Towards Checking Dynamic Controllability of Processes with Temporal Loops

Marco Franceschetti, Johann Eder

Department of Informatics-Systems, Alpen-Adria Universität Klagenfurt, Austria
firstname.lastname@aau.at

Abstract. We propose a technique for checking the dynamic controllability of processes with temporally constrained loops. The temporal control structures t-split and t-loop are adequate concepts for modelling the behaviour of processes dependent on the speed of execution of a process instance. While the run-time semantics of these constructs is straightforward, the assessment of the temporal properties of processes with temporal control structures is complex. Here we propose a procedure for checking the dynamic controllability of processes with temporal loops by mapping them to processes without t-loops, which then can be mapped to CSTNUDs, simple temporal networks with uncertainty and decisions. For CSTNUDs procedures for checking dynamic controllability are available.

1 Introduction

An important part of the modelling of processes is to represent their temporal properties and to model the temporal constraints expressing temporal requirements and restrictions. Compliance to temporal constraints like deadlines, reaction times, process durations rank among the most important quality measures [3, 14, 25, 2] for (business) process management. In the past two decades many modelling languages addressing temporal aspects and corresponding procedures for checking temporal properties of modelled processes at design time have been developed. Elaborate concepts are available for expressing temporal constraints in a declarative way [19, 11] and for checking satisfiability or controllability of process definitions with temporal constraints.

Temporal qualities of processes rely on the temporal characteristics of the activities and services they are composed of. These activities might have some degrees of freedom, or some degrees of uncertainty formulated in their temporal descriptions. For an example the time-span needed for executing an activity might range between some minimum and maximum duration, without any control of the actual duration by the process controller. On the other hand, the process controller has to guarantee some temporal properties (like e.g. maximum duration of the whole process) in spite of these uncertainties. Therefore, the notion of controllability rather than satisfiability is adequate as correctness criterion for processes with temporal constraints. Satisfiability only requires that it is possible to satisfy all constraints, for *some* durations of the activities and for

some flow decisions, however not necessarily *for all* durations of these activities. Controllability, in contrast, requires that all temporal constraints are fulfilled for *all* durations of invoked activities and for all possible flows.

Checking satisfiability, controllability and its less restrictive but more complex variant dynamic controllability is already well established for basic control patterns [7, 9, 17, 16]. Nevertheless, the current situation has the following shortcomings:
(i) the separation of concerns between proscriptive control structures and declarative temporal constraints in process definitions is sometimes cumbersome,
(ii) constructs for addressing temporal aspects in process enactment systems are mostly too low level (timer events, exception handling) and need to be raised to the level of conceptual modeling of processes.

As an approach to overcome these problems, the notion of temporal control structures (temporal splits and temporal loops) for process models was introduced and their semantics was formally defined [24]. In particular, [24] proposed a temporal loop construct, which allows - to the best of our knowledge for the first time - to meaningfully extend the notion of controllability to processes with *while* loops. A temporal loop has the following format: *while elapsed* $\leq c$. The operational semantics of temporal loops is straightforward: when the loop-head is executed, the temporal condition is evaluated, i.e. whether the actual elapsed time of the process is less than the threshold c, and on basis of this evaluation the system makes the decision whether another iteration of the loop body is launched.

Although the operational semantics of temporal control structures is easily defined and implemented, algorithms for design time checks of controllability and dynamic controllability, and for pro-active monitoring of the compliance of the process execution at runtime turned out to be quite challenging. In [8] we proposed a technique for checking the controllability of processes with temporal splits by transforming the temporal splits to non-contingent decision nodes together with some constraints. Such a transformed process can then be mapped to CSTNUDs which can be checked for dynamic controllability.

In this paper we consider the dynamic controllability of processes with temporal loops. We will introduce a procedure which transforms a process with a temporal loop to a process without any loop with the property, that the first process is dynamically controllable, if and only if the second is dynamically controllable and thus we can reduce the checking to the already solved case for processes without temporal loops for which we already proposed effective procedures for checking dynamic controllability [8].

The remainder of this paper is organized as follows: we revisit the definition of process models with temporal loops and the definition of their semantics in section 2. In section 3 we discuss dynamic controllability and introduce some process transformations which eliminate temporal loops. The results of these transformations can then be checked for dynamic controllability instead of the original process. Related work (section 4) and conclusions (section 5) complete this paper.

2 Temporal Control Structures

2.1 Process Model

As a basis for the introduction of procedures for checking dynamic controllability, we first define the process model with temporal control structures. It is basically the process model in [24] which introduced temporal splits and temporal joins for the first time, however we extend this model with explicit temporal constraints (upper-bound and lower-bound constraints between process events). All the definitions and examples are taken from [24] and are repeated here to make the paper self-contained.

