Decidability and Complexity of Simulation Preorder for Data-Centric Web Services

Lakhdar Akroun[1], Boualem Benatallah[2],
Lhouari Nourine[1], and Farouk Toumani[1]

[1] LIMOS, CNRS, Blaise Pascal University, Clermont-Ferrand, France
{akroun,nourine,ftoumani}@isima.fr
[2] CSE, UNSW, Sydney, Australia
boualem.benatallah@gmail.com

Abstract. This paper studies the problem of checking the simulation preorder for data-centric services. It focuses more specifically on the underlying decidability and complexity issues in the framework of the Colombo model [1]. We show that the simulation test is EXPTIME-complete for Colombo services without any access to the database (noted $Colombo^{DB=\emptyset}$) and 2EXPTIME-complete when only bounded databases are considered (the obtained model is noted $Colombo^{bound}$). This is a decidability border since we have shown in previous work that the simulation test for unbounded Colombo is undecidable. Moreover, as a side effect of this work, we establish a correspondance between $Colombo^{DB=\emptyset}$, restricted to equality, and Guarded Variable Automata (GVA) [2]. As a consequence, we derive EXPTIME-completeness of simulation for GVA.

Keywords: data-centric services, simulation preorder, variable automata, verification and synthesis.

1 Introduction

Business protocols, and associated representation models (e.g., state machines [3, 4], , Petri-nets), are used for specifying external behavior of services. They open the opportunity for formal analysis, verification and synthesis of services. For example, business protocols have been used as a basis to develop techniques for compatibility and replaceability analysis of web services [5] and also to study the web service composition problem [6]. In the aforementioned research works, the *simulation preorder* [7] plays a fundamental role to solve the considered problems. Indeed, simulation preorder enables to formalize the idea that a given service is able to faithfully reproduce the externally visible behavior of another service.

Recently, the need of incorporating data as a *first-class citizen* in business protocols has been widely recognized and a number of research works has been carried out in this direction, laying the foundations of a *data-centric* approach to web services [1, 8, 9]. Formal models used to describe such specifications, called *data-centric services*, are essentially communicating guarded transitions systems

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 535–542, 2014.

in which transitions are used to model either the exchange of messages between a service and its environment (i.e. a *client*), or service's actions (i.e., read, write) over a global database shared among the existing services. A configuration (or a state) of a data-centric service is made of a control state of the transition system augmented with the current instance of the global database. The incorporation of data turns out to be very challenging since it makes service specifications infinite which leads, in most cases, to undecidability of many analysis and verification problems.

In this paper, we investigate the decidability and the complexity issues of service simulation in the framework of the Colombo model [1]. A Colombo service is specified as a guarded transition system, augmented with a global database as well as a set of variables that are used to send and receive messages. Two sources of infiniteness makes the simulation test difficult in this context: (i) the variables used by a service range over infinite domains and hence the number of potential concrete messages that can be received by a service in a given state may be infinite; (ii) the number of possible initial instances of the global database is infinite which makes the number of configurations of a service infinite.

