# Efficient Semantic Communication via Compatible Beliefs*

Brendan Juba
Harvard University and MIT CSAIL
bjuba@mit.edu

Madhu Sudan
Microsoft Research
madhu@mit.edu

October 14, 2010

**Abstract**

In previous works, Juba and Sudan [6] and Goldreich, Juba and Sudan [4] considered the idea of "semantic communication", wherein two players, a user and a server, attempt to communicate with each other without any prior common language (or communication) protocol. They showed that if communication was *goal-oriented* and the user could *sense* progress towards the goal (or verify when it has been achieved), then meaningful communication is possible, in that the user's goal can be achieved whenever the server is *helpful*.

A principal criticism of their result has been that it is inefficient: in order to determine the "right" protocol to communicate with the server, the user enumerates protocols and tries them out with the server until it finds one that allows it to achieve its goal. They also show settings in which such enumeration is essentially the best possible solution.

In this work we introduce definitions which allow for efficient behavior in practice. Roughly, we measure the performance of users and servers against their own "beliefs" about natural protocols. We show that if user and server are efficient with respect to their own beliefs and their beliefs are (even just slightly) compatible with each other, then they can achieve their goals very efficiently. We show that this model allows sufficiently "broad-minded" servers to talk with "exponentially" many different users in polynomial time, while dismissing the "counterexamples" in the previous work as being "narrow-minded," or based on "incompatible beliefs."

[*] Portions of this work are presented in modified form in the first author's Ph.D. thesis. [5, Chapter 4]. Research supported in part by NSF Awards CCF-0915155 and CCF-0939370.

# 1 Introduction

In this work we continue the work initiated previously by Juba and Sudan [6] and Goldreich, Juba and Sudan [4] who considered "universal semantic communication" in the presence of "misunderstanding." Here, we address one of the main criticisms faced by these prior works, and to this end, we summarize these works first.

The two works mentioned above consider two-party (or more generally $n$ party) communication in the setting where the two players don't necessarily understand each other. Lack of understanding is modeled by letting each of the two players, the *user* and the *server*, come from a large (possibly infinite) class of potential users or servers, and each player does not know which specific member of the other class it is communicating with. The semantics of the communication are modeled by *goals*. In [6] the goal of the user is computational: Specifically, it wishes to decide some instance of a (potentially hard) decision problem and the hope is that the server can simply solve the question and *communicate* the answer to the user. In [4] this setting is extended to consider *any* possible goal of communication. In the exposition below, we stick to the computational goal of [6] to describe our work, though the results do generalize to the broader context of [4].

The work [6] considers the setting where the class of users is all probabilistic polynomial time[1] interacting stateful machines, interacting with servers that are also stateful machines, but with unrestricted time complexity. [6] shows that for the goal of deciding a PSPACE-complete problem, there is a *universal user* that can achieve its goal (i.e., solve the decision problem on the given instance) in polynomial time when interacting with any *helpful server*, i.e., one for which there exists *some (other) efficient user* who reliably achieves the goal with the server. Thus this setting captures the lack of common language: a server can pick the language of communication (or protocol) of its choice while still being helpful (since it allows the user that speaks the same language to achieve its goal), and the result shows that the *universal user* can achieve its goal despite the lack of common language (and effectively implies that the user learns the language in a functional sense).

A weakness of the result above is what we shall refer to as the "enumeration bottleneck." The way the universal user $U$ achieves its goal is to try and enumerate every user $U'$ in the class of all users and attempt to simulate the interaction of $U'$ with the server. (If correct, such an interaction will not only help $U$ solve the decision problem, but also generate an (interactive) proof. If such a valid proof is not obtained, then it must be that $U'$ is not the right user to interact with the server.) Unfortunately, this whole process works in a reasonable amount of time only if the correct user $U'$ appears early in the enumeration used by $U$. As in the "classical theories" a la Kolmogorov [8] and Levin [7], every protocol is enumerated within "constant" time, and this also allows [6] to obtain their results. In particular, their user only succeeds in time exponential in the shortest encoding of a $U'$ that works with the server. This raises the question of whether it is possible to construct systems where the user can "learn" the server's language more actively.

In [6] it is shown that in their model such a more efficient user can not be achieved. Specifically they note that the server may pick a $k$-bit password and only be helpful to users that attach this password as a prefix to every question (and reply with the null string on all other questions). A universal user would have to "discover" this password to achieve its goal and this discovery clearly takes exponential time in $k$ (whereas the description of the server has length linear in $k$).

[1] Strictly, they consider all users that run in polynomial time when interacting with any given server, though the polynomial depends on the server. This seems to be the more natural definition of polynomial time in this setting, and we stick to the same convention.

In view of this limitation result, one needs to search for alternative definitions that allow user-server interactions to be more efficient. A priori this seems to require a search for "natural" properties of "natural languages," but we see no evidence for any universal consensus on what should be taken to be "natural"—the various notions of which ways to communicate are natural seem highly contextual and appear not to yield any universal truths. In view of this, we propose another way to model efficient learning, which we believe will be useful in designing better servers.

### 1.1 Our model and main result

Our approach to make progress on this somewhat elusive question is to model the "intent" of the server better. For example, in the "password-protected" server example above, one may ask, why is the server insisting on this password? If it is for security reasons, or to discourage frivolous users then a $2^k$ lower bound on the communication complexity is really consistent with the intent of the server and little can be done to overcome this limit. On the other hand, if the server is picking this password as a very natural $k$ bit string that ought to be evident to any "intelligent" person as a natural choice (e.g., when PS files are expected to be prefaced with "`%!PS-Adobe` ..." by postscript printers) then the server doesn't think of the $k$-bit password as a deterrent to potential users, and this ought to be somehow captured by the "beliefs" of the server as to what is natural. In this work, we set out to do exactly this.

