A Catalog of Biologically-inspired Primitives for Engineering Self-Organization

Radhika Nagpal

Department of Systems Biology, Harvard Medical School
240 Longwood Avenue, Boston MA 02115, USA
rad@eecs.harvard.edu

Abstract. The Amorphous Computing project is aimed at developing programming methodologies for systems composed of vast numbers of locally-interacting, identically-programmed agents. This paper presents some of the building blocks for robust collective behavior that have emerged as part of this effort, and describes how organizing principles from multi-cellular organisms may apply to multi-agent systems.

1 Introduction

During embryogenesis, cells with identical DNA cooperate to form complex structures such as ourselves, starting from a mostly homogeneous egg. The process involves a complex cascade of spatial and functional decisions taken by individual cells. Even so, the process of development is incredibly robust to variations in individual cell behavior, varying cell size and division rates, and cell death. Many organisms develop correctly independently of initial embryo size; for example, frog embryos with half as many cells develop into half-size larva. Even after development, many retain the ability to regenerate entire structures that have been damaged. For example, newts can regenerate amputated limbs and some species of starfish can regenerate an entire new body from a limb. The ability of multi-cellular organisms to achieve such complexity and reliability is awe-inspiring for an engineer.

Emerging technologies such as MEMS devices (micro-electronic mechanical devices), are making it possible to bulk-manufacture millions of tiny computing and sensing elements and embed these into materials, structures and the environment. Applications being envisioned include smart environments where sensors are ``painted'' onto walls or surfaces, reconfigurable robots/structures composed of millions of identical modules that self-assemble into different shapes to achieve different tasks, armies of ant-like robots that can collectively achieve complex global tasks [1,2,3,4]. The charm of such systems is the potential of replacing specialized engineering with vast numbers of cheaply bulk-manufactured generic parts, that can achieve many purposes through programming.

This raises a fundamental question: how do we program such systems? We expect that there will be too many elements to individually program or name, and it will not be possible to control precise layout and precision interconnects. Individual agents are likely to have limited resources, limited reliability, and only local information and communication. Ideally we would like the system to produce robust global behavior in spite of individual limitations and failures, and be self-maintaining without human intervention. How do we achieve robust global behavior from vast numbers of unreliable agents? And can we abstractly describe global goals and automatically derive robust distributed agent programs?

The objective of the Amorphous Computing project is to invent programming methodologies for such multi-agent systems [5]. The earliest emphasis was on pattern-formation and self-assembly, and much of the inspiration came from multicellular organisms. Several simulated systems were designed that could translate user-specified patterns and shapes into robust agent programs. The agent programs did not rely on regular placement, synchronous behavior, or perfectly reliable agents [6,7,8].

A common methodology emerged from these examples: goals were described at a high level using programming languages based on generative rules, and these generative rules were then mapped to local programs for agent behavior [9]. In addition, a common set of simple and powerful primitives for agent interactions emerged that were sufficient to create large classes of complex structures. Two important aspects of these primitives are (1) they are generally insensitive to variations in individual behavior, numbers of agents, timing, and placement (2) it is possible to theoretically analyze their behavior. Many of these primitives were directly inspired by studies from developmental biology, i.e. how cells organize to make coordinated spatial decisions robustly during embryogenesis. As we continue to think about designing self-maintaining and self-repairing systems, the analogy to multi-cellular behavior continues to be extremely fruitful [10].

While the global-to-local programming methodology has been presented elsewhere [9], this paper is a first attempt towards assembling a catalog of primitives for multi-agent control, inspired by metaphors from multi-cellular systems.

2 Primitives for Robust Local Behavior

Biologists have long been fascinated with the ability of cells in an embryo to locally coordinate to develop into complex organisms and the ability of this process to regulate in the face of failures and natural variation. There is a large literature of conceptual ideas of how cells might achieve this robustness and complexity [11,12] and how embryogenesis can be thought of as a program of interacting cells [13]. These concepts can provide a basis for designing robust distributed algorithms that achieve similar goals in our artificial systems.

