Verifying the Termination of Workflows

Glória Cravo* and **Jorge Cardoso**

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

Received 25 June 2006, accepted 11 July 2006

Key words Graphs, Classical Propositional Logic, Workflows, Process Modeling, Business Processes.
Subject classification 03B05, 05C20, 68R10

In this paper we describe the behavior of workflows using graph theory and logic. A workflow is an abstraction of a business process that consists of one or more tasks to be executed to reach the goal or objective of the business process. Graphs are a formal notation that may be used for representing business processes. We use propositional logic to describe all possible models or cases present in a workflow. We conclude the paper by studying the termination of workflows, a very important property that allows us to verify under which conditions a business process finishes its execution.

1 Introduction

In this paper we use mathematical tools, such as graph theory [10] and concepts of logic [16], to approach and solve a problem from computer science.

Currently, systems and infrastructures are being developed to support Web services, that can be integrated as part of workflow processes. A workflow is the formal definition of a process used to manage business processes [3], that consists in one or more tasks to be executed. The tasks are represented with vertices and the partial ordering of tasks is modeled with arcs, known as transitions.

Workflows have been successfully deployed to various domains, such as bio-informatics [7, 11], healthcare [4], the telecommunications industry [14], the military [12], and school administration [6].

In the last decade, important advancements have been accomplished in the implementation of workflow systems and in the development of theoretical foundations to allow workflow modeling, verification, and analysis. A number of formal frameworks have been proposed for workflow modeling such as State and Activity Charts [15], Graphs, Event-Condition-Action rules [8, 9], Petri Nets [1, 2], Temporal Logic [5] and Markov chains [13]. The use of directed graphs to model the control flow of workflows has been the main formalism used in workflow systems implementation (e.g. METEOR-S, TIBCO Workflow, and Staffware Process Suite).

In this paper we present a formal framework, based on control flow graphs theory. We also establish three important rules that allow us to describe all models (i.e., simple parts of the workflow) present in the workflow. Finally, we study a very important property of workflows, their logical termination. Our main result describes a necessary and sufficient condition for the logical termination of a workflow.

2 Logical Termination

In our approach we model workflows with tri-logic acyclic directed graphs. This type of graphs has an input/output logic operator associated with each vertex of the graph. We start by giving a formal definition of a workflow structure. The semantics of these vertices are well-known and have been widely used.

Definition 2.1 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 $A = \{a_{\sqcup}, a_{\sqcap}, a_1, a_2, \ldots, a_m\}$

* Corresponding author: e-mail: gcravo@uma.pt, Phone: +351 291 705 150, Fax: +351 291 705 199

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'.

A workflow starts its execution when transition $a_{\sqcup}$ is enabled. The transition can be enabled explicitly by a user or implicitly by an external event.

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

Definition 2.2 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 2.3 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 $\succ t_i$. 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 incoming/outgoing transition then the condition does not have a logical operator.

Definition 2.4 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.

Remark 2.5 The expressions f_E and f_C are logically equivalent. However, we use the symbol $\rightsquigarrow$ to represent this equivalence, which is suggestive to indicate the flow in the workflow.

Remark 2.6 For any EA model $t_i : f_E \rightsquigarrow f_C$, its behavior has two possible distinct modes: when f_E is evaluated to true and when f_E is evaluated to false. In the first case, its outgoing transitions are enabled or disabled in such way that the outgoing condition is true. In the second situation, task t_i disables all its outgoing transitions and consequently f_C becomes false.

The symbol $\leftrightarrow$ is used in the following way: $S_1 \leftrightarrow S_2$ means that the compound statements S_1 and S_2 are logically equivalent, using substitution rules from the Laws of Logic.

Note that a workflow has as many EA models as tasks. When EA models are combined, new models can be derived, based on their Boolean expressions. The creation of new models can be accomplished with the logical implication of EA models. The following rules allow us to create new models based on existing ones.

