A canonical proof-theoretic approach to model theory

Carlos G. Lopez Pombo[1,2], Paula D. Chocrón[1,2], Ignacio Vissani[1,2], and Tomas S.E. Maibaum[3]

[1] Department of Computing, FCEyN, Universidad de Buenos Aires
[2] Consejo Nacional de Investigaciones Científicas y Tecnológicas (CONICET)
[3] Department of Computing and Software, McMaster University

Logic has proved essential as a formal language for describing different aspects of software artefacts. These formal descriptions, frequently called specifications, have served not only as requirements documentation but also for proving properties, provided the logical language in which the specification is written has an appropriate reasoning tool. Semantics is an integral part of logic, as providing logical descriptions of real-world phenomena requires people to agree on how these descriptions should be interpreted. In this sense, model theory has been seen as providing the cornerstone for the satisfaction of this need. Model theory is usually understood as the study of classes of mathematical structures satisfying formulae in a formal language of choice. Model theory is a tool for characterising semantic notions, like meaning and truth, associated to syntactic objects, like formulae and proofs, of a corresponding language. From a category theory point of view the model theory of a logic has been formalised as an *institution* [1]. An institution is a structure $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models_\Sigma\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ formed by: 1) a category of signatures Sign, 2) a grammar functor $\mathbf{Sen} : \mathsf{Sign} \to \mathsf{Set}$ providing a set of sentences for each signature, 3) a model functor $\mathbf{Mod} : \mathsf{Sign}^{op} \to \mathsf{Cat}$ providing a class of semantic structures for each signature, 4) a family of binary relations $\{\models_\Sigma\}_{\Sigma \in |\mathsf{Sign}|}$ such that given $\Sigma \in |\mathsf{Sign}|$, $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times |\mathbf{Sen}(\Sigma)|$, and 5) satisfying that for all $\sigma : \Sigma \to \Sigma' \in ||\mathsf{Sign}||$, $\alpha \in \mathbf{Sen}(\Sigma)$ and $\mathcal{M} \in \mathbf{Mod}(\Sigma')$, $\mathcal{M} \models_{\Sigma'} \mathbf{Sen}(\sigma)(\alpha)$ if and only if $\mathbf{Mod}(\sigma)(\mathcal{M}) \models_\Sigma \alpha$.

In mathematical logic, given a logical system, we are forced to consider all possible semantic structures that can interpret its sentences, while in computer science we are mostly concerned about the analysability of the semantics as we rely on it to prove properties of software artefacts (as well as meta properties of the logic itself). Usually, we place trust only in those structures that can be described resorting to a formal logical language; typically they are maximal consistent theories in the language of choice, like those used in Henkin's completeness proof for equational logic [2], or theories over some formalisation of set theory. Our aim is to undertake a recasting of the notion of semantics in syntactic terms to support this approach.

Behavioural specifications, such as those written in any dynamic logic [3], temporal logics, both linear time [4], and branching time [5,6], etc., usually involve the following common elements: 1) an interpretation of a subset of symbols whose interpretation is fixed for all states usually referred to as rigid, 2) an ordering of states (for example, sequences of states in linear temporal logics, trees

of states in branching time temporal logics, a single state in dynamic logics, etc.) such that each of the constituent states provide an interpretation of a different subset of symbols, referred to as flexible symbols, and 3) a satisfaction relation providing meaning for behavioural logic operators. Generally, the ordering of states is obtained from a binary relation between them; for example, in dynamic logics there is a set of atomic actions and regular programs defined over them; in temporal logics, both linear time and branching time, there is a single transition relation; in deontic logics there are events produced by actions, etc.

This paper addresses the question of whether such a class of structures can be constructed in a canonical way so the definition of the functor **Mod**, in the definition of institutions, can be given in concrete representable terms.

Equational logic extended with extra-logical predicate symbols has been widely accepted as an appropriate specification language for describing the operations of abstract data types [7]. Equational theories can also be used to provide interpretations, of extra-logical symbols by considering formulae of the form $f(t_1, \ldots, t_n) = t$ and $P(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ and t are ground terms of the logical language of choice. On the other hand, we extend the *Elementary Theory of Binary Relations* [8] by incorporating the additional relational operators of ω-closure fork algebras [9]. This class of relation algebras have been used to reason about relations due to their complete (almost) equational calculus and its easy-to-understand concrete semantics, build out of a set of binary relations.

In this work we propose a general framework that facilitates the definition of the semantics of a logical system by identifying and properly characterising its static and dynamic properties. To make the approach flexible, we propose the use of a higher-order extension of equational logic to formalise the static aspects, while the dynamic properties characterising the accessibility relations between states are expressed by means of concrete models for fork algebraic terms.

References

1. Goguen, J.A., Burstall, R.M.: Institutions: abstract model theory for specification and programming. Journal of the ACM **39**(1) (1992) 95–146
2. Henkin, L.A.: The logic of equality. The American Mathematical Monthly **84**(8) (1977) 597–612
3. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. Foundations of Computing. MIT Press.
4. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems. Springer-Verlag.
5. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. Jour. of Comp. and Syst. Sciences **30**(1) (1985) 1–24
6. Pnueli, A.: The temporal logic of programs. In: Proceedings of 18th. Annual IEEE Symposium on Foundations of Computer Science, Los Alamitos, CA, USA, IEEE Computer Society, IEEE Computer Society (1977) 46–57
7. Ehrig, H., Mahr, B., Orejas, F.: Introduction to algebraic specification: Formal methods for software development. Computer Journal **35**(5) (1992) 468–477
8. Tarski, A.: On the calculus of relations. Jour. of Symb. Logic **6**(3) (1941) 73–89
9. Frias, M.F.: Fork algebras in algebra, logic and computer science. Volume 2 of Advances in logic. World Scientific Publishing Co., Singapore (2002)