Efficient Formal Verification in Banking Processes

Antonella Santone  
Dipartimento di Ingegneria, Università degli Studi del Sannio, Italy  
e-mail: santone@unisannio.it

Valentina Intilangelo  
e-mail: intivale@hotmail.it

Domenico Raucci  
Dipartimento di Economia, Università degli Studi di Chieti-Pescara  
e-mail: d.raucci@unich.it

Abstract—Model checking is a very used method to verify concurrent and distributed systems which is traditionally applied to computer system design. We examine the applicability of model checking to validation of Business Processes that are mapped through the systems of Workflow Management. The use of model checking in business domain is still not widely used. This is due also to the state explosion problem, which says that the state space grows exponentially in the number of concurrent processes. In this paper we consider property-based methodology developed to combat the state explosion problem. Our focus is two fold; firstly we show how model checking can be applied in the context of business modelling and analysis and secondly we evaluate and test the property-based methodology using as a case study a real-world banking workflow of a loan origination process. Our investigations suggest that the business community, especially in the banking field, can benefit from this efficient methodology developed in formal methods since it can detect errors that were missed by traditional verification techniques, and being cost-efficient, it can be adopted as a standard quality assurance procedure. We show and discuss the experimental results obtained.

Keywords—Business Process Management; Formal Methods; CCS; Workflow Verification; Banking Process.

I. INTRODUCTION

The application of formal techniques consists of an algorithmic approach to verification of such systems that can be represented by a formal model. Several techniques for formal verification have been developed over the past three decade among them model checking. In the model checking framework, systems are modelled as transition systems and requirements are expressed as formulae in temporal logic. A model checker then accepts two inputs, a system described, for example, in process-algebraic notations and a temporal formula, and returns “true” if the system satisfies the formula and “false” otherwise.

Model checking is a promising technique for the improvement of software quality. However, it requires detailed specifications of systems and requirements, and is therefore not very accessible, above all in certain restricted fields of application. One of these domains is business process management. In particular, we examine the applicability of model checking to validation of Business Processes that are mapped through the systems of Workflow Management. The use of this formal method in the domain of business process management however, is still not widely used. This is due also to the state explosion problem, which says that the state space grows exponentially in the number of concurrent processes. In fact, the parallelism between the processes of the system leads to a number of reachable states which may become very large, in some cases on the order of millions or billions of states. When the number of states is too large to fit in a computer’s main memory, verification quickly breaks down. In the business process taken into account, we came across the state explosion problem. Specifically, this problem has emerged as a result of a first modelling of the banking process, in which the excessive number of processes in parallel has made impracticable the verification using standard model checker.

Several approaches have been developed to solve or reduce the state explosion problem. In this paper to combat the state explosion problem we consider a methodology which is based on the property to be checked. Often the property one wants to check does not concern the whole transition system, but only some parts of it. An approach to reduce the number of states is the definition of suitable abstraction criteria by means of which a smaller transition system can be obtained, including only the parts that “influence” the property. In [1] a temporal logic is proposed, called selective mu-calculus, which has the characteristic that each formula allows immediately pointing out the parts of the transition system that do not alter the truth value of the formula itself. In particular, given a logic formula φ of the selective μ-calculus, only the transitions labelled by the actions syntactically occurring in φ have to be considered. In [2] a methodology is proposed based on the selective μ-calculus and on systems specified using Milner’s Calculus of Communicating Systems (CCS) [3], which is one of the most well known process algebras and it is largely used for modeling concurrent and distributed systems. Given a formula in the selective μ-calculus the CCS process is syntactically transformed into a smaller one (corresponding to a reduced transition system), where the reduction is driven by the formula to be checked. The formula is then checked on the reduced transition system. A prototype tool has been defined implementing the methodology.
The prototype produces the reduced transition system in the Concurrency Workbench of North Carolina (CWB-NC) [4] format. The CWB-NC is a verification environment including several different specification languages, among which CCS. The CWB-NC tool can then be used to check selective mu-calculus formulae, which can be easily translated into the temporal logic mu-calculus, available in the CWB-NC.

Our focus is two-fold:

- we show how model checking can be applied in the context of business modelling and analysis;
- we evaluate and test the property-based methodology using, as a case study, a real-world banking workflow of a loan origination process.