Throughout the following, we consider users whose goal is to decide some decision problem $\Pi$, i.e., to compute $\Pi(w)$ for some input instance $w$ of length $n$.

**Server's Beliefs** We model the beliefs of a server $S$ by a distribution $Q$ over all potential user protocols it "intends" to serve. We then measure the efficiency $T_S$ of the server by the expected time that it takes a user $U$, chosen from the distribution $Q$, to achieve its goal. (The complexity bounds we consider are worst-case over instances and average case over users, and we study this bound as a function of the size of the instances, $n$; we can also take the distribution $Q$ be an ensemble parameterized by the "length of a protocol" $\ell$ for a given encoding of protocols.) The pair $(Q, T_S)$ thus describe the beliefs and intentions of the server $S$. A server is helpful *with respect to its own beliefs* $Q$ if $T_S$ is polynomial in $n$ and $\ell$.

Of course, a server who is efficient with respect to its own beliefs need not be efficient for every user. To understand which users it can serve efficiently, we also need to study the beliefs of the users.

**User's Beliefs** In our model, users also have beliefs about which servers they may be talking to, and one could model this by a distribution over all possible servers. But this makes it hard to compare and evaluate the compatibility of the server's beliefs and the user's beliefs (since they are defined on different universes). Instead, we model the user's beliefs by a distribution $P$ on the users that it thinks that a typical server may serve. (For instance, $P$ could be the distribution induced on users by picking a server $S$ according to the user $U$'s beliefs about which server it may be talking to, and then picking a user $U'$ according to the distribution $Q = Q_S$, assuming the user can easily obtain samples from $Q_S$ given $S$.) We don't dwell on how the user arrives at this distribution, but rather insist that the user should express its beliefs in such terms, and further that this distribution $P$ should be efficiently sampleable.

Having defined the beliefs of the user and the server, we now ask, when are these *compatible?* To this end, we define the agreement between two distributions, denoted $\alpha(D_1, D_2)$, to be the quantity $1 - ||D_1 - D_2||$, where $||D_1 - D_2||$ denotes the total variation distance between $D_1$ and $D_2$. (Equivalently, $\alpha(D_1, D_2) = \sum_{\omega \in \Omega} \min\{D_1(\omega), D_2(\omega)\}$.)

**Main Result** Our main theorem shows that for every sampleable distribution $P$ on users, there exists a universal user $U$ with $P_U = P$ that can achieve its goal (of deciding $\Pi$) in time $\mathrm{poly}(T_S/\alpha(P, Q_S))$ when communicating with server $S$. Thus, as the time required by the universal user to communicate with a server depends only polynomially on the agreement of the user's distribution with $Q_S$, the user only needs to find a distribution $D$ that has "reasonable" agreement with the distribution $Q_S$ to communicate with $S$ in a similarly "reasonable" amount of time. This result is stated formally in Theorem 7.

We note that once the definitions are in place the theorem is not hard to prove. We discuss the utility of the theorem in the next section.

## 1.2 Implications

The utility of Theorem 7 depends on the ability of the "universal user" to guess an appropriate distribution $P$ that is compatible with that of the server and *simultaneously*, the ability of the server to efficiently service a large class of users (those with large probability mass in $Q$). Below we argue that the latter can be done, and the former is roughly the best hope we have.

We first discuss the possibility of designing servers that can service a large class of users simultaneously.

We note first that this is already being done quite often in practice, and we are merely providing the right definitions to support this practice. For instance a USB (Universal Synchronous Bus) driver (acting as the server) on a standard laptop very quickly learns the identity of the user (the USB device, be it a CD player, a memory stick, a printer etc., and therefore its corresponding protocol) and quickly learns to serve it (i.e., send/collect information with the right instructions). We formalize such actions by a simple theorem. which shows that there exist servers that are helpful to exponentially many user protocols. Specifically there is a server $S$ that has uniform support on exponentially many user protocols of description length $\ell$, while allowing each to reach its goal in time $\mathrm{poly}(\ell)$ on this distribution. (See Theorem 8.)

Of course, simply counting the number of user protocols is not sufficient, but it seems to be a minimal requirement which we can claim to satisfy. In our proof, the exponential class of users seem to be quite diverse, and can each demand service in completely different languages and yet may all be simultaneously serviced if they functionally identify themselves.

A more subtle question is, how can one guess a distribution $P$ that might be compatible with the server distribution? Indeed, this seems to be as hard a problem as proposing any "natural" or "logical" restrictions on "language," and there are a multitude of inconsistent opinions on this. A nice aspect of our definition, based on "flexible beliefs" as opposed to dogmatic certainty, is that it naturally allows a moderate number of (potentially inconsistent) "naturalness" restrictions to be incorporated: if distributions $P_1$ and $P_2$ (possibly supported on disjoint sets of users) are proposed as "natural" candidate distributions for what the server may service, then the distribution $P$ which puts half its mass on $P_1$ and half on $P_2$ will allow the user to achieve its goal with just a constant factor slowdown.

