Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 95 (2008)

Universal Semantic Communication II: A Theory of Goal-Oriented Communication

Brendan Juba* Madhu Sudan †

MIT CSAIL
{bjuba,madhu}@mit.edu

February 13, 2009

Abstract

We continue the investigation of the task of *meaningful* communication among intelligent entities (players, agents) without any prior common language. Our generic thesis is that such communication is feasible provided the goals of the communicating players are verifiable and compatible. In a previous work we gave supporting evidence for one specific goal — where one of the players wished to solve a hard computational problem and communicated with the other in the hope of finding a solution.

In this work we initiate a "generic" study of the goals of communication. We present two definitions: one of a "generic" *meta-goal*, which captures the (potentially unrealizable) wishes of communicating agents, and the other being a "generic" *syntactic goal*, which captures effects that can observed by an agent. We then show, under some technical conditions, that those meta-goals that have corresponding syntactic versions are also *universally achievable*, i.e., achievable when the two communicators do not (necessarily) share a common language.

We also show how our formalism captures a variety of commonplace examples of goals of communication, including simple *control-oriented* goals that aim to effect a remote physical action by communication, as well as more subtle *intellectual* goals where the communicator's intent is mostly to gain *knowledge*. Our examples from the latter class include a variety of settings where meta-goals differ significantly from syntactic goals.

*Supported by a NSF Graduate Research Fellowship, and NSF Awards CCR-0514915 and CCR-0726525.
†Supported in part by NSF Awards CCR-0514915 and CCR-0726525.

ISSN 1433-8092

1 Introduction

In this paper we continue the study of "semantic communication." Specifically we ask how can two intelligent entities (or agents), that do not a priori *understand* each other, communicate to reach a state of understanding? What is *understanding*?

We assert that in order to answer the above questions, we need to look into the question: Why do intelligent, selfish, entities communicate? What *goals* are they trying to achieve by communicating? We suggest that by studying the goal formally one can attempt to get a formal notion of understanding. Specifically, if two intelligent entities, Alice and Bob are communicating with each other, we assert that each must have a "selfish" goal that they wish to achieve by the process of communication. Further, from the perspective of one of the entities, say Bob, the achievement of a goal that requires communication (seemingly with another entity that understands Bob) gives an operational meaning to the notion that Bob *understands* Alice. Additionally if Bob can determine whether the goal has been achieved (at any given point of time during his interaction with Alice), then this gives him a means to test, and improve upon, the current level of understanding.

Motivation: In a previous work [11], we considered the assertions above in a restricted setting where the goal of communication was taken to be of a particular form. In that work, Bob was a probabilistic polynomial time bounded algorithm who wished to solve (an instance of) a hard computational (decision) problem. Alice had unbounded computational power and wished to help Bob, except she did not know his language. We showed that Alice could help Bob if and only if the problem that Bob wished to solve was in PSPACE (specifically, solutions to the problem should be verifiable, possibly interactively, by Bob).

In this work we extend the previous work to study *general* "goals of communication." As motivation, observe that communication certainly serves many purposes, many of which have nothing to do with solving computational problems. For example, there are many situations where we wish to remotely exhibit control, e.g., to print a document on a printer or to purchase a book on Amazon. Moreover, the results of our prior work were only meaningful when we could interact with some entity who could decide some problem outside of probabilistic polynomial time; since this seems to be an unrealistic assumption, one might ask whether or not similar models and techniques could be used to provide some computational benefits among probabilistic polynomial time entities, or model other modes of interaction based on intellectual curiosity. We attempt to capture this diversity of motivations with a single definition of a *generic goal of communication* (though we end up defining two different kinds of goals).