This section discusses a set of basic building blocks that can be achieved by simple agent programs. Complexity is achieved by the composition of these building blocks in principled ways. The primitives are:

1. Morphogen gradients and positional information
2. Chemotaxis and directional information
3. Local inhibition and local competition
4. Lateral inhibition and spacing
5. Local monitoring
6. Quorum sensing and counting:
7. Checkpoints and consensus
8. Random exploration and selective stabilization:

The first five have been extensively used in Amorphous Computing. The remaining three are basic behaviors that may prove useful as we design systems with different types of goals. These primitives represent a small subset of the mechanisms actually used in embryogenesis; for example, they do not touch upon mechanical methods for interaction such as cell movement, differential adhesion, or local forces.

2.1 Morphogen Gradients and Positional Information

Regional specification is an important aspect of embryogenesis. Undifferentiated cells must determine what part of the structure to form. In many organisms, the embryo starts with a small population of cells as organizing centers or poles, which create gradients of diffusable factors called morphogens. Other cells in the embryo can use the morphogen concentration to determine what region they lie in relative to the pole. For example in the fruit fly embryo, poles are established at the anterior and posterior ends, and gradients of factors such as *bicoid* determine the initial segmentation into head, thorax and abdomen regions. Finer-grain segmentation occurs as these coarse regions produce other morphogens [12]. Conceptually morphogen gradients have played an important role in understanding how cells acquire positional information [14,13].

Morphogen gradients have been a key primitive in amorphous computing systems, and similar primitives have independently emerged in sensor networks and reconfigurable robots for self-organizing position information [2,4]. A simple agent program can produce a "morphogen gradient": an agent creates a morphogen gradient by sending a message to its local neighborhood with the morphogen name and a value of zero. Agents who hear the message forward the message to their local neighbors with the value incremented by one, and so on until the morphogen has propagated over the entire system. Each agent stores and forwards only the minimum value it has heard for a particular morphogen name. Thus the value of the morphogen increases as one moves away from the source agent, and the value reflects the distance to the source. The source may even be multiple agents. An important variation on this primitive is *active morphogens*, where the value of the morphogen decays over time. Here the source agent must constantly produce morphogen messages to maintain the gradient and the gradient adapts as the network topology changes. The morphogen primitive is easy to analyze, provides reasonable distance estimates, and is robust to many different types of variations and failures [6].

Morphogen gradients can be used to infer positional and geometric information in many ways. For example, agents can compare their local value to a threshold to determine if they are within a region relative to the source, or agents can compare different morphogens to determine where they lie relative to two sources. It is an extremely versatile primitive, and appears as part of the agent program for many of the primitives described below. In that way it can be considered to be more fundamental than the rest.

2.2 Chemotaxis and Directional Information

Chemotaxis refers to the ability of an organism to follow the gradient of a diffusing substance. There are a plethora of examples in biology where gradients are used to sense direction: neurons detect and follow chemical gradients, yeast cells create extensions in the direction of mating signals, bacteria can migrate towards higher concentrations of nutrients [11].

An agent program can be created that emulates this behavior by combining the morphogen primitive from above with the ability of agents to query their neighbors. An agent can collect the values of a morphogen in its neighborhood and by comparing these values determine the direction of the gradient. An agent can then select a neighbor agent that is closer to the source. This process can be used to create a route from an agent to the source, and an interesting aspect of this primitive is that it guarantees connectivity in the event of large regional failures. Thus unlike the previous primitive that was useful for geometric inferences, this primitive is good for direction and connectivity. This primitive was a key piece in the work by Coore on self-organizing topological patterns, and since has been extensively exploited in amorphous computing systems [7].

