Towards Grammars for Cradle-to-Cradle Design

Douglas H. Fisher
Vanderbilt University
douglas.h.fisher@vanderbilt.edu

Mary Lou Maher
University of Maryland, College Park
marylou.maher@gmail.com

Abstract

Cradle to cradle design (C2C) considers material use and reuse as integral to the composition and specification of the design. Since grammars have been used extensively to model design processes, we consider the possibilities of grammars to model C2C design. Central to our proposal is the idea that products cannot be designed in isolation, but C2C desiderata can only be achieved by explicit design of *families of products* that share material reuse possibilities. Grammars, heuristic search and various forms of machine learning are highlighted as critical in grappling with the complexities of C2C design.

Introduction

Cradle-to-cradle (C2C) design (McDonough & Braungart, 2002) recognizes that nothing short of full recycling of materials with no degradation in material quality is necessary for long-term planet sustainability. C2C advocates looking to the natural world as an ideal model of recycling, where organic materials are continually recycled through processes of decay and growth. They propose design methodology that separates biological cycles and synthetic-material cycles, enabling biological material to be reclaimed through natural processes without synthetic and toxic residue, and enabling the full reuse of synthetic material through technological processes so as to eliminate resource depletion and toxic poisoning. While domain specialists define and elaborate these interacting cycles of material use, there are ample opportunities for Artificial Intelligence techniques to manage the complexities and articulate C2C design processes.

C2C goals suggest the necessity of holistic approaches that design the cycling of material, and include attention to the energy required to maintain the cycles. Reuse of material from one product line can be cycled back to the same product line or another product line. In reality, this already happens through normal recycling (e.g., plastic bottles are recycled into park benches), but the sources and targets of recycling are typically identified after major design decisions, leading to inefficiencies, material loss, and degradation. *By designing C2C **product families**, reuse cycles can be made more efficient, with known and predictable trajectories for reused material*.

An open question is whether C2C design is an entirely different concept from traditional design, or whether it is reasonably modeled as traditional design with additional constraints. Figure 1a first illustrates by the oval that a critical problem in traditional design is that a product is designed in isolation. In contrast, the products shown in the square box of Figure 1b illustrate the concept of a product family, where multiple products are designed within a system of material use and reuse, which flows between product lines. While there may still be materials that come from outside the family and there are materials that are byproducts of the family production, a family design would seek to minimize these and to exploit them in a still larger context. That is, product families are dense subgraphs within a larger network.

Two example domains for C2C design are: low cost housing, and products made of recyclable plastics. Sass (2005) has developed low-cost designs of houses for developing-world communities where a set of designs could be considered as a product family. While not concerned with C2C explicitly, a large collection of house designs used in this project illustrates the importance of product-family design over (simply) many individual house designs to minimize overall waste. A second domain, concerned with C2C explicitly, is the Preserve product lines (www.preserveproducts.com). Whereas the Sass project is a product family of same-type products (i.e., low-cost houses), the Preserve line is heterogeneous, containing many different kinds of products.

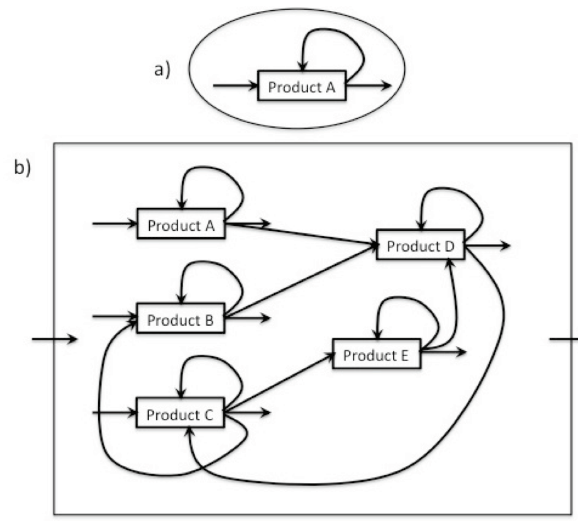

Figure 1: Traditional design vs C2C design.

Design Grammars

When considering the role of AI in sustainable design, we start with grammars as a formal language that can describe, generate, and/or check for C2C design. By starting with a grammar as descriptive of C2C design, we begin to formalize the principles of C2C design as a language, which can also be used to generate example C2C designs. Our focus is on the design of product families and the subordinate designs of individual products, to include lifecycle concerns.