### 1.3 Related models in the literature

The notion of using beliefs to model and cope with uncertainty is of course not new, and is a standard theme in statistics, AI, and Game theory. The usual model here tends to assume some globally known beliefs about various random variables, and focuses on the design and analysis of processes which work well when the random variables come from this distribution. For example, in traditional Bayesian inference, one may have a (parameterized) family of models, and a prior belief about which members of the family are likely, often given as a distribution over the settings of the parameters in a model family. The objective is then to guess the settings of the parameters capturing a real process after observing a sample of data generated by that process. The work is in obtaining a "best guess" from the *posterior*, i.e., the distribution obtained by incorporating the new data into the prior using Bayes' rule. If the prior distribution is good – e.g., if it is informative and the process was drawn from this prior distribution over models – then we expect that the best guess given by Bayesian methods should be pretty good. One of the drawbacks of the Bayesian approach, though, is that it really doesn't provide much guidance in choosing a prior, and there are *no* guarantees about the performance with a *bad* prior.

A model inspired by traditional Bayesian inference, but similar to ours, is the PAC-Bayes approach to inference. This approach, suggested by Shawe-Taylor and Williamson [12] and developed by McAllester [10], attempts to repair the weakness in Bayesian methods mentioned above by establishing a bound on the quality of the guesses that holds for any quality of prior—precisely, the generalization error of members of the model family are uniformly bounded by a function of how much our posterior distribution differs from our prior. Informally then, a PAC-Bayes bound expresses the quality of the guesses *in terms of* the quality of the prior. Of course, possession of such a bound naturally suggests an analogue of the *structural risk minimization* principle from statistical learning theory [13], which suggests that, rather than the guess indicated directly by the posterior distribution, the best guess for a model is the one that has the lowest total loss on the sample and in the generalization bound. Thus, the PAC-Bayes approach allows one to incorporate the beliefs from prior distribution, but moreover fails gracefully if these beliefs are not accurate.

Our approach is similar, though of course the prime difference is that we are not interested in learning/inference, but rather in communicating. In our setting, a priori it is not even clear how to express beliefs of the users and servers (or how to compare them) and we view our main contribution to be a method to do so which allows for an improved efficiency analysis.

### 1.4 Organization of this paper

In Section 2 we describe the basic notions of semantic communication from [6]. In Section 3 we describe our model and prove our main result. In Section 4 we show that there exist servers that serve exponentially many different users in polynomial time. Some concluding thoughts are given in Section 5.

## 2 Preliminaries

We start by reviewing the principal notions from the previous works [6, 4], capturing communication in the absence of a fixed common language; in this case, correctness is determined by fixing a *goal of communication* that the parties should achieve. [6] fixes some arbitrary decision problem $\Pi : \{0,1\}^* \to \{0,1\}$ and considers a user $U$ interacting with a server $S$ with the aim of computing

$\Pi(w)$ for some given $w \in \{0,1\}^n$. We now formally express that the user achieves its goal in the absence of a common language by saying that it successfully computes $\Pi$ with a large *class* of servers $\mathcal{S}$, "speaking" many different languages:

**Definition 1 (($\Pi, \mathcal{S}$)-Universal)** *We say that a user $U$ is a universal decider for a decision problem $\Pi$ with a class of servers $\mathcal{S}$, or $(\Pi, \mathcal{S})$-universal, if for any server $S \in \mathcal{S}$, and any initial state $\sigma$ of $S$ $S$ starting in state $\sigma$ helps $U$ decide $\Pi$ and there exists a polynomial $p_S$ such that for every instance $w \in \{0,1\}^*$ $U$ runs in expected time $p_S(|w|)$.*

The principal result of [6] is that for PSPACE-complete problems $\Pi$, there a *universal* probabilistic polynomial time user that can compute $\Pi$ by interacting with *any* "$\Pi$-helpful" server. We describe the result more formally below, but we must first recall the definition of $\Pi$-helpful:

**Definition 2 ($\Pi$-Helpful)** *For a decision problem $\Pi$, we say that a server $S$ is* $\Pi$-helpful *if there exists a probabilistic user algorithm $U_S$ and a polynomial $p$, such that for every state $\sigma$ of $S$, $S$ starting in state $\sigma$ helps $U_S$ decide $\Pi$ in $p(n)$ steps.*

Note that it is a *minimal* requirement that $\mathcal{S}$ contain only $\Pi$-helpful servers, since $(\Pi, \mathcal{S})$-universal users witness the $\Pi$-helpfulness of any $S \in \mathcal{S}$. Now, the main theorem of [6] is:

**Theorem 3 ([6])** *Fix a problem $\Pi$ and let $\mathcal{S}$ be the set of all $\Pi$-helpful servers. If $\Pi$ is PSPACE complete then there is $(\Pi, \mathcal{S})$-universal user. Conversely if there is a $(\Pi, \mathcal{S})$-universal user, then $\Pi$ is in PSPACE.*

The positive direction in the theorem above leverages the existence of interactive proofs for PSPACE problems [9, 11] to create the universal user. As stated above, the theorem does not clarify the status for PSPACE-intermediate problems, but as noted by [4] (and shown in detail in [5]), the technique generalizes. We give their more refined version of the above theorem, which characterizes problems with universal users exactly in terms of languages with "competitive interactive proofs," a notion studied in [2], in the appendix. In any event, the theorem stated above suffices to make our concerns explicit and address them.

