A Symbolic Time Constraint Propagation Mechanism Proposal for Workflow Nets

Lorena Rodrigues Bruno and Stéphane Julia
Federal University of Uberlândia, Uberlândia-MG, Brazil

Keywords: Time Petri Net, Symbolic Expression, Workflow Net, Linear Logic, Forward Propagation, Backward Propagation.

Abstract: The model of a Workflow Management System should describe the time constraints of resources over the activities of the corresponding business process. In general, typical temporal phenomena include activity execution delays, limits to the occurrence of valid intervals over the activities, limits to valid intervals over resources (limit to resources life cycle), limits to duration of process execution, time distance between two activities, etc. In this study, a Workflow net model incremented with time intervals to describe the duration of activities and waiting times is presented. To define the execution of activities minimum and maximum intervals, a time constraint propagation mechanism based on the sequent calculus of Linear Logic and on symbolic dates is proposed.

1 INTRODUCTION

Workflow Management Systems are systems that manage the execution of Workflow processes (Murata, 1989). A Workflow process represents the sequence of activities that have to be executed by members of the same working group, while respecting some conditions that determine their order of execution, in order to treat specific cases and to reach a well-defined goal.

Petri nets (Murata, 1989) are very well adapted to model Real Time Systems, as these allow for a good representation of conflict situations, shared resources, synchronous and asynchronous communication, precedence constraints and explicit quantitative time constraints in the time Petri net case. Petri nets can also be an efficient tool for the modeling and analysis of Workflow Management Systems, as these have a graphic representation, are easy to learn, can be used as a communication language between specialists from different areas, allow static and dynamic description of systems to be represented, and also have a mathematical formalism that enables the use of important analysis methods (Murata, 1989). The Petri nets that model Workflow processes are defined in (Van Der Aalst et al., 2004) and are called Workflow nets.

Time management of a Workflow process is an important aspect in the study of processes due to the fact that there exist many types of time restrictions in business processes. A study of Workflow time management is concentrated mainly on the planning of Workflow time execution, on the estimation of activity durations, on avoiding time restriction violations in activities or processes, and on the treatment of time restriction violation exceptions.

In the real world, due to dynamic features of resources and activities in business processes, the majority of information is uncertain and cannot be precisely described. In this study, a Workflow net model with time interval constraints, defined by minimum and maximum bounds for each activity duration and waiting time between activities, is presented. In order to compute minimum and maximum bounds corresponding to the beginning dates of the activities, two different time constraint propagation mechanisms are proposed: one to find the earliest start dates of the process activities, and the other to find the latest start dates of process activities.

The proposed work presents an approach based on Linear Logic to prove the Soundness correctness criterion defined for Workflow nets and, based on the resulting proof tree, computes symbolic formulas based on $(max, +)$ or $(min, -)$ operators that express the possible beginning dates of process activities. Such symbolic expressions will then be used to define the numerical time constraints associated to specific cases treated by the Workflow process and will be consid-

Bruno, L. and Julia, S.
A Symbolic Time Constraint Propagation Mechanism Proposal for Workflow Nets.
DOI: 10.5220/0011064400003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 1*, pages 537-544
ISBN: 978-989-758-569-2; ISSN: 2184-4992

ered in the Planning/Scheduling problem of resources associated with the activities linked to the process.

2 TIME WORKFLOW NET

2.1 Workflow Net

A Petri net that models a Workflow process is called a Workflow net (Van Der Aalst et al., 2004). A Workflow net satisfies the following properties:

- It has only one source place named *Start* and only one sink place named *End*. These are special places, such that the place *Start* has only outgoing arcs and the place *End* has only incoming arcs.
- A token in *Start* represents a case that needs to be handled and a token in *End* represents a case that has been handled.
- Every task t (transition) and condition p (place) should be in a path from place *Start* to place *End*.

2.2 Process

A process defines which tasks need to be executed and in which order (Van Der Aalst et al., 2004). Modeling a Workflow process in terms of a Workflow net is rather straightforward: transitions are active components and models the tasks, places are passive components and model conditions (pre and post), and tokens model cases (Van Der Aalst et al., 2004).