More precisely, in this paper we use CCS for modelling the implementation of the banking process and we express in selective mu-calculus logic the properties of the expected behaviours.

Our results obtained by evaluating a real-world banking process are validated by comparing them to those obtained by the CWB-NC. Applying our approach we can prove the correctness of the system obtaining a considerable reduction of both state space size and time with respect to CWB-NC. This confirms that our approach is more scalable than CWB-NC.

Our investigations suggest that the business community, especially in the banking field, can benefit from this efficient methodology developed in formal methods since it can detect errors that were missed by traditional verification techniques, and being cost-efficient, it can adopted as a standard quality assurance procedure.

The remainder of the paper is organized as follows. Section II is a review of the basic concepts of formal methods, while Section III briefly describes the property-based approach for model checking and its applicability in the business process management in banking industry. In Section IV the experimental results we obtained are reported and discussed. Finally, comparisons with related work and our conclusions are presented in Section V.

II. Formal Methods

In this section we introduce the basic concepts of formal methods. Formal methods are mathematically based languages, techniques, and tools for specifying and verifying complex systems. For applying formal methods, we need:

1) a precise notation for defining systems.

Specification is the process of describing a system. As starting point, we assume that the system behaviour is represented as an automaton. It basically consists of a set of nodes together with a set of labelled edges between these nodes. A node represents a system state, while a labelled edge represents a transition from one system state to the next. That is, if the automaton contains an edge \( s \xrightarrow{a} s' \), then the system can evolve from state \( s \) into state \( s' \) by the execution of action \( a \). One state is selected to be the root state, i.e., the initial state of the automaton. However, for the purpose of mathematical reasoning it is often convenient to represent the automaton algebraically in the form of terms. For this aim, we use Milner’s Calculus of Communicating Systems (CCS) [3]. CCS is one of the most well known process algebras and it is largely used for modeling concurrent and distributed systems. Readers unfamiliar with CCS are referred to [3] for further details. CCS contains basic operators to build finite processes, communication operators to express concurrency, and some notion of recursion to capture infinite behaviour. More precisely, the syntax of processes is the following:

\[
p := nil \mid \alpha.p \mid p + p \mid p|p \mid p\setminus L \mid p[f] \mid x
\]

where \( \alpha \) ranges over a finite set of actions \( A = \{\tau, a, \bar{a}, b, \bar{b}, \ldots\} \). Input actions are labeled with “non-barred” names, e.g. \( a \), while output actions are “barred”, e.g. \( \bar{a} \). The action \( \tau \in A \) is called internal action. The set \( L \) ranges over sets of visible actions \( (A - \{\tau\}) \), \( f \) ranges over functions from actions to actions, while \( x \) ranges over a set of constant names; each constant \( x \) is defined by a constant definition \( x \overset{\text{def}}{=} p \).

We give the semantics for CCS by induction over the structure of processes.

- The process \( \text{nil} \) can perform no actions.
- The process \( \alpha.p \) can perform the action \( \alpha \) and thereby become the process \( p \).
- The process \( p + q \) can behave either as \( p \) or as \( q \).
- The operator \( | \) expresses parallel composition: if the process \( p \) can perform \( \alpha \) and become \( p' \), then \( p|q \) can perform \( \alpha \) and become \( p'|q \), and similarly for \( q \).
- The operator \( \langle \) expresses the restriction of actions. If \( p \) can perform \( \alpha \) and become \( p' \), then \( p\setminus L \) can perform \( \alpha \) to become \( p'|L \) only if \( \alpha, \bar{\alpha} \notin L \). The operator \( [f] \) expresses the relabeling of actions. If \( p \) can perform \( \alpha \) and become \( p' \), then \( p[f] \) can perform \( f(\alpha) \) and become \( p'[f] \).
- Each relabeling function \( f \) has the property that \( f(\tau) = \tau \). Finally, a constant \( x \) behaves as \( p \) if \( x \overset{\text{def}}{=} p \).

The operational semantics of a process \( p \) is a labelled transition system, i.e. an automaton whose states correspond to processes (the initial state corresponds to \( p \)) and whose transitions (arcs) are labelled by actions in \( A \). This automaton is called standard transition system of \( p \) and denoted by \( S(p) \).

