Efficiency, Economy, Optimality, Simplicity, Minimality, Search, Mapping, Function, State
An Attempt at Clarifying Definitions

Richard L. Lewis
Department of Psychology
University of Michigan

March 9, 2010

1 Background

Current theorizing in the Minimalist framework appeals to notions of optimality, economy, efficiency, simplicity, minimal search etc. Most of these terms have well-defined meanings/usages in computer science and mathematics, and I think that making these explicit might help to achieve further precision in specifying particular claims in Minimalism. So here's one take on such a glossary.

2 Definitions

efficiency Asserting that something is more or less *efficient* than some other thing requires identifying a *resource* that is being used or consumed by those things. *Computational efficiency* is usually concerned about the resources of *time* and *space (memory)*. Thus, computer scientists will formally analyse *algorithms* to characterize their time and space efficiency. This is typically done by characterizing how long some algorithm will take to run as a function of some measure of the size of its input. For example, I might have an algorithm that sorts lists of length n, or parse sentences of length n, and I am interested in how long that algorithm will run as a function of n—often called its "time complexity". It might take "order n^2" or $O(n^2)$, which means it takes no more than $K + Cn^2$ steps to complete. There are many different ways this is done: one can ignore the constants K and C, or be worried about them, one can be concerned about "worst-case upper bounds", or be concerned about average expected run times, etc.[1] Similarly for space/memory. Then one can inquire about the "hardness" of *problems* (like sort-

[1] Note that these analyses do not actually predict time in seconds—that is a "constant multiplicative factor" that goes into C that depends on your particular hardware.

ing)—and prove claims about the possible existence (or nonexistence) of algorithms of certain time or space complexities to solve those problems.

minimal I take *minimal* to simply mean that I have some space or set of objects X, a function $F(x)$ that assigns some quantity to each $x \in X$, and I am interested in finding the x_{min} such that for all x, $F(x_{min})$ is less than or equal to $F(x)$, or:

$$\forall x \in X, F(x_{min}) \leq F(x)$$

There is a simple notation for this called "arg min"—finding the argument that minimizes some function.

$$x_{min} = \arg\min_{x \in X} F(x)$$

Perhaps a lot of notation to say give me the smallest apple. But if my function F is about computational cost—either space or time—then I have a framework for asking about the most efficient (i.e. minimal cost) object x in my space of possible objects X.

economy I take *economy* to mean *minimality*.

optimality Talking about an object x as *optimal* requires two things: specifying a space X of possible x's and a function F that is to be maximized or minimized. So, no different than minimality above. If F is to be minimized, it can be thought of as a cost function. If F is to be maximized, it can be thought of as a goodness function. If F is a goodness function, then

$$x_{optimal} = \arg\max_{x \in X} F(x)$$

If you have not specified both F and X, then you do not have an optimality framework.

simplicity It is important to distinguish simplicity of a *theory* from simplicity of the *derivations*, *implications*, or *accounts* generated by the theory. In both cases, one can *formalize a simplicity metric* that is predicated over representations of the theory or its derivations (one such metric is *description length*).

For example, I can have a theory of the natural numbers that I write down as the *Peano axioms*. I claim it is a *simple* theory because it has a small number of axioms (< 10). But the *derivations* of particular arithmetic facts that it produces may *not* be simple, as measured by, say, length of the derivation, when compared to derivations that I might be able to produce from more *complex* (measured here by axiom count) theories.

Similarly, one can ask about the simplicity of a theory of *grammar* independently of the simplicity of either the *derivations* or *representations* that it produces. In Minimalist theory, both kinds of simplicity are apparently at work:

1. Minimalism is seeking *simplicity in theory* as a methodological aim, a goal that it shares with all scientific inquiry. This kind of simplicity need *not*, in principle, care about the simplicity/complexity of (representations of) implications or derivations

that result from this theory. E.g., when one complains about *complex syntactic trees* with lots of nodes, etc., one is complaining about the simplicity (or lack thereof) of the *implications* of some theory—which itself might be (maximally) simple.

This distinction is a real issue in current syntactic theory; see, e.g., the criticism of Culicover and Jackendoff (2005) in stating the "simpler syntax hypothesis": as I understand it, they are concerned first with simplicity of resulting representations.

2. But Minimalism is also exploring *simplicity in derivations* in the technical sense of *minimal* or *economical* above—making such assertions an explicit part of the theory. This is *not* the same as the methodological aims of theoretical simplicity... though if we are lucky, they will be aligned (because both aims will be moving the theory toward making veridical claims about human mental states).

relation Relations are just subsets of cross-products of sets. For example if you have sets S and T, then a binary relation ρ must be some particular subset of $S \times T$. No other constraints. The cross-product set $S \times T$ is the set of all possible ordered pairs (s, t) such that $s \in S$ and $t \in T$.

function Functions are typically taken as special kinds of relations between two sets. If S and T are sets, then a function $f : S \rightarrow T$ is a subset of $S \times T$ where each member of S appears exactly once as the first component of the ordered pair.[2]

mapping "Mapping" typically means the same thing as "function"... but the definitions can vary so be careful.

state Usually "state" has a well-defined meaning in a computational or dynamical system: it is the characterization of the system *at some point in time* such that its future trajectory can be determined (perhaps only probabilistically) as a function of that state.

Very often in science *differential equations* are used to express such state-dependent dynamics over time, characterizing system states in terms of a set of "state variables" and characterizing the dynamics via dynamics equations (differential equations) that describe how those states evolve over time. Simon (1992) used the term *difference equations* to denote a more general class that includes descriptions of how computational systems evolve over time as well—and such descriptions naturally take the form of algorithms or programs, not differential equations. But the idea is the same: a state evolving over time according to some posited laws.

Trajectories of most systems of interest to cognitive scientists (like parsers) cannot be determined analytically and are often of sufficient complexity (doesn't take much) to warrant *computational simulation* as the primary method of drawing out the implications of theories of such systems.

[2]Here's a counter-intuitive result from computational theory. One can consider the space of all possible functions over, say, the integers. And one can consider the space of all possible programs for computing functions. In fact, one can *enumerate* all those programs (it is an infinite set, but all the programs can be placed in one-to-one correspondence with the integers, program1, program2, etc.). Sadly, the number of possible functions of integers *cannot* be placed into one-to-one correspondence with the integers... even though there are infinitely many of both of them, in some well-defined mathematical sense there are "more" functions than there are integers, hence more possible functions than possible computational programs. Darn: this means that there must be some *non-computable functions* (there are lots of famous examples).

search Most (all?) interesting or hard computational problems are *search problems*: there is some (potentially large) space of possible objects of interest, and the goal is to find one/some of those objects that satisfy certain properties. *Search algorithms* can be characterized in terms of their *efficiency* (see above) as a function of size of the space. Having a well-defined search problem requires specifying a space and specifying features of candidate solutions. For hard problems the search space is never pre-constructed but constructed on the fly in the course of exploration. Specifying how that space is incrementally constructed and explored is what specifying a search algorithm is about.

Basically everything in artificial intelligence, computer science, biology, cognitive science, life. . . is a search problem of some kind.

References

Chomsky, N. (1998). Some observations on economy in generative grammar. In P. Barbosa, D. Fox, P. Hagstrom, M. McGinnish, & D. Pesetsky (Eds.), *Is the best good enough? optimality and competition in syntax* (pp. 115–127). Cambridge, MA: MIT Press.

Culicover, P. W., & Jackendoff, R. (2005). *Simpler syntax*. Oxford: Oxford University Press.

Simon, H. A. (1992). What is an explanation of behavior? *Psychological Science*, *3*, 150–161.