In order to illustrate the mapping of a process into a Workflow net, the process for handling complaints, shown in (Van Der Aalst et al., 2004) can be understood as follows: an incoming complaint is first recorded. Then the client who has complained along with the department affected by the complaint are contacted. The client is approached for more information. The department is informed of the complaint and may be asked for its initial reaction. These two tasks may be performed in parallel, i.e. simultaneously or in any order. After that, data is gathered and a decision is made. Depending upon the decision, either a compensation payment is made or a letter is sent. Finally, the complaint is filed. In Fig. 1, a Workflow net that correctly model this process is shown.

2.3 Time Petri Net

Time and Timed Petri nets can associate time stamps to places (p-time and p-timed Petri net) or transitions (t-time and t-timed Petri nets). In the p-timed

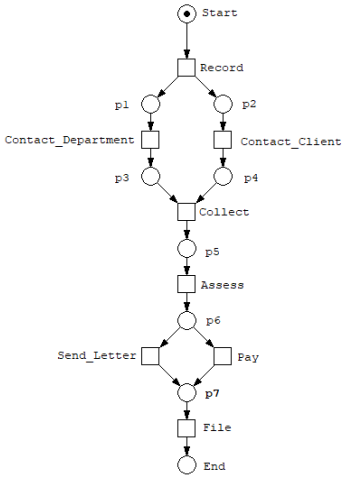

Figure 1: Handle Complaint Process.

Petri net case, according to (Sifakis, 1979), time constraints are represented by durations (positive rational numbers) associated with places. In the t-timed Petri net case, according to (Ramamoorthy and Ho, 1980), time constraints are represented by durations (positive real numbers) associated with transitions. In the t-time Petri net case, time constraints are represented by an interval $[\delta_{min}, \delta_{max}]$ associated with each transition; the time interval associated with a specific transition corresponds to an imprecise enabling duration, according to (Merlin, 1974) and (Menasche, 1982). For example, the interval $[8,12]$ associated with a transition indicates that the transition will be fired at least eight time units after it has been enabled and, at the most, twelve time units after its enabling instant.

3 LINEAR LOGIC

Linear Logic (Girard, 1987) was proposed by Jean-Yves Girard in 1987. In Linear Logic, the propositions are considered as resources which are consumed and produced in each change of state, which is different from classic logic where propositions have Boolean values (true or false) (Pradin-Chézalviel et al., 1999). The Linear Logic introduces seven new connectors divided into three groups, but in this paper, just two of these connectives will be used: *times*, denoted by $\otimes$ and *linear implies*, denoted by $\multimap$. The first symbol represents the simultaneous availability of resources; for example, $A \otimes B$ represents simultaneous availability of resources A and B. The second symbol represents a possible change of state; for instance $A \multimap B$ means that B is produced when A is consumed.

The translation of a Petri net into formulas of Linear Logic is performed as presented in (Riviere et al., 2001). A marking M is a monomial in $\otimes$, which is represented by $M = A1 \otimes A2 \otimes ... \otimes Ak$ in which Ai are place names. In Fig.1, the initial marking is just *Start* because of the token in place *Start*. A transition is an expression of the form $M1 \multimap M2$ where $M1$ and $M2$ are markings. For instance, the transition *Record* of the Workflow net in Fig.1 can be represented as $Record = Start \multimap p1 \otimes p2$.

A sequent represents the triggering of a transition (or a sequence of possible transitions). The sequent $M, t_i \vdash M'$ represents a scenario where M is the initial marking, M' is the final marking, and t_i is a set of non-ordered transitions. To prove the correctness of a sequent, a proof tree should be built by applying the rules of sequent calculus. According to (Girault et al., 1997) the proof of a sequent in linear logic is equivalent to the corresponding reachability problem in the Petri net theory.

