Hindawi Publishing Corporation
The Scientific World Journal
Volume 2015, Article ID 481767, 7 pages
http://dx.doi.org/10.1155/2015/481767

Research Article

Workflow Modelling and Analysis Based on the Construction of Task Models

Glória Cravo[1,2]

[1]*Center for Linear Structures and Combinatorics, University of Lisbon, 1649-003 Lisbon, Portugal*
[2]*Center of Exact Sciences and Engineering, University of Madeira, Funchal, 9020-105 Madeira, Portugal*

Correspondence should be addressed to Glória Cravo; gcravo@uma.pt

Received 30 September 2014; Accepted 3 December 2014

Academic Editor: Dar-Li Yang

We describe the structure of a workflow as a graph whose vertices represent tasks and the arcs are associated to workflow transitions in this paper. To each task an input/output logic operator is associated. Furthermore, we associate a Boolean term to each transition present in the workflow. We still identify the structure of workflows and describe their dynamism through the construction of new task models. This construction is very simple and intuitive since it is based on the analysis of all tasks present on the workflow that allows us to describe the dynamism of the workflow very easily. So, our approach has the advantage of being very intuitive, which is an important highlight of our work. We also introduce the concept of logical termination of workflows and provide conditions under which this property is valid. Finally, we provide a counter-example which shows that a conjecture presented in a previous article is false.

1. Introduction

A workflow is an abstraction of a business process that consists on the execution of a set of tasks to complete a process (e.g., hiring process, loan application, and sales order processing). Tasks represent unities of work to be executed that can be processed by a combination of resources, such as a computer program, an external system, or human activity.

Recall that a business process is a collection of interconnected tasks that takes one or more kinds of input and creates an output that is of value to the customers. The construction of process models is very often a difficult accomplishment for humans since their design can be logically incorrect and enclose errors. So the development of tools to support the design of business processes is indispensable and needs to be based on a solid theory. We believe that our technique is appropriate to model workflows.

Workflows have been successfully deployed to various domains, such as bioinformatics, healthcare, the telecommunication industry, the military, insurance, school administration, mobile computing, systems management, multidata bases, Internet, application development, object technology, operating systems, and transaction management.

In the present paper, we use Graph Theory and Propositional Logic to describe the structure of workflows. It is important to point out that a workflow describes all of the tasks needed to achieve each step in a business process. Documents, information, work orders, reports, and so forth are passed from one task to another for action, according to a set of rules defined by the workflow. Employees or automated applications are the entities that carry out the execution of tasks.

In particular we model workflows as trilogic acyclic directed graphs. The use of trilogic graphs to represent workflows was selected since most business process languages support three types of connectors: AND, OR, and XOR. It is important to emphasize that the inclusion of workflows with OR vertices is a significant advantage of our approach.

Furthermore, our approach is based on the execution of all tasks present on the workflow. This analysis has the advantage of being simple and very intuitive. On the other hand, we can create new models based on the existing ones.

Besides our formal framework allows checking the logical termination of workflows. The logical termination is an important property for workflows because it is indispensable to know if a workflow, such as the hiring process, will

eventually finish. Analyzing the termination of workflows is an important assignment since research and commercial products, such as METEOR and TIBCO, have no support for verification. Errors made at design-time are not detected and result in very costly failures at run-time.

The use of Propositional Logic has the advantage of transforming a workflow into a set of Event-Action (EA) models. Specialized EA models can be easily created to represent new advanced workflow patterns. Afterwards, Propositional Logic and Inference can be carried out on the EA models to analyze properties of workflow models.

It is important to point out that, in the last decade, the rapid increase of business process modelling and management through the adoption of Workflow Management Systems has originated the need for frameworks that can be used to provide a formal technique for defining and analyzing workflows [1–3]. Important advancements have been accomplished in the development of theoretical foundations to allow workflow modeling, verification, and analysis. Several formal methods have been proposed, such as State and Activity Charts [4], Event-Condition-Action rules [5, 6], Petri Nets [7–11], Temporal Logic [12], Markov chains [13], Process and Event Algebras [14, 15], and Six Sigma Techniques [16, 17]. Nevertheless more research is required and specially focused on the use of Graph Theory. Based on this need, we develop our formalism that uses a natural combination of Graph Theory and Propositional Logic to model workflows. Besides, our formalism provides a formal framework based on trilogic acyclic directed graphs that facilitate modeling and analyzing workflows. Finally, our formal framework allows checking the logical termination of workflows.

An important highlight of this paper is the emphasis on the tasks present in the workflow, which allows us to identify easily the dynamism present in the workflow. Finally, we describe the logical termination in a very intuitive form and we present conditions under which this property is valid.

We still provide a counter-example which shows that a conjecture presented in a previous article is false.

2. Workflow Modelling and Analysis

This section is devoted to the presentation of our main results. In particular, we start this section by providing the formal definition of a workflow. In other words, we furnish the formal structure of a business process. Notice that this workflow structure can be also found in [18–21]. It is also important to point out that this type of graphs has an input/output logic operator associated with each vertex. Further, we analyze each model present on the workflow and give special emphasis to the execution of all tasks present in a workflow. Besides, we will create new models based on the existing ones. Finally, we will describe conditions under which a workflow logical terminates. In conclusion, our approach allows us to provide a complete description of workflows.

