Information Processing Letters 115 (2015) 212–220

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A ground-complete axiomatization of stateless bisimilarity over Linda ☆

Luca Aceto [a,b,*], Eugen-Ioan Goriac [c], Anna Ingolfsdottir [a]

[a] ICE-TCS, School of Computer Science, Reykjavik University, Menntavegur 1, IS 101 Reykjavik, Iceland
[b] Gran Sasso Science Institute, INFN, Viale F. Crispi 7, 67100 L'Aquila, Italy
[c] Icelandic Heart Association, Holtasmári 1, IS 201 Kópavogur, Iceland

ARTICLE INFO

Article history:
Received 22 June 2013
Received in revised form 16 May 2014
Accepted 15 September 2014
Available online 22 September 2014
Communicated by J.L. Fiadeiro

Keywords:
Concurrency
Process algebra
Stateless bisimilarity
Linda
Equational logic

ABSTRACT

This paper offers a finite, ground-complete axiomatization of stateless bisimilarity over the tuple-space-based coordination language Linda. As stepping stones towards that result, axiomatizations of stateless bisimilarity over the sequential fragment of Linda without the *nask* primitive, and over the full sequential sub-language are given. It is also shown that stateless bisimilarity coincides with standard bisimilarity over the sequential fragment of Linda without the *nask* primitive.

1. Introduction

The goal of this paper is to contribute to the study of equational axiomatizations of behavioural equivalences for processes with data—see, e.g., the references [14,18–20] for earlier contributions to this line of research. Specifically, we present a ground-complete axiomatization of stateless bisimilarity from [8,12,17,24] over the well-known, tuple-space-based coordination language Linda [11,15].

Linda is a, by now classic, example from a family of coordination languages that focus on the explicit control of interactions between parallel processes. (More modern coordination languages are Reo [2] and BIPL [5,7].) A communication between Linda processes takes place by accessing tuples in a shared memory, called the *tuple space*, which is a multiset of tuples. The communication mechanism in Linda is asynchronous, in that send operations are non-blocking. Our presentation of the syntax and the semantics of Linda follows those given in [9,24].

In the light of its intuitive appeal and impact, Linda has received a fair amount of attention within the concurrency theory community. For instance, the relative expressive power of fragments of Linda has been studied in [9] and the paper [14] studies testing semantics, in the sense of De Nicola and Hennessy [13], over applicative and imperative process algebras that are inspired by Linda. The paper [13] also provides complete inequational axiomatizations of the studied calculi with respect to testing semantics.

Testing semantics can be viewed as the most natural notion of behavioural equivalence for a language from the programmer's perspective. Indeed, it is the formalization of the motto that 'two program fragments should be considered equivalent unless there is a context/test that tells

☆ The authors have been partially supported by the projects 'Meta-theory of Algebraic Process Theories' (nr. 100014021) and 'Nominal Structural Operational Semantics' (nr. 141558-051) of the Icelandic Research Fund. Eugen-Ioan Goriac was also funded by the project 'Extending and Axiomatizing Structural Operational Semantics: Theory and Tools' (nr. 1102940061) of the Icelandic Research Fund.

* Corresponding author at: ICE-TCS, School of Computer Science, Reykjavik University, Menntavegur 1, IS 101 Reykjavik, Iceland.
E-mail addresses: luca@ru.is (L. Aceto), eugen.goriac@me.com (E.-I. Goriac), annai@ru.is (A. Ingolfsdottir).

http://dx.doi.org/10.1016/j.ipl.2014.09.014
0020-0190/

them apart.' Testing semantics is, however, not very robust. In particular, if one extends a language with new features that increase the observational power of tests, the resulting notion of 'testing equivalence' for the extended language will be finer than the one for the original language. This means that the results one had worked hard to establish for the original language will have to be established anew.

Stateless bisimilarity [8,12,17,24] is a variation on the classic notion of bisimilarity [21,25] that is suitable for reasoning compositionally about open, concurrent and state-bearing systems. It is the finest notion of bisimilarity for state-bearing processes that one can find in the literature and comes equipped with a congruence rule format for operational rules [24]. It is therefore interesting to study its equational theory in the setting of a seminal language like Linda, not least because equational axiomatizations of stateless bisimilarity may form the core of axiom systems for coarser notions of equivalence over that language.

The main contribution of this paper is a ground-completeness result for stateless bisimilarity over Linda given in Section 3. We first present a complete axiom system for stateless bisimilarity over the sequential fragment of Linda without the *nask* primitive, which tests for the *absence* of a tuple in the tuple space (Theorem 3.2). Interestingly, it turns out that stateless bisimilarity over this fragment of Linda has the same axiomatization of standard bisimilarity, when the considered sub-language of Linda is viewed as Basic Process Algebra (BPA) with deadlock and the empty process [26]. We formalize the connection between the two languages and their respective semantics, culminating in Theorem 3.4.