In this paper, only some of the Linear Logic rules will be considered. These rules will be used to build the proof trees in the context of the reachability problem of the Petri net theory. For such, F, G and H are considered as formulas and, Γ and Δ are blocks of formulas. The following rules will be used in this paper:

- The $\multimap_L$ rule, $\frac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H} \multimap_L$ expresses a transition firing and generates two sequents. The left sequent represents the tokens consumed by the transition firing and the right sequent represents the subsequent that still needs to be proved.
- The $\otimes_L$ rule, $\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H} \otimes_L$, transforms a marking into an atom list.
- The $\otimes R$ rule, $\frac{\Gamma \vdash F \quad \Delta \vdash G}{\Delta, \Gamma \vdash F \otimes G} \otimes_R$, transforms a sequent such as $A, B \vdash A \otimes B$ into two identity sequents $A \vdash A$ and $B \vdash B$.

A Linear Logic proof tree is read from bottom-up. The proof stops when the identity sequent $End \vdash End$ is produced, when there is no longer any rule that can be applied, or when all the leaves of the proof tree are identity sequent.

4 TIME CONSTRAINT PROPAGATION MECHANISM

Typically, the time required to execute an activity in a Workflow process is non-deterministic. According to (Riviere et al., 2001), explicit time constraints, which exist in systems with real-time characteristics, can be formally specified using a static time interval associated to each task (transition) of the model. The dynamic behavior of the corresponding time Petri net thus depends on the marking of the network, as well as on the tokens temporal situation that is given by a visibility interval (dos Santos Soares et al., 2008). A visibility interval $[(\delta_p)_{min}, (\delta_p)_{max}]$ associated with a token in a place p of a time Petri net specified the minimum date $(\delta_p)_{min}$ at which a token is available in p to trigger an output transition of p (earliest start date of an activity), and the maximum date after which the token becomes unavailable (dead) and cannot be used to trigger any transition (latest start date of the corresponding activity).

In a Workflow Management System, the visibility interval depends on a global clock connected to the entire net which calculates the passage of time from date $\delta = 0$, which corresponds to the start of system operations. In particular, the existing waiting time between sequential activities will be represented by a visibility interval, for which the minimum and maximum bounds will depend on the earliest and latest delivery of a case process by the Workflow net. Through knowledge of the beginning date and the maximum duration of a case, it should be possible to calculate the visibility intervals associated with the tokens in the waiting places.

The calculus of the visibility intervals associated with the tokens in the waiting places will be realized in this proposal by considering the different kinds of routes that can exist in a Workflow process, and which can be expressed through the sequents of Linear Logic. An iterative route can be replaced by a global activity as shown in (Soares Passos and Julia, 2016) and will not be considered in this work.

The constraint propagation technique proposed in this article is based on two different approaches: a forward reasoning to produce the minimum bounds of the visibility intervals (earliest dates to initiate the activities of the process), and a backward reasoning to produce the maximum bounds of the visibility intervals (latest dates to initiate the activities of the process). The produced visibility intervals will be given through symbolic date expressions instead of numerical ones. The main advantage of using symbolic dates is that once the expressions have been calculated for a specific Workflow process, these can be used for any case that will be handled by the same Workflow process.

In this paper, D_i will denote a date and d_i a duration associated with a transition (t_i) firing. A pair (D_p, D_c) will be associated with each token of the proof tree and represents respectively the production and consumption date of a token. As transition firings

are instantaneous ones in t-time nets (there is no token reservation), the following definition can be produced:

Definition 1. *The production date D_p of a token is equal to the firing date of the transition which has produced it and the consumption date D_c is equal to the firing date of the transition which has consumed it.*

For each triggered activity on the t-Time Workflow net analyzed, the dates of production D_p and consumption D_c of the atoms that represents the preconditions of the activity should be extracted. The date of production of the atom D_p, corresponds to the start execution of the activity associated with the transition and the date of consumption, D_c, corresponds to its conclusion. Thus, an interval of dates $[D_p, D_c]$ will be generated, and the resource that will handle the referred activity will be able to be requested within the calculated interval.