2.3 Local Inhibition and Local Competition

Within regions of cells, some cells may differentiate to take on special roles. For example in epithelial (skin) cells, only some cells differentiate to produce hair follicles [12]. How these cells are chosen can be thought of as a process of local competition and local inhibition. Through some random process, a cell chooses to differentiate and it simultaneously creates a local signal that inhibits neighboring cells from differentiating. The competition may also dependent on the fitness of the individual cells. A particularly interesting example of such competition is in the developing wing of the fruit fly, where fast dividing cells cause slowly dividing cells to commit apoptosis, even though the slower cells would be capable of producing the entire wing in the absence of the fast cells. The faster cells are able to signal neighboring slower cells, causing them to lose the competition [15,16].

Local inhibition and competition are useful primitives for leader election within a spatially local group of candidate agents. A simple agent program can be designed to achieve this, by combining a random process with limited range morphogen gradients.

All competing agents pick random numbers and count down. If an agent reaches zero without being interrupted then it becomes a leader and produces a morphogen that inhibits other agents from competing further. This algorithm is also simple to analyze and has been shown to be robust to asynchronous behavior and agent death and addition [5]. A variation on this primitive allows agents to create morphogens with a value related to their fitness, and agents are only inhibited by morphogens with higher fitness [8]. This primitive also has the ability to automatically adapt if the leader agent dies, since the inhibiting influence of the morphogen also disappears and the competition can resume. In Amorphous Computing this primitive has been used extensively for choosing an agent from a set of agents.

2.4 Lateral Inhibition and Spacing

Many patterns in biology exhibit beautiful regular spacing; for example, bristle and hair patterns, scales, and fruit fly eyes. Many different theoretical models have been developed to explain how such spacing is generated [12]. One such model is called lateral inhibition and is believe to be the mechanism by which bristles emerge at regular intervals. It is in fact almost identical to the previous model, except that cells produce an inhibition morphogen with a much smaller range than the extent of competition. Thus a differentiated cell may inhibit spatially local cells from differentiating, but globally the competition continues until all cells are either inhibited or differentiated.

The agent program is the same as before. Agents pick random numbers and if an agent counts down to zero without being inhibited, it creates a morphogen of range *d*. The remaining uninhibited agents continue to compete. At the end of the competition, no two leader agents will be within distance *d* of each other with high probability, and all agents will be within distance *d* of some leader. Thus a regular spacing of leader agents is produced within a field of agents. This algorithm has been used to design several simple schemes for self-organizing hierarchical structure [5].

2.5 Local Monitoring

Most organisms are capable of wound repair, which depends on the ability of neighboring cells to detect regional death and respond appropriately. Detection may be through mechanical means, such as contact, and repair may then consist of growth or recruiting migrating cells to fill in the void.

In agents, we can emulate detection of regional death through local monitoring. Each agent periodically broadcasts a message of *aliveness* to its neighbors, and keeps track of when it last heard an aliveness message from its neighbors. In the event of a missing neighbor, the agent can trigger some response. This primitive presents an interesting tradeoff. The frequency of aliveness messages from neighbors determines the speed with which faults can be detected and the lag in response time. Faster response time comes with a higher cost of communication. Local monitoring can be used to make many of the previously presented primitives capable of self-repair, such as the chemotaxis primitive. As mentioned before, the chemotaxis primitive has been used to self-organize topological patterns. By using active gradients and local monitoring, this primitive creates topological patterns that seamlessly adjust connectivity as regions of agents die or are replaced [10].

2.6 Quorum Sensing and Counting

Many single cells organisms exhibit the capability of acting as a coordinated multicellular system. An interesting example of this is bioluminescent bacteria, which coordinate to produce light only when the number of bacteria within a confined region is sufficiently high. The bacteria secrete small diffusible molecules, called autoinducers, which accumulate in the confined space. When there are many bacteria the level of autoinducer in the confined space rises rapidly. The detection of high levels of autoinducer causes bacteria to secrete even more autoinducer and as a result of this positive feedback the level rises exponentially. When the level exceeds some threshold, the bacteria activates its bioluminescent behavior [17].

