TERMINATION OF WORKFLOWS

Glória Cravo, Jorge Cardoso
Departamento de Matemática e Engenharias,
Universidade da Madeira, 9000-390 Funchal, Portugal.
{gcravo, jcardoso}@uma.pt

ABSTRACT

A workflow is a set of activities usually organized using a graph structure that has one beginning and one end. A workflow includes human participants and software applications that have the responsibility to carry out activities. A workflow is known to be the formal definition of the process used to manage business processes (e.g., sales order processing, article reviewing, member registration, etc). In this paper we describe and analyze the behavior of workflows using graph theory to verify an important property: their termination. It is essential to formally verify if a workflow, such as a sales order processing, will eventually terminate and be completed. We verify the termination of workflows using a new approach based on what we call snapshot-based theory.

1 Introduction

In this paper we describe and analyze the behavior of workflows using graph theory. A workflow is an abstraction of a business process that consists of one or more activities that need to be executed to complete a business process (for example, sales order processing, article reviewing, member registration, etc). Activities are represented with vertices and the partial ordering of activities is modeled with arcs, known as transitions. Each task of a workflow represents a unit of work to be executed by a computer program or a person. Workflows allow organizations to streamline and automate business processes, reengineer their structure, as well as, increase efficiency and reduce costs.

In the last decade, important advancements have been accomplished in the development of theoretical foundations to allow workflow modeling, verification, and analysis. Several formal modeling methods have been proposed to model workflows, such as graph theory [8], State and Activity Charts [9], Event-Condition-Action rules [4, 5], Petri Nets [1], Temporal Logic [2], Markov chains [7] and Process and Event Algebras [6, 10].

Despite the existence of several formal methods to model workflows, a vast number of widely well-known commercial workflow systems, such as TIBCO Workflow (www.tibco.com) and METEOR-S [8], have decided to use graphs to model their workflows.

While important advancements have been accomplished in the development of theoretical foundations for workflow modeling, verification, and analysis (especially in the context of Petri Nets [1]) more research is required especially with respect to the modeling and analysis of workflows using graphs.

Therefore, in this paper we present a formal framework, based on graphs theory, to check the termination of workflows. Termination is an important property for workflows because it is indispensable to know if a business process, such as a loan application or insurance claim, will eventually be completed. In our approach we model workflows with tri-logic acyclic directed graphs and develop a formalism to verify the logical termination of workflows. Our formalism uses a snapshot-based methodology which captures the different behaviors that a workflow may have.

2 Logical Termination

Definition 1 *A workflow is a tri-logic acyclic direct 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'*.*

An example of a workflow is presented in Figure 1. For more details and practical examples see [3].

Definition 2 *The incoming transitions for task* $t_i \in T$ *are the tuples of the form* $a_j = (x, t_i), x \in T, a_j \in A$*, and the outgoing transitions for task* t_i *are the tuples of the form* $a_l = (t_i, y), y \in T, a_l \in A$*.*

Definition 3 *The incoming condition for task* $t_i \in T$ *is a Boolean expression with terms* $a' \in A'$*, where* a *is an incoming transition of task* t_i*. The terms* a' *are connected with the logical operator* $\succ t_i$*. If the task has only one incoming transition then the condition does not have a logical operator.*

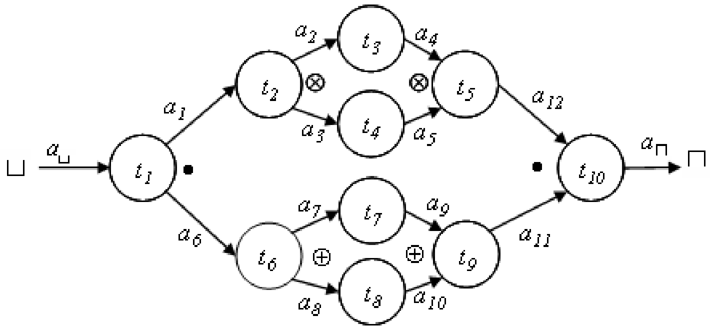


Figure 1: Example of a tri-logic acyclic direct graph (i.e., a workflow)

Definition 4 *The outgoing condition for task $t_i \in T$ is a Boolean expression with terms $a' \in A'$, where a is an outgoing transition of task t_i. The terms a' are connected with the logical operator $t_i \prec$. If the task has only one outgoing transition then the condition does not have a logical operator.*

Definition 5 *Given a workflow $WG = (T, A)$, an Event-Action (EA) model for a task $t_i \in T$ 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. For any EA model $t_i : f_E \rightsquigarrow f_C$, f_E and f_C have always the same Boolean value.*

Examples of the above definitions can be found in [3].

