Science of Computer Programming 109 (2015) 36–60

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Choreographies in the wild [☆]

Massimo Bartoletti [a], Julien Lange [b,*], Alceste Scalas [a], Roberto Zunino [c]

[a] Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari, Italy
[b] Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ, London, UK
[c] Dipartimento di Matematica, Università degli Studi di Trento, Italy

A R T I C L E I N F O

Article history:
Received 17 February 2014
Received in revised form 30 September 2014
Accepted 14 November 2014
Available online 29 November 2014

Keywords:
Contracts
Choreographies
Session types

A B S T R A C T

We investigate the use of choreographies in distributed scenarios where, as in the real world, mutually distrusting (and possibly dishonest) participants may be unfaithful to their expected behaviour. In our model, each participant advertises its promised behaviour as a *contract*. Participants may interact through multiparty sessions, created when their contracts allow to synthesise a *choreography*. We show that systems of *honest* participants (which always adhere to their contracts) enjoy *progress* and *session fidelity*.

1. Introduction

Designing reliable large-scale distributed systems is a difficult challenge. Several intrinsic issues (e.g. concurrency, physical distribution, non-reproducibility of bugs, etc.) make it extremely unlikely that any non-trivial distributed application actually works as its designers intended. To cope with this grand challenge, a lot of foundational and applied research has been carried over in the last 15 years.

One of the most notable approaches is that which studies *session types* [1–3] as an abstraction of the communication behaviour of distributed applications. In the *top-down* approach to session types, the designer of a distributed application specifies its overall communication behaviour through a *choreography*, which validates some global properties of the application (e.g. safety and deadlock-freedom). However, a distributed application is nothing more than the composition of a number services, running on different nodes in the network. In order to globally enjoy the properties validated by the choreography, all the components forming the application have to be verified; this can be done e.g. by projecting the choreography to *end-point types*, against which these services are actually type-checked.

The above-mentioned top-down approach is quite suitable in case the designers have complete control over the application, e.g., when they develop all the needed services. However, in many realistic situations this is not the case: for instance, one might need to integrate with already-existing third-party services, e.g., e-commerce facilities, maps, etc. In this case, the top-down approach might not work as expected, because it could be impossible to adapt the end-point type projected from the choreography to the actual interface of the third-party service. In the last few years, approaches mixing top-down and *bottom-up* composition have been studied to cope with this kind of situation [4–6].

[☆] Work partially supported by Aut. Region of Sardinia under grant L.R.7/2007 CRP-17285 (TRICS) and P.I.A. 2010 project "Social Glue", MIUR PRIN 2010-11 project "Security Horizons", EU COST Action IC1201 (BETTY), and EPSRC EP/K034413/1.

[*] Corresponding author.
E-mail address: j.lange@imperial.ac.uk (J. Lange).

http://dx.doi.org/10.1016/j.scico.2014.11.015
0167-6423/© 2014 Elsevier B.V. All rights reserved.

Dropping the (unrealistic) hypothesis that a distributed application is built of services coming from a single provider has a further consequence. We can no longer suppose that services live in a "civil society" where they smoothly collaborate with each other; rather, they live in a "wild" environment (like Hobbes' *condition of mere nature* [7]) where they must compete for resources, and possibly diverge from the intended behaviour in case they find it convenient to do so. Indeed, this situation is quite reasonable in inter-organisational scenarios, where mistrusted providers compete to achieve conflicting goals. Although some proposals like OASIS [8] try do deal with this issue by prescribing runtime monitoring and logging, no general techniques exist for checking that services actually adhere to their specifications, unless one controls the "bottom layer" of the software stack [9].

