European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 5, no. 3-4, 2011, pp. 153-161
ISSN 1790–8140

Logical Termination of Workflows: An Interdisciplinary Approach[1]

Glória Cravo[2]

Centro de Ciências Exactas e da Engenharia,
Universidade da Madeira,
9000-390 Funchal, Madeira, Portugal

Received 17 January, 2009; accepted in revised form 14 December, 2010

Abstract: In this paper we present a new formalism to study the structure of workflows. A workflow is an abstraction of a business process that consists of one or more activities that need to be executed to reach a final objective. Our formalism is based on Graph Theory, Propositional Logic, and Boolean matrices. Indeed, we model workflows with tri-logic acyclic directed graphs. Moreover, we analyze the behavior of workflows using a certain type of Boolean matrices. In particular, we establish a necessary and sufficient condition for the logical termination of workflows.
Finally, we show that a derived workflow has the same behavior as the original workflow.

Keywords: Graphs, Classical Propositional Logic, Boolean Matrices, Workflows, Process Modeling, Business Processes

Mathematics Subject Classification: 03B05, 03G05, 05C20, 05C50, 05C60, 68R10

1 Introduction

Our main goal is to provide a new theoretical mathematical foundation that can describe and analyze workflows. A workflow is an abstraction of a business process that consists of one or more tasks to be executed to reach a final objective (for example, hiring process, sales order processing, loan application, insurance claim, and so on). In other words, workflows are modeled business processes.
Our formalism is based on the use of Graph Theory, Propositional Logic and Boolean matrices. In our approach workflows are modeled with graphs, whose tasks are represented with vertices and the transitions are modeled with arcs. Each task of a workflow represents a unit of work that can be executed by humans or software applications. We consider tasks as atomic pieces of the workflow, since they generate all the transitions and models (i.e., simple parts of the workflow) present in the workflow.
Workflows have been applied to several domains, including the telecommunications industry [12], school administration [1], bio-informatics [7, 10], healthcare [5] and so on.

[1]Published electronically December 30, 2010
[2]Corresponding author. Member of the Centro de Estruturas Lineares e Combinatórias. E-mail: gcravo@uma.pt

In the last decade, the rapid increase of business process modeling and management through the adoption of workflows, has originated the need for frameworks that can provide a formal technique for defining and analyzing workflows. A vaste number of formal frameworks have been proposed to allow workflow modeling verification and analysis. We emphasize some relevant tools as State and Activity Charts [13], Graphs [2], Event-Condition-Action rules [8, 9], Petri Nets [3, 4], Temporal Logic [6] and Markov chains [11]. However more research is required with respect to the use of Graph Theory.
In this paper we present a formal definition of a workflow, based on Graph Theory and simple concepts of Logic. In our approach we model workflows with tri-logic acyclic directed graphs and develop a formalism to verify their termination. The study of the termination of workflows is based in a specific type of Boolean matrices. This approach is completely new and allows to check easily if a workflow logically terminates. We still show that a derived workflow has the same behavior as the original workflow, i.e., given two workflows, if one is derived from the other, then necessarily both workflows logically terminate.

2 Logical Termination of Workflows

In our approach we model workflows with tri-logic acyclic directed graphs. Each vertex of the graph has an input/output logic operator. In the following definition we provide the formal structure of a workflow.