We assume that the reader understands the basics of grammars, so we only review terminology. A grammar is a finite specification of a possibly infinite set of strings (or language) – in this case, each terminal string (or henceforth, simply *string*) is a sequence of *terminal symbols* or members of an alphabet; for now, each string represents a product design, but in subsequent sections a string will correspond to a product family. A string is a technical term that refers to the terminal symbols and may include symbols that separate one product from another. A grammar is specified by a set of terminal symbols, a set of *non-terminal symbols*, and a set of *productions* (aka rewrite rules) of the form $\alpha \rightarrow \beta$, where α and β are collections of terminal and non-terminal symbols. In the case of a *context–free grammar*, α is always a single non-terminal symbol and β can be any finite sequence of symbols. A *derivation* is defined by a sequence of transitions from a specially-designated *start symbol*, which is a member of the non-terminal symbols, to a sequence of only terminal symbols. A *sentential form* is an intermediate set of symbols obtained through a sequence of zero or more productions, beginning with the start symbol.

There is a rich history of using grammars to describe design knowledge generally in AI. In particular, shape grammars (Stiny 2008, 1980; Stiny and Gips 1972; Knight, 1998) are a variant on context free grammars (CFG) that include shape rewrite rules that can be applied in a step-by-step manner to generate a set of designs. An example of a grammar for designing virtual worlds is described in Gu and Maher (2004). Shape grammars have developed over the years to include many extensions, such as parametric grammars, color grammars, description grammars, structure grammars, parallel grammars, and so on, to address different aspects of designs.

Knight (1999) proposes at least two approaches to developing a grammar-based system that produces designs meeting specified goals under constraints. A strictly *generative approach* incorporates knowledge into grammar rules so that all generated designs satisfy the constraints. In a *generate-and-test approach* the grammar is under-constrained, and the initially generated designs are examined using an evaluation function for those that satisfy the constraints. Naturally, there need not be a hard schism between strictly generative and generate-and-test methods – even if all hard constraints can be encoded in the grammar, there will likely still be preferences that are best captured in an evaluation function. In general, the space of grammar-based design methods extends with constraint-based and optimization methods, and hybrids of them.

We propose to adapt grammar-based approaches to C2C design by (a) defining grammars for languages of product family designs rather than languages of individual product designs, (b) defining constraints and preference functions for ranking product family designs by their amenability to C2C principles (manufacture, use, disassembly and reuse), and (c) taking steps through both first-principle analysis and machine learning techniques towards strong(er) grammars with derivations to strings that satisfy many of the hard constraints and perhaps even some preferences.

Grammars for Product Design

We initially make some important simplifying assumptions. Foremost, we assume that a product is only represented by its material components. Each *terminal symbol* represents 'one unit' of material, so the string 'aabbbc' represents a product with two units of material 'a', three units of 'b', and one of 'c'. With reference to Figure 1, this assumption would translate to replacing each label (e.g., 'Product A') by a string (e.g., 'aabbbc').

Continuing, we assume that each non-terminal symbol defines a (sub)language of product components -- the start symbol of the grammar is a special case, defining the language of products. Putting this together with our assumptions about the representation of a product, we might define a grammar for toothbrushes, with start symbol T and productions:

T → Handle Head
Handle → Grip Back
Head → Base Bristles
Grip → aa
Grip → ab
Grip → aba
Back → bb
Back → b
Base → b
Bristles → c

One (*leftmost*) derivation in this grammar is

```
T ➔ Handle Head            (using T → Handle Head)
   ➔ Grip Back Head        (using Handle → Grip Back)
   ➔ aa Back Head          (using Grip → aa)
   ➔ aa bb Head            (using Back → bb)
   ➔ aa bb Base Bristles   (using Head → Base Bristles)
   ➔ aa bb b Bristles      (using Base → b)
   ➔ aa bb b c             (using Bristles → c)
```

Note that 'ababbc' is derivable, as well as 'aabbbc', and whether these were regarded as 'equivalent' or not would be judged by a preference function.

