Designing Systems to Design Themselves

Eric Schweikardt
Computational Synthesis Laboratory
Mechanical & Aerospace Engineering
Cornell University
Ithaca, NY 14853 USA
ees68@cornell.edu

Mark D. Gross
Computational Design Laboratory
School of Architecture
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213 USA
mdgross@cmu.edu

CHI 2009, April 4 – April 9, 2009, Boston, MA, USA
ACM 978-1-60558-247-4/08/04.

Abstract
Reconfigurable systems have many benefits over single-purpose machines but there are many obstacles to their widespread implementation. We describe roBlocks, a reconfigurable robotic construction kit, and the Erstwhile Agent, an evolutionary design system and discuss extensions that would enable them to design improvements to themselves, and also their eventual replacements.

Keywords
Evolutionary design, modular robotics.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous. See [3] for help using the ACM Classification system.

Functional Brittleness
Manufactured products are usually intended for a single use. Even though they have many parts in common, refrigerators chill our food and air conditioners chill the air in our rooms. They do only the single task for which they are designed and are not adaptable to new uses. These products are *functionally brittle.* This makes sense; products are designed like this so that they can be made as inexpensively and as efficiently as possible.

Designing for a single usage scenario allows manufacturers to pare away material and redundancy so that the product is optimized for cost, but also only useful for its intended function.

Another possibility is to design with reconfiguration in mind. A reconfigurable “construction kit” of parts has many advantages over a brittle manufactured product including adaptability and easy repair. LEGO, for instance, typically occupies children for far more time than a single-purpose toy. By assembling the parts into a certain configuration, then breaking it down and reassembling a different configuration, kids can effectively have many different toys. Reconfigurable kits are vastly superior to single-purpose products when thinking about longer time scales: instead of adding to the landfill, a kit can be reconfigured for a different use, user, or scenario. How often does one see a LEGO set, compared to a single-purpose toy, in the trash?

Costs

A kit of modules that can be configured into a refrigerator or an air conditioner will necessarily be more expensive than either of the optimized single-use items it is designed to replace. Cost is one important reason that these types of kits are not available, although the staggering number of adults interested in children’s construction kits suggests that there may be a market for “adult” reconfigurable systems, even at great cost. But cost scales will change in the future. As energy and raw materials become more expensive, re-use and reconfigurable products will become more attractive. The problem of cost is also closely related to scale. If kits are designed with a reasonably small number of different modules (that can then be configured to replace several existing single-use products), economies of mass production reduce manufacturing costs significantly. Another reason that such kits are not available is that so far little effort has gone toward engineering and manufacturing them. This is both a consequence of cost, but also because until quite recently the components and manufacturing technologies made it prohibitively difficult to seriously consider this strategy. However, recent shifts in materials and manufacturing are changing the balance of this equation. As it becomes thinkable to build reconfigurable complex engineered systems out of a construction kit, new challenges and opportunities in design arise. We have been considering these in the microcosm of a toy system, roBlocks.

roBlocks

roBlocks is a construction kit that children as young as about eight can use to build robots before learning how to program. The kit is a collection of 40mm cubes with

Figure 1. A simple roBlocks robot.

magnetic connectors that can be easily assembled – any module can connect to any other module. The several different module types include sensors, actuators, and operators. Each module contains a microcontroller, and the behavior of a construction emerges from local interactions between adjacent modules. Behavior is a direct result of topology, so as kids reconfigure modules, they are both assembling the physical robot and at the same time programming it. Our user testing suggests that roBlocks can be an effective tool for thinking about complexity, but that constructions with many modules can quickly become confusing [1]. We have been experimenting with the Erstwhile Agent, an automated design system we built to explore, more systematically, the design space afforded by systems like roBlocks.

Erstwhile Agent

The Erstwhile Agent (EA) is a software system that attempts to design roBlocks constructions that meet requirements specified by a user. It uses evolutionary algorithms (crossover, mutation, and elitism) to generate and refine a population of candidate constructions over a series of generations. Each candidate is evaluated against design requirements in 3D physical simulation and assigned a fitness score. We have seen promising results from the EA when it is directed to design both fast moving and stable constructions, and we have even seen the EA produce robots that are innovative.

