Information and Computation 207 (2009) 237–257

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Bialgebraic methods and modal logic in structural operational semantics

Bartek Klin [1]

Computer Laboratory, University of Cambridge, William Gates Building, 15 JJ Thomson Avenue, Cambridge CB3 OFD, UK

ARTICLE INFO

Article history:
Received 15 February 2007
Accepted11 October 2007
Available online 7 November 2008

Keywords:
Structural operational semantics
Coalgebra
Bialgebra
Modal logic
Congruence format

ABSTRACT

Bialgebraic semantics, invented a decade ago by Turi and Plotkin, is an approach to formal reasoning about well-behaved structural operational semantics (SOS). An extension of algebraic and coalgebraic methods, it abstracts from concrete notions of syntax and system behaviour, thus treating various kinds of operational descriptions in a uniform fashion.
In this paper, bialgebraic semantics is combined with a coalgebraic approach to modal logic in a novel, general approach to proving the compositionality of process equivalences for languages defined by structural operational semantics. To prove compositionality, one provides a notion of behaviour for logical formulas, and defines an SOS-like specification of modal operators which reflects the original SOS specification of the language. This approach can be used to define SOS congruence formats as well as to prove compositionality for specific languages and equivalences.

1. Introduction

Structural operational semantics (SOS) [51,1] is one of the most successful frameworks for the formal description of programming languages and process calculi. There, the behaviour of programs or processes is described by means of transition relations, also called *labeled transition systems* (LTSs), induced by inference rules following the syntactic structure of processes. For example, the rules:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x}||\mathtt{y} \xrightarrow{a} \mathtt{x}'||\mathtt{y}} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x}||\mathtt{y} \xrightarrow{a} \mathtt{x}||\mathtt{y}'} \tag{1}$$

define the behaviour of a binary parallel composition operator $||$ without communication. In particular, the rule on the left says that if a process can do a transition labelled with *a*, then the same process put in parallel with any other process can do a similar transition. One could also enrich states and/or transitions in SOS specifications with environments, stores, probabilities, time durations etc., to induce other, more sophisticated kinds of transition systems. The intuitive appeal of SOS and, importantly, its inherent support for modeling nondeterministic behaviour, makes it a natural framework for the formal description of process algebras (see [8] for many examples).

For reasoning about processes a suitable notion of *process equivalence* is needed. Various equivalences on LTSs have been proposed (see [22] for a survey). Bisimilarity is the most widely studied, but other equivalences such as trace equivalence or testing equivalence have also been considered. Several equivalences have also been proposed for probabilistic, timed and other kinds of transition systems, including their respective notions of bisimilarity.

To support inductive reasoning, it is important for the chosen process equivalence to be *compositional*; indeed, it is useful to know that if a part of a process is replaced by an equivalent part then the resulting process will be equivalentto the original

E-mail adresses: bk291@cam.ac.uk
[1] Supported by EPSRC Grant EP/D039045/1.

0890-5401/$ - see front matter
doi:10.1016/j.ic.2007.10.006

one. Compositionality proofs for specific languages can be quite lengthy, therefore in the literature many *congruence formats* have been proposed. Such a format is a syntactic restriction on SOS specifications that guarantees a specific equivalence to be compositional on the induced transition system. The most popular format is GSOS [10], which guarantees the compositionality of bisimilarity, but formats for other equivalences and/or kinds of transition systems have also been studied (see [1,23]).

The task of finding a reasonably permissive congruence format for a given equivalence is usually quite demanding, therefore it would be desirable to have a general framework for the derivation of formats as well as for proving compositionality for specific languages. To be sufficiently general, such a framework should be parametrized by the process equivalence and by the kind of transition system. It is the purpose of this paper to provide such a framework.