Definition 1 *A workflow is a tri-logic acyclic directed graph* $WG = (T, A)$, *where* $T = \{t_1, t_2, \ldots, t_n\}$ *is a finite nonempty set of vertices representing workflow tasks. Each task* t_i (*i.e., a vertex*) *has an input logic operator* (*represented by* $\succ t_i$) *and an output logic operator* (*represented by* $t_i \prec$). *An input/output logic operator can be the logical AND* ($\bullet$), *the OR* ($\otimes$), *or the XOR -exclusive-or-* ($\oplus$). *The set* $A = \{a_{\sqcup}, a_{\sqcap}, a_1, a_2, \ldots, a_m\}$ *is a finite nonempty set of arcs representing workflow transitions. Each transition* $a_i, i \in \{1, \ldots, m\}$, *is a tuple* (t_k, t_l) *where* t_k, $t_l \in T$. *The transition* $a_{\sqcup}$ *is a tuple of the form* $(\sqcup, t_1)$ *and transition* $a_{\sqcap}$ *is a tuple of the form* $(t_n, \sqcap)$. *The symbols* $\sqcup$ *and* $\sqcap$ *represent abstract tasks which indicate the entry and ending point of the workflow, respectively. We use the symbol* $'$ *to reference the label of a transition, i.e.,* a'_i *references transition* a_i, $a_i \in A$. *The elements* a'_i *are called Boolean terms and form the set* A'.

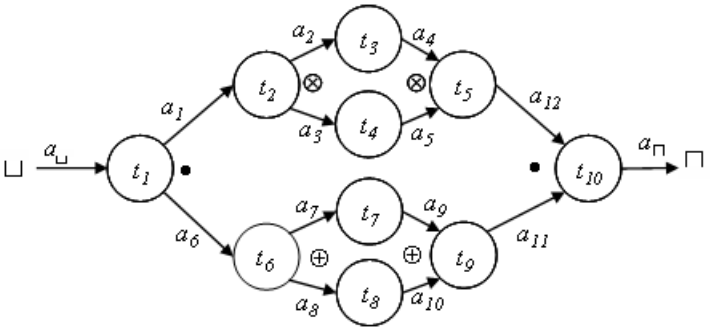

Figure 1: Example of a tri-logic acyclic directed graph (i.e., a workflow)

Definition 2 *Let* $t_i \in T$. *The incoming/outgoing condition for task* t_i *is a Boolean expression* $a'_{k_1}\varphi a'_{k_2}\varphi \ldots \varphi a'_{k_l}, \varphi \in \{\bullet, \otimes, \oplus\}$, *where* $a'_{k_1}, a'_{k_2}, \ldots, a'_{k_l} \in A'$, *and* $a_{k_1}, a_{k_2}, \ldots, a_{k_l}$ *are the incoming/outgoing transitions of task* t_i. *The terms* $a'_{k_1}, a'_{k_2}, \ldots, a'_{k_l}$ *are connected with the logical operator* $\succ t_i / t_i \prec$.

Remark 3 *When task t_i has only one incoming/outgoing transition we can assume that the condition does not have logical operator.*

Example 4 *In Figure 1 is shown a workflow $WG = (T, A)$, where $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$, $A = \{a_{\sqcup}, a_{\sqcap}, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}\}$ and $A' = \{a'_{\sqcup}, a'_{\sqcap}, a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}\}$. The tuple $a_2 = (t_2, t_3)$ is an example of a transition. In task t_{10}, the input logic operator $(\succ t_{10})$ is an AND $(\bullet)$; in task t_2 the output logic operator $(t_2 \prec)$ is an OR $(\otimes)$.*
The incoming transition for task t_2 is $a_1 = (t_1, t_2)$ and the outgoing transitions are $a_2 = (t_2, t_3)$ and $a_3 = (t_2, t_4)$.
The incoming condition of task t_2 is a'_1, and its outgoing condition is $a'_2 \otimes a'_3$.

Definition 5 *Let $WG = (T, A)$ be a workflow and let $t_i \in T$. An Event-Action (EA) model for task t_i is an implication of the form $t_i : f_E \rightsquigarrow f_C$, where f_E and f_C are the incoming and outgoing conditions of task t_i, respectively, whose Boolean values are described according to Table 1. The condition f_E is called the event condition and the condition f_C is called the action condition.*

Table 1: Behavior of an EA model

f_E	f_C	$f_E \rightsquigarrow f_C$
0	0	0
1	1	1

Definition 6 *Let $WG = (T, A)$ be a workflow and let $t_i : f_E \rightsquigarrow f_C$ be an EA model. We say that the EA model is positive if its value is 1, otherwise we say that the model is negative.*

