint bound v , the rule derives an upper-bound constraint from P to the activation time point of the contingent link, with bound $w - u$.

► **Lemma 4.** P-Upper derives the broadest non-contingent constraint $c = (A^C \leq P + w - u)$ between time point P and time point A^C in a dynamically controllable STNU S , which makes the constraint $(C \leq P + w)$ between P and C redundant.

Proof. Redundancy: the introduction of c in S requires to satisfy $A^C \leq P + w - u$, i.e. $A^C + u \leq P + w$. $\forall d : l \leq d \leq u, A^C + d \leq C$. So $A^C \leq P + w - u \implies C \leq P + w$.

Broadness: suppose that the derived constraint is less strict, e.g., $A^C \leq P + w - u + 1$. Then the scenario in which $P = A^C - w + u - 1$ and the contingent link takes its maximum duration ($C = A^C + u$), the original constraint $C \leq P + w$ is violated by 1 time unit. Therefore $A^C \leq P + w - u$ is the most lenient constraint between P and A^C without contradicting the constraint between P and C . ◀

With these basic inference rules we can now define a procedure for inferring parameter ranges and restrictions through iterated application of these rules.

4.3 Local Inference of Parameter Restrictions

The inference procedure iteratively applies the basic inference rules to a dynamically controllable STNU S until quiescence (i.e. no new edge can be derived). We call the resulting STNU the *closure* of S .

► **Lemma 5.** *The inference procedure is sound and complete.*

Proof. (1) The procedure always terminates, as each rule only adds constraints, no rule has the absence of a constraint as condition, increments in constraints are always multiples of 1 chronon, and there is an overall deadline. (2) The correctness of a derived constraint is an immediate consequence of the correctness of each of the three rules. (3) As each of the rules derives a constraint which is at least as strict as a constraint derived by the rules in [3] but is the weakest constraint satisfying the requirements (Lemma 1-3) completeness follows from the completeness of the rules in [3]. ◀

The closure can now be analyzed to extract parameter restrictions. We are mainly interested in the derived links between a parameter and *zero* and between 2 parameters. For a node n , a non-contingent edge (zero, n, w) means that n is allowed to take at most value w , for the STNU to be dynamically controllable. A non-contingent edge $(n, \text{zero}, -v)$ means that n has to take a value greater or equal to v , for the STNU to be dynamically controllable. We call $[v, w]$ the range restriction for n .

► **Lemma 6.** *Let S be the closure of the STNU for a process P . Let $p \in \text{param}(S)$ (p is a temporal parameter of P) such that $\forall q \in \text{param}(S) \nexists (p, q, \delta), (q, p, \delta') \in S$. Let $(\text{zero}, p, w), (p, \text{zero}, -v)$ be edges in S . Then $[v, w]$ is the broadest range of values for p which is compatible with the dynamic controllability of S .*

Proof. Let us assume that S as above is dynamically controllable. It is easy to see, that fixing p (by introducing edges $(\text{zero}, p, \bar{p})$ and $(p, \text{zero}, -\bar{p})$) to any value \bar{p} either smaller than v , or larger than w , would introduce a negative loop in the STNU, thus making it not dynamically controllable. Now let us assume that S is dc but there is a value $v \leq \bar{p} \leq w$ for p , such that $S' = S \cup \{(\text{zero}, p, \bar{p}), (p, \text{zero}, -\bar{p})\}$ is not dc, i.e. there is a negative cycle in S' . Then with the Decomposability theorem [10] we can conclude that the negative cycle was already in S - which is a contradiction to the assumption that S is dynamically controllable. ◀

STNU edges derived between any two nodes m and n , represent constraints restricting the values for the timestamps of these nodes with respect to each other, which as well need to be fulfilled, in order for the STNU to be dynamically controllable. For instance, a derived edge $(m, n, 20)$ would mean *the timestamp of n must be no more than 20 time units after the timestamp of m* .

► **Lemma 7.** *Let S be the closure of the STNU for a process P . Let p and q be nodes in S for temporal parameters. Let (p, q, w) be an edge in S . Then $q \leq p + w$ is the broadest constraint between the timestamps of p and q which is compatible with the dynamic controllability of S .*

Proof. It is easy to see, that fixing q (by introducing edges $(\text{zero}, q, \bar{q})$ and $(q, \text{zero}, -\bar{q})$) to any value \bar{q} violating $q \leq p + w$, would introduce a negative loop in S , thus making it not dynamically controllable.