Quorum sensing is a concept used to describe the ability of cells to determine whether there are sufficient cells (i.e. a quorum) to trigger some activity, such as bioluminescence or virulence. A quorum sensing agent program can be designed based on the bacteria model, which would be useful for situations where it was necessary to determine when the number of agents in a particular state exceeded the minimum threshold for activity.

2.7 Checkpoints and Consensus

Certain decisions require that a set of parallel actions all be completed before proceeding into the next phase. An example of the need for consensus happens during cell division, where the process of dividing can only take place after all chromosome pairs have separated. Such a decision represents a "checkpoint" in the cell cycle; a single failing chromosome pair can halt the process of division until it is successfully separated. The mechanism by which this is achieved is conceptually a very simple distributed process. Each un-separated chromosome pair emits a halting signal and the cell cycle only resumes when all halting signal has disappeared [18].

This simple form of consensus amongst agents can be implemented using active morphogen gradients. Each agent is the source of the halting morphogen, and it discontinues being a source when its internal condition has been satisfied. However the overall morphogen persists until all sources have discontinued. When the last source agent stops producing the morphogen, agents detect the lack of morphogen and can move to the next phase.

Quorum sensing and consensus represent two closely related primitives for achieving coordinated behavior and synchronization, in one case a quorum of agents is sufficient to proceed whereas in the second case all agents must agree to proceed.

2.8 Random Exploration and Selective Stabilization

This last example is another commonly used paradigm at many different levels in biology. The basic concept is: many parallel but random explorations are started and terminated on a regular basis, however occasionally some explorations succeed and these selected explorations stabilize. Furthermore the stabilized explorations may bias the generation of new explorations to occur within its vicinity. Ant foraging behavior is an example of this paradigm. Ants lay random trails in search of food but the trails disappear over time. Successful trails get stabilized by attracting more ants to follow that path and reinforce the scent. Surprisingly, similar processes are observed even within single cells. During cell division, microtubules form a spindle around the chromosomes pairs and then later supply the force to separate them. This spindle is generated by organizing centers that create short-lived microtubules in random directions. However whenever a microtubule attaches to a chromosome, it stabilizes. Eventually all chromosomes become attached and this results in the cell moving on to the next phase of cell division. This general principle is called random exploration and selective stabilization [18].

Agent programs for mobile robots have been designed based on this principle. Such a process however is energy intensive. An alternative method to achieve the same result would be to have the "food" or "chromosomes" create a long-range morphogen gradient and use chemotaxis. While this would be more time and energy efficient, it requires the ability for long-range signaling. This represents an interesting tradeoff between two choices of primitives.

3 Primitives for Robust Global Behavior

In addition to making robust primitives, we also need organizing principles for combining these primitives in ways that produce globally robust results. This area is much less well understood, and the discussion in this section is preliminary. However there are some clear common patterns in development that can guide how we design global languages and compositions of primitives. In this section I will briefly describe five such concepts:

1. Cell differentiation or context-based roles
2. Asynchronous timing and dataflow
3. Compartments and spatial modularity
4. Scale-independence
5. Regeneration

3.1 Cell differentiation or Context-Based Roles

Rather than start with a completely pre-specified plan for each cell, many embryos first divide into a large number of cells that then slowly differentiate into the various parts of the structure. Often this process takes place as successive rounds of refinement, with coarse regions being determined first and then patterning occurring within those coarse regions. This process has the following interesting property --- cells take on roles based on being *at the right place at the right time*. This simple principle provides a key level of robustness to cell death and cell addition, irrespective of the structure being formed. In many cases cells are capable of changing their fate if conditions change, and this further increases the ability to adapt. At a global programming level this suggests that organizing information as needed and allocating agents when needed, as opposed pre-planning at the level of individual agents, automatically provides a certain degree of robustness, In many of the amorphous computing systems it has been our experience that locally creating information when needed, results in a process that can tolerate small mistakes throughout the system and removes the reliance on any single agent's survival.