We use a simple process metamodel, which focuses on the standard minimum workflow control patterns [27]. However, the definitions and algorithms can be easily extended to other control flow patterns.

Time points and duration intervals are represented by natural numbers $\mathbb{N}$. Time is measured in an atomic time unit (chronon), like minutes, hours or days. A point in time marks a point on an increasing time axis and represents the temporal distance to a given reference point, for instance the start of a process.

The maximum process duration is constrained by a deadline, that must not be exceeded. Activity instances start at a certain point in time (start event) and end at a certain point in time (end event). The distance between these two points is the actual execution duration of this activity instance. Contingent activities have a duration between their best-case (fastest) and worst-case (slowest) duration which can only be observed (but not controlled). In the following we regard all activities as contingent.

Definition 1 (Process Graph). *A* process graph $G = (N, E, C, \omega)$ *with a set of nodes* N *connected by a set of edges* E *with set of temporal constraints* C *and deadline* ω *forms a directed acyclic graph, where* $n.type = activity \mid start \mid end \mid xor\text{-}split \mid xor\text{-}join \mid and\text{-}split \mid and\text{-}join$*, which have the usual semantics, or a temporal node with* $n.type = t\text{-}loop \mid t\text{-}split$*.* $n.cond$ *is the boolean condition for a node* n *of type* $xor - split$*.*

$n.d_b$ *is the minimum and* $n.d_w$ *maximum duration of a node* $n \in N$ *such that* $0 \leq n.d_b \leq n.d_w$ *and* $n.d_b, n.d_w \in \mathbb{N}$*.*

Each edge $(n_1, n_2) \in E$*,* $n_1, n_2 \in N$*, describes a precedence constraint between nodes* n_1 *and* n_2*. The successors and predecessors of a node* n *are denoted* $n.Succ = \{m|(n, m) \in E\}$ *and* $n.Pred = \{m|(m, n) \in E\}$ *respectively.*

There is exactly one start activity (which has no predecessor) and one end activity (which has no successor). The number of predecessors $|n.Pred|$ *per node-type* $n.type$ *is as follows: 0 for start, 2 for* $xor - join$ *and* $t - join$*,* > 1 *for* $and - join$*, and 1 for all other types. The number of successors* $|n.Succ|$ *per node-type* $n.type$ *is restricted as follows: 0 for end, 2 for* $xor - split$ *and* $t - split$*,* > 1 *for* $and - join$*, and 1 for all other types.*

The variable elapsed represents the distance between the start of the process and a point in time.

A temporal split node n *has a threshold* $n.c$ *where* $n.c \in \mathbb{R}_{\geq 0}$*, and a true successor and a false successor.*

A temporal loop $tl \in N$, $tl.type$ = t-loop has a loop body $tl.B = G'$ where G' is a process graph, and a loop condition $l.cond = [P\wedge](elapsed \leq c), c \in \mathbb{N}$ with the optional predicate P. The duration interval of the loop-body is represented by $l.b_b$ and $l.b_w$.

We write tsplit(n,c,u,d) for a t-split node n with threshold c, true successor u and false successor d, and tloop(n,P,c,B) for a t-loop node n with predicate P, threshold c, and loop body B.

C is a set of temporal constraints of type upper- and lower-bound, which we write as $ubc(m, n, \delta)$, resp. $lbc(m, n, \delta)$, where m, n are nodes and δ is a constant. $ubc(m, n, \delta)$ forces n to start no more than δ time units after m started; $lbc(m, n, \delta)$ forces n to start at least δ time units after m started. □

We assume that a process graph is full-blocked (proper nesting of matching pairs). Xor-joins are of type simple merge, i.e. it is not possible that both predecessors will be executed in a single process instance.

Without loss of generality in this paper we only consider the temporal conditions ($elapsed \leq c$).

A temporal loop $tloop(n, P, c, B)$ iterates over its body as long as the condition is true. A temporal loop's condition specifies an upper bound in addition to a regular loop-condition. The body of a loop is again a process graph and might include temporal control structures. The variable *elapsed* is always defined relative to the start node of the process graph.

A t-split node is a special variant of an xor-split node with a temporal condition. Here we currently provide the condition "$elapsed \leq n.c$", where $n.c$, the threshold value, is a constant. The operational semantics is as follows: when a t-split node is executed, the condition is evaluated, i.e. the time elapsed since process start is compared to the threshold $n.c$. If it is below or equal to the threshold, the "T" branch is executed, otherwise, execution continues with the "F" branch. We assume that after starting execution of a t-split its condition is evaluated immediately.