Overview of the theory: Starting with a mathematical description of the interacting entity, Bob, we describe "meta-goals," which capture potential goals of communication for Bob as commonly perceived, and "syntactic-goals," which describe goals in terms of information that is actually available to Bob (and computable by Bob within his resource limits). We stress that the definitions are somewhat subtle, counterintuitive, and possibly debatable. Part of the challenge is attempting to create a definition that allows Bob to be adaptive (so as to learn the language of communication with Alice) without varying the goal of the communication itself. As an analogy, one could recall the setting of multiparty computations, where it was a non-trivial task to define a generic task in multiparty computation which could be achieved by different protocols, with or without a trusted party. Our definition leads to an architecture for communication among intelligent agents which is somewhat different from the standard architecture for communication. Specifically, in our model, the role of an "interpreter" of communications emerges as a central one.

Using the resulting definition of a generic goal of communication, we extend the notion of universal semantic communication, where Bob is attempting to converse semantically with some Alice, whom he does not know a priori. We show sufficient conditions on the goal under which a generic goal is achievable universally. The net effect is that we are able to formalize and prove the following "meta-theorem:"

Theorem 1 (Main theorem, informal statement) *A goal can be achieved universally iff some sufficient related verifiable goal exists.*

The formalization of this theorem appears as Theorem 14 in Section 2.2. To get a sense of the conditions that are shown to be sufficient, we compare with our prior work [11]. The previous work showed that in the computational setting, the ability to *verify* the result (i.e., obtaining a proof) was necessary and sufficient. Verifiability remains a central component in our extensions. Indeed, using a similar technique to our prior paper, it follows that whenever Bob can verify success based on his view of the interaction with Alice, it is possible to give a universal protocol for his goal.

Applications and examples: This observation already extends the class of goals we can achieve universally from only computational goals to include the goals we discuss in Section 3.2: goals involving "remote control" where Bob can observe whether or not he has succeeded. In fact, in our main theorem, we show that for the classes of algorithms of interest, a sufficient algorithm for verifying achievement of the goal exists if and only if there exists an algorithm that is efficient with respect to the given resource bounds and achieves the goal universally. In a sense, this is the analogue to our results in the computational setting showing that the class of computational goals that can be achieved universally are computationally equivalent to IP. In this case, it demonstrates that the verifiability of a goal is necessary and sufficient for universal communication.

This is just one of several consequences of our definitions and theorem that we examine. We show how a number of common goals of communication can be modeled by our notions of generic goals. We describe a few more such examples below. In all cases coming up with a meta-goal that captures our generic goal is relatively straightforward. The more interesting part is seeing how the goal in these examples can be supported by syntactic goals that satisfy the sufficiency conditions of our main theorem, and thus achieved universally.

For example, consider the situation in a classroom where Alice is a student and Bob is a teacher. How might Bob be convinced of Alice's ability to solve some class of problems that he knows he is already capable of solving? We show, by taking a more careful view of the communication and measuring the computational resources used by Bob in interpreting Alice, one approach to such a problem in Section 3.4. Our definitions and approach are sufficiently generic to apply to other natural classes of algorithms – in particular, logspace machines – and we show in Section 3.1.2 how to use a recent result of Goldwasser et al. [9] to obtain a logspace protocol that can delegate any polynomial-time computation to any "server" with a communications protocol that can be implemented in logspace. Finally, in Section 3.3, we propose an approach by which we might try to obtain wisdom from a computationally bounded Alice who might only have knowledge of a relatively sparse collection of theorems.

The theme in these last few examples is that we carefully measure or restrict the computational resources used during interpretation in pursuit of our goals. Again, as a consequence of our main theorem, all of these goals can be achieved universally, and in all of these settings, a goal can only be achieved if it can be verified. We suggest that, in manner similar to these examples, all reasonable

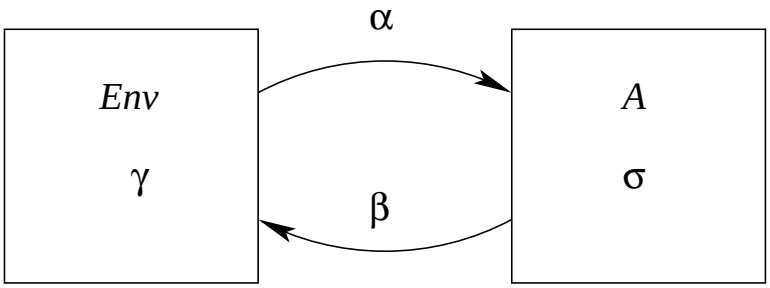

Figure 1: An agent A with environment *Env*

goals for communication can be formulated as syntactic goals in our framework, and therefore that language is not a barrier to semantic communication. (We elaborate on this further in Section 4.)