2) a precise notation for defining properties.

This need can be solved using a temporal logic [5]. Temporal logics present constructs allowing to state in a formal way that, for instance, all scenarios will respect some property at every step, or that some particular event will eventually happen, and so on. The most noticeable examples are:

- Safety properties, which state that an undesirable situation will never arise. For instance, in the banking sector, the requirements can forbid that the system permits to authorize the granting of a bank loan without
having received the information necessary for this purpose from the bank;
- **Liveness properties**, which state that some actions will always be followed by some reactions; a typical example is to check that every loan application to the bank is followed by a positive outcome (approval) or a negative one (refusal).

A model checker then accepts two inputs, a system described, for example, in process-algebraic notations and a temporal formula, and returns “true” if the system satisfies the formula and “false” otherwise. In this paper we use the logic selective mu-calculus [1]. It was defined with the goal of reducing the number of states of the transition systems in such a way that the reduction is driven by the formulae to be checked, and in particular by the syntactic structure of the formulae. The selective mu-calculus is a variant of the mu-calculus [5], and differs from it in the definition of the modal operators. The syntax of the selective mu-calculus is the following, where $K$ and $R$ range over sets of actions, while $Z$ ranges over a set of variables:

$$
\phi ::= \mathit{tt} | \mathit{ff} | Z | \phi_1 \lor \phi_2 | \phi_1 \land \phi_2 | [K]_R \phi | (K)_R \phi | \nu Z.\phi | \mu Z.\phi
$$

The satisfaction of a formula $\phi$ by a state $s$ of a transition system, written $s \models \phi$, is defined as follows:

- each state satisfies $\mathit{tt}$ and no state satisfies $\mathit{ff}$;
- a state satisfies $\phi_1 \lor \phi_2$ ($\phi_1 \land \phi_2$) if it satisfies $\phi_1$ or (and) $\phi_2$.
- $[K]_R \phi$ and $(K)_R \phi$ are the selective modal operators: they require that the formula $\phi$ is satisfied after the execution of an action of $K$, provided that it is not preceded by any action in $K \cup R$. More precisely:
  - $[K]_R \phi$ is satisfied by a state which, for every performance of a sequence of actions not belonging to $R \cup K$, followed by an action in $K$, evolves in a state obeying $\phi$.
  - $(K)_R \phi$ is satisfied by a state which can evolve to a state obeying $\phi$ by performing a sequence of actions not belonging to $R \cup K$ followed by an action in $K$.

The precise definition of the satisfaction of a closed formula $\varphi$ by a state $s$ (written $s \models \varphi$) is given in [1]. As in standard mu-calculus, a fixed point formula has the form $\mu Z.\varphi$ and $\nu Z.\varphi$ where $\mu Z.\varphi$ binds free occurrences of $Z$ in $\varphi$. An occurrence of $Z$ is free if it is not within the scope of a binder $\nu Z.\varphi$. A formula is closed if it contains no free variables. $\mu Z.\varphi$ is the least fix-point of the recursive equation $Z = \varphi$, while $\nu Z.\varphi$ is the greatest one. To give an intuition of their meaning, consider the formulae $\varphi = \mu Z.(\psi \lor (a \varphi Z))$ and $\varphi' = \nu Z.(\psi \land (a \varphi Z))$. A transition system satisfies $\phi$ if it can evolve to a state satisfying $\psi$ after a finite number of occurrences of action $a$ (ignoring all other actions), while it satisfies $\varphi'$ if it satisfies $\psi$ along any path containing $a$ (ignoring all other actions).

A transition system $T$ satisfies a formula $\phi$, written $T \models \phi$, if and only if $q \models \phi$, where $q$ is the initial state of $T$. A CCS process $p$ satisfies $\phi$ if $S(p) \models \phi$.

**Example 2.1**: We give some examples of selective mu-calculus formulae to explain the use of the selective operators.

- $\psi_1 = [a]_b \mathit{ff}$: “it is not possible to perform an action $a$ if an action $b$ has not been previously performed”.
- $\psi_2 = (a)_{\mathit{tt}}$: “it is possible to perform an action $a$ preceded by any action”.
- $\psi_3 = \nu Z.\mathit{ff} \land [a]_{b,c} \mathit{ff}$: “it always holds that, after an action $a$ has occurred, a successive $a$ cannot occur before either an action $b$ or an action $c$ has occurred”.

