Vagueness, Rationality and Undecidability: A Theory of Why There is Vagueness

Mark A. Changizi[‡]
Department of Computer Science
National University of Ireland
Cork, Ireland
changizi@cs.ucc.ie

1 Introduction

Vagueness is not undecidability, but undecidability does enter into an explanation of why there is vagueness. My theory, called the Undecidability Theory of Vagueness, explains vagueness largely as a result of the fact that we are computationally bound.[1] Vagueness is not due to any particularly human weakness, but due to a weakness that any computationally bound agent possesses; even HAL from *2001: A Space Odyssey* will probably experience vagueness. Furthermore, I will argue that vagueness is good for you. I will do so by showing that if you were a computationally bound, rational alien agent given the task of figuring out what our natural language predicates mean, you would very probably end up with vagueness. That is, unless two highly plausible hypotheses are false, it would be rational for you—i.e., in your best interest—to choose concepts in such a way that they are vague. Given that you are computationally bound, avoiding vagueness brings in greater costs than accepting it.

[*]This paper appeared in *Synthese* 120, pp. 345–374, 1999.

[†]Current address of correspondence can be obtained at http://www.changizi.com, or by contacting changizi@changizi.com

[1]Very early ideas of mine along these lines appeared in Changizi ([Cha95]). A paper on my theory was presented at the 1998 vagueness conference in Bled, Slovenia (Changizi, [Cha99b]), and at the 1998 Irish Conference on Formal Methods (Changizi, [Cha99a]). The latter concentrates on logics and semantics of vagueness motivated by the Undecidability Theory of Vagueness.

It is useful here in the introduction to present a brief, preliminary run-through of my theory and how it explains vagueness. The "vagueness is good for you" arguments will not appear in this introduction. I will take you to be my example natural language user.

(1) The first hypothesis is the *Church-Bound Hypothesis*, and it states that you can compute no more and no less than what a computer can compute. (2) The second hypothesis is the *Programs-in-Head Hypothesis*, and it states that what natural language predicates extensionally mean to you is determined by programs in your head. For example, an object is a dog to you if and only if your program in the head for 'dog' outputs YES when the (name of the) object is input into the program. It is much less plausible that many scientific, mathematical and technical predicates get their meaning to you via programs in the head, and this difference is what prevents my theory from concluding that such predicates, many which are not vague, are vague. (3) The third and last hypothesis is the *Any-Algorithm Hypothesis*, and it states that you allow yourself the choice of any algorithm when choosing programs in the head for determining your natural language predicate meanings. (An *algorithm* is a program that halts on every input; programs sometimes do not halt on some inputs.) Informally and crudely, the three hypotheses are that (1) you are a computer, (2) you have programs in the head determining what natural language predicates mean to you, and (3) you allow yourself the fullest range of possible meanings for natural language predicates.

If these three hypotheses are true, what follows? The Programs-in-Head Hypothesis says you choose programs to determine your meanings of natural language predicates. The Any-Algorithm Hypothesis says that the set of programs from which you are choosing is a superset of the set of all algorithms. But here is the catch: one of the basic undecidability results implies that any such set of programs is undecidable. (A set is *decidable* if and only if there is program that outputs YES whenever input with an object from the set and NO whenever input with an object not in the set.) Because of the Church-Bound Hypothesis, this undecidability is a difficulty for you: in choosing from the set of programs you cannot always obtain algorithms. In fact, because picking algorithms is computationally more difficult than picking non-algorithms, you will "usually" pick non-algorithms; "most" of your programs determining the meanings of natural language predicates will not be algorithms. So, in an attempt to acquire a meaning for 'dog' via a program in the head that outputs YES when something is a dog to you and NO when something is not a dog to you, *there will be objects on which your*

program does not halt at all. This does not mean that you will actually run into an infinite loop; it just means that you will eventually give up when running the program on such inputs.

What does this have to do with vagueness? Consider the set of objects for which the program for 'dog' does not halt. For any object in this set the program will neither say YES nor NO; the object will neither be a dog to you nor not a dog to you. My first theoretical claim is that this is the set of borderline cases for the predicate. What about higher-order vagueness, the phenomenon that the boundaries of the borderline region are vague? Consider trying to determine exactly which objects are part of the borderline region. To determine that some object is in the borderline region of 'dog' requires that you determine that your program for 'dog' does not halt on that object. But now we have another catch: possibly the most well-known undecidability result is the "halting problem," which says that whether or not a program will halt on a given input is undecidable. This undecidability is a difficulty for you because of the Church-Bound Hypothesis: objects in the borderline region are generally difficult to determine as such, and where the boundaries of the borderline region are is not generally possible for you to determine. Imagine moving from 'dog' cases to borderline cases. Your program for 'dog' will no longer output YES, and will, in fact, never halt; but you will not know it will never halt. You will be unable to see the boundary. My second theoretical claim is that this inability is the phenomenon of higher-order vagueness. This claim appears to have some empirical support, as discussed in Subsection 3.3. Here is a simple representation of the behaviour of your program for 'dog', where 'Y' denotes YES, 'N' denotes NO, and '↑' denotes "does not halt".

Y Y Y Y Y Y ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ N N N N N N N

{ - - 'dog' - - }{- - borderline - - }{- - 'not dog' - - }

So, the three hypotheses entail that for "most" of your natural language predicate meanings there are objects for which your program for that predicate does not halt. Add to this my two theoretical claims just mentioned and it follows that "most" natural language predicates are vague. That is the Undecidability Theory of Vagueness in a nutshell. Now to develop and defend it in more detail. Section 2 presents the Undecidability Theory and Section 3 discusses how it explains vagueness.

2 Theory

In this section I discuss the three hypotheses comprising the Undecidability Theory of Vagueness and show how they lead to what I call the "Thesis," which is central to the Undecidability Theory of Vagueness's characterization of vagueness. Here is the Thesis, followed by an explanation of the terminology used.

Thesis: For "most" natural language predicates R

1. your interpretation of R is determined by a program in the head that is capable of semideciding but not deciding it,

2. your interpretation of 'not R' is determined by a program in your head that is capable of semideciding but not deciding it, and

3. there are objects neither in your interpretation of R nor in your interpretation of 'not R'.

I will explain the scare quotes around 'most' later. By a "program in the head" I mean the method used by you to determine whether or not a given object is in your interpretation of R. One may usefully and informally think of the program as your intension of the predicate R. The "interpretation of R ('not R') determined by a program" is the set of objects on which the program in the head for R ('not R') outputs YES. A set is *decidable* by a program P if and only if for all x, P on input x outputs YES if x is in the set, and outputs NO otherwise. A set is *semidecidable* by a program P if and only if for all x, P on input x outputs YES exactly when it is in the set; if x is not in the set P may well not halt at all, though. Do not confuse the notion of a set being semidecidable but not decidable by the program for it with the notion of an underdefined or incompletely specified set. The former, which appears in my theory, is a precise set that happens to be computationally difficult for the program to identify nonmembers, whereas the latter is not a well-defined set at all. Also, do not confuse a set's being semidecidable but not decidable by a program P with a set's being semidecidable but not decidable simpliciter. The latter means the set is computationally complex (in fact, it means it is recursively enumerable but not recursive), but the former, which appears in my theory, only means that the set is complex as far as the program P is concerned; P is unable to decide it, even though it may well be decidable.

Can the methods your brain uses to classify objects ever really be as fixed and deterministic as programs? Surely you do not actually have programs in your head! All that I mean by "program in the head" is that at some sufficiently high level of explanation your classification methods can be modeled as ($\{0,1\}$-valued) programs. In reality perhaps your programs change through time (perhaps even randomly)[2] or in different contexts, and some theories of vagueness depend in part on this.[3] If this is true, the hypotheses, principles and Thesis presented in this section can be modified to include "at any time t," and the Thesis will still imply vagueness at any one moment. Time and context-dependence are therefore unnecessary to explain vagueness. Although my theory does not depend on it, I often act as though the programs are fixed and unchanging in the head.