The unfortunate consequence of this level of generality that we address here is that the universal user constructed in Theorem 3 experiences an overhead in its running time that can be exponential in the user's shortest encoding of a protocol for using the server. Note that if a third party (who knew both the server and the user) was available to describe the shortest protocol to the user, the user's running time would have only had a polynomial dependence on the encoding of this protocol. It was shown in [6] that such efficiency cannot be obtained without the presence of a third party, and we recall this result next.

For simplicity, we will present the result in terms of the "password closure" of a helpful server:

**Definition 4** *Given any server $S$, the* password closure *of $S$, denoted $\mathcal{PW}(S)$, is the following class of servers: for each $x \in \{0,1\}^*$, $\mathcal{PW}(S)$ contains the password-protected server with password $x$, $S^x$, described as follows. $S^x$ has a copy of the states of $S$ and in addition, a "waiting for password" state, from which it sends only empty messages to the user until it first receives the message $x$, whereupon it enters a designated "initial state" from the states of $S$.*

Notice that the password closure of a $\Pi$-helpful server $S$ contains only $\Pi$-helpful servers (in particular, that help various other users with the same asymptotic running time); therefore, an exponential lower bound for the running time of a $\Pi$-universal user on this class is also a meaningful lower bound for the overhead of the $\Pi$-universal user in general.

**Theorem 5** *Let* $\Pi$ *be a* PSPACE*-complete decision problem and let* $S$ *be a* $\Pi$*-helpful server. Suppose there exists* $(\Pi, \mathcal{PW}(S))$*-universal user* $U$ *running in time* $T(n,m)$ *on instances of length* $n$ *when interacting with servers* $S^x$ *for* $|x| = m$*. If for some* $m(n) = \omega(\log n)$*,* $T(n, m(n))$ *is bounded by a polynomial in* $n$ *(and thus* $T(n,m) = o(2^m)$*), then* PSPACE = BPP.

Thus, the above theorem shows that enumeration is unfortunately qualitatively optimal for our goal of interest in the basic universal setting. Since this exponential "constant" in the running time of a universal user protocol is extremely undesirable, we need to explore means of restricting the class of servers so that it does not contain password-protected servers in particular, but is still broad enough to yield useful protocols. In particular, this result strongly suggests that merely restricting the computational complexity of the user protocols that the servers help cannot suffice for obtaining a more efficient universal user.

In the next section, we explore some definitions that rule out the "hiding" behavior of the password-protected servers and allow us to measure the *compatibility* of a server with a user. We can then exhibit a protocol for which the efficiency scales appropriately with this quantity, and therefore under some natural conditions a universal protocol can run more efficiently.

# 3 Our Model and Results

In the previous section, we saw that as a consequence of our counting "password-protected" servers as "helpful" servers that our universal users were expected to work with, we had no hope of giving a really efficient user strategy—the number of rounds required under such conditions grows exponentially in the length of the user protocol needed to successfully communicate with the server. This is a dissatisfying state of affairs since, in applications, one surely never expected a protocol that could quickly break into a password-protected server; we would have been quite happy to use a protocol that was only efficient when the server was not designed to keep us out in the first place. Thus, we desire a refinement of our notion of "helpfulness" to include only easy-to-access servers, and a protocol that can take advantage of such servers, both of which we will develop presently, inspired by "PAC-Bayesian" analyses in learning theory [10, 12].

## 3.1 Motivating the notions

Our model was already introduced informally in Section 1, and we will formalize it later in this section. But before doing so we explain why some alternative approaches fail.

As a starting point, consider a *server designer* who is attempting to design an easy-to-access server for some fixed goal that is known to all parties. Once we have a notion of what kind of server a benign designer might produce, we will be able to ask whether or not we, as users, can generally access such servers and achieve the goal efficiently with their assistance.

A first attempt at developing a notion of an easy-to-access server might proceed by considering what went wrong with password-protected servers: the reason a long password provides security is that, for a user who does not have the password, accessing the server requires searching through an exponentially large space, but this only holds if the password is properly chosen—if the password does not have enough "randomness," then it may be possible to break it by searching through a smaller space, such as searching through the words in the dictionary, for example. Relative to the dictionary, such weak passwords have short descriptions, and may be considered "easier to guess" or more "natural." Thus, as a first stab at a notion of easy-to-access along these lines, we might wish to say that a server should operate with a user protocol with a short description.

The problem with the "short description" requirement is that there could be a gap between *our* notion of a short description and the server designer's notion. One might be tempted to retort that a basic result in Kolmogorov complexity is that these description lengths should not differ by more than a constant [8], but this is exactly the deficiency of the prior works in semantic communication [6, 4]. Thus, it is clear that this first attempt is inadequate.

A second approach is to consider somewhat more restricted classes of users/servers. Common suggestions include: (a) servers/users whose languages/protocols have features similar to natural/programming languages; (b) servers/users whose behavior shows strong "analogies" in different contexts; or (c) servers/users who announce their preferred "protocol" before starting to communicate. While each of these restrictions may seem natural, they lead us away from universality, and we cannot insist on such behavior.

This leads to our suggestion that preference for such restrictions should be expressed as *beliefs*. Beliefs, by their very nature, allow opinions to be expressed without full justification. A "natural" restriction in the opinion of the server can thus hopefully be captured by its "belief" and similarly for the user. Now we don't have to insist on a universal notion of "natural" programs: different users/servers may have different notions of "naturalness" which are captured by different beliefs. For example, the length-weighted uniform distribution over user protocols is a natural sampleable belief capturing our first attempt, while our proof of Theorem 8 uses a belief similar to (c) above.