As we have shown in [8], with t-split nodes a process controller has some influence on the selection of the successor. Take for an example a t-split n with the condition $elapsed \leq 20$: if the condition is evaluated before time-point 20, then the "$true$" path is chosen, if it is evaluated after time point 20, the "$false$" path is selected. Therefore, the process controller might be able to control the result of the condition evaluation and thus might control which of the successors is chosen. This is, however, only possible, if the execution of the process so far allows for such a shift over the threshold of the t-split.

2.2 Scenarios

To formalize the semantics of the proposed temporal control structures and to reason about the correctness of a given model we now introduce the concepts scenario, schedule, and controllability. In a nutshell: a scenario is a timed instance of process. It is correct, if the time stamps of the start and end events of the activities satisfy all explicit and implicit temporal constraints (an activity starts

after termination of its predecessor). A schedule assigns a start and end time intervals to each activity, such that all scenarios are valid if start and end-time are within the bounds defined in the scenario. A process is controllable, if it admits a correct schedule. The following requires some formal elaboration. This is necessary for being able to check controllability of process models.

In a scenario time stamps for the start and the end are assigned to each node, representing one out of many possible process executions.

Definition 2 (Scenario). *A scenario $\bar{S}$ for a process graph $P(N, E, C, \omega)$ associates each $n \in N$ and the body $l.B$ of each t-loop l with two time stamps $t_s, t_e \in \mathbb{N}$, representing the start time and end time of n respectively. We call $(n, t_s, t_e) \in \bar{S}$ a scenario entry.* □

We define a valid scenario as formalization of the conceptual semantics of control flow structures described in Section 2.

Definition 3 (Valid Scenario). *A scenario $\bar{S}$ for a process graph $P(N, E, C, \omega)$ is valid, iff:*

(1) $\forall n \in N$: $n.t_s \leq n.t_e \leq \delta$
(2) $\forall n \in N, n.type \neq t-loop : n.t_s + n.d_b \leq n.t_e \leq n.t_s + n.d_w$
(3) $\forall (m,n) \in E, m.type \neq\ t-split : m.t_e \leq n.t_s$
(4) $\forall n \in N, t-split(n,c,u,d)$:
$n.t_s \leq c \Rightarrow n.t_s \leq u.t_s$
$n.t_s > c \Rightarrow n.t_s \leq d.t_s$
(5) $\forall n \in N, tloop(n, True, c, B), n.t_s \leq c$:
$c < n.t_e \leq c + n.b_w$,
(6) $\forall n \in N, tloop(n, P, c, B), n.t_s \leq c$:
$n.t_s \leq c \Rightarrow n.t_s \leq n.t_e \leq c + n.b_w$
(7) $\forall ubc(m,n,\delta) : n.t_s \leq m.t_s + \delta$
(8) $\forall lbc(m,n,\delta) : m.t_s \leq n.t_s - \delta$ □

For the new control structures the following must hold: If a temporal loop with the condition ($elapsed \leq c$) starts before or at c, it will run through the loop body at least until c is reached, plus one final loop-iteration (worst-case duration of the loop body). The end of the loop is always after the cut-off point c. If the loop starts after c, the loop body will not be executed. If the body of the loop is executed, its last iteration does not start before the beginning of the loop, and it does require at least the minimum duration of the body as distance between start and end of the loop. For the end of the loop we require that it can accommodate all durations of the loop body between best-case and worst-case duration. And finally (6) states that t-loops with the condition $P \wedge (elapsed \leq c)$ can finish anytime before c due to the condition P, but they can last until $c +$ duration of the loop body.

An example for a process with one possible valid (valid) scenario, is shown in Figure 1 [24]. In this particular scenario A starts at 0 and ends at 15, and B starts at 20 and ends at 32, and so on. The t-split U ends at 80, and as

elapsed ≤ 40 the false-successor W will be chosen over V (which is given but can be neglected in this scenario, hence its start is set to 40). The loop L was not entered, as the start time of L is 80 and the condition of the temporal loop is *while* (*elapsed* ≤ 70). The process ends within the deadline of 200 at 112.