■ Listing 1 Compute, distribute, and receive parameter restrictions

```

repeat
    S' := S;
    S' := infer_restrictions(S);
    for (n in N | n.type = receive)
        transmit_restrictions(n);
    end for
    for (n in N | n.type = send)
        S' := S' ∪ receive_restrictions(n);
    end for
until S' = S;
for (n in N | n.type = receive)
    make_contingent(S, n);
end for

```

Now suppose that q is fixed (by introducing edges as above) to any value \bar{q} fulfilling $q \leq p + w$ and all other constraints on q , and that a negative cycle is in S . Then the negative cycle must be due to some other configuration of constraints, which is a contradiction since S is dynamically controllable. ◀

4.4 Communicating and Negotiating Constraints

A local STNU communicates the inferred restrictions to the senders resp. receivers of the parameters, with the aim of deriving a shared set of restrictions leading to a constellation of communicating STNUS each of which is dynamically controllable.

A difficulty for such a procedure are inter-dependencies between parameters. Consider, as an example, a local STNU LP_0 , which receives two input parameters: parameter p_1 is received from a STNU LP_1 , and parameter p_2 from a different STNU LP_2 . Through inference, it is discovered that, for the dynamic controllability of LP_0 , both p_1 and p_2 have to take values in the range $[15, 20]$; and p_2 must be at least equal to $p_1 - 1$, and at most equal to $p_1 + 1$. So it is not sufficient that LP_0 asks LP_1 and LP_2 to provide parameters with values in the ranges, since, e.g., $p_1 = 16$ and $p_2 = 20$ would be invalid instantiations.

In Listing 1 we provide a general formulation of an algorithm for deriving a shared set of restrictions on parameters. For each local STNU S , restrictions for its parameters can be derived by applying the procedure for range inference. Then, all inferred restrictions for input parameters are sent to the respective senders. Restrictions to output parameters computed at the receivers are then received and corresponding non-contingent edges are added to the STNU. The procedure repeats until no change occurs at any local STNU. The procedure assumes that all parameter restrictions are shared between all STNUS, and, therefore, each local STNU can observe when a fix-point is reached or a contradiction is derived. Nonetheless, the algorithm does not require exposing internal constraints of local STNUS.

There are several variants for implementing this basic procedure, with different effects and assumptions, e.g.:

1. Global choreography for the cross-organizational process, with shared parameter space: “*everybody knows all parameter constraints*”;
2. A central coordinator, with full access to all parameter restrictions acts as mediator between local processes, and recognizes, when a fix-point is reached;

3. Global hierarchical process without a global view or central coordinator: a partial order of parameter and process dependencies can be constructed;
4. Fully distributed procedure based on distributed cycle detection (e.g. [2, 27]) for addressing dependencies between parameters.

A discussion of each of these variants and a comparison of their advantages and disadvantages is out of the scope of this paper, and we reserve it for future work. Future work also includes the development of algorithms, which are adaptive to the topology of the communication structure.

4.5 Interpreting the Results

If in the procedures described any STNU gets to a negative cycle, then the cross-organizational process is not dynamically controllable.

Otherwise, the algorithm in Listing 1 results in a set of dynamically controllable STNUs which are coherent in their constraints on the shared temporal parameters. The restrictions on these parameters can now be interpreted as follows: the constraints on input parameters define for which constellation of parameter values the network is dynamically controllable and are thus requirements sending processes have to fulfill. The restrictions on output parameters are guarantees which the receiving processes are allowed to assume. Constraints between parameters show the trade-offs to be resolved by further negotiations.

4.6 Correctness and Complexity of the Procedure

We observe that the correctness of the algorithm in Listing 1 is based (1) on the correctness of the procedure for inferring parameter restrictions called in line 4, and (2) on the fact that introducing contingent edges specifying the ranges for input parameters in line 13 does not violate dynamic controllability. For (1), an informal sketch of proof is based on the inference procedure deriving ranges which are not less restrictive than the ones derived by the more general rules of [3]. Thus, if a closure of the STNU can be computed, the STNU is dynamically controllable. For (2), we give proof to the following Lemma:

► **Lemma 8.** *Let P be a process; let S be the STNU for P . Let p_1, \dots, p_n be input parameters, and $r_i = [p_{i_{\min}}, p_{i_{\max}}]$ be range restrictions for each p_i derived from the closure of S . Then $S \bigcup_i (\text{zero}, p_{i_{\min}}, p_{i_{\max}}, p_i)$ is dynamically controllable.*