Let us consider the transition systems in Figure 1. It holds that:

$$
S_1 \not\models \psi_1 \quad S_2 \models \psi_1 \quad S_3 \not\models \psi_1
$$

$$
S_1 \models \psi_2 \quad S_2 \not\models \psi_2 \quad S_3 \models \psi_2
$$

$$
S_1 \models \psi_3 \quad S_2 \not\models \psi_3 \quad S_3 \not\models \psi_3
$$

3) an algorithm to check if a system satisfies a property.

Formal verification is a systematic process that uses mathematical reasoning to verify whether a design satisfies some requirements (properties). For this last need, several techniques for verification are provided among them model checking. In model checking the technique [6] one formulates certain wanted properties in a temporal logic. Each property is checked for validity against the model modelled as transition systems. A model checker then accepts two inputs, a transition system and a temporal formula, and returns true if the system satisfies the formula and false otherwise.

Roughly speaking, the check is performed as an exhaustive state space search that is guaranteed to terminate since the model is finite. The technical challenge in model checking is in devising algorithms that allow us to handle large search spaces.

One of the most popular environments for verifying concurrent systems is the Concurrence Workbench of New Century (CWB-NC) [4], which supports several different specification languages, among which CCS. In the CWB-NC the verification of temporal logic formulae is based on model checking [6].

**III. A PROPERTY-BASED APPROACH TO COMBAT THE STATE EXPLOSION PROBLEM IN A LOAN ORIGINATION PROCESS**

Formal methods cannot be easily scaled due to the state explosion problem, which says that the state space grows exponentially in the number of concurrent processes. In fact, the parallelism between the processes of the system leads to a number of reachable states which may become very large, in some cases on the order of millions or billions of states. When the number of states is too large to fit in a computer’s main memory, verification quickly breaks down. Several approaches have been developed to solve or reduce the state explosion problem.

In this paper we consider a methodology to combat the state explosion problem for model checking, based on properties
expressed using the selective mu-calculus. The basic characteristic of the selective mu-calculus is that the actions relevant for checking a formula $\phi$ are those ones explicitly mentioned in the modal operators used in the formula itself. Thus we define the set $\mathcal{O}(\phi)$ of occurring actions of a formula $\phi$ as the union of all sets $K$ and $R$ appearing in the modal operators $\langle[K]_{R}\psi, \langle K \rangle_{R}\psi \rangle$ occurring in $\phi$.

In the work [1] $\rho$-equivalence is defined, formally characterizing the notion of “the same behavior with respect to a set $\rho$ of actions”:

**two transition systems are $\rho$-equivalent if and only if they satisfy the same set of formulae with occurring actions $\rho$.**

The definition of $\rho$-bisimulation is based on the concept of $\alpha$-ending path: an $\alpha$-ending path is a sequence of transitions labeled by actions not in $\rho$ followed by a transition labeled by the action $\alpha$ in $\rho$. Two states $S_1$ and $S_2$ are $\rho$-bisimilar if and only if for each $\alpha$-ending path starting from $S_1$ and ending into $S_1'$, there exists an $\alpha$-ending path starting from $S_2$ and ending into a state $\rho$-bisimilar to $S_1'$, and vice-versa.

In [1] it is proved that “two transition systems are $\rho$-equivalent if and only if they satisfy the same set of formulae with occurring actions $\rho$.”

As a consequence, a formula of the selective mu-calculus with occurring actions in $\rho$ can be checked on any transition system $\rho$-equivalent to the standard one. Thus, improvements in model checking can be obtained by minimizing the transition system with respect to the actions in $\rho$. Obviously, the degree of reduction depends on the size of $\rho$ with respect to the size of the whole set $A$ of actions.

