# MODEL CHECKING FOR BEHAVIOR ANALYSIS ON BPM

CHEN ZHENYU
HTTP://WWW.MATHLOGIC.NET/CZY

ABSTRACT. Companies today are becoming ever more focused on their business processes. Business process management(BPM) systems are driven by explicit process models which are typically used to improve mission critical business processes of an organization. Hence, the correctness, effectiveness, and efficiency of business processes supported by the BPM systems are vital to the organization. In this paper, we present a model checking framework for behavior analysis on BPM and bring out four unsolved problems. Moreover, a consistent abstraction of state space is addressed to fight the state explosion problem.

**Keywords**: abstraction; behavior analysis; business process; model checking

## 1. INTRODUCTION

Business processes constitute an important approach for understanding the work conducted by companies. A business process describes the general activities done to produce a good or deliver a service. Derived from such processes are the workflows which are actually implemented by the company. There exist a number of information technology solutions to support the definition, execution and monitoring of workflows. Business Process Management (BPM) can be considered as an extension of classical Workflow Management (WFM) systems and approaches. Workflow and business process are thus very similar. We use the terms "business process" and "workflow" as synonyms in this paper, although there are minor differences.

BPM includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes. Two popular approaches to support formal semantics for business processes are petri nets [20] and $\pi$-calculus [19]. The detail definitions of model languages could not be described in this paper. BPM systems are typically used to improve mission critical business processes of an organization. Hence, the correctness, effectiveness, and efficiency of business processes supported by the BPM system are vital to the organization. In [2], three basic types of analysis methods are presented. We improve it in model checking framework and describe it as follows:

1. Logic Analysis: is concerned with the logical correctness of business process definitions, i.e., to guarantee that there is no potential deadlocks and never-ending loops in business processes.
2. Behavior Analysis: is mainly concerned with the gap between the specified business process and the intended one. Behavior ananlysis needs to be done by domain experts and analysis is context dependent.

This work is supported by NSFC and national research foundation for the doctorial program of high education of education ministry of China.

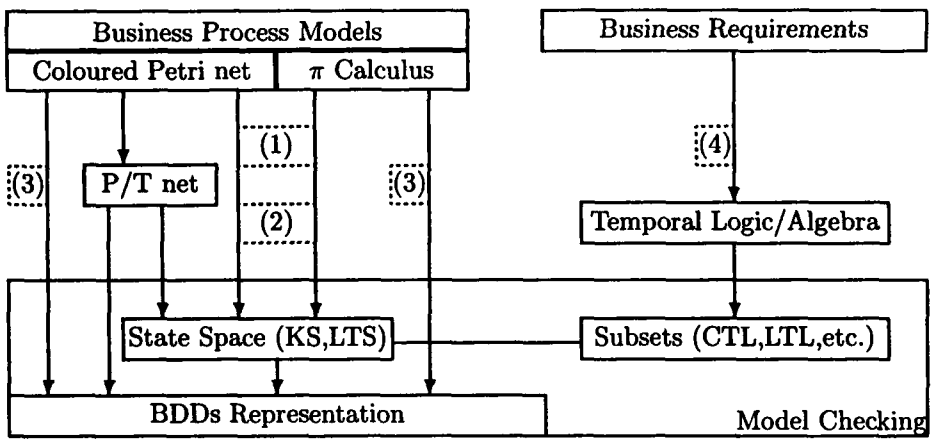


FIGURE 1. Model Checking Framework for Behavior Analysis on BPM

3. Performance Analysis: is evaluating the ability to meet requirements with respect to throughput times, service levels, and resource utilization.

Most exist works concern with logic analysis and performance analysis. The criterion of soundness [1] based on petri nets is addressed to check the logic correctness of business processes. This criterion combines properties are "option to complete" , "proper completion" and "no dead tasks". The corresponding criterion of soundness based on $\pi$-calculus is presented in [5]. The performance analysis [22] is addressed to evaluate the ability with respect to throughput times, service levels and resource utilization. However, how to implement behavior analysis on BPM is still a challenge. In the state space context, we do a broad survey and present a model checking framework for behavior analysis on BPM. Most exist works could support partial functions in this framework, but there are several unsolved problems in this framework [6]. The goal of this paper is to attack some of these problems.