Definition 6 *Let WG 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 $true$.*

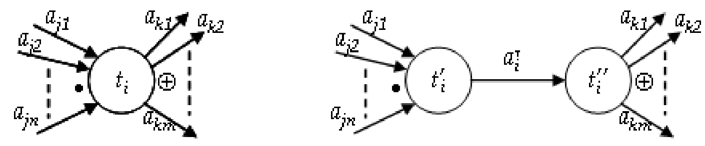
(2) *Let $t_1 : a'_\sqcup \rightsquigarrow f_{C_1}$. Then f_{C_1} has the Boolean value of $a'_\sqcup$, i.e., since the workflow starts its execution, f_{C_1} is always $true$.*

(3) *The workflow correctly terminates when $a'_\sqcap$ is asserted to $true$.*

Since the behavior of a workflow is determined by its *EA* models a natural concern is the exhaustive study of the *EA* models. We start by defining three different types of *EA* models.

Definition 7 *An EA model $f_E \rightsquigarrow f_C$ is said to be simple if $f_E = a'_i$ and $f_C = a'_j$, $i, j \in \{\sqcup, \sqcap, 1, \ldots, m\}$, with $i \neq j$.*

Definition 8 *An EA model $f_E \rightsquigarrow f_C$ is said to be complex if $f_E = a'_i$ and $f_C = a'_{j_1} \varphi a'_{j_2} \varphi \ldots \varphi a'_{j_k}$, or $f_E = a'_{j_1} \varphi a'_{j_2} \varphi \ldots \varphi a'_{j_k}$ and $f_C = a'_i$, where $\varphi \in \{\otimes, \bullet, \oplus\}$.*



a) Hybrid EA model b) Derived Complex EA models

Figure 2: Splitting a hybrid *EA* model into two equivalent complex *EA* models

Definition 9 *An EA model* $f_E \rightsquigarrow f_C$ *is said to be hybrid if* $f_E = a'_{i_1}\varphi a'_{i_2}\varphi \dots \varphi a'_{i_l}$ *and* $f_C = a'_{j_1}\psi a'_{j_2}\psi \dots \psi a'_{j_k}$*, where* $\varphi, \psi \in \{\otimes, \bullet, \oplus\}$.

The study of simple *EA* models is very easy. Our concern is to study complex and hybrid *EA* models. In the following result we establish a connection between hybrid and complex *EA* models.

Theorem 10 *A hybrid EA model* $f_E \rightsquigarrow f_C$ *can be split into two derived equivalent complex EA models* $f_E \rightsquigarrow a_i^*$ *and* $a_i^* \rightsquigarrow f_C$.

Proof. Suppose that $t_i : f_E \rightsquigarrow f_C$ is a hybrid *EA* model (Figure 2.a). Then both f_E and f_C are Boolean terms with an AND ($\bullet$), an OR ($\otimes$), or a XOR ($\oplus$). Let us create two auxiliary tasks t_i', t_i'' and an auxiliary transition $a_i^{\intercal} = (t_i', t_i'')$. Let a_i^* be the Boolean term associated with the auxiliary transition $a_i^{\intercal}$, such that a_i^* has the same Boolean value of f_E. Let $t_i' : f_E \rightsquigarrow a_i^*$ and $t_i'' : a_i^* \rightsquigarrow f_C$ be new *EA* models. Since a_i^* has the same Boolean value of f_E and, as a consequence, f_C has its Boolean value depending on the Boolean value of a_i^*, when we consider these new *EA* models instead of the initial hybrid *EA* model, the behavior of the workflow is not modified (Figure 2.b). Clearly the new *EA* models $f_E \rightsquigarrow a_i^*$ and $a_i^* \rightsquigarrow f_C$ are complex and so the result is satisfied. ∎

