Understanding Behavior of Business Process Models*

Pablo A. Straub and Carlos Hurtado L.

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Vicuña Mackenna 4860 (143), Santiago 22, Chile
straub@ing.puc.cl churtado@ing.puc.cl

1 Introduction

Business processes have intrinsic parallelism given by the relative independence of some activities. Parallelizing a process might "dramatically reduce cycle times and the resultant costs" [5], but it might lead to anomalous behavior, like deadlock or useless activities. There are three main approaches to avoid behaviorally incorrect models, a notion that is not even well defined: 1) Build a model and then verify its properties (e.g., building the space state, or finding net invariants [2]). 2) Build a model that is correct by construction (e.g., using activity annotations [1] akin to parbegin parend pairs). 3) Use only small models by abstracting models and submodels. The first approach does not explain why or where a particular model has behavioral problems. The second approach constrains the forms of parallelism: it is impossible to model a PERT chart. The third approach is just a rule of thumb that may lead to oversimplified models.

This work intends to: define a notion of correct behavior in business process models and give a framework to allow behavioral analysis.

2 A Notion of Correct Behavior

We use our Copa notation [6]; it is easy to map concepts presented here into similar languages like ICN [3], Action Workflow [4], VPL/Rasp [8], whose behavioral semantics are defined in terms of Petri nets. Fig. 1a shows a simple but anomalous process model. Connector x splits execution in two parallel activities (a and b). Activity a proceeds to c while b may choose to proceed to either c (causing a deadlock in y) or to connector y (synchronizing with c).

The semantics of Copa is given by a Petri net obtained by translating each element of the model and connecting corresponding elements (Fig. 1b). Allowable initial states have a single token in one input socket and no other tokens.

One basic property of a good model is that it does not deadlock, that is, each process of the model reaches an output socket. Another basic property of a good process model is single-response, that is, each enaction of the process produces exactly one output. Process models can suffer from prescribing too

* This work is funded in part by CONICYT through project FONDECYT 1940677.

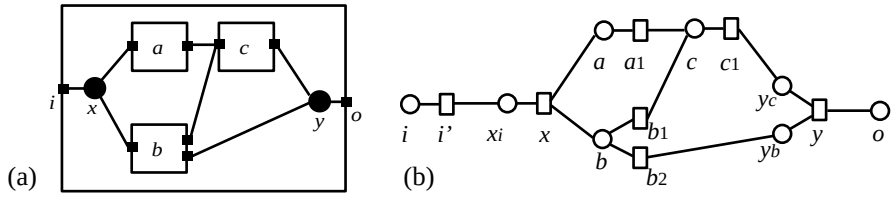

Fig. 1. Example of a Copa net with an anomalous process model.

much work, so that some activities are unnecessarily performed, because in some particular execution no output socket depends on them. For example, if there were a connection from activity b to output socket o in Fig. 1, choosing this connection would render a and c useless.

Given a process model $\mathcal{M}$ whose set of output sockets is S_o:

Property 1 (Deadlock freedom) A final state M_f is a *deadlock* if for all $s_o \in S_o$, $M_f(s_o) = 0$. $\mathcal{M}$ is *deadlock-free* if none of its final states is a deadlock.

Property 2 (Single Response) $\mathcal{M}$ is *single-response* if all final states M_f satisfy $\sum_{s_o \in S_o} M_f(s_o) = 1$. It is *multiple-response* if there is a final state M_f such that $\sum_{s_o \in S_o} M_f(s_o) > 1$.

Property 3 (Usefulness) An activity a is *useless* within a process (in the sense of Petri nets) of $\mathcal{M}$ if there is no path from a to an output socket. A process model $\mathcal{M}$ is *useful* if no activity is useless in any process.

Property 4 (Simple control) $\mathcal{M}$ has *simple control* if every final state M_f satisfies $\sum_{s_o \in S_o} M_f(s_o) = 1 = \sum_{p \in P} M_f(p)$.

Simple control implies that if a model begins with a single token in one of its input sockets, it ends with a token in one of its output sockets and there are no other tokens. Simple control implies the other properties [6].

3 A framework to understand control properties

In a sense, simple control means "at the end there is exactly one token". Place invariants capture the idea of having the right number of tokens in a system. A place invariant I is an assignment of weights to places such that for any transition t, the sum of the weights of its predecessors equals the sum of the weights of its successors, thus the weighted sum of their tokens is constant [2].

Theorem 1. *Let I be a positive invariant of a process model with an input socket s_i that has a path to an output socket s_o, then if $I(s_i) \neq I(s_o)$, the initial state that has a token in s_i leads to multiple response, an overloaded state, or deadlock.*

A necessary but not sufficient condition for simple control is the existence of a positive balanced invariant (i.e., all sockets have the same weight). The example has a positive balanced invariant but it may deadlock.

A theory of threads of control whose results are proven in [7] is an extension of the work on invariants. A thread of control captures the idea of parallel subprocesses within a process. Unlike the usual sense given in operating systems, our concept of threads is static. Within a process model, every activity should belong to one and only one thread; to tell which one, we propose a labeling that assigns a set of thread labels to each place or transition.

Definition 2 Model labeling. The labeling of a model is a function τ from nodes of the net to sets of labels, defined by: 1) The only label for an output socket is $\mathbf{1}$. 2) Each label of a transition is computed by adding a consistent set of labels[2], one from each successor. 3) The label of a place p that does not correspond to an output socket is computed by multiplying a label from a successor $t \in p^\bullet$ that leads to an output socket, by $\sigma(t, \{p\})$.