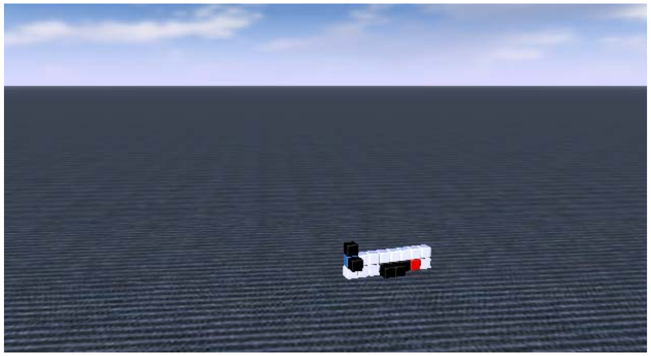

Figure 2. A roBlocks robot designed by the Erstwhile Agent is evaluated in simulation.

Asking More Questions of an Algorithm

Watching the EA design and evaluate different constructions in an attempt to satisfy our requirements is appealing; we can enjoy a quiet cup of coffee while the algorithm does our design work for us. But the question arises: is roBlocks the best kit of parts to meet these design requirements? After all, many of the decisions in the roBlocks kit were somewhat arbitrary; functions are encapsulated into modules in a scheme that makes intuitive sense to us, but not in a scheme that was tested to show the most flexibility in recombination. Fortunately, the EA's architecture supports asking this question: the specifications of the hardware kit are encoded as variables in the framework's code base, so these variables can be mutated and evolved just as the data structures that represent constructions are changed over time.

We might begin by allowing the algorithm to modify the program that runs on each module's microcontroller. Evolving robot control programs has been successful in many applications [2], and changing the code in each module would allow for more complex control structures and reduce the need for chains of operator blocks. We might allow the algorithm to modify the physical geometry of the modules. Does a kit of dodecahedrons afford more meaningful configurations than the cubes of roBlocks? Perhaps a different basic module shape would improve the construction's locomotion capabilities. Maybe the shapes should not be uniform at all. Mapping different functions to different shapes would increase flexibility, but then which modules should be which shape?

These possibilities all represent changes to the kit's *modules*, but changes to the *interface* between modules could lead to even greater benefits. It is the interface, after all, that defines the structure of a kit; the protocol and rules for how modules can connect and interact. Perhaps the evolutionary algorithm would alter the way data passed between modules. roBlocks, for instance, communicate a one-byte scalar data value; an increase in data transmission might lead to an increase in capability. Maybe the EA would modify the physical connector to support multiple data lines, perhaps including a clock or high-voltage connectors.

A Bow Tie for Reconfigurable Robots

We are only beginning to enable the EA to address these questions. But as we do, we ask the system not only to assemble a certain set of modules, but to design the entire kit that it uses as a construction medium. The system, which we first gave the task of assembling components into interesting configurations, is now designing a new kit of components. The algorithm can tell us what are the most interesting kits to build. But an important question emerges: what goal should the evolutionary system have in mind when designing the kit? The fitness functions used by the EA are overly specific; they result in a product that is optimized for a single use, not for optimal reconfiguration.

Recent research in complexity describes a universal *bow-tie* structure that underlies many systems, both biological and engineered. Csete and Doyle describe how in metabolism, for instance, a large number of possible nutrients are transformed (at the bow-tie's knot) into a small number of carriers, which then recombine to synthesize a large number of proteins [3]. Money, they also note, creates a bow tie structure of exchange, allowing many different types of work to relate to many different types of products through the common currency of, well, currency.

Ideal reconfigurable systems should be bow-tie structures as well, funneling a large number of possible materials and functions into a small number of reconfigurable modules that can then be combined to create the largest number of useful configurations. If we can succeed in translating the need for an optimized bow-tie structure into the algorithms and fitness functions necessary for automating design, we may begin to see powerful, adaptable construction kits that make single-purpose machines seem like a tremendous waste.

References

[1] E. Schweikardt and M. D. Gross, "Learning About Complexity with Modular Robots," in *DIGITEL 2008: The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning*, Banff, Canada, 2008.

[2] S. Nolfi and D. Floreano, *Evolutionary Robotics*. Cambridge, MA: MIT Press, 2000.

[3] M. Csete and J. Doyle, "Bow ties, metabolism and disease," *TRENDS in Biotechnology,* vol. 22, pp. 446-450, 2004.