Definition 11 *A hybrid workflow is a workflow that contains hybrid EA models. A workflow is said to be non-hybrid if it contains only simple and complex EA models, i.e., no hybrid EA models exist.*

Example 12 *The workflow from Figure 1 is non-hybrid.*

Definition 13 *A hybrid workflow* WG *is said to be equivalent to a non-hybrid workflow* WG' *if* WG' *is obtained from* WG *by decomposing all hybrid EA models of* WG *into equivalent derived complex EA models.*

Theorem 14 *A hybrid workflow can be transformed into an equivalent non-hybrid workflow.*

Table 1: *EA* Models structures

EA model structure	*EA* model name	*EA* model type
$t_u : a'_{i_1} \bullet a'_{i_2} \bullet \cdots \bullet a'_{i_k} \rightsquigarrow a'_l$	AND-join	Complex
$t_u : a'_i \rightsquigarrow a'_{j_1} \bullet a'_{j_2} \bullet \cdots \bullet a'_{j_l}$	AND-split	Complex
$t_u : a'_{i_1} \oplus a'_{i_2} \oplus \cdots \oplus a'_{i_k} \rightsquigarrow a'_l$	XOR-join	Complex
$t_u : a'_i \rightsquigarrow a'_{j_1} \oplus a'_{j_2} \oplus \cdots \oplus a'_{j_l}$	XOR-split	Complex
$t_u : a'_{i_1} \otimes a'_{i_2} \otimes \cdots \otimes a'_{i_k} \rightsquigarrow a'_l$	OR-join	Complex
$t_u : a'_i \rightsquigarrow a'_{j_1} \otimes a'_{j_2} \otimes \cdots \otimes a'_{j_l}$	OR-split	Complex
$t_u : a'_i \rightsquigarrow a'_l$	Sequence	Simple

Proof. Follows immediately from Theorem 10 and Definition 13. ■

Since a hybrid workflow can be transformed into a non-hybrid workflow, in this paper we will address only the study of non-hybrid workflows. When no ambiguity can arise we will refer to non-hybrid workflows simply as workflows. As we will consider only non-hybrid workflows, the behavior of a workflow will depend on its complex and simple *EA* models.

A non-hybrid workflow can contain seven different *EA* model structures: AND-join, AND-split, XOR-join, XOR-split, OR-join, OR-split and Sequence. Table 1 illustrates the structure of these seven different *EA* models.

These *EA* models can be classified as deterministic and non-deterministic. The AND-join, AND-split, XOR-join, OR-join and Sequence models are deterministic, while XOR-split and OR-split are non-deterministic.

For any **deterministic model** $t_u : f_E \rightsquigarrow f_C$ knowing that the Boolean value of the incoming condition f_E is $true$ allows us to infer that all its outgoing transitions will be set to $true$. Consequently, in these cases we know which task(s) will be executed after t_u (i.e., connected to t_u).

For any **non-deterministic model** $t_u : f_E \rightsquigarrow f_C$ knowing that the Boolean value of the only incoming transition of f_E is $true$ does not allow us to infer which outgoing transition(s) will be set to $true$. Nevertheless, we know that if f_E is $true$ then f_C is also $true$. Let us analyze each case individually.

(1) XOR-split. In this case, if f_E is $true$, we just know that only one of the outgoing transitions a'_{j_r}, $r \in \{1, \ldots, l\}$, is $true$.

(2) OR-split. In this case, if f_E is $true$, we only know that a nonempty subset of the outgoing transitions a'_{j_r}, $r \in \{1, \ldots, l\}$, are $true$.

In these two cases, knowing that f_E is $true$ does not allow us to infer which task(s) will be executed after t_u (i.e., connected to t_u). Therefore, we call these models non-deterministic.

Definition 15 *A non-deterministic task is a task associated with a XOR-split or OR-*

split model (see Table 1).

