Information and Computation 246 (2016) 62–76

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Proving language inclusion and equivalence by coinduction

Jurriaan Rot [d,*,1], Marcello Bonsangue [a,b], Jan Rutten [b,c]

[a] LIACS, Leiden University, Niels Bohrweg 1, Leiden, Netherlands
[b] Centrum Wiskunde en Informatica, Science Park 123, Amsterdam, Netherlands
[c] ICIS, Radboud University Nijmegen, Heyendaalseweg 135, Nijmegen, Netherlands
[d] Université de Lyon, CNRS, ENS de Lyon, UCBL, LIP laboratory, 46 Allée d'Italie, Lyon, France

ARTICLE INFO

Article history:
Received 20 July 2013
Available online 2 December 2015

Keywords:
Language equality
Bisimulation
Coinduction
Bisimulation-up-to
Deterministic automata
Simulation

ABSTRACT

Language equivalence and inclusion can be checked coinductively by establishing a (bi)simulation on suitable deterministic automata. In this paper we present an enhancement of this technique called *(bi)simulation-up-to*. We give general conditions on language operations for which bisimulation-up-to is sound. These results are illustrated by a large number of examples, giving new proofs of classical results such as Arden's rule, and involving the regular operations of union, concatenation and Kleene star as well as language equations with complement and intersection, and shuffle (closure).

1. Introduction

The set of all languages over a given alphabet can be turned into an (infinite) deterministic automaton. By the *coinduction* principle, any two languages that are *bisimilar* as states in this automaton are equal. The typical way to show that two languages x and y are bisimilar is by exhibiting a *bisimulation*, a relation on languages satisfying certain properties, which contains the pair (x, y). Indeed, this is the basis of a practical coinductive proof method for language equality [30], which has, for example, been applied in effective procedures for checking equivalence of regular languages [8,19,21,30].

In this paper we present *bisimulation up to congruence*, in the context of languages and automata. This is an enhancement of bisimulation originally stemming from process theory [27,33]. In order to prove bisimilarity of two languages, instead of showing that they are related by a bisimulation, one can show that they are related by a *bisimulation-up-to*, which in many cases yields smaller, easier and more elegant proofs. As such, we introduce a proof method which improves on the more classical coinductive approach based on bisimulations.

Bisimulation-up-to(-congruence) techniques essentially make use of the underlying (algebraic) structure induced on (the automaton of) languages by the operations and expressions under consideration. In this paper we will first focus on languages presented by the regular operations of union, concatenation and Kleene star. We will exemplify our coinductive proof method based on bisimulation-up-to by novel proofs of several classical results such as Arden's rule and the soundness of the axioms of Kleene algebra. This introduces the main ingredients, and the practical use of the proof technique.

* Corresponding author at: ENS de Lyon, 46 Allée d'Italie, Lyon, France.
E-mail addresses: jurriaan.rot@ens-lyon.fr (J. Rot), marcello@liacs.nl (M. Bonsangue), jan.rutten@cwi.nl (J. Rutten).

[1] The research of this author was carried out at Leiden University and CWI, and was funded by the Netherlands Organisation for Scientific Research (NWO), CoRE project, dossier number: 612.063.920. The first author is supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

http://dx.doi.org/10.1016/j.ic.2015.11.009
0890-5401/

Our aim however is to deal with a wide variety of operations on languages. To this end we introduce a general format of *behavioural differential equations*, based on a similar format introduced for streams [20,32] (called *stream GSOS* in [14]). In the current paper we prove that for any operator defined in this format, the associated bisimulation-up-to techniques are sound. We then apply these results by considering language equations involving intersection and complement, and show the usefulness and versatility of the techniques by giving a full coinductive proof of the fact that two particular context-free languages defined in terms of language equations, that of palindromes and that of non-palindromes, indeed form each other's complement. Moreover we give a number of example proofs for the operations of shuffle and shuffle closure.

While bisimilarity can be used to prove equality, the notion of *similarity* can be used to prove language *inclusion*. We will introduce *simulation-up-to* techniques, and show that these are sound whenever the operations under consideration adhere to the above format of behavioural differential equations, and additionally satisfy a monotonicity condition. While language inclusion $x \subseteq y$ can of course be reduced to equality $x + y = y$, it turns out that in practical cases it can be much more convenient and efficient to use simulation-up-to directly, instead of using this reduction and prove equality by bisimulation-up-to.