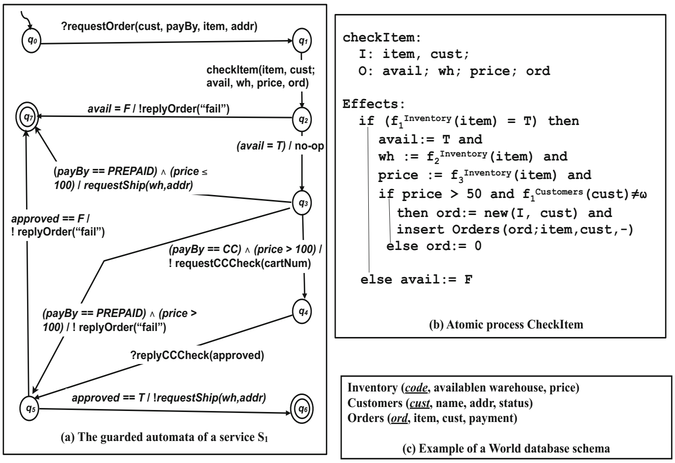
In a preliminary work [10], we showed that checking simulation in a Colombo model with unbounded accesses to the database, called $Colombo^{unb}$, is undecidable. To complete the picture and provide a decidability border of simulation in the Colombo framework, we study in this paper the simulation problem in the case of Colombo services with bounded databases (i.e. the class of Colombo services restricted to global databases having a number of tuples that cannot exceed a given constant k). Such a class is called $Colombo^{bound}$. We show that the simulation is 2EXPTIME-complete for $Colombo^{bound}$. The proof is achieved in two steps: (i) first we show that checking simulation is EXPTIME-complete for Colombo services without any access to the database (called DB-less services and denoted $Colombo^{DB=\emptyset}$). $Colombo^{DB=\emptyset}$ services are also infinite-state systems, because they still manipulate variables ranging over infinite domains. However, a finite symbolic representation of such services can be obtained by partitioning the original infinite state space into a finite number of equivalence classes. As a side effect of this work, we establish a correspondence between $Colombo^{DB=\emptyset}$, restricted to equality, and Guarded Variable Automata (GVA) [2]. As a consequence, we derive EXPTIME-completeness of simulation for GVA; (ii) then we show that the simulation test for $Colombo^{bound}$ services exponentially reduces to checking simulation in $Colombo^{DB=\emptyset}$. The exponential blow-up is an unavoidable price to pay since we prove that simulation in $Colombo^{bound}$ is 2EXPTIME-complete. For space reasons, the proofs are omitted and are given in the extended version of this paper [11].

Organization of the paper. We start by giving an overview of the Colombo framework in Sect.2, then we present our results on $Colombo^{DB=\emptyset}$ and $Colombo^{bound}$ in Sect.3. Finally, we discuss related work in Sect.4 while we conclude in Sect.5.

2 Overview on the Colombo Model

A *world* database schema, denoted $\mathcal{W}$, is a finite set of relation schemas having the form $R(A_1, \ldots, A_k; B_1, \ldots, B_n)$, where the A_is form a key for R. A world database (or a world instance) is an instance over the schema $\mathcal{W}$. Let $R(A_1, \ldots, A_k; B_1, \ldots, B_n)$ be a relation schema in $\mathcal{W}$, then $f_j^R(A_1, \ldots, A_k)$ is an access function that returns the j-th element of the tuple t in R identified by the key $(A_1, \ldots, A_k)$ (i.e. $j \in [1, n]$). Given a set of constants C and variables V, the set of accessible terms over C and V is defined recursively to include all the terms constructed using C, V and the f_j^R functions.

Example 1. Figure 1(c) depicts an example of a world database. For example, the access to the relation Inventory(code, available, warehouse, price) is only possible through the access function $f_j^{Inventory}(code)$ with $j \in [1, 3]$.


?requestOrder(cust, payBy, item, addr)
q0
q1
checkItem(item, cust; avail, wh, price, ord)
avail = F / !replyOrder("fail")
q7
q2
(avail = T) / no-op
(payBy == PREPAID) ∧ (price ≤ 100) / requestShip(wh,addr)
q3
approved == F / ! replyOrder("fail")
(payBy == CC) ∧ (price > 100) / ! requestCCCheck(cartNum)
(payBy == PREPAID) ∧ (price > 100) / ! replyOrder("fail")
q4
?replyCCCheck(approved)
approved == T / !requestShip(wh,addr)
q5
q6

(a) The guarded automata of a service S1

```
checkItem:
  I: item, cust;
  O: avail; wh; price; ord

Effects:
  if (f_1^Inventory(item) = T) then
    avail:= T and
    wh := f_2^Inventory(item) and
    price := f_3^Inventory(item) and
    if price > 50 and f_1^Customers(cust)≠ω
      then ord:= new(I, cust) and
      insert Orders(ord;item,cust,-)
     else ord:= 0

  else avail:= F
```

(b) Atomic process CheckItem

Inventory (*code*, availablen warehouse, price)
Customers (*cust*, name, addr, status)
Orders (*ord*, item, cust, payment)

(c) Example of a World database schema

Fig. 1. Example of Colombo service (from [1])

