Journal of Logical and Algebraic Methods in Programming 85 (2016) 759–781

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Realizable temporal logics for web service choreography

R. Ramanujam [a], S. Sheerazuddin [b,*]

[a] Institute of Mathematical Sciences, Chennai, India
[b] SSN College of Engineering, Chennai, India

ARTICLE INFO

Article history:
Received 10 April 2015
Received in revised form 12 June 2016
Accepted 13 June 2016
Available online 7 July 2016

Keywords:
Web services
Choreography
Formal methods
Local temporal logics
Communicating automata

ABSTRACT

Web service choreographies specify conditions on observable interactions among the services. An important question in this regard is realizability: given a choreography C, does there exist a set of service implementations I that conform to C? Further, if C is realizable, is there an algorithm to construct implementations in I?

We propose two **local** temporal logics, p-LTL and q-LTL, to specify choreographies. The choreographies written in these logics are realizable **by design**. The semantics of the logics are defined in terms of partial orders, called Lamport diagrams. For specifications in these logics, we give algorithms to construct service implementations as communicating automata. p-LTL choreographies are realized in terms of nondeterministic finite state automata with a coupling relation, whereas q-LTL choreographies are realized in terms of nondeterministic finite state automata with FIFO queues.

1. Introduction

The study of composition of distributed web services has received great attention. When we know what kind of services are available, specifying a sequence of communications between services can well suffice to describe the overall service required. Such a *global* specification of interaction composition has been termed **choreography** [1]. The distributed services can then be synthesized as autonomous agents that interact in conformance with the given choreography. This offers an abstract methodology for the design and development of web services. The choreography and its implementation may be put together in a **choreography model** [2], $M = (C, I)$, where, as already mentioned, C is a specification of the desired global behaviors (a choreography), and I a representation of local services and their local behaviors (an implementation) which collectively should satisfy the specified global behavior. A **choreography modeling language** [2] provides the means to define choreography models, i.e., choreographies, service implementations, and their semantics including a mechanism to compare global behaviors generated by service implementations with a choreography.

1.1. The problem

A principal challenge for such a methodology is that choreographies be **realizable** [3]. Given a choreography C, we say that C is realizable if there exist a set of service implementations I that conform to C. Further, if C is realizable, is there an algorithm to construct implementations in I?

* Corresponding author.
E-mail address: ahmad.sheeraz@gmail.com (S. Sheerazuddin).

http://dx.doi.org/10.1016/j.jlamp.2016.06.002
2352-2208/

What may seem simple global specifications may yet be hard, or even impossible, to implement as a composition of distributed services. The reason is simple: while the global specification requires a communication between service 1 and service 2 to precede that between service 3 and service 4, the latter, lacking knowledge of the former, may well communicate earlier. Thus the composition would admit forbidden behaviors. In general, many seemingly innocuous specifications may be unrealizable. Even checking whether a choreography is realizable may be hard, depending on the expressive power of the formalism in which the choreography is specified.

Closely related, but more manageable, is the problem of **conformance**: check whether a given set of services implement the given choreography specification. Once again, the expressive power of the specification formalism is critical for providing algorithmic solutions to the problem.

The two problems relate to the satisfiability and model checking problems of associated logics. Since the 1980's a rich body of literature has been built in the study of such problems [4].

In the literature, choreographies have been formally specified using automata [5], UML collaboration diagrams [6], interaction Petri nets [7], or process algebra [8]. The service implementations have been modeled variously as Mealy machines, Petri nets or process algebra. Visual formalisms (such as message sequence charts [9]) are naturally attractive and intuitive for choreography specifications but can be imprecise. For instance, it is hard to distinguish between interactions that are permissible and those that must indeed take place. While machine models are precise they might require too much detail.

A natural idea in this context is the use of a logical formalism for choreography specification and that of finite state machines for their implementation. When the formulas of the logic specify global interaction behavior and models for the logic are defined using products of machines, realizability and conformance naturally correspond to the satisfiability and model checking problems for the logic.

Once again, a natural candidate for such a logic is that of temporal logic, linear time or branching time [10,4]. One difficulty with the use of temporal logics for global interaction specifications is that sequentiality is natural in such logics but concurrency poses challenges. It is rather easy to come up with specifications in temporal logics that are not realizable. On the other hand, if we wish to algorithmically decide whether a temporal specification is realizable or not, the problem is often undecidable, and in some cases of high complexity even when decidable. (See [11] and [12] for decidability of the closely related problem of realizability of message sequence graphs.)

One simple way out is to design the temporal logic, limit its expressiveness drastically, so that we can ensure *by diktat* that every formula in it is realizable. This is the line we follow here, initiated by [13] and developed by [14] and [15].

1.2. Related work

In [8], Carbone et al., describe a theory of end-point projection (choreography realizability) via two formal calculi based on session types and a mapping between the two. They describe a global calculus, based on Web Service Choreography Description Language (WSCDL), for choreography specification and an end-point calculus, based on π-calculus, for service implementations. They identify three basic principles for global descriptions so that a sound and complete end-point projection can be defined, thereby ensuring realizability by design.

Another work which ensures realizability by design is explicated in [16]. Here, the authors define process algebra based languages, *Chor* for choreography descriptions and *Role* for service implementations. *Projection* is used to derive service implementations from choreography descriptions in *Chor*. In order to ensure realizability, a dominant role is fixed with each choice structure at the choreography level by the designer. The *Chor* and *Role* languages are extended with *dominated choice* and *dominated loop* structures. A variant of projection is defined that maps the *dominated choice* and *dominated loop* in *Chor* to the corresponding choice and loop structures for the dominant and dominated roles in the *Role* language.

Our work is similar to that of McNeile [17] who extend the process algebra based formalism of Protocol Modeling [18] to define a notion of protocol contract and describe choreographies and participant contracts. They give sufficient conditions for realizability in both synchronous and asynchronous collaborations.

The language-based choreography realizability problem considered in this paper was proposed for conversation protocols in [5] where sufficient conditions for realizability were given. Hallé & Bultan [19] consider the realizability of a particular class of choreographies called arbitrary-initiator protocols for which sufficiency conditions given in [5] fail. The algorithm for choreography realizability works by computing a finite-state model that keeps track of the information about the global state of a conversation protocol that each service can deduce from the messages it sends and receives. Thereafter, the realizability can be checked by searching for disagreements between services' deduced states.

Salaün et al. [20] model choreography as UML collaboration diagrams and check their realizability. They have also implemented a tool which not only checks the realizability of choreography specified using collaboration diagrams but also synthesize the service implementations that realize the choreography [21].

Basu et al. [22] consider the realizability problem for choreographies modeled as conversation protocols [5] (finite automata over send events). They give necessary and sufficient conditions which need to be satisfied by the conversations for them to be realizable. They implement the proposed realizability check and show that it can efficiently determine the realizability of a subclass of contracts [23] and UML collaboration diagrams [6], apart from conversation protocols.

The work presented in [24] checks choreography realizability using the concept of controllability. Given a choreography description, a monitor service is computed from that choreography. The monitor service is used as a centralized orchestrator of the interaction to compute the distributed services. The choreography is said to be realizable if the monitor service is