Our approach then relies on two crucial properties of these "beliefs." First, *each user and server can evaluate their own behavior with respect to their beliefs to see how "broad-minded" they seem*—for example, if the server $S^x$ in the password protected case produced $x$ by picking $x$ uniformly at random from $\{0,1\}^m$, then it should understand that it is acting "narrow-mindedly" since the user should not have much of a chance at guessing $x$ in any reasonable amount of time. On the other hand, if the server chose $x$ as a string of "low complexity," it may legitimately believe that "natural" intelligent users should be able to guess this string. Of course, there still remains the question as to whether the user would also think $x$ has low complexity, but this is where the (in)compatibility of the beliefs works in to the efficiency: the second crucial property is that *we can define a measure of compatibility of beliefs that captures its effect on the efficiency of communication.*

## 3.2 Beliefs and Compatibility

As mentioned in Section 1, the "prior" beliefs of a server $S$ are modeled by a distribution $Q_S$ on the users. We now define the "benchmark" running time of the server when interacting with a random user chosen according to some distribution.

**Definition 6 (Benchmark running time)** *For a problem $\Pi$, a server $S$, and a user protocol $U$, let $t_{U,S}(n)$ denote the maximum expected (over internal coin tosses of $U$ and $S$) running time of $U$ to compute $\Pi(x)$ when interacting with $S$, over instances $x \in \{0,1\}^\ell$ for $1 \leq \ell \leq n$ and starting states of $S$. Then, for a distribution over user protocols $Q$, the $Q$-*benchmark running time for $\Pi$ with $S$, *denoted $t_{Q,S}(n)$, is given by the expected value of $t_{U,S}(n)$ when $U$ is sampled from $Q$.*

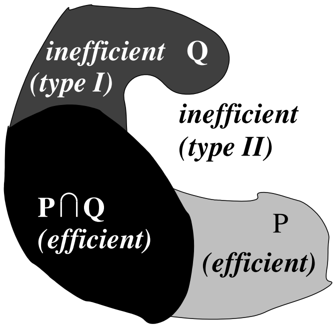

Figure 1: An illustration of the two types of inefficiencies; $P$ denotes the set of user protocols given high weight by the prior distribution $P$, and similarly $Q$ denotes the set of user protocols given high weight by the prior distribution $Q$.

We compare the performance of a server $S$ with belief $Q$ with that of a user $U$ designed with prior $P$. To compare them we use the "compatibility" of $P$ and $Q$. Specifically, for distributions $P$ and $Q$ supported on $\Omega$ (where $P(\omega), Q(\omega)$ represent the probability of a point $\omega \in \Omega$), let the agreement of $P$ and $Q$, denoted $\alpha(P, Q)$, be the quantity $\sum_{\omega\in\Omega} \min\{P(\omega), Q(\omega)\}$.

Our main theorem shows that for every sampleable distribution $P$ there is a user $U$ (who knows $P$) that computes $\Pi$ efficiently whenever (I.) $P$ has noticeable agreement with the server's beliefs $Q_S$, and (II.) $S$ is helpful with respect to its beliefs $Q_S$.

**Theorem 7 (Universal users for close priors)** *For a class of servers $\mathcal{S}$, let a distribution on user protocols $Q_S$ be given for each $S \in \mathcal{S}$ and let $t_S : \mathbb{N} \to \mathbb{N}$ denote the benchmark running time of $S$ on distribution $Q_S$ (i.e., $t_S(n) = t_{S,Q_S}(n)$). Let $\Pi$ be a PSPACE complete problem. Then there exist polynomials $q = q_\Pi$ and $r = r_\Pi$ such that for every efficiently sampleable distribution $P$ over users, there is universal user $U = U_P$ that computes $\Pi$ in time $T_S(n) = q(n, 1/\alpha(P, Q)) \cdot (t_S \circ r)(n)$ when interacting with server $S$.*

As with Theorem 3, the above theorem can also be extended to other problems that have competitive interactive proofs (see Appendix). Figure 1 pictorially describes the two types of conditions required by Theorem 7 to get efficient universal users.

**Proof:** We use a variant of the protocol used in the proof of Theorem 3 in which the enumeration of protocols is replaced by simply sampling repeatedly from our given distribution $P$.

**Construction.** Consider any interactive proof system for $\Pi$ in which each message of the prover is a single bit. Since we know that the optimal prover strategy can be simulated in PSPACE, there is a polynomial time reduction that, given the current message history and instance $x$, produces an instance $y$ such that $\Pi(y)$ is the prover's next message. Let $r(n)$ be the corresponding polynomial upper bound on the length of $\Pi(y)$ for $x$ of length $n$.

Now, since the reduction can be computed in polynomial time, we see that the entire interaction between the prover and the verifier can be simulated in polynomial time relative to an oracle for $\Pi$. Let $q_1$ be the time bound, and let $U^{(\cdot)}(x, b)$ be a protocol that computes this simulation for common input $\Pi(x) = b$, answering oracle queries for the prover strategy by invoking its oracle $O(\log q_1(n))$ times per query, and returning a majority vote.