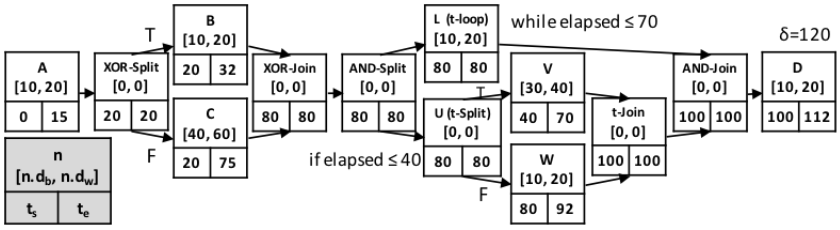

Fig. 1. Process Model with Valid Scenario [24]

3 Dynamic Controllability

3.1 Dynamic Controllability and Loops

Traditionally, temporal correctness of a process was discussed with the notions consistency or satisfiability, which require the existence of at least one trace which meets all temporal constraints. Nevertheless, as several works pointed out, satisfiability is a too weak notion for temporal correctness, since designers look for stronger guarantees that the violation of temporal constraints can be avoided. In particular, it is desirable to know at design time, whether all constraints can be satisfied for all considered circumstances, i.e. for possible durations and conditions. This need led to the formulation of the notion of (dynamic) controllability (see [7] for formal definition) and to the development of techniques to check at design time whether a process is dynamically controllable. In a nutshell: a process is dynamically controllable, if the start times of its activities can be dynamically assigned in response to the durations and conditions observed at run time in such a way, that no temporal constraint will be violated. In order to assess the dynamic controllability of a process, it is necessary to know all possible duration intervals for activities.

Time constrained processes with unbounded loops (while loops, repeat-until loops, etc.) can never be dynamically controllable, since the number of iterations and, therefore, the maximum duration of the loop cannot be known at design time or even at run-time before the loop finished. For temporal loops the situation is different, as the temporal condition in a t-loop defines a temporal bound for the duration of a t-loop and hence also limits the number of possible iterations.

For checking, whether a process with t-loops is dynamically controllable, we follow a model transformation approach [10, 8]. In particular, we propose the

following procedure: a process with t-loops is mapped to a process without t-loops which is equivalent to the first one in terms of dynamic controllability. Then the dynamic controllability of the latter one can be checked with existing procedures, in particular by mapping the process to a CSTNUD (Conditional Simple Temporal Network with Uncertainty and Decisions).

In the following, we introduce three transformations: the first is based on an unfolding of the t-loop, which is also the basis for the other 2 transformations, which can be seen as compression of the loop unfolded process and is significantly more compact and smaller, while having equivalent dynamic controllability.

3.2 Loop Unfolding

Loop unfolding transforms a process with t-loops to a process without t-loops with identical behavior (i.e. it admits exactly the same set of traces, resp. the same set of scenarios). Unfolding of general loops suffers, of course, from the problem that the number of iterations cannot be known (and may even be infinite). For t-loops we exploit the temporal bound of the loop to terminate the unfolding procedure. Therefore, the unfolding procedure has to be interleaved with some temporal calculations. In particular, unfolding has to include the computation of the earliest possible execution time of nodes, which we represent as an additional attribute for each node, which will be computed in analogy to the procedures in [11].

Definition 4 (earliest execution time). *Let $P = (N, E, C, \omega)$ be a process. For each node $n \in N$, the earliest execution time $n.e$ is computed as follows:*

1. *$start.e := 0$;*
2. *for nodes n following a temporal loop with $tloop(m, P, c, B)$: $n.e := m.e$;*
3. *for nodes n following a temporal loop with $tloop(m, True, c, B)$: $n.e := c + 1$;*
4. *for and-join nodes: $n.e := max(\{m.e + m.d_{min} | (m, n) \in E\}$;*
5. *for all other nodes: $n.e := min(\{m.e + m.d_{min} | (m, n) \in E\}$.* □

We define here an elementary transformation step of one elementary unfolding step of one t-loop. The transformation of the whole process is then achieved by the successive application of this elementary transformation step as long as it can be applied, i.e. as long as there is a t-loop in the process.

The elementary transformation step has 2 possible outcomes: if the loop cannot be executed, because the earliest time-point the loop node can be started is after the threshold, then the loop is simply removed. Otherwise, the loop is replaced by an xor-split checking P, nested in its $True$ branch a t-split checking the temporal loop condition, and nested within its $True$ branch, a copy of the original loop.

Fig. 2 shows a graphical representation of the transformation step in the context of a node A preceding a temporal loop and a node X as immediate successor of the loop. Please note, that according to Definition 1 a loop has exactly one predecessor and exactly 1 successor. XS and XJ represent xor-split and xor-join, TS and TJ denote temporal split and -join. For the formal

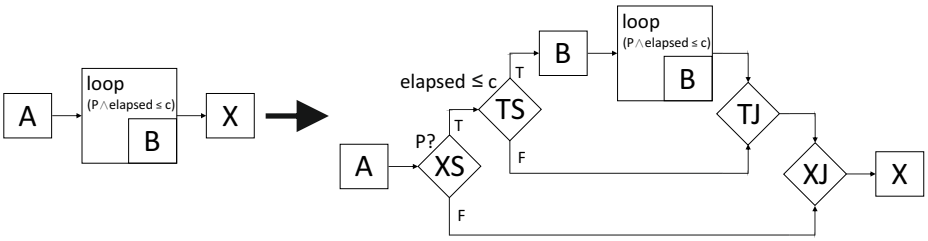

Fig. 2. Transformation step for loop unfolding.

definition of the loop unfold step we use the following shorthand: $(a, b, T) \in E$ denotes that b is the $True$ successor of a split node a.

