Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Qualitative Analysis of WorkFlow nets using Linear Logic: Soundness Verification

Lígia Maria Soares Passos, Stéphane Julia
Faculdade de Computação – Universidade Federal de Uberlândia, UFU
P.O. Box 593, 38400-902 – Uberlândia/MG, Brazil
ligia@pos.facom.ufu.br, stephane@facom.ufu.br

***Abstract*—This paper presents a method for the qualitative analysis of WorkFlow nets based on the proof trees of linear logic. The analysis is concerned with the proof of the correctness criterion Soundness defined for WorkFlow nets. To prove the Soundness property, a proof tree of linear logic is built for each different scenario of the WorkFlow net. Based on this approach, a method is proposed to verify in linear time if the WorkFlow net is sound.**

***Index Terms*—Petri net, Linear logic, WorkFlow net, Soundness**

I. Introduction

The purpose of Workflow Management Systems [1] is to execute Workflow Processes. Workflow Processes represent the sequences of activities which have to be executed within an organization to treat specific cases and to reach a well defined goal.

Petri nets [2] are very well adapted to model Real Time Systems, as they allow for a good representation of conflicting situations, shared resources, synchronous and asynchronous communication, precedence constraints and explicit quantitative time constraints in the time Petri net case.

There are several technics to analyze Workflow Processes and many papers have already considered the analysis of Workflow Processes. Two major analysis approaches are presented in [1]: *qualitative* and *quantitative*. Qualitative analysis is concerned with the logical correctness of the Workflow Process while quantitative analysis is concerned with performance and capacity requirements. The Soundness property is an important criterion to be satisfied when treating Workflow Processes, and the proof of Soundness is related to the qualitative analysis. In [1], three methods are presented to prove Soundness: the first method is based on reachability graphs, the second one is based on *liveness* and *boundedness* analysis for short-circuited nets and the last one is based on replacements of well-formed blocks. In [3], for example, the proposed method to prove Soundness is based on the construction of the coverability graph which may be very time consuming. The author also proposes a new class of WorkFlow nets, the Free-choice WorkFlow nets which allows the determining of Soundness in a polynomial time. In [4] the Well-Structured WorkFlow nets that have a number of desirable properties and for which Soundness can also be verified in polynomial time are presented. In [5] a method based on Propositional Logic is proposed to verify good properties in Workflow Processes. In particular, an activity-based workflow modeling formalism for logic-based workflow verification is presented. In this case, the complexity of the algorithm for workflow verification is $O(n^2)$ but it is not complete and cannot handle certain overlapping workflow patterns. In [6] a subset of the classical WorkFlow nets is proposed: the Normalized WorkFlow nets (NWF-nets). The NWF-nets add some structure constraints on the WorkFlow net routing patterns that guarantee the Soundness property. However, some Workflow Processes cannot be modeled using the NWF-net definition because of its construction limitations.

In this paper, an approach based on linear logic is proposed to analyze the qualitative properties of WorkFlow nets. The qualitative analysis is concerned with the proof of Soundness correctness criterion for WorkFlow nets.

In section II the definition of WorkFlow nets and of Soundness correctness criterion are provided. In section III an overview of linear logic is given. The qualitative analysis is presented in section IV. Finally, the last section concludes this work with a short summary, an assessment about the approach presented and an outlook on the future work.

II. WorkFlow nets

A Petri net that models a Workflow Process is called a WorkFlow net [1], [7]. A WorkFlow net satisfies the following properties [7]:

- 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*.

Soundness is a correctness criterion defined for WorkFlow nets. A WorkFlow net is sound if, and only if, the following three requirements are satisfied [1]:

- For each token put in the place *Start*, one and only one token appears in the place *End*.
- When the token appears in the place *End*, all the other places are empty for this case.
- For each transition (task), it is possible to move from the initial state to a state in which that transition is enabled, i.e. there is not any dead transition.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