On a related note there are theories of vagueness that appeal in part to the claim (which is probably true) that different individuals do not share precisely the same interpretation of 'bald'.[4] The Thesis upon which my Undecidability Theory of Vagueness depends explains why you experience vagueness in "most" of natural language without having to assume anything about anyone else's programs. Thus, vagueness still would exist even if we all shared the same interpretations of natural language predicates. Vagueness in my theory's sights is a phenomenon of the individual, not of the masses. This fits better with the intuitively appealing conjecture that the world would still seem vague had I been the only man on Earth ever.

There is a simpler, equivalent way of stating the Thesis, one I implicitly used in the introduction. I had written there about a single program in the head, call it $P_{R/nonR}$, doing the work for both a predicate and its natural language negation: the interpretation of R was the set of objects on which $P_{R/nonR}$ outputs YES, and the interpretation of 'not R' was the set of objects on which the *same* program outputs NO. In the Thesis and throughout the remainder of the paper the single program is treated as two distinct programs: one program, P_R, for R; and another, P_{nonR}, for 'not R'. The interpretation of R is the set of objects on which P_R outputs YES, and the interpretation of 'not R' is the set of objects on which P_{nonR} outputs YES. Each of these two programs can output only a YES, if they halt at all; they do not ever output NO. Realize that there is no difference in these approaches: running $P_{R/nonR}$ on an input is equivalent to simultaneously

[2]One may treat a single randomized program as if it is a program randomly changing to new (but perhaps similar) programs through time.

[3]See Kamp ([Kam81]), Bosch ([Bos83]), and Burns ([Bur91], pp. 179-180).

[4]See Burns ([Bur91], p. 179).

running both P_R and P_{nonR} on the input and seeing who halts first (if any); if P_R halts first then $P_{R/nonR}$ would have output YES, but if P_{nonR} halts first then $P_{R/nonR}$ would have output NO. In terms of a single program, the Thesis would be the following: *for "most" natural language predicates R there are objects for which $P_{R/nonR}$ does not halt.* Although this is simpler than the statement at the start of this section, the two-programs version helps to clearly identify distinct aspects of the Thesis.

In the subsections that follow I indulge in a sort of fantasy. I imagine that you are a rational, computationally bound agent who has entered into our culture. Your task is to learn language for the first time and to determine what our natural language predicates mean. We will see that the Thesis is very likely to be true of any such agent. Such an agent will likely choose to have vagueness because its costs are less than the costs of avoiding it. I will also show that it is plausible that the Thesis does, in reality, apply to you.

As part of the fantasy I suppose that the true extensions (as opposed to your interpretations) of natural language predicates are determinate; that is, every object is either in the extension of the predicate or its complement.[5] I use 'extension' to refer to the true meaning of a predicate, and 'interpretation' to refer to whatever you mean by the predicate. All capital letters will be used to signify the extension of a predicate; e.g., 'bald' has the set BALD as its extension, and 'not bald' has the complement of BALD as its extension. In trying to figure out your interpretations for the language in the fantasy scenario, I suppose that you are presented with examples, somehow, from the true extensions. What I wish to communicate by this is that even if the true semantics of natural language predicates were determinate (via, perhaps, a semantic externalist account), you would still very likely end up with interpretations as specified in the Thesis (and thereby end up with vagueness). Thus, while standard classical two-valued logic would be a correct model of natural language true semantics, we will see that it is not a correct model of the way we actually interpret natural language predicates. On the question what *really* is the true semantics of natural language predicates my theory can remain agnostic.

2.1 Church-bound

The Undecidability Theory of Vagueness applies only to those agents that are computationally bound. Specifically, it applies only to those agents that

[5]For defenses of a determinate semantics within the vagueness literature see Campbell ([Cam74]), Cargile ([Car79]), Sorensen ([Sor88], [Sor94]) and Williamson ([Wil94]).

are "finite" and "sufficiently powerful."

By a *finite agent* I mean an agent (i) that has a finite but possibly unbounded memory, (ii) that has an upper bound on the speed at which it can compute, (iii) whose primitive computations are simple (e.g., adding 1 to a number), and (iv) who cannot (or at least does not) utilize in its computing any aspects of the universe allowing it to achieve supertasks (i.e., to achieve infinitely many steps in a finite period of time finite "brain"). To defend my use of the term 'finite' in this way, I informally rephrase these requirements as follows: by a finite agent I mean an agent that is finite in (i) memory, (ii) speed, (iii) degree of complexity of primitive computations and (iv) resourcefulness in utilizing nature to achieve supertasks.

Without (i), an agent could have infinitely large look-up tables in the head. Such an agent could compute any function at all by simply storing the entire function (i.e., storing every pair $\langle x, f(x)\rangle$) in its head, so long as the function has domain and range with cardinality no greater than the cardinality of the look-up table. The agent could merely check his look-up table to see what $f(x)$ is. Without (ii), an agent could compute the first step of a computation in half a second, the next in a fourth, the next in an eighth, etc., thereby computing infinitely many steps in one second (such an agent is called a Plato Machine). Without (iii), an agent may have primitive computations that are themselves as mathematically computationally difficult as one pleases; of course, from such an agent's point of view these computations would *seem* utterly simple, requiring only the least amount of "thinking" to compute.[6] Finally, without (iv), it is logically possible that the laws of physics might make it possible to compute supertasks (despite (ii)).[7] Being a finite agent severely constrains what an agent can compute, as I now describe.

We have an informal, pre-theoretic notion of what it is to compute something. Such an intuitive notion of a computation typically connotes that there be only a finite number of steps involved, that the amount of memory (and scratch paper) required also be finite, that each primitive step be relatively simple (enough to understand), and that one cannot engage in supertasks. That is, the intuitive notion of a computation exactly corresponds to those computations a finite agent can compute. We are inclined to say that a function f from the natural numbers to the natural numbers is intuitively computable if, for each natural number n, $f(n)$ is intuitively

[6] See Copeland ([Cop98]).

[7] See Earman and Norton ([EN93], [EN96]) and Hogarth ([Hog94].

computable.

The Turing machine formalism provides an abstract, precise notion of what a computation is and leads to a particular set of functions on the natural numbers as the set of Turing-computable functions. Any computation a modern computer can do, a Turing machine can, in principle, do also; and vice versa. There is the well known hypothesis that the set of functions that are intuitively computable just is the set of Turing-computable functions; this hypothesis is referred to as *Church's Thesis* (or the Church-Turing Thesis).

The hypothesis is not a mathematical assertion; it refers to our intuitions and it does not make sense to ask whether it has been mathematically proven. Nearly everyone believes in Church's Thesis, though, as do I. One reason for this is that no one has yet provided a convincing case of an intuitively computable function that is not Turing-computable; the longer we go without such a case being found, the higher our inductive probability goes toward one that the sets are identical.[8] A second, more slippery, reason nearly everyone believes in Church's Thesis is that half a dozen very different formalizations of computation have been concocted by different people and each leads to precisely the same set of computable functions.

If a finite agent can compute a function on the natural numbers, then the function must be intuitively computable. But then by Church's Thesis that function must be Turing-computable. Therefore, the only functions on the natural numbers a finite agent can possibly compute are those that are Turing-computable.