Definition 5 (Transformation step). *We define a transformation step as follows: Let $P = (N, E, C, \omega)$ be a process. Let $n \in N$ with $tloop(n, P, c, B)$. $P' = (N', E', C', \omega) = T^1(P, n)$ is defined as:*

1. *If $c < n.e$, then n is deleted.*
   (a) $N' := N - \{n\}$;
   (b) $E' := E - \{(n, x) \in E, (y, n) \in E\} \cup \{(m, x) | (m, n) \in E, (n, x) \in E\}$
2. *If $c \geq n.e$, then n is unfolded as follows:*
   (a) $N' := N \cup \{XS, XJ, TS, TJ, B\}$:
   $XS.type = xor - split$, $XJ.type = xor - join$, $XS.cond = P$,
   $TS.type = t - split$, $TJ.type = t - join$, $B = n.B$
   (b) $E' := E - \{(n, x) \in E, (y, n) \in E\} \cup \{(y, XS) | (y, n) \in E\}$
   $\cup \{(XJ, x) | (n, x) \in E\} \cup \{(XS, TS, T), (XS, XJ, F\}$
   $\cup \{(TS, B, T), (TS, TJ, F\} \cup \{TJ, XJ), (B, n), (n, TJ)\}$ □

The rationale for the transformation step is as follows: if the body B of a t-loop n can be entered, it is extracted from the loop block and put in the true-branch of a nesting of xor-split and t-split, which evaluate the same boolean condition, resp. temporal condition of n. This realizes the same semantics as entering one iteration of the loop. A copy of the t-loop following B in the true-branch makes it possible to execute further loop iterations. By iteratively unfolding the t-loop into a nesting of xor-splits and t-splits we obtain a new process model temporally equivalent to the original one, but with all t-loops replaced with the conditional splits.

Definition 6 (Transformation). *We define the loop-unfolding transformation as follows: Let $P = (N, E, C, \omega)$ be a process. The loop unfolding transformation $T^*(P) = P' = (N', E', C', \omega)$ is defined as follows:*
If $\exists n \in N$ with $tloop(n, P, c, B)$: $P' = T^*(T^1(P, n))$. $P' = P$, otherwise.

Lemma 1. Let $P = (N, E, C, \omega)$ be a process. P and $T^*(P)$ have the same set of valid scenarios.

Proof. It is easy to see that for a $tloop(n, P, c, B)$, a process P and the result of a transformation step $T^1(P, n)$ has the same set of scenarios. The lemma then follows by induction. □

With the transformation for loop unfolding we defined here, we transformed a process with temporal loops to an equivalent process without temporal loops (but with temporal splits). For such process, an effective procedure for checking the dynamic controllability has been proposed in [8], which is based on mapping such processes to an equivalent Conditional Simple Temporal Network with Uncertainty and Decisions (CSTNUD) for which a procedure for checking dynamic controllability has been presented in [29].

3.3 Simple transformations

The transformation defined above generates a series of nested occurrences of the loop body B. For the checking of controllability, howeve,r we do not need all these occurrences of B, since there are no upper- or lower-bound constraints to the loop body admitted. For checking dynamic controllability all we need to know is essentially the possible start and end-times of the loop and thus any temporal restrictions and uncertainties for the adjacent nodes.

Therefore, we present here a much more compact transformation of a temporal loop which is not execution - equivalent to the original process but controllability equivalent.

We first present the simple transformation for temporally constrained loop without an additional predicate, i.e. a $tloop(n, True, c, B)$.

Definition 7 (Simple transformation).
Let $P = (N, E, C, \omega)$ be a process. Let $n \in N$ with $tloop(n, True, c, B)$. The result of the simple transformation $T^s(P, n) = P' = (N', E', C', \omega)$ is defined as:

1. $N' := N \cup \{TS, TJ, B'\}$,
   $tsplit(TS, c), TJ.type = tjoin$,
   $B'.type = activity$, $B'.d_{min} = 0, B'.d_{max} = B.d_{max}$
2. $E' := E \cup \{(TJ, x) | (n, x) \in E\}$
   $\cup\{(x, TS)\} - \{(x, n) \in E\}$
   $\cup\{(TS, TJ, F), (TS, B', T), (B', TJ)\}$
3. $C' := C \cup \{lbc(start, B'.s, c + 1)\}$ □