A workflow starts its execution when transition $a_{\sqcup}$ is enabled. A transition is enabled/disabled if the respective Boolean term is asserted to be true/false. Hence, the workflow starts its execution by asserting $a'_{\sqcup}$ to be true.
On the other hand, a workflow is a set of EA models, i.e., a set of tasks and transitions. The tasks can be considered as atomic pieces of the workflow, since they generate all transitions and, consequently, all EA models present in the workflow.
Given an EA model, it is clear that the value of the incoming condition is propagated to the outgoing condition. More generally, the value of each EA model is propagated to the following EA models connected to it. We need to exploit how this propagation affects the execution of the workflow. We start by defining the behavior of the workflow.

Definition 7 *Let $WG = (T, A)$ be a workflow. The behavior of WG is described by its EA models, according to the following rules*:
(1) *The workflow starts its execution by asserting $a'_{\sqcup}$ to be true.*
(2) *For every $t_i : f_{E_i} \rightsquigarrow f_{C_i}$, $i \in \{2, \ldots, n\}$, the Boolean values of f_{E_i} and f_{C_i} will be asserted according to Table 1.*
(3) *The workflow stops its execution when one of the following cases occur*:
(3.1) *$a'_{\sqcap}$ is asserted to be true.*
(3.2) *$a'_{\sqcap}$ is asserted to be false.*

One important structural property of workflows is its logical termination. It is important to know if a business process will be finished at runtime. For example, it is important to know if a loan application will be complete. Next, we introduce the concept of logical termination.

Definition 8 *Let $WG = (T, A)$ be a workflow. We say that WG logically terminates if $a'_{\sqcap}$ is true whenever $a'_{\sqcup}$ is true.*

To carry out an exhaustive characterization of workflows we will focus our study on the EA models. We start by considering two different types of EA models.

Definition 9 *Let $WG = (T, A)$ be a workflow. Let $t_i : f_E \rightsquigarrow f_C$ be an EA model. The EA model is said to be simple if $f_E = a'_j$ and $f_C = a'_l$, $j, l \in \{\sqcup, \sqcap, 1, \ldots, m\}$, with $j \neq l$. Otherwise, we say that the EA model is non-simple.*

Example 10 *In Figure 1 the EA model $t_3 : a'_2 \rightsquigarrow a'_4$ is a simple EA model, while the EA models $t_2 : a'_1 \rightsquigarrow a'_2 \otimes a'_3$ and $t_9 : a'_9 \oplus a'_{10} \rightsquigarrow a'_{11}$ are non-simple EA models.*

Remark 11 *Note that if $t_i : f_E \rightsquigarrow f_C$ is a non-simple EA model, then it has one of the following forms:*
(a) $f_E = a'_{j_1} \varphi a'_{j_2} \varphi \ldots \varphi a'_{j_k}, \varphi \in \{\bullet, \otimes, \oplus\}$ and $f_C = a'_l$;
(b) $f_E = a'_j$ and $f_C = a'_{l_1} \varphi a'_{l_2} \varphi \ldots \varphi a'_{l_r}, \varphi \in \{\bullet, \otimes, \oplus\}$;
(c) $f_E = a'_{j_1} \varphi a'_{j_2} \varphi \ldots \varphi a'_{j_k}, \varphi \in \{\bullet, \otimes, \oplus\}$ and $f_C = a'_{l_1} \psi a'_{l_2} \psi \ldots \psi a'_{l_r}, \varphi \in \{\bullet, \otimes, \oplus\}$.