Definition 1 (see [18–21]). A workflow is a trilogic acrylic directed graph $\mathrm{WG} = (T, A, A', M)$, 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 attributed 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. The transition $a_\sqcup$ is the tuple $(\sqcup, t_1)$ and transition $a_\sqcap$ is the tuple $(t_n, \sqcap)$, where the symbols $\sqcup$ and $\sqcap$ represent abstract tasks which indicate the entry and ending point of the workflow, respectively. Every transition a_i, $i \in \{1, \ldots, n\}$ corresponds to a tuple of the form (t_k, t_l), where $t_k, t_l \in T$.

We use the symbol $'$ to reference the label of a transition; that is, a_i' references transition a_i, $a_i \in A$. The elements a_i' are called Boolean terms and form the set A'.

Given $t_i \in T$, the incoming transitions for task t_i are the tuples of the form (t_l, t_i), $t_l \in T$, and the outgoing transitions are the tuples of the form (t_i, t_l), $t_l \in T$.

The incoming/outgoing condition of task t_i is the Boolean expression $a_{k_1}' \varphi \cdots \varphi a_{k_l}'$, where $\varphi \in \{\bullet, \otimes, \oplus\}$, $a_{k_1}', \ldots, a_{k_l}' \in A'$ and $a_{k_1}, \ldots, a_{k_l}$ are the incoming/outgoing transitions of task t_i. The terms $a_{k_1}', \ldots, a_{k_l}'$ are connected with the logic operator $\succ t_i$, $t_i \prec$, respectively. If task t_i has only one incoming/outgoing transition we assume that the condition does not have logic operator.

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. An EA model has the behavior with two distinct modes: when f_E is evaluated to true, f_C is also evaluated to true; when f_E is evaluated to false, f_C is always false. And the EA model $t_i : f_E \rightsquigarrow f_C$ is true if both f_E, f_C are true; otherwise it is false. We say that the EA model $t_i : f_E \rightsquigarrow f_C$ is positive if its Boolean value is true; otherwise it is said to be negative.

We denote by M the set of all EA models present in WG.

Task t_i is said to be executed if the EA model $t_i : f_E \rightsquigarrow f_C$ is positive. In this case, task t_i has attributed the Boolean value true.

Remark 2. Given an expression whose Boolean value is true (resp., false), we simply can represent this fact by 1, (resp., 0).

Remark 3. Given an EA model $t_i : f_E \rightsquigarrow f_C$, if f_E is false, then task t_i disables all its outgoing transitions. Consequently f_C is also false.

Notice that the workflow starts its execution by enabling transition $a_\sqcup$, that is, by asserting $a_\sqcup'$ to be true. In other words, the workflow starts its execution by executing task t_1.

Notice that a_i' is true if transition a_i is enabled; otherwise a_i is false. Transitions can be enabled by a user or by an external event. If the EA model $t_i : f_E \rightsquigarrow f_C$ is negative, then both f_E, f_C are false. In this case, all the transitions of f_C are disabled.

Example 4. In Figure 1 we present a workflow $\mathrm{WG} = (T, A, A', M)$, where $T = \{t_1, t_2, \ldots, t_9\}$, $A = \{a_\sqcup, a_\sqcap, a_1, a_2, \ldots, a_{11}\}$, $A' = \{a_\sqcup', a_\sqcap', a_1', a_2', \ldots, a_{11}'\}$, and $M = \{t_1 : a_\sqcup' \rightsquigarrow a_1' \bullet a_2', t_2 : a_1' \rightsquigarrow a_3' \oplus a_4', t_3 : a_2' \rightsquigarrow a_8', t_4 : a_3' \rightsquigarrow a_5' \oplus a_6', t_5 : a_4' \rightsquigarrow a_7', t_6 : a_5' \rightsquigarrow a_9', t_7 : a_6' \rightsquigarrow a_{10}', t_8 : a_7' \oplus a_9' \oplus a_{10}' \rightsquigarrow a_{11}', t_9 : a_8' \bullet a_{11}' \rightsquigarrow a_\sqcap'\}$.

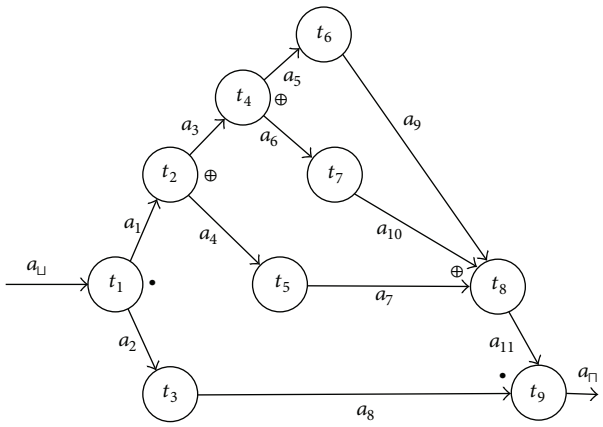

Figure 1: Example of a workflow.