Definition 16 *All transitions have a Boolean label a'_i that references the transitions a_i (definition 1). Additionally, each outgoing transition of a task associated with a XOR-split or OR-split models has a snapshot Boolean variable denoted by $\vec{a}_i$, which is related to the non-determinism of the task.*

Definition 17 *The non-deterministic task behavior $(t^{ND}(t_i))$ of a non-deterministic task t_i is the set of all snapshot Boolean variables associated with its outgoing transitions, i.e., $t^{ND}(t_i) = \{\{\vec{a}_{j_1}, \vec{a}_{j_2}, \dots, \vec{a}_{j_l}\} | t_i : f_E \rightsquigarrow f_C, f_E = a_i$ and $f_C = a_{j_1} \varphi a_{j_2} \varphi ... \varphi a_{j_l}$, $\varphi \in \{\otimes, \oplus\}\}$.*

Definition 18 *The non-deterministic workflow behavior, denoted by $w^{ND}(WG)$, of a workflow WG is the set of all non-deterministic task behaviors of the workflow, i.e., $w^{ND}(WG) = \{t^{ND}(t_{i_1}), t^{ND}(t_{i_2}), \dots, t^{ND}(t_{i_k})\}$, where $t_{i_1}, t_{i_2}, \dots, t_{i_k} \in T$, are the non-deterministic tasks.*

Definition 19 *Let t_i be a non-deterministic task. Let $P \dot{\cup} N$ be a partition of $t^{ND}(t_i)$ such that $P = \{\vec{a} \in t^{ND}(t_i) |\ \vec{a}$ is a snapshot Boolean variable asserted to true$\}$ and let $N = \{\vec{a} \in (t^{ND}(t_i) \backslash P) | \vec{a}$ is a snapshot Boolean variable asserted to false$\}$. Let $P' \dot{\cup} N'$ be a partition of $t^{ND}(t_i)$ such that $P' = \{\vec{a} \in 2^{t^{ND}(t_i) \backslash \emptyset} | \vec{a}$ is a snapshot Boolean variable asserted to true$\}$ and let $N' = \{\vec{a} \in (2^{t^{ND}(t_i) \backslash \emptyset} \backslash P) | \vec{a}$ is a snapshot Boolean variable asserted to false$\}$. A snapshot of t_i, denoted by $tss(t_i)$ is a set of asserted snapshot Boolean variables such that, if t_i is a XOR-split then $tss(t_i) = P \dot{\cup} N$; if t_i is an OR-split then $tss(t_i) = P' \dot{\cup} N'$.*

Remark 20 *Clearly $P \cap N = P' \cap N' = \emptyset$. Note that P, P' and N are always nonempty sets, but N' can be empty. When N' is empty, it means that all the Boolean terms of the outgoing condition of the task t_i are true, i.e., all snapshot Boolean variables are asserted to true.*

Notation 21 *We denote by $tss(t_i) \circlearrowleft t^{ND}(t_i)$ to specify that $tss(t_i)$ is a snapshot with all snapshot Boolean variables in $t^{ND}(t_i)$.*

Example 22 *The task t_2 of the workflow from Figure 1 has the task snapshot $tss_1(t_2) = P'_1 \dot{\cup} N'_1$, where $P'_1 = \{\vec{a}_2\}$ and $N'_1 = \{\vec{a}_3\}$, i.e., $\vec{a}_2$ is asserted to true and $\vec{a}_3$ is asserted to false. It has also the task snapshot $tss_2(t_2) = P'_2 \dot{\cup} N'_2$, where $P'_2 = \{\vec{a}_3\}$ and $N'_2 = \{\vec{a}_2\}$, i.e., $\vec{a}_3$ is asserted to true and $\vec{a}_2$ is asserted to false.*

Definition 23 *Workflow snapshot. Let WG be a workflow. Suppose that $ND = \{i_1, i_2, \dots, i_k\}$, i.e., $t_{i_1}, t_{i_2}, \dots, t_{i_k}$ are the non-deterministic tasks of WG. For every $l \in \{1, \dots, k\}$ let $tss(t_{i_l})$ be a snapshot of t_{i_l}. A snapshot of WG, denoted by $wss(WG)$, is an element of the form $(tss(t_{i_1}), tss(t_{i_2}), \dots, tss(t_{i_k}))$.*

