Application of Equivalence Checking in a Loan Origination Process in Banking Industry

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, Italy  
e-mail: d.raucci@unich.it

Abstract—Equivalence checking is traditionally applied to computer system design. It is a promising formal technique for the improvement of software quality. However, it requires detailed specifications of systems and is therefore not very accessible, above all in certain restricted fields of application. One of this domain is business process management. In particular, we examine the applicability of equivalence checking to validation of Business Processes that are mapped through the systems of Workflow Management. The usage of formal methods in business domain, 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 fact, 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 for checking process equivalence we consider an efficient procedure, based on heuristic search [1], proposed in [2]. The procedure is applied to processes defined through a specification language very compact, the Calculus of Communicating Systems (CCS)[3] defined by Milner, which is one of the most well known process algebras and it is largely used for modeling concurrent and distributed systems. The approach uses a heuristic function that suggests to expand first the states that offer the most promising way to deduce that two systems are not equivalent. This makes it possible to avoid the exhaustive exploration of the global state graph of the two systems when they are not equivalent. One of the authors of this paper has contributed to the design of Grease (GReedy Algorithm for System Equivalence), a C++ tool supporting the heuristic approach to check equivalence of CCS processes. The use of Grease on a sample of small CCS processes has shown a significant reduction of both state-space size and time, compared to traditional equivalence checking algorithms. Our proposal is two-fold: (i) we show how equivalence checking can be applied in the context of business modelling and analysis; (ii) we evaluate and test the heuristic-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 the process algebra area to prevent significant errors. We show and discuss the experimental results obtained.

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

I. INTRODUCTION

Formal methods are powerful techniques for specifying and verifying complex systems. Several techniques for formal verification have been developed over the past three decade among them equivalence checking. Equivalence checking is the process of determining whether two systems are equivalent to each other according to some mathematically-defined notion of equivalence. Equivalence checking is typically used to verify if a system design conforms to its high-level service specification. Although equivalence checking is currently not applicable to all domains, it is useful for certain restricted fields of application. One of this domain is business process management. In particular, we examine the applicability of equivalence checking to validation of Business Processes that are mapped through the systems of Workflow Management. The usage of this formal method in the domain of business process management however, is still not widely used. This is
a real-world banking process are validated by comparing them to those obtained by a state-of-the-art tool for equivalence checking, i.e., the Concurrency Workbench of New Century (CWB-NC) [4], which one of the most popular environments for verifying concurrent system. 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. Actually, for the proposed case study, differently from Grease, the CWB-NC is not able to give an answer, after waiting more than half an hour. This is due to the big number of states. 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 the process algebra area to prevent significant errors.

The remainder of the paper is organized as follows. Section II is a review of the basic concepts of equivalence checking, while Section III briefly describes the heuristic-based approach for equivalence 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. EQUIVALENCE CHECKING

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. Several techniques for formal verification have been developed over the past three decades among them equivalence checking.

For applying equivalence checking we first need a precise notation for defining systems. Specification is the process of describing a system. As starting point, we assume that 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 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. The semantics of a CCS term $p$ is precisely defined by means of the structural operational semantics. The semantic definition is given by a set of conditional rules describing the transition relation of the automaton corresponding to the behavior expression defining $p$. This automaton is called standard transition system for $p$ and is denoted by $S(p)$.

More precisely, the syntax of processes is the following:

$$p ::= \text{nil} \mid \alpha.p \mid p + p \mid p\{p\} \mid p.L \mid p[f] \mid x$$

where $\alpha$ ranges over a finite set of actions $\mathcal{A} = \{\tau, a, \pi, b, \ldots\}$. Input actions are labeled with “non-barred” names, e.g. $a$, while output actions are “barred”, e.g. $\pi$. The action $\tau \in \mathcal{A}$ is called internal action. The set $L$ ranges over sets of visible actions ($\mathcal{A} - \{\tau\}$), $f$ 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 $x \defeq p$.

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 $\{p\}$ 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 $q$. Furthermore, if $p$ can perform a visible action $l$ and become $p'$, and $q$ can perform $l$ and become $q'$, then $p[q]$ can perform $\tau$ and become $p'[q']$.
- The operator $\backslash$ expresses the restriction of actions. If $p$ can perform $\alpha$ and become $p'$, then $p.L$ can perform $\alpha$ to become $p'/L$ only if $\alpha, \pi \notin L$.
- 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 $x \defeq p$.