Definition 12 *In conditions of Remark 11 the EA models with the forms (a) and (b) are called complex, while the EA models with the form (c) are said to be hybrid.*

Remark 13 *If all EA models of the workflow are simple, then its structure is the following:*

$$\sqcup \xrightarrow{a_{\sqcup}} t_1 \xrightarrow{a_1} t_2 \xrightarrow{a_2} t_3 \ldots t_{n-1} \xrightarrow{a_{n-1}} t_n \xrightarrow{a_{\sqcap}} \sqcap.$$

In this case, the set of non-simple EA models is empty. This situation is a trivial case of logical termination, since all the EA models present in the workflow are positive, and consequently, $a'_{\sqcap}$ is true whenever $a'_{\sqcup}$ is true, i.e., the workflow logically terminates.

From now on, we will assume that the workflow contains non-simple EA models.

Definition 14 *Let $WG = (T, A)$ be a workflow. A materialized worflow instance of WG is an assignment of Boolean values to all Boolean terms $a'_j \in A'$, according to Table 1.*

Notation 15 *Let $N = \{i \in \{1, \ldots, n\} | t_i : f_{E_i} \rightsquigarrow f_{C_i}$ is a non-simple EA model$\}$.*

Definition 16 *Assume that $N = \{i_1, i_2, \ldots, i_l\}$ and the elements $i_1, i_2, \ldots, i_l$ appear in increasing order, i.e., $i_1 < i_2 < \cdots < i_l$. For any materialized workflow instance of WG, let us consider the following Boolean matrix $B = [b_{i,j}] = \mathrm{diag}(\mathrm{b_{i_1,i_1}}, \mathrm{b_{i_2,i_2}}, \ldots, \mathrm{b_{i_l,i_l}}) \in \mathrm{F}^{l \times l}$, where b_{i_s,i_s} is the Boolean value of the EA model $t_{i_s} : f_{E_{i_s}} \rightsquigarrow f_{C_{i_s}}$, $s \in \{1, 2, \ldots, l\}$. The matrix B is called the Event Action Boolean matrix.*

The following result provides a necessary and sufficient condition for the logical termination of workflows.

Theorem 17 *Let $WG = (T, A)$ be a workflow and assume that $N = \{i_1, i_2, \ldots, i_l\}$, $i_1 < i_2 < \cdots < i_l$. Then, WG logically terminates if and only if one of the following conditions is satisfied:*
(a) If the input logic operator of task t_n ($\succ t_n$) is an AND, every Event Action Boolean matrix is equal to the identity matrix of type $l \times l$ (i.e., I_l);
(b) If the input logical operator of task t_n ($\succ t_n$) is an OR, there exists at least a path of WG for which every Event Action Boolean matrix is equal to the identity matrix of type $l \times l$ (i.e., I_l);
(c) If the input logical operator of task t_n ($\succ t_n$) is a XOR, there exists one and only one path of WG for which every Event Action Boolean matrix is equal to the identity matrix of type $l \times l$ (i.e., I_l).