3.2 Asynchronous Timing and Dataflow

Embryogenesis involves a cascade of decisions that must take place in a particular sequence. Most embryos show a considerable amount of tolerance to variations in timing and overall development time can be slowed or speeded up without much damage. In a large part, this robustness comes from the fact that the sequence of events does not rely on an absolute global clock [13,11]. Instead the cascade is linked through the flow of information. For example, in the frog embryo the neural plate can not fold into the neural tube before gastrulation (formation of the gut) happens, because it is the process of gastrulation which induces a region of cells to become the neural plate [13]. This form of asynchronous timing is often described as a set of dominos, such that toppling one causes the next to be set in motion. As a result the system can tolerate considerable variation in the timing of individual decisions without significantly affecting the success of the overall cascade. Even more complex forms of timing regulation exist, for example in the wing of the fruit fly different compartments develop independently and may take different amounts of time. However a compartment will wait for the remaining compartments to complete before proceeding to the next phase [15]. One can compare this conceptually to the notion of fork and join in dataflow programming. By using similar notions of asynchronous timing and flow of global information, and by avoiding global clocks, we can design multi-agent systems that can tolerate external causes of timing variations. Most amorphous computing systems use a domino-like timing strategy, and a few actually implement fork and join [7].

3.3 Compartments and Spatial Modularity

A common strategy in embryogenesis is to divide the cells into regions or compartments. An example is the imaginal discs in insects, which are the regions that develop into wings, limbs or antenna [12]. Much of the program for setting up the imaginal disc is common, and abstracted away from the program that generates a specific structure, such as leg. As a result, simple gene manipulations can change the specific structure from one to another, or create additional imaginal discs. This strategy has interesting implications. By capturing common and generic patterns as a spatial module, it leads to a significant reduction in program complexity. It also provides a form of isolation, because cells in two different but non-overlapping compartments can execute the same sequence (make a leg) without interfering. One can think of this as the ability to simultaneously execute the same procedure in different spatial contexts, which provides both program and timing modularity. Regions can be used in multi-agent systems to provide similar benefits, for example morphogens can be confined to propagate within a region thus allowing multiple parts of the system to use the same morphogen name and programs without interference [6].

3.4 Scale-independence

As mentioned before, many organisms can develop correctly over a large variation in initial embryo size. Frog and sea urchin embryos develop correctly over 8-fold variation and the hydra develops over a few magnitudes of size difference. The ability of an embryo to proportionally regulate with size, without any change in the development program, is called scale-independence or size-invariance [11]. Several models have been proposed for how locally-generated complex patterns could scale. Wolpert [14] introduced the canonical French Flag problem: write a program for a region of cells such that the cells differentiate into three equally sized sections of blue, white and red, irrespective of the size of the region. One solution he proposed was to use "balancing gradients"; each cell always used the ratio of two morphogens, one from each pole of the region, to determine which part of the global pattern it belonged to. Another method for generating scale-independent patterns has been used in Amorphous Computing, which is to form the pattern by recursively subdividing the space [6]. Each new segmentation decision is taken proportional to the existing set of segmentations, By building in the notion of proportional decisions at the global level of description, one can create complex systems with the ability to adapt to large variations in the number of agents.