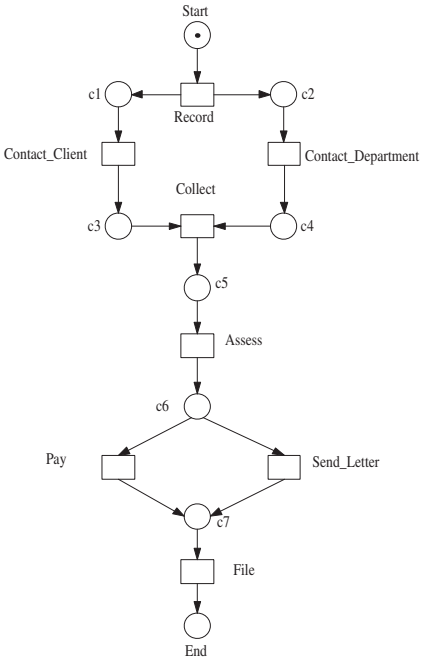

Fig. 1. WorkFlow net for the process "handle complaints" without modeling errors.

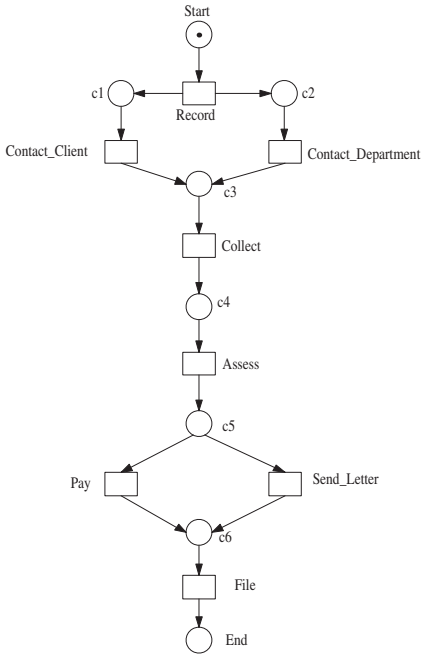

Fig. 2. WorkFlow net for the process "handle complaints" with a modeling error.

A. Process

A process defines which tasks need to be executed and in which order [7]. Modeling a Workflow Process in terms of a WorkFlow net is rather straightforward: transitions are active components and model the tasks, places are passive components and model conditions (pre and post) and tokens model cases [1], [7]. To illustrate the mapping of process into WorkFlow nets, the process for handling complaints that is shown in [1] can be considered: "an incoming complaint first is recorded. Then the client who has complained and 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 this, the data are gathered and a decision is taken. Depending upon the decision, either a compensation payment is made or a letter is sent. Finally, the complaint is filed". Fig. 1 shows a WorkFlow net that correctly models this process. Fig. 2 shows a WorkFlow net that models this process with a modeling error.

B. Routing constructs

Tasks can be optional, i.e. there are tasks that just need to be executed for some cases and the order in which tasks will be executed can vary from case to case [1]. Four basic constructions for routing are presented in [1] and [7]:

- *Sequential*: tasks are executed one after another sequentially, clearly demonstrating dependence among these tasks: one needs to finish for the other to start;
- *Parallel*: if more than one task can be executed simultaneously or in any order. In this case, both tasks can be executed without the result of one interfering in the result of the other;
- *Conditional* (or selective routing): when there is a choice between two or more tasks;
- *Iterative*: when it is necessary to execute the same task multiple times.

Some variations of these four basic constructions can be found in [1] and [7].

Considering the process "handle complaint", shown in Fig. 1, tasks *Contact_Client* and *Contact_Department* are an example of parallel routing. Tasks *Collect* and *Assess* are an example of sequential routing. And tasks *Pay* and *Send_Letter* are an example of conditional routing.

In the proposed approach, iterative routings will be substituted by a global activity as shown in Fig. 3. As a matter of fact, in practice, if a Workflow Process has to respect a deadline, it cannot indefinitely repeat the same activity. The hierarchical structure of Petri nets based on the concept of "Well-formed block" [8] can be used to represent iterative routing through a single task. Thus, a maximum duration will be associated with this task in order to specify in an implicit way the amount of time the "Well-formed block" could be executed before a detection mechanism points out a problem in the execution of the activity.

