Towards Universal Semantic Communication

(Version 0 - Very Preliminary)

Brendan Juba* Madhu Sudan †

MIT CSAIL
{bjuba,madhu}@mit.edu

February 12, 2007

Abstract

Is is possible for two intelligent beings to communicate meaningfully, without any common language or background? This question has interest on its own, but is especially relevant in the context of modern computational infrastructures where setting up a common protocol between computers is getting to be increasingly burdensome, and where "universal communication protocols" may start to become attractive.

In this paper we attempt to formalize this problem in a computational setting, where the goal of one of the interacting players is to gain some computational wisdom from the other player. We show that if the second player is "sufficiently" helpful and powerful, then the first player can gain significant computational power (deciding PSPACE complete languages).

Our work highlights some of the definitional issues underlying the task of formalizing universal communication, but also suggests some interesting phenomena and highlights potential tools that may be used for such communication.

*Supported by an Akamai Presidential Fellowship, a NSF Graduate Research Fellowship, and NSF Award CCR-0514915.

†Supported in part by NSF Award CCR-0514915.

1 Introduction

Consider the following scenario: Alice, an extraterrestrial, decides to initiate contact with a terrestrial named Bob by means of a radio wave transmission. How should he respond to her? Will he ever be able to understand her message? In this paper we explore such scenarios by framing the underlying questions computationally.

1.1 Motivation

We believe that the question raised above has intrinsic interest, in that it raises some fundamental questions. How does one formalize the concept of understanding? Does communication between intelligent beings require a "hardwired" common sense of *meaning* or *language*? Or can intelligence substitute for such requirements? What role, if any, does computational complexity play in all this?

However this is not the sole motivation for studying this question. We believe that this question goes to the heart of "protocol issues" in modern computer networks. Modern computational infrastructures are built around the concept of communication and indeed a vast amount of effort is poured into the task of ensuring that the computers work properly as communication devices. Yet as computers and networks continue to evolve at this rapid pace, one problem is becoming increasingly burdensome: that of ensuring that every pair of computers is able to "understand" each other, so as to communicate meaningfully.

Current infrastrusctures ensure this ability for pairs to talk to each other by explicitly going through a "setup" phase, where a third party who knows the specifications of both elements of a pair sets up a common language/protocol for the two to talk to each other, and then either or both players learn (download) this common language to establish communication. An everyday example of such an occurence is when we attempt to get our computer to print on a new printer. We download a device driver on our computer which is a common language written by someone who knows both our computer and the printer.

We remark that this issue is a fundamental one, and not merely an issue of improper design. It is almost unreasonable to establish a strict protocol for computers to converse for each other and expect this protocol to remain static in the future—asking for a protocol to remain static seems no more reasonable than asking humans to stick to a single fixed language over time. Thus it is essential to understand how to monitor the process of "updating" a computer to maintain universal connectivity over time.

The current model for maintaining connectivity is based (implicitly) on trusted "third parties" who advise us on when to update our languages. This process however leads to compromises in terms of efficiency as computers spend more of their time downloading languages and less on real computation or communication; reliability, since the protocols often lead to inadvertent errors; and security, because many viruses and other corrupting elements are introduced at this stage. In particular defining interfaces/protocols is an extremely complex task, made especially so because of the fact that the players at the two ends are "universal computers" and could behave in any of a countable number of different ways. Anticipating all possible behaviors and reasoning about them is a daunting task, to say the least.

This leads us to consider a somewhat radical alternative scenario. Perhaps we should not set computers up with common languages, but rather exploit the universality in our favor, by letting them evolve a common language. This leads us to the (somewhat colorful) problem described in

the opening paragraph: We model the two players that wish to communicate with each other as an alien and a human who may have no prior common background, and ask if they can eventually come to understanding about the meanings of sentences.

1.2 Some history

Despite the fantastic nature of the topic (of conversation with an alien), it has been seriously considered in the past, though mostly in an empirical context. Most notably, Freudenthal [7] claims that it is possible to code messages describing mathematics, physics, or even simple stories in such a radio transmission which can be understood by any sufficiently humanlike recipient. Ideally, we would like to have such a rich language at our disposal; it should be clear that the "catch" lies in Freudenthal's assumption of a "humanlike" recipient, which serves as a catch-all for the various assumptions that serve as the foundations for Freudenthal's scheme.

We do not mean to dismiss Freudenthal's contributions entirely. It is possible to state more precise assumptions which permit Freudenthal's scheme to work, but among these will be some fairly strong assumptions about how the recipient interprets the message, and an assumption that all semantic concepts of interest can be characterized by lists of syntactic examples. (See Section C of the appendix for a more detailed discussion.) These assumptions seem, for the moment, to be too strong to be considered truly universal. Instead of making such strong assumptions, we will restrict the class of semantic concepts under consideration to *computational* properties. We will do so by setting up the problem as "goal-oriented communication." In this context there are some natural prior approaches that we should consider and contrast with.

The general theory of communication [15] typically does not investigate the meaning associated with information and simply studies the process of communicating the information. The theory of communication complexity [17] starts to associate meaning to communication (e.g., the reason Alice is sending information to Bob may be so that he can compute $f(x, y)$ where Alice knows x and Bob knows y), but still assumes Alice and Bob know a common language (e.g., the function f). Finally, the theory of interactive proofs [8, 1] (and also the related theory of program checking [3]) gets further into the gap between Alice and Bob, by ascribing to them different intents, though they still share common semantics. It turns out this gap already starts to get to the heart of the issues that we consider, and this theory is very useful to us at a technical level.

1.3 Modelling issues

Our goal is to cast the problem of "meaningful" communication between Alice and Bob in a purely mathematical setting. We start by considering how to formulate the problem where the presence of a "trusted third party" would easily solve the problem.

Consider the informal setting in which Alice and Bob speak different natural languages and wish to have a discussion via some binary channel. We would expect that a third party who knows both languages could give finite encoding rules to Alice and Bob to facilitate this discussion, and we might be tempted to require that Alice's statements translate into the same statements in Bob's language that the third party would have selected and vice-versa.

In the absence of the third party, this is unreasonable to expect, though: suppose that Alice and Bob were given encoding rules that were identical to those that a third party would have given them, except that some symmetric sets of words have been exchanged—say, Alice thinks "left" means "right," "clockwise" means "counter-clockwise," etc. Unless they have some way to tell that

these basic concepts have been switched, observe that they would still have a conversation that is entirely sensible to each of them. Thus, if we are to have any hope at all, we must be prepared to accept interactions that are indistinguishable from successes as "successes" as well. We do not wish to take this to an extreme, though: Bob cannot distinguish among Alices who say nothing, and yet we would not classify their interactions as "successes."

At the heart of the issues raised by the discussion above is the question: What does Bob hope to get out of this conversation with Alice? In general, why do computers, or humans communicate? Only by pinning down this issue can we ask the question, "Can they do it without a common language?"

1.4 Our approach and results

In this paper we consider the following simple computational motivation for communication (as viewed from Bob's perspective). Bob is a probabilistic polynomial time bounded machine, whose goal is to decide membership in some (presumably hard) language L. (For instance, Bob's goal may be to detect if his computer has any viruses.) His hope is that by interacting with Alice, he can solve this hard computational problem. We consider two settings within this context.