Since production and consumption dates depend on d_i (duration associated to a transition firing) and take there values within time intervals $\Delta_i = [\delta_{imin}, \delta_{imax}]$, many tasks execution intervals can be considered according to a strategic planning. For instance, the execution interval $I_{Exec} = [D_{Pmin}, D_{Cmax}]$ considers that the allocation of resources to handle the task can occur between the earliest beginning and the latest conclusion of this activity. To illustrate the proposal of this work, the Workflow net shown in Fig. 1 is considered.

4.1 Forward Propagation

The forward propagation consider the Workflow net in its normal flow, which is a case represented by a token that begins in place *Start* and finishes in place *End*, while following the ordered places and transitions.

The date computation in the canonical proof for the Workflow net is then the following one:

- Assign a production date D_i to the initial token in place *Start* (initial marking of the t-Time Petri net model).
- For each $\multimap L$, compute the firing date of the corresponding transition: it is equal to the maximum of the production dates of the consumed atoms, increased by the enabling duration d_j associated with the considered transition.
- Update all the temporal stamps of the atoms which have been consumed and produced.

As in t-Time Petri net model (Merlin, 1974), an enabling duration d_i takes its values from within a time interval $\Delta_i = [\delta_{imin}, \delta_{imax}]$. The computed symbolic dates depend on d_i and its domains will depend on time interval.

Considering Fig. 2, two linear sequents must be proved, where each one represents a different scenario. Such linear sequents can be obtained automatically applying a t-invariant algorithm (linear equation solving) as proposed in (Oliveira and Julia, 2020).

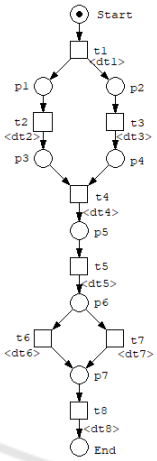

Figure 2: t-Time Workflow net - Forward Propagation.

For scenario S_{c1}, the following sequent has to be proved:

$$Start, t1, t2, t3, t4, t5, t6, t8 \vdash End \quad (1)$$

For scenario S_{c2}, the following sequent has to be proved:

$$Start, t1, t2, t3, t4, t5, t7, t8 \vdash End \quad (2)$$

The transitions of the Workflow net are represented by the following formulas of Linear Logic:
$t_1 = Start \multimap p1 \otimes p2$; $t_2 = p1 \multimap p3$; $t_3 = p2 \multimap p4$; $t_4 = p3 \otimes p4 \multimap p5$; $t_5 = p5 \multimap p6$; $t_6 = p6 \multimap p7$; $t_7 = p6 \multimap p7$; $t_8 = p7 \multimap End$.

Considering $Seq = D_S + d_{t1} + max\{d_{t2}, d_{t3}\} + d_{t4}$, the proof tree with dates corresponding to scenario S_{c1} is represented in Equation 3. The proof tree with dates for scenario S_{c2}, represented by the Equation 4, is similar to the one for S_{c1}, the only difference is that in the first lines of the proof, instead of d_{t6} (duration of the transition firing $t6$), it is d_{t7} (duration of the transition firing $t7$) which is considered. The entire proofs can be found in (Soares Passos and Julia, 2016).

$$\dfrac{\dfrac{p7(Seq+d_{t5}+d_{t6},Seq+d_{t5}+d_{t6}+d_{t8})\vdash p7 \quad End(Seq+d_{t5}+d_{t6}+d_{t8},.)\vdash End}{p6(Seq+d_{t5},Seq+d_{t5}+d_{t6})\vdash p6 \quad p7(Seq+d_{t5}+d_{t6},.),p7\multimap End\vdash End}\multimap_L}{\ldots}\multimap_L \tag{3}$$

$$Start(D_S,.),Start\multimap p1\otimes p2,t2,t3,t4,t5,t6,t8\vdash End$$

$$\dfrac{\dfrac{p7(Seq+d_{t5}+d_{t6},Seq+d_{t5}+d_{t7}+d_{t8})\vdash p7 \quad End(Seq+d_{t5}+d_{t7}+d_{t8},.)\vdash End}{p6(Seq+d_{t5},Seq+d_{t5}+d_{t7})\vdash p6 \quad p7(Seq+d_{t5}+d_{t7},.),p7\multimap End\vdash End}\multimap_L}{\ldots}\multimap_L \tag{4}$$

$$Start(D_S,.),Start\multimap p1\otimes p2,t2,t3,t4,t5,t7,t8\vdash End$$

The symbolic dates obtained from the proof tree of forward propagation mechanism are shown in the Table 1.