Fig. 3 shows graphically the simple transformation for the case of a t-loop having a condition with $P = True$. The temporal loop $tloop(n, True, c, B)$ is replaced by temporal split construct with t-split node TS and t-join node TJ. The node TS represents the first loop split node, TJ the last loop-join node. The temporal split node TS has the same temporal conditions as the loop $elapsed \leq c$. The $False$ branch of TS is empty, the $True$ branch contains a node B' which represents the part of the loop body of the last iteration, which is executed after the threshold c. The node B' is contingent with a minimum duration of 0 and

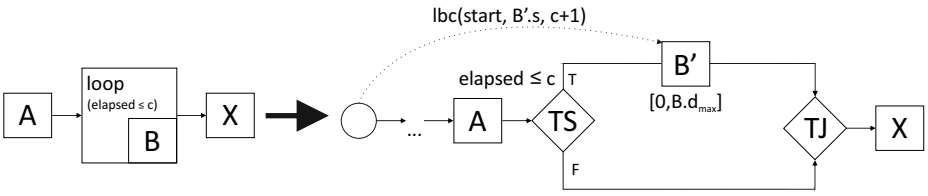

Fig. 3. Transformation for loop with condition with parameter $P = True$.

a maximum duration of $B.d_{max}$ and may not start before or at c expressed by the lower-bound constraint $lbc(start, B.s, c + 1)$.

The rationale for this transformation is follows:
(a) If the loop is executed at least once, i.e. the loop condition at least one evaluates to true, i.e. it is executed before the threshold c. This corresponds to executing TS in the transformed graph before the threshold c. In this case, the last iteration starts before or at time-point c and ends after c. The last iteration ends in the fastest case immediately after c, in the slowest case $B.d_{max}$ after c. This uncertainty is represented by the contingent duration of B'.

(b) If the process arrives at TS after the threshold c, the loop is never executed, and the whole loop is left and the process may continue immediately.

Lemma 2. Let $P = (N, E, C, \omega)$ be a process definition. Let $n \in N$ with $tloop(n, True, c, B)$ and let P' be the result of the simple transformation $T^s(P, n) = P' = (N', E', C', \omega)$. P and P' have equivalent sets of valid scenarios. □

Proof. We consider 2 scenarios as equivalent, if they are identical with the exception that the first loop-head execution is mapped to the t-split and the last loop-head execution is mapped to the t-join to which the t-loop is mapped to. Then the lemma follows immediately with the rationale detailed above. □

3.4 General Simple Transformation

In the case a temporal loop has an additional predicate P, i.e. $tloop(n, True, c, B)$ we further have to consider the case where a loop ends before the threshold c, because the predicate P evaluates to $False$.

Fig. 4 shows a graphically the general simple transformation for the case of a t-loop having a an additional condition P.

As in the simple transformation the t-loop is replaced by a t-split structure. While the $False$ branch of the t-split is the same, in the general case an Xor-structure (XS, XJ) is inserted into the $True$ branch of the t-split. The $True$ branch of XS represents the termination of the t-loop because of the temporal condition, and the $False$ branch represents the early termination, if P eventually evaluates to $False$. This additional branch contains a dummy activity D with duration 0, as the successor of the loop can be executed without delay. This branch, however can only be finished before or at the threshold c, which is expressed by the upper-bound constraint $ubc(start, D.s, c)$.

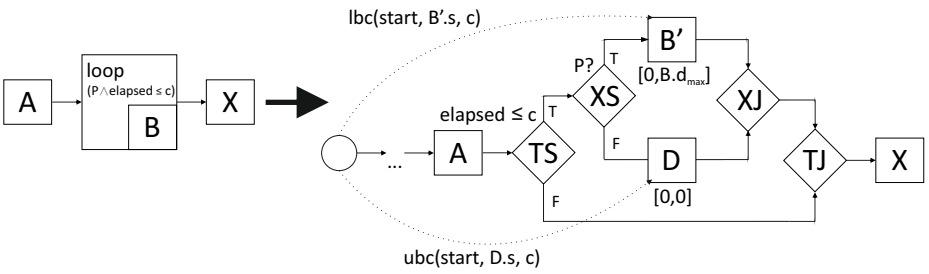

Fig. 4. Transformation for loop with condition with generic boolean parameter P.