Table 2: Behavioral task models

EA model structure $t : f_E \rightsquigarrow f_C$	Behavioral task model $b(t)/b(t,s)$	Task Snapshot $s = tss(t)$
$t : a'_{i_1} \bullet a'_{i_2} \bullet \cdots \bullet a'_{i_l} \rightsquigarrow a'_j$	$a'_{i_1} = a'_{i_2} = \cdots = a'_{i_l} = a'_j$	—
$t : a'_i \rightsquigarrow a'_{j_1} \bullet a'_{j_2} \bullet \cdots \bullet a'_{j_l}$	$a'_i = a'_{j_1} = a'_{j_2} = \cdots = a'_{j_l}$	—
$t : a'_{i_1} \oplus a'_{i_2} \oplus \cdots \oplus a'_{i_l} \rightsquigarrow a'_j$	$a'_j = a'_{i_1} \oplus a'_{i_2} \oplus \cdots \oplus a'_{i_l}$	—
$t : a'_i \rightsquigarrow a'_{j_1} \oplus a'_{j_2} \oplus \cdots \oplus a'_{j_l}$	$a'_{j_1} = a'_i \wedge \vec{a}_{j_1}$, $a'_{j_2} = a'_i \wedge \vec{a}_{j_2}$, $\vdots$ $a'_{j_l} = a'_i \wedge \vec{a}_{j_l}$	$s \circlearrowleft t^{ND}(t)$
$t : a'_{i_1} \otimes a'_{i_2} \otimes \cdots \otimes a'_{i_l} \rightsquigarrow a'_j$	$a'_j = a'_{i_1} \otimes a'_{i_2} \otimes \cdots \otimes a'_{i_l}$	—
$t : a'_i \rightsquigarrow a'_{j_1} \otimes a'_{j_2} \otimes \cdots \otimes a'_{j_l}$	$a'_{j_1} = a'_i \wedge \vec{a}_{j_1}$, $a'_{j_2} = a'_i \wedge \vec{a}_{j_2}$, $\vdots$ $a'_{j_l} = a'_i \wedge \vec{a}_{j_l}$	$s \circlearrowleft t^{ND}(t)$
$t : a'_i \rightsquigarrow a'_j$	$a'_i = a'_j$	—

Example 24 *The workflow from Figure 1 has several snapshots. As* $ND = \{2,6\}$, $w^{ND}(WG) = \{t^{ND}(t_2), t^{ND}(t_6)\}$, $t^{ND}(t_2) = \{\vec{a}_2, \vec{a}_3\}$, $t^{ND}(t_6) = \{\vec{a}_7, \vec{a}_8\}$*. Let* $tss(t_2) = P' \dot{\cup} N'$*, where* $P' = \{\vec{a}_2\}$ *and* $N' = \{\vec{a}_3\}$*, i.e.,* $\vec{a}_2$ *is asserted to true and* $\vec{a}_3$ *is asserted to false. Let* $tss(t_6) = P \dot{\cup} N$*, where* $P = \{\vec{a}_7\}$ *and* $N = \{\vec{a}_8\}$*, i.e.,* $\vec{a}_7$ *is asserted to true and* $\vec{a}_8$ *is asserted to false. Then one snapshot of* WG*, is* $(tss(t_2), tss(t_6)) = (P' \dot{\cup} N', P \dot{\cup} N)$*, i.e.,* $\vec{a}_2 = true, \vec{a}_3 = false, \vec{a}_7 = true, \vec{a}_8 = false$*.*

Remark 25 *If* t_i *is a XOR-split then it has* $|t^{ND}(t_i)|$ *snapshots, if* t_i *is an OR-split then it has* $|2^{t^{ND}(t_i)}| - 1$ *snapshots. If the workflow* WG *does not contain non-deterministic tasks,* $ND = \emptyset$*. Therefore, there are no workflow snapshots.*

Definition 26 *A behavioral task model of a task* t *is a behavioral expression denoted by* $b(t)$ *when* t *is a deterministic task; and if* t *is a non-deterministic task it is denoted by* $b(t,s)$*, where* s *is a task snapshot. The behavioral expressions* $b(t)$ *and* $b(t,s)$ *are expressed in Table 2 and depend on the type the of the EA models associated to them.*

Definition 27 *Let* WG *be a workflow. Let* $t : f_E \rightsquigarrow f_C$ *be an EA model. If* t *is a deterministic task, i.e.,* $t : f_E \rightsquigarrow f_C$ *is an AND-join, AND-split, XOR-join, OR-join or Sequence, we say that* $b(t)$ *is positive when:*

(a) If t *is an AND-join or AND-split, all its Boolean terms are true,*

(b) If t *is a XOR-join, OR-join or Sequence, both sides of its equalities are true.*

If t is a non-deterministic task, i.e., $t : f_E \rightsquigarrow f_C$ is a XOR-split or an OR-split, we say that $b(t,s)$ is positive when:

(a) If $t : f_E \rightsquigarrow f_C$ is a XOR-split, there is only one of its equalities with both sides true,

(b) If $t : f_E \rightsquigarrow f_C$ is an OR-split, there is at least one of its equalities with both sides true.