For example, part of the labeling of the model in Fig. 1b is: $\tau(o) = \{\mathbf{1}\}$, $\tau(c) = \{\mathbf{1} \otimes \sigma(y, \{y_c\}) \otimes \sigma(c_1, \{c\})\}$, and $\tau(b) = \{\mathbf{1} \otimes \sigma(y, \{y_b\}) \otimes \sigma(b_2, \{b\}),\ \mathbf{1} \otimes \sigma(y, \{y_c\}) \otimes \sigma(c_1, \{c\}) \otimes \sigma(b_1, \{b\})\}$.

Definition 3 Label equivalence. Label equivalence, denoted $\doteq$, is defined by: 1) $\oplus$ is commutative, 2) $\oplus$ and $\otimes$ are associative, 3) $\otimes$ distributes by the right over $\oplus$, 4) $\sigma(t, x) \oplus \sigma(t, y) \doteq \sigma(t, x \cup y)$ if $x \cap y = \varnothing$, and 5) $x \otimes \sigma(t, {}^\bullet t) \doteq x$.

A *thread* is a non-empty equivalence class of labels. A labeling τ such that all labels assigned to a given node x are equivalent defines a threading ψ. We usually denote a thread by a label (one of its members). Threadings are non-numeric invariants [7].

From definition 3 we see that the example has no threading because activity b belongs to two threads of control, namely $\mathbf{1} \otimes \sigma(y, \{y_b\})$ and $\mathbf{1} \otimes \sigma(y, \{y_c\})$. This fact implies the model has control anomalies.

Definition 4. A threading ψ is *balanced* if the thread of all input sockets is $\mathbf{1}$. A model is balanced if its threading is.

If a model is balanced, in all reachable states M the summation of the active threads (those that have a token) is $\mathbf{1}$; besides, only one activity is executing in a thread, that is, M marks at most one place in the same thread. The main result from this theory is that threadings capture our notion of correctness:

Theorem 5. *A model is balanced if and only if it has simple control.*

There are three possible causes for not having a balanced threading, which can be interpreted in terms of behavioral properties. First, there might be a place with no labels. If a place in the model has no label, this implies there is a proper trap in the net (i.e., a set of places that once they receive a token they always have some tokens [2]) and the model has an overloaded state, unless a deadlock upstream impedes reaching the trap. Second, there might be two

[2] A set of labels is *consistent* if all occurrences of a place refer to the same transition.

unequivalent labels for a place, which means that this place is an activity whose output sockets are connected to different threads: this implies either deadlock or overloaded state. Third if there a threading, but it is unbalanced, then Theorem 1 applies [7].

4 Conclusion

There are two reasons to use simple control as a notion of behavioral correctness. First, it implies there are no control anomalies, like deadlock, useless activities, and multiple response [6]. Second, a simple control model behaves like an activity; this allows consistent or-abstraction in which a process model can be used safely within a larger model. Or-behavior is the most commonly accepted from of model abstraction (as in, e.g., ICN, VPL/Rasp, Action Workflow).

Simple control is related to other behavioral properties of free-choice Petri nets. In fact we prove in [6] that a model has simple control if and only if a connected free choice net derived from the model is live and safe in the sense of [2]. This implies in turn that simple control in free choice nets (i.e., in all Copa models) can be decided in polynomial time.

Our framework formalizes threads of control in business process models and relates them to behavioral correctness, by demanding that threads of control be adequately combined. We have recognized several applications of thread theory within workflow models: diagnostic of why a model might fail, model building, advanced exception handling, unanticipated exceptions.

References

1. Giorgio De Michelis and M. Antonietta Grasso. How to put cooperative work in context: Analysis and design requierements. *Issues of Supporting Organizational Context in CSCW Systems.* L. Banon and K. Schmidt, August 31, 1993.
2. Jörg Desel and Javier Esparza. *Free-choice Petri Nets.* Tracts in Theoretical Computer Science, Cambridge University Press, 1994.
3. Clarence A. Ellis and Gary J. Nutt. Modeling and enactment of workflow systems. In *14th Int'l Conf. on Application and Theory of Petri Nets*, June 1993.
4. Raúl Medina-Mora, Terry Winograd, Rodrigo Flores, Fernando Flores. The Action Workflow approach to workflow. *Management Technology Proceedings of CSCW*, November 1992.
5. Michael Parry. *Reengineering the Business Process.* The Workflow paradigm, Future Strategies Inc., ISBN 0-9640233-x.
6. Pablo Straub and Carlos Hurtado L. The simple control property of business process models. In *XV Int'l Conf. of the Chilean Computer Science Society*, Arica, Chile, October 30th to November 3rd, 1995.
7. Pablo Straub and Carlos Hurtado L. A theory of parallel threads in process models. Tech. Report RT-PUC-DCC-95-05, Computer Science Dept., Catholic Univ. of Chile, August, 1995. In `ftp://ing.puc.cl/dcc/techReports/rt95-05.ps`.
8. Keith D. Swenson. Visual support for reengineering work processes. In *Proc. of the Conf. on Organizational Computing Systems*, November 1993.

This article was processed using the LaTeX macro package with LLNCS style