But any finite agent worth considering carries out computations on objects besides the natural numbers. What constraints are these computations under? Although there are objects besides natural numbers that are objects of such computations (i.e., such an agent computes functions over objects besides the natural numbers), we can encode all of the objects the finite agent can grasp—including natural numbers—onto the natural numbers. Supposing each different possible state of the finite agent's mind is finitely describable, the set of all such finite descriptions can be bijectively encoded onto the natural numbers (hopefully in an intuitively computable fashion). (Such an encoding is *bijective* if and only if each object gets assigned to a unique natural number and each natural number is used in the encoding.) '4' may now be the code for mental state p_1 which holds the information of the finite agent's mother, '37' the code for mental state p_2 which holds the

[8] One may refer to my [CB98] for references to the induction literature.

information of a particular fist fight the finite agent once witnessed, '18' the code for mental state p_3 which holds the information of the feeling of love-at-first-sight the finite agent felt upon meeting its spouse, '103' the code for the mental state p_4 which holds the information of the natural number 5, '1000' for the mental state p_5 which holds the information of the finite agent's favorite shade of blue, etc. Intuitively, every possible dog, every possible shade of color, every possible action, etc., is given its own natural number. With such an encoding, all of the finite agent's computations may be interpreted as computations on the natural numbers, and *the finite agent's computational power is constrained in such a way that it can compute only the Turing-computable functions on this set of codings.*

One should not find this too fantastic, given that the same sort of thing is true about every computer. In a physical computer, as opposed to an abstract model, there are no numbers actually input into the machine nor output from the machine; numbers are abstract objects. Rather, an input or output is some physical state and it encodes certain information. Each physical state is finitely describable and can be coded onto the natural numbers. '4' may be the code for physical state p_1 which holds the information of a black and white picture of a rooster, '37' may be the code for physical state p_2 which holds the information of natural number 5, '18' may be the code for physical state p_3 which holds the information of the sentence "Press any key to continue," etc. It is only through such means that one can meaningfully say that computers are subject to the same ultimate computational constraints as Turing machines, and it is also only through such means that one can meaningfully say that a finite agent is subject to the same ultimate computational constraints as Turing machines.

Worries over which coding is being employed for the finite agent are sometimes raised. For example, what if the coding makes intuitively uncomputable problems computable by having a non-intuitively computable coding? Or, is there a privileged coding and, if so, what determines it? I wish to sidestep all such issues. To whatever extent these are legitimate worries, they are worries for anyone claiming that even computers are bound by Church's Thesis. This latter claim is uncontroversial, however, and so I am under no special obligation to explain or address issues of coding with respect to finite agents.

One might complain that the universe has uncountably many possible objects, and so no bijection is possible onto the natural numbers. Supposing for the moment that there are indeed uncountably many possible objects, I only care about what possible objects the finite agent can hold before

its mind. Since it is finite, it can only entertain countably many possible objects. Its universe is countable, regardless of the cardinality of the real universe. This brings in its own trouble: if the universe is uncountable and the finite agent's universe countable, is not it going to have a false model of the world? The Downward Lowenheim-Skolem Theorem can help to alleviate this worry to an extent: as long as the finite agent notices only first-order properties of the universe, it is possible for its model to be such that the set of all truths is the same as God's (whose model is the true uncountable one). Should we, however, believe that it is confined to first-order properties? Perhaps, perhaps not; there are many things that can be said on this issue, but I have no need to pursue them here since nothing hinges on the agent's model being true.

Thus, I am confining discussion to finite agents, which means that I am confining discussion to agents capable of computing *only* the Turing-computable.

By "sufficiently powerful" I mean that the finite agent is capable of computing *at least* the Turing-computable.

Together, "finite" and "sufficiently powerful" imply that the computational powers of the agents I wish to discuss are bound by Church's Thesis and only bound by Church's Thesis. I sometimes say "Church-bound" instead of "finite and sufficiently powerful." I record this as the Church-Bound Constraint.

Church-Bound Constraint: *The agent can compute any function (over natural numbers coding mental states, which in turn represent objects in the world) so long as it is Turing-computable.*

Related to this constraint is the Church-Bound Hypothesis, which states that you are under the Church-Bound Constraint. The Church-Bound Hypothesis is, by assumption, true of the fantasy you. Is it true of the real you? Yes, and here is why. It is plausible that you are finite in the four senses discussed above,[9] and so cannot compute the Turing-uncomputable. Furthermore, you are, in principle, able (given enough time and scratch paper) to compute any Turing-computable function. We know this because any of us can easily mimic the simple actions of a Turing machine as long as we please.

[9] Although see Penrose ([Pen90]).

2.2 Programs in the head

Suppose that you, in the fantasy, are exposed to enough examples of things you have reason to believe are in the true extension of 'bald' (BALD) and others that are not that you acquire an educated guess as to what BALD is. Your guess determines some set as your "shot" at BALD, and this is your interpretation of 'bald'. In what can such a guess consist? There are infinitely many (possible) objects in your universe, infinitely many of them are bald and infinitely many are not. You are not, then, able to simply guess what the extension is, for you cannot store the extension since you are finite.

You must employ some sort of intension. You need to find some finite description of the set that determines your interpretation of 'bald' and your educated guess at BALD, and some finite description of 'not bald'. Recalling that these sets may be considered to be sets of natural numbers, one may wonder whether your interpretation of 'bald' can be described "in your head" as, say, a first-order sentence in the language of arithmetic (such sets are called *arithmetically definable*). For example, you may interpret 'bald' to be the set $\{n \mid \exists x \forall y\ P(n, x, y)\}$, where $P(n, x, y)$ is some recursive formula without quantifiers. The problem with this is that although it is indeed a finite description, the set is not recursively enumerable and since you are Church-bound it is generally too difficult for you to handle. (A set is *recursive* if and only if it is decidable by some program. A formula is recursive if and only if the set of objects satisfying it is recursive. A set is *recursively enumerable* if and only if it is semidecidable by some program.)

The same is true for any arithmetically definable set... except those that are recursively enumerable. For a recursively enumerable set it is possible for you to have a program in the head that says YES when and only when presented with objects in the set (although the program may never halt at all when presented with objects not in the set), but sets any more computationally difficult than recursively enumerable are beyond your reach. A program in your head, then, is what you must be employing to determine your interpretation of 'bald' if you wish to have an interpretation that is accessible to you. Your interpretation would then be the set of objects for which the program for 'bald' outputs YES, and this is recursively enumerable. This motivates the first rationality principle.

Principle of Program-Favoring: *Without good reason to the contrary, you should assume that the extension of natural language predicate R and its natural language negation 'not R' are capable of being correctly determined using programs in the head.*

This does seem to be a compelling principle of rationality: why choose interpretations that are not generally possible for you to actually use unless you have a good reason?

Supposing we believe that the Principle of Program-Favoring really is a constraint on rationality, is there good reason for believing that programs will not suffice to correctly interpret natural language predicates and their natural language negations? For example, in mathematics there *is* good reason for believing that programs do not suffice for certain predicates because there are predicates with interpretations that you know are not recursively enumerable. Consider the predicate 'not a theorem of Peano Arithmetic', for example. You know its extension is not recursively enumerable (since its complement is known to be recursively enumerable but not recursive). Your interpretation of 'not a theorem of PA' is set to its extension, regardless of the fact that you are incapable of generally recognizing things that are not theorems of PA. "To me, something is not a theorem of PA exactly if it does not follow from Peano's Axioms; I have no program for it, though." You might acquire a program in the head as a heuristic device aimed to approximately semidecide your interpretation of 'not a theorem of PA', but you are not confused into conflating your heuristic with your interpretation; you know that no such heuristic can possibly be the extension. Thus, you as a mathematician do have predicates for which you have good reason to believe the extension is not determinable via a program in the head, and your interpretations are, accordingly, not determined using programs in the head. (This is, in passing, why mathematical predicates such as 'not a theorem of PA' are not vague.) Given that you can acquire good reasons to believe programs are inadequate and can have interpretations that are not recursively enumerable, what reason is there for you not to do the same for natural language predicates?