4.2 Backward Propagation

From the proof tree obtained by the forward propagation model, considering the start date D_S, it is possible to determine part of the visibility intervals to trigger the tasks (represented by a token in P_i). As in (Soares Passos and Julia, 2016), the forward propagation technique allows us to find the minimum bounds of the visibility intervals.

Backward propagation is used to calculate the maximum bounds of the same visibility interval. To accomplish this goal, the approach is based on an inverted model as those presented in (Khalfhoui et al., 2002) and (Oliveira and Julia, 2020). In (Khalfhoui et al., 2002), a backward reasoning was applied on a Petri net model with all the arcs reversed in order to identify feared scenarios in mechatronic systems. Paper (Oliveira and Julia, 2020) also proposed a backward reasoning applied on a Workflow net with all arcs inverted in order to diagnose the causes of deadlock situations in Service-Oriented Architectures.

For the backward propagation technique proposed in this article, the arcs of the Workflow net are then inverted in such a way that the initial marking starts in place *End* and reaches the *Start* place at the end of the execution of the inverted Workflow net, as illustrated in Fig. 3.

The proof of a sequent on the inverted Workflow net is obtained using the same proof algorithm as in (Soares Passos and Julia, 2016) for the applied rules of linear logic, but with different computation of temporal stamps. We denote D_i for dates and d_i for durations.

When the consumption date is not fully computed, some pairs are denoted $(D_p,.)$ will appear in the proof tree. Identity sequents generated on the left side of the tree will produce pair stamps (D_p,D_c) completely defined. These final leaves stamps will correspond to the main temporal results (maximum bound of visibility intervals) of the backward propagation mechanism.

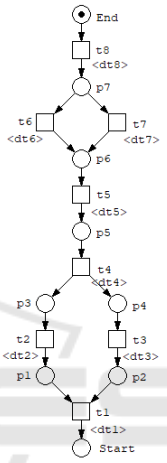

Figure 3: t-Time Workflow net - Backward Propagation.

The date computation in the canonical proof tree for the inverted Workflow net is then the following one:

- Assign a production date D_i to the initial token in place *End* (initial marking of the inverted net).
- For each $\multimap L$, compute the firing date of the corresponding transition: it is equal to the minimum of the production dates of the consumed atoms, decreased by the enabling duration d_j associated with the considered transition.
- Update all the temporal stamps of the atoms which have been consumed and produced.

The rest of the process to prove the sequent is the same as the one presented in the forward propagation mechanism.

Considering the Workflow net in Fig. 3, the proof of correctness on this inverted model corresponds to two different scenarios given by the following sequents:

Table 1: Symbolic date intervals for forward propagation in scenario S_{c1}.

Tasks	Production Date	Consumption Date
Start	D_s	$D_s + d_{t1}$
$p1$	$D_s + d_{t1}$	$D_s + d_{t1} + d_{t2}$
$p2$	$D_s + d_{t1}$	$D_s + d_{t1} + d_{t3}$
$p3$	$D_s + d_{t1} + d_{t2}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$
$p4$	$D_s + d_{t1} + d_{t3}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$
$p5$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5}$
$p6$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6}$
$p7$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8}$
End	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8}$	D_{End}

For scenario S_{c1}:

$$End, t8, t6, t5, t4, t3, t2, t1 \vdash Start \quad (5)$$

and for scenario S_{c2}:

$$End, t8, t7, t5, t4, t3, t2, t1 \vdash Start \quad (6)$$

The transitions of the Workflow net are represented by the following formulas of Linear Logic: $t_8 = End \multimap p7$; $t_7 = p7 \multimap p6$; $t_6 = p7 \multimap p6$; $t_5 = p6 \multimap p5$; $t_4 = p5 \multimap p3 \otimes p4$; $t_3 = p4 \multimap p2$; $t_2 = p3 \multimap p1$; $t_1 = p1 \otimes p2 \multimap Start$.