In the Colombo model, service *actions* are achieved using the notion of *atomic processes*. An atomic process is a triplet $p = (I, O, CE)$ where: I and O are respectively input and output signatures (i.e., sets of typed variables) and $CE = \{(\theta, E)\}$, is a set of conditional effects, with:

- Condition θ is a boolean expression over atoms over accessible terms over some family of constants[1] and the input variables $u_1, \ldots, u_n$ in I,
- A set of effects E where each effect $e \in E$ is a pair (es, ev) with:
  - es, effect on world database, is a set of modifications on the global database (i.e., expressions of the form insert, delete or modify),

[1] The symbol ω is used to denote an *undefined* (or null) value.

- *ev*, effects on output variables, is a set of assignment statements of the forms: $v_j := t$, $\forall v_j \in O$ such that either $t = \omega$ or t is an accessible term over some set of constants and over the input variables $u_1, \ldots, u_n$.

A *message type* has the form $m(p_1, \ldots, p_n)$ where m is the message name and $p_1, \ldots, p_n$ are message parameters. Each parameter p_i is defined over a domain $\mathcal{D}$ (w.l.o.g., we assume that all the messages parameters are defined over the same values domain $\mathcal{D}$). The behavior of a Colombo service is given by the notion of *guarded automaton* as defined below.

Definition 1. *A* ***guarded automaton*** *(GA) of a service S is a tuple $GA(S) = \langle Q, \delta, q_0, F, LStore(S) \rangle$, where :*

- *Q is a finite set of control states with $q_0 \in Q$ the initial state,*
- *$F \subseteq Q$ is a set of final states,*
- *$LStore(S)$ is a finite set of typed variables,*
- *the transition relation δ contains tuples (q, θ, μ, q') where $q, q' \in Q$, θ is a condition over LStore (no access to world instance), and μ has one of the following forms:*
  - *(incoming message) $\mu = ?m(v_1, \ldots, v_n)$ where m is a message having as signature $m(p_1, \ldots, p_n)$, and $v_i \in LStore(S), \forall i \in [1, n]$, or*
  - *(send message) $\mu = !m(b_1, \ldots, b_n)$ where m is a message having as signature $m(p_1, \ldots, p_n)$, and $\forall i \in [1, n]$, each b_i is either a variable of $LStore(S)$ or a constant, or*
  - *(atomic process invocation) $\mu = p(u_1, \ldots, u_n; v_1, \ldots, v_m, CE)$ with p an atomic process having n inputs, m outputs and CE as conditional effects, and $\forall i \in [1, n]$, each u_i (respectively, v_i) is either a variable of $LStore(S)$ or a constant.*

Semantics. We use the notion of an *extended automaton* to define the semantics of a Colombo service. At every point in time, the behavior of an instance of a Colombo service S is determined by its *instantaneous configuration (or simply, configuration).* A configuration of a service is given by a triplet $id = (l, \mathcal{I}, \alpha)$ where l is its current control state, $\mathcal{I}$ a world database instance and α is a valuation over the variables of *LStore*.

An execution of a service S starts at an initial configuration $id_0 = (l_0, \mathcal{I}_0, \alpha_0)$, with l_0 the initial control state of $GA(S)$, $\mathcal{I}_0$ an arbitrary database over $\mathcal{W}$ and $\alpha_0(x) = \omega$, $\forall x \in LStore(S)$. Then, a service makes a move denoted $id_i \xrightarrow{\mu_i} id_j$ according to the mechanics defined by the set of transitions of $GA(S)$. More specifically, given an $id_i = (l_i, \mathcal{I}_i, \alpha_i)$ and a transition $(l_i, \theta, \mu, l_{i+1}) \in \delta$ such that $\alpha_i(\theta) \equiv \mathsf{true}$ then $id_i \xrightarrow{\mu_i} id_j$ where:

- if $\mu = ?m(v_1, \ldots, v_n)$ then only $(v_1, \ldots, v_n)$ receive new values. The others variables and the database do no change.
- if $\mu = !m(b_1, \ldots, b_n)$ then there is no modification on the variables nor the database.
- if $\mu = p(u_1, \ldots, u_n; v_1, \ldots, v_m, CE)$ then