The output logic operator of task t_2 ($t_2 \prec$) is XOR ($\oplus$), while the input logic operator of task t_9 ($\succ t_9$) is an AND ($\bullet$).

The incoming transition for task t_2 is $a_1 = (t_1, t_2)$ and its outgoing transitions are $a_3 = (t_2, t_4)$ and $a_4 = (t_2, t_5)$. Hence the incoming condition for task t_2 is a_1', while its outgoing condition is $a_3' \oplus a_4'$.

Task t_2 is executed if the EA model $t_2 : a_1' \rightsquigarrow a_3' \oplus a_4'$ is positive, that is, if a_1' is true and only one of the Boolean terms a_3', a_4' is true.

Notice that the workflow from Figure 1 corresponds to the following real situation. Indeed, it can represent the tasks necessary to be executed for a person driving a new car. Let us assume that tasks t_i, $i \in \{1, \ldots, 9\}$ have the following meanings:

t_1: deciding to purchase a new car to own use;

t_2: payment of the car;

t_3: getting the drivers license;

t_4: deciding to pay by credit;

t_5: deciding to pay without credit;

t_6: getting rental credit;

t_7: getting bank credit;

t_8: purchasing of the car;

t_9: driving the new car.

Clearly, the decision of purchasing a new car to own use implies to pay the car and to get the drivers license. The payment of the car implies the execution of only one of the situations to pay by credit or to pay without credit. And to get credit implies to get a rental credit or a bank credit. It is clear that the purchase of the car depends on the execution of only one of the tasks: decide to pay without credit, get rental credit, and get bank credit.

Hence, the possibility to drive a new car depends on the purchase of the car and in obtaining the drivers license. In other words, the execution of task t_9 depends on the execution of both tasks t_3, t_8.

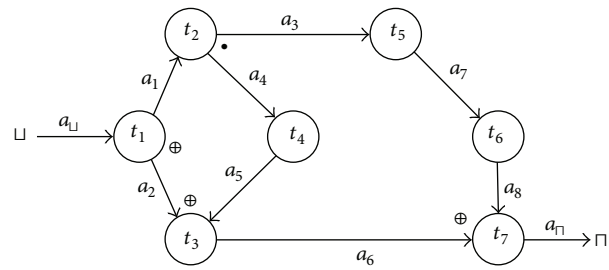

Figure 2: Example of a workflow.

Many other examples can be given. Indeed, too many situations in our life can be described by workflows. For example, the request for a credit card or a loan application is simple examples of workflows.

Proposition 5. *Let $WG = (T, A, A', M)$ be a workflow. Let $a_l = (t_i, t_j) \in A$, $t_i, t_j \in T$. If a_l' is true, then t_i is necessarily executed.*

Proof. Let us assume that a_l' is true. Let $t_i : f_{E_i} \rightsquigarrow f_{C_i}$ be the EA model associated to task t_i. If task t_i is not executed, then the EA model $t_i : f_{E_i} \rightsquigarrow f_{C_i}$ is negative. Since the EA model is negative, all outgoing transitions of task t_i are disabled; in particular a_l is disabled, that is, a_l' is false, wich is a contradiction. Hence task t_i is executed. □

Remark 6. The condition of Proposition 5 is not sufficient. For example in the workflow from Figure 1, if task t_2 is executed, then the EA model $t_2 : a_1' \rightsquigarrow a_3' \oplus a_4'$ is positive. For a_1' = true, a_3' = true, a_4' = false, and $a_4 = (t_2, t_5)$, t_2 is executed, but a_4' is false.

Remark 7. Let us consider the Boolean term a_l' where $a_l = (t_i, t_j) \in A$, $t_i, t_j \in T$. If a_l' is true, task t_j is not necessarily executed. For example, in the workflow from Figure 2, let us assume that $a_\sqcup'$ = true, a_1' = true, a_2' = false, a_3' = true, a_4' = true, a_5' = true, a_6' = true, a_7' = true, a_8' = true, and $a_\sqcap'$ = false. Hence, for this assignment the EA model $t_7 : a_6' \oplus a_8' \rightsquigarrow a_\sqcap'$ is negative, which means that task t_7 is not executed. Nevertheless, $a_8 = (t_6, t_7)$ and a_8' is true.

Next we introduce the concept of logical termination. This is a very important structural property, since its analysis will allow to verify if a workflow will eventually finish, according to the initial specifications.

Definition 8. Let $WG = (T, A, A', M)$ be a workflow. One says that WG logically terminates if task t_n is executed whenever task t_1 is executed.

In the following result we establish a necessary and sufficient condition for the logical termination.

Theorem 9. *Let $WG = (T, A, A', M)$ be a workflow. Then WG logically terminates if and only if $a_\sqcap'$ is true whenever $a_\sqcup'$ is true.*

Proof. Let us assume that WG logically terminates; that is, task t_n is executed whenever task t_1 is executed. This means that the EA model $t_n : f_{E_n} \rightsquigarrow a'_{\sqcap}$ is positive whenever the EA model $t_1 : a'_{\sqcup} \rightsquigarrow f_{C_1}$ is positive. Bearing in mind that WG starts its execution by executing task t_1, then the EA model $t_1 : a'_{\sqcup} \rightsquigarrow f_{C_1}$ is positive. Hence the EA model $t_n : f_{E_n} \rightsquigarrow a'_{\sqcap}$ is also positive. Consequently, $a'_{\sqcup}$, f_{C_1}, f_{E_n}, and $a'_{\sqcap}$ are true. Thus, $a'_{\sqcap}$ is true whenever $a'_{\sqcup}$ is true.