**Example 3.1: Consider the transition systems illustrated in Figure 1 and the formulae of the Example 2.1.** $S_1$ is $\{a, b\}$-equivalent to $S_3$. The two transition systems give the same value for the formulae containing only actions in $\{a, b\}$. In particular, they satisfy $\psi_2$, while they do not satisfy $\psi_1$. Note that $\mathcal{O}(\psi_1) = \{a, b\}$ and $\mathcal{O}(\psi_2) = \{a\} \subseteq \{a, b\}$. On the contrary $S_2$ is not $\{a, b\}$-equivalent to $S_3$, since it can perform an action $b$ without performing an action $a$.

In [2] a method is defined, which, given a CCS process $p$ and a set of interesting actions $\rho$ occurring in a formula $\phi$ to be checked, transforms $\rho$ into another process $q$, corresponding to a smaller transition system than that of $p$, on which $\phi$ can be equivalently checked. The tool is based on a set of syntactic transformations rules.

A prototype tool which implements the algorithm has been defined in [2]: it is written in SICStus Prolog [7]; the output is in the format of the CWB-NC tool: a file .cws, containing the reduced CCS process. Thus, both the construction of the transition system and the checking of the properties can be made using the CWB-NC environment.

In this paper we evaluate the methodology on a real-world banking workflow of a loan origination process.

A. Banking Process Overview

Formal techniques and tools, as stated above, have a number of features that make it possible to extend their use to specific business domains, including the formal verification of business processes that are mapped and administrated through the Workflow Management systems. The managers use these techniques generally to automate all or part of business processes by flushing documents, information and tasks from one participant to another, according to a set of procedural rules and according to economic and organizational control principles [8], [9]. With specific reference to dynamic workflows, i.e. those that evolve according to the events that occur and to the ways in which tasks are handled by users, a particular problem is represented by fairness since it is necessary to ensure that workflows have been adequately defined with respect to desired properties. To deal with this situation it is possible to use formal verification techniques and associated tools, like model checking. The underlying idea consists in representing each business process, including interactions between human components, databases, and hardware/software elements of the information system as a finite state transition system. In such way, the application of formal techniques can also be relevant for strategic management decisions regarding the reliability and correctness of the design and implementation of critical business processes and according to the approaches of Business Process Improvement [10] and Business Process Reengineering [11]. However, the complexity of model checking ends to limit its applicability to restricted areas where its use is justified in the light of economic considerations arising from the natural trade-off between the costs of implementation and the benefits achieved. We refer in particular to those business processes so-called business critical or commercially-critical, where the characteristics of fairness and integrity are prerequisites for the success of a market or business transaction [12], [13], [14]. In management literature there are many classifications of business processes according to the different purposes, fields of analysis and business sectors. In banking industry are generally identified these types of business processes: Management processes, that govern the bank (examples include Strategic Management, Risk Management, Internal Audit, etc.); Support processes, which support the core processes (examples include Accounting, Human Resource, ICT, Security, Technical support, etc.); Operating processes.
better the type of client that requested the funding. If the latter corresponds to the set of features and price conditions that fits the products that are part of the bank's offer, the one which of the loan credit-worthiness. Afterwards it is selected among of the same, thanks to which it is possible to express the level synthetic judgment of reliability based on assets and income controls in order to assign a rating to the customer, which is a Then there is the need to make a series of internal and external simultaneous location in the segmentation made by the bank.

Into account the information provided, the process proceeds record in the database if he/she is a new customer. Taking into account the new procedures supported by Workflow Management tools. We introduce, at this point, the banking process in question, providing a broad overview of the same thanks to typical logic of workflow tools.

B. Loan Origination Process in Bank

According to the mentioned outlook we evaluate and test the property-based methodology using as a case study a real-world banking workflow of a loan origination process. We think that the business community, especially in the banking field, can benefit from this efficient methodology developed in formal methods area to prevent significant errors. The application of model checking techniques to verification all of part of this typical banking process implies an effective separation of duties policies, according to economic and organizational control principles, and a correct implementation of the features of IT [19], [20]. In this way it is possible to preserve an operational flexibility and to realize, contextually, a rigid control in the loan origination process (i.e. tight access control and at the same time support for discretionary delegation of workflow tasks and rights) avoiding the undesired propagation of access rights or indirect access through some other granted resource. At the end these anomalies and inefficiency could prejudice the good result of the loan operation process [21], [22]. This banking process generally includes a number of phases whose lasted various from bank to bank [23], [24]. The process begins whenever a customer of the bank applies for a loan and it starts with the update of customer data if he/she is already present in the registry or with the insertion of a new record in the database if he/she is a new customer. Taking into account the information provided, the process proceeds with the identification of the type of customer and with its simultaneous location in the segmentation made by the bank. Then there is the need to make a series of internal and external controls in order to assign a rating to the customer, which is a synthetic judgment of reliability based on assets and income of the same, thanks to which it is possible to express the level of the loan credit-worthiness. Afterwards it is selected among the products that are part of the banks offer, the one which corresponds to the set of features and price conditions that fits better the type of client that requested the funding. If the latter is satisfied with the proposal received, the bank proceeds to the signing of the contract and, therefore, the formalization of the agreement, reaching the end of the process.