The grammar above is context free, but *context sensitive* transitions might also be introduced. For example, if it was desirable that the total amount of material derived from Back was sensitive to the amount derived from Grip, then productions such as

aba Back → aba b
aa Back → aa bb
ab Back → ab bb

might be used instead of the context-free Back rules (here, we ignore the possible reorganization of a grammar so as to eliminate context sensitive rules, resulting in another context free grammar).

In design grammars generally, there is attention to representing connections between components in the grammar and in derivations. We have thus far not addressed issues of component connectedness, but rather viewed a product as simply a bag of materials.

While these assumptions are very limiting, they still allow progress on developing a framework for C2C design grammars and methods for their development. For example, we imagine a possibility in which standards for product grammars are established, and automated means are developed that find patterns across these product grammars in search of desirable product families. The base grammars over which this search occurs can be bootstrapped from existing products. For each product, a grammar can be induced for which that product is but one string. By analyzing the composite structure of the product, non-terminal symbols can be introduced that reflect an aspect of the composition, with at least one production enabling eventual derivation of the product, but with other productions leading to other, functionally equivalent or very similar products. These new strings might, for example, be composed of different materials than the 'seed' product, where functionally and physically related materials would be encoded as background knowledge. This strategy would be an analytic form of generalization from very limited data that was akin to *explanation-based learning* (DeJong and Mooney, 1986; Yoo and Fisher, 1991) and *experimental goal regression* (Porter and Kibler, 1986).

Weak Grammars for Product Family Design

A grammar for a language of product families can be constructed from other grammars for product languages. For example, suppose that we have a grammar that defines a set (language, class) of toothbrushes, any one of which is a possible product/artifact; the start symbol for this grammar is T. Suppose that we also have a grammar defining a language of hairbrushes, with start symbol H. A language of a product family designs that include hairbrush and toothbrush products is defined by

S → [T] [H]

Assuming that derivations starting with T and H produce exactly one (non-NIL) terminal product each, the grammar with start symbol S defines a language of product pairs, of one toothbrush and one hairbrush per terminal product family. Delimiters such as '[' and ']' may be retained throughout a derivation and in the final terminal strings, but such boundaries might also be removed/ignored, allowing product sentential forms to be interleaved at various points in a product family derivation.

Instead of the production above, we could introduce new productions

S → SS
S → [T]
S → [H]

Together with the productions given in the constituent grammars, this new grammar defines a language of product families of hairbrushes and toothbrushes with at least one toothbrush or at least one hairbrush. It allows product families with duplicate products (e.g., 2 instances of a particular toothbrush t_i and 3 instances of a particular hairbrush h_k. The language also allows for families that contain only toothbrushes or only hairbrushes.

In the case of duplicates, the cardinality of differing product (duplicates) could be interpreted as a proportionality specification (e.g., in a product family there is a 2/3 the number of t_i as there are of h_k), which could be relevant for resource planning and the like.

A grammar for a product family defines an infinite set of product combinations and duplicate cardinalities. The size of the terminal strings (aka families) can be bound to a finite value, which also bounds the precision of product proportionalities (e.g., bounding family size to 5 or less allows 2/3 of t_i to h_k but disallows an expression of 4/5 of t_i to h_k); the size of the language may remain infinite, even with such a bound on family size, if any of the constituent grammars define infinite languages.

Having defined a product family grammar as just described, it is obvious that this too could be a constituent grammar in a still 'larger' grammar and that the product family could be a terminal in a still larger and richly organized language of product collectives. Nonetheless, our examples, for now, will correspond to the intuition that we have grammars for 'individual products', from which we can construct grammars for product families. In building composite grammars from constituent grammars, we assume that terminal and non-terminal symbols that have the same names in different constituent grammars are the same materials in the case of terminals and represent the same component sets in the case of non-terminals. Thus, care must be taken to standardize apart symbols across constituent grammars if such symbols represent different sets. *If constituent grammars share terminal and non-terminal symbols, this presents opportunities for reorganization of the composite grammar to more explicitly represent possible interactions between product derivations within a product family*.

We return to Figure 1b at this point, to illustrate some of our assumptions. Figure 1b, *without the arcs*, and with product labels in the boxes replaced by strings of material units (e.g., 'aabbbc' for Product A, 'addeeef' for Product B, 'bcdddee' for Product D) is an instance of a product family. Figure 2 explicitly shows such a structure, which would be a string resulting from a derivation in a product family grammar.