2 Generic Semantic Communication: Models and Theorems

In this section we describe the basic model of communicating agents and their goals. Goals come in two forms: *meta-goals*, which descibe our wishes, and *syntactic goals* which can be realistically achieved. This description is a consequence of our main theorems in this section, demonstrating that this latter kind of a goal is characterized by a verifiable condition.

2.1 Communicating Agents and Meta-goals

Agents: We start by formalizing one of the most basic ingredients in communication, namely the communicating *agent*. We hypothesize the agent as being in one of (possibly uncountably) many *states*. We denote the state space by Ω with some fixed state $\sigma_0 \in \Omega$ denoting a default home state. The agent has a bounded number k of *input* channels and a bounded number ℓ of *output* channels. For simplicity we assume all channels carry a finite sequence of symbols from a (finite) alphabet Σ.[1]

Mathematically, the agent A is thus defined by its input/output map $A : \Omega \times (\Sigma^*)^k \to \Omega \times (\Sigma^*)^\ell$, where $A(\sigma, \alpha_1, \ldots, \alpha_k) = (\tau, \beta_l, \ldots, \beta_\ell)$ implies that if the agent is in state ω and receives signals $\alpha_1, \ldots, \alpha_k$, then it changes state to τ and outputs signals $\beta_1, \ldots, \beta_k$.

When appropriate, we will require the state space to be countable and we will require the maps to be computable in polynomial time in appropriate parameters. We may also require the agent to have a distinguished "halting" state, $\omega_F \in \Omega$ satisfying $A(\omega_F, \alpha_1, \ldots, \alpha_k) = (\omega_F, \epsilon, \ldots, \epsilon)$ for all $\alpha_1, \ldots, \alpha_k$, where ϵ denotes the empty string.

Of the input and output channels, we distinguish one each as the *private input* channel and the *private output*. When considering probabilistic agents we will introduce the randomness explicitly through a second private input channel. All other inputs and outputs go into some *environment*, where the environment is itself be modeled as another agent *Env* (though at times, we may choose to model it as a collection of agents), which has its own state space Γ and input/output map, where the

[1] We note that the restriction that a channel only carry finitely many distinguishable symbols in one unit of time seems to be an unstated conclusion of Shannon's model. In particular even if a channel is capable of carrying an arbitrary real number in $[-1, 1]$, but introduces Gaussian error, the capacity of the channel reduces to a finite amount.

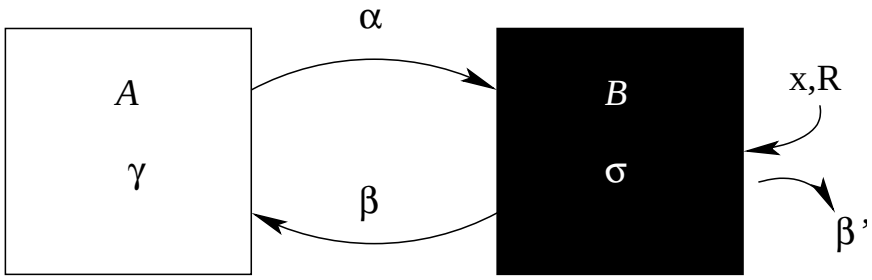

Figure 2: **Bob's Meta-Goal** is a function of everything except Bob's internal workings.

inputs to the environment are the outputs of our agent and vice versa. The one principal difference is that we will often allow *Env* to be a non-deterministic or probabilistic function. (Note that a collection of agents is also an agent and so this "bundling" of agents into a single "environment" is mathematically consistent.)