Our protocol is then as follows. In parallel, we run two algorithms, one directly computing $\Pi$ in time $2^{q_2(n)}$ for some polynomial $q_2$, and one computing the following:

- For $i = 1, 2, \ldots$, repeat the following:
  - For $j = 1, 2, 3, \ldots, i - 2\log i$, and $k = 1, 2, 3, \ldots, 2^j$, repeat the following:
    1. Sample a protocol $\tilde{U}$ from $P$.
    2. For up to $t = 2^{i-j-2\log j}$ steps, invoke $\tilde{U}$ $O(\log q_1(n))$ times and take a majority vote of the verdicts to get a candidate $b$ for $\Pi(x)$, and then in the remaining steps, invoke $U^{\tilde{U}}(x, b)$ up to $O(q_2(n))$ times, and if the verifier accepts in a majority of these runs, output $b$ and halt.

Whenever one of these two algorithms halts, we halt and return that algorithms' answer.

**Analysis.** Correctness is clear: we run the proof system in the inner loop of the second algorithm at most $2^{q_2(n)}$ times, where its soundness guarantees that it only incorrectly accepts with probability 1/3. So, for an appropriate choice of constants, the inner loop halts with a wrong answer before the first algorithm returns a correct answer with probability less than 1/3.

It thus remains to analyze the running time. Since $U^{(\cdot)}$ simulates an interactive proof system, for any user strategy $\tilde{U}$ that decides $\Pi$ with $S$ with probability 2/3, the verifier accepts in each run of $U^{\tilde{U}}$ with probability at least 2/3. If a $1 - \delta'$ fraction of the user strategies under $P$ decide $\Pi$ with $S$ on instances of length up to $r(n)$ with probability at least 2/3 in $t'$ steps, then if $t > Ct'q_1(n)q_2(n)\log q_1(n)$ for an appropriate $C$, with probability at least 2/3 a majority of the runs of $U^{\tilde{U}}$ accept, given that we successfully sampled such a $\tilde{U}$. We will call these $\tilde{U}$ *good protocols.*

If $i \geq i^* = \log(Ct'q_1(n)q_2(n)\log q_1(n)) - \log(1-\delta') + 2\log\log\frac{1}{1-\delta'}$, then for $j = 1, \ldots \log\frac{1}{1-\delta'}$, we run each protocol we sample for at least $Ct'q_1(n)q_2(n)\log q_1(n)$ steps, and there are precisely $\sum_{j=1}^{\log\frac{1}{1-\delta'}} 2^j = 2\frac{1}{1-\delta'} - 1$ such samples in phase $i^*$; we therefore obtain a good protocol in phase $i^*$ and run it for sufficiently many steps with probability at least $1 - (1-\delta')^{\frac{1}{1-\delta'}} \geq 1 - \frac{1}{e}$, where each time we then succeed and halt with probability at least 2/3. Moreover, in phase $i^* + r$, there are $2^{r+1}\frac{1}{1-\delta'} - 1$ such samples, and thus if we group our samples into batches of $\frac{1}{1-\delta'}$, in each of our first $2^{r+1} - 1$ batches, we only fail when we either fail to hit a good protocol, or when a good protocol fails, which by a union bound occurs in each group with probability at most $1/3 + 1/e$, and thus at most $(1/3 + 1/e)^{2^{r+1}-1}$ overall. Since the total running time up to phase $i^* + r$ is $\sum_{i=1}^{i^*+r}\sum_{j=1}^{i-2\log i} 2^j \frac{2^i}{j^2 2^j} \leq \frac{\pi^2}{3}2^{i^*+r}$ our expected running time is at most

$$\frac{\pi^2}{3}2^{i^*}\left(1 + \sum_{r=0}^{\infty} 2^{r+1}(\frac{1}{3} + \frac{1}{e})^{2^{r+1}-1}\right) = O(2^{i^*}) = O\left(t'q_1(n)q_2(n)\log q_1(n)\frac{1}{1-\delta'}\log^2\frac{1}{1-\delta'}\right)$$

We now note that if the probability of sampling a good protocol under $Q_S$ is at least $1 - \delta$, $1 - \delta' \geq \alpha(P, Q_S) - \delta$. Moreover, for $t' = 2(t_S \circ r)(n)/\alpha(P, Q_S)$, it follows by Markov's inequality that $\delta \leq \alpha(P, Q_S)/2$, we therefore find for this choice of $t'$ that our expected running time is at most

$$O\left((t_S \circ r)(n)q_1(n)q_2(n)\log q_1(n)\left(\frac{1}{\alpha(P, Q_S)}\log\frac{1}{\alpha(P, Q_S)}\right)^2\right)$$

■

# 4 Servers serving wide classes of users

We note in this section that it is possible to construct servers who serve an exponentially large class of *distinct* users efficiently. Our construction, though simple, suggests a "simple" abstraction of how such universal protocols are being implemented in practice.

We say that a distribution $D$ on $\{0,1\}^*$ has *exponential* support if the probability of picking a length $k$ string under this distribution is inverse polynomial in $k$ and, conditioned on this event, it is uniform on an exponentially large subset of length $k$ strings.

**Theorem 8** *For any PSPACE-complete problem $\Pi$, there exists a $\Pi$-helpful server $S$ with associated distribution $Q$ with exponential support such that the benchmark running time $t_{S,Q}$ is polynomial in the input length and in the length of the user description.*

**Proof:** The class of users $U_\Gamma$ we consider includes one for each LINSPACE-complete problem $\Gamma$. On an instance $y$ of $\Pi$, the user $U_\Gamma$ sends the message "$E(\Gamma); x$" where $E(\Gamma)$ is an encoding of the problem $\Gamma$ in some fixed universal language and $x = R_\Pi(y)$ for some reduction from $\Pi$ to $\Gamma$, and returns the server's response as its answer.