Conversely, let us assume that $a'_{\sqcap}$ is true whenever $a'_{\sqcup}$ is true. Let us assume that task t_1 is executed. This means that the EA model $t_1 : a'_{\sqcup} \rightsquigarrow f_{C_1}$ is positive. Bearing in mind that $a'_{\sqcap}$ is true, according to the behavior of the EA models, necessarily f_{E_n} is true. Hence the EA model $t_n : f_{E_n} \rightsquigarrow a'_{\sqcap}$ is positive, which means that task t_n is executed. So we can conclude that task t_n is executed whenever task t_1 is executed, which means that WG logically terminates. □

Example 10. It is not hard to check that, in the workflow from Figure 1, $a'_{\sqcap}$ is true whenever $a'_{\sqcup}$ is true. Thus, the workflow logically terminates.

Next we address our study on the dynamism present in a workflow. Obviously the dynamism is associated with the sequential execution of its tasks. In the workflow from Figure 1 the execution of task t_1 implies the execution of both tasks t_2, t_3; the execution of task t_2 implies the execution of only one of the tasks t_4, t_5; the execution of task t_4 implies the execution of only one of the tasks t_6, t_7; the execution of only one of the tasks t_5, t_6, and t_7 implies the execution of task t_8. Finally, the execution of both tasks t_3, t_8 implies the execution of task t_9. Hence, we can state the execution of task t_1 implies the execution of $t_2 \bullet t_3$; the execution of task t_2 implies the execution of $t_4 \oplus t_5$; the execution of task t_4 implies the execution of $t_6 \oplus t_7$; the execution of $t_5 \oplus t_6 \oplus t_7$ implies the execution of task t_8; the execution of $t_3 \bullet t_8$ implies the execution of task t_9. Notice that when we consider $t_2 \bullet t_3$, the operator $\bullet$ is the output logic operator of task t_1, while when we consider $t_5 \oplus t_6 \oplus t_7$, $\oplus$ is the input logic operator of task t_8.

These remarks led us to introduce the following concept.

Definition 11. Let WG $= (T, A, A', M)$ be a workflow. The compound tasks of WG are the elements of the following form: $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}, t_{i_1}, t_{i_2}, \ldots, t_{i_k} \in T$, $\varphi \in \{\bullet, \otimes, \oplus\}$. The set of all compound tasks of WG is denoted by T'; that is,

$$T' = \left\{ t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k} : t_{i_1}, t_{i_2}, \ldots, t_{i_k} \in T,\ \varphi \in \{\bullet, \otimes, \oplus\} \right\}. \quad (1)$$

Example 12. In the workflow from Figure 1, $T' = \{t_2 \bullet t_3, t_4 \oplus t_5, t_6 \oplus t_7, t_5 \oplus t_6 \oplus t_7, t_3 \bullet t_8\}$.

Remark 13. Since every task t_i has associated a Boolean value, according to its execution, it is also natural to attribute a Boolean value to the compound tasks of WG. The natural attribution is the following. Given any compound task of WG, $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}, \varphi \in \{\bullet, \otimes, \oplus\}$.

If $\varphi = \bullet$, then the Boolean value of $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}$ is 1 if and only if the Boolean value of all tasks $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ is equal to 1.

If $\varphi = \otimes$, then the Boolean value of $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}$ is 1 if and only if there exists at least one of the tasks $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ whose Boolean value is equal to 1.

If $\varphi = \oplus$, then the Boolean value of $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}$ is 1 if and only if there exists only one of the tasks $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ with Boolean value equal to 1.

Naturally, we can state that a compound task $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}$ is executed if and only if its Boolean value is equal to 1, which means that the compound task $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}$ is positive. In other words, $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k}$ is executed if and only if, one of the following cases holds:

If $\varphi = \bullet$, all tasks $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ are executed.

If $\varphi = \otimes$, at least one of the tasks $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ is executed.

If $\varphi = \oplus$, only one of the tasks $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ is executed.

In what follows, we introduce a new type of task models that we designate by compound task models.

Definition 14. Let WG $= (T, A, A', M)$ be a workflow. Let t_i, $t_j, t_{i_1}, t_{i_2}, \ldots, t_{i_k}, t_{j_1}, t_{j_2}, \ldots, t_{j_l} \in T, \varphi, \psi\{\bullet, \otimes, \oplus\}$. A compound task model is an implication with one of the following forms:

(1) $t_i \hookrightarrow t_{j_1}\psi t_{j_2}\psi \cdots \psi t_{j_l}$;

(2) $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k} \hookrightarrow t_j$;

(3) $t_{i_1}\varphi t_{i_2}\varphi \cdots \varphi t_{i_k} \hookrightarrow t_{j_1}\psi t_{j_2}\psi \cdots \psi t_{j_l}$.