C. Specification and Properties

In order to apply formal verification of the banking process, first we specify each component of the system using CCS. Figure 2 shows a typical loan origination process in the banking domain, as previously described. To formally verify the banking process it is firstly required to give a formal specification of each component of the system. Each component of the banking process involves complex interactions since the component can dialogue with other components. For example, the customer dialogue with the first phase clerk, for instance to ask for a loan request.

Figure 3 shows a view of all the processes (the boxes in the figure) involved in this loan origination process as well as the synchronisations between these entities. Synchronisations are denoted using lines joining the involved agents with actions and their direction (prefixed by a quote for emissions).

All components have been specified in CCS; in Table I we show only the CCS specification of the first phase clerk, just to give the reader the flavor of the approach followed. From a textual CCS specification it is possible to generate the corresponding labeled transition system, which can be successively used for model checking. We prefer textual syntax since it is more adapted to proof-writing and formal reasoning, as well as to the description of large-scale systems. Graphical notations are anyway complementary and can be used by user-friendly front-ends.

The banking process was designed to guarantee the following properties:

\[ P_1 \] “after a request by the customer for a loan, the procedure terminates either with success or with a failure”.

\[ P_2 \] “the customer receives a copy of the contract only if both the loan application was successful and the customer has affixed his signature”.

\[ P_3 \] “the customer obtains a negative result for the loan request only after the supervisor has examined three times the characteristics proposed by the clerk preprocessor”

IV. EXPERIMENTAL RESULT OBTAINED BY VERIFYING THE BANKING PROCESS

We now apply our methodology to verify the banking process. First we express the required properties using the selective mu-calculus. Then we apply our tool to transform the specification into a smaller one, where the reduction is driven by the actions occurring in the formulae. Finally, we check the properties on the reduced specification.

The properties described in the previous section can be expressed with the selective mu-calculus in the following way.

\[ P_1 = [loanRequest]_0 (\langle success\rangle_0 \\& \langle failure\rangle_0) \]

\[ P_2 = [copyContract]_0 (\langle success\rangle_0 \\& \langle failure\rangle_0) \]

\[ P_3 = [failure]_0 (\langle check\rangle_0 \\& \langle signature\rangle_0) \]
It holds that the sets of interesting actions of each of the above properties are:

\[ \rho_1 = \mathcal{O}(P_1) = \{\text{loanRequest, success, failure}\} \]
\[ \rho_2 = \mathcal{O}(P_2) = \{\text{copyContract, signature, success}\} \]
\[ \rho_2 = \mathcal{O}(P_2) = \{\text{failure, check_3}\} \]

To make the experiments we used the CWB-NC tool on a 64 bit, 2.67 GHz Intel i5 CPU equipped with 8 GiB of RAM and running Gentoo Linux. Table II shows the number of states and transitions of \( P_i \)-reduction, \( i \in \{1, 2, 3\} \), where \( P_i \)-reduction is the reduced CCS process obtained by applying the property-based methodology to banking process with \( \rho_i \), for each \( i \in \{1, 2, 3\} \). It is worth noting that the reduction we perform of the states and transitions is significant. Note that in some cases we can reach a reduction of 99%.

Table III shows the time employed by the CWB-NC to check the formulae and the time reduction obtained with our methodology.

In general, the usefulness of our tool depends both on the number of actions occurring in the formula and on the structure of the formula to be checked.