These definitions naturally fit the same basic outline of agents described elsewhere (e.g., in the AI textbook of Russell and Norvig [15] and references described there). Likewise, in the following section, we will introduce a notion of *goals* of communication for these agents, that resembles definitions appearing elsewhere. The difference here is that the goals, rather than the agents, come to play the central role in our study. In particular, the crucial difference between our setting and some settings considered elsewhere (e.g., the universal agents of Hutter [10]) is that we do not assume that "utilities" are computed for us by the environment—in our setting, it is up to the agent to decide whether or not it is satisfied.

Goals of communication: Our principal thesis is that communication can be made "robust" (across settings) if the "goals" of communication are explicit. To formalize this we first need to formalize "goals" of communication. In this work, we will consider two notions of "goals:" a global, objective definition of a goal (or "meta-goal," when we want to make the distinction clear) that will serve as our standard of correctness, and a local, subjective definition of a "syntactic goal" that we will see suffices for most practical purposes.

We will begin by defining "meta-goals." Given the mathematical definition of an agent and its communication environment, several natural definitions may occur. Perhaps the agent wishes to reach a certain state, or perhaps it wishes to see some pattern in its interaction. Somewhat surprisingly the real notion of a goal turns out to be a bit more subtle. Note that an intelligent communication agent can and should be able change its behavior and meaning of its own states in order to achieve its goals, and indeed, much of this work focuses on what algorithms the agent may employ to achieve its goals. In view of this flexibility it does *not* make sense to define goals in terms of the states of the agent! Even more surprisingly, typical goals seem to be about the environment and its capabilities and state, even though this may not be measurable by the agent. We simply ignore this obstacle for now – we will return to it when we consider syntactic goals – and define the goal as being described by a Boolean predicate G that takes as input a sequence of states of the environment and a transcript of interactions, and accepts or rejects, indicating whether the goal has been achieved or not. (See Figure 2.)

Definition 2 (Meta-goal) *A* meta-goal *is given by a function* $G : \mathcal{E} \times \left(\Gamma \times (\Sigma^*)^k \times (\Sigma^*)^\ell\right)^n \to \{0,1\}$ *where* $G(Env, \langle \gamma^{(i)}, \alpha_1^{(i)}, \ldots, \alpha_k^{(i)}, \beta_1^{(i)}, \ldots, \beta_\ell^{(i)} \rangle_{i \in [n]}) = 1$ *indicates that the goal is achieved at*

time n given that the environment Env evolved through states $\gamma_1, \ldots, \gamma_n$ in the first n rounds, and the agent received signals $\alpha_1^{(i)}, \ldots, \alpha_k^{(i)}$ at time i and sent signals $\beta_1^{(i)}, \ldots, \beta_\ell^{(i)}$ in response, for every $i \in [n]$.

As such the goal of an agent seems to completely at the "mercy" of the environment—here, we invoke the private input and output. These, as well as the private randomness, are assumed to be outside the control of the environment and thus allow the agent to be selfish. We will see the use of these in Section 3 where we give examples on how common goals are modeled.

In most examples below we will name our goal-interested agent "Bob," and consider the case when he is talking to a single agent within his environment, named Alice. For simplicity we will assume that Bob has one input channel coming from Alice and one output channel going to Alice. (All other signals are being ignored.) Also, if Alice and Bob are both deterministic (or probabilistic, but not non-deterministic), then the eventual achievement of the goal becomes purely a function of their initial states, and the private input for Bob (which is assumed to be held constant during an interaction). Thus we use the notation $G(A_\gamma \leftrightarrow B_\sigma(x), n)$ to denote the condition that the goal is achieved after n rounds of interaction by Alice A starting in state γ and Bob B starting in state σ with Bob's private input being x. If Bob is probabilistic, then we use $G(A_\gamma \leftrightarrow B_\sigma(x; R), n)$ to denote the same condition with Bob's randomness being R. If Alice is probabilistic, or we wish to ignore the specific randomness used by Bob, we continue to use the notation $G(A_\gamma \leftrightarrow B_\sigma(x), n)$ to denote the outcome, with the understanding that this is now a random variable.

Example goals: We now present a