In the first of the settings, we consider an all-powerful Alice who can decide membership in an arbitrary language. If Alice and Bob understood and trusted each other completely, then this could allow Bob to solve any membership problem! Note that the setting where Alice and Bob understand each other, but don't necessarily trust each other is the setting considered in interactive proofs, and here we have that Bob can decide languages (only) in PSPACE [11, 14]. Our setting is the "dual" one where Alice and Bob don't necessarily mistrust each other, but they definitely don't start off understanding each other. Somewhat strikingly, we show that this setting turns into the same setting as its dual, and that Bob can decide languages (only) in PSPACE.

We show this in two parts: First we show that given an all-powerful (or even a PSPACE-powerful) and sufficiently helpful Alice, Bob can decide membership for any language in PSPACE (see Theorem 2.3). So, Alice manages to communicate the answer to a language membership question to Bob in spite of their lack of a common language to begin with. The crux of this step is to come up with a proper definition of "sufficiently helpful" that helps avoid pitfalls such as expecting to do this with an Alice that doesn't communicate, or just spits out random answers, without restricting Alice so much that the issue of common language simply goes away. Our definition is given in Section 2.2 where we discuss why this is a ("most") reasonable definition.

Next, we show that under any sufficiently strong definition of "lack of common language" between Alice and Bob, Bob can not decide languages outside PSPACE even if Alice is sufficiently helpful (see Theorem 2.5).

Both our proofs above are based on standard "enumeration" arguments. In the case where $L \in$ PSPACE, Bob enumerates potential methods for intepreting Alice's language and rules out incorrect choices by using his power to verify answers given by Alice (using this interpreted version of Alice as a prover). On the other hand, when L is outside PSPACE, our proof uses diagonalization to show that for every Bob there exists an Alice that is helpful (to somebody) and yet communicates misleading things to Bob. The enumeration argument in our positive result yields a somewhat weak positive result. In order to establish communication between Alice and Bob, Bob runs in time exponential in the description length of the protocol for interpreting Alice in his encoding of Turing machines. In Theorem 2.6, we argue that such an exponential behavior is necessitated by our model of "lack of understanding" assuming BPP $\neq$ PSPACE.

Our second setting is motivated by the observation that we would like to be able to establish communication with even an Alice whose computational abilities are bounded by probabilistic polynomial time, with whom the protocol described in Theorem 2.3 is unlikely to succeed. We motivate this setting by supposing that Alice has accumulated some "wisdom," which can be utilized to solve *some* instances of a problem that Bob does not know how to solve. In particular, we allow for the possibility that even if we understood Alice completely, she may still fail to help us solve some instances.

We model Alice's unreliable computational abilities as solving some $L \in \text{MA}$ with observable (non-negligible) probability over some distribution chosen by Bob. We chose this setting for several reasons: perhaps most significantly, we know that the setting is non-trivial – there exist distributions under which seemingly hard computational problems can be solved with reasonable probability (e.g., random satisfiability) – and yet is of some interest, since it is always possible that Alice's heuristic solves some instance on which Bob's heuristics fail. Our goal in this setting will be to find a pair of mappings translating instances of our problem into Alice's language, and translating her responses back into witnesses in our own language.

Our results in this setting again rely on enumeration arguments, but differ somewhat substantially in their analysis. We first prove that, for an Alice whose responses are strongly history-independent, we can efficiently identify mappings under which Alice provides witnesses to instances of our problem at non-negligible rates (see Lemma B.3). Our main theorem in this setting uses a similar construction to show how we can use a few bits of state to maintain a pair of mappings which we tentatively treat as translating to and from Alice's language, with a guarantee that if we use them on "enough" instances, then a helpful Alice will have provided witnesses to a non-negligible fraction (see Theorem 3.4). Again, we observe that these positive results are somewhat weak: since we search through an enumeration, we take time exponential in the lengths of the mappings' representations, and we show in Theorem B.4 that assuming trapdoor permutations exist, superpolynomial time may be necessary to find effective methods for translating in many situations.

2 Communication with an all-powerful Alice

In this section we formalize the computational goal for communication and prove feasibility and infeasibility results for universal communication in this setting. More specifically, we define a notion of a "helpful" Alice, and the goal of a "universal" Bob. We show that there is a universal Bob that can decide PSPACE complete languages when interacting with any sufficiently powerful and helpful Alice. We also prove a matching negative result showing that Bob can not decide languages outside of PSPACE, provided the "language" of Alice is sufficiently unknown. Again crucial to this step is formalizing the concept of the language being unknown.

2.1 Basic Notation

We start by setting up our basic notation for interactive computation (borrowed from the classical setting of interactive proofs). We assume that Alice and Bob exchange messages by writing finite length strings from a binary alphabet on common tape. We assume they are well synchronized[1], i.e., they alternate writing, and they know when the other has finished writing. Thus a history $\mathbf{m}$ of the interaction consists of a sequence of strings $\mathbf{m} = \langle m_1, \ldots, m_k \rangle$ where $m_i \in \{0,1\}^*$. The odd

[1]This and similar syntactic assumptions may already be questioned in the general setting of "intergalactic communication." Indeed these simplify our task, however they do not trivialize the problem and we are optimistic that these can be removed at a later stage.

messages are written by Alice, and the even ones by Bob. The $k+1$th message if written by, say, Alice is a (potentially probabilistic) function of the history thus far given by some function $\mathrm{A}(\mathbf{m})$. Similarly Bob's messages are also (probabilistic) functions of the history and any private inputs he may have. Bob's message on private input x and history $\mathbf{m}$ is denoted $\mathrm{B}(x;\mathbf{m})$.

Conditioned on a history $\mathbf{m}$, Alice's responses in the future may be viewed as a new incarnation of Alice. We use $\mathrm{A}_{\mathbf{m}}$ to denote her future responses and thus $\mathrm{A}_{\mathbf{m}}(\mathbf{m}') = \mathrm{A}(\mathbf{m} \circ \mathbf{m}')$ where $\mathbf{m} \circ \mathbf{m}'$ denotes the concatenation of the histories $\mathbf{m}$ and $\mathbf{m}'$.

At the end of an interaction with Alice, Bob will output a Boolean verdict. $(\mathrm{A}, \mathrm{B}(x))$ will denote the random variable produced by Bob's output following the interaction between Alice and Bob, where Bob has private input x. We will abuse notation in a natural way to let a language L also denote a boolean function: $L(x) = 1$ if $x \in L$ and $L(x) = 0$ otherwise. We say that Alice helps Bob decide a language L if for every $x \in \{0,1\}^*$, it is the case that $\Pr\left[(\mathrm{A}, \mathrm{B}(x)) = L(x)\right] \geq 2/3$.

2.2 Goals of communication

We now define the goals of "universal semantic communication" in our setting. Our general approach is to consider the situation where Bob interacts with some member of a large class $\mathcal{A}$ of Alices, but does not know which specific member of the class he is interacting with. Essentially, we would like Bob to be successful in deciding the language L for every member of the class $\mathcal{A}$. We may also wish to consider what Bob does when Alice does not belong to $\mathcal{A}$.

In order to make this viable, the class $\mathcal{A}$ should only include Alices that are powerful (enough to decide L), and helpful. While the former is easy to formalize, the latter notion is somewhat subtle. One aspect we'd like to capture here is that her ability to decide L should be based on her "external characteristics," namely in her input/output response, but this is still insufficient. For instance suppose that for each round i, Alice has chosen a random bit b_i (known only to her) and then answers any question y with the bit $L(y) \oplus b_i$. In this setting her input/output response at time i represents her ability to decide L – someone who knew some b_i would be able to easily obtain $L(y)$ for any y – but this is clearly not detectable by (poor) Bob. This introduces an additional element that "helpfulness" must capture, namely Alice's behavior as a function of the history of messages thus far.