- if there is no $(c, E) \in CE$ where c is verified (or there is more than one) then there is no modification of the variables nor the database.
- let (c, E) be the unique conditional effects in CE s.t c is verified, and let (es, ev) be a non-deterministicall chosen element of E, then :
  - ∗ for each statement *insert* $R(t_1, \ldots, t_k, s_1, \ldots, s_l)$, *delete* $R(t_1, \ldots, t_k)$, or *modify* $R(t_1, \ldots, t_k, s_1, \ldots, s_l)$ in es, apply the corresponding modifications. The obtained instance is the database $\mathcal{I}_{i+1}$.
  - ∗ for all $v_j := t$ in ev, execute the affectations, all the other variables v of $LStore(S)$ do not change.

The semantics of a Colombo service can be captured by the following notion of an extended infinite state machine.

Definition 2. *(extended state machine) Let $GA(S) = \langle Q, \delta, l_0, F, LStore(S) \rangle$ be a guarded automaton of a service S. The associated infinite state machine, noted $E(S)$, is a tuple $E(S) = (\mathbb{Q}, \mathbb{Q}_0, \mathbb{F}, \Delta)$ where:*

- *$\mathbb{Q} = \{(l, \mathcal{I}, \alpha)\}$ with $l \in \mathbb{Q}$, $\mathcal{I}$ a database over $\mathcal{W}$ and α a valuation over the variables of $LStore$. The set $\mathbb{Q}$ contains all the possible configurations of $E(S)$.*
- *$\mathbb{Q}_0 = \{(l_0, \mathcal{I}_0, \alpha_0)\}$, with $\mathcal{I}_0$ an arbitrary database over $\mathcal{W}$ and $\alpha_0(x) = \omega$, $\forall x \in LStore(S)$. $\mathbb{Q}_0$ is the infinite set of initial configurations of $E(S)$.*
- *$\mathbb{F} = \{(l_f, \mathcal{I}, \alpha) \mid l_f \in F\}$. F is the set of final configurations of $E(S)$.*
- *Δ is an (infinite) set of transitions of the form $\tau = (l_i, \mathcal{I}_i, \alpha_i) \xrightarrow{\mu_i} (l_j, \mathcal{I}_j, \alpha_j)$.*

We define now the notion of simulation between two Colombo services.

Definition 3. ***(Simulation)*** *Let S and S' be two Colombo services and let $E(S) = (\mathbb{Q}, \mathbb{Q}_0, \mathbb{F}, \Delta)$ and $E(S') = (\mathbb{Q}', \mathbb{Q}'_0, \mathbb{F}', \Delta')$ be respectively their associated extended state machines.*

- *Let $(id, id') \in \mathbb{Q} \times \mathbb{Q}'$. The configuration $id = (l, \mathcal{I}, \alpha)$ is simulated by $id' = (l', \mathcal{I}', \alpha')$, noted $id \preceq id'$, iff:*
  - *$\mathcal{I} = \mathcal{I}'$, and*
  - *$\forall id \xrightarrow{\mu} id_j \in \Delta$, there exists $id' \xrightarrow{\mu'} id'_l \in \Delta'$ such that $\mu = \mu'$ and $id_j \preceq id'_l$*
- *The extended state machine $E(S)$ is simulated by the extended state machine $E(S')$, noted $E(S) \preceq E(S')$, iff $\forall id_0 \in \mathbb{Q}_0, \exists id'_0 \in \mathbb{Q}'_0$ such that $id_0 \preceq id'_0$*

- *A Colombo service S is simulated by a Colombo service S', noted $S \preceq S'$, iff $E(S) \preceq E(S')$.*

Informally, if $S \preceq S'$, this means that S' is able to faithfully reproduce the external visible behavior of S. The external visible behavior of a service is defined here with respect to the content of the world database as well as the exchanged *concrete* messages (i.e., message name together with the values of the message parameters). The existence of a simulation relation ensures that each execution tree of S is also an execution tree of S' (in fact, a subtree of S'), modulo a relabeling of control states.