Next we offer a ground-complete axiomatization of stateless bisimilarity over the full sequential fragment of Linda (Section 3.1). In this setting, we have to deal with the subtle interplay of *ask* and *nask* primitives, which test for the presence and absence of some tuple in the tuple space, respectively. In Theorem 3.5, we show that two equation schemas are enough to capture equationally the effect that combinations of *ask* and *nask* primitives may have on the behaviour of Linda terms.

Following rather standard lines, we give a ground-complete axiomatization of stateless bisimilarity over the full Linda language we consider in this paper in Section 3.2.

We end the paper by comparing our work with that presented in [10] (Section 4), as well as with some concluding remarks and suggestions for future research (Section 5).

2. Preliminaries

In this section we present the syntax and operational semantics for the classic, tuple-space-based coordination language Linda [11,15]. (Our presentation follows those given in [9,24].) Moreover, we introduce the notion of stateless bisimilarity and the basic definitions from equational logic used in this paper.

Linda's signature Σ_D for data (the so-called tuple space) consists of the constant $\emptyset$ for the empty tuple space, a (possibly infinite) set $\mathcal{U}$ of constants standing for memory tuples and a binary separator $\_\ \_$ that is associative and commutative, but not idempotent, and has $\emptyset$ as left and right unit. (The store is a multiset of tuples.) The set $T(\Sigma_D)$ of closed *data terms* is given, therefore, by the following BNF grammar:

$$d ::= \emptyset \mid u \mid d\ d,$$

where $u \in \mathcal{U}$. Each data term d determines a multiset $\{u_1, \ldots, u_k\}$ of tuples in the obvious way. In what follows, we write $u \in d$ when there is at least one occurrence of the tuple u in the multiset denoted by d.

Following [9], the signature Σ_P for Linda is implicitly given by the BNF grammar defining the set $\mathbb{T}(\Sigma_P)$ of open *process terms* over a countably infinite set V_P of *process variables*:

$$t ::= x \mid \delta \mid \epsilon \mid ask(u) \mid nask(u) \mid tell(u) \mid get(u) \mid t + t \mid t; t \mid t \parallel t,$$

where $x \in V_P$ and $u \in \mathcal{U}$. Closed terms are terms without occurrences of variables. The set of closed process terms is denoted by $T(\Sigma_P)$. A substitution σ is a function of type $V_P \to \mathbb{T}(\Sigma_P)$. A closed substitution is a substitution whose range is included in $T(\Sigma_P)$. We write $\sigma(t)$ for the term resulting by replacing each occurrence of a variable x in t with the term $\sigma(x)$. Note that $\sigma(t)$ is a closed term whenever σ is a closed substitution.

Intuitively, δ is a constant process that symbolizes deadlock, which satisfies no predicates and performs no actions. The constant ϵ denotes a process that satisfies the successful termination predicate, denoted by $\downarrow$ in what follows, and performs no action. The constants *ask*, *nask*, *tell*, and *get* are the basic Linda instructions for operating with the data component. $ask(u)$ and $nask(u)$ check whether tuple u is and, respectively, is not in the store. $tell(u)$ adds tuple u to the store, while $get(u)$ removes one of its occurrences if it is present. The $ask(u)$, $get(u)$ and $nask(u)$ operations are blocking, in the sense that a process executing them blocks if u is not in the tuple space for *ask* and *get*, and if it is in the tuple space for *nask*. The operations $\_ + \_$, $\_ ; \_$ and $\_ \parallel \_$ are, respectively, the standard alternative, sequential and interleaving parallel composition operations familiar from process algebras—see, for instance, [3].

Definition 2.1 *(Transition System Specification for Linda).* The operational semantics of Linda is given in terms of a unary immediate termination predicate $\downarrow$ and a binary transition relation $\to$ over configurations of the form (p, d), with $p \in T(\Sigma_P)$ and $d \in T(\Sigma_D)$. Intuitively, $(p, d) \downarrow$ means that the process term p can terminate immediately in the context of the tuple space d, whereas

$$(p, d) \to (p', d')$$

indicates that the configuration (p, d) can evolve into (p', d') in one computational step. Formally, $\downarrow$ and $\to$ are the least relations over configurations satisfying the following set of rules.

$$\frac{}{(\epsilon, d) \downarrow} \qquad \frac{(x, d) \downarrow}{(x + y, d) \downarrow} \qquad \frac{(y, d) \downarrow}{(x + y, d) \downarrow}$$