In the following definition we attempt to formalize the notion of a powerful and helpful Alice by setting what appear to be fairly minimal restrictions on Alice. Roughly, we insist that Alice be able to help *some* Bob$'$ decide L, conditioned on *any* prior history. The requirement that Alice should be able to help some Bob$'$ decide L is clearly necessary if we would like to design a specific Bob that decides L by interacting with Alice. The requirement that this should happen independent of any history does restrict Alice somewhat, but we argue that it is a simple way to overcome issues such as the time-varying Alice described above, and therefore a reasonable "universal" principle.

Definition 2.1 (L-Helpful Alice) *We say that Alice is* L-helpful *if there exists a probabilisitic polynomial time bounded Bob, such that for every prior history* $\mathbf{m}$*, the incarnation of Alice conditioned on the history,* $A_{\mathbf{m}}$*, helps Bob decide* L*.*

We now formalize Bob's goal in this computational setting.

Definition 2.2 (L-Universal) *We say that Bob is a universal decider for a language* L*, or* L*-universal, if it satisfies the following conditions:*

Completeness *If Alice is L-Helpful, then Alice helps Bob decide L and there exists a polynomial p such that for every $x \in \{0,1\}^*$ Bob runs in expected time $p(|x|)$.*

Soundness *For every Alice and all x, the probability that $(A, B(x)) \neq L(x)$ is at most $1/3$.*

We are now ready to state our main theorem in this setting.

Theorem 2.3 *For every PSPACE complete language L, there is a Bob that is L-universal.*

Since the theorem rests crucially on the definition of L-universality, we discuss this definition a little. The completeness condition above is mostly natural given our goals as described earlier: Bob should be able to decide L with the help of any helpful Alice. The only weakness is that we allow Bob's running time to depend on Alice. This is essential to get universality and we prove this formally in Proposition A.1 below. The soundness condition is somewhat strong, and expects Bob not to be fooled by Alice. So in this sense it already reflects some of the concerns of "mistrust of Alice" coming from the setting of interactive proofs. For a positive result, this is certainly a good feature to have, but later when we try to argue the "non-existence" of L-universal Bob for L not in PSPACE, this condition makes this part trivial (given the previous work on IP).

We assert first that the soundness condition prescribes a natural precaution that Bob should take when trying to seek computational wisdom from third parties even if he trusts them. ("Trust, but check.") Thus while it is possible to consider the definition without the soundness restriction, we don't believe it would really serve the purpose of being a universal decider for a language.

However, this discussion turns out to be moot and this restriction is essentially just aesthetic. We prove this by showing that if we consider any sufficiently broad class of helpful Alices $\mathcal{A}$, then for every probabilistic polynomial time bound Bob, there exists an Alice in the class $\mathcal{A}$ such that the language (promise problem to be more precise) that Alice helps Bob decide is contained in PSPACE. Of course this theorem in turn relies on the definition of "sufficient broadness" above, which we specify next.

Definition 2.4 (Semantically closed class) *We say that a class of Alices $\mathcal{A}$ is semantically closed if for every invertible function f where f and f^{-1} are computable in polynomial time, and for every member $A \in \mathcal{A}$, it is the case that every incarnation A^f, given by $A^f(f(m_1), \ldots, f(m_k)) = A(m_1, \ldots, m_k)$, is also in $\mathcal{A}$.*

The incarnation A^f is simply A who uses the dictionary f to convert from A's language to hers and the dictionary f^{-1} to convert back. Note that A^f may have arbitrary behavior outside the range of f! We now state our impossibility result for L-universal deciders for $L \notin$ PSPACE.

Theorem 2.5 *Let L be a language that is not in PSPACE. Let $\mathcal{A}$ be a semantically closed class of L-helpful Alices. Then for every probabilistic algorithm B, there exists an Alice $A \in \mathcal{A}$ such that B fails to decide L with the help of A.*

The insistence that we deal only with semantically closed classes is somewhat broad so we discuss the definition next. This definition is consistent with our goal that we make few assumptions about language, and this definition suggests that every polynomial time computable translation of

a "natural language" is also natural. This is, of course, somewhat broad, yet we don't believe this is the aspect that limits the power of universal semantics. This is merely a definition under which we can *prove* the limitations.

Perhaps a more distressing consequence of the level of generality we seek is that the running time of Bob is exponentially long in the description length of the shortest protocol for interpreting Alice (in Bob's encoding). Notice that with a trusted third party, Bob would have had only a polynomial dependence on the encoding of this protocol. The following theorem asserts that this is necessary:

Theorem 2.6 *Let L be a PSPACE-complete language, let $\mathcal{A}$ be a semantically closed class of L-helpful Alices, and let $\mathcal{A}|_c \subset \mathcal{A}$ be the subclass that help some protocol running in time $O(n^c)$. Then, unless PSPACE=BPP, if a probabilistic algorithim Bob decides instances of L using the help of an arbitrary Alice in $\mathcal{A}|_c$ in time $O(n^k)$, Bob's running time must also be exponential in the description length of the shortest protocol that is helped by Alice in time $O(n^c)$.*

To prove this theorem we exhibit an infinite family of Alices such that each Alice helping a protocol of description length $k+O(1)$ expects a different "password" of length k. Since Alice is a black-box to Bob, a Bob running in time subexponential in k does not have time to try every password, and thus must output a verdict without Alice's help.

The proofs of Theorems 2.3, 2.5 and 2.6, as well as the statement and proof of Proposition A.1 are in Appendix A.

3 Communication with a computationally bounded Alice

The setting of a powerful Alice, interacting with Bob to help him decide some language is not uncommon. Indeed the tasks of detecting viruses, or filtering spam, may well correspond to the case where Bob is trying to use Alice to solve *undecidable* problems, and our results may be viewed as suggesting limitations in such settings (as opposed to "enabling" powerful languages to be decided by a polynomial time B). However this scenario certainly does not model all possible settings for communication.

In this section we consider a setting where Alice is less powerful. More importantly we consider scenarios where Alice can *fail* to decide the language of interest on some instances (or fail to prove some of her claims). For the sake of simplicity, we fix one such setting, where the language L to be decided is in MA (i.e., instances in the language have short witnesses verifiable by some probabilistic verifier). We consider the setting where Bob is getting inputs drawn from some unknown distribution $\mathcal{D}$ for which he would like to decide membership in L. We assume that he is only able to solve this problem on a negligible fraction of instances, while Alice is able to solve the problem on a non-negligible fraction of instances. We study how to define and effect communication in this setting.

First some motivation: We are roughly interested in cases where Alice may be able to offer "heuristic" help to Bob. One can imagine various ways in which Alice may be getting this ability to help: for example, Alice may have collected some "wisdom" over the years modelled as some small non-uniformity that allows her to solve instances of the problem.

