



Applications of propositional logic to workflow analysis[☆]

Glória Cravo

Departamento de Matemática e Engenharias, Universidade da Madeira, 9000-390 Funchal, Madeira, Portugal

ARTICLE INFO

Article history:

Received 3 April 2009

Received in revised form 14 October 2009

Accepted 19 October 2009

Keywords:

Graphs

Classical propositional logic

Workflows

Process modeling

Business processes

ABSTRACT

In this paper our main goal is to describe the structure of 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. In our approach we describe a workflow as a graph whose vertices represent workflow tasks and the arcs represent workflow transitions. Moreover, every arc (t_k, t_l) (i.e., a transition) has attributed a Boolean value to specify the execution/non-execution of tasks t_k, t_l . With this attribution we are able to identify the natural flow in the workflow.

Finally, we establish a necessary and sufficient condition for the termination of workflows. In other words, we identify conditions under which a business process will be complete.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

In this paper, we use graph theory and propositional logic to describe and analyze workflows. In particular, the use of propositional logic is a fundamental instrument to determine if a workflow has been correctly designed by an end user from the termination point of view. A workflow is an abstraction of a business process that consists of one or more tasks that need to be executed to complete a process (for example, hiring process, sales order processing, article reviewing, member registration, etc.), that can include human activity and/or software applications to carry out activities. A workflow can be represented by a graph, whose tasks are represented with vertices and the tasks are modeled with arcs, known as transitions. Each task represents a unit of work to be executed either by humans or application programs. A workflow describes all of the tasks needed to achieve each step in a business process.

Workflows may involve many distinct, heterogeneous, autonomous, and distributed tasks that are interrelated in complex ways. The complexity of large workflows requires a precise modeling to ensure that they perform according to initial specifications.

A vast number of formal frameworks have been proposed to allow workflow modeling verification and analysis, such as State and Activity Charts [1], Graphs [2], Event-Condition-Action rules [3,4], Petri Nets [5–8], Temporal Logic [9] and Markov chains [10]. Other approaches can be found in [11–13].

In this paper our formalism is based on graph theory and propositional logic. One relevant aspect of our approach is the use of propositional logic. In particular, the attribution of Boolean values to each arc of the workflow is very important, since it allows us to identify the natural flow in the workflow.

Finally, we identify conditions under which a workflow logically terminates. In other words, we are able to verify if a business process will be complete.

2. Workflow analysis

In this section we analyze the structure of workflows. We start by presenting the formal concept of a workflow.

[☆] This research was done within the activities of the *Centro de Estruturas Lineares e Combinatórias*.
E-mail address: gcravo@uma.pt.

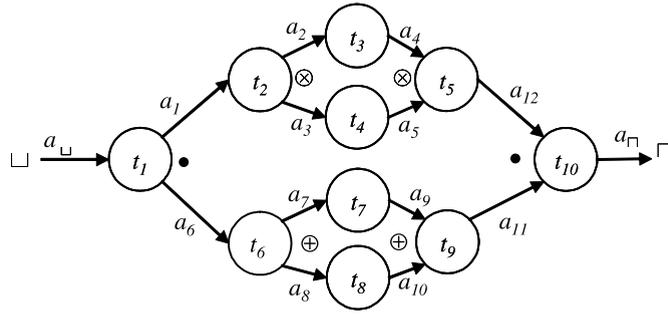


Fig. 1. Example of a workflow.

Definition 1. A workflow is a tri-logic acyclic directed graph $WG = (T, A)$, where $T = \{t_1, t_2, \dots, t_n\}$ 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 $A = \{a_{\square}, a_{\square}, a_1, a_2, \dots, a_m\}$ is a finite nonempty set of arcs representing workflow transitions. Each transition $a_i, i \in \{1, \dots, m\}$, is a tuple (t_k, t_l) where $t_k, t_l \in T$. The transition a_{\square} is a tuple of the form (\square, t_1) and transition a_{\square} is a tuple of the form (t_n, \square) . The symbols \square and \square 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' .

Example 2. In Fig. 1 is shown a workflow $WG = (T, A)$, where $T = \{t_1, t_2, \dots, t_{10}\}, A = \{a_{\square}, a_{\square}, a_1, a_2, \dots, a_{12}\}$ and $A' = \{a'_{\square}, a'_{\square}, a'_1, a'_2, \dots, a'_{12}\}$. 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).

Definition 3. For any task $t_i \in T$, the incoming transitions are the tuples of the form $a_j = (x, t_i), x \in T, a_j \in A$, and the outgoing transitions are the tuples of the form $a_l = (t_i, y), y \in T, a_l \in A$.

Example 4. In Fig. 1, 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 $a_3 = (t_2, t_4)$.

Definition 5. Given any task $t_i \in T$, the incoming condition is the Boolean expression $a'_{k_1} \varphi \dots \varphi a'_{k_l}, \varphi \in \{\bullet, \otimes, \oplus\}$, where the terms $a'_{k_1}, \dots, a'_{k_l} \in A'$, and a_{k_1}, \dots, a_{k_l} are the incoming transitions of task t_i . The terms $a'_{k_1}, \dots, a'_{k_l}$ are connected with the logical operator $\succ t_i$. If the task has only one incoming transition then the condition does not have logical operator.

The outgoing condition for task t_i is the Boolean expression $a'_{k_1} \varphi \dots \varphi a'_{k_l}, \varphi \in \{\bullet, \otimes, \oplus\}$, where the terms $a'_{k_1}, \dots, a'_{k_l} \in A'$, and a_{k_1}, \dots, a_{k_l} are the outgoing transitions of task t_i . The terms $a'_{k_1}, \dots, a'_{k_l}$ are connected with the logical operator $t_i \prec$. If the task has only one outgoing transition then the condition does not have logical operator.

Example 6. Consider task t_2 in Fig. 1. Its incoming condition is a'_1 and its outgoing condition is $a'_2 \otimes a'_3$.

A workflow is a set of tasks and transitions. The tasks can be considered as atomic pieces of the workflow, since they generate all transitions. Clearly, knowing the tasks and transitions of the workflow, allows to know the precise structure of the workflow. However, we need to exploit under which conditions an arbitrary task is executed and the consequences of its execution, i.e., we need to determine the natural flow of the workflow. Notice that when a workflow is correctly designed, it terminates by enabling the ending transition a_{\square} . Our main goal, is to identify conditions under which the ending transition a_{\square} is enabled, i.e., the workflow is correctly designed.

In order to analyze the consequences of the execution of a certain task, we introduce the concept of Event-Action model.

Definition 7. 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. 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. The condition f_E is called the event condition and f_C is called the action condition.

Every EA model $t_i : f_E \rightsquigarrow f_C$ has attributed a Boolean value, according to the following rules:

- (i) If both f_E, f_C are true, then its Boolean value is true;
- (ii) If both f_E, f_C are false, then its Boolean value is false.

A workflow starts its execution when transition a_{\square} is enabled. A transition is enabled/disabled if the respective Boolean term is asserted to be true/false. Thus, the workflow starts its execution by asserting a'_{\square} to be true.

For any EA model, the incoming condition propagates its Boolean value to the respective outgoing condition, i.e., the Boolean value of the outgoing condition is not arbitrary, it always depends on the Boolean value of the incoming condition, according to Definition 7. In other words, an EA model has a behavior with two distinct modes: when f_E is evaluated to true

متن کامل مقاله

دریافت فوری ←

ISIArticles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات