Science of Computer Programming 168 (2018) 94–117

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Full-abstraction for client testing preorders

Giovanni Bernardi [a,*], Adrian Francalanza [b]

[a] Université Paris-Diderot/IRIF, Paris, France
[b] University of Malta, Msida, Malta

ARTICLE INFO

Article history:
Received 30 October 2017
Received in revised form 9 July 2018
Accepted 20 August 2018
Available online 30 August 2018

Keywords:
Behavioural equivalences
Full-abstraction
Foundations of web-services

ABSTRACT

Client testing preorders relate tests (clients) instead of processes (servers), and are usually defined using either must testing or a compliance relation. Existing characterisations of these preorders are unsatisfactory for they rely on the notion of *usable* clients which, in turn, are defined using an existential quantification over the servers that ensure client satisfaction. In this paper we characterise the set of usable clients wrt must testing for finite-branching LTSs, and give a sound and complete decision procedure for it. We also provide novel coinductive characterisations of the client preorders due to must and compliance, which we use to show that these preorders are decidable, thus positively answering the question opened in [5,3].

1. Introduction

The testing theory of De Nicola–Hennessy [14,19] is a well-known approach to define preorders and equivalences for communicating processes. In this theory a process p_2 is considered as good as another process p_1 if *every* test r passed by p_1 is also passed by p_2, and two processes are equivalent if they pass the same tests. The standard notion of passing a test is formalised predominantly by the so called must testing relation: p must r whenever every run of the system $r \,||\, p$ leads the test r to a successful state. Concretely, the formal definition of the well-known must preorder is thus

$$p_1 \sqsubseteq p_2 \text{ iff } \forall r \,.\, (p_1 \text{ must } r) \text{ implies } (p_2 \text{ must } r) \tag{1}$$

During the last decade, testing theory has been adapted and enriched to lay the theoretical foundations for web-services, where processes are seen as servers, and tests as clients (or peers). *Adapted* in that an alternative relation to must has been proposed, which fits better the setting of web-services and client/server satisfaction. This novel relation, called *compliance*, states that a client r complies with a server p, denoted r cmp p, if whenever a computation of $r \,||\, p$ cannot go on or p diverges, the client is in a successful state [10,29]. *Enriched* in that in addition to the classical preorder for servers,[1] also preorders for clients and peers have been investigated [2,5]. Preorders for clients have a natural definition similar to (1), for example the compliance client preorder is defined by letting

$$r_1 \sqsubseteq_{\text{cmp}} r_2 \text{ if } \forall p \,.\, (r_1 \text{ cmp } p) \text{ implies } (r_2 \text{ cmp } p) \tag{2}$$

* Corresponding author.
E-mail addresses: gio@irif.fr (G. Bernardi), adrian.francalanza@um.edu.mt (A. Francalanza).

[1] *Processes* according to the classic terminology.

https://doi.org/10.1016/j.scico.2018.08.004
0167-6423/

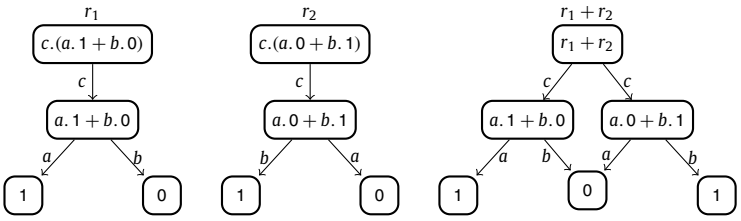

Fig. 1. LTS depictions of the behaviours described in Eq. (3).

that is a client r_2 is as good as a client r_1 whenever *every* server p that satisfies r_1 also satisfies r_2. In this paper we dwell on the client preorders due to the must relation [5] and the compliance relation [3].

Definitions such as (1) and (2) are intuitive and easy to understand, but they are hard to use in practice for they are *contextual*: they contain a universal quantification over contexts, and hence give no effective proof method to determine pairs in the preorders being defined. To overcome this problem, testing preorders are usually presented together with *alternative characterisations* that avoid universal quantification over contexts, and that are amenable to the development of proof methods and decision procedures.