Considering $Seq_1 = D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$ the proof tree with dates for S_{c1} is the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p1(Seq_1-d_{t2},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p1 \quad p2(Seq_1-d_{t3},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p2}{p1(Seq_1-d_{t2},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1}),p2(Seq_1-d_{t3},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p1\otimes p2}\otimes_R \quad Start(Seq_1-min\{d_{t2},d_{t3}\}-d_{t1},.)\vdash Start}{p3(Seq_1,Seq_1-d_{t2})\vdash p3 \quad p1(Seq_1-d_{t2},.),p2(Seq_1-d_{t3},.),p1\otimes p2\multimap Start\vdash Start}\multimap_L}{p4(Seq_1,Seq_1-d_{t3})\vdash p4 \quad p3(Seq_1,.),p2(Seq_1-d_{t3},.),p3\multimap p1,t1\vdash Start}\multimap_L}{p4(Seq_1,.),p3(Seq_1,.),p4\multimap p2,t2,t1\vdash Start}\otimes_L}{p5(D_{End}-d_{t8}-d_{t6}-d_{t5},D_{End}-d_{t8}-d_{t6}-d_{t5}-d_{t4})\vdash p5 \quad p3\otimes p4,p4\multimap p2,t2,t1\vdash Start}\multimap_L \quad (7)$$

$$\dfrac{p6(D_{End}-d_{t8}-d_{t6},D_{End}-d_{t8}-d_{t6}-d_{t5})\vdash p6 \quad p5(D_{End}-d_{t8}-d_{t6}-d_{t5},.),p5\multimap p3\otimes p4,t3,t2,t1\vdash Start}{\multimap_L}$$

$$\dfrac{p7(D_{End}-d_{t8},D_{End}-d_{t8}-d_{t6})\vdash p7 \quad p6(D_{End}-d_{t8}-d_{t6},.),p6\multimap p5,t4,t3,t2,t1\vdash Start}{\multimap_L}$$

$$\dfrac{End(D_{End},D_{End}-d_{t8})\vdash End \quad p7(D_{End}-d_{t8},.),p7\multimap p6,t5,t4,t3,t2,t1\vdash Start}{\multimap_L}$$

$$End(D_{End},.),End\multimap p7,t6,t5,t4,t3,t2,t1\vdash Start$$

Considering $Seq_2 = D_{End} - d_{t8} - d_{t7} - d_{t5} - d_{t4}$, the proof tree with dates for scenario S_{c2} is similar to the proof tree of S_{c1}, the only difference is that instead of d_{t6} (duration of the transition firing $t6$), the duration considered is d_{t7} (duration of the transition firing $t7$).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p1(Seq_2-d_{t2},Seq_2-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p1 \quad p2(Seq_2-d_{t3},Seq_2-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p2}{p1(Seq_2-d_{t2},Seq_2-min\{d_{t2},d_{t3}\}-d_{t1}),p2(Seq_2-d_{t3},Seq_2-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p1\otimes p2}\otimes_R \quad Start(Seq_2-min\{d_{t2},d_{t3}\}-d_{t1},.)\vdash Start}{\multimap_L}}{\ldots}}{p5(D_{End}-d_{t8}-d_{t7}-d_{t5},D_{End}-d_{t8}-d_{t7}-d_{t5}-d_{t4})\vdash p5 \quad p3\otimes p4,p4\multimap p2,t2,t1\vdash Start}\multimap_L}{p6(D_{End}-d_{t8}-d_{t7},D_{End}-d_{t8}-d_{t7}-d_{t5})\vdash p6 \quad p5(D_{End}-d_{t8}-d_{t7}-d_{t5},.),p5\multimap p3\otimes p4,t3,t2,t1\vdash Start}\multimap_L \quad (8)$$