III. Linear Logic

Linear logic was proposed in 1987 by Girard [9]. In linear logic, propositions are considered as resources which are

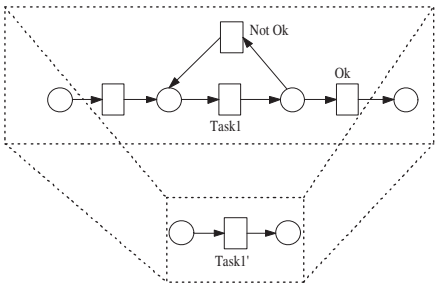

Fig. 3. Well-formed block.

consumed and produced at each state change [10]. Linear logic introduces new connectives, like the connectives *"par"*($\parr$), *"times"*($\otimes$), *"with"*($\&$), *"plus"*($\oplus$) and *"linear implies"*($\multimap$). In this paper just two connectives of linear logic will be used:

- The *times* connective, denoted by $\otimes$, represents simultaneous availability of resources. For instance, $A \otimes B$ represents the simultaneous availability of resources A and B.
- The *linear implies* connective, denoted by $\multimap$, represents a state change. For instance, $A \multimap B$ denotes that consuming A, B is produced (note that after the production of B, A will not be available).

The translation of a Petri net in formulas of linear logic was presented in [11]. A marking M is a monomial in $\otimes$ and is represented by $M = A_1 \otimes A_2 \otimes \ldots \otimes A_k$ where A_i are place names. For instance, the initial marking on the WorkFlow net in Fig. 1 is simply $Start$ because of the token in place $Start$. A transition is an expression of the form $M_1 \multimap M_2$ where M_1 and M_2 are markings. For example, transition Record of the WorkFlow net in Fig. 1 is noted $Record = Start \multimap c1 \otimes c2$. A sequent $M, t_i \vdash M'$ represents a scenario where M and M' are respectively the initial and final markings, and t_i is a list of non-ordered transitions [12]. For instance, considering the WorkFlow net shown in Fig. 1, the sequent $Start, t_1, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End$ represents one possible scenario of this WorkFlow net. A sequent can be proved by applying the rules of the sequent calculus. It was proved in [13] that a proof of the sequent calculus is equivalent to a reachability problem in a Petri net. The deduction system of linear logic is similar to the Gentzen's deduction system proposed in 1934 [14].

In this paper, just some rules of linear logic will be considered. These rules will be used to build the proof trees presented in this paper. For this, F, G, and H are considered as formulas and Γ and Δ as blocks of formulas. The following rules will be the ones used in this paper [11]:

- The $\multimap_L$ rule, $\dfrac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H}\ \multimap_L$, expresses a transition firing and generates two sequents such that the right sequent represents the subsequent remaining to be proved and the left sequent represents the consumed tokens by this firing. For example, considering the transition $Record = Start \multimap c1 \otimes c2$ of the WorkFlow net shown in Fig. 1, when this transition is fired, two sequents are generated: $Start \vdash Start$ represents the consumed tokens by this firing and $c1 \otimes c2$ the remaining subsequent to be proved.
- The $\otimes_L$ rule, $\dfrac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H}\ \otimes_L$ transforms a marking in an atoms list. For example, the subsequent $c1 \otimes c2$ generated by the firing of transition $Record = Start \multimap c1 \otimes c2$ of the WorkFlow net shown in Fig. 1 will use the rule $\otimes_L$ to be transformed in a list of atoms $c1, c2$.
- The $\otimes_R$ rule, $\dfrac{\Gamma \vdash F \quad \Delta \vdash G}{\Delta, \Gamma \vdash F \otimes G}\ \otimes_R$ transforms a sequent such $A, B \vdash A \otimes B$ into two identity sequents $A \vdash A$ and $B \vdash B$. For example, considering the firing of the transition $Collect = c3 \otimes c4 \multimap c5$ of the WorkFlow net shown in Fig. 1, the sequent that represents the consumed tokens by this firing, $c3, c4 \vdash c3 \otimes c4$, also needs to be proved, using the $\otimes_R$ rule, i.e. $\dfrac{c3 \vdash c3 \quad c4 \vdash c4}{c3, c4 \vdash c3 \otimes c4}\ \otimes_R$.

In the approach presented in this paper, a linear logic proof tree is read from the bottom-up. The proof stops when the atom End is produced, i.e. the identity sequent $End \vdash End$ appears in the proof tree, when there is not any rule that can be applied or when all the leaves of the proof tree are identity sequents. This proof is decidable once each transition of the WorkFlow net appears at the latest once in the linear sequent. Therefore, if the number of transitions is finite, the construction of the proof tree is decidable.