The server $S$ on receiving a message $E(\Gamma); x$ uses the canonical reduction $R_\Gamma$ from $\Gamma$ to $\Pi$ to compute $R_\Gamma(x)$ and responds with $\Pi(R_\Gamma(x)) = \Gamma(x)$ $(= \Pi(y))$. This server is thus $\Pi$-helpful to every user $U_\Gamma$ and thus under, e.g., some universal distribution on $U_\Gamma$, has polynomial benchmark running time.

The theorem follows from the fact that the number of different LINSPACE-complete problems with description length $E(\Gamma) \leq k$ is exponential in $k$. ■

# 5 Conclusions

We introduced a new measure of compatibility between users and servers that allows each to pick their own favorite "language" or "protocol" of communication, and measures the compatibility between the two. The measure, based on measuring the proximity between the beliefs of the user and server (about who they are talking to), allows for the design of "broad-minded" servers and users that allow many user/server pairs to reach understanding quickly.

We believe this measure is the right one to explain the general success of natural (human-to-human) communication, while allowing for occasional miscommunication.

This measure also articulates the main challenge in "robust" server design: a server should attempt to efficiently service many different users, and this is not always easy. As we understand it, the way "USB servers" manage to do so is not very different from the example construction in the proof of Theorem 8. In practice, such servers maintain a "large" list of possible device identifiers and for each such identifier maintain a piece of software/instructions indicating how this device ought to be treated. While such designs remain ad-hoc, our notions provide a way to formally measure their performance. They also raise the question as to whether there are other ways to design servers that are efficient while having beliefs that have exponential support. In particular, there is a large body of literature on Bayesian inference in the Machine Learning community, which might suggest ways to construct more sophisticated servers that learn (how to help) a user's protocol during their interaction, e.g., if the prior over user protocols is for some appropriate parameterized class of protocols.

Moving away from computers to "human-to-human" communication, one could ask how users and/or servers form "beliefs" about who they are talking to. We feel this is a natural phenomenon where the users/servers attempt to generalize from multiple interactions. Such interactions ought to lead each user/server to create some general models capturing the diversity of users/servers they interact with, along with some priorities if the frequencies with which they deal with different kinds of users/servers is very different.

Thus, in our opinion, modeling the efficiency of communication in terms of beliefs and compatibility is possibly the right way to seek efficient protocols, while capturing existing attempts to do so, both in engineering and in nature.

## Acknowledgements

This work was inspired by conversations with Adam Kalai, Dan Roy, and David Sontag; it has benefitted from the insightful input and criticisms of Oded Goldreich and Sanjeev Khanna.

## References

[1] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[2] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM J. Comput.*, 23(1):91–119, 1994.

[3] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995. Preliminary version appeared in Proc. 21st STOC, pp86–97, 1989.

[4] Oded Goldreich, Brendan Juba, and Madhu Sudan. A theory of goal-oriented communication. Technical Report TR09-075, ECCC, 2009.

[5] Brendan Juba. *Universal Semantic Communication.* PhD thesis, MIT, 2010.

[6] Brendan Juba and Madhu Sudan. Universal semantic communication I. In *Proc. 40th STOC*, 2008.

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

[8] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications (2nd ed.).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[9] 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.

[10] David A. McAllester. Some PAC-Bayesian theorems. *Mach. Learn.*, 37(3):355–363, 1999.

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

[12] John Shawe-Taylor and Robert C. Williamson. A PAC analysis of a Bayesian estimator. In *Proc. 10th COLT*, pages 2–9, 1997.

[13] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

# Appendix

## A Extensions to Problems with Competitive Interactive Proofs

Theorem 3 as stated in Section 2 (which, in turn, is nearly identical to the main result of Juba and Sudan [6]) does not address the feasibility of constructing $\Pi$-universal users for problems $\Pi$ that are in PSPACE but not PSPACE-complete. As pointed out in subsequent work by Goldreich, Juba, and Sudan [4], the construction actually works in somewhat greater generality; moreover, *given the right definition*, it is then possible to obtain an exact characterization in terms of interactive proofs [5], similar to the characterization of program checking in terms of interactive proofs by Blum and Kannan [3]. We will review the appropriate definitions for this characterization, and sketch the argument.

The construction of universal users described above relies on the availability of an appropriate kind of proof system for the problem. Again, we already know that $\Pi$ must lie in IP in order for it to be feasible for us to give a universal user for $\Pi$. Moreover, though, the constructions of Theorem 3 and 7 worked by searching the space of user protocols for a protocol that simulated the prover in an interactive proof for $\Pi$ with the given server, where we could show that some such user protocol existed. Now, in general, there may be a "gap" between algorithms that decide $\Pi$ with a server $S$, and algorithms that can *prove* membership in $\Pi$ with $S$, even given that some protocol for deciding $\Pi$ exists. The appropriate kind of proof system for our characterization will be one in which no such "gap" exists.