The answer is that in the case of such a mathematical predicate you know what the definition of the extension is, and so you set your interpretation accordingly. For a natural language predicate, however, you have no God's eye view of its extension. The extension of 'bald' is learned via induction; you infer your interpretation of 'bald' from seeing objects you have reason to believe (somehow) are in BALD or its complement. You cannot easily acquire the definition for BALD, and as many examples of BALDness and its complement you might confidently find, you still will not have access to its definition in the way you have access to that of 'not a theorem of PA', for you have no luxury of setting your interpretation to that determined by the definition written on paper before you as you do for mathematical predicates

(and this has nothing to do with the fact that you are Church-bound). Given that you cannot have access to BALD in the way you have access to the extension of 'not a theorem of PA', it is also reasonable to suppose that you cannot learn that BALD is not recursive enumerable (supposing this were indeed true) in the way you learn that the extension of 'not a theorem of PA' is not recursively enumerable. I cannot discount the logical possibility of you, a Church-bound agent, learning (in the fantasy) through time that no recursively enumerable interpretation of 'bald' seems to fit the examples of BALD and its complement, and in this way assigning high probability to the hypothesis that BALD is not recursively enumerable, and therefore no program in the head is sufficient. The reasonable hypothesis, though, seems to be that for most (if not all) natural language predicates you have no good reason for believing that programs will not work. I record this as the following hypothesis.

No-Good-Reason-for-Non-Programs Hypothesis: *For most natural language predicates R and their natural language negation 'not R' you have no good reason to believe that programs in the head are inadequate for correctly determining their interpretation.*

The No-Good-Reason-for-Non-Programs Hypothesis together with the Principle of Program-Favoring imply the following hypothesis.

Programs-in-Head Hypothesis: *For most natural language predicates R and their natural language negation 'not R', their interpretations are determined by you using programs in the head.*

You in the fantasy scenario are, then, very likely to fall under the Programs-in-Head Hypothesis. "Very likely" because it is very likely that the No-Good-Reason-for-Non-Programs Hypothesis is true, and given that you are rational you will follow the Principle of Program-Favoring and thus fall under the Programs-in-Head Hypothesis.

Does the Programs-in-Head Hypothesis apply to the real you? Here is an intuitive reason to think so. For most natural language predicates R you are capable of recognizing, given enough time, any cases of R and 'not R'. E.g., given enough time you are capable of recognizing, for any bald-to-you person, that he is bald to you; and, for any not-bald-to-you person, that he is not bald to you. To suppose otherwise would imply, implausibly, that there is a person that is bald (not bald) to you, but you are utterly incapable of recognizing him as such. The only way for you, who are Church-bound, to have this recognition capability is to have programs in the head doing the

work.

2.3 Any algorithm

By the Programs-in-Head Hypothesis you have for most natural language predicates a program in the head as the intension determining your recursively enumerable interpretation of the predicate, and this interpretation is your attempt to fit the extension of the predicate. You would like to have a single program in the head capable of determining your interpretation of both 'bald' and 'not bald'; that is, a program that not only says YES exactly when an object is in your interpretation of 'bald', but says NO exactly when an object is not in your interpretation of 'bald'. This is just to say that you would like to have a program to decide the interpretation of 'bald', not just semidecide it. Such a program would be an algorithm since it would halt on every input, and the corresponding recursively enumerable interpretation of 'bald' would be recursive.

But alas, you are Church-bound, and a well-known undecidability result says that there is no algorithm for algorithmhood; there is no general procedure by which either you or a Turing machine can always choose programs (from the set of all possible programs) that are algorithms. It is not, then, generally the case that your programs in the head are algorithms, and your corresponding interpretations for natural language predicates and their natural language negations may generally be only semidecided by the programs for them.[10] If the interpretation of 'bald' ('not bald') is determined by a program in the head that semidecides but not decides it, then supposing that 'bald' ('not bald') is one of the predicates covered by the Programs-in-Head Hypothesis, that program cannot be what is determining the interpretation of 'not bald' ('bald'). This is because the Programs-in-Head Hypothesis states that 'not bald' ('bald') must have a program semideciding its interpretation, and the program for 'bald' ('not bald') cannot possibly be that program. Thus, 'not bald' ('bald') must have its own program in the head. I have now shown 1 and 2 of the Thesis.

How about 3 from the Thesis? It is possible for the interpretation of 'bald' and that of 'not bald' to cover every object, but by the Church-Bound Hypothesis this is not generally possible for you to accomplish. If it were generally possible, then the two programs semideciding each interpretation

[10] And in fact things are even worse than this, for a related undecidability result says that the corresponding interpretations are not generally even recursive; semidecide is all that any possible program can do in these cases.

could serve as a single algorithm (run both programs simultaneously until one halts), and you could therefore always acquire algorithms. But this is impossible. Thus, it is not generally the case that your interpretation of 'bald' and that of 'not bald' cover every object.

Notice that none of this hinges on either interpretation being non-recursive; what matters is the program for the interpretation semideciding but not deciding it. Predicates with finite interpretations (arguably 'small natural number') are therefore subject to the same conclusion just made concerning 'bald'.

Except for the use of "most" in the statement of the Thesis, I now seem to have shown that you are subject to the Thesis. Concerning "most," it is easier to acquire non-algorithms than algorithms, since in order to achieve algorithmic status the program must halt on every input, whereas to achieve non-algorithmic status there needs to be only one input on which the program does not halt.[11] This observation makes it convenient and informally true to say that for "most" natural language predicates your corresponding programs are not algorithms. This is really just elliptical for the proposition that you are not generally able to acquire algorithms and that it is more difficult to acquire algorithms than non-algorithms. To differentiate this use of 'most' (or 'usually') with genuine uses of it, I always put scare quotes around it.

With this it appears I have now shown you are subject to the Thesis. There is just one remaining problem. I wrote above (second paragraph of this subsection) that "there is no general procedure by which either you or a Turing machine can always choose programs (from the set of all possible programs) that are algorithms." The parenthetic remark merits some examination. Why should you be required to choose from among the set of *all* possible programs? Although the set of all algorithms is not recursively enumerable, there do exist proper subsets of the set of all algorithms that are recursively enumerable, and even recursive. Could you be choosing your programs from one of these subsets? For example, the set of primitive recursive programs is recursive, and perhaps you are choosing from this. If so, you can be sure that every program you choose is an algorithm, and thus that every one of your interpretations for natural language predicates is decidable by the program responsible for it (and is therefore recursive).

[11] More formally and in recursion theoretic terminology, this is captured by the fact that the set of algorithms is Π_2, and the set of non-algorithms Σ_2; the relative difficulty of acquiring algorithms versus non-algorithms is analogous to the relative difficulty of determining cases where a program does not halt versus when it does.

The Thesis would, then, not follow after all.

There is a good reason for the fantasy you not to confine yourself in such a fashion. Your interpretations of natural language predicates are a result of a learning process of some sort. You see cases you have reason to believe are in the extension of 'bald' (i.e., you are guided by the true semantics somehow), and you make an educated guess at the extension with your interpretation. *A priori*, you have no reason to believe that all concepts of the world can be correctly determined (or even adequately approximated) with algorithms from some recursively enumerable subset of the set of algorithms. Why should you believe that all extensions may be correctly determined with, say, primitive recursive interpretations? This motivates the following rationality principle.

Principle of No-R.E.-Subsets-of-Algorithms: *Without good reason to the contrary, you should not presume that there is a recursively enumerable subset of the set of all algorithms such that for all natural language predicates R (or 'not R'), algorithms from this subset supply the best interpretation for R ('not R').*

This is a compelling principle: why purposely choose a language with less rich interpretations without good reason? In fact, any recursively enumerable subset of the set of all algorithms is, in a certain real mathematical sense, infinitely less rich than the set of all algorithms.

Supposing we believe that the Principle of No-R.E.-Subsets-of-Algorithms is a constraint on rationality, is there good reason to believe that there are recursively enumerable subsets of the set of all algorithms sufficiently rich for natural language predicate interpretations? Although I am willing to suppose that it may be logically possible for you to acquire high probability in such a hypothesis (after, say, many years of searching for uses of algorithms outside of this recursively enumerable subset and not finding one), there would not appear to be actual evidence for such a supposition. This goes to support the following hypothesis.