Usually we represent a compound task model by $t_{I_i} \hookrightarrow t_{O_i}$, where t_{I_i} is called the incoming task and t_{O_i} is called the outgoing task. We say that a compound task model $t_{I_i} \hookrightarrow t_{O_i}$ is positive if both incoming and outgoing tasks are positive, that is, if both tasks t_{I_i}, t_{O_i} are executed.

In particular, the implication of the form $t_i \hookrightarrow t_j$ is called a simple task model. Clearly, it is positive if both tasks t_i, t_j are executed.

The set of all simple and compound task models present in WG is called the set of task models of WG and is denoted by TM.

The task models have the behavior with two distinct modes: if its incoming task is true, necessarily its outgoing task is true; if the incoming task is false, the outgoing task is false. In other words, if $t_{I_i} \hookrightarrow t_{O_i}$ is a compound task model, then t_{I_i} is executed if and only if t_{O_i} is executed.

Notice that, in a compound task model $t_{I_i} \hookrightarrow t_{O_i}$, at least one of the tasks t_{I_i}, t_{O_i} is compound.

Example 15. In the workflow from Figure 1, the set of its task models is TM $= \{t_1 \hookrightarrow t_2 \bullet t_3, t_2 \hookrightarrow t_4 \oplus t_5, t_4 \hookrightarrow t_6 \oplus t_7, t_5 \oplus t_6 \oplus t_7 \hookrightarrow t_8, t_3 \bullet t_8 \hookrightarrow t_9\}$.

From now on, we use the symbol $\leftrightarrow$ with the following meaning: $X \leftrightarrow Y$ means that the compound statements X and Y are logically equivalent.

According to simple rules of Logic and taking into account the behavior of the task models, we can infer the following result. And the establishment of this result allows us to identify new task models present in the workflow.

In what follows we establish some properties that will allow us to create new task models based on the existing ones.

Proposition 16. *Let $WG = (T, A, A', M)$ be a workflow. Suppose that the task models $t_{I_i} \hookrightarrow t_{O_i}$ and $t_{O_i} \hookrightarrow t_{O_j}$ belong to TM. Then the model $t_{I_i} \hookrightarrow t_{O_j}$ still hods in WG.*

Proof. The proof is trivial. □

Theorem 17. *Let $WG = (T, A, A', M)$ be a workflow.*

(a) *If both task models*

$$t_{I_i} \hookrightarrow t_{O_i}, \tag{2}$$

$$t_{I_j} \hookrightarrow t_{O_j} \tag{3}$$

belong to TM, where $t_{O_i} \leftrightarrow t_{I_j}$, then the model $t_{I_i} \hookrightarrow t_{O_j}$ still holds in WG.

(b) *If both task models $t_{I_i} \hookrightarrow t_{O_i}$ and $t_{I_j} \hookrightarrow t_{O_j}$ belong to TM, where $t_{O_i} \leftrightarrow t_L \varphi t_{I_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the compound task model $t_{I_i} \hookrightarrow t_L \varphi t_{O_j}$ still holds in WG.*

(c) *If both task models $t_{I_i} \hookrightarrow t_{O_i}$ and $t_{O_j} \hookrightarrow t_{I_j}$ belong to TM, where $t_{O_i} \leftrightarrow t_L \varphi t_{I_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the compound task model $t_{I_i} \hookrightarrow t_L \varphi t_{O_j}$ still holds in WG.*

(d) *If both task models $t_{I_i} \hookrightarrow t_{O_i}$ and $t_{I_j} \hookrightarrow t_{O_j}$ belong to TM, where $t_{I_i} \leftrightarrow t_L \varphi t_{I_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the compound task model $t_L \varphi t_{O_j} \hookrightarrow t_{O_i}$ still holds in WG.*

(e) *If both task models $t_{I_i} \hookrightarrow t_{O_i}$ and $t_{O_j} \hookrightarrow t_{I_j}$ belong to TM, where $t_{I_i} \leftrightarrow t_L \varphi t_{I_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the compound task model $t_L \varphi t_{O_j} \hookrightarrow t_{O_i}$ still holds in WG.*

Proof. (a) Let us assume that both task models (2) and (3) belong to TM. Notice that if either (2) or (3) is negative, since $t_{O_i} \leftrightarrow t_{I_j}$, then necessarily both task models (2) and (3) are negative. So, we just need to verify the result when both task models (2) and (3) are positive.

Let us assume that both tasks models (2) and (3) are positive. Since (2) is positive, necessarily both t_{I_i}, t_{O_i} are true. In other words, some of the tasks from t_{I_i}, t_{O_i} are executed allowing that t_{I_i}, t_{O_i} are executed. Bearing in mind that $t_{O_i} \leftrightarrow t_{I_j}$, then t_{I_j} is true. So according to the behavior of the task models, necessarily t_{O_j} is true. Hence we can state that t_{O_j} is executed, whenever t_{I_j} is executed. Therefore the model $t_{I_i} \hookrightarrow t_{O_j}$ still holds in WG.