We stress that it is not straightforward to translate the arguments of the previous section to the current setting. First, the class of languages is not closed under complement and Bob can not ever verify "non-membership." Furthermore, Alice is not perfect and only finds witnesses on some

(non-negligible fraction) of the inputs. So to allow Bob to interact with her, we need to extend Bob into one that is trying to decide a sequence of instances $x_1, \ldots, x_t$ drawn from some distribution $\mathcal{D}$ on which Alice has a good chance of deciding membership (and finding witnesses). We would like Bob also to develop this ability, by using Alice's help. We are now ready to define helpful Alices and Bob's "universality" goal in this setting.

Definition 3.1 (Helpful under a distribution) *For a language $L \in$ MA and ensemble of distributions $\mathcal{D} = \{\mathcal{D}_n\}$, we say that Alice is* $(L, \mathcal{D})$-Helpful *(or helpful for L under distribution $\mathcal{D}$) if $\exists f, g$ computable in polynomial time and a polynomial p such that for all n and $\mathbf{m}$,*

$$\Pr_{x \in_R \mathcal{D}_n}[(g \circ A_{\mathbf{m}} \circ f)(x) \text{ is a witness for } x \in L] \geq \frac{1}{p(n)}$$

The above definition allows Alice to be ineffective on a large fraction of the instances, as long as she produces witnesses on a non-negligible fraction. Her ability to produce the witness is modelled by the "instance translator" f and the "witness translator" g. Bob's goal is to find such translators. He is expected to do this after seeing a sequence of instances $x_1, \ldots, x_t$. However (for justifiable reasons) we have to allow him to keep refining his hypotheses over time. So instead we only require him to be able to produce witnesses to the instances given to him with non-negligible probability. The following two definitions describe Bob's "operational" behavior and his goal in terms of finding membership witnesses.

Definition 3.2 (Online stateful verifier) *B is said to be an online stateful verifier if for any Alice A, B interacts with A on a sequence of instances $x_1, \ldots, x_t$ producing a sequence of outputs $w_1, \ldots, w_t$ and states $s_1, \ldots . s_t$ where Bob's computations during the ith phase depend only on s_{i-1}, x_i (and the responses from A). We say B has $m(t)$ bits of state if $s_i \in \{0,1\}^{m(i)}$ for every i.*

Definition 3.3 ($(L, \mathcal{D})$-universal) *For a language $L \in$ MA and ensemble of distributions $\mathcal{D} = \{\mathcal{D}_n\}$, we say that an online stateful verifier B is* $(L, \mathcal{D})$-universal *if for every $(L, \mathcal{D})$-Helpful Alice A the following hold when $x_1, \ldots, x_t$ are drawn independently at random from $\mathcal{D}_n$:*

1. *B runs in time polynomial in n, t (where the running time may depend on A).*
2. *There exist polynomials P, Q such that with all but negligible probability, we have that for all $t \geq Q(n)$, the outputs produced by B on interacting with A form witnesses to at least a $1/P(n)$ fraction of the x_i's.*

Our main theorem about the distributional setting within MA is the following:

Theorem 3.4 *For every $L \in$ MA and ensemble of distributions $\mathcal{D}$, there exists an $(L, \mathcal{D})$-universal (online stateful) verifier Bob with $O(\log t + \log n)$ bits of state.*

We prove this theorem in Section B of the appendix. As in the PSPACE setting, the Bob B takes time exponential in the description length of the shortest possible "dictionary" between Alice and Bob. This also appears inevitable, though in this case we assume the existence of public-key encryptions (or trapdoor one-way permutations) for the proof—see Theorem B.4 for details.

4 Conclusions and directions for future work

The two previous sections give a natural progression where we define the goal of communication to be that of extracting "computational wisdom/advice" and then show how this can be achieved without a common language. In both cases, the basic idea used to converge to some common understanding is that one of the players, Bob, makes various hypotheses about Alice, and then uses his own power of verification to rule out the incorrect hypotheses.

It would be natural at this stage to ask whether this implies "semantic communication." It is clear that if a probabilistic polynomial time algorithm Bob uses the communication with Alice to decide a language L that is not in BPP, then Bob must have gained something from the communication. This does not (and should not) lead to the conclusion that Bob has learned Alice's language. However, this does give a functional version of such a theorem. To the extent that Bob knows, Bob has learned to communicate with Alice, and he can convince himself of this fact by verifiably deciding membership in some language L that is, or is believed to be hard. Of course, in the setting where the language L is in BPP, as may be the case if PSPACE=BPP, then Bob may be fooling himself and we can't rule this out.

So this leads naturally to the question, what if Alice is no more powerful than Bob? What is the motivation for Bob to talk to Alice then? Real life is filled with examples of such interactions leading to some mutual benefit. For example, Bob may be a cryptographer and Alice a number theorist, and together they may discover some new cryptographic schemes (or break an old one). Modelling the computational benefit of scenarios is itself an interesting task, and may involve more careful modelling of how thoughts and ideas are formulated in our brain. We don't attempt this here, but we note that the work of Blum and Williams [4] seems to shed light on this topic. (On a related note, additional work along these lines by Blum et al. [2] may also lead to methods by which one can state formal results about the work of Freudenthal [7] and in particular about his language LINCOS.)

That being said, it turns out that we can avoid these hard questions raised above and start to suggest ways in which Alice and Bob may try to understand each other, even when they are totally symmetric. In particular, we suggest two goals that Bob may have when interacting with such an Alice: (1) Bob may be interested in exerting some "control" on Alice, and in particular on how Alice responds to him. For instance, he may send Alice a function f and an input x and hope she responds with $f(x)$. If Alice starts to respond in such a manner it suggests that she understands Bob. (2) Alice may pose a sequence of hard problems to Bob and then help him solve them. Note that using cryptographic primitives this is feasible for Alice even though she is no more powerful than Bob: for a one-way function f described by a program to compute f, she picks random inputs x and sends $(f, f(x))$ to Bob. Bob's goal is to find x and this is hard for him, but not Alice.

Indeed, combining the two goals above, we suggest a simple mechanism for Alice and Bob to try to communicate with each other. Bob sends f to Alice along with the instruction: "Pick a random x, compute $f(x)$ and send it to me. Then send x." If Alice does this to Bob, Bob can try to check that she chose x at random, e.g., by trying to compute x or later, when Alice sends x to him, attempting to compress x. Bob is then also convinced that she does know x, and hence must have computed $f(x)$ since otherwise finding an x that maps to $f(x)$ is infeasible, thereby implying that she understood at least the part of the instruction which said "compute f." By setting a conversation such as the above as his goal, Bob can now start enumerating hypotheses about Alice, and start rejecting the incorrect ones, thereby learning her language. (Note that he should start

every such conversation with the preamble: "Here is a game we will play. If we manage to play this, we can become friends ...")

Among other things, the protocol above suggests a means by which Alice can convince Bob that she is "intelligent enough." She can compute functions he sends her and can toss random coins—or at least do things that he can't predict. This seems to lead to a novel test of "intelligence," a question suggested to us by Steven Rudich [12], where Alice convinces Bob by challenging him with puzzles and then revealing the solutions. At this point, a comparison with Turing's test [16] is unavoidable; we only remark that it seems an appropriate time to consider what testable properites we might wish to require of an intelligent individual, other than indistinguishability. For that matter, we might also reconsider with respect to *what* we should require this individual be indistinguishable—demanding knowledge of English is clearly not a satisfactory universal principle.