3 Main Results

3.1 DB-Less Services ($\boldsymbol{Colombo^{db=\emptyset}}$)

We consider the simulation problem in the class of Colombo services without any access to the database, called DB-less services and denoted $Colombo^{DB=\emptyset}$. Let S be a $Colombo^{db=\emptyset}$ service. The associated state machine is a tuple $E(S) = (\mathbb{Q}, \mathbb{Q}_0, \mathbb{F}, \Delta)$. A configuration of $E(S)$ has the form $id = (l, \emptyset, \alpha)$ while there is only one initial configuration $id_0 = (l_0, \emptyset, \alpha_0)$ with $\alpha_0(x) = \omega$, $\forall x \in LStore(S)$. Moreover, in $Colombo^{db=\emptyset}$ services, atomic processes can only assign constants to variables of $LStore(S)$ or affect value of a variable to another. Note that $E(S)$ is still an infinite state system. This is due to the presence of input messages with parameters taking their values from a possibly infinite domain. Using a symbolization technique, it is possible however to abstract from concrete values and hence turns extended machines associated with $Colombo^{db=\emptyset}$ services into finite state machines. The main idea is to use the notion of *regions* to group together states of E(S). Interestingly, the obtained representation, called $Colombo^{db=\emptyset}$ *region automaton*, is a finite state machine and hence a simulation algorithm can be devised for this case.

Theorem 1. *Given two DB-less Colombo services S and $S^{'}$, checking whether $S \preceq S^{'}$ is* EXPTIME*-complete.*

The detailed proof of this theorem is given in the extended version of this paper [11]. As said before starting from a test of simulation between two DB-less Colombo services S and $S^{'}$, we construct a test of simulation between two corresponding (finite) $Colombo^{db=\emptyset}$ region automaton R^S and $R^{S'}$. The problem is clearly exponential because the numbers of symbolic states in R^S and $R^{S'}$ is exponential in the size of the two services S and S'. The proof of hardness is obtained from a reduction of the problem of the existence of an infinite execution of an alternating Turing machine M working on space polynomially bounded by the size of the input [12] to a simulation test between two DB-less Colombo services.

3.2 Bounded Services ($\boldsymbol{Colombo^{bound}}$)

We consider now the simulation problem in the setting of a Colombo model with a *bounded* global database, denoted $Colombo^{bound}$. A service belonging to $Colombo^{bound}$ is restricted to use during all his executions a global database $\mathcal{W}$ whose size never exceed a given constant k. Given two services S and $S^{'}$, we say that S is *k-bounded simulated* by $S^{'}$ (noted $S \preceq_k S^{'}$) if $S^{'}$ does simulate S if we restrict the attention to executions where the size of the database is at most equal to k.

Theorem 2. *Let S and S' be two Colombo services, then testing $S \preceq_k S'$ is* 2-EXPTIME *complete.*

The proof of decidability of Theorem 2 is achieved by mapping the *k-bound* simulation test $S \preceq_k S^{'}$ into a standard test of simulation between two DB-less Colombo services $\mathcal{M}(S) \preceq \mathcal{M}(S^{'})$. The main idea of the reduction is to encode the bounded databases into a set of variables.

4 Related Works

Data-centric services and artifact-centric business processes attracted a lot of attention from the research community these recent years [8, 9]. Most of these research works focus on the verification problem. In the context of data-centric services, the verification problem is undecidable in the general case. Existing works focus on identification of specific models and restrictions in which the verification problem can be solved. In [13], the author consider the problem of verifying artifact system against specifications expressed in quantified temporal logic. The verification problem is undecidable in the general setting. So, the paper considers a restricted fragment obtained by bounding the number of values stored in a given execution state of the system. The authors use a specific abstraction technique to construct a finite symbolic system which is bisimilar to the original infinite system. By this way, model checking can be carried out over the (finite) symbolic model instead of the original infinite artefact system. The upper bound time complexity of the proposed procedure is doubly exponential. In [14], the authors study the composition problem for data-centric services using an approach based on the simulation relation. Like our approach, they prove the decidability of simulation by bounding the size of database instance, but the model is less expressive than $Colombo^{bound}$.