Now, in order to prove (b) and (c), we start with the following argument. Bearing in mind that $t_{O_i} \leftrightarrow t_L \varphi t_{I_j}$, according to the behavior of the task models, we can consider that $t_{O_i} \hookrightarrow t_L \varphi t_{I_j}$ still holds in WG. Since $t_{I_i} \hookrightarrow t_{O_i}$ and $t_{O_i} \hookrightarrow t_L \varphi t_{I_j}$ holds in WG, according to Proposition 16 we can conclude that

$$t_{I_i} \hookrightarrow t_L \varphi t_{I_j} \tag{4}$$

still holds in WG.

(b) Taking into account that $t_{I_j} \hookrightarrow t_{O_j}$ belong to TM, and consequently t_{I_j}, t_{O_j} have the same Boolean value, we can replace t_{I_j} by t_{O_j} in (4), obtaining the new model $t_{I_i} \hookrightarrow t_L \varphi t_{O_j}$, which means that $t_{I_i} \hookrightarrow t_L \varphi t_{O_j}$ still holds in WG.

(c) Taking into account that $t_{O_j} \hookrightarrow t_{I_j}$ belong to TM, and consequently t_{O_j}, t_{I_j} have the same Boolean value, we can replace t_{I_j} by t_{O_j} in (4), obtaining the new model $t_{I_i} \hookrightarrow t_L \varphi t_{O_j}$, which means that $t_{I_i} \hookrightarrow t_L \varphi t_{O_j}$ still holds in WG.

Analogously, in order to prove (d) and (e) we start by making the next statement. Taking into account that $t_{I_i} \leftrightarrow t_L \varphi t_{I_j}$, according to the behavior of the compound task models, we can consider that $t_L \varphi t_{I_j} \hookrightarrow t_{I_i}$ holds in WG. Since $t_L \varphi t_{I_j} \hookrightarrow t_{I_i}$ and $t_{I_i} \hookrightarrow t_{O_i}$ hold in WG, according to Proposition 16 we can conclude that

$$t_L \varphi t_{I_j} \hookrightarrow t_{O_i} \tag{5}$$

still holds in WG.

(d) Bearing in mind that $t_{I_j} \hookrightarrow t_{O_j}$ holds in WG, and consequently t_{I_j}, t_{O_j} have the same Boolean value, we can replace t_{I_j} by t_{O_j} in (5), obtaining the new model $t_L \varphi t_{O_j} \hookrightarrow t_{O_i}$, which means that $t_L \varphi t_{O_j} \hookrightarrow t_{O_i}$ still holds in WG.

(e) Bearing in mind that $t_{O_j} \hookrightarrow t_{I_j}$ holds in WG, and consequently t_{O_j}, t_{I_j} have the same Boolean value, we can replace t_{I_j} by t_{O_j} in (5), obtaining the new model $t_L \varphi t_{O_j} \hookrightarrow t_{O_i}$, which means that $t_L \varphi t_{O_j} \hookrightarrow t_{O_i}$ still holds in WG. □

The previous results allow us to identify new task models based on the existing ones, as it is described below.

Definition 18. Let $WG = (T, A, A', M)$ be a workflow. An extended task model is a model obtained by applying a finite sequence of some of the properties established in Proposition 16 and Theorem 17. One adopts the notation TM' to represent the set of all extended task models of WG.

Example 19. In the workflow from Figure 1, bearing in mind that $t_1 \hookrightarrow t_2 \bullet t_3, t_2 \hookrightarrow t_4 \oplus t_5 \in TM$, according to Theorem 17 we can conclude that the model $t_1 \hookrightarrow (t_4 \oplus t_5) \bullet t_3$ still holds in WG. Therefore, we can state that $t_1 \hookrightarrow (t_4 \oplus t_5) \bullet t_3$ is an extended task model of WG.

Notice we adopt the same notation of the task models to represent the extended task models. Furthermore, the extended task models verify the same properties of the task models. In particular, given an extended task model $B \hookrightarrow C$, necessarily both B, C have the same Boolean value.

Naturally when we consider the set of all task models and extended task models presented in WG, we obtain all possible models that can be generated by the task models of the workflow. This set of task models will be designated by the closure of TM. In certain sense, we can state that the set of all task models from WG is a set of generators of all possible models of WG.

Definition 20. Let $WG = (T, A, A', M)$ be a workflow. One defines the closure of TM as the set of all task models and extended task models in WG. This set is denoted by TM^*. In other words, $TM^* = TM \cup TM'$.

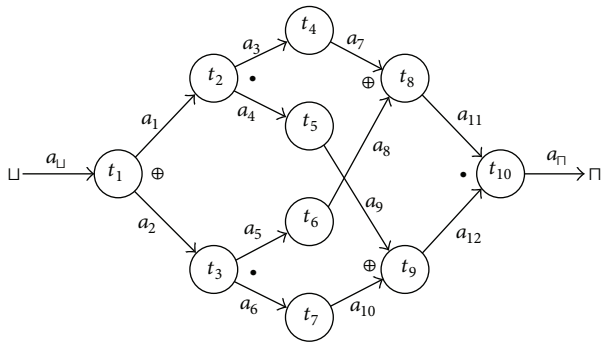

Figure 3: Example of a workflow.