Proof. For parameters for which only range restrictions are derived in the closure, the proof directly follows from Lemma 6. For parameters having additional restrictions, the proof follows from Lemma 7, and the requirement that all derived restrictions are enforced by the senders, in whose STNU these restrictions are non-contingent edges. ◀

At the basis of the algorithm in Listing 1 is the repeated application of the procedure for inferring restrictions. From previous results [25], the complexity of STNU constraint propagation algorithms is polynomial in the number of STNU nodes $O(N^4)$. Existing figures [11] for the local application of such procedures to STNUs derived from process definitions indicate the practical applicability of the approach at design time. The local execution of the algorithm may invoke several times the inference procedure, if re-computation is necessary. However, each new invocation only restricts previously computed bounds, and the procedure stops in case a negative cycle is found. Thus, the overall complexity of the distributed execution of the algorithm in Listing 1 at each local participant, is given by $O(N^4)$, with N the number of nodes in the global STNU.

5 Related Work

A substantial body of research is devoted to time management for business processes: general overviews of works in the area can be found in [7, 9, 14]. Checking whether deadlines and time constraints can be fulfilled in time-constrained process definitions is addressed by early works such as [1, 24], which are based on network analysis, scheduling, or constraint networks. The specific case of cross-organizational processes and service compositions is addressed by works such as [4, 18]; modularized processes are addressed in [21]. However, none of these approaches considers using temporal parameters for expressing temporal properties or temporal requirements; here we have shown how temporal parameters enable the expression of temporal constraints crossing the scope of a single intra-organizational process.

Pro-active monitoring of the compliance of process instances to their process model, which is considered in, e.g., [19, 22], plays an essential role in the management of timed processes in general. Collaborative processes in particular are addressed in [23], which is based on timed automata and model checking techniques. However, all these approaches consider satisfiability rather than dynamic controllability as the notion for temporal correctness.

Here we showed an algorithm for computing constraints on parameters, which is based on inferring knowledge from the temporal aspect of a local process definition, and the communications with other local process definitions, with no visibility of their internals. Alternatively, process mining techniques [29] may be used to derive missing temporal qualities for a model; however, this is only possible if there is a sufficient number of traces available in the process logs. In contrast to such an approach, here we focus on new process definitions, and on a design time check of their temporal properties.

As an implementation for the proposed algorithm, we showed an approach based on mapping process definitions to Simple Temporal Networks with Uncertainty (STNUs) [26]. Considerable research efforts have been devoted in the last decades both to developing different notions of controllability and more expressive network models [21, 31, 32]. Considering the increasing complexity for verifying dynamic controllability of these more refined networks, we regard STNUs as a suitable formalism for representing the temporal dimension of a process model and deriving missing temporal information at design time.

6 Conclusions

Temporal parameters proved as a highly adequate means for expressing temporal obligations and guarantees between the participants of a cross-organizational business process. Negotiating constraints on temporal parameters has to respect the need for keeping internals of the participating processes secret, while arriving at a solution, which allows the distributed control of processes in a way that no temporal constraint is violated.

The procedures proposed in this paper are a first attempt to successfully support the negotiation of temporal commitments through the computation of maximum ranges for input parameters and minimum ranges for output parameters in an effective way. These parameter ranges serve as obligations and guarantees of temporal properties for the participants in the cross-organizational business process. Additionally, they build the basis for the distributed and autonomous scheduling of the participating processes, without risking time failures and temporal exceptions.