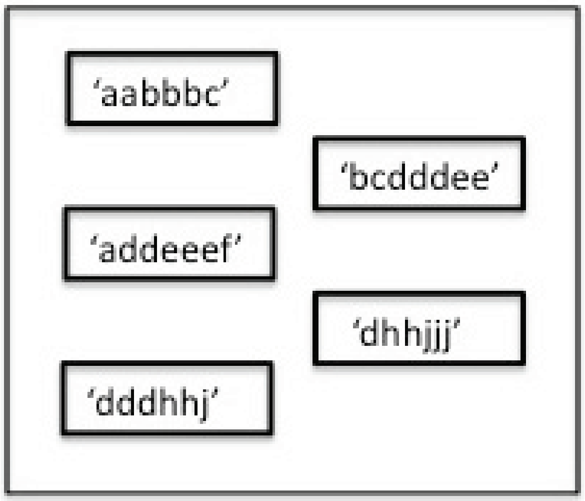

Figure 2: A product family under simplifying assumptions

Constraints and Preferences

In the previous two sections we have seen that grammars can be constructed for generating and describing both product designs and product family designs. That is, grammars define a search space. Now we turn our attention to constraints and preferences that can be used to evaluate product family designs. As such, they can also be used to guide the search for product family designs. Hard constraints define whether designs are acceptable or not. Preferences define the desirability of some designs over others.

We begin discussion of constraints and preferences by considering families composed of existing product designs only. This is an activity that tenacious recyclers might perform; looking as best they can for relationships between existing products – grouping products designs together in a way that maximizes recycled material with minimal cost (e.g., energy). Given our assumptions, each and every product design would be represented as a vector of materials and essentially an integer amount of each material. Given the huge space of all such represented product designs (e.g., 'abbbcc', aaabddee', bdefff', …), we could apply unsupervised learning methods of clustering (Fisher, 2002) to group product designs, optimizing a tradeoff between maximizing within-family overlap in material shared among product designs and minimizing between-family overlap in shared material. Though the product family representation we are currently considering does not include arcs showing material flow trajectories, the intent of maximizing within and minimizing between family material overlap can be mapped to maximizing recycled content at minimal cost.

So-called *constrained clustering* (Basu et al, 2008) allows analysts to impose constraints that require that some objects (product designs) be grouped together, while requiring that other products not be. For example, analysts might know that certain materials are highly toxic in a way that cannot be easily controlled during reclamation, therefore indicating that such materials be separated entirely from product designs that can undergo routine reclamation.

Other constraints and preferences that can be used to design product families as clusters of existing products could be based on expected lifetimes of existing products, generating families in which time spans can be coordinated to better insure material replenishment, or methods (and their costs) of returning products to be recycled at the end of life – we would want to favor products that can be packaged together for example, thereby minimizing transportation energy. These preferences and constraints all arise from externalities, and absent an ability to explicitly represent certain constraint and preference knowledge in the grammar, it would remain in the evaluation function.

Preferences and constraints that are useful for evaluating families of existing products, are also presumably useful for guiding the search for entirely new product designs, and in our case this search is embedded in a search of product families. Moreover and importantly, C2C constraints and preferences can be encoded into grammar rules or preference functions so as to exclude new designs that violate C2C principles. For example, McDonough and Braungart (2002) talk about 'monstrous hybrids', which are products that combine synthetic and organic material to the detriment of recycling either effectively. To a large extent we imagine that monstrous hybrids can be excluded by suitable crafting of production rules of constituent product grammars.

More generally, in addition to preferences and constraints applied to a string (i.e., product or product family), constraints and preferences can be functions of the string derivations, represented as a sequence or as a parse tree. While a design-grammar derivation need not map directly to manufacturing and/or disassembly steps of the terminal product of the derivation, it is a natural bias to consider the grammar as constructive. Indeed, if we assume a mapping between grammar derivations and steps of manufacturing, as well as disassembly procedures, then costs associated with these activities can be reflected in an augmented grammar. Costs could correspond to represent energy; they could also represent materials used as positive costs and reclaimed as negative costs, or rewards.