Example 21. As we saw in Example 15 in the workflow from Figure 1, $\mathrm{TM} = \{t_1 \hookrightarrow t_2 \bullet t_3, t_2 \hookrightarrow t_4 \oplus t_5, t_4 \hookrightarrow t_6 \oplus t_7, t_5 \oplus t_6 \oplus t_7 \hookrightarrow t_8, t_3 \bullet t_8 \hookrightarrow t_9\}$. Since $t_1 \hookrightarrow t_2 \bullet t_3, t_2 \hookrightarrow t_4 \oplus t_5 \in \mathrm{TM}$, according to Theorem 17 we can deduce that $t_1 \hookrightarrow (t_4 \oplus t_5) \bullet t_3 \in \mathrm{TM}^*$. Now bearing in mind that $t_4 \hookrightarrow t_6 \oplus t_7 \in \mathrm{TM}$, applying again Theorem 17 we can conclude that $t_1 \hookrightarrow ((t_6 \oplus t_7) \oplus t_5) \bullet t_3 \in \mathrm{TM}^*$. As $(t_6 \oplus t_7) \oplus t_5 \leftrightarrow t_5 \oplus t_6 \oplus t_7$ we can state that $t_1 \hookrightarrow (t_5 \oplus t_6 \oplus t_7) \bullet t_3 \in \mathrm{TM}^*$. Bearing in mind that $t_5 \oplus t_6 \oplus t_7 \hookrightarrow t_8$, applying once more Theorem 17 we infer that $t_1 \hookrightarrow t_8 \bullet t_3 \in \mathrm{TM}^*$. As $t_8 \bullet t_3 \leftrightarrow t_3 \bullet t_8$, applying again Theorem 17 we conclude that $t_1 \hookrightarrow t_9 \in \mathrm{TM}^*$.

Notice the workflow from Figure 1 logically terminates and $t_1 \hookrightarrow t_9 \in \mathrm{TM}^*$. Furthermore, we studied many other examples of workflows that logically terminates and simultaneously $t_1 \hookrightarrow t_n \in \mathrm{TM}^*$. The analysis of these different cases led us to formulate the following conjecture.

Conjecture 22 (see [21]). *Given a workflow $WG = (T, A, A', M)$, then WG logically terminates if and only if $t_1 \hookrightarrow t_n \in TM^*$.*

After the analysis of many cases, we start believing that the conjecture was true. Nevertheless it is false, as the following example proves. Let us consider the following workflow.

Example 23. It is not hard to check that the workflow from Figure 3 logically terminates; nevertheless the condition $t_1 \hookrightarrow t_n \in \mathrm{TM}^*$ is not valid.

Indeed the condition of the Conjecture 22 is not necessary. However it is sufficient, as we prove in the following result.

Theorem 24. *Let $WG = (T, A, A', M)$ be a workflow. If $t_1 \hookrightarrow t_n \in TM^*$, then WG logically terminates.*

Proof. Let us assume that $t_1 \hookrightarrow t_n \in \mathrm{TM}^*$.

Case 1. Suppose that $t_1 \hookrightarrow t_n \in \mathrm{TM}$. This means that WG has the following structure:

$$\sqcup \xrightarrow{a_{\sqcup}} t_1 \xrightarrow{a_{\sqcap}} t_{\sqcap}. \tag{6}$$

In this case, $t_1 = t_n$. Since WG starts its execution by executing task t_1 we can conclude that task t_n is executed whenever task t_1 is executed, which means that WG logically terminates.

Case 2. Suppose that $t_1 \hookrightarrow t_n \in \mathrm{TM}^* \setminus \mathrm{TM} = \mathrm{TM}'$. So, $t_1 \hookrightarrow t_n$ was obtained by applying some of the results established in Proposition 16 and Theorem 17. Since $t_1 \hookrightarrow t_n \in \mathrm{TM}^*$ and the workflow starts its execution by executing task t_1, that is, by asserting t_1 to be true, according to the behavior of the extended models, necessarily t_n is still asserted to be true. This means that task t_n is executed. Hence, task t_n is executed whenever task t_1 is executed. Thus WG logically terminates. □

3. Conclusions

In this paper we develop a formalism to describe and analyse the structure of workflows, based on graphs and Propositional Logic. Indeed, we describe the structure of a workflow as a graph whose vertices represent tasks and the arcs are associated to workflow transitions. To each task an input/output logic operator is associated and this logic operator can be the logical AND (•), the OR (⊗), or the XOR—exclusive-or—(⊕). Furthermore, we associate a Boolean term to each transition present in the workflow.

It is important to point out that our main emphasis is the analysis of a workflow through the study of its task models which allows us to describe the dynamism of the workflow in a very simple and intuitive way.

Another relevant aspect of our approach is the introduction of the concept of compound tasks. This concept allows us to identify new task models based on the existing ones. Through these new task models we are able to describe the dynamism present in a workflow in a very simple way. Clearly, the study of the dynamism of a workflow is equivalent to analyse the sequential execution of its tasks.