$$\dfrac{p7(D_{End}-d_{t8},D_{End}-d_{t8}-d_{t7})\vdash p7 \quad p6(D_{End}-d_{t8}-d_{t7},.),p6\multimap p5,t4,t3,t2,t1\vdash Start}{\multimap_L}$$

$$\dfrac{End(D_{End},D_{End}-d_{t8})\vdash End \quad p7(D_{End}-d_{t8},.),p7\multimap p6,t5,t4,t3,t2,t1\vdash Start}{\multimap_L}$$

$$End(D_{End},.),End\multimap p7,t7,t5,t4,t3,t2,t1\vdash Start$$

The symbolic dates obtained from the proof tree of backward propagation mechanism are shown in the

Table 2: Symbolic date intervals for backward propagation in scenario S_{c1}.

Tasks	Production Date	Consumption Date
End	D_{End}	$D_{End} - d_{t8}$
$p7$	$D_{End} - d_{t8}$	$D_{End} - d_{t8} - d_{t6}$
$p6$	$D_{End} - d_{t8} - d_{t6}$	$D_{End} - d_{t8} - d_{t6} - d_{t5}$
$p5$	$D_{End} - d_{t8} - d_{t6} - d_{t5}$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$
$p4$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t3}$
$p3$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t2}$
$p2$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t3}$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$
$p1$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t2}$	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$
Start	$D_{End} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$	D_s

Table 2.

Finally, symbolic visibility intervals can be produced for the tokens in each places of the Workflow net of Fig. 2. The minimum bounds correspond to the production dates D_p of Table 1, and the maximum bounds correspond to the production dates of Table 2.

5 NUMERICAL RESULTS FOR RESOURCE PLANNING

Considering the fact that the process in Fig. 1 starts at date 0 (i.e. $D_s = 0$), the maximum time for it to complete is 105, and the operation duration are the ones presented in Table 3. Visibility intervals for task execution can be calculated by replacing the symbolic visibility production dates presented on Tables 1 and 2 with the numerical values of Table 3. The minimum duration of operation (minimum bound of operation duration intervals in Table 3) are used to replace the symbolic production date values produced by the forward mechanism (Table 1), and the maximum duration of operation (maximum bounds of operation duration intervals in Table 3) are used to replace the symbolic production date values produced by the backward mechanism (Table 2).

Table 3: Transition duration interval.

Transition	Imprecise Duration
t1	[5,10]
t2	[20,30]
t3	[25,35]
t4	[0,0]
t5	[15,25]
t6	[5,15]
t7	[20,30]
t8	[0,0]

We can then calculate the visibility intervals by considering the production date values given in Table 1 for the forward propagation, as the minimum bounds of the visibility interval, because it simulates the normal task propagation flow of the process, and the production date values given in Table 2 for the backward propagation as the maximum bounds of the visibility interval, because it simulates the task propagation inverse flow. The result is presented in Table 4.

As it can be seen in Table 4, considering the latest date of completion of the process at date 0 (start date of the process) $+105$ (maximum duration of the process) $= 105$, and the symbolic formulas produced by the backtrack mechanism, it is possible to reach the place *Start* considering the inverted process at date $+20$ for scenario 1, and at date $+5$ for scenario 2, which means that if the initial marking in place *Start* at the beginning of the process is trigger with a delay of five time units, even so the task can be finished on time, no matter the scenario in order to end the process at the latest date 105 (maximum duration for process completion).

In order to validate this new approach, it can be observed that the visibility intervals produced in Table 4 are the same as the ones presented in (dos Santos Soares et al., 2008) where the visibility intervals were produced for the same t-time Workflow net using a pure numerical approach based on the graph of the corresponding Petri net model.

Table 4: Numerical visibility intervals for task execution.