Theorem 2.7 *Transitivity Rule: Let $WG = (T, A)$ be a workflow. Suppose that the EA models $f_{E_i} \rightsquigarrow f_{C_i}$ and $f_{E_j} \rightsquigarrow f_{C_j}$ hold. If $f_{C_i} \leftrightarrow f_{E_j}$, then the model $f_{E_i} \rightsquigarrow f_{C_j}$ also holds.*

Corollary 2.8 *Logical Implication of EA models: Let $WG = (T, A)$ be a workflow. Suppose that the EA models $f_A \rightsquigarrow f_B$ and $f_B \rightsquigarrow f_C$ hold. Then the model $f_A \rightsquigarrow f_C$ also holds.*

Theorem 2.9 *Right Partial Transitivity Rule: Let $WG = (T, A)$ be a workflow.*

1. *If both EA models $f_{E_i} \rightsquigarrow f_{C_i}$ and $f_{E_j} \rightsquigarrow f_{C_j}$ hold, where $f_{C_i} \leftrightarrow f_n \varphi f_{E_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the model $f_{E_i} \rightsquigarrow f_n \varphi f_{C_j}$ also holds.*
2. *If both EA models $f_{E_i} \rightsquigarrow f_{C_i}$ and $f_{C_j} \rightsquigarrow f_{E_j}$ hold, where $f_{C_i} \leftrightarrow f_n \varphi f_{E_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the model $f_{E_i} \rightsquigarrow f_n \varphi f_{C_j}$ also holds.*

Theorem 2.10 *Left Partial Transitivity Rule: Let $WG = (T, A)$ be a workflow.*

1. *If both EA models $f_{E_i} \rightsquigarrow f_{C_i}$ and $f_{E_j} \rightsquigarrow f_{C_j}$ hold, where $f_{E_i} \leftrightarrow f_n \varphi f_{E_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the model $f_n \varphi f_{C_j} \rightsquigarrow f_{C_i}$ also holds.*
2. *If both EA models $f_{E_i} \rightsquigarrow f_{C_i}$ and $f_{C_j} \rightsquigarrow f_{E_j}$ hold, where $f_{E_i} \leftrightarrow f_n \varphi f_{E_j}$, $\varphi \in \{\bullet, \otimes, \oplus\}$, then the model $f_n \varphi f_{C_j} \rightsquigarrow f_{C_i}$ also holds.*

Definition 2.11 Let $WG = (T, A)$ be a workflow. An extended EA model is a model obtained from the EA models in WG, applying the previous rules (transitivity, right partial transitivity and left partial transitivity).

Remark 2.12 The extended EA models are represented with the same symbology of the EA models. Moreover, they have the same behavior of the EA models, i.e., if $X \rightsquigarrow Y$ is an extended EA model, then X and Y have always the same Boolean value.

From now on, we will denote by M the set of all EA models defined over WG.

Definition 2.13 Let $WG = (T, A)$ be a workflow. Let $X \rightsquigarrow Y$ be an extended EA model. In this situation we say that M logically implies $X \rightsquigarrow Y$, or $X \rightsquigarrow Y$ is logically reached by M.

Definition 2.14 Let $WG = (T, A)$ be a workflow. The set of all EA models and extended EA models that hold over WG is called the closure of M and it is denoted by M^+, i.e.,

$$M^+ = M \cup \{X \rightsquigarrow Y : \ M \text{ logically implies } X \rightsquigarrow Y\}.$$

These three rules (transitivity, right partial transitivity and left partial transitivity) are sound because any extended EA model still holds over WG. On the other hand, they are complete because they generate all models in WG, i.e., if $X \rightsquigarrow Y$ is in the closure of M, M^+, then it can be deduced using these three rules.

Definition 2.15 Let $WG = (T, A)$ be a workflow. We say that WG logically terminates if transition $a_{\sqcap}$ is enabled at some point in time after transition $a_{\sqcup}$ has been enabled.