We still analyze the concept of logical termination and we provide necessary and sufficient conditions under which this property is valid.

Finally, given a workflow WG we provide a counterexample which shows that the conjecture of $t_1 \hookrightarrow t_n \in \mathrm{TM}^*$ being a necessary and sufficient condition under which WG logically terminates is false. In fact, the condition is necessary; nevertheless it is not sufficient.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] J. Cao, C. Chan, and K. Chan, "Workflow analysis for web publishing using a stage-activity process model," *Journal of Systems and Software*, vol. 76, no. 3, pp. 221–235, 2005.

[2] M. Hu, J. Zhang, and X. Chen, "Grid worflow for decision resource scheduling," in *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*, pp. 533–539, Hangzhou, China, April 2006.

[3] J. L. Kmetz, *Mapping Workflows and Managing Knowledge: Simply, Sensibly, Flexibly, and Without Software*, 2010.

[4] P. Muth, D. Wodtke, J. Weissenfels, G. Weikum, and K. Dittrich, "Enterprise-wide workflow management based on state and activity charts," in *Workflow Management Systems and Interoperability*, vol. 164 of *NATO ASI Series*, pp. 281–303, Springer, Berlin, Germany, 1998.

[5] U. Dayal, M. Hsu, and R. Ladin, "Organizing long-running activities with triggers and transactions," in *Proceedings of the ACM SIGMOD International Conference on Management of Data Table of Contents*, pp. 204–214, ACM Press, Atlantic City, NJ, USA, 1990.

[6] J. Eder, H. Groiss, and H. Nekvasil, "A workflow system based on active databases," in *Proceedings of the 9th Austrian-Informatics Conference on Workflow Management: Challenges, Paradigms and Products (CON '94)*, G. Chroust and A. Benczur, Eds., pp. 249–265, Linz, Austria, 1994.

[7] W. M. P. van der Aalst, "The application of petri nets to workflow management," *Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[8] W. M. P. van der Aalst, "Workflow verification: finding control-flow errors using petri-net-based techniques," in *Business Process Management*, W. M. P. van der Aalst, J. Desel, and A. Oberweis, Eds., vol. 1806 of *Lecture Notes in Computer Science*, pp. 161–183, Springer, Berlin, Germany, 2000.

[9] F. Gottschalk, W. M. P. van der Aalst, M. H. Jansen-Vullers, and M. La Rosa, "Configurable workflow models," *International Journal of Cooperative Information Systems*, vol. 17, no. 2, pp. 177–221, 2008.

[10] F. Gottschalk, W. M. P. van der Aalst, M. H. Jansen-Vullers, and H. M. W. Verbeek, "Protos2CPN: using colored Petri nets for configuring and testing business processes," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 1, pp. 95–110, 2008.

[11] H. M. W. Verbeek, W. M. P. van der Aalst, and A. H. M. Hofstede, "Verifying workflows with cancellation regions and or-joins: an approach based on relaxed soundness and invariants," *Computer Journal*, vol. 50, no. 3, pp. 294–314, 2007.

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

[13] J. Klingemann, J. Wasch, and K. Aberer, "Deriving service models in cross-organizational workflows," in *Proceedings of the 9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE '99)*, pp. 100–107, Sydney, Australia, March 1999.

[14] A. H. M. ter Hofstede and E. R. Nieuwland, "Task structure semantics through process algebra," *Software Engineering Journal*, vol. 8, no. 1, pp. 14–20, 1993.

[15] M. P. Singh and K. van Hee, "Semantical considerations on workflows: an algebra for intertask dependencies," in *Proceedings of the 5th International Workshop on Database Programming Languages (DBLP '95)*, Springer, Umbria, Italy, 1995.

[16] B. B. Manish, A. Bhardwaj, and A. P. S. Rathore, "Six sigma methodology utilization in telecom sector for quality improvement—a DMAIC process," *International Journal of Engineering Science and Technology*, vol. 2, no. 12, pp. 7653–7659, 2010.

[17] S. H. Han, M. J. Chae, K. S. Im, and H. D. Ryu, "Six sigma-based approach to improve performance in construction operations," *Journal of Management in Engineering*, vol. 24, no. 1, pp. 21–31, 2008.

[18] G. Cravo, "Applications of propositional logic to workflow analysis," *Applied Mathematics Letters*, vol. 23, no. 3, pp. 272–276, 2010.

[19] G. Cravo, "Logical termination of workflows: an interdisciplinary approach," *Journal of Numerical Analysis, Industrial and Applied Mathematics*, vol. 5, no. 3-4, pp. 153–161, 2011.

[20] G. Cravo and J. Cardoso, "Termination of workflows: a snapshot-based approach," *Mathematica Balkanica*, vol. 21, no. 3-4, pp. 233–243, 2007.

[21] G. Cravo, "Workflow analysis—a task model approach," in *Proceedings of the 2014 International Conference om Pure Mathematics, Applied Mathematics, Computational Methods (PMAMCM '14)*, E. Nikos, E. Mastorakis, M. Panos, R. P. Agarwal, and L. Kocinac, Eds., Mathematics and Computers in Science and Engineering Series 29, pp. 2227–4588, July 2014.