The paper is organized as follows. In the next section, we introduce the model checking framework for behavior analysis on BPM. Section 3 presents an abstraction method for state space of coloured petri nets. The conclusion and future work are drawn in the last section.

## 2. Model Checking Framework for Behavior Analysis

Formal methods for analysis on BPM have been an active area of research, most existing works consider logic analysis and performance analysis. However, I only focus on behavior analysis based on model checking. Figure 1 illustrates the model checking framework for behavior analysis on BPM.

Generally speaking, model checking[8] is to decide whether a given structure $M$ is a model of a logical formula $f$ , i.e., whether $M$ satisfies $f$, abbreviated as $M \models f$. In this framework, $M$ is a business process model, typically modeled by a CP-net or $\pi$ calculus, which could be translated into a state space. And $f$, a business requirement, is typically drawn from temporal logic or algebraic term. The model-checker then provides a fully automatic approach for proving whether the business process model $M$ enjoys this business property $f$.

It is beneficial to distill a reference for the many material ontologies in this framework. Business process models and business requirements provide graphic user interfaces and property specifications to give a business syntax. When looking at a business process, three perspectives can be employed [11]: control-flow, data-flow and resource allocation. In order to model these perspectives, the formal semantics of business processes are commonly defined by Coloured Petri nets [12] (CP-nets) or $\pi$-calculus [19]. Business requirements could be given by temporal logics or some algebra-based terms.

In general, CP-nets could be unfolded to low level Petri nets, such as P/T (Place/Transition) nets with equivalent behaviors[13]. State spaces of bounded petri nets could be constructed by reachability graphs [20]. However, the unfolded form of a CP-net will generally be much bigger than the CP-net itself. In [13,14], the state space method avoiding this unfolding is presented. Similarly, the state space of $\pi$-calculus also could be constructed by the operational semantics [19]. On the other hand, BDD [3] as a compact and efficient data structure would be concerned with. Symbolic model checker constructs more than $10^{20}$ states by BDDs [4]. An approach for direct symbolic analysis of bounded Petri nets is addressed in [21].

In the past two decades, model checking has emerged as a promising and powerful approach to fully automatic verification of systems. In model checking, basic concepts, analysis methods and computer tools are ample and mature. There are many successful model checker in practice, such as SMV for CTL [17], SPIN for LTL [10], etc.

However, in the whole framework of model checking for behavior analysis on BPM, there at least four unsolved problems:

(1) The state spaces of CP-nets and $\pi$-calculus suffer from state explosion problem. It is not known how to construct a small state space by abstraction or reduction and which properties could be preserved in business process context.
(2) The decision as to which details should be omitted for the verification task is made always manually. It is not know how to generate abstraction functions automatically from the property specifications.
(3) How to build a BDDs for a CP-net and $\pi$-calculus with abstraction methods. Does avoiding unfolded P/T nets and state spaces make it work more efficiently?
(4) In many cases, temporal logics is obscure to someone without logic knowledge. How to describe proper business requirements is a challenge to users. Comprehensible standard specifications are still missing.

The goal of this paper is to attack one of these problems. The next section presents a consistent abstraction approach to address the problem (1). The other problems are discussed in the last section.

## 3. Abstraction-based State Space Methods

Abstraction is a method for reducing the state space of the checked system [9, 15, 16]. Intuitively, the reduction is achieved by hiding (abstracting away) some of the system details that might be irrelevant for the checked property. Verifying the simplified models is in general more efficient than verifying properties of the

original ones. Abstraction techniques can be classified as assertive abstraction or consistent abstraction techniques [7].

- assertive abstraction techniques systematically remove irrelevant behaviors from the system. An under-approximation technique establishes a relationship between the abstract model and the original one, so that falseness at the abstract level will imply falseness of the original system.
- consistent abstraction techniques systematically release constraints, and thus add more behaviors to the system. They establish a relationship between the abstract model and the original one such that correctness at the abstract level implies correctness of the original system.

In order to reason about the relations between abstractions and original models, we introduce the bisimulation relation and the simulation preorder over Kripke structures [18]. Given two Kripke structures $M = (S, R, L)$ and $M' = (S', R', L')$ with the same set of atomic propositions $AP$, a relation $B \subseteq S \times S'$ is a bisimulation relation if and only if for all s and s', if B(s,s') then the following conditions hold:

1. $L(s) = L'(s')$.
2. For every state $s_1$ such that $R(s, s_1)$, there is a state $s'_1$ satisfying $R'(s', s'_1)$ and $B(s_1, s'_1)$.
2. For every state $s'_1$ such that $R'(s', s'_1)$, there is a state $s_1$ satisfying $R(s, s_1)$ and $B(s_1, s'_1)$.