Proposition 1: The time complexity to prove a linear sequent of linear logic that represents a scenario of a WorkFlow net is $O(n)$, where n is the number of transitions.

Proof: If a sequent $M, t_i \vdash M'$ represents a scenario where M and M' are respectively the initial and final markings, and if t_i is a list of i non-ordered transitions and $i \leq n$, where n is the number of transitions in the WorkFlow net analyzed. In the worst case, for a given scenario of a WorkFlow net, all the transitions t_i of the WorkFlow net will appear as linear logic expressions of the form $c1 \otimes c2 \multimap c3 \otimes c4$. Thus, the three rules $\multimap_L$, $\otimes_R$ and $\otimes_L$ will be applied to prove each transition firing. The $\multimap_L$ rule will represent the transition firing, the rule $\otimes_R$ will be used to prove the left sequent generated by the transition firing, and the $\otimes_L$ rule will be used to transform the remaining subsequent to be proved in a list of atoms.

Hence, the time complexity to prove a linear sequent that represents a scenario will be $O(3n) = O(n)$, where n is the number of transitions in the WorkFlow net analyzed. ■

IV. Qualitative Analysis: Soundness verification

In the context of WorkFlow nets, the main aim of the qualitative analysis is to prove the Soundness correctness criterion. Thus, it is necessary to prove linear sequents that represent the WorkFlow net and analyze the proof trees built to prove these sequents. Initially, if the WorkFlow net has more than one route, it is then necessary to build and prove a different linear sequent for each existing scenario. A scenario corresponds to a well defined route mapped into the WorkFlow net. Only the

conditional routings can derive new scenarios. For example, in Fig. 1, there are two different scenarios: the first scenario, Sc_1, where task *Pay* will be executed (firing of transition *Pay*) and the second scenario, Sc_2, where task *Send_Letter* will be carried out (firing of transition *Send_Letter*). Thus, for this WorkFlow net, two linear sequents will have to be proved. Each linear sequent considers only one route.

For scenario Sc_1, the following sequent has to be proved:

$$Start, t_1, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End$$

For scenario Sc_2, the following sequent has to be proved:

$$Start, t_1, t_2, t_3, t_4, t_5, t_7, t_8 \vdash End$$

The transitions of the WorkFlow net are represented by the following formulas of linear logic:
$t_1 = Record = Start \multimap c1 \otimes c2$,
$t_2 = Contact\_Client = c1 \multimap c3$,
$t_3 = Contact\_Department = c2 \multimap c4$,
$t_4 = Collect = c3 \otimes c4 \multimap c5$,
$t_5 = Assess = c5 \multimap c6$,
$t_6 = Pay = c6 \multimap c7$,
$t_7 = Send\_Letter = c6 \multimap c7$,
$t_8 = File = c7 \multimap End$.

To prove these sequents, proof trees of linear logic will be produced. Each linear sequent has only one atom Start which represents the initial marking of the WorkFlow net. To analyze the Soundness criterion, just one token in place *Start* is necessary and sufficient. Thus, the proof method proposed in this paper always considers that there is always only one token in place *Start*. The proof tree corresponding to scenario Sc_1 is the following one:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c7 \vdash c7} \quad End \vdash End}{c6 \vdash c6} \quad c7, c7 \multimap End \vdash End}{c5 \vdash c5} \quad c6, c6 \multimap c7, t_8 \vdash End}{\frac{c3 \vdash c3 \quad c4 \vdash c4}{c3, c4 \vdash c3 \otimes c4} \otimes_R} \quad c5, c5 \multimap c6, t_6, t_8 \vdash End}{c2 \vdash c2} \quad c3, c4, c3 \otimes c4 \multimap c5, t_5, t_6, t_8 \vdash End}{c1 \vdash c1} \quad c2, c3, c2 \multimap c4, t_4, t_5, t_6, t_8 \vdash End}{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}}{Start \vdash Start} \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}$$

$$Start, Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End$$

The proof tree corresponding to scenario Sc_2 is the following one:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c7 \vdash c7} \quad End \vdash End}{c6 \vdash c6} \quad c7, c7 \multimap End \vdash End}{c5 \vdash c5} \quad c6, c6 \multimap c7, t_8 \vdash End}{\frac{c3 \vdash c3 \quad c4 \vdash c4}{c3, c4 \vdash c3 \otimes c4} \otimes_R} \quad c5, c5 \multimap c6, t_7, t_8 \vdash End}{c2 \vdash c2} \quad c3, c4, c3 \otimes c4 \multimap c5, t_5, t_7, t_8 \vdash End}{c1 \vdash c1} \quad c2, c3, c2 \multimap c4, t_4, t_5, t_7, t_8 \vdash End}{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}}{Start \vdash Start} \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}$$

$$Start, Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_7, t_8 \vdash End$$