For simplicity, we restrict ourselves to the case where disassembly is the inverse of construction, and production is associated with a forward cost (e.g., representing a constructive step) and a backward cost (e.g., representing a disassembly step). In general, the backward and forward costs may be quite different. Under these assumptions, a preference function for designs is informed by the derivations for those designs, and in particular, costs associated with those derivations. Because costs are associated with individual productions, sentential forms at intermediate points can be evaluated, and used to guide a heuristic search through the space of possible derivations. In the simplest case, the sum of forward and backward derivation costs would be a factor in evaluating proposed product families.

Challenges Ahead

From constituent grammars for product languages, each augmented with production costs, we can define cost augmented grammars for product family languages. But in the weak approach outlined in the previous section, the evaluative cost of a product family would necessarily be a simple summation over the total forward and backward costs across the products in a family. There are substantive challenges in defining grammars for product family designs that more tightly couple the constituent grammars for individual product designs. Put another way, we want grammars that capture the rich interactions between product designs suggested by Figure 1b.

An approach that we are considering, with machine learning at the heart of it, is to transition from weak to strong methods. Recall that a weak generate-and-test method is one in which the grammar is under-constrained, defining a language that is a large superset of the C2C designs. Designs that are found/generated during search are subjected to an evaluation function that encodes constraints and preferences; constraints eliminate those that violate principles of C2C design, and preferences serve to rank the surviving strings.

In contrast, the strongest methods are those in which a grammar derives only strings that satisfy the principles of C2C design; these are the strictly generative approaches described by Knight (1998). Designs stemming from strong methods would likely still be evaluated by preferences for choosing among C2C designs.

As a rule it is much easier to formalize a weak method than a strong method for complicated domains. In fact, as we have mentioned early on, methods span a continuum between the weakest and strongest methods; machine learning can traverse this continuum. Thus, a cost-effective strategy for obtaining methods at the strong end of the spectrum is to first specify a weak method, to include an under-constrained grammar plus preferences and constraints. The weak grammar defines a search space, and machine learning of search control knowledge, which is a form of *speedup learning* (Yoo and Fisher, 1991), transforms the system to the strong method that is more desired.

We believe that this automated learning approach, which backs up constraints and preferences into the generator is a promising future component in a grammar based framework for C2C design.

Acknowledgments. We acknowledge the information available on the Preserve.com web site about their C2C products and a discussion with Christie Lee at Preserve about their approach to designing products with reusable plastics.

References

Basu, S., Davidson, I., and Wagstaff, K. 2008. *Constrained Clustering*. Chapman & Hall.

DeJong, G. and Mooney, R. 1986. Explanation-based Learning: An Alternative View. *Machine Learning*. **1**:145-176

Fisher, D. H. 2002. Conceptual Clustering. In W. Klosgen and J. Zytkow (eds) *Handbook of Data Mining and Knowledge Discovery*. Oxford, UK: Oxford University Press, pp. 388-396.

Gu, N. and Maher, M.L. 2004. A Grammar for the Dynamic Design of Virtual Architecture Using Rational Agents, *International Journal of Architectural Computing*, **4**(1): 489-501.

Knight, T.W. 1998. Shape Grammars. *Environment and Planning B: Planning and Design* Anniversary Issue, pp. 86-91.

McDonough W and Braungart M. 2002. *Cradle to Cradle*, New York: North Point Press.

Porter, B. and Kibler, D. 1986. Experimental Goal Regression: A Method of Learning Problem Solving Heuristics. *Machine Learning* **1**: 249-286.

Sass L. 2005. A Production System for Design and Construction with Digital Fabrication, MIT; Cambridge, MA/USA:http://ddf.mit.edu/projects/CABIN/cabin_mit_2005.pdf

Stiny, G. 2008. *Shape: Talking about Seeing and Doing*, MIT Press.

Stiny G. 1980. Introduction to Shape Grammars, *Environment and Planning B* **7**:343-351.

Stiny G. and Gips J. 1972. Shape Grammars and the Generative Specification of Painting and Sculpture, in C.V. Freiman (ed), *Proceedings of Information Processing 71*, North Holland, Amsterdam, pp. 1460-1465.

Wackernagel M. and Rees W. 1996. *Our Ecological Footprint: Reducing Human Impact on the Earth,* New Society Publishers.

Yoo, J., & Fisher, D. 1991. Concept formation over explanations and problem-solving experiences. *Proceedings of the International Joint Conference on Artificial Intelligence*, Sydney, Australia: Morgan Kaufmann, 630--636.