This entire process could be carried out non-interactively, with Alice sending messages containing f, $f(x)$ and a computation history of f on input x in sequence to Bob. In particular if f is a pseudorandom generator and x is its seed, Bob finds an appealing simple test, motivated by the well-known connections between compression and prediction (e.g., [5, 10, 6, 13]): he tries to "compress" Alice's string within his resource bounds, but fails to compress much after seeing the first $|f| + |f(x)|$ bits of her string. But then once she sends the last portion of the message, it should be easy for Bob to compress the entire triple $(f, f(x), history(f, x))$ to just $|f| + |x| + O(1)$ bits. This highly non-monotone amount of compressibility tells us three things: it tells us that the message is almost assuredly not random; that the message exhibits some computationally hard problem (puzzle); and perhaps most significantly, that Alice can explain the solution to this puzzle to us. Thus, we may justifiably view this as a sign of intelligence.

In terms of questions to be addressed in the future, we can think of several directions and we mention a couple here. First, the entire setup so far has been asymmetric in terms of responsibilities: the responsibility of learning the common language has been assigned to Bob. Must we assume this, or is it reasonable to expect Alice to be more proactive in attempting to understand Bob as well? We are not sure, but have not been able to suggest any computational settings where the asymmetry is truly helpful. Similarly, is there some way we could leverage a more proactive Alice to obtain a better protocol? Finding such examples would be useful.

Perhaps a more important question is whether the "exponential" behavior of Bob, with respect to the length of his shortest protocol for interpreting Alice, is always necessary. While we did prove in Theorem 2.6 that such a dependence is necessary, this is only true when Alice (or her language) is unrestrictedly broad and any polynomial time computable transformation of her language is considered semantically equivalent. But this allows for the possibility that there may be other reasonable ways of restricting the class of languages that one is allowed to deal with. Such restrictions may lead to universal Bobs who are efficient even in terms of the length of the protocol. More importantly they may suggest some insight into how languages are/ought to be structured to enable easy learning. This would be quite fruitful, if it is at all feasible.

Acknowledgements

We would like to thank Manuel Blum, Silvio Micali, Ronitt Rubinfeld, and Steven Rudich for their encouragements and many illuminating conversations.

References

[1] Laszlo Babai and Shlomo Moran. A randomized proof system and a hierarchy of complexity classes. *JCSS*, 36:254–376, 1988.

[2] Manuel Blum, Avrim Blum, Lenore Blum, Steven Rudich, Ryan Williams, and Luis von Ahn. Mathematical foundations for understanding and designing conceptualizing strategizing control (CONSCS) systems. Available at `http://www.cs.cmu.edu/~mblum/search/`, 2004.

[3] Manuel Blum and Sampath Kannan. Designing programs that check their work. In *Proc. 21st STOC*, pages 86–97, 1989.

[4] Manuel Blum and Ryan Williams. Applying compuational complexity theory and cryptography to the pursuit of concept understanding: An overview of current work. Available at `http://www.cs.cmu.edu/~mblum/search/`, 2006.

[5] Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987.

[6] Raymond Board and Leonard Pitt. On the necessity of occam algorithms. In *Proc. 22nd STOC*, pages 54–63, 1990.

[7] Hans Freudenthal. *LINCOS: Design of a Language for Cosmic Intercourse.* North-Holland Publishing Company, Amsterdam, 1960.

[8] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[9] Leonid A. Levin. Universal search problems. *Probl. Inform. Transm.*, 9:265–266, 1973.

[10] Ming Li, John Tromp, and Paul Vitányi. Sharpening occam's razor. *Inf. Process. Lett.*, 85(5):267–274, 2003.

[11] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, October 1992.

[12] Steven Rudich. Personal communication.

[13] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[14] Adi Shamir. IP = PSPACE. *JACM*, 39(4):869–877, 1992.

[15] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[16] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[17] Andrew C.-C. Yao. Some complexity questions related to distributed computing. In *Proc. 11th STOC*, pages 209–213, 1979.

A Proofs of theorems in the unbounded setting

We start with the proof of Theorem 2.3 that claims the existence of a universal Bob for PSPACE-complete languages.

Proof (of Theorem 2.3): Recall that we need to construct a Bob B that can interact with an Alice A to compute $L(x)$. We start with an overview first.

Note that since Alice is helpful there exists some Bob B^* that can decide L with Alice's help. The Bob B we construct enumerates all Bobs $\tilde{B}$ hoping to guess B^*. If $\tilde{B} = B^*$ then Bob gets an "oracle prover" with the power to answer any PSPACE problem. This allows our Bob B to simulate an interactive proof of the fact that $x \in L$ (using the fact that PSPACE in contained in IP [11, 14], where the prover only needs be in PSPACE). If $\tilde{B} \neq B^*$, then since B can never be convinced of a wrong assertion (of the form $x \in L$ when actually $x \notin L$), this simulation does not lead B astray. It simply rules out $\tilde{B}$ as the guess and moves on to the next guess for B^*. In the details below we attempt to ensure that B's running time is not too large when Alice is helpful, and also to ensure termination (though not in polynomial time) when Alice is not helpful.

Construction of B: Let V_L denote the verifier for the language $L' = \{(x, b) | L(x) = b\}$. Let P_L denote the "optimal binary prover" for this language, i.e., $P_L(x, b, \mathbf{m}, y)$ gives the Boolean answer to question y that is most likely to cause V_L to accept on input (x, b) after history of interactions being $\mathbf{m}$. Note the P_L is computable in PSPACE and so there exists a reduction f_L such that f_L reduces the optimal prover's computation to a decision problem in L. In other words $P_L(x, b, \mathbf{m}, y) = L(f_L(x, b, \mathbf{m}, y))$. We show how to use f_L and V_L to help construct Bob B. Our final Bob B runs two Bobs B_1 and B_2 in parallel and halts whenever either of them halts and outputs the answer of the one that halted.

B_1 is simply the Bob that runs in time $2^{p(|x|)}$ for some polynomial p to decide if $L(x) = b$. We hope not to use this Bob much with a helpful Alice. It is used to provide a bound on the number of times we simulate the interaction between P_L and V_L, and thereby permits us to guarantee a negligible probability of failure.

B_2 is the crucial element in our construction. On input x, B_2 enumerates all triples $(\tilde{B}, t, b)$ where $\tilde{B}$ is a probabilistic algorithm, t is an integer, and b a bit. (We don't specify the exact sequence of the ordering here, only that it follows the same outline as in Levin [9]. Specifically we enumerate pairs $\tilde{B}, t$ in phases: In the ith phase the first i $\tilde{B}$s are enumerated with the t corresponding to the jth Bob being 2^{i-j}. This ensures that for any desired pair $\tilde{B}, t$ it is encountered early enough.) For a fixed such triple, let $\tilde{B}|_t$ denote $\tilde{B}$ restricted to run in time t. B_2 attempts to use $(A, \tilde{B}|_t(\cdot))$ as an oracle for PSPACE to simulate the interactive proof to verify if $L(x) = b$ as follows:

- It simulates the verifier V_L in probabilistic polynomial time.
- To simulate the optimal prover it computes $P_L(x, b, \mathbf{m}, y) = L(f_L(x, b, \mathbf{m}, y))$ by simulating the interaction of $(A, \tilde{B}|_t(x, b, \mathbf{m}, y))$.

We repeat this simulation $n + 18p(n)$ times, and we keep count of how many times the conversation halts with V_L accepting after t rounds of interaction. If the majority of the simulations accept, then B_2 halts and outputs $L(x) = b$; if not it continues on to the next triple $(\tilde{B}, k, b)$.