No-Good-Reason-for-R.E.-Subsets-of-Algorithms Hypothesis: *There is no good reason for you to presume that there is a recursively enumerable subset of the set of all algorithms such that for all natural language predicates R ('not R'), algorithms from this subset supply the best interpretation for R ('not R').*

One might wonder whether there is nevertheless the following good pragmatic reason for confining algorithm choice to a recursively enumerable sub-

set of the set of all algorithms: by so confining oneself one does indeed avoid the Thesis (and vagueness). The solution comes with a painful price, though. For all you know there are algorithms that can provide the correct interpretation. Yes, not confining yourself to a recursively enumerable subset of the set of all algorithms brings with it the cost of there being objects in neither the interpretation of R nor the interpretation of 'not R'. However, it is possible for the interpretations to be only "finitely mistaken," where by this I mean that they are complements save for finitely many objects in neither interpretation. Constraining yourself to a recursively enumerable subset of the set of algorithms only for the pragmatic reason of avoiding the Thesis runs the risk that there are predicates, perhaps many, that are not only not correctly interpretable using algorithms from that subset, but will be infinitely mistaken. For example, if one constrains oneself to the set of primitive recursive functions without reason to believe that no predicates should best be interpreted using non-primitive recursive algorithms, then in all those cases where a predicate should be best interpreted using a non-primitive recursive algorithm you are guaranteed to incorrectly classify the objects on infinitely many occasions. Worse than this, it may be that no primitive recursive algorithm even "comes close" to the best algorithm for the predicate. It might be like using the set of odd numbers as an approximation to the prime numbers.

The No-Good-Reason-for-R.E.-Subsets-of-Algorithms Hypothesis conjoined with the Principle of No-R.E.-Subsets-of-Algorithms imply the following hypothesis.

No-R.E.-Subsets-of-Algorithms Hypothesis: *You do not confine your choice of programs to a recursively enumerable subset of the set of all algorithms when interpreting natural language predicates and their natural language negations.*

You in the fantasy scenario are, then, very likely to fall under the No-R.E.-Subsets-of-Algorithms Hypothesis. "Very likely" because it is very likely that the No-Good-Reason-for-R.E.-Subsets-of-Algorithms Hypothesis is true, and given that you are rational you will follow the Principle of No-R.E.-Subsets-of-Algorithms and thus fall under the No-R.E.-Subsets-of-Algorithms Hypothesis.

Does the No-R.E.-Subsets-of-Algorithms Hypothesis apply to the real you? There are reasons to think so. In fact, there is reason to think that the real you is subject to the following hypothesis.

Any-Algorithm Hypothesis: *You are free to choose from the set of*

all algorithms when interpreting natural language predicates or their natural language negations.

If the Any-Algorithm Hypothesis is true of you, then so is the No-R.E.-Subsets-of-Algorithms Hypothesis. This is because any set containing the set of all algorithms is not a recursively enumerable subset of the set of all algorithms.

What reasons are there to think that the Any-Algorithm Hypothesis is true of the real you? It is difficult to tell a plausible story about how (the real) you could have come to restrict program choice to exclude some algorithms, especially since by the Church-Bound Hypothesis you are capable of computing any algorithm. The man on the street does not know recursion theory, and even if he does, as I do, I cannot imagine attempting to restrict myself to, say, primitive recursive intensions for every new interpretation I acquire. Nor does it seem plausible to suppose that we humans might have evolved to exclude certain algorithms. It is, in fact, very difficult to avoid allowing yourself the choice of any algorithm since once you allow yourself the use of *'while' loops*—i.e., the ability to implement programs including statements like "while such and such is true, continue doing blah"—you are able to build, in principle, any algorithm (presuming you can also carry out some trivial basic operations).

To avoid this conclusion you would have to ban the use of 'while' loops, using only *'for' loops*—i.e., the ability to implement programs including statements like "for i becomes equal to 1 to n do blah", or "do blah n times"—which is very restrictive. One could argue that your 'while' loops are in reality bounded since you do not (and cannot) let them run forever; thus, it is not the case that every algorithm can be implemented. But this does not mean that the proper representation of your program does not use a 'while' loop. No real computer, after all, can actually implement unbounded 'while' loops, but it would be a mistake to say they cannot run unbounded 'while' loops and any algorithm.

It can be noted that the idea of animals employing 'while' loops has some empirical support, namely in the *Sphex ichneumoneus* wasp, which has been observed to enter into what is plausibly represented as an infinite loop. Consider the following often quoted excerpt from Woolridge.[12]

> When the time comes for egg laying, the wasp *Sphex* builds a burrow for the purpose and seeks out a cricket which she stings in

[12] [Woo63], p. 82.

> such a way as to paralyze but not kill it. She drags the cricket into the burrow, lays her eggs alongside, closes the burrow, then flies away, never to return. In due course, the eggs hatch and the wasp grubs feed off the paralyzed cricket, which has not decayed, having been kept in the wasp equivalent of deep freeze. To the human mind, such an elaborately organized and seemingly purposeful routine conveys a convincing flavor of logic and thoughtfulness—until more details are examined. For example, the Wasp's routine is to bring the paralyzed cricket to the burrow, leave it on the threshold, go inside to see that all is well, emerge, and then drag the cricket in. If the cricket is moved a few inches away while the wasp is inside making her preliminary inspection, the wasp, on emerging from the burrow, will bring the cricket back to the threshold, but not inside, and will then repeat the preparatory procedure of entering the burrow to see that everything is all right. If again the cricket is removed a few inches while the wasp is inside, once again she will move the cricket up to the threshold and re-enter the burrow for a final check. The wasp never thinks of pulling the cricket straight in. On one occasion this procedure was repeated forty times, always with the same result.

I am not suggesting that you are possibly subject to such infinite loops. I am only suggesting that 'while' loops are plausibly part of your computational grammar. In fact, one might say that the wasp has a "concept" of 'readied burrow' which is determined by the following program:

```
WHILE burrow not ready do
   IF burrow is clear and cricket has not moved when I emerge
    THEN burrow is ready;
```

As an example showing that you regularly engage in 'while' loops (or an equivalent) as well, in order to determine if the bath temperature is good, you may well keep increasing the hot until it is comfortable or too hot; if the latter then you keep decreasing until comfortable; and so on. That is, you implement the following program:

```
WHILE temperature not comfortable do
 IF temperature too cold
  THEN increase hot water;
  ELSE decrease hot water;
```

'while' loops seem to be an integral part of your (and my) computational grammar. And if this is true, the Any-Algorithm Hypothesis is sure to apply

to the real you. Thus, the No-R.E.-Subsets-of-Algorithms Hypothesis also applies to the real you.

What does the No-R.E.-Subsets-of-Algorithms Hypothesis tell us? There is a set of all possible programs you can attain (and this is recursively enumerable since you are bound by Church's Thesis). This set is not a subset of the set of all algorithms, as the No-R.E.-Subsets-of-Algorithms Hypothesis requires. This means you are not generally capable of choosing algorithms. In particular, if the Any-Algorithm Hypothesis is true, then since the set of all algorithms is undecidable, you are not generally capable of choosing algorithms. We saw above that the Church-Bound Hypothesis and the Programs-in-Head Hypothesis "almost" imply the Thesis. What was missing was some reason to believe that the set of algorithms from which you determine your interpretations is not recursively enumerable. The No-R.E.-Subsets-of-Algorithms Hypothesis finishes the argument, and these three hypotheses together entail the Thesis.

2.4 Thesis

In this subsection I bring together the previous three subsections. Here is the Thesis again.

For "most" natural language predicates R

1. your interpretation of R is determined by a program in your head that is capable of semideciding but not deciding it,

2. your interpretation of 'not R' is determined by another program in your head that is capable of semideciding but not deciding it, and

3. there are objects neither in your interpretation of R nor in your interpretation of 'not R'.