Proof. Let $N = \{i \in \{1,\ldots,n\} | t_i : f_{E_i} \rightsquigarrow f_{C_i}$ is a non-simple EA model$\}$. Assume that $N = \{i_1, i_2, \ldots, i_l\}$ and the elements $i_1, i_2, \ldots, i_l$ appear in increasing order, i.e., $i_1 < i_2 < \cdots < i_l$.
Case 1. Suppose that the input logic operator of task t_n ($\succ t_n$) is an AND.
Let us assume that WG logically terminates, i.e., $a'_{\sqcap}$ is true whenever $a'_{\sqcup}$ is true.
Let us assume, by contradiction, that there exits an Event Action Boolean matrix different from the identity matrix of type $l \times l$. Then there exists at least a materialized workflow instance such that the respective Event Action Boolean matrix differs from the identity matrix of type $l \times l$. This means that there exists at least $j \in \{i_1, \ldots, i_l\}$ such that the EA model $t_j : f_{E_j} \rightsquigarrow f_{C_j}$ is negative. Recall the workflow starts its execution by enabling $a_{\sqcup}$, i.e., by asserting $a'_{\sqcup}$ to be true.
Since $a'_{\sqcup}$ is asserted to be true, according to the behavior of the workflow f_{C_1} is true. Therefore, $j > 1$. We may assume, without loss of generality, that $j = i_r$, $1 < r < l$. Since $t_{i_r} : f_{E_{i_r}} \rightsquigarrow f_{C_{i_r}}$ is negative, necessarily both $f_{E_{i_r}}, f_{C_{i_r}}$ are false. So for every $k > i_r$, $t_k : f_{E_k} \rightsquigarrow f_{C_k}$ is negative, if $t_k : f_{E_k} \rightsquigarrow f_{C_k}$ is connected to $t_{i_r} : f_{E_{i_r}} \rightsquigarrow f_{C_{i_r}}$. Hence, $a_{\sqcap}$ is not enabled, which means that $a'_{\sqcap}$ is false. Clearly, this fact is a contradiction, since the workflow logically terminates. Consequently, every Action Boolean matrix is equal to the identity matrix of type $l \times l$.
Conversely, suppose that the workflow starts its execution, i.e., $a'_{\sqcup}$ is asserted to be true.
According to the hypothesis, every Event Action Boolean matrix is equal to I_l.
Hence, for every materialized workflow instance of WG and for every $i \in N$, the EA model $t_i : f_{E_i} \rightsquigarrow f_{C_i}$ is positive, which means that both incoming and outgoing conditions, respectively, f_{E_i}, f_{C_i} are true. Clearly one of the following cases must occur:
(a) $n \in N$;
(b) $n \notin N$.
Subcase 1.1. Suppose that $n \in N$. Then according to the hypothesis the EA model $t_n : f_{E_n} \rightsquigarrow a'_{\sqcap}$ is positive. Consequently, both conditions $f_{E_n}, a'_{\sqcap}$ are true. In particular, $a'_{\sqcap}$ is true.
Subcase 1.2. Suppose that $n \notin N$. Necessarily the EA model $t_n : f_{E_n} \rightsquigarrow a'_{\sqcap}$ is simple. Hence $f_{E_n} = a'_m$. Again, one of the following situations must occur:
(a) $n - 1 \in N$;
(b) $n - 1 \notin N$.
Subcase 1.2.1. Suppose that $n - 1 \in N$. Then the EA model $t_{n-1} : f_{E_{n-1}} \rightsquigarrow a'_m$ is positive. Therefore, both conditions $f_{E_{n-1}}, a'_m$ are true. Since a'_m is true, according to the behavior of the workflow, $a'_{\sqcap}$ is also true.
Subcase 1.2.2. Suppose that $n - 1 \notin N$. Then the EA model $t_{n-1} : f_{E_{n-1}} \rightsquigarrow a'_m$ is simple. Hence, $f_{E_{n-1}} = a'_{m-1}$. Carrying out recursively this reasoning we can conclude that there exists $r \in \{2, \ldots, n-1\}$ such that the EA model $t_r : f_{E_r} \rightsquigarrow f_{C_r}$ satisfies the following conditions:
(i) $r \in N$;
(ii) $f_{C_r} = a'_k$, for some $k \in \{1, \ldots, m-2\}$;
(iii) For every $j > r$ the EA model $t_j : f_{E_j} \rightsquigarrow f_{C_j}$ is a simple EA model.
Since $r \in N$, then the EA model $t_r : f_{E_r} \rightsquigarrow f_{C_r}$ is positive. Therefore, both its incoming and outgoing conditions are true. According to the behavior of the workflow, for every $j > r$, the EA model $t_j : f_{E_j} \rightsquigarrow f_{C_j}$ is positive. Hence, both conditions f_{E_j}, f_{C_j} are true. In particular, $a'_{\sqcap}$ is true.
Bearing in mind that in both Subcases 1.1, 1.2, we get $a'_{\sqcap}$ asserted to be true, we can conclude that the workflow logically terminates.
Case 2. Suppose that the input logical operator of task t_n ($\succ t_n$) is an OR.
Suppose that WG logically terminates.
Let us assume, by contradiction, that for every path of WG, there exits an Event Action Boolean matrix different from the identity matrix of type $l \times l$. Using similar arguments to those used in Case 1, we can conclude that $a_{\sqcap}$ is not enabled, and consequently, $a'_{\sqcap}$ is false. Clearly, this fact is a contradiction since WG logically terminates. Hence, there exists at least a path of WG for which every Event Action Boolean matrix is equal to the identity matrix of type $l \times l$.