A relation $H \subseteq S \times S'$ is a simulation relation if and only if for all s and s', if H(s,s') then the following conditions hold:

1. $L(s) = L'(s')$.
2. For every state $s_1$ such that $R(s, s_1)$, there is a state $s'_1$ satisfying $R'(s', s'_1)$ and $H(s_1, s'_1)$.

We say that $M$ and $M'$ are *bisimilar*, denoted $M \approx M'$ if there exists a bisimulation relation B over $M$ and $M'$. We say that $M'$ simulates $M$, denoted $M \preceq M'$ if there exists a simulation relation H over $M$ and $M'$. Obviously, if $M'$ is an assertive abstraction of $M$, then $M' \preceq M$. In contrast, if $M'$ is a consistent abstraction of $M$, then $M \preceq M'$. Two useful theorems as follows:

**Theorem 1.** [8] *Suppose $M \approx M'$. Then for every CTL* formula $f$, $M \models f \Leftrightarrow M' \models f$.*

**Theorem 2.** [8] *Suppose $M \preceq M'$. Then for every ACTL* formula $f$, $M' \models f \Rightarrow M \models f$.*

State space methods are some of the most prominent analysis approaches for CP-nets [13, 14]. The basic idea underlying state sapces is to compute all reachable states and state changes of the system. The formal definition as follows:

**Definition 3.** *A state space of CP-net over a color domain D(maybe infinite) is a triple $M = (S, R, L)$. Where*

1. *$S = [M_0\rangle$, i.e., S is a set of all reachable markings.*
2. *$R = \{(M_1, M_2) | \exists b \in BE, M_1[b\rangle M_2\}$.*
3. *$L = D^n$, where n is the number of places.*

The number of states for CP-nets are usually very large, even infinite. In [13, 14], OE-graph is addressed to take in place of O-graphs (i.e., reachability graphs) by equivalence relations. In this paper, we use existential abstraction [9, 15, 16]

functions to define a consistent state space. An abstraction function is a mapping from a domain D and an abstraction domain A, i.e., $h(d) = a, d \in D, a \in A$. We extend the mapping $h$ to n-tuples in $D^n$ : $h((d_1, ..., d_n)) = (h(d_1), ..., h(d_n))$. We say that two states $s$ and $s'$ are equivalence if and only if $h(L(s)) = h(L(s'))$. $[M]$ is the set of all states equivalent to $M$.

**Definition 4.** *Given a state space of CP-net $M = (S, R, L)$ and an abstraction function $h$, a C-(consistent) state space is a triple $M^c = (S^c, R^c, L^c)$ satisfying the requirements below:*

1. $S^c = \{[M] | [M] \cap [M_0\rangle \neq \emptyset\}$.
2. $R^c = \{([M_1], [M_2]) | \exists M_1' \in [M_1], M_2' \in [M_2], b : M_1'[b\rangle M_2'\}$.
3. $L^c(s) = h(L(s))$.

**Theorem 5.** *Given a CP-net and an abstraction function $h$ , every C-state space $M^c$ simulates state space $M$, i.e., $M \preceq M^c$. Therefore, every ACTL* formula $f$, $M^c \models f \Rightarrow M \models f$.*

Actually, the above method is only weak preservation. In that case, if a property is true for the abstract model then it is also true for the original model. If a property is not true for the abstract model, then no conclusion can be reached with regards to the original model. The advantage of weak preservation is that it enables more significant reductions. However, it also increases the likelihood that we will be unable to determine the truth of a property in the system.

## 4. Conclusion and Future Work

In this paper, A model checking framework for behavior analysis on BPM is presented. These techniques include CP-nets unfolding to P/T nets [12], state spaces of petri nets constructed by reachability graphs [13, 14, 20], state spaces of $\pi$-calculus computed by operational semantics [19], temporal logic specifications [8], BDDs' representations [3], model checking for CTL and LTL [8]. Moreover, we state four unsolved problems challenging this framework.