If t is any task of T, we say that t is negative if is not positive.

Definition 28 *Let WG be a workflow. The behavioral workflow model of WG (denoted by $B(WG,s)$) is a system of equalities formed by the behavioral task models of all tasks $t_i \in T$, i.e.,*

Case 1. If WG does not contain non-deterministic tasks, then the behavioral workflow model is $\bigwedge\limits_{i=1}^{n} b(t_i)$.

Case 2. If WG contain non-deterministic tasks, suppose that $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ are the non-deterministic tasks of WG. For any workflow snapshot $s = (s_{i_1}, s_{i_2}, \ldots, s_{i_k}) = wss(WG) = (tss(t_{i_1}), tss(t_{i_2}), \ldots, tss(t_{i_k}))$ the behavioral workflow model is $\bigwedge\limits_{i=1}^{n} b(t_i, s_i)$, where

$$b(t_i,s_i) = \begin{cases} b(t_i),\ i \in \{1,\ldots,n\} \backslash \{i_1,i_2,\ldots,i_k\}, \\ b(t_{i_l},s_{i_l}),\ l \in \{1,2,\ldots,k\}. \end{cases} \tag{1}$$

Remark 29 *If all the tasks $t_i \in T$ are deterministic and therefore there is no workflow snapshots, we can denote $B(WG,s)$ simply by $B(WG)$.*

Example 30 *The workflow from Figure 1 has the following behavioral workflow model $B(WG,s)$:*

$$\begin{gathered} a'_2 = a'_4, a'_3 = a'_5, a'_7 = a'_9, a'_8 = a'_{10}, \\ a'_{\sqcup} = a'_1 = a'_6, a'_{11} = a'_{12} = a'_{\sqcap}, \\ a'_7 = a'_6 \wedge \vec{a}_7, a'_8 = a'_6 \wedge \vec{a}_8, a'_{11} = a'_9 \oplus a'_{10}, \\ a'_2 = a'_1 \wedge \vec{a}_2, a'_3 = a'_1 \wedge \vec{a}_3, a'_{12} = a'_4 \otimes a'_5. \end{gathered}$$

Definition 31 *We say that WG logically terminates if $a'_{\sqcap}$ is $true$ whenever $a'_{\sqcup}$ is $true$ and we say that WG never logically terminates if $a'_{\sqcap}$ is $false$ whenever $a'_{\sqcup}$ is $true$.*

Definition 32 *Let WG be a workflow and $B(WG,s)$ be its behavioral workflow model. We say that $B(WG,s)$ is positive if $a'_{\sqcap}$ in $B(WG,s)$ is $true$, whenever $a'_{\sqcup}$ is asserted to $true$ in $B(WG,s)$. We say that $B(WG,s)$ is negative if $a'_{\sqcap}$ in $B(WG,s)$ is $false$, whenever $a'_{\sqcup}$ is asserted to $true$ in $B(WG,s)$.*

Theorem 33 *Let WG be a workflow and let $B(WG,s)$ be its behavioral workflow model. Then, WG logically terminates if and only if $B(WG,s)$ is positive.*

Proof. Case 1. Suppose that WG does not contain non-deterministic tasks, i.e., all the tasks present in WG are deterministic. Then, $B(WG,s) = B(WG) = \bigwedge_{i=1}^{n} b(t_i)$. Since WG is formed by all its EA models, and according to Definition 26, every EA model $t_i : f_{E_i} \rightsquigarrow f_{C_i}$ is described by its behavioral task model $b(t_i)$, consequently the behavior of the workflow is described by $B(WG)$. Hence, $a'_{\sqcap}$ is true when $a'_{\sqcup}$ is true in WG if and only if $a'_{\sqcap}$ is true when $a'_{\sqcup}$ is true in $B(WG)$, i.e., WG logically terminates if and only if $B(WG)$ is positive.