Behavioural differential equations have been studied extensively, mostly in the context of streams [14,20,32]. The format for language operations which we introduce in this paper, is an extension of the format for stream operations introduced in [14,20]. In [20] it is shown that the operations which can be given by behavioural differential equations are precisely the *causal* functions. For an operation f on streams, f is causal if for any $n > 0$: the first n elements of $f(\sigma_1, \ldots, \sigma_n)$ depend only on the first n elements of each argument $\sigma_1, \ldots, \sigma_n$. This notion has a natural counterpart for languages, and we show that indeed the correspondence holds in our case as well. This gives an additional *semantic* characterization of a large class of operations for which bisimulation-up-to techniques are sound. A similar result is in [9], where causality is taken as a sufficient condition for soundness of up-to techniques for streams.

The main contribution of this paper is the presentation of the coinductive proof technique of (bi)simulation-up-to in the context of languages and automata, together with a large number of examples, providing an accessible explanation of these techniques requiring little background knowledge from the reader. An earlier version of this work appeared as a conference paper [29]. With respect to [29] we have the following new contributions. First, the present paper is entirely self-contained, in contrast to [29] which relies on the abstract theory of coalgebraic bisimulation-up-to [28]. Second, we add here a number of new examples, for other operations such as shuffle. Third, we introduce the notion of *simulation-up-to*. Finally the result stating that operations adhering to the format of behavioural differential equations are precisely the causal functions, is new in the context of languages.

The outline of this paper is as follows. In Section 2 we recall the notions of bisimulation and coinduction in the context of languages and automata. Then in Section 3 we present bisimulation-up-to for the regular operations. In Section 4 we discuss bisimulation-up-to for other operations, by introducing a format for which bisimulation-up-to is guaranteed to be sound and providing a number of examples. In Section 5 we introduce simulation-up-to techniques for language inclusion. In Section 6 we prove the correspondence between behavioural differential equations and causal functions. In Section 7 we place our work in the context of coalgebraic theory and discuss related work, and finally in Section 8 we conclude.

2. Languages, automata, bisimulations and coinduction

Throughout this paper we assume a fixed alphabet A, which is simply a (possibly infinite) set. We denote by A^* the set of *words*, i.e., finite concatenations of elements of A; we denote the empty word by ε, and the concatenation of two words w and v by wv. The set of *languages* over A is given by $\mathcal{P}(A^*)$, and ranged over by x, y, z. We denote the empty language by 0 and the language $\{\varepsilon\}$ by 1. Moreover when no confusion is likely to arise, we write a to denote the language $\{a\}$, for alphabet letters $a \in A$.

A *(deterministic) automaton* over A is a triple (S, o, δ) where S is a set of states, $o\colon S \to \{0, 1\}$ is an output function, and $\delta\colon S \times A \to S$ is a transition function. Notice that S is not necessarily finite, and there is no initial state. We say a state $s \in S$ is *final* or *accepting* if $o(s) = 1$. For each automaton (S, o, δ) there is a function $l\colon S \to \mathcal{P}(A^*)$ which assigns to each state $s \in S$ a language, inductively defined as follows:

$$\varepsilon \in l(s) \text{ iff } o(s) = 1 \qquad\qquad aw \in l(s) \text{ iff } w \in l(\delta(s, a))$$

The classical definition of *bisimulation* [22,26] concerns labelled transition systems, which, in contrast to deterministic automata, do not feature output and may have a non-deterministic branching behaviour.[2] We will base ourselves on a different notion of bisimulation specific to deterministic automata, which is an instantiation of the general *coalgebraic* definition of bisimulation (see Section 7).

Definition 2.1. Let (S, o, δ) be a deterministic automaton. A *bisimulation* is a relation $R \subseteq S \times S$ such that for any $(s, t) \in R$:

1. $o(s) = o(t)$, and
2. for all $a \in A$: $(\delta(s, a), \delta(t, a)) \in R$.

[2] In fact, the standard reference [26] introduces bisimulations for automata rather than transition systems, and Theorem 2.2 appears already there.