Conversely, suppose that WG starts its execution. According to the hypothesis there exists at least a path of WG for which every Event Action Boolean matrix is equal to the identity matrix of type $l \times l$. Using similar arguments to those from Case 1, we can conclude that $a'_{\sqcap}$ is true. This means that WG logically terminates.
Case 3. Suppose that the input logical operator of task t_n ($\succ t_n$) is a XOR.
Suppose that WG logically terminates. And let us assume, by contradiction, that for every path of WG, there exits an Event Action Boolean matrix different of the identity matrix of type $l \times l$. Again, using similar arguments to those used in Case 1, we can conclude that $a_{\sqcap}$ is not enabled, and consequently, $a'_{\sqcap}$ is false. This fact is a contradiction, since WG logically terminates. So, there exists at least a path of WG for which every Boolean matrix is the identity matrix of type $l \times l$. Bearing in mind that the input logical operator of task t_n ($\succ t_n$) is a XOR, necessarily it exists only one path in these conditions, otherwise WG would not logically terminate.
Conversely, suppose that WG starts its execution. According to the hypothesis there exists one and only one path of WG for which every Event Action Boolean matrix is equal to the identity matrix of type $l \times l$. Using again similar arguments to those from Case 1, we can conclude that $a'_{\sqcap}$ is true, which means that WG logically terminates. ■

This is a very important result, since it allows to study a very important sructural property of workflows, their logical termination.

Example 18 *The workflow from Figure 1 has the following non-simple EA models:* $t_1 : a'_{\sqcup} \rightsquigarrow a'_1 \bullet a'_6$, $t_2 : a'_1 \rightsquigarrow a'_2 \otimes a'_3$, $t_5 : a'_4 \otimes a'_5 \rightsquigarrow a'_{12}$, $t_6 : a'_6 \rightsquigarrow a'_7 \oplus a'_8$, $t_9 : a'_9 \oplus a'_{10} \rightsquigarrow a'_{11}$, $t_{10} : a'_{11} \bullet a'_{12} \rightsquigarrow a'_{\sqcap}$. *Hence* $N = \{1, 2, 5, 6, 9, 10\}$. *We have as many Event Action Boolean matrices as materialized workflow instances of* WG. *Notice the input logic operator of task* t_{10} *is an AND. It is easy to verify that every Event Action Boolean matrix is equal to the identity matrix of type* 6×6. *Therefore, the workflow logically terminates.*

In what follows we will emphasize the study of hybrid EA models. Indeed, we will prove that every hybrid EA model can be split into two new complex EA models.

Theorem 19 *Let* $WG = (T, A)$ *be a workflow and let* $t_i \in T$. *Let* $t_i : f_E \rightsquigarrow f_C$ *be a hybrid EA model. Then* $t_i : f_E \rightsquigarrow f_C$ *can be split into two complex EA models.*

Proof. Let $t_i : f_E \rightsquigarrow f_C$ be a hybrid EA model. Then both incoming and outgoing conditions, f_E and f_C, respectively, are Boolean terms with an *AND* ($\bullet$), *an OR* ($\otimes$), *or a XOR* ($\oplus$).
Now we can create two auxiliary tasks t_i^1, t_i^2 and an auxiliary transition $b_i = (t_i^1, t_i^2)$. Let b'_i be the Boolean term associated with the auxiliary transition b_i such that b'_i has the same Boolean value of f_E. Hence, $t_i^1 : f_E \rightsquigarrow b'_i$ and $t_i^2 : b'_i \rightsquigarrow f_C$ are EA models.
Bearing in mind that b'_i has the same Boolean value of f_E, and therefore, f_C has its Boolean value depending on the Boolean value of b'_i, when we consider these new EA models instead of the initial hybrid EA model, the behavior of the workflow does not modify. It is clear that the EA models $t_i^1 : f_E \rightsquigarrow b'_i$ and $t_i^2 : b'_i \rightsquigarrow f_C$ are complex. ■