Case 2. Suppose that WG contains non-deterministic tasks. Suppose that $ND = \{i_1, i_2, \ldots, i_k\}$, i.e., $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ are the non-deterministic tasks of WG. Let $s = (s_{i_1}, s_{i_2}, \ldots, s_{i_k}) = (tss(t_{i_1}), tss(t_{i_2}), \ldots, tss(t_{i_k}))$ be a workflow snapshot of WG. Then $B(WG,s) = \bigwedge_{i=1}^{n} b(t_i, s_i)$, where $b(t_i, s_i)$ is defined by (1).

Bearing in mind that WG is formed by all its EA models, and according to Definition 26, every EA model $t_i : f_{E_i} \rightsquigarrow f_{C_i}$ is described by its behavioral task model $b(t_i, s_i)$, then the behavior of the workflow is described by $B(WG,s)$. Therefore, $a'_{\sqcap}$ is true when $a'_{\sqcup}$ is true in WG if and only if $a'_{\sqcap}$ is true when $a'_{\sqcup}$ is true in $B(WG,s)$, i.e., WG logically terminates if and only if $B(WG,s)$ is positive. ■

Theorem 34 *Let WG be a workflow and let $B(WG,s)$ be its behavioral workflow model. Then, WG never logically terminates if and only if $B(WG,s)$ is negative.*

Proof. Using similar arguments as those from the proof of the previous Theorem, we can state that $a'_{\sqcap}$ is false whenever $a'_{\sqcup}$ is true in WG if and only if $a'_{\sqcap}$ is false when $a'_{\sqcup}$ is true in $B(WG,s)$. Thus, WG never logically terminates if and only if $B(WG,s)$ is negative. ■

3 Conclusions

To guarantee that workflows successfully terminate, it is necessary to verify their properties at design time. In this paper we present a formal theory, based on graphs, to check the termination of workflows. In our approach we model workflows with tri-logic acyclic directed graphs and develop a snapshot-based formalism to investigate the termination of workflows. The analysis of graphs-based workflows is important since many of the most well-known and widespread workflow systems use a notation based on graphs. While it is possible to transform a graph-based workflow into a Petri net-based workflow and then verify its termination, we believe that it is more practical for workflow vendors to directly implement into their systems the theory that we have developed. This solution will allow commercial applications to be less complex and eliminates the need to implement a software layer to interpret Petri nets. 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.

References

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

[2] P. Attie, et al. Specifying and enforcing intertask dependencies. In *Proc. 19th Int. Conference on Very Large Data Bases*, pp. 134–145, Ireland, 1993.

[3] J. Cardoso and G. Cravo Verifying the logical termination of workflows. In *Proc. 5th Annual Hawaii International Conference on Statistics, Mathematics and Related Fields*, pp. 330–346, ISSN: 1550-3747, 16-18 January, Hawaii, USA, 2006.

[4] Umeshwar Dayal, et al. Organizing long-running activities with triggers and transactions. In *ACM SIGMOD international conference on Management of data table of contents*, pp. 204–214, 1990.

[5] J. Eder, et al. A workflow system based on active databases. In *Proceedings of CON '94, Workflow Management: Challenges, Paradigms and Products*, pp. 249–265, Austria, 1994.

[6] A.H.M. ter Hofstede and E.R. Nieuwland. Task structure semantics through process algebra. *Software Engineering Journal*, 8(1):14–20, 1993.

[7] J. Klingemann, et al. Deriving service models in cross-organizational workflows. In *Proceedings of RIDE - Information Technology for Virtual Enterprises (RIDE-VE '99)*, pp. 100–107, Sydney, Australia, 1999.

[8] METEOR. Meteor (managing end-to-end operations) project home page, 2004.

[9] P. Muth, et al. Enterprise-wide workflow management based on state and activity charts. In *Proceedings NATO Advanced Study Institute on Workflow Management Systems and Interoperability*. Springer Verlag, 1998.

[10] M.P. Singh. Semantical considerations on workflows: An algebra for intertask dependencies. In *Fifth International Workshop on Database Programming Languages*, Electronic Workshops in Computing, Italy, 1995.