Analysis: We now analyze the completeness and soundness of B. The soundness of B is a straightforward consequence of the soundness of the verifier V_L, the verifier for the language L'; notice that we finish within $2^{p(n)}$ phases, where we have reduced the probability of failure in a

phase to $2^{-p(n)} \cdot \text{negl}(n)$. Thus, if V_L halts and accepts a pair (x, b) with noticeable probability, it must be that $L(x) = b$. So this yields that the probability that B_2 halts and produces the wrong answer is negligible. B_1 never produces the wrong answer so we always have the right answer.

To finish up, we need to show that if A is L-helpful then B halts in some fixed polynomial time and outputs $L(x)$. To see this note that B^*, the Bob that makes A helpful, must be enumerated sometime by B_2, with $t = n^k$, if B^* runs in time n^k. By choosing the enumeration ordering appropriately, we can ensure that the total time spent by B_2 to get to this pair is $O(t^2 p(n)) = O(n^{2k} p(n))$ (since we are simulating B^* for t^2 rounds). At this stage B_2 has access to an optimal prover for PSPACE which will convince it to accept $L(x) = b$, for the right choice of b. This ensures completeness of the Bob B. ■

We remark that, if B_A denotes the shortest efficient B helped by A and k_A denotes the running time of B_A, then it is also easy to construct a universal Bob achieving running time $O(2^{4|B_A|} n^{4k_A} p(n))$ for every L-helpful A: let the jth Bob in phase i run for $i - j + 1$ steps instead. As an asymptotic function of n, this is a little worse, but the dependence on $|B_A|$ is (only) singly exponential. We will see in a moment that this exponential dependence cannot be removed entirely.

The next proposition proves that we can not improve Bob's running times over all L-helpful Alices to polynomials with bounded exponents.

Proposition A.1 *If for a PSPACE complete language L, there exists a B and a k such that for every L-helpful A, B decides L with the A's help in time $O(n^k)$, then $BPP = PSPACE$.*

Proof The proof is a simple padding argument. Assume that such a Bob B exists with running time $O(n^k)$ (that achieves exponentially small error). Consider an Alice who decides $\{x10^{|x|^{2k}} : x \in L\}$. Clearly such an Alice is helpful. Now consider the task of deciding if $x \in L$ when $n = |x|$ is sufficiently large. While trying to decide this, Bob only has time to query Alice with queries of the form $y10^{|y|^{2k}}$ for $|y| = \sqrt{n}$. Thus such a Bob gives a (probabilistic) Cook reduction from L instances of length n to at most n^k instances of L of length $O(\sqrt{n})$. We can now use the same Bob recursively to solve the smaller problems. Analyzing the resulting recurrence (and using the fact that the probabilities of error are negligible), we get a net probabilistic running time of $O(n^{2k})$ to decide L, where L was PSPACE complete. ■

Next we show that Bob's running time, as a function of the length of the shortest protocol for receiving help from Alice, really needs to be exponential, assuming BPP is different from PSPACE.

Proof (of Theorem 2.6): Let any $A \in \mathcal{A}|_c$ be given. We start by constructing a family of L-helpful Alices $\{A_\sigma\}_\sigma$ for every $\sigma \in \{0,1\}^*$, where A_σ behaves as follows: if her input is not of the form $\sigma \circ x$, she responds with the empty string. Otherwise, she constructs the following "edited history," and responds as A would have responded given this message history: she removes all pairs of consecutive messages in which Bob's message was not of the form $\sigma \circ x$, and for all remaining messages from Bob, she replaces $\sigma \circ x$ with x.

Clearly, for the Bob B^* that is helped by A and runs in time $O(|x|^c)$, the Bob B^*_σ who converts each query to $\sigma \circ x$ and otherwise computes the same function as B^* has description length $|\sigma| + O(1)$ and runs in time $O(|x|^c)$. Furthermore, every A_σ is in the semantic closure of A, and thus is clearly also in $\mathcal{A}|_c$.

Now suppose there is a L-universal Bob B whose probabilistic running time when interacting with any A_σ on input x is $2^{o(|\sigma|)}|x|^k$, i.e., $O(|x|^k)$ and subexponential in $|\sigma|$. We show that B can be used to decide L, which is assumed to be PSPACE-complete.

We simulate B interacting with an Alice who always responds with the empty string. Notice that if B runs for less than $2^{2k \lg |x|}$ steps, then there is some σ of length $2k \lg |x|$ such that A_σ is consistent with this set of responses. Since B has running time $O(|x|^{1.5k})$ for all such A_σ, for all sufficiently large x B must halt without receiving a response from Alice other than the empty string, and in this case we can output Bob's verdict. Since all A_σ help Bob decide L, this is sufficient. ■

Finally we prove Theorem 2.5 which shows that even if the soundness condition were dropped from the definition of L-universality, Bob would still be essentially restricted to accepting languages in PSPACE.

Proof (of Theorem 2.5): We prove this theorem in two steps. First we show that if a Bob B is L'-universal (meeting both the completeness and soundness conditions), then L' must be in PSPACE. We then show that if for some language $L \notin$ PSPACE, Bob meets the completeness condition for L-universality, but is unsound against some Alice, then he is unsound against some helpful Alice.

Step 1: Let B be universal and let A^* be any L'-helpful Alice; since B satisfies the completeness condition, there is some polynomial p such that the interaction $(A^*, B(x))$ halts within $p(|x|)$ steps in expectation for every x.

Now, given x, consider A_x, the Alice who sends B a message that maximizes the probability that B halts within $6p(|x|)$ steps. Since Markov's inequality says that when B interacts with A^* the probability that B takes more than $6p(|x|)$ steps is less than $1/6$, the interaction $(A_x, B(x))$ also halts within $6p(|x|)$ steps with probability greater than $5/6$ (since A_x maximizes this).

Since B satisfies the soundness condition, we find that $(A_x, B(x)) = L'(x)$ within $6p(|x|)$ steps with probability greater than $1/2$. We now observe that in polynomial space, on input x, we can calculate $\Pr[(A_x, B(x)) = 1$ within $6p(|x|)$ steps$]$ where we will decide L' if we accept precisely when this probability is greater than $1/2$.

Step 2: We now consider a language $L \notin$ PSPACE and a Bob B that satisfies the completeness condition of L-universality. Since B can not be sound (by Step 1), we have that there exists A and x such that B halts on interacting with A and $(A, B(x)) \neq L(x)$. We now convert such an Alice A into a helpful one A'_x: Let $\tilde{A}$ be any L-helpful Alice in $\mathcal{A}$ with $\tilde{B}$ being a Bob that can decide L with the help of $\tilde{A}$. Let ℓ be the length of the longest question asked by $B(x)$ when interacting with A. Then A'_x answers queries of length at most ℓ as A does, and answers queries of the form $0^{\ell+1} \circ y$ as $\tilde{A}$ answers y. Clearly A'_x is helpful (to the Bob B'_x who, on input w, queries $0^{\ell+1} \circ w$ to simulate $\tilde{B}$). Yet B returns the wrong answer on x when interacting with A'_x violating the completeness condition. Finally note that A'_x is in the semantic closure of $\tilde{A}$. ■

B Proofs in the Computationally Bounded Setting

We start with the following definition of MA.