Notation 20 *The set of all auxiliary tasks created from* T, *will be denoted by* T^* *and the set of all auxiliary transitions created from* A *will be denoted by* A^*.

Definition 21 *Let* $WG_1 = (T_1, A_1)$ *and* $WG_2 = (T_2, A_2)$ *be workflows. Suppose that* $T_2 = T_1 \cup T_1^*$ *and* $A_2 = A_1 \cup A_1^*$. *Let* NH_i *be the set of all non-hybrid EA models of* $WG_i, i \in \{1, 2\}$. *We say that* WG_2 *is derived from* WG_1, *or* WG_1 *derives* WG_2, *if the following conditions are satisfied:*
(a) $NH_1 = NH_2$;
(b) Every hybrid EA model of WG_1 *is split into two complex EA models of* WG_2.

The next auxiliary result is an immediate consequence of the proof of Theorem 19, and allow us to verify that a derived workflow has the same behavior as the original workflow.

Lemma 22 *Let $WG_1 = (T_1, A_1)$ and $WG_2 = (T_2, A_2)$ be workflows such that WG_2 is derived from WG_1. Then WG_1 and WG_2 have the same behavior.*

Proof. Since WG_2 is derived from WG_1, then $NH_1 = NH_2$ and every hybrid EA model of WG_1 is split into two complex EA models of WG_2.
Since $NH_1 = NH_2$, to verify that WG_1 and WG_2 have the same behavior, it remains to verify that when we split an hybrid EA model of WG_1 into two derived complex EA models of WG_2, the behavior of these new EA models coincide with the behavior of the initial hybrid EA model.
Let $t_i : f_E \rightsquigarrow f_C$ be an arbitrary hybrid EA model of WG_1. According to the proof of Theorem 19 when we split $t_i : f_E \rightsquigarrow f_C$ into two complex EA models of WG_2, $t_i^1 : f_E \rightsquigarrow b_i'$ and $t_i^2 : b_i' \rightsquigarrow f_C$, the behavior of these new EA models coincide with the behavior of the initial hybrid EA model from WG_1. Therefore, the behavior of WG_1 coincides with the behavior of WG_2. ■

As a consequence of the previous results, we describe an important structural property of workflows, by showing that given workflows, WG_1, WG_2 such that WG_2 is derived from WG_1, then they have the same properties from the point of view of their logical termination, i.e., WG_1 logically terminates if and only if WG_2 logically terminates.

Theorem 23 *Let $WG_1 = (T_1, A_1)$ and $WG_2 = (T_2, A_2)$ be workflows and assume that WG_2 is derived from WG_1. Then, WG_1 logically terminates if and only if WG_2 logically terminates.*

Proof. Since WG_2 is derived from WG_1, according to Lemma 22 WG_1 and WG_2 have the same behavior.
Suppose that WG_1 logically terminates and $a'_{\sqcup}$ is asserted to be true in WG_2. As WG_1 and WG_2 have the same behavior, $a'_{\sqcup}$ is necessarily asserted to be true in WG_1. On the other hand, since WG_1 logically terminates, $a'_{\sqcap}$ is asserted to be true in WG_1.
Now taking again into account that WG_1 and WG_2 have the same behavior, then $a'_{\sqcap}$ is necessarily asserted to be true in WG_2. Hence WG_2 logically terminates.
The proof of the converse is analogous. ■