It can be shown that adding more customers, differently from our approach, the CWB-NC is not able to give an answer due to the big number of states. This confirms that our approach is more scalable than CWB-NC.

V. Conclusions and Related Work

The experimental result obtained by our property-based methodology have allowed to verify that the CCS model, applied to the loan origination process in banking industry, satisfies all the properties formalized in selective mu-calculus. Even if only one property had been rejected by the model checker it would be to plan again this banking process and to modify, consequently, the descriptive model. The application of formal technique can offer an effective support in the initiatives of business process engineering and improvement where it is critical guarantee the safety and the reliability of the software applications. Model checking can be used to prove the correctness and the reliability of the IT used to support tasks and operation rules in work organization but can also be a useful tool for the software houses (in the segment Business to Business - B2B) that should integrate the techniques for formal verification in their business. Model checking moreover is a powerful technique to validation of business processes that are mapped through the systems of workflow.
management. It can be applied for formal verification of reliability and correctness of the design and implementation of critical business processes (i.e., loan origination process in bank) so that to prevent significant errors and costs in the run-time phase. However, the complexity of model checking justifies its use in the light of economic considerations (or security factors) arising from the natural trade-off between the costs of implementation and the benefits achieved. A real-world banking workflow of a loan origination process was chosen as a case study to demonstrate the usefulness of our methodology which reduces the state explosion problem of access rights or indirect access through some other users and preserves an operational flexibility according to the correct implementation of organizational controls in a bank. Similarly to our investigation, in [25] the authors analyse the verification of the safety of a system which is also one of the goals of access control research. In this way they consider the interaction between delegation and the revocation activities, in the context of dynamic separation of duty policies, and the delegation and revocation of access right/authorizations and tasks/obligations, mainly focusing on organizational control principles. In this context, the application of model checking can be employed in the verification of the impossibility from an only user of the system to carry out all the activities and all the authorizations necessary to finish the process or the impossibility to realize such operations in complete autonomy (i.e., avoiding frauds and other deliberate or accidental threats) reproducing a double control typical of the critical processes. However, differently by our property-based model checking approach, in [25] the authors propose a formal methodology using SMV tool. Moreover, our aim is also to reduce the state explosion problem arising when dealing with complex systems such as the loan origination processes in banking industry. As shown by the above experiments, our approach obtains good results when compared with CWB-NC both in performance, i.e., the speed at which the tool returns its
results, and its scalability, i.e., the extent to which the tool can manage increasingly large systems. Moreover, in [26] the authors examine the applicability of existing model checking methods and tools to e-commerce software systems developed at Intershop.

It is worth noting that the degree of reduction we obtain with our approach essentially depends on the number of actions syntactically occurring in the modal operators of the formula to be checked. When these actions are almost the whole set $A$ of actions, we do not obtain significant reductions.

All the verification systems which base their behaviour on the analysis of transition systems can profit from our formula-based reductions. In particular, our approach can be integrated with the on-the-fly methodology [27]. For example, given a formula of the selective mu-calculus with occurring actions $\rho$, we can on-the-fly verify it during the generation of the reduced transition system, instead of during the generation of the standard one. A main difference of our approach from other existing ones [28] is that the reduction is not provided by the user (often based on an informal reasoning and proved correct case by case), but it is completely automatic and transparent to the user. In fact, the reductions are driven by the formulae to be checked. The papers [29], [30] follow the partial order approach to model checking, in which only a representative is considered among all interleavings of actions generated by a parallel composition. The properties well handled by these approaches do not concern precedence relations between actions, and can be profitably used to prove, for example, deadlock freedom. In our approach, properties concerning precedence relations between actions are in general described by formulae which induce a consistent reduction of the state space of the transition system. On the contrary, the formula describing deadlock freedom induces no reduction, since it involves all actions. Thus the two approaches can be considered complementary.

<table>
<thead>
<tr>
<th>standard transition system</th>
<th>CWB-NC</th>
<th>reduced transition system</th>
<th>our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>71.775s</td>
<td>0.006s</td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td>50.084s</td>
<td>0.005s</td>
<td></td>
</tr>
<tr>
<td>$P_3$</td>
<td>50.961s</td>
<td>0.005s</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III:** Verification time of $P_1$, $P_2$, and $P_3$

[References]