Definition 8 (General simple transformation).
Let $P = (N, E, C, \omega)$ be a process. Let $n \in N$ with $tloop(n, P, c, B)$. The result of the simple transformation $T^g(P, n) = P' = (N', E', C', \omega) =$ is defined as:

1. $N' := N \cup \{XS, XJ, TS, TJ, B', D\}$,
   $XS.type = xsplit, XJ.type = xjoin,$
   $tsplit(TS, c), TJ.type = tjoin,$
   $B'.type = activity,\ B'.d_{min} = 0, B'.d_{max} = B.d_{max}$
   $D.type = activity,\ D.d_{min} = 0, D.d_{max} = 0$
2. $E' := E \cup \{(TJ, x) | (n, x) \in E\}$
   $\cup\{(x, TS)\} - \{(x, n) \in E\}$
   $\cup\{(TS, TJ, F), (TS, XS, T), (XS, D, F), (XS, B', T)\}$
   $\cup\ \{(B', XJ), (D, XJ), (XJ, TJ)\}$
3. $C' := C \cup \{lbc(start, B'.s, c + 1)\} \cup \{ubc(start, D.s, c)\}$ □

3.5 Checking Dynamic Controllability

Theorem 1. A process model P with temporal loops is dynamically controllable, if each loop body is temporally controllable and the process $\bar{P} = T^g(P)$ is dynamically controllable.

Proof 1. Proof sketch: The theorem follows from the observation that the process and the transformed process have equivalent sets of valid scenarios. □

With this result we now can reduce the checking of the dynamic controllability of processes with t-loops to checking the dynamic controllability of processes with t-splits but without t-loops. For these types of processes we already introduced a checking procedure [8].

4 Related Work

A general overview of time management for workflows, orchestrations, and business process management and its development over the past 20 years is given in [12, 5]. Early approaches checking temporal qualities of process definitions are

[21, 1, 11] with techniques rooted in network analysis, scheduling, or constraint networks. These techniques stimulated the development of more advanced networks, the consideration of interorganizational processes and for supporting temporal service level agreements for service compositions [4, 15]. Process mining [26] offers a different approach for deriving temporal qualities of process definitions. However, process mining requires a substantial amount of observed traces before temporal qualities of a process can be reliably calculated. In contrast to the approaches we focus on here, process mining is thus not applicable to new processes, or for changed processes.

We concentrate here on related work with respect to formulating temporal constraints involving control structures, checking correctness and other temporal properties of process definitions with such temporal constraints.

BPMN [22] allows to use time information in conditions, however the specification is on a rather abstract generic level. More precise expression definition is offered by some system vendors, e.g. Oracle BPM [23]. [13] extend BPMN with graphical elements for temporal concepts with defined semantics, mainly to express inter-task constraints and more complex timed triggers. However, all these approaches do not consider the verification of defined processes.

[20, 18] gathered and unified different notions for representing temporal constraints for process models. Temporal constraints are classified in a set of 10 different time patterns. Our work here addresses the patterns TP6: Time-based Restrictions and TP8: Time-dependent Variability as they use time information in conditions of XOR-gateways or loops, respectively. Time pattern TP8 allows to execute different XOR branches where the XOR condition depends on temporal information. A t-split [24] is a specialization of time pattern TP8 since it allows to execute different branches depending on the elapsed time since process start. Time pattern TP8 alone, however, does not address the controllability aspects, which were addresses in [8].

Controllability and dynamic controllability are the most elaborated notions of correctness of temporally constrained process definitions. Recently, controllability and dynamic controllability for more expressive network models like *Conditional Simple Temporal Network with Uncertainty (CSTNU)* provide new sophisticated means to check the properties of temporally constrained process definitions [6]. However, they are not able to express the kind of temporal control structures we propose here. To bridge the gap between *CSTNU*s and control structures in which decisions are taken by a process controller and not simply observed, [28] introduced *CSTNUD*s, to capture control structures in which controllers may make decisions about the continuation of a process and introduced a dynamic controllability checking approach based on Timed Game Automata (TGA). Although this constitutes a considerable step forward in the expressiveness within the framework of temporal networks, the approach does not take into account *how* decisions are made, which we address here with regard to decisions depending on the temporal status of the ongoing process execution. In particular, we consider that controllers may influence decisions by deciding *when* to execute temporal conditions.

Temporal control structures for processes have been introduced in [24] for the first time. In [8] a procedure for checking the dynamic controllability of processes with temporal splits has been proposed. This paper is a continuation of this work.

5 Conclusions

Dynamic Controllability has been established as the most suitable criterion for the temporal correctness of process definitions. However, so far only procedures for acyclic processes have been developed, as processes with loops are by definition not dynamically controllable. The introduction of temporally constrained loops in process models opens interesting new possibilities for representing temporal constraints in process models with loops. Here we showed how dynamic controllability of processes with temporal loops can effectively be checked at design time. The proposed procedure relies on a series of transformations of process definitions with the final target of mapping a process definition to simple temporal networks with uncertainty and decisions (CSTNUD), for which procedures for checking dynamic controllability have been proposed. For practical implementations, however, a process model can be directly translated to an equivalent CSTNUD, short-cutting the series of transformations.