There are two ways of arguing toward the Thesis. The first is via the fantasy scenario and is largely but not entirely prescriptive, concluding that the Thesis, and thus vagueness, follows largely but not entirely from rationality considerations alone (and the Church-Bound Constraint). The second is via the real you scenario and is descriptive, concluding that the Thesis follows from hypotheses that are true about us. The first would be rather worthless without the second, because a theory that claims that vagueness would exist in the fantasy scenario but says nothing about the real us would

be incomplete at best, since it is us who experience vagueness, not some idealized, rational fantasy agents. The second, however, is made more interesting by the first. The Thesis, and thus vagueness, does not follow from some human irrationality or quirk, but is, on the contrary, something to which nearly any rational, sufficiently powerful, finite agent will converge.

I finish this section by (i) cataloguing the premises of both the prescriptive (fantasy you) and the descriptive (real you) argument, (ii) reminding us that the premises of the prescriptive argument entail those of the descriptive argument, and (iii) summarizing how the Thesis follows from the descriptive argument (and thus by (ii) also from the prescriptive argument). Below are the two arguments.

Descriptive Argument (Real you)	Prescriptive Argument (Fantasy you)
Church-Bound Hypothesis.	Church-Bound Constraint.
Programs-in-Head Hypothesis.	Principle of Program-Favoring.
	No-Good-Reason-for-Non-Programs Hypothesis.
No-R.E.-Subsets-of-Algorithms Hypothesis.	Principle of No-R.E.-Subsets-of-Algorithms.
	No-Good-Reason-for-R.E.-Subsets-of-Algorithms Hypothesis.

The prescriptive argument says that any rational (Principles of Program-Favoring and No-R.E.-Subsets-of-Algorithms), Church-bound (Church-Bound Constraint) agent is subject to the Thesis so long as (a) he has no good reason to believe that the extensions of most natural language predicates are not recursively enumerable (No-Good-Reason-for-Non-Programs Hypothesis), and (b) he has no good reason to presume that there is a recursively enumerable subset of the set of all algorithms that suffices for adequate interpretations of natural language predicates (No-Good-Reason-for-R.E.-Subsets-of-Algorithms Hypothesis). Because (a) and (b) are very difficult to imagine being false, the Thesis follows "largely" from the Church-Bound Constraint and the two principles of rationality. Supposing (a) and (b) are true, the Thesis (and thus vagueness) is good for you, fantasy and real.

The descriptive argument says that (α) we humans are Church-bound (Church-Bound Hypothesis), (β) for most natural language predicates and their natural language negations we use programs in the head to determine our interpretations of them (Programs-in-Head Hypothesis), and (γ) any algorithm may possibly be used by us as a determiner of the interpretations of natural language predicates or their natural language negations (Any-

Algorithm Hypothesis), and thus we do not confine ourselves to a recursively enumerable subset of the set of all algorithms for interpreting natural language predicates or their natural language negations (No-R.E.-Subsets-of-Algorithms Hypothesis).

The prescriptive premises (for the fantasy scenario) imply the descriptive premises in the following sense. If you satisfy the Church-Bound Constraint then the Church-Bound Hypothesis is true. If you follow the Principle of Program-Favoring and the No-Good-Reason-for-Non-Programs Hypothesis is true, then the Programs-in-Head Hypothesis is true; the converse is not true. If you follow the Principle of No-R.E-Subsets-of-Algorithms and the No-Good-Reason-for-R.E.-Subsets-of-Algorithms Hypothesis is true, then the No-R.E.-Subsets-of-Algorithms Hypothesis is true; the converse is not true.

The descriptive premises entail the Thesis as follows. The Programs-in-Head Hypothesis states that you use programs in the head to determine most of your interpretations of natural language predicates and their natural language negations. Most of your interpretations are therefore semidecidable by the programs responsible for them. The No-R.E.-Subsets-of-Algorithms Hypothesis states that the set of programs at your disposal for natural language predicate interpretations is not a recursively enumerable subset of the set of all algorithms. This entails that the set of algorithms from which you can possibly choose is not recursively enumerable. The Church-Bound Hypothesis states that you can only compute the Turing-computable, and thus you cannot generally choose programs for your interpretations of natural language that are algorithms (even if your interpretations are recursive, or even finite). For "most" of your interpretations of natural language predicates R (or 'not R') your program for it will be able to semidecide but not decide it. Since "most" predicates can only semidecide their interpretation, this means that for "most" predicates there must be one program for R, and another program for 'not R', and each can only semidecide the interpretation for which it is responsible. We have so far concluded that "most" natural language predicates satisfy 1 and 2 of the Thesis. Your interpretations of R and 'not R' cannot cover every object, because if they could be then it would be possible to take the two programs and use them as one algorithm to simultaneously decide the interpretations, and this would contradict the impossibility of generally acquiring algorithms. Thus, "most" of the time your interpretations of predicates and their natural language negations do not cover all objects; there are objects in neither interpretation. We now have that "most" predicates satisfy 1, 2 and 3; the Thesis follows from the

three hypotheses.

It is important to note that the Thesis is an important claim about natural language whether or not one believes that the Thesis has anything to do with vagueness. If it is true, then, informally, our concepts have "unseeable holes" in them. As for vagueness, my theory's characterization of vagueness is that a predicate is *vague* if and only if it satisfies 1, 2 and 3 from the Thesis; the truth of the Thesis implies that "most" natural language predicates are indeed vague, as we know they are.

3 Vagueness

In this section I demonstrate how the Undecidability Theory of Vagueness explains vagueness. Recall that the theory's characterization of vagueness from Subsection 2.4 is as follows: Predicate R is vague to you if and only if

1. your interpretation of R is determined by a program in your head that is capable of semideciding but not deciding it,

2. your interpretation of 'not R' is determined by a program in your head that is capable of semideciding but not deciding it, and

3. there are objects in neither your interpretation of R nor your interpretation of 'not R'.

The Thesis stated that "most" natural language predicates satisfy 1, 2 and 3, i.e., "most" natural language predicates are vague. In terms of a single program $P_{R/nonR}$ that outputs YES whenever an object is R and outputs NO whenever an object is 'not R', the characterization is that a predicate R is vague to you if and only if there are objects on which your program $P_{R/nonR}$ does not halt. The corresponding Thesis is that "most" natural language predicates have a region of objects for which the program does not halt. In what follows the Thesis is assumed to be true.

Please notice that in my theory's characterization of vagueness, vague predicates are not in any way required to be computationally complex. I have a great deal of difficulty with people thinking that my theory somehow equates non-recursiveness with vagueness. The only appeal to non-recursiveness has been to the non-recursiveness of the set of algorithms and the halting set (the set of all pairs of programs and inputs such that the program halts on that input), not to the non-recursiveness of natural language predicate interpretations. The interpretations of vague predicates may well

be recursive, and even finite, and vagueness is unscathed. And even if a predicate's interpretation is not recursive, the vagueness comes *not* from this but, as we will see, from the facts that the interpretations do not cover all objects and that the programs are not algorithms.

3.1 Borderline region

Your interpretations of R and 'not R' do not cover all objects; there are objects c such that the natural language sentences 'c is R' and 'c is not R' are both false. These objects comprise the borderline region. This fits well with the datum of a borderline region: that there are objects which do not seem to fit neatly into just one category. The development in Section 2 served in part to show that (i) any rational, Church-bound agent in the fantasy is very likely to have a borderline region for "most" natural language predicates, and (ii) you do, in fact, have such a borderline region for "most" natural language predicates.

In epistemic theories of vagueness the borderline region is characterized differently than merely "not fitting neatly into just one category." Rather, for epistemicists the borderline region is comprised of those objects for which knowledge of membership is unattainable, where "membership" refers to membership in the true extension. The Undecidability Theory explains this sort of borderline region as well. Suppose that BALD is the true extension of 'bald'. You are not generally capable of acquiring a program in the head that decides BALD, even if BALD is decidable, because you are not generally capable of acquiring algorithms. Your interpretation of 'bald' is semidecidable but not generally decidable by the program responsible for it, and even if you are so lucky to correctly interpret it (i.e., your interpretation is equal to the extension BALD), if you want to be able to respond to queries about 'not bald' you must acquire a second program in the head, and this program will not generally correctly interpret 'not bald' (i.e., the 'not bald' program will not semidecide the complement of BALD). Your interpretations of 'bald' and 'not bald' do not cover every object, and the programs for each only semidecide them. There are therefore objects for which you are incapable of determining or even knowing, using your programs in your head, whether or not it is a member of BALD.