For the WorkFlow net shown in the Fig. 2, two different sequents also needs to be proved. The sequent

$$Start, t_1, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End$$

needs to be proved for the scenario Sc_3, where the task *Pay* will be executed, and the sequent

$$Start, t_1, t_2, t_3, t_4, t_5, t_7, t_8 \vdash End$$

needs to be proved for the scenario Sc_4, where the task *Send_Letter* will be carried out, considering that:
$t_1 = Record = Start \multimap c1 \otimes c2$,
$t_2 = Contact\_Client = c1 \multimap c3$,
$t_3 = Contact\_Department = c2 \multimap c3$,
$t_4 = Collect = c3 \multimap c4$,
$t_5 = Assess = c4 \multimap c5$,
$t_6 = Pay = c5 \multimap c6$,
$t_7 = Send\_Letter = c5 \multimap c6$,
$t_8 = File = c6 \multimap End$.

The proof tree corresponding to scenario Sc_3 is the following one:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c6 \vdash c6} \quad c3, End \vdash End}{c5 \vdash c5} \quad c3, c6, c6 \multimap End \vdash End}{c4 \vdash c4} \quad c3, c5, c5 \multimap c6, t_8 \vdash End}{c3 \vdash c3} \quad c3, c4, c4 \multimap c5, t_6, t_8 \vdash End}{c2 \vdash c2} \quad c3, c3, c3 \multimap c4, t_5, t_6, t_8 \vdash End}{c1 \vdash c1} \quad c2, c3, c2 \multimap c3, t_4, t_5, t_6, t_8 \vdash End}{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}}{Start \vdash Start} \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}$$

$$Start, Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End$$

The proof tree corresponding to scenario Sc_4 is the following one:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c6 \vdash c6} \quad c3, End \vdash End}{c5 \vdash c5} \quad c3, c6, c6 \multimap End \vdash End}{c4 \vdash c4} \quad c3, c5, c5 \multimap c6, t_8 \vdash End}{c3 \vdash c3} \quad c3, c4, c4 \multimap c5, t_7, t_8 \vdash End}{c2 \vdash c2} \quad c3, c3, c3 \multimap c4, t_5, t_7, t_8 \vdash End}{c1 \vdash c1} \quad c2, c3, c2 \multimap c3, t_4, t_5, t_7, t_8 \vdash End}{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}}{Start \vdash Start} \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}$$

$$Start, Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_7, t_8 \vdash End$$

1) Each proof tree must be analyzed respecting the following steps:
   a) If just one atom End was produced in the proof tree (this is represented in the proof tree by the identity sequent $End \vdash End$), then the first requirement for Soundness is proved, i.e just one token appears in the place *End*.
   b) If there is not any available atom for consumption on the proof tree, then it means that all places are

empty, i.e. the second requirement for Soundness is proved.

c) There is not any available transition formula in the proof tree that was not fired.

2) Considering all scenarios $Sc_1, Sc_2, ..., Sc_n$ for the WorkFlow net analyzed, each transition $t \in T$ needs to appear in, at least, one scenario. This proves that all transitions were fired (i.e., there is not any dead transition), i.e. the third requirement for Soundness.

If the conditions 1 and 2 above are satisfied, the WorkFlow net is *Sound*.

Considering the proof trees for scenarios Sc_1 and Sc_2, the WorkFlow net shown in Fig. 1 is sound because the conditions 1 and 2 are satisfied, i.e.:

- Just one atom End was produced in each scenario.
- There was not any available atom for consumption in the proof trees.
- There was not any available transition formula in the proof trees that was not fired.
- Considering the scenarios Sc_1 and Sc_2, each transition of the WorkFlow net appears in, at least, one scenario and each one of these transitions was fired (for example, transition $t_2 = c1 \multimap c3$ appears in the two scenarios, Sc_1 and Sc_2, and was fired in the second application of the rule $\multimap_L$). Consequently, there are not any dead transitions in the WorkFlow net.