Equivalence checking is the process of determining whether two systems are equivalent to each other according to some mathematically defined notion of equivalence, such as strong and weak equivalence previously defined. The CCS language, like all other process algebras, can be used to describe both implementations of processes and specifications of their expected behaviours. Therefore CCS supports the so-called single language approach to process theory, that is, the approach in which a single language is used to describe both actual processes and their specifications. One process description, say SYS, may describe an implementation, and another, say SPEC, may describe a specification of the expected behaviour. This approach to program verification is also sometimes called implementation verification. For example, if the observable behaviour of a communications protocol is identical to that of a perfect communication channel that delivers all messages in order, then it would be justifiable to deem the protocol correct.

One of the most popular environments for verifying concurrent systems is the Concurrency Workbench of New Century (CWB-NC) [4], which supports several different specification languages, among which CCS. In the CWB-NC the verification can be based both on model checking [5] and on equivalence checking.
III. HOW TO GREASE 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. We now present a methodology to combat the state explosion problem for equivalence checking.

In [2] an efficient approach based on heuristic search techniques to check equivalences has been defined. In general, a heuristic search strategy [1] uses an evaluation function to determine the order in which the nodes in the search graph are selected for expansion. The evaluation function measures the distance to a goal node based on an heuristic function which gives the estimated shortest distance from the given node to a goal node. In [2] the equivalence checking has been formalized as a search problem on AND/OR graphs. Moreover, an efficient equivalence checking procedure has been proposed, defining a heuristic function that suggests to expand first the states that offer the most promising way to deduce that two systems are not equivalent. This makes it possible to avoid the exhaustive exploration of the global state graph of the two systems when they are not equivalent. One of the authors of this paper has contributed to the design of Grease1 (GReedy Algorithm for System Equivalence), a C++ tool supporting the heuristic approach to check equivalence of CCS processes. The use of Grease on a sample of small CCS processes has shown a significant reduction of both state-space size and time, compared to traditional equivalence checking algorithms. In this paper we evaluate the methodology on a real-world banking workflow of a loan origination process.

A. Banking Process Overview

As stated above, the application of formal techniques consists of an algorithmic approach to verification of such systems that can be represented by a formal model. According to the mentioned outlook, these methodologies are primarily related to verification of hardware and software systems. However, formal techniques and tools 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 systems of Workflow Management. The latter are generally used by managers 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. 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 because it is necessary to ensure that workflows have been defined adequately with regard to desired properties. To deal with this situation it is possible to use the formal verification techniques and the associated tools. 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 [6], [7], [8]. However, the complexity of this approach 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 [9], [10], [11], [12], [13]. The above-mentioned banking process of loan origination fits into this context. The need to increase efficiency and reduce costs is pushing for a long time many banks to review internal processes, models, organizational procedures and technological applications, in order to ensure cost-effectiveness and maintain, at the same time, a high level of quality of services provided to customers. In this vein are situated very important projects including the reengineering of the process of investigation and provision of different forms of credit on mortgages and personal loans, aimed in particular to the introduction of new procedures supported by Workflow Management tools [14], [15], [16]. We introduce, at this point, the banking process in question, providing a broad overview of the same thanks to typical logic of workflow tools. The process is instantiated 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 bank’s 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.

IV. EXPERIMENT RESULTS

In this section we present and discuss our experience with using the Grease, implementing the heuristic-based approach, to verify the banking process previously described. The aim is to evaluate the performances of the approach and compare it against the CWB-NC. Experiments were executed on a 64 bit, 2.67 GHz Intel i5 CPU equipped with 8 GiB of RAM and running Gentoo Linux. The tool is freely available for download at the url http://www2.ing.unipi.it/~a080224/grease/.

Figure 1 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 2 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 component 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 equivalence 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 results for the verification of our system using weak equivalence are shown in Table II. In the table the second column shows the results in terms of generating states (gen) and time, expressed in seconds, using Grease. The third column should show the number of generated states and time resulting from the CWB-NC. The CCS specification (namely $p$) representing the CCS banking process has 105409 states and 635905 transitions. To apply equivalence checking we have to specify in CCS also the expected behaviours. Thus, in the first experiment we represent as CCS process (namely $q$) the following requirement: "the customer receives a copy of the contract not only if the loan application was successful but also when another condition is satisfied, i.e., when the customer he has affixed his signature."