3.2 Higher-order vagueness

Although you cannot draw a sharp line between 'bald' and 'not bald', can you draw a sharp line between 'bald' and 'borderline bald'? There is, in fact, a sharp line here posited by my theory, but are you generally capable of drawing it? No. The two programs in the head for baldness (one for 'bald' and one for 'not bald') are not powerful enough to determine the lines. To see this intuitively, imagine starting in the 'bald' region and moving toward the borderline bald region. While in the 'bald' region your program for 'bald' halts and says YES and the program for 'not bald' never halts. When you move into the borderline region the program for 'not bald' still does not halt, but the program for 'bald' suddenly now never halts as well. You are not, though, generally able to know that the program for 'bald' will never halt—you cannot generally know when you have crossed the line. This seems to be consistent with our observations of higher-order vagueness (see also Subsection 3.3), and it solves the problem without having to postulate semantically distinct higher-order borderline regions. This latter aspect is good since it puts a stop to the regress of higher and higher order semantically distinct borderline regions, all which amount to nothing if when one is finished there is still a knowable sharp line between the definite region and the non-definite region.

Can this phenomenon really be the phenomenon of higher-order vagueness? In my theory what does it "feel like" to not be capable of determining the boundaries of the borderline region? Well it feels like whatever it feels like to attempt to decide a set using a program that only semidecides it. One might try to make the following criticism: Let us take the set of even numbers and supply you with a program that only says YES exactly when a number is even, and is otherwise silent. Do the evens now seem vague through the lens of this program? There are a number of problems with such a criticism as stated. First, it is not enough that the program simply says YES when an input is even and is silent otherwise. When we say that the program semidecides but does not decide the set of evens we mean that if the program is silent we are not sure whether it will at any moment converge and say YES. The program's silence is not translatable to NO. Second, it is difficult to judge our intuitions with a predicate like 'even' for which we already have a program in the head for deciding it. We should imagine instead that it is some new predicate R for which we have no intuitions. The third problem is that even with these fixes the question the critic needs to ask is not whether R-ness seems vague, but whether R-ness seems to have

whatever feel higher-order vagueness has. This is because R is not vague according to my theory since it does not satisfy part 2 of the characterization of vagueness (i.e., we are not given a program for semideciding 'not R'). On this modified question it is unclear that we have any compelling intuitions that the answer is NO. When using the given program to attempt to decide the extension of R, you will be incapable of seeing where exactly the boundary is, and therefore you will be unable to classify many objects. These objects plausibly are just like the borderline borderline objects (i.e., second-order borderline objects).

Another critic may ask the following: Let us suppose that you have two programs that only semidecide their respective interpretations, and let us also suppose that the interpretations do not cover every object. If these programs are for some predicate R and 'not R' then is R-ness necessarily vague? For example, let us take the predicate 'theorem of arithmetic', whose extension is not even recursively enumerable. You are surely capable of determining some theorems and some non-theorems, and you must therefore utilize a program in the head for 'theorem' and another for 'not theorem'. But surely 'theorem' is not now vague! There is a difference between this case and vague natural language predicates. You as a mathematician are conscious that you are not actually deciding theoremhood with your programs. You understand that they are only heuristics, and it is possible that each might even occasionally be incorrect, e.g., saying that a theorem is not a theorem. That is, your programs for theoremhood do not determine what you mean by 'theorem'. You mean by 'theorem of arithmetic' whatever follows from its definition. 1 and 2 from the characterization of vagueness are not satisfied.

3.3 Experiment

In addition to claiming the Undecidability Theory fits our pre-theoretic intuitions concerning the phenomenon of vagueness, I have carried out two experiments, one to record how genuine vagueness looks and the other to test my theory against the results of the first experiment.

In the first experiment I presented subjects with a sorites series of 27 color patches from red to yellow. For each color patch subjects were asked to write whether they believed it was red, borderline red (or gray area), or not red. In addition, for each response subjects were asked to give a confidence rating of 1, 2 or 3 indicating how confident they were of their answer (3 being highly confident). Of 17 subjects the average borderline

region consisted of 5 (plus or minus 3) color patches. This is hard evidence that there is indeed a borderline region; borderline vagueness exists. (In pilot data I have found that subjects given the same task but not given the option of answering "borderline red" give low confidence ratings throughout a large region, which seems best interpreted as a borderline region. Thus, existence of the "borderline red" option does not seem to be creating a borderline region.) The typical subject begins at the red patch with an answer of "red, 3", i.e., "red with highest confidence". Then at the subject's transition from "red" to "borderline red" the confidence ratings are low. At the center of what a subject calls the borderline region the confidence rating increases from that at the transition. At the subject's transition from "borderline red" to "not red" the confidence ratings are again low. Finally, at the 27th color patch the answer is "not red" with highest confidence. For the 17 subjects involved in this experiment the mean confidence ratings for the five distinguished points (just mentioned) across the subjects is shown below; standard deviations are shown in parentheses. (The "transition" numbers are obtained by taking the mean of all subjects' confidence ratings on either side of the transition.)

Red:	3.00 (0.00)
Red-borderline transition:	2.11 (0.69)
Borderline:	2.62 (0.55)
Borderline-not red transition:	2.21 (0.84)
Not red:	3.00 (0.00)

The low confidence ratings at the boundaries of the borderline region (i.e., the transitions) are evidence for the existence of higher-order vagueness, and I will take it as an operational definition of higher-order vagueness.

To test my own theory of vagueness against this data I presented subjects with a sorites series of 27 made-up figures gradually morphing from the first figure to the final. Subjects were told that the first object is pirnk, and that the series changes gradually so that the last figure is zuff. It is possible but not necessary, I told them, that there are some figures in between that are neither pirnk nor zuff. Of course, the subjects did not know what the properties pirnkness and zuffness are, so I supplied them with a computer program (their program in the head) that did. Subjects were asked to enter the number of the object into the program, and they were told that the program would respond "pirnk" if the input was pirnk and "zuff" if the input was zuff. They were warned that the program may well take some time to halt; furthermore, they were warned that if the figure is neither pirnk nor

zuff, the program will never halt. They were instructed to terminate the program if they wished to give up waiting for it to halt. The program was set up so that the 'neither' region consisted of figures 11 through 15 and the figures nearer the neither region took longer to halt (this amounts to a natural but new assumption for the sake of the experiment). The program on inputs 10 and 16, for example, took about three minutes, were anyone to be patient enough; the program on inputs 1 and 27, for example, took less than one second. 'pirnk' is the analog of 'red', 'zuff' the analog of 'not red', and 'neither' the analog of 'borderline red'. My prediction was that subjects' confidence ratings at the distinguished points along the sorites series would be similar to those in the 'red' experiment. The means over all nine subjects are as follows:

Pirnk:	3.00 (0.00)
Pirnk-neither transition:	1.72 (0.96)
Neither:	2.22 (0.83)
Neither-zuff transition:	1.67 (0.91)
Zuff:	3.00 (0.00)

As in the case of known vagueness (i.e., the 'red' experiment), we see here low confidence ratings at the boundaries of the analog of the borderline region; my theory captures what I operationally defined as higher-order vagueness. In addition, we also see a higher mean confidence rating at the centers of the 'neither' region, just as is the case for the center of the borderline red region. The results of this experiment help increase the plausibility of my theory, and I hope they motivate further empirical studies.