3.5 Regeneration

Cockroaches can regenerate amputated limbs in remarkable ways and provide an elegant example of conceptual models of self-repair [11]. If a limb is amputated, the cockroach will regenerate the remaining part. If a section of the limb is removed, it is able to regenerate the missing intervening part. Moreover, if a section of the leg is grafted on backwards, then an entire leg is regenerated in the intervening part. This can be conceptually explained by assuming that there is a gradient of positional information along the leg and that the cells obey the "rule of normal neighbors". If a cell locally detects that its neighbor's positional value is not what it normally expects, it attempts to create a new neighbor with the correct positional value. The new neighbor than repeats this process until all cells are satisfied with their neighbors. This conceptually simple principle applies to regeneration in many different organisms, even though the detailed behavior differs significantly. We have recently used this strategy to successfully create self-repairing patterns and structures in Amorphous Computing. Agents develop a notion of "normal" neighbors, use local monitoring to detect the absence of a neighbor, and react by attempting to recreate/recruit a normal neighbor or committing apoptosis [8.10].

4 Conclusions

In this paper I have presented several examples of biologically-inspired local primitives and global patterns for engineering robust collective behavior. Multi-cellular organisms can provide a metaphor for programming multi-agent systems and for thinking about the kinds of robustness we would like to achieve. However, the goal is not to mimic biology, but to use these evolved systems to help design artificial systems that robustly achieve global goals that we specify. In addition to developing a catalog of basic building blocks one also has to develop the global languages and methods for analysis. Also, different types of goals will require different types of local primitives. For example, development and evolution are two very different processes and solve different problems; whereas development may tell you how to build a fly in spite of the environment, evolution may tell you whether a fly is the appropriate for the environment. Social insects may represent yet another dimension. As we build multi-agent systems we will encounter many different types of problems and it will be an interesting challenge to know when which strategy is appropriate.

5 Acknowledgements

This paper discusses many of the projects that were done as part of the Amorphous Computing Project at the Massachusetts Institute of Technology, in Cambridge, MA. The author was supported in part by a grant from NSF on Quantum and Biologically Inspired Computing at MIT.

References

1. Butler et al.: Generic Decentralized Control for a Class of Self-reconfigurable Robots, International Conference on Robotics and Automation (2002)
2. Butera, W.: Programming a Paintable Computer. PhD Thesis, Massachusetts Institute of technology (2001)
3. Payton et al.: Pheromone Robots, Autonomous Robots. vol. 11, no. 3. (2001)
4. Bojinov et al,.: Multiagent Control of Self-reconfigurable Robots. Intl. Conf. on Multi-agent Systems (2000).
5. Abelson et al.: Amorphous Computing. Communications of the ACM, vol. 43, no. 5, (2000)
6. Nagpal,: Programmable Self-assembly Using Biologically-inspired Multi-agent Control,. Proc. of Autonomous Agents and Multiagent Systems (2002)
7. Coore,: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer", PhD Thesis, Massachusetts Institute of Technology (1999)
8. Kondacs, Biologically-inspired Self-assembly of 2D shapes, Using Global-to-local Compilation", Intl. Joint Conference on Artificial Intelligence (2003)
9. Nagpal et al.: Programming Methodology for Biologically-inspired Self-assembly. AAAI Spring Symposium (2003)
10. Clement, Nagpal: Self-Assembly and Self-repairing Topologies. Workshop on Adapatability in Multiagent Systems, Australian Open RoboCup (2003)
11. Wolpert, L.: Principles of Development. Oxford University Press, UK (1998)
12. Lawrence, P.: The Making of a Fly. Blackwell Science, Oxford UK (1992)
13. Slack, J.: From Egg to Embryo. Cambridge University Press, UK (1991)
14. Wolpert, L.: Positional Information and the Spatial Pattern of Cellular Differentiation, Journal of Theoretical Biology, vol. 25, pages 1-47, (1969)
15. Day, Lawrence: Morphogens: Measuring Dimensions, the Regulation of Size and Shape. Development 127, (2000)
16. Lawrence, P.: Morphogens: how big is the big picture?. Nature Cell Biology, vol. 3, (2001)
17. Miller, M., Bassler, B.: Quorum Sensing in Bacteria. Ann. Rev. Microbiology 55:165-99 (2001)
18. Alberts et al.: Molecular Biology of the Cell. Garland Publishers, 4th Edition (2002)