Plan for Theory Sessions of CS266

Dan Yamins

August 9, 2006

Here is my rough plan for the two weeks (four class sessions) of local-to-global theory in CS266.

Session 1, 11/14/2006 – Two Solutions to Equigrouping

In this session, we will discuss pattern formation on the direction one-dimensional discrete lattice. Specifically, we will:

- Introduce a simple model of state-less agents on the 1-d directed lattice, and describe several pattern formation problems that can be formulated in this model (equigrouping, etc...)
- Construct and analyze the high-r low s standard solution to the equigrouping problem, no generalizing.
- Constuct and analyze the high-state low r equigrouping algorithm (but not introducing a formal model of state)

Session 2, 11/16/2006 – General Patterns

In this session, we will generalize to arbitrary repeat patterns. Specifically, we will:

- Construct and analyze the high-r low s solution to general repeat patterns.
- Construct the standard shift-reduction for 1-D repeat patterns to get a high-s low r solution to general repeat patterns.

Session 3, 11/21/2006 – Locality and the Radius/State Trade-off

In this session, we will describe the notion of local checkability and its consequences and use. Specifically, we will:

- Introduce the concept of local checkability and describe why it is a necessary condition for the existence of a local rule to work on arbitrary initial conditions.
- Give a bunch of examples of local checkability (and the lack of it), and show how it depends on the number of available states.
- Show how gradients are an example of a local check.
- Describe the radius/state trade-off.
- Analyze the previous day's constructions in these terms and show how local checks can optimize the constructions.

Session 4 – 11/28/2006

I am considering two options for session 4.

Option 1 – Symmetries, Polarity Emergence, Compromise, and Cooperativity
In this session, we would discuss how symmetries impose a further constraint on what is possible in local systems, and how various symmetry-breaking and resolution mechanisms work. We will apply this to the emergence of polarity. Specifically, we would:

- Introduce the undirected 1-d lattice and ring lattice topologies.
- Show how simple pattern formation problems become harder/impossible in light of the symmetries of these topologies.
- Show how asynchronicity can be used to break symmetry – by constructing a way to emerge polarity on the 1-d undirected line. The punchline: an initial local asymmetry can be globally propagated just like in the case of the ant algorithms.
- Show how the radius/state trade-off that worked for local checkability breaks down at "edge-cases" symmetry and polarity problems, leading to the need for inter-agent cooperation.
- Show how asynchronicity is NOT good enough on the ring lattice – but can be overcome if the problem admits "averaging" or "compromise." The punchline: this is the intuitive reason that flocking algorithms work.

Or:

Option 2 – Glocal Compilation
In this session, we would discuss the question of Glocal Compilation, and provide the construction of the first step of glocal compilation in 1-D. Specifically, we would:

- Introduce a formal framework for multi-agent systems, and introduce logic modeling in this framework.
- Define the Glocal Compilation Problem formally given these descriptive tools.
- Describe the division of glocal compilation in 1-D into two steps: from logic to static local check ($\varphi \to \Theta$); and from local check to dynamic local rule ($\Theta \to f$), of which we will only pursue the first part.
- Show that simple logics in 1-D admit a procedure by which logical formulas can be written in a standard form from which local checks (or the lack of local checkability if that be the case) can be easily "read off."
- Describe the classification of 1-D local check schemes and the decidability of the 1-D "Global Satisfiability" problem.
- Compare several known local algorithms to the local check schemes constructed by the above procedure, and interpret the (few) existing partial compilers in this light.