Journal of Logical and Algebraic Methods in Programming 93 (2017) 1–22

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Coalgebraic completeness-via-canonicity for distributive substructural logics

Fredrik Dahlqvist [*], David Pym

University College London, United Kingdom

ARTICLE INFO

Article history:
Received 1 February 2016
Received in revised form 21 July 2017
Accepted 31 July 2017
Available online xxxx

Keywords:
Completeness
Canonicity
Coalgebraic logic
Substructural logic
Resource modelling
Separation logic

ABSTRACT

We prove strong completeness of a range of substructural logics with respect to a natural poset-based relational semantics using a coalgebraic version of completeness-via-canonicity. By formalizing the problem in the language of coalgebraic logics, we develop a modular theory which covers a wide variety of different logics under a single framework, and lends itself to further extensions. Moreover, we believe that the coalgebraic framework provides a systematic and principled way to study the relationship between *resource models* on the semantics side, and *substructural logics* on the syntactic side.

1. Introduction

This work lies at the intersection of resource semantics/modelling, substructural logics, and the theory of canonical extensions and canonicity. These three areas respectively correspond to the semantic, proof-theoretic, and algebraic sides of the problem we tackle: to give a systematic, modular account of the relation between resource semantics and logical structure. Our approach will mostly be semantically driven, guided by the resource models of separation logic. We will therefore not delve into the proof theory of substructural logics, but rather deal with the equivalent algebraic formulations in terms of residuated lattices ([35] and [20] give an overview of the correspondence between classes of residuated lattices and substructural logics).

Resource semantics and modelling. Resource interpretations of substructural logics — see, for example, [26,34,36,21,11] — are well-known and exemplified in the context of program verification and semantics by Ishtiaq and O'Hearn's pointer logic [27] and Reynolds' separation logic [38], each of which amounts to a model of a specific theory in the logic of Boolean Bunched Implication (henceforth Boolean BI). Resource semantics and modelling with resources has become an active field of investigation in itself (see, for example, [10]). Certain requirements, discussed below, seem natural (and useful in practice) in order to model naturally arising examples of resource.

1. We need to be able to compare at least some resources. Indeed, in a completely discrete model of resource (i.e., where no two resources are comparable) it is impossible to model key concepts such as 'having enough resources'. On the other hand, there is no reason to assume that *any two* resources be comparable (e.g., heaps). This suggests at least a

[*] Corresponding author.
E-mail addresses: f.dahlqvist@ucl.ac.uk (F. Dahlqvist), d.pym@ucl.ac.uk (D. Pym).

http://dx.doi.org/10.1016/j.jlamp.2017.07.002
2352-2208/© 2017 Elsevier Inc. All rights reserved.

preorder structure on models. In fact, we take the view that comparing two resources is fundamental and, in particular, if two resources cannot be distinguished in this way then they can be identified. We thus add antisymmetry and work with posets.

2. We need to be able to combine *some* – but not necessarily all – resources to form new resources (e.g., union of heaps with disjoint domains [27]). We denote the partial combination operation on resources by $\otimes$ (see also [21] for examples of partial structures in pointer logic and separation logic). An alternative, relational, point of view is that we should be able to specify how resources can be 'split up' into pairs of constituent resources. From this perspective, it makes sense to be able to list for a given resource r, the pairs (s_1, s_2) of resources which combine to form a resource $s_1 \otimes s_2 \leq r$.
3. All reasonable examples of resources possess 'unit' resources with respect to the combination operation $\otimes$; that is, special resources that leave other resources unchanged under the combination operation.
4. The last requirement is crucial, but slightly less intuitive. In the most well-behaved examples of resource models (e.g., heaps or $\mathbb{N}$), if we are given a resource r and a 'part' s of r, there exists a resource s' that 'completes' s to make r; that is, we can find a resource s' such that $s \otimes s' = r$. More generally, given two resources r, s, we want to be able to find the best s' such that $s \otimes s' \leq r$. In a model of resource without this feature, it is impossible to provide an answer to legitimate questions such as 'how much additional resource is needed to make statement ϕ hold?'. Mathematically, this requirement says that the resource composition is a residuated mapping in both its arguments.

The literature on resource modelling, and on separation logic in particular, is vast, but two publications – [7] and [5] – are strongly related to this work. Both show completeness of 'resource logics' by using Sahlqvist formulas, which amounts to using completeness-via-canonicity ([4,29]).

Completeness-via-canonicity and substructural logics. The logical side of resource modelling is the world of substructural logics, such as BI, and of their algebraic formulations; that is, residuated lattices, residuated monoids, and related structures. The past decade has seen a fair amount of research into proving the completeness of relational semantics for these logics (for BI, for example, [36,21]), using, among other approaches, techniques from the duality theory of lattices. In [19], Dunn et al. prove completeness of the full Lambek calculus and several other well-known substructural logics with respect to a special type of Kripke semantics by using duality theory. This type of Kripke semantics, which is two-sorted in the non-distributive case, was studied in detail by Gehrke in [22]. The same techniques have been applied to prove Kripke completeness of fragments of linear logic in [12]. Finally, the work of Suzuki [39] explores in much detail completeness-via-canonicity for substructural logics. Our work follows in the same vein but with some important differences. Firstly, we use a dual adjunction rather than a dual equivalence to connect syntax and semantics. This is akin to working with Kripke frames rather than descriptive general frames in modal logics: the models are simpler and more intuitive, but the tightness of the fit between syntax and semantics is not as strong. Secondly, we use the topological approach to canonicity of [23,25,40] because we feel it is the most flexible and modular approach to building canonical (in)equations. Thirdly, we only consider distributive structures. This is to some extent a matter of taste. Our choice is driven by the desire to keep the theory relatively simple (the non-distributive case is more involved), by the fact that, from a resource-modelling perspective, the non-distributive case does not seem to occur 'in the wild' and, finally, because we place ourselves in the framework of coalgebraic logic, where the category of distributive lattices forms a particularly nice 'base category'.

Completeness-via-canonicity, coalgebraically. The coalgebraic perspective brings many advantages to the study of completeness-via-canonicity. First, it greatly clarifies the connection between canonicity as an algebraic method and the existence of 'canonical models'; that is, strong completeness. Second, it provides a generic framework in which to prove completeness-via-canonicity for a vast range of logics ([16]). Third, it is intrinsically modular; that is, it provides theorems about complicated logics by combining results for simpler ones ([8,15]). We will return to the advantages of working coalgebraically throughout the paper.

2. A coalgebraic perspective on substructural logics

We use the 'abstract' version of coalgebraic logic developed in, for example, [31], [32] and [28]; that is, we require the following basic situation:

$$\begin{tikzcd}
\mathscr{C} \arrow[loop left, "L"] \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathscr{D}^{\mathrm{op}} \arrow[loop right, "T^{\mathrm{op}}"] \arrow[l, bend left, "G"]
\end{tikzcd} \tag{1}$$

The left hand-side of the diagram is the syntactic side, and the right-hand side the semantic one. The category $\mathscr{C}$ represents a choice of 'reasoning kernel'; that is, of logical operations which we consider to be fundamental, whilst L is a syntax constructing functor which builds terms over the reasoning kernel. Objects in $\mathscr{D}$ are the carriers of models and T specifies the coalgebras on these carriers in which the operations defined by L are interpreted. The functors F and G relate the syntax and the semantics, and F is left adjoint to G. We will denote such an adjunction by $F \dashv G : \mathscr{C} \to \mathscr{D}$. Note, as mentioned in the introduction, that we only need a dual adjunction, not a full duality.