Considering the proof trees for scenarios Sc_3 and Sc_4, the WorkFlow net shown in Fig. 2 is not sound because the condition 1b is not satisfied, i.e. there is an available atom c3 for consumption in the proof trees. This fact shows that when a token is produced in the place *End*, there is still another place of the WorkFlow net that is not empty. Thus the second requirement for Soundness is not satisfied.

V. Conclusion

This paper presents a method for qualitative analysis of WorkFlow nets without iterative routings using linear logic. This approach is based on the proof of Soundness correctness criterion defined for WorkFlow nets.

The advantages of such an approach are diverse. The fact of working with linear logic permits one to prove the correctness criterion Soundness in a linear time without considering the construction of the reachability graph. In other words, the proof considers the proper structure of the WorkFlow net instead of considering the corresponding automata.

Moreover, some automatic theorem provers of linear logic, as LinTAP [15], Linprove [16] and llprover [17], can be adapted and used to automate the proof of the linear sequents presented in this paper.

As a future work proposal, it will be interesting to achieve a quantitative analysis of WorkFlow nets, based on the computation of symbolic dates for the planning of resources, using the proof trees with dates of linear logic as the ones presented in [11].

Acknowledgment

The authors would like to thank FAPEMIG for financial support to this work.

References

[1] W. M. P. van der Aalst and K. van Hee, *Workflow Management: Models, Methods, and Systems*. The MIT Press, March 2004.

[2] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989. [Online]. Available: http://dx.doi.org/10.1109/5.24143

[3] W. M. P. van der Aalst, "Verification of workflow nets," in *ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*. London, UK: Springer-Verlag, 1997, pp. 407–426.

[4] ——, "Structural characterizations of sound workflow nets." *Computing Science Reports/23*, no. 96, 1996.

[5] H. H. Bi and J. L. Zhao, "Applying propositional logic to workflow verification," *Information Technology and Management*, vol. 5, no. 3–4, pp. 293–318, July 2004.

[6] S. Li and B. Song, "Normalized workflow net (nwf-net): its definition and properties," *Future Gener. Comput. Syst.*, vol. 21, no. 7, pp. 1004–1014, 2005.

[7] W. M. P. van der Aalst, "The aplication of petri nets to workflow management," in *The Journal of Circuits,Systems and Computers*, 1998, pp. 21–66.

[8] R. Valette, "Analysis of Petri Nets by Stepwise Refinements," *Journal of Computer and System Sciences*, vol. 18, pp. 35–46, 1979.

[9] J.-Y. Girard, "Linear logic," *Theor. Comput. Sci.*, vol. 50, no. 1, pp. 1–102, 1987.

[10] B. Pradin-Chezalviel, R. Valette, and L. A. Kunzle, "Scenario durations characterization of t-timed petri nets using linear logic," in *PNPM '99: Proceedings of the The 8th International Workshop on Petri Nets and Performance Models*. Washington, DC, USA: IEEE Computer Society, 1999, p. 208.

[11] N. Riviere, R. Valette, B. Pradin-Chezalviel, and I. A. . Ups, "Reachability and temporal conflicts in t-time petri nets," in *PNPM '01: Proceedings of the 9th international Workshop on Petri Nets and Performance Models (PNPM'01)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 229.

[12] M. Soares and S. Julia, "Centralized architecture for real time scheduling of batch systems," in *Proceedings of the 11th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2004)*, Salvador, Brazil, 2004.

[13] F. Girault, *A logic for Petri nets*, JESA, Ed. Eddition Hermes, 1997, vol. 31, no. 3.

[14] P. Gochet and P. Gribomont, *Logique: méthodes pour l'informatique fondamentale*. Hermès, 1990, vol. Vol 1.

[15] H. Mantel and J. Otten, "lintap: A tableau prover for linear logic," in *TABLEAUX '99: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. London, UK: Springer-Verlag, 1999, pp. 217–231.

[16] T. Tammet, "Proof strategies in linear logic," *Journal of Automated Reasoning*, vol. 12, no. 3, pp. 273–304, October 1994.

[17] N. Tamura, "A linear logic prover (llprover)," http://bach.istc.kobe-u.ac.jp/llprover, 1997.