Once $a_{\sqcup}$ is enabled, tasks of the workflow start their execution. The processing of the workflow stops when one of the following cases occur:
(a) the workflow finishes by enabling transition $a_{\sqcap}$,
(b) the processing stops at some task because the incoming condition is false.

Our main result is the following theorem, where we establish a necessary and sufficient condition for the logical termination of workflows.

Theorem 2.16 *Let $WG = (T, A)$ be a workflow. Then WG logically terminates if and only if $a_{\sqcup} \rightsquigarrow a_{\sqcap} \in M^+$.*

This theorem allows us to check the termination of workflows. While the definition of logical termination obliges to study all EA models present in WG, and the connection between each others, with our approach the most important advantage is to focus our attention in only one model: $a_{\sqcup} \rightsquigarrow a_{\sqcap}$. Clearly, with this approach we establish a more practical and easy process to verify this important property of workflows: their termination.

3 Conclusions

Workflow management systems are capable of hosting e-commerce applications by integrating business functionalities in a short time and with low costs. This is of significant importance for global and competitive markets. Workflows describing e-commerce applications require a precise modeling, verification and analysis to ensure that they perform according to initial specifications. To guarantee that workflows are successfully executed at runtime, it is necessary to verify their properties at design time. In this paper we present a formal framework, based on control flow graphs theory, to check workflow specifications for correctness, i.e., the logical termination of the workflow.

The contribution of our work will enable the development of tools that will support and allow business process analysts to verify the correctness of their workflows at design time.

Acknowledgements This work has been carried out in the context of project POSC/EIA/56164/2004 which has been financed by FCT and FEDER - Portugal.

References

[1] W. M. P. v. d. Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
[2] W. M. P. v. d. Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In W. v. d. Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806, pages 161–183. Springer-Verlag, Berlin, 2000.
[3] W. M. P. v. d. Aalst and K. van Hee. Workflow Management-Models, Methods, and Systems. The MIT Press, 2002

[4] 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, 2003. 35(2): p. 83-98.

[5] P. Attie, M. Singh, A. Sheth, M. Rusinkiewicz, Specifying and enforcing intertask dependencies. In *Proc. 19th Int. Conference on Very Large Data Bases*, pp. 134–145, Ireland, 1993.

[6] CAPA, Course Approval Process Automation (CAPA). 1997, LSDIS Lab, Department of Computer Science, University of Georgia: Athens, GA.

[7] J. Cardoso, R.P. Bostrom, and A. Sheth, "Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications". *Information Technology and Management Journal*. Special issue on Workflow and E-Business, Kluwer Academic Publishers, Vol. 5, Nos. 3/4, July/October, 2004, pp. 319-338.

[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. P. Grimaldi. Discrete and Combinatorial Mathematics an Applied Introduction. Addison-Wesley Publ. Company, second edition, 1989.

[11] R. Hall, J. Miller, J. Arnold, K. Kochut, A. Sheth, M. Weise, Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequence Initiative, in Genomics of Plants and Fungi, R.A. Prade and H.J. Bohnert, Editors. 2003, Marcel Dekker, Inc.: New York, NY. p. 359-371.

[12] M. Kang, J. Froscher, A. Sheth, K. Kochut, J. Miller, A Multilevel Secure Workflow Management System. in Proceedings of the 11th Conference on Advanced Information Systems Engineering. 1999. Heidelberg, Germany: Springer-Verlag.

[13] J. Klingemann, J. Wsch, K. Aberer, 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.

[14] Z. Luo, Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes, in Department of Computer Science. 2000, University of Georgia: Athens, GA. p. 171.

[15] P. Muth, D. Wodtke, J. Weissenfels, G. Weikum, and A. Kotz 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.

[16] D. J. Pym and E. Ritter. Reductive Logic and Proof-Search. Clarendon Press, 2004.