In section 3, we make the first step. An abstraction method is addressed to construct an efficient consistent state space of a CP-net. In general, the number of states of a CP-net is $2^{|P|\times|D|}$, where $P$ is the set of places and $D$ is the color data domain. Therefore, the reduction methods refer to $P$ and $D$.

- **Structure Reduction**: is to reduce the irrelevant places of a CP-net, i.e., reduce $P$.
- **Data Abstraction**: is to abstract away the irrelevant color data of a CP-net, i.e., abstract $D$.

There exist two preservations: strong preservation for bisimulation and weak preservation for simulation preorder. The method in section 3 is one of data abstraction approaches, and it's a weak preservation. However, there are many possibilities for further work in this framework.

1. The formal definitions of structure reduction and data abstraction will be considered.
2. How to generate a abstraction function automatically for a given business requirement (problem 2).
3. How to construct BDDs representations for CP-nets and $\pi$ calculus with abstraction methods directly (problem 3).

4. How to define a comprehensible specification for a business requirement (problem 4).

## 5. Acknowledgement

Thanks to Prof. DING Decheng, Prof. SU Kaile and TAO Zhihong for their valuable comments on model checking. Thanks to all members of business integration team and business value team in IBM China research lab for their valuable discussions and suggestions on my research framework.

## References

[1] W.M.P. van der Aalst. Verification of Workflow Nets. *Application and Theory of Petri Nets in LNCS, 407-426.* Springer-Verlag, Berlin, 1997.

[2] 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.

[3] R.E. Bryant. Graph-Based algorithms for boolean function manipulation. *IEEE Transactions on Computers C35(8):677-691,* 1986. J.C.M.

[4] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking $10^{20}$ States and Beyond. *In Proc. of the Fifth Annual Symposium on Logic in Computer Science,* 1990.

[5] Z.Y. Chen. PI Calculus based Business Process, Technical Report, IBM China Research Lab, Apr. 2004.

[6] Z.Y. Chen. Progress Report on First Year of My Ph.D. Study, Technical Report, Department of Mathematics, Nanjing University, Aug. 2004.

[7] Z.Y. Chen and D.C. Ding. Assertive-Consistent Model Checking, *submitted to JCST,* Aug. 2004.

[8] E.M.Clarke, O.Grumberg, and D.A.Peled. Model checking. Cambridge, MA: The MIT Press, 1999.

[9] O. Grumberg. Abstractions and Reductions in Model Checking. *in Nato Science Series,* Vol. 62, 2001.

[10] G. Holzmann. The spin model checker. *IEEE Transaction on Software Engineering 23:279-295,* 1997.

[11] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, London, UK, 1996.

[12] K. Jensen. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. Monographs in Theoretical Computer Science. Springer-Verlag, 1992.

[13] K. Jensen. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods. Monographs in Theoretical Computer Science. Springer-Verlag, 1994.

[14] L. M. Kristensen. State Space Methods for Coloured Petri Nets. PhD thesis, Department of Com- puter Science, University of Aarhus, Denmark, 2000.

[15] D. E. Long. Model Checking, Abstraction and Compositional Reasoning. PhD thesis. Carnegie Mellon University, 1993.

[16] Y. Lu. Automatic Abstraction in Model Checking. PhD thesis. Carnegie Mellon University, 2000.

[17] K.L. McMillan. Symbolic Model Checking. Kluwer Academic Publishers, Boston, 1993.

[18] R.Milner. An algebraic definition of simulation between programs[A]. *In Proceeding of the 2nd international Joint Conference on Artificial Intelligence[C],*1971.481-489.

[19] R. Milner. Communicating and Mobile Systems: the $\pi$-Calculus. Cambridge University Press, 1999.

[20] T. Murata. Petri nets: Properties, Analysis and Applications. *Proc. of the IEEE 77(4),* 1999.

[21] E. Pastor, J. Cortadella and O. Roig. Symbolic Analysis of Bounded Petri Nets. *IEEE Transactions on Computers 50(5),* 2001.

[22] A.P. Sheth, W.M.P. van der Aalst. Arpinar. Processes Driving the Networked Economy. IEEE Concurrency 7(3): 18-31, 1999.

DEPARTMENT OF MATHEMATICS, NANJING UNIVERSITY, P.R. CHINA