References

- 1 Claudio Bettini, X.Sean Wang, and Sushil Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002.
- 2 Lubos Brim, Ivana Černá, and Lukás Hejtmánek. Distributed negative cycle detection algorithms. In *Advances in Parallel Computing*, volume 13, pages 297–304. Elsevier, 2004.
- 3 Massimo Cairo and Romeo Rizzi. Dynamic controllability of simple temporal networks with uncertainty: Simple rules and fast real-time execution. *Theor. Comput. Sci.*, 797:2–16, 2019. doi:10.1016/j.tcs.2018.11.005.
- 4 Jorge Cardoso, Amit Sheth, John Miller, Jonathan Arnold, and Krys Kochut. Quality of service for workflows and web service processes. *J of Web Semantics*, 1(3):281–308, 2004.
- 5 Giuseppe Castagna. Covariance and contravariance: conflict without a cause. *ACM Transactions on Programming Languages and Systems*, 17(3):431–447, 1995.
- 6 Issam Chebbi, Schahram Dustdar, and Samir Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering*, 56(2):139–173, 2006.
- 7 Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. The temporal perspective in business process modeling: a survey and research challenges. *Service Oriented Computing and Applications*, 9(1):75–85, 2015.
- 8 Carlo Combi and Mauro Gambini. Flaws in the flow: The weakness of unstructured business process modeling languages dealing with data. In *OTM Confederated International Conferences*, pages 42–59. Springer, 2009.
- 9 Carlo Combi and Giuseppe Pozzi. Temporal conceptual modelling of workflows. In *Conceptual Modeling-ER 2003*, pages 59–76. Springer Berlin Heidelberg, 2003.
- 10 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95, 1991.
- 11 Johann Eder, Marco Franceschetti, and Julius Köpke. Controllability of business processes with temporal variables. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 40–47. ACM, 2019.
- 12 Johann Eder, Marco Franceschetti, Julius Köpke, and Anja Oberrauner. Expressiveness of temporal constraints for process models. In *International Conference on Conceptual Modeling*, pages 119–133. Springer, 2018.
- 13 Johann Eder, Nico Kerschbaumer, Julius Köpke, Horst Pichler, and Amirreza Tahamtan. View-based interorganizational workflows. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, pages 1–10. ACM, 2011.
- 14 Johann Eder, Euthimios Panagos, and Michael Rabinovich. Workflow time management revisited. In *Seminal Contributions to Information Systems Engineering*, pages 207–213. Springer Berlin Heidelberg, 2013.
- 15 Marco Franceschetti and Johann Eder. Checking temporal service level agreements for web service compositions with temporal parameters. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 443–445. IEEE, 2019.
- 16 Marco Franceschetti and Johann Eder. Designing decentralized business processes with temporal constraints. In *CAiSE Forum (forthcoming)*, 2020.
- 17 Paul Grefen and Yigal Hoffner. Crossflow-cross-organizational workflow support for virtual organizations. In *Proceedings 9th Int. Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises.*, pages 90–91. IEEE, 1999.
- 18 Nawal Guermouche and Claude Godart. Timed model checking based approach for web services analysis. In *ICWS 2009.*, pages 213–221. IEEE, 2009.
- 19 Mustafa Hashmi, Guido Governatori, Ho-Pun Lam, and Moe Thandar Wynn. Are we done with business process compliance: state of the art and challenges ahead. *Knowledge and Information Systems*, pages 1–55, 2018.
- 20 Julius Köpke, Marco Franceschetti, and Johann Eder. Optimizing data-flow implementations for inter-organizational processes. *Distributed and Parallel Databases*, pages 1–45, 2018.

- 21 Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert. Controlling time-awareness in modularized processes. In *Enterprise, Business-Process and Information Systems Modeling*, pages 157–172. Springer, 2016.
- 22 Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M.P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information systems*, 54:209–234, 2015.
- 23 Sihem Mallek, Nicolas Daclin, Vincent Chapurlat, and Bruno Vallespir. Enabling model checking for collaborative process analysis: from bpmn to ‘network of timed automata’. *Enterprise Information Systems*, 9(3):279–299, 2015.
- 24 Olivera Marjanovic and Maria E. Orlowska. On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems*, 1(2):157–192, 1999.
- 25 Paul Morris. A structural characterization of temporal dynamic controllability. In *International Conference on Principles and Practice of Constraint Programming*, pages 375–389. Springer, 2006.
- 26 Paul H Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *AAAI*, pages 1193–1198, 2005.
- 27 Gabriele Oliva, Roberto Setola, Luigi Glielmo, and Christoforos N Hadjicostis. Distributed cycle detection and removal. *IEEE Trans. on Control of Network Systems*, 5(1):194–204, 2016.
- 28 Irfan Ul Haq, Altaf Huqqani, and Erich Schikuta. Aggregating hierarchical service level agreements in business value networks. In *International Conference on Business Process Management*, pages 176–192. Springer, 2009.
- 29 Wil M.P. van der Aalst, M.Helen Schonenberg, and Minseok Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011.
- 30 Wil MP van der Aalst and Mathias Weske. The p2p approach to interorganizational workflows. In *International Conference on Advanced Information Systems Engineering*, pages 140–156. Springer, 2001.
- 31 Thierry Vidal. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):23–45, 1999.
- 32 Matteo Zavatteri and Luca Viganò. Conditional simple temporal networks with uncertainty and decisions. *Theoretical Computer Science*, 797:77–101, 2019.
- 33 Michael Zur Muehlen and Jan Recker. How much language is enough? theoretical and practical use of the business process modeling notation. In *Seminal Contributions to Information Systems Engineering*, pages 429–443. Springer, 2013.