3 Conclusions

Workflows have been widely exploit with a significant number of formal frameworks that arose to allow workflow modeling verification and analysis. However, more research is required, particularly on the concern of Graph Theory and Propositional Logic.
Workflows describing critical applications, such as healthcare process, require a precise modeling, verification and analysis to ensure that they perform according to initial specifications. To guarantee that workflows are sucessfully executed at runtime, it is necessary to verify their properties at design time.
In this paper we present a formal framework, based on Graph Theory, Propositional Logic and Boolean matrices. We also study a very important property of workflows: their logically termination. Our approach has the advantage to check very easily if a workflow logically terminates. The contribution of our work will enable the development of a new set of tools that will support and allow business process analysts to verify the correct design of their workflows in an early phase of the workflow lifecycle development.
Moreover, our approach allows to reduce the study of some workflows to others. In particular, we show that a derived workflow has the same behavior as the original workflow.

Acknowledgment

The author wishes to thank the anonymous referees for their careful reading of the manuscript and their fruitful comments and suggestions.

References

[1] CAPA, course approval process automation. Technical Report. LSDIS Lab, Department of Computer Science, University of Georgia: Athens, GA, July 1, 1996 - June 30, 1997.

[2] *METEOR (Managing End-To-End OpeRations) Project Home Page*, LSDIS Lab, 2006. Acessed from http://lsdis.cs.uga.edu/projects/past/METEOR/.

[3] W. V. M. P. v. d. Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[4] W. V. M. P. v. d. Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In W. V. M. P. v. d. Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, pages 161–183. Springer-Verlag, Berlin, 2000.

[5] K. Anyanwu, A. Sheth, J. Cardoso, J. Miller, and K. Kochut. Healthcare enterprise process development and integration. *Journal of Research and Practice in Information Technology, Special Issue in Health Knowledge Management*, 35(2):83–98, 2003.

[6] P. Attie, M. Singh, A. Sheth, and M. Rusinkiewicz. Specifying and enforcing intertask dependencies. In *Proceedings of 19th International Conference on Very Large Data Bases*, pages 134–145, Dublin, Ireland, 1993. Morgan Kaufman.

[7] J. Cardoso, R. P. Bostrom, and A. Sheth. Workflows management systems and ERP systems: Differences, commonalities, and applications. *Information Technology and Management Journal, Special issue on Workflow and E-business*, 5(3-4):319–338, 2004. Kluwer Academic Publishers.

[8] U. Dayal, M. Hsu, and R. Ladin. Organizing long-running activities with triggers and transactions. In *ACM SIGMOD International Conference on Management of Data Table of Contents*, pages 204–214, Atlantic City, New Jersey, 1990. ACM Press, New York, NY, USA.

[9] J. Eder, H. Groiss, and H. Nekvasil. A workflow system based on active databases. In G. Chroust and A. Benczur, editors, *Proceedings of CON' 94, Workflow Management: Challenges, Paradigms and Products*, pages 249–265, Linz, Austria, 1994.

[10] R. Hall, J. Miller, J. Arnold, J. Kochut, A. Sheth, and M. Weise. Using workflw to build an information management system for a geographically distributed genome sequence iniciative. In R. A. Prade and H. J. Bohnert, editors, *Genomics of Plants and Fungi*, pages 359–371. Marcel Dekker, Inc.: New York, NY, 2003.

[11] J. Klingemann, J. Wäsch, and K. Aberer. Deriving service models in cross-organizational workflows. In *Proceedings of RIDE-Information Technology for Virtual Enterprises (RIDE-VE' 99)*, pages 100–107, Sydney, Australia, 1999.

[12] Z. Luo. *Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes*. PhD thesis, University of Georgia, 2000.

[13] P. Muth, D. Wodtke, J. Weissenfels, G. Weikum, and K. Dittrich. Enterprise-wide workflow management based on state and activity charts. In A. Dogac, L. Kalinichenko, T. Ozsu, and A. Sheth, editors, *Proceedings NATO Advanced Study Institute on Workflow Management Systems and Interoperability*. Springer-Verlag, 1998.