A worry one might express is that in my invented sorites series (pirnk to zuff) I have presumed that the borderline region is both contiguous and sandwiched between the definite regions. Why should this be? The reason is that a sorites series is, by its nature, a sequence moving as gradually as possible from one extreme to another. An object being neither R nor not R means it is intuitively less R than R things and more R than not-R things, so all such objects must be placed on a sorites series in between the R and the not-R things.

3.4 The sorites paradox

Finally we arrive at the sorites paradox, which I give here in the following form: (i) 0 hairs is bald, (ii) for all n, if n hairs is bald, so is $n + 1$, (iii) therefore you are bald no matter how many hairs you have. Notice that I

have stated the argument in natural language; many researchers on vagueness state the paradox in some logical language, which is strange since the paradox is one in natural language. Presenting it in a logical language inevitably makes certain biased presumptions; for example that 'not' is to be translated to the classical negation '¬'.

What is usually dangerous about rejecting premise (ii) is that it implies there is an n_0 such that n_0 hairs is bald but n_0+1 hairs is not; i.e., it usually leads to there being no borderline region. This is bad because borderline regions surely exist. In my theory's case, though, what happens? A sorites series moves along a "path" that is most gradual from R to 'not R'; it must therefore cross the borderline region lest it not be "most gradual." Imagine starting in the 'bald' region and moving toward the borderline region. Eventually there will be a number n_0 such that n_0 hairs is bald but it is not case that $n_0 + 1$ is bald, and you cannot in general determine where this occurs. However, this in no way prevents $n_0 + 1$ from being borderline bald, i.e., being neither bald nor not bald. Eventually the denial of (ii) will occur—and you will not know when—but it does not imply the lack of a borderline region. The sorites paradox is thus prevented without losing vagueness.

3.5 Universality of vagueness

I now touch on three issues related to the universality of vagueness.

The first concerns whether all natural language predicates are vague. By the Thesis and the characterization of vagueness, "most" of your natural language predicates are vague. "most," however, does not mean all, and according to my theory there may be some non-vague predicates. But are not all natural language (nonrelational) predicates vague? It is not clear that the answer to this question is 'Yes'. For example, Sorensen[13] cites 'flat', 'clean' and 'empty' as example non-vague predicates. These predicates are often applied in "restricted domains of discourse; not all bumps, dirt, and contents are relevant,"[14] but if I am asked if, strictly speaking, some surface is flat, I am sure that my answer is either YES or NO (i.e., not neither). "Strictly speaking," surfaces are either flat or not, whereas for 'bald' there is no such "strictly speaking" analog. I also believe 'mortal' and 'everlasting', for example, to be non-vague. There are explanations consistent with my theory for why non-vague predicates are rare at best. The first emanates from the observation made in Section 2.3 that the set

[13] [Sor88], p. 201.
[14] ibid.

of algorithms is much more difficult than its complement, and this is what motivated the scare quotes around 'most' in the first place. The second explanation helps to explain why there are few to no non-vague "complex" natural language predicates. By complex predicates I informally mean those predicates like 'dog', 'bald', 'people', 'chair', etc., that depend on a number of more "primitive" predicates like 'red', 'circular', etc., for their application. Most of our every day predicates—the ones we use to carve up the world—are complex. In order for one of these predicates to be non-vague, *every* more primitive concept it employs must be non-vague,[15] and this is probably never the case, given that "most" (primitive) concepts are, according to my theory, vague.

The second universality issue is that given that some predicates might be non-vague, we do not find in our experiences cases where, say, 'dog' is vague but 'cat' is not; similar sorts of predicates should either both be vague or neither. The observation just mentioned concerning complex versus primitive concepts explains this datum. Similar predicates make use of similar more primitive concepts, and thus inherit the vagueness (or lack thereof) of the more primitive concepts.

On the third issue of universality, the Thesis is about your interpretations, stating that "most" of the time your natural language predicates are vague. The Thesis obviously also applies to any of us individually. One datum of vagueness seems to be that we don't find ourselves disagreeing about the vagueness of predicates. What reason have we to believe, in my theory's sights, that you and I have the same vague predicates? Why should your "most" and my "most" coincide? The answer to this query is as follows: If you believe 'bald' is vague and I believe it is non-vague, then it is not the case that we have the same concept of baldness save that one is vague and the other not. Your concept of baldness consists of two interpretations which do not cover every object. My concept of baldness, on the other hand, consists of just a single classical interpretation; I have no "hole" in my concept. We disagree about more than just baldness's vagueness since our concepts are genuinely different. Therefore, in order to explain why we all agree on which predicates are vague, it suffices to explain why we all tend to have the same concepts for predicates. Explaining *this*, however, is not something my theory is subject to any more than any other theory; any adequate account of our shared concepts suffices to explain why we agree

[15] Although see Sorensen ([Sor88], pp. 228-229) for some nice and unusual counterexamples.

about the vagueness of predicates.

4 Conclusion

I have argued that it is very likely to be in a Church-bound agent's (i.e., finite and sufficiently powerful) best interest to have accessible (i.e., interpreted via programs in the head), maximally accurate (i.e., any algorithm a possible meaning determiner) interpretations of natural language predicates. I have also argued that such an agent having such interpretations experiences vagueness. Vagueness, then, is in such an agent's best interest. If we, too, are Church-bound, then vagueness is very likely *good* for us; the only possible ways for us to avoid vagueness are either to lose the accessibility of our natural language meanings or to confine our meanings to an "infinitely less rich" choice, each very likely to be more costly than the costs of vagueness.

References

[Bos83] P. Bosch. "vagueness" is context-dependence. A solution to the sorites paradox. In T. T. Ballmer and M. Pinkal, editors, *Approaching Vagueness*, pages 189–210. North-Holland, 1983.

[Bur91] L. C. Burns. *Vagueness: An Investigation into Natural Languages and the Sorites Paradox*. Kluwer, 1991.

[Cam74] R. Campbell. The sorites paradox. *Philosophical Studies*, 26:175–191, 1974.

[Car79] J. Cargile. The sorites paradox. *Phil. Studies*, 26:175–191, 1979.

[CB98] M. A. Changizi and T. P. Barber. A paradigm-based solution to the riddle of induction. *Synthese*, 117:419–484, 1998.

[Cha95] M. A. Changizi. Fuzziness in classical two-valued logic. In *Proc. of the Third Int. Symp. on Uncertainty Modeling and Analysis, and the Annual Conf. of the North American Fuzzy Information Processing Soc.*, pages 483–488. IEEE, 1995.

[Cha99a] M. A. Changizi. Motivation for a new semantics for vagueness. *EWIC (Irish Workshop on Formal Methods)*, 1999.

[Cha99b] M. A. Changizi. Vagueness and computation. To appear in Acta Analytica, 1999.

[Cop98] B. J. Copeland. Turing's O-machines, Searle, Penrose and the brain. *Analysis*, 58(2):128–138, 1998.

[EN93] J. Earman and J. D. Norton. Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science*, 60:22–42, 1993.

[EN96] J. Earman and J. D. Norton. Infinite pains: The trouble with supertasks. In A. Morton and S. P. Stich, editors, *Benacerraf and his Critics*, pages 65–124. Blackwell, 1996.

[Hog94] M. Hogarth. Non-Turing computers and non-Turing computability. *Proceedings of the Philosophy of Science Association*, 1:126–138, 1994.

[Kam81] J. A. W. Kamp. The paradox of the heap. In U. Monnich, editor, *Aspects of Philosophical Logic*, pages 225–277. D. Reidel, 1981.

[Pen90] R. Penrose. Précis of *the emperor's new mind: concerning computers, minds, and the laws of physics*. *Behavioral and Brain Sciences*, 13:643–655 and 692–705, 1990.

[Sor88] R. A. Sorensen. *Blindspots*. Clarendon Press, 1988.

[Sor94] R. A. Sorensen. A thousand clones. *Mind*, 103(409):47–54, 1994.

[Wil94] T. Williamson. *Vagueness*. Routledge, 1994.

[Woo63] D. Woolridge. *The Machinery of the Brain*. McGraw Hill, 1963.