Concept Variation and Example Generation: Some Preliminary Thoughts Working Note 23

Peter Clark
Knowledge Systems
Boeing Maths and Computing Technology
peter.e.clark@boeing.com

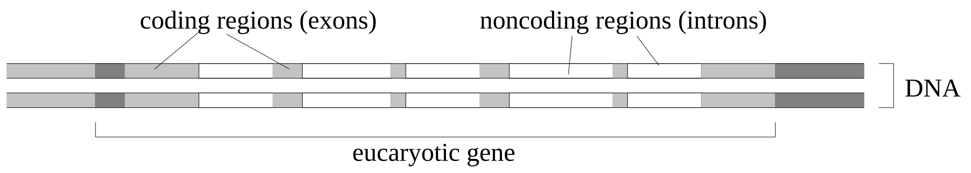
March 2001

Abstract

This little working note presents some brief thoughts on handling variation among members of a concept, and generating examples of that concept. This note only sketches some very preliminary thoughts, to (hopefully) be elaborated in later notes.

1 Motivation

As part of DARPA's RKF project, we have needed to represent knowledge about (among other things) DNA. For the purposes of this working note, we concern ourselves only with the "regional viewpoint" of DNA, i.e., viewing DNA in terms of genes, introns, exons, coding regions etc. (as opposed to other viewpoints such as chemical, spatial, or structural). We will refer to the description of DNA in terms of the sequence of regions it contains as the DNA's "regional structure". This can be thought of as a one-dimensional "map" of the DNA. An example of such a "map" is shown below, taken from [Alberts et al., 1998], Figure 7-13, p220.



One challenge for building such a representation is that there is no unique "regional structure": Different DNA strands have different numbers of genes, introns, etc., at varying positions. Instead, the relevant knowledge (mainly) consists of *constraints* on structure. For example, note the many constraints expressed in this paragraph:

> "Most eucaryotic genes have their coding sequences interrupted by noncoding sequences, called introns. The scattered pieces of coding sequence, called exons, are usually shorter than the introns, and the coding portion of a gene is often only a small fraction of the total length of the gene. Most introns range in length from about 80 nucleotides to 10,000 nucleotides, although even longer introns exist." [Alberts et al., 1998], p220.

We could (with some work) express these constraints in logic. However, for some reasoning tasks, in particular simulation, we also require an *example* of the DNA structure to work with. For example, to simulate (and thus answer questions about) how an intron is removed by RNA splicing, the reasoning engine needs to first create some initial situation (containing, among other things, a piece of DNA with some introns and exons placed in some specific sequence), and then run the simulation forward from there.

We could imagine some reasoning machinery which could, given some appropriate constraints on the structure of an object, be able to generate examples of that object (perhaps even using probability distributions, so that the probability of generation corresponds to the probability of occurrence in the world). We will call representations that support example generation **generative representations** (although strictly the generative capability is a property of both the representation and the example generator machinery together). Note that not all representations are generative, for example if too much information is missing, then the example generator may be unable to perform its task (e.g., the example generator may fail if no constraints whatsoever were specified on the length of a DNA strand).

Or, as an alternative, we could hand-build a "representative" example of the concept in question (e.g., a particular strand of DNA, with normal-sized introns and exons placed in a typical order). Then, to generate variants, we would "mutate" this hand-build example by modifying some feature of it, e.g., the number or spacing of the introns. We call such representations **mutable representations**, although again the representation's mutability is really a property of both it and the mutation machinery together. Mutating a representation is often more complex than simply adding or deleting a fact – there may be additional changes needed to make sure that the new representation is still "sensible", i.e., consistent with the KB. For example, one way of mutating a DNA representation would be (conceptually) to remove an intron, but implementing this as an operation which simply deleted all assertions mentioning that intron would be inadequate: at best, the new representation would now denote two disconnected strands of DNA (broken at the point where the intron was), at worst it would be incoherent (i.e., violate some KB constraints). Rather, the "mutation" operator would need to be a bit more intelligent, and know that the now exposed ends of the broken DNA should be reconnected in the mutated representation. In addition, mutating an example of a concept like this would need to be guided, to ensure that the mutated example still satisfied the basic constraints on that concept.

2 Summary and Related Work

The specific observation here is that, for some physical objects, their physical structure may vary, and thus we need ways of both specifying the range of that variation, and generating examples of that object for reasoning purposes. We could do this either generatively from a set of constraints, or "mutably" from a pre-build example of that concept. More generally, such variation may occur along any dimension, not just physical structure.

These ideas are not particularly new, and this working note is mainly reminder rather than insight. Some related work comes to mind, which we now list.

First, work in AI in Design (e.g., [Brown, 2001]) is closely related. The design task can be specified as, given a space of possible designs, find the "best" according to some evaluation function. Clearly this task requires generating examples from constraints, and then analyzing them.

Second, work in case-based/examplar-based reasoning is related to the “mutable representation” approach, where a concept with variable properties is represented by a set of examples, and a similarity metric assesses how “close” a new example is to one of these. The similarity metric can be viewed as assessing how much of a “mutation” is required to convert the nearest stored example into the new example.

Third, numeric Monte Carlo simulation techniques are also related. Given some constraints, this appraoch involves generating examples randomly, but according to a probability distribution reflecting their likelihood of occurring in the real world. Then each example is analyzed and the results compiled into some summary.

Fourth, “mutable representations” have a flavor of working with prototypes, where new examples are described as modifications of old. This approach was applied in Garnet [Carnegie Mellon University, 2001], a user interface development environment based on prototypes (rather than classes), in which in which new prototypes are defined in terms of the *modifications* applied to old prototypes.

Fifth, mutating examples (and then patching the mutation so it becomes consistent again with the KB) is reminiscent of other “tweak-then-patch” systems in AI, such as HACKER [Sussman, 1975], which would plan by tweaking some incomplete plan, while some “demons” would watch over looking for problems thus introduced, and if triggered suggest ways of fixing the problems.

As a closing comment, the traditional reasoning process of “Here’s a concept, deduce some facts” seems a rather inadequate characterization of intelligence. Exploring a space of actualities, as described here, adds an extra dimension to this type of reasoning, and may provide some scope for more interesting reasoning by the machine.

References

[Alberts et al., 1998] Alberts, B., Bray, D., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (1998). *Essential Cell Biology.* Garland, NY.

[Brown, 2001] Brown, D. (2001). The AI in design webliography. http://www.cs.wpi.edu/Research/aidg/AIinD-hotlist.html.

[Carnegie Mellon University, 2001] Carnegie Mellon University (2001). The garnet project home page. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/garnet/www/garnet-home.html.

[Sussman, 1975] Sussman, G. J. (1975). *A Computer Model of Skill Acquisition.* Elsevier, NY. (also published as MIT AI TR-197 report, 1973).