Semantics for Non-symbolic Computation: Neural Networks and Other Analog Computers

Levin Hornischer

LMU Munich, Munich Center for Mathematical Philosophy, Munich, Germany
levin.hornischer@lmu.de

Introduction Despite the great technological progress, we are lacking a foundational theory of modern artificial intelligence (AI). Specifically, we want to interpret, explain, and verify the 'sub-symbolic' computation performed by neural networks that drive this success. For classical 'symbolic' computation, this problem was solved by *semantics*: the mathematical description of the meaning of program code. In this talk, we develop one approach to an analogous semantics for non-symbolic computation performed by neural networks and other analog computers. To do so, we first summarize the three semantics for symbolic computation, and then we describe our analogous components—systems, domains, and logic—for non-symbolic computation, visualized in figure 1. The key idea is to represent the dynamics of the non-symbolic computation as a limit of symbolic approximations, which are given by observations.

Semantics for symbolic computation Symbolic computation is specified by some *program code* P written in some programming language, and semantics should assign meaning to P. There are three approaches. First, 'systems': *Operational semantics* describes P by the steps a machine would take to implement this program, so the meaning of our program is given by a transition system. Second, 'domains': *Denotational semantics* describes P by the function (or denotation) $[\![P]\!]$ that it computes and the finite approximations to this function. The set of all denotations and approximations of programs of a given type σ forms a so-called domain D_σ. Third, 'logic': *Logical semantics* describes P by the properties it has: e.g., if the input is 1, then executing P yields an even output, which is written as the Hoare triple $\{\mathsf{is\ 1}\}P\{\mathsf{is\ even}\}$.

Ideally, these three semantics are in harmony: *Partial correctness* requires that if a Hoare triple $\{\varphi\}P\{\psi\}$ is provable, then running program P in a state satisfying φ results in a state satisfying ψ (if P terminates). *Full abstraction* requires that two programs have the same denotation iff the machines running the two programs show the same behavior. *Stone duality* requires that the properties of P jointly determine the denotation $[\![P]\!]$, and vice versa [1].

For an analogous semantics for non-symbolic computation, we now explicate the italic terms.

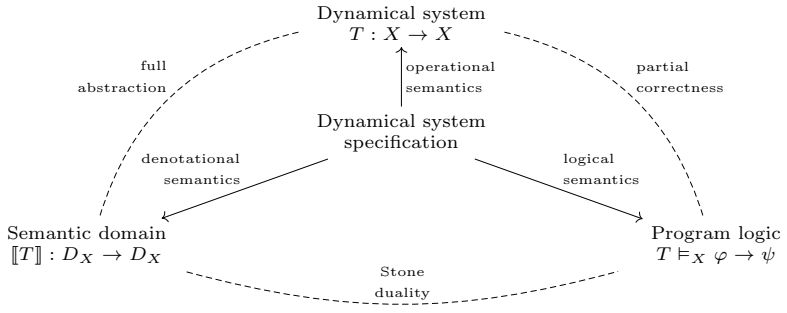

Figure 1: The threefold semantics for non-symbolic computation.

Systems If symbolic computation is specified by program code, how is non-symbolic computation specified? The answer is: by dynamical systems [2]. Let's consider neural networks as an example. (Other famous examples are cellular automata or differential analyzers.) Understanding their training dynamics is essential for a theory of deep learning. Neural networks are trained by backpropagation: given a batch of training data, it updates the current weights w of the network to weights w' with smaller training loss. Hence backpropagation specifies a dynamical system $T : W \times D^\omega \to W \times D^\omega$, where W is the weight space and D is the set of batches of data, and T maps a pair (w, d) of a weight w and a sequence of batches d to the pair $(w', \sigma(d))$, where w' is the result of updating w with the batch d_0 and $\sigma(d) = d_1 d_2 \dots$ (i.e., σ is the shift operator on sequences). Thus, the analogue of a program code is a dynamical system specification like backpropagation (or the rule of a cellular automaton or the differential equation specifying the differential analyzer, etc.). The analogue of a transition system is a dynamical system $T : X \to X$. Formally, we take X to be a zero-dimensional compact Polish space and T a continuous function (as studied in the field of topological dynamics in dimension zero [3]; though in [4] we also cover probabilistic systems).

Domains As in symbolic semantics, we obtain the 'meaning' of the dynamical system in the limit of finite, 'interpretable' approximations to the system. These approximations are given via observations about the system: e.g., that with the current set of weights w the neural network classifies this given image correctly. As will be explained in the talk, we package these observations as finite domains and, by refining the observations, we obtain in the limit (in the category-theoretic sense) a domain D_X with a Scott-continuous function $[\![T]\!] : D_X \to D_X$, which we call the dynamical domain. Formally, we develop this idea as a functor from the category of dynamical systems to the category of dynamical domains. This functor has a left adjoint, which naturally restricts to an equivalence—this can be regarded as a form of full abstraction [4].

Logic Finally, the finite observations of the system can be identified with clopen subsets of the state space X. The Hoare triple $\{\varphi\}T\{\psi\}$ then says: whenever we observe the system having property φ now, we observe property ψ next, i.e., $\varphi \subseteq T^{-1}(\psi)$. We can reformulate this as a Boolean algebra with operators (BAO): let A be the Boolean algebra of clopen subsets of X and let $\Box := T^{-1} : A \to A$. Then the Hoare triple is the conditional $a \to b := \neg a \vee \Box b$, which is valid when equal to X. So Stone duality not only links these BAOs to our dynamical domains but can also be regarded as a form of partial correctness.

References

[1] Samson Abramsky. Domain theory in logical form. *Annals of pure and applied logic*, 51(1-2):1–77, 1991.

[2] O. Bournez and A. Pouly. A survey on analog models of computation. In V. Brattka and P. Hertling, editors, *Handbook of Computability and Complexity in Analysis*, pages 173–226. Springer, Cham, 2021.

[3] T. Downarowicz and O. Karpel. Dynamics in dimension zero: A survey, 2016. https://doi.org/10.48550/arXiv.1610.02727.

[4] L. Hornischer. *Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation*. PhD thesis, University of Amsterdam, Institute for Logic, Language and Computation, 2021. https://www.illc.uva.nl/Research/Publications/Dissertations/DS-2021-10.text.pdf.