Definition B.1 (MA) MA *is the class of all* efficiently provable *languages, i.e., languages L for which there is an efficient verifier V and a polynomial q such that*

1. $\forall x \in L$ $\exists$ *a circuit* P_x *such that* $|P_x| \leq q(|x|)$ *and* $\Pr[(P_x, V(x)) \text{ accepts}] \geq 2/3$.

2. $\forall x \notin L$ $\forall$ *circuits* P^* *with* $|P^*| \leq q(|x|)$ $\Pr[(P^*, V(x)) \text{ accepts}] \leq 1/3$.

The above definition is easily seen to be equivalent to the more standard interactive definition. In the following, when we refer to a "witness" w to an instance $x \in L$, we refer to a circuit P_x as above, i.e., one that can be probabilistically verified to satisfy the condition that $\Pr[(P_x, V(x)) \text{ accepts}] \geq 2/3$.

In this section we prove Theorem 3.4 which asserts the the existence of an $(L, \mathcal{D})$ universal Bob in the MA setting. To prove this theorem, we first need an intermediate notion:

Definition B.2 (History independent) *We say that Alice is* history independent *if* $\exists$ *a fixed ensemble of distributions* $\{A'_x\}$ *such that* $\forall x, y, \mathbf{m} \Pr[A_{\mathbf{m}}(x) = y] = \Pr[A'_x = y]$

Our route to proving Theorem 3.4 will go through history independent Alices. In particular the following lemma shows how to construct a (stateless) Bob that is universal for such an Alice.

Lemma B.3 *For any* $L \in MA$ *there is an algorithm* $B_L^{(\cdot)}$ *such that for every distribution* $\mathcal{D}$ *and every history independent Alice* A *who is* $(L, \mathcal{D})$*-helpful the following holds when* B_L *is given a sequence* $S = \langle x_1, \ldots, x_t \rangle$ *of independent samples from* $\mathcal{D}_n$*: With non-negligible probability* $(A, B_L(S)) = (f, g, p)$ *such that*

$$\Pr_{x \in_R \mathcal{D}_n}[(g \circ A \circ f)(x) \text{ is a witness for } x \in L] \geq \frac{1}{p(n)}$$

Proof sketch Consider any enumeration of all triples $\{(f_i^0, g_i^0, p_i^0)\}$ where f_i^0 and g_i^0 are polynomial-time functions and p_i^0 is a polynomial. We can use this enumeration to construct a new enumeration $\{(f_i, g_i, p_i)\}$ in which which every triple occurs infinitely often as follows: for $k = 1, 2, \ldots$ we loop over the elements (f_i^0, g_i^0, p_i^0) for $i = 1, 2, \ldots, k$; it is easy to verify that in this enumeration, the element in index i will appear again by index $2i$. It is also easy to guarantee that up to index i, our polynomials (in the running times and the p_i compnents) are $O(in^{\sqrt{2i}})$.

We observe that given access to S and A, since Alice is history independent, we can efficiently test the hypothesis "$\Pr_{x \in_R \mathcal{D}_n}[(g_i \circ A \circ f_i)(x)$ is a witness for $x \in L] \geq 1/p_i(n)$:" it follows from Hoeffding's inequality that if this statement holds, then if we evaluate $g_i \circ A \circ f_i$ on $\Omega(p_i(n)^2)$ samples from S, then with high probability we will accept witnesses to at least a $1/2p_i(n)$ fraction of our samples. Likewise, if it were the case that $\Pr_{x \in_R \mathcal{D}_n}[(g_i \circ A \circ f_i)(x)$ is a witness for $x \in L] < 1/4p_i(n)$, then the probability that we would accept witnesses to a $1/2p_i(n)$ fraction of our samples is low. Thus, $B_L^{(\cdot)}$ walks through our enumeration, testing A on each triple until some triple at index i^* passes, which it outputs. Again, under the assumption that Alice is helpful under $\mathcal{D}_n$, we know that some triple at some index N should pass; thus, we anticipate that we will output a triple at some $i^* \leq N$. Clearly, since Alice is assumed to be history independent, this triple is satisfactory.

Again, the issue is that this index N is unbounded a priori. This time, we will use $\Theta(p_i(n)^2 ni)$ samples so that the probability that we fail on the ith experiment is at most $2^{-\Omega(ni)}$; thus, the probability that some experiment fails and we output an invalid triple is at most 2^{-n}. Observe now that the running time of the ith experiment is bounded by $O(i^4 n^{3\sqrt{2i}+1})$. Moreover, thanks to

our choice of enumeration, we can now verify that our running time is polynomial in expectation: the probability that we only succeed after the ℓth time we test index N (by index $2^\ell N$) is at most the probability that the $(\ell-1)$th test failed, which occurs with probability $2^{-\Omega(n2^\ell N)}$. Since the total running time of all tests up to index $2^\ell N$ is at most $O(2^{5\ell}N^5 n^{3\sqrt{2^{\ell+1}N}+1})$, we find that the expected running time is dominated by the time to find and test index N the first time, which is polynomial as before. ■

We needed the strong history independence assumption so that our experiments on Alice's abilities to respond to instances from a distribution in the past would be representative of her ability in the future; we remove this assumption below to prove Theorem 3.4.

Proof sketch We will use a construction similar to the one in Lemma B.3 to find a triple (f, g, p) such that under f and g, Alice provided witnesses to a $1/2p(n)$ fraction of our instances drawn from $\mathcal{D}_n$, except that instead of drawing our samples from S, we will use the instances provided as input, and we will output Alice's response. Observe that between samples, we only need to remember our index in the enumeration, how many samples we have tested the current mappings with, and how many of these samples were successful. The more substantial difference is that we will separately track the index in the enumeration i and the number of "rounds" of testing we have conducted j; these are no longer the same because we no longer output the first triple that passes a test. In particular, when a triple fails, we increment both i and j, but when one passes, we only increment j. The number of samples per round will now be $\Theta(p_i(n)^2 nj)$ so that the total probability of failure on any round is at most 2^{-n}. Since these values will all be polynomially large in n and t, it is clear that we can store them using $O(\log t + \log n)$ bits. It is also easy to see that we will still be efficient with respect to a helpful Alice. Finally, we will also use a slightly different enumeration in which the p component is nondecreasing; notice that when Alice is helpful, a suitable tuple still exists.