In the second experiment, we represent as CCS process (namely $q$) the following requirement: "after a request by the customer for a loan, the procedure terminates with success".

The result is that $p$ is not weak equivalent to $q$, since obviously the procedure can terminate also with a failure. It is worth pointing out that, differently from Grease, the CWB-NC is not able to give an answer, after waiting more than half an hour for both the requirements. Therefore no verification can be performed. This is due to the big number of states. This confirms that our approach is more scalable than CWB-NC.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Grease gen</th>
<th>Grease time</th>
<th>CWB-NC gen</th>
<th>CWB-NC time</th>
</tr>
</thead>
<tbody>
<tr>
<td>first requirement</td>
<td>16380</td>
<td>64.85</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>second requirement</td>
<td>8163</td>
<td>8.53</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

TABLE II: Results - weak equivalence.

V. Conclusions and Related Work

The work presented in this paper shows that equivalence checking can be very useful to enhance the quality in the domain of business management. However, using traditional equivalence checkers we find some difficulties to prove the concreteness of real-world banking. This is mainly due to the state explosion problem. To combat this problem we consider an efficient procedure, based on heuristic search, which uses
Fig. 2: Graphic representation of the interactions among all the components involved in a loan origination process.

TABLE I: The CCS first phase clerk specification

| proc FirstPhaseClerk = loanRequest.'informationRequest.customerInformation.{ |
| 'insertCustomerData.'ratingRequest.FirstPhaseClerk + |
| 'updateCustomerData.'ratingRequest.FirstPhaseClerk |

a heuristic function that suggests to expand first the states that offer the most promising way to deduce that two systems are not equivalent. This makes it possible to avoid the exhaustive exploration of the global state graph of the two systems when they are not equivalent. We use Grease (GREedy Algorithm for System Equivalence), a C++ tool supporting the heuristic approach to check equivalence of CCS processes.

Our investigations suggest that the business community, especially in the banking filed, can benefit from the efficient heuristic-based methodology developed in the process algebra area to prevent significant errors. A real-world banking workflow of a loan origination process was chosen as a case study to demonstrate the usefulness of Grease which reduces the state explosion problem.

As shown by the above experiments, Grease 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.

Similarly to our investigation, in [17], the authors analyse the verification of the safety of a system (i.e., avoiding the undesired propagation of access rights or indirect access
through some other granted resource) which is one of the goals of access control research. In particular, they consider delegation and revocation functionalities mainly focusing on organizational control principles. However, differently by our equivalence-checking approach, they propose a model-checking based 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.

The most challenging task when applying automated formal verification in practice is to conquer the state explosion problem. Several approaches have been developed to solve or reduce the state explosion problem for model checking. Among them, reduction techniques based on process equivalences [18], symbolic model checking techniques [19], on-the-fly techniques [20], partial order techniques [21], compositional techniques [22], [23], and abstraction approaches [24].

For equivalence checking algorithms with minimal space complexity are of particular interest. Two algorithmic families can be considered to perform the equivalence checking. The first one is based on refinement principle: *given an initial partition, find the coarsest partition stable with respect to the transition relation* see for example the algorithm proposed by Paige and Tarjan in [25]. The other family of algorithms is based on a Cartesian product traversal from the initial state [26], [27]. These algorithms are both applied on the whole state graph, and they require an explicit enumeration of this state space. This approach leads to the well-known state explosion problem. Classical reduction algorithms already exist [28], [29], but they can be applied only when the whole state space has been computed, which limits their interest. A possible solution is to reduce the state graph before performing the check as shown in [30], where symbolic representation of the state space is used.

Other algorithms are the ones by Bustan and Grumberg [31] and by Gentilini, Piazza and Policriti [32]. For an input graph with \( N \) states, \( T \) transitions and \( S \) simulation equivalence classes, the space complexity of both algorithms is \( O(S^2 + N \log S) \). The approach of Gentilini et al. represents the simulation problem as a generalised coarsest partition problem.

**REFERENCES**