Independently from the Web service area, the simulation problem between infinite transition systems has been addressed. This problem is undecidable in the general case but there are few classes, e.g., one-counter nets [15], automate with finite memory [2], where the problem is known to be decidable. Close to our work, [16] introduces a new formalism *Variable automaton*, which is an automaton where transitions are labelled with letters from an alphabet or variables. During an execution, the values assigned to variables are fixed and only one special variable can be *refreshed* (reinitialized). In [2], the authors define a variable automaton where variables are *refreshed* at specific states, named *FVA* (Fresh Variable Automata). They prove the decidability of the simulation for this model. Then, they extend *FVA* model with equality guards over variables. The obtained model is called *GVA* (Guarded Variable Automata). The authors show the decidability of the simulation for *GVA* and provide an upper bound (EXPTIME).

5 Conclusion

We studied decidability and complexity issues related to the simulation problem in the framework of the Colombo model. Our results, ranging from EXPTIME to undecidability show that the marriage between data and web service business protocols gives rise to some challenging issues. The decidability and complexity results, EXPTIME-complete for $Colombo^{DB=\emptyset}$ and 2EXPTIME-complete for $Colombo^{bound}$ are far from being straightforward, due to the fact we are dealing with infinite state systems. This paper proposed also a *symbolic* procedure based on the notion of *region automata* to handle the infiniteness of the framework. Finally, as side effect of our work, we derived a tight complexity of the simulation problem for automata over infinite domain, namely GVA. Our future works will be devoted to the definition of a generic framework that enables to capture the main features of data-centric services and which can be used as basis to study the problems underlying formal analysis, verification and synthesis of data-centric services.

References

[1] Berardi, D., Calvanese, D., Giacomo, G.D., Hull, R., Mecella, M.: Automatic composition of transition-based semantic web services with messaging. In: VLDB, pp. 613–624 (2005)

[2] Belkhir, W., Chevalier, Y., Rusinowitch, M.: Guarded variable automata over infinite alphabets. CoRR abs/1304.6297 (2013)

[3] Bultan, T., Fu, X., Hull, R., Su, J.: Conversation specification: A new approach to design and analysis of e-service composition. In: WWW 2003. ACM (2003)

[4] Benatallah, B., Casati, F., Toumani, F.: Web service conversation modeling: A cornerstone for e-business automation. IEEE Internet Computing 08, 46–54 (2004)

[5] Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing web service protocols. DKE 58, 327–357 (2006)

[6] Muscholl, A., Walukiewicz, I.: A lower bound on web services composition. In: Seidl, H. (ed.) FOSSACS 2007. LNCS, vol. 4423, pp. 274–286. Springer, Heidelberg (2007)

[7] Milner, R.: Communication and concurrency. Prentice-Hall, Inc., Upper Saddle River (1989)

[8] Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1152–1163. Springer, Heidelberg (2008)

[9] Calvanese, D., De Giacomo, G., Montali, M.: Foundations of data-aware process analysis: A database theory perspective. In: PODS, pp. 1–12 (2013)

[10] Akroun, L., Benatallah, B., Nourine, L., Toumani, F.: On decidability of simulation in data-centeric business protocols. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 352–363. Springer, Heidelberg (2013)

[11] Akroun, L., Benatallah, B., Nourine, L., Toumani, F.: Decidability and complexity of simulation preorder for data-centric web services (extended version). Technical report (2014), `http://fc.isima.fr/~akroun/fichiers/journal_version_colombo.pdf`

[12] Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. J. ACM 28, 114–133 (1981)

[13] Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of deployed artifact systems via data abstraction. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 142–156. Springer, Heidelberg (2011)

[14] Patrizi, F., Giacomo, G.D.: Composition of services that share an infinite-state blackboard (extended abstract). In: IIWEB (2009)

[15] Abdulla, P.A., Cerans, K.: Simulation is decidable for one-counter nets. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 253–268. Springer, Heidelberg (1998)

[16] Grumberg, O., Kupferman, O., Sheinvald, S.: Variable automata over infinite alphabets. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 561–572. Springer, Heidelberg (2010)