Thus, we only need to verify that B produces witnesses with non-negligible probability. Clearly, if N is the index of the pair of mappings for which Alice is helpful, we cannot hope to have output *any* witnesses unless $j \geq N$. Recall that the number of samples in round j is $Cp_i(n)^2 nj$ for some $i \leq N$ and constant C. Observe that there are at most $N-1$ failed tests, (before we reach index N) and so suppose that the last round in which a test fails is round $N-1+\ell$. Now, in the worst case, (since the p_i components are nondecreasing) the test at $i=1$ passed ℓ times, the next $N-1$ tests failed, and we recieved the maximum number of failures permitted at index N (before recieving our polynomial number of successes). In this case, we obtained $Cn\frac{\ell(\ell+1)}{4}p_1(n)$ witnesses and if every sample in a failed test did not yield a witness, we lost at most $Cn\left(\frac{N(N+1)}{2} + N\ell\right)p_N(n)^2$ samples; overall, our success rate was at least $\left(1 + \frac{2p_N(n)^2(N(N+1)+2\ell N)}{p_1(n)\ell(\ell+1)}\right)^{-1}$ which is an increasing function of ℓ for $\ell > 0$. Thus, if we have at least N tests (if there are at least $Q(n) = Cnp_N(n)^2 N(N+1)/2$ samples) then our success rate must be at least $1/P(n) = 1/4(N+1)^2 p_N(n)^2$. ■

Once again, our search through the enumeration for an efficient mapping with which to converse with Alice would not by any means itself be regarded as "efficient," in terms of the length of the desired functions' shortest descriptions. The only saving grace here is that this length is regarded as constant with respect to the lengths of the instances being discussed, and thus we manage to ignore it in our notions of efficiency. We are content to ignore it in this general setting because there is strong reason to suspect that this sort of efficiency is unachievable in many settings of interest:

Theorem B.4 *Let any $L \in NP$ be given such that $\forall x, y \in L$ $\{w : w \text{ is a witness of } x\} \cap \{w : w \text{ is a witness of } y\} = \emptyset$ and let any distribution $\{\mathcal{D}_n\}$ be given such that some helpful and efficient history independent Alice A_0 exists for L under $\{\mathcal{D}_n\}$ and $\forall x \in L_n$ $\Pr_{y \in_R \mathcal{D}_n}[x = y] < \text{negl}(n)$. Let (G, E, D) be a public key encryption scheme such that private keys of length k are secure against non-uniform key dependent adversaries running in time $O(t(k))$ for some $t(k) = k^{\omega(1)}$. Then there is an efficient and history independent A who is helpful for L under $\{\mathcal{D}_n\}$ using mappings f_0 and g_0 that have length $O(k)$ such that any B for which $(A, B(0^n)) = (f, g, p)$ such that f, g are polynomial time functions and with all but negligible probability*

$$\Pr_{x \in_R \mathcal{D}_n}[(g \circ A \circ f)(x) \text{ is a witness for } x] \geq \frac{1}{p(n)}$$

must run for $\omega(t(k))$ steps.

Proof sketch Observe first that for $(SK, PK) \in G(0^k)$, if we put $g_0(x) = A_0(D(SK, x))$ take f_0 to be the identity function, then $A(x) = E(PK, x)$ is helpful, efficient, and history independent. Moreover, g_0 and f_0 can be represented in $O(k)$ bits.

Suppose to the contrary that Bob took time $O(t(k))$ and that for all $A(x) = E(PK, x)$ with $(SK, PK) \in G(0^k)$, with all but negligible probability Bob output (f, g, p) as above. We construct an adversary who breaks the indistinguishability of (G, E, D) as follows: we will use our advice to fix coin tosses for B so that for all PK, there will be at least one for which B successfully outputs (f, g, p) as above. Note that we can test whether or not a given triple is valid; the adversary will select (f, g, p) to be the first such tuple output by B that passes the test, using $E(PK, x)$ to simulate $A(x)$. We will fix coin tosses to be used in this simulation of A as well, so that for each PK, B will output a fixed, valid triple.

Since $\forall x \in L_k$ $\Pr_{y \in_R \mathcal{D}_k}[x = y] < \text{negl}(k)$ but the sets of witnesses for instances x and y in L are disjoint and $\Pr_{x \in_R \mathcal{D}_k}[(g \circ A \circ f)(x) \text{ is a witness for } x] \geq 1/p(k)$, we know that for k sufficiently large there must exist $x_0 \neq x_1 \in L_k$ for which $\Pr[(g \circ A \circ f)(x_1) \text{ is a witness for } x_1] \geq 1/p(k)$ and $\Pr[(g \circ A \circ f)(x_0) \text{ is a witness for } x_1] \leq 1/2p(k)$.

Let $m_0 = f(x_0)$ and $m_1 = f(x_1)$, and let V be the polynomial-time verifier function for L; the adversary first checks if $V(x_1, g(c))$ accepts on its input ciphertext c, outputs "1" if so, and outputs "0" otherwise. Clearly, by our choice of x_0 and x_1, the adversary achieves an advantage of $1/2p(k)$ here, contradicting the security of the encryption scheme since g, V, and E are all guaranteed to be polynomial time functions and $t(k) = k^{\omega(1)}$. Thus, it must be that B takes time $\omega(t(k))$. ■

Thus, in this case, B has running time superpolynomial in the representation lengths of the functions B is searching for. Still, it should be noted that there is a tremendous gap between the superpolynomial time requirements suggested here and the exponential time taken by our search through our enumeration, so there is certainly room for improvement. Also, we observe that the mapping f into Alice's language can be seen to be similarly hard to learn if Alice first applies a trapdoor permutation to messages she receives.

C On LINCOS

The aim of LINCOS [7] is to provide a means to send meaningful messages across intergalactic distances to "humanlike" recipients with whom we have had no prior contact. Because our current understanding of the laws of physics suggests that communication across such vast distances is

necessarily slow, it is essentially necessary that the method be non-interactive (in contrast to the protocols we have considered). Thus, over the course of a transmission, Freudenthal attempts to develop a vocabulary, beginning with simple, concrete concepts, and gradually building on these to introduce more and more abstract concepts, until a sufficiently rich vocabulary has been developed to permit the desired message to be sent in a form that the recipient will understand.

Freudenthal assumes that the message will be transmitted by means of radio waves, and uses physical properties of the waves to illustrate the earliest concepts (again, in contrast to our setting of pure information, where we have abstracted away these properties). For example, a pulse of a given duration is used to illustrate a length of time, and a natural number n is initially illustrated by n pulses. New vocabulary are introduced by including a new "symbol" (waveform) in a list of example messages illustrating the use of that symbol. Again, for example, suppose that we wish to define a symbol that means "equals." Let $\cdot$ denote a pulse, and let $=$ denote the waveform we wish to associate with the notion of equality. We would then send a message of the following form, letting spaces be communicated by short pauses and new lines be communicated by longer pauses:

$$\cdot = \cdot$$
$$=$$
$$\cdot\cdot\cdot\cdot\cdot = \cdot\cdot\cdot\cdot\cdot$$
$$\cdot\cdot\cdot\cdot\cdot\cdot\cdot = \cdot\cdot\cdot\cdot\cdot\cdot\cdot$$
$$\cdot\cdot\cdot = \cdot\cdot\cdot$$
$$\vdots$$

After many, many such examples, roughly corresponding to messages of the form "$n = n$" for various values of n, Freudenthal asserts that a "humanlike" recipient will have observed that the quantities preceding and following the waveform for $=$ are always equal, and has inferred that "=" is generally used when these quantities are equal. Therefore, we assume that henceforth the symbol "=" may be used to communicate the concept of equality.

Our primary criticism of LINCOS is that Freudenthal does not define precisely what his setting is, where in the absence of formal definitions, there is no way to prove that the intentions are achieved. If we attempt to provide a formal setting for LINCOS, we quickly see that there are some unresolved issues, as follows. Aside from taking basic, concrete notions (pauses, natural numbers, etc.) for granted, we may observe that the central assumptions of LINCOS are that:

1. Sufficiently many examples are given so that the intended concept is "obvious"
2. The concept being illustrated is associated with the symbol being introduced

Where, even if we take the subtle issue in assumption 2 for granted, it is still unclear how to determine in general the number of examples necessary for assumption 1 to hold. A more rigorous analysis of LINCOS would be desirable, but this seems to be beyond our capabilities at the moment.