References

1. C. Bettini, X. Wang, and S. Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002.
2. R. Breu, S. Dustdar, et al. Towards living inter-organizational processes. In *2013 IEEE 15th Conference on Business Informatics*, pages 363–366. IEEE, 2013.
3. J. Cardoso, A. Sheth, and J. Miller. Workflow quality of service. In *Enterprise Inter-and Intra-Organizational Integration*, pages 303–311. Springer, 2003.
4. J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1(3):281–308, 2004.
5. S. Cheikhrouhou, S. Kallel, N. Guermouche, and M. Jmaiel. The temporal perspective in business process modeling: a survey and research challenges. *Service Oriented Computing and Applications*, 9(1):75–85, 2015.
6. C. Combi, L. Hunsberger, and R. Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty - revisited. In *Agents and Artificial Intelligence*, pages 314–331. Springer Berlin Heidelberg, 2014.
7. C. Combi and R. Posenato. Controllability in temporal conceptual workflow schemata. In *Business Process Management*, pages 64–79. Springer, 2009.
8. J. Eder, M. Franceschetti, and J. Köpke. Controllability of orchestrations with temporal SLA: Encoding temporal XOR in CSTNUD. In *Int. Conf. on Information Integration and Web-based Applications & Services*, pages 234–242. ACM, 2018.
9. J. Eder, W. Gruber, and E. Panagos. Temporal modeling of workflows with conditional execution paths. In *Database and Expert Systems Applications*, pages 243–253. Springer, 2000.

10. J. Eder, W. Gruber, and H. Pichler. Transforming workflow graphs. In *Interoperability of Enterprise Software and Applications*, pages 203–214. Springer, 2006.
11. J. Eder, E. Panagos, and M. Rabinovich. Time constraints in workflow systems. In *Advanced information systems engineering*, pages 286–300. Springer, 1999.
12. J. Eder, E. Panagos, and M. Rabinovich. Workflow time management revisited. In J. Bubenko, J. Krogstie, O. Pastor, B. Pernici, C. Rolland, and A. Sølvberg, editors, *Seminal Contributions to Information Systems Engineering*, pages 207–213. Springer Berlin Heidelberg, 2013.
13. D. Gagne and A. Trudel. Time-bpmn. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 361–367. IEEE, 2009.
14. M. Gillmann, G. Weikum, and W. Wonner. Workflow management with service quality guarantees. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 228–239. ACM, 2002.
15. N. Guermouche and C. Godart. Timed model checking based approach for web services analysis. In *ICWS*, pages 213–221. IEEE, 2009.
16. L. Hunsberger, R. Posenato, and C. Combi. The dynamic controllability of conditional stns with uncertainty. *arXiv preprint arXiv:1212.2005*, 2012.
17. A. Lanz, R. Posenato, C. Combi, and M. Reichert. Controllability of time-aware processes at run time. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, pages 39–56. Springer, 2013.
18. A. Lanz, M. Reichert, and B. Weber. Process time patterns: A formal foundation. *Information Systems*, 57:38–68, 2016.
19. A. Lanz, B. Weber, and M. Reichert. Workflow time patterns for process-aware information systems. In *Enterprise, Business-Process and Information Systems Modeling*, pages 94–107. Springer, 2010.
20. A. Lanz, B. Weber, and M. Reichert. Time patterns for process-aware information systems. *Requir. Eng.*, 19(2):113–141, 2014.
21. O. Marjanovic and M. Orlowska. On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems*, 1(2):157–192, 1999.
22. ObjectManagementGroup. Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/BPMN/2.0, 2011.
23. OracleFusion. Adding delays, deadlines, and time based cycles to your process. https://docs.oracle.com/middleware/1221/bpm/bpm-develop/GUID-1684501B-AA69-4982-B3BC-9B05E33B4EB3.htm#BPMPD579, 2019. Accessed: 2019-08-08.
24. H. Pichler, J. Eder, and M. Ciglic. Modelling processes with time-dependent control structures. In *Int. Conf. on Conceptual Modeling*, pages 50–58. Springer, 2017.
25. H. Pichler, M. Wenger, and J. Eder. Composing time-aware web service orchestrations. In *Advanced Information Systems Engineering*, pages 349–363. Springer, 2009.
26. W. van der Aalst, M. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011.
27. W. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.
28. M. Zavatteri. Conditional simple temporal networks with uncertainty and decisions. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 90. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
29. M. Zavatteri. *Temporal and Resource Controllability of Workflows Under Uncertainty*. PhD thesis, Universita degli Studi di Verona, 2018.