This relationship between the complexity of deciding a problem $\Pi$ and the complexity of the prover for an interactive proof system for $\Pi$ has been studied in the past—Bellare and Goldwasser [2] introduced *competitive proof systems* to consider precisely such questions. Roughly, these are interactive proof systems for set membership in which the prover can be efficiently simulated using oracle queries to the set. In particular, the question of the existence of competitive interactive proof systems is a generalization of the decision-versus-search question for NP proof systems—simulating the interaction between the prover and the verifier using an oracle for the set allows one to generate "proofs" of membership in polynomial time, given the ability to decide membership. Precisely, the definition is as follows:

**Definition 9 (Competitive interactive proof system)** *Let $P$ be a probabilistic polynomial time interactive oracle Turing machine and let $V$ be a probabilistic polynomial time interactive Turing machine. We say that $(P, V)$ is a* competitive interactive proof system *for a set $S$ if*

1. **(Completeness)** *For every $x \in S$, the probability that $V$ accepts when interacting with $P^S$ on common input $x$ is at least $2/3$.*

2. **(Soundness)** *For every $x \notin S$ and every interactive Turing machine $\tilde{P}$, the probability that $V$ accepts when interacting with $\tilde{P}$ on common input $x$ is at most $1/3$.*

*We say that $P$ is a* competitive prover strategy*. We let* compIP *denote the class of decision problems $\Pi$ with competitive interactive proof systems for membership in $\Pi$ (viewed as a set), and we let* $\mathrm{co}-\mathrm{compIP}$ *denote the class of decision problems such that their complements are in* compIP.

It may now be verified that the proof of Theorem 7 applies to *any* $\Pi$ in $\mathrm{compIP} \cap \mathrm{co}-\mathrm{compIP}$ since the only property of PSPACE-complete $\Pi$ that was used was the existence of an efficient prover relative to an oracle for $\Pi$, i.e., a competitive prover strategy. We therefore obtain:

**Theorem 10 (Universal users for problems with competitive proof systems)** *For a class of servers $\mathcal{S}$, let a distribution on user protocols $Q_S$ be given for each $S \in \mathcal{S}$ and let $t_S : \mathbb{N} \to \mathbb{N}$ denote the benchmark running time of $S$ on distribution $Q_S$ (i.e., $t_S(n) = t_{S,Q_S}(n)$). Let $\Pi$ be a problem in $\mathrm{compIP} \cap \mathrm{co}-\mathrm{compIP}$. Then there exist polynomials $q = q_\Pi$ and $r = r_\Pi$ such that for every efficiently sampleable distribution $P$ over users, there is universal user $U = U_P$ that computes $\Pi$ in time $T_S(n) = q(n, 1/\alpha(P, Q)) \cdot (t_S \circ r)(n)$ when interacting with server $S$.*

Note that this theorem generalizes the positive direction of Theorem 3 since the uniform distribution over protocols yields agreement $2^{-|U_S|-2\log|U_S|}$ with the distribution $Q_S$ that puts all of its mass on a protocol $U_S$ helped by $S$ of length $|U_S|$ in the user's encoding of protocols.

Now, as suggested previously, it is also possible to obtain an *exact* characterization of when universal users can be constructed in terms of competitive interactive proofs. The key observation, as noted by Goldreich, Juba, and Sudan [4], is that any $(\Pi, \mathcal{S})$-universal user for a set $\mathcal{S}$ containing all $\Pi$-helpful servers must not halt with a wrong verdict with probability any greater than 1/3 with *any* server, even if the server is *not* $\Pi$-helpful. Otherwise, it is possible to construct a $\Pi$-helpful server that simulates the unhelpful server in interactions with the proposed user, and thus forces the user to output a wrong verdict with probability greater than 1/3. (A similar argument was also used to show that $\Pi$ must be in PSPACE for $(\Pi, \mathcal{S})$-universal users to exist in the earlier work of Juba and Sudan [6].)

In the present context, this observation allows us to construct competitive proof systems for $\Pi$ and its complement as follows: the competitive prover strategy simulates the oracle for $\Pi$ directly by relaying messages back and forth between the verifier and its oracle, while the verifier runs the universal user strategy for $\Pi$ for a bounded amount of time, and accepts if the strategy returns $\Pi(x) = 1$ (resp. $\Pi(x) = 0$ for the complement) within the given time bound. Since an oracle for $\Pi$ is a $\Pi$-helpful server strategy, it follows that we can choose a polynomial time bound for the universal user such that with this server strategy, the user outputs $\Pi(x)$ and halts within the time bound with high probability. (This argument is given in more detail in [5].) We therefore obtain:

**Theorem 11 ([4, 5])** *There is a $(\Pi, \mathcal{S})$-universal user for any set $\mathcal{S}$ of $\Pi$-helpful servers iff $\Pi \in \mathrm{compIP} \cap \mathrm{co}-\mathrm{compIP}$.*

It is instructive to contrast this result with the characterization of problems that have program checkers in terms of *function-restricted* interactive proofs, i.e., proof systems in which even *dishonest* provers are assumed to respond according to a (stateless) function. Recall that a program checker is an efficient algorithm that with high probability, when given an input $x$ and access to a purported program for $\Pi$, either returns $\Pi(x)$ if the program computes $\Pi$, or otherwise, either (a) returns $\Pi(x)$ anyway or (b) outputs "BUGGY." (We refer the reader to the work of Blum and Kannan [3] for the precise definitions of program checkers and function-restricted interactive proofs.)

**Theorem 12 ([3])** *There is an efficient program checker for $\Pi$ iff both $\Pi$ and its complement have a function-restricted interactive proof system.*

Thus, we see that universal users and program checkers have a relationship to one another analogous to the relationship between (competitive) interactive proof systems and multi-prover interactive proof systems. In particular, Bellare and Goldwasser [2] noted that every problem with a competitive proof system has a function-restricted proof system but the converse is of course unlikely to hold unless EXP is contained in PSPACE since EXP-complete problems have program checkers [1]—so, every problem with a universal user also has a program checker, but the converse is unlikely.