In [5,3] the authors develop such characterisations for the client preorders due to must and compliance, however neither of these characterisations are fully-abstract, nor are decision procedures for the preorders discussed. In particular, the alternative preorders given in [5, Definition 3.10] and [3, Definition 5.2.16] are not fully-abstract for they are defined modulo *usable* clients, *i.e.,* clients that are satisfied by *at least one* server. In other words, the definitions of these preorders rely explicitly on the interactions that clients may have with servers.

Usability is a pivotal notion that appears frequently in the literature on process calculi as foundations for web-service: it has been called viability in [22,30] and controllability in [9,28], and has already been studied in various settings [22,7,5, 30]. While usable clients wrt the compliance relation have been characterised in [29], the situation remains unclear with respect to the must testing. The characterisation of usable clients is indeed problematic, for solving it requires finding the conditions under which one can either (a) construct a server p that satisfies a given client, or (b) show that every p does *not* satisfy a given client. Whereas proving (b) is complicated by the universal quantification over *all* servers, the proof of (a) is complicated by the non-deterministic behaviour of clients. In particular, determining usability using approach (a) is complicated because client usability is *not* compositional. For instance consider the following two clients, whose behaviour is depicted in Fig. 1:

$$r_1 = c.(a.1 + b.0) \qquad \text{and} \qquad r_2 = c.(a.0 + b.1) \tag{3}$$

where 1 denotes satisfaction (success). Both clients are usable, since r_1 is satisfied by the server $\overline{c}.\overline{a}.0$, and r_2 is satisfied by the server $\overline{c}.\overline{b}.0$. However, their composition $r_1 + r_2$ is *not* a usable client, *i.e.,* $p \neg\text{must}\ r_1 + r_2$ for every p; intuitively, this is because r_1 and r_2 impose opposite constraints on the processes that pass one or the other (*e.g.,* $\overline{c}.(\overline{a}.0 + \overline{b}.0)$ does not satisfy $r_1 + r_2$). A compositional analysis is even more unwieldy for recursive tests. For instance, the recursive client $\mu x.\big(c.(a.1 + b.x) + c.(a.0 + b.1)\big)$ is *not* usable wrt must because of the non-determinism analogous to $r_1 + r_2$, and the unsuccessful computations along the infinite trace $(c.b)^*$; this argument works because infinite unsuccessful computations are catastrophic in must testing settings.

This paper presents a sound and complete characterisation for usable clients wrt must within a finite-branching LTS. Through the results of [5] — in particular, the equivalence of usability for clients and peers stated on [5, pag. 11] — our characterisation directly yields a fully-abstract characterisation for the must preorder for clients and peers. These characterisations, though, are still hard to use in practice when reasoning on recursive clients. Spurred by this observation, we define a new *coinductive* and fully-abstract characterisation for the client preorders due to must and compliance, which we find easier to use than the ones of [5,3]. These coinductive characterisation are informed by our study on usability, and differs subtly from the coinductive characterisations of preorders for servers given in [22,29,6]. Finally, our inductive definition for usable clients also provides insights into the must client preorder of [5]: we show that limiting contexts to servers offering only *finite* interactions preserves the discriminating power of the original preorder. The contributions of this paper are thus:

- a fully-abstract characterisation of usable clients wrt must (Theorem 1);
- two coinductive, fully-abstract characterisations of the client preorders due to must (Theorem 2) and compliance (Theorem 5);
- a proof that non-recursive contexts are sufficient to define the client preorder due to must (Theorem 3);
- decidability results for usable clients and the client preorder due to must (Theorem 4).

We hope that this work has an impact outside of testing theory and foundations for web-services, in the following sense. Our original motivation to study usability of clients was to arrive at decision procedures for client preorders. However, it turns out that our study of usability is relevant to controllability issues in service-oriented and monitor-oriented architectures [25,34,16]. For instance, the symbolic characterisation for consistently-detecting monitors in [16] called controllability