To overcome the state of nature, and move towards the "social contract" (as in Hobbes' Leviathan), some recent works have proposed to discipline the interaction of services through *contracts* [10,11]. The idea is a contract-oriented, bottom-up composition, where only those services with *compliant* contracts can establish sessions through which they interact. To give some intuition about the underlying dynamics, let us assume three participants $\mathtt{A}$, $\mathtt{B}$ and $\mathtt{K}$, where the latter plays the role of a *contract broker*. A possible interaction among these participants could be the following:

1. $\mathtt{A}$ advertises its contract $c_{\mathtt{A}}$ to $\mathtt{K}$;
2. $\mathtt{B}$ advertises its contract $c_{\mathtt{B}}$ to $\mathtt{K}$;
3. $\mathtt{K}$ checks if $c_{\mathtt{A}}$ and $c_{\mathtt{B}}$ are compliant; if so, it creates a fresh session s between $\mathtt{A}$ and $\mathtt{B}$;
4. at this point, $\mathtt{A}$ and $\mathtt{B}$ can start interacting through s, by performing the communications prescribed by their contracts. While doing that, they can advertise other contracts, so establishing other sessions (possibly, with different participants).

One key aspect of this contract-oriented approach is that the runtime behaviour of a service may diverge from the advertised one. For instance, participant $\mathtt{B}$ above may fail to perform some of the actions in $c_{\mathtt{B}}$ (either maliciously or accidentally); if $\mathtt{A}$ assumes that $\mathtt{B}$ respects its contract, then $\mathtt{A}$ may get stuck on session s, and consequently it may fail to fulfil its contracts in some other sessions (so possibly damaging itself or other participants). A notable question then arises: how can designers guarantee that their services are *honest*, in that they always respect the advertised contracts, despite the possible dishonesty of the services they may interact with?

In this paper we address the issue of reconciling the top-down and the bottom-up approaches to service composition in wild environments. To this purpose, we propose a combination of two approaches. Like in the contract-oriented approach [10], we assume that applications are built bottom-up, by composing services whose advertised contracts admit agreements. However, unlike in previous contract-oriented calculi, service contracts are *local session types*, in a form similar to those used in [12,13]. A set of contracts admits an agreement whenever it is possible to synthesise from them a choreography — i.e. a global type — whose projections are the contracts themselves [6]. The existence of such a choreography ensures that, if all services behave honestly, the overall application enjoys *progress* and *session fidelity*. However, by the wilderness assumption, the actual behaviour of services may deviate from the promised one. In this case, progress no longer holds, but at least we can single out the services which are responsible for contract violations.

Contributions. We have combined notions and techniques from the session types literature to devise a model of distributed systems where services may be composed bottom-up according to their contracts, and they may deviate from the expected behaviour at runtime. Contracts are expressed as local session types, and multiparty agreements are constructed by adapting the synthesis technique for choreographies of [6]. This allows us to import some results from the session types literature — in particular, the existence of a choreography ensures that contractual agreements enjoy safety and progress (Theorem 3.9). Using choreographies as the basis for agreements is quite flexible: indeed, it allows us to impose further constraints before establishing sessions, e.g. on the number of involved participants, whether or not the session may terminate, etc.

We model services in the CO_2 calculus [14], which is adapted here to use (multiparty) local session types as contracts. We define when a service is *culpable* (i.e., responsible for the next move) in a session, and we show that at least one culpable participant exists in each non-terminated session (Theorem 4.10). If a system gets stuck, it is always possible to identify which participants violated their contracts. We then adapt the notion of *honesty* of [11]. We show that an honest service is always able to satisfy its contracts by firing some of its actions, even when the other services do not cooperate (Theorem 6.1). Furthermore, we show that systems of honest services preserve the properties enjoyed by choreographies: in particular, *session fidelity* (Theorem 6.2) ensures that no extra-contractual interactions are possible in a session, and that honest participants will eventually perform the actions required by their contracts.

We present a case study where the key features of our framework are put into practice. Multiparty sessions, honest participants, and choreography-based contractual agreements are used to model distributed systems where brokers let participants start a new session only if they are advertising contracts such that, as a whole, they implement a class of gossip protocols [15].

Synopsis. The rest of the paper is structured as follows. In Section 2, we sketch an example that we will use through the paper to illustrate our framework. In Section 3, we introduce a contract model based on multiparty session types, and we define contractual agreement as choreography synthesis. In Section 4 we present CO_2, suitably adapted to deal with the contracts of Section 3, and we highlight some of its main features. In Section 5, we formalise the notion of honesty. Our main technical results are presented in Section 6, where we show how global properties of choreographies are projected