Journal of Logical and Algebraic Methods in Programming 90 (2017) 2–30

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Reversibility in session-based concurrency: A fresh look

Claudio Antares Mezzina [a,*], Jorge A. Pérez [b,*]

[a] IMT, School for Advanced Studies Lucca, Italy
[b] University of Groningen & CWI, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 25 July 2016
Received in revised form 2 March 2017
Accepted 15 March 2017
Available online 21 March 2017

Keywords:
Concurrency
Reversible computation
Behavioral types
Process calculi

ABSTRACT

Much research has studied foundations for correct and reliable *communication-centric software systems*. A salient approach to correctness uses verification based on *session types* to enforce structured communications; a recent approach to reliability uses *reversible actions* as a way of reacting to unanticipated events or failures. In this paper, we develop a simple observation: the semantic machinery required to define asynchronous (queue-based), monitored communications can also support reversible protocols. We propose a framework of session communication in which monitors support reversibility of (untyped) processes. Main novelty in our approach are *session types with present and past*, which allow us to streamline the semantics of reversible actions. We prove that reversibility in our framework is causally consistent, and define ways of using monitors to control reversible actions.

1. Introduction

Much research has studied foundations for reliable *communication-centric* software systems, cf. [1–4]. Our interest is in programming models that support the analysis of message-passing programs building on foundations offered by core calculi for concurrency. While early such models focused on (static) verification of protocol correctness, as enforced by properties such as safety, fidelity, and progress (deadlock-freedom), extensions of the basic models with external mechanisms have been proposed to enforce protocol correctness even in the presence of unanticipated events, such as failures or new requirements. Such mechanisms include, e.g., exceptions, interruptions and compensations [5–7], adaptation [8], and monitors [9]. They also include *reversible semantics* [10–12], the main topic of this paper.

Comprehensive approaches to correctness and reliability, which enforce both kinds of requirements, seem indispensable in the principled design of communication-centric software systems. As these systems are typically built using heterogeneous services whose provenance/correctness cannot always be certified in advance, static validation techniques (such as type systems) fall short. Correctness must then be guaranteed by mechanisms that (dynamically) inspect the (visible) behavior of interacting services and take action if they deviate from prescribed communication protocols.

In this work, we aim at uniform approaches to correct and reliable communicating systems. We address the interplay between concurrent models of reversible computation [13,14] and *session-based concurrency* [1]. In reversible models of concurrency the usual *forward* semantics is coupled with a *backward* semantics that enable to "undo" process actions. In this setting, a central correctness criterion is *causal consistency*, which ensures that a computational step is reversed only when all its causes (if any) have already been reversed. In this way, causally consistent reversibility leads to states

* Corresponding authors.
E-mail addresses: claudio.mezzina@imtlucca.it (C.A. Mezzina), j.a.perez@rug.nl (J.A. Pérez).

http://dx.doi.org/10.1016/j.jlamp.2017.03.003
2352-2208/© 2017 Elsevier Inc. All rights reserved.

of the system that could have been reached by performing forward steps only. In session-based concurrency, concurrent interactions between processes can be conceptually divided in two phases: first, processes requesting/offering protocols seek a compatible partner; subsequently, the (compatible) partners establish a session and interact following the stipulated protocols. Session protocols are *resource-aware*: the first phase defines non-deterministic interactions along unrestricted names; the second one uses deterministic interaction sequences along linear names.

Following the seminal work of Danos and Krivine [13], a key technical device in formalizing reversible semantics are *memories*: these are run-time constructs that make it possible to revert actions. Memories are the bulk of a reversible model; their definition and maintenance requires care, as demonstrated by Tiezzi and Yoshida [10,12], who were the first to adapt known approaches to reversible semantics [13,15] into session-based concurrency. Using different kinds of memories (recording events for actions, choices, and forking), their work shows that the standard (untyped) reduction semantics for the session π-calculus satisfies causal consistency.

While insightful, the route to reversibility in session-based concurrency taken in [12] is somewhat unsatisfactory, for session types do not play any role in the underlying (reversible) semantics nor in the proof of causal consistency. If one considers that session types offer a compact abstraction of the communication behavior of the channels/names in a process, then it is natural to think of them as auxiliary mechanisms in the definition of forward and backward reduction semantics. That is, the communication structures given by session types already contain valuable information for enabling causally consistent reversible semantics. If one further considers that once a session is established processes behave deterministically, as dictated by their session protocols, then it is natural to expect that reversibility and causal consistency in session-based concurrency arise more orderly than in untyped models of concurrency, such as those in, e.g., [13,14].

Following these considerations, in this paper we investigate to what extent session types can streamline the definition of reversible, causally consistent semantics for interacting processes. Our main discovery is that external mechanisms typically used to support asynchronous (queue-based) and monitored semantics in session-based concurrency (cf. [16–20]) can also effectively support the definition of reversible sessions. In such semantics, monitors are run-time devices that register the current state of the session protocols implemented in and executed by a process. We explore a fresh approach to reversibility by using *monitors as memories*. The key idea is simple: we exploit the type information in monitors to define the reversible semantics of session processes. Since these types enable and guide process behavior, we may uniformly define forward and backward reductions by carefully controlling such types and their associated run-time information.

Contributions. The main contributions of this paper are the following:

- We define a fresh approach to reversible semantics in session-based concurrency by exploiting monitors as uniform memories that enable and support backward communication steps.
- We show that the reversible semantics in our approach is *causally consistent*, directly exploiting the disciplined interaction scenario naturally induced by session-based concurrency.
- We show that our approach can be extended to enforce *controlled reversibility* by using enriched session types (rather than explicit process constructs) for guiding process behavior.

To highlight the merits of our approach, we rely on a core process model without recursion nor asynchrony, which are important in modeling but largely orthogonal to our reversible semantics. These and other features can be accommodated in our approach while retaining its essence.

In our view, the use of monitors for defining reversible semantics has at least two significant implications. *First*, it is encouraging to discover that monitor-based semantics—introduced in [16–18] for asynchronous communications with events and used in [19,20] to define run-time adaptation—may also inform the semantics of reversible protocols. Monitors have also been used for enforcing security properties (such as information flow [21,22]) and for assigning blame to deviant session processes [9]. Therefore, monitor-based semantics encompass an array of seemingly distinct concerns in structured communications. *Second*, we see our work as a first step towards validation techniques for communication and reversibility based on run-time verification. Session frameworks with run-time verification have been developed in, e.g., [23,6]. As these works do not support reversibility, our work may help to enhance their dynamic verification techniques. Indeed, since the framework in [23,6] introduces constructs for delimiting interruptible sub-protocols, one could re-use such constructs (and their underlying mechanisms, such as type memories) to safely enable reversible actions within distributed protocols.

Outline. This paper is structured as follows. In the following section we motivate further the key ideas of our development. Section 3 presents the syntax and operational semantics of our process model with sessions and reversibility, and illustrates it via a running example. The main property of our model, *causal consistency* (Theorem 4.1), is established in Section 4. Section 5 discusses an extension of our framework to enforce controlled reversible actions. Other extensions and enhancements (including asynchrony, delegation, and recursion) are discussed in Section 6. Section 7 elaborates on related works. Section 8 closes the paper by collecting some concluding remarks and highlighting some directions for future work. The appendix (Appendix A) collects omitted proofs.

This paper is a revised and substantially extended version of the workshop paper [24] and the short communication [25]: here we offer full technical details, new examples, and an extended account of related works. In particular, Section 3 has been streamlined and extended to handle reversible labeled choices, not supported in [24]. Moreover, the content of Sections 4, 5, and 6 is new to this paper.