A Mapping Process Models into STNUs

Previous works have already shown how to map a process model into an equivalent temporal network, such as the Simple Temporal Networks with Uncertainty (STNU), for expressing the temporal semantics and verifying temporal correctness of the process model [11, 16]. The advantage of mapping to temporal networks is that it is an established formalism with sound and complete procedures for temporal reasoning. To be self-contained, we report here a brief definition of the STNU, and the rules which allow mapping the process model of Def. 1 into a STNU.

A.1 STNU

A STNU $S = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is a directed weighted graph, in which nodes (set \mathcal{T}) represent time points, and edges (sets \mathcal{C}, \mathcal{L}) represent temporal constraints between pairs of time points. A special node *zero* (Z) marks the reference in time after which all other time points occur. Two types of edges exist: non-contingent (set \mathcal{C}), and contingent (set \mathcal{L}). Non-contingent

STNU edges between time points A and B have the form (A, B, δ) : they require to assign A and B timestamps, such that $B \leq A + \delta$ holds. A contingent STNU edge (also called link) between from a time point A^C (called the activation time point) to a contingent time point C have the form (A^C, l, u, C) : they state that the timestamp of C will be observed to fall between $A^C + l$ and $A^C + u$.

Dynamic controllability of a STNU requires the existence of a dynamic execution strategy, which assigns values to the non-contingent nodes, such that all temporal constraints are met, for all possible assignment of values to the contingent nodes. A frequently adopted approach to check the dynamic controllability of a STNU is based on propagating its constraints. Constraint propagation derives new edges according to a number of propagation rules. Different systems of rules have been proposed over the years, e.g., [3, 26]. A STNU is dynamically controllable if it is not possible to derive any negative loop through constraint propagation.

A.2 Mapping Rules

Given the process model of Def. 1, we show here how to map it into a STNU. We formulate the mapping in terms of mapping rules as follows:

► **Definition 9** (Mapping to STNU). *Let $P = (\text{proc_id}, N, E, V, C, \Omega)$ be a process defined as in Def. 1. The STNU $S = (\mathcal{T}, \mathcal{C}, \mathcal{L})$, equivalent to P , is obtained by applying the following rules:*

1. *Each $n \in N$ with $n.s, n.e \in N^e$ is mapped into corresponding time points $n.s, n.e \in \mathcal{T}$;*
2. *Each $v \in V^I \cup V^O$ is mapped into a corresponding time point $v \in \mathcal{T}$;*
3. *Each $e = (m, n) \in E$ is mapped into a corresponding non-contingent edge $(n.s, m.e, 0) \in \mathcal{C}$;*
4. *$(\text{start}.s, \text{zero}, 0) \in \mathcal{C}$, and $(\text{zero}, \text{end}.e, \Omega) \in \mathcal{C}$;*
5. *Each duration constraint $d(n, d_{\min}, d_{\max}) \in C$ is mapped into a corresponding contingent link $(n.s, d_{\min}, d_{\max}, n.e) \in \mathcal{L}$;*
6. *Each range constraint $r(v, v_{\min}, v_{\max}) \in C$, with $v \in V^I$, is mapped into a corresponding contingent link $(\text{zero}, v_{\min}, v_{\max}, v) \in \mathcal{L}$;*
7. *Each range constraint $r(v, v_{\min}, v_{\max}) \in C$, with $v \in V^O$, is mapped into corresponding non-contingent edges $(\text{zero}, v, v_{\max}), (v, \text{zero}, -v_{\min}) \in \mathcal{C}$;*
8. *Each $\text{ubc}(a, b, \delta) \in C$, is mapped into a corresponding non-contingent edge $(a, b, \delta) \in \mathcal{C}$;*
9. *Each $\text{lbc}(a, b, \delta) \in C$, is mapped into a corresponding non-contingent edge $(b, a, -\delta) \in \mathcal{C}$.*

To keep the STNU compact and without loss of generality, process nodes with duration 0 can be collapsed into a single STNU node; process control nodes may not be included in the STNU, by linking their predecessors to their successors; similarly, we may make *start* coincide with *zero* (see Fig. 2 and Fig. 3).

We use the mapping rules of Def. 9 to bring the definition of the temporal aspect of a process model into STNU terms for further temporal reasoning, such as checking its dynamic controllability, and deriving missing temporal information.