Tasks	S_{c1}		S_{c2}	
	Min	Max	Min	Max
Start	0	20	0	5
p1	5	35	5	20
p2	5	30	5	15
p3	25	65	25	50
p4	30	65	30	50
p5	30	65	30	50
p6	45	90	45	75
p7	50	105	65	105
End	50	105	65	105

6 CONCLUSIONS

In this study, a Workflow net model incremented with symbolic date intervals for describing activity durations and waiting times was presented. In order to produce the minimum and maximum intervals for the execution of activities, two constraint propagation mechanisms based on the sequent calculus of Linear Logic were proposed. The first is a forward mechanism that uses (max,+) operators to produce formulas that indicate the earliest dates for the beginning of the activities of the cases treated by the Workflow process. The second is a backward mechanism that uses (min,−) operators to produce formulas that indicate the latest dates for the beginning the activities of those cases dealt with by the Workflow net process. The particular case of the backward mechanism is to prove the Soundness property of the Workflow net considering an inverted model with all arcs reversed. Due to the sequent calculus being based on such a model, this corresponds to a kind of go back in time each time a transition is fired, thus the time is decreased on the corresponding formula.

The computation of symbolic dates for the execution of each activity mapped into a time Workflow net, using the proof trees of Linear Logic, allows the planning of the utilization of the resources. These are the resources involved in the activities of the Workflow process for any case handled by the corresponding Workflow process, since the computed dates are symbolic instead of numerical. This kind of reuse is not provided when an approach based on a graph-oriented method (as the conventional ones) is considered.

As a future study, the authors intend to combine the time constraint propagation mechanisms with a formal definition of a resource allocation mechanism, as that presented in (Medeiros and Julia, 2017), and propose a conflict resolution mechanism with the aim of calculating a sequence of activities that respects the disjunctive constraints (resource allocation mechanisms) as well as the time constraints (date intervals).

ACKNOWLEDGEMENTS

The authors would like to thank FAPEMIG, CNPq and CAPES for the financial support provided.

REFERENCES

dos Santos Soares, M., Julia, S., and Vrancken, J. (2008). Real-time scheduling of batch systems using petri nets and linear logic. *Journal of Systems and Software*, 81(11):1983–1996.

Girard, J.-Y. (1987). Linear logic. *Theoretical computer science*, 50(1):1–101.

Girault, F., Pradier-Chezalviel, B., and Valette, R. (1997). A logic for petri nets. *Journal européen des systèmes automatisés*, 31(3):525–542.

Khalfhoui, S., Demmou, H., Guilhem, E., and Valette, R. (2002). An algorithm for deriving critical scenarios in mechatronic systems. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 6–pp. IEEE.

Medeiros, F. F. and Julia, S. (2017). Constraint analysis based on energetic reasoning applied to the problem of real time scheduling of workflow management systems. In *ICEIS (3)*, pages 373–380.

Menasche, M. (1982). *Analyse des réseaux de Petri temporisés et application aux systèmes distribués*.

Merlin, P. M. (1974). *A study of the recoverability of computing systems.* University of California, Irvine.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Oliveira, K. S. and Julia, S. (2020). Detection and removal of negative requirements of deadlock-type in service-oriented architectures. In *2020 International Conference on Computational Science and Computational Intelligence*.

Pradin-Chézalviel, B., Valette, R., and Kunzle, L. A. (1999). Scenario durations characterization of t-timed petri nets using linear logic. In *Proceedings 8th International Workshop on Petri Nets and Performance Models (Cat. No. PR00331)*, pages 208–217. IEEE.

Ramamoorthy, C. and Ho, G. S. (1980). Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Transactions on software Engineering*, (5):440–449.

Riviere, N., Pradin-Chezalviel, B., and Valette, R. (2001). Reachability and temporal conflicts in t-time petri nets. In *Proceedings 9th International Workshop on Petri Nets and Performance Models*, pages 229–238. IEEE.

Sifakis, J. (1979). Use of petri nets for performance evaluation. *Acta Cybernetica*, 4(2):185–202.

Soares Passos, L. M. and Julia, S. (2016). Linear logic as a tool for qualitative and quantitative analysis of workow processes. *International Journal on Artificial Intelligence Tools*, 25(03):1650008.

Van Der Aalst, W., Van Hee, K. M., and van Hee, K. (2004). *Workflow management: models, methods, and systems.* MIT press.