Our approach is based on the categorical framework of *bialgebraic semantics* [57], where process syntax is modeled via algebras, and transition systems are viewed as coalgebras. For example, LTSs are coalgebras for the functor $(\mathcal{P}-)^A$ on the category **Set** of sets and functions, where $\mathcal{P}$ is the powerset functor and A a set of labels, and other kinds of transition systems are coalgebras for other functors, called *behaviour* functors in this context. Coalgebras also provide a general and abstract notion of bisimilarity (for more information on the coalgebraic theory of systems, see [53]). As it turns out, SOS specifications in the GSOS format are essentially *distributive laws* of syntax functors over $(\mathcal{P}-)^A$. Moreover, the process of inducing an LTS with a syntactic structure on processes from SOS rules is a special case of an abstract construction, where distributive laws of syntax over behaviour induce *bialgebras*, i.e., coalgebras with algebraic structures on their carriers. Also the fact that GSOS is a congruence format for bisimilarity can be proved at the level of distributive laws. This makes bialgebraic semantics a general framework for deriving congruence formats for bisimilarities, parametrized by the kind of transition systems; it was used to this purpose in [7,16,30] for probabilistic, timed and name-passing systems. In this paper, the framework is further parametrized by the notion of process equivalence.

Typically, process equivalences are characterized by *modal logics*. For example, two processes in an LTS are bisimilar if and only if they satisfy the same formulas in Hennessy–Milner logic [24], and fragments of that logic characterize other interesting equivalences on LTSs. Several attempts have been made to generalize such logics to coalgebras of arbitrary type [48,40,49,54]. Recently [38], based on earlier insights of [11,12,41,50], we have proposed a categorical generalization of modal logics for coalgebras in arbitrary categories. There, the syntax of a logic is modeled via algebras for an endofunctor, and its semantics via a suitable natural transformation connecting the logic syntax with the process behaviour.

The main contribution of this paper is a combination of the coalgebraic perspective on modal logic taken in [38] with the bialgebraic approach to SOS from [57]. Roughly speaking, to merge a logic and its semantics with a distributive law representing an SOS specification, one should provide a suitable notion of behaviour for the logic, and define a "dual", logical distributive law, where formulas play the role of processes, in a way that reflects the SOS specification. One might think of the logical behaviour as a way to decompose logical formulas over the syntax of processes. Our main result says that if such a logical distributive law exists, then the equivalence characterized by the logic is compositional on the transition system induced by the SOS specification.

For some kinds of logical behaviours, logical distributive laws can be presented as SOS-like inference rules where formulas act for processes, logical operators (modalities) for syntactic constructs, and logical inference operators for transitions. For example, rules:

$$\frac{\phi \dashv \psi || \sigma}{\langle a \rangle \phi \dashv \langle a \rangle \psi || \sigma} \qquad \frac{\phi \dashv \psi || \sigma}{\langle a \rangle \phi \dashv \psi || \langle a \rangle \sigma} \tag{2}$$

are used to define a logical distributive law reflecting (1. In particular, the rule on the left says that if a formula ϕ holds for every process of the form $x||y$ such that ψ holds for x and σ holds for y, then the formula $\langle a \rangle \phi$ holds for every process of the form $z||w$ such that $\langle a \rangle \psi$ holds for z and σ holds for w. Since $\langle a \rangle \phi$ means that a process can do an a-transition to a process for which ϕ holds, this corresponds to the left rule in 1).

The framework proposed here can be seen as a very general "meta-congruence format", parametrized both by the notion of process equivalence and by the kind of transition system. It can be used directly to prove compositionality for specific languages and equivalences. Obviously it is hard to expect that such a general approach would be as easy to use as syntactic congruence formats designed for specific equivalences, and indeed finding the right logical distributive law and presenting it in a readable form is not always easy. However, our framework can also be used to derive specialized formats by proving that suitable distributive laws exist for a whole class of SOS specifications. The direct application to specific languages can be then left to problematic cases that do not fit in any known format.

The structure of the paper is as follows. The basics of classical SOS and congruence formats are presented in Section 2. In Section 3 the bialgebraic approach of [57] is explained on a series of very simple examples. A brief description of our approach to coalgebraic modal logic [38] follows in Section 4. In Section 5, the main technical result of the paper is obtained by merging the two approaches, and it is illustrated in Section 6 on some simple examples. Finally, Section 7 sketches some related and future work. Some familiarity with basic category theory is expected; [3,44] are good references.

The present paper is a full version of extended abstracts [36] and [37], with more detailed explanations provided and with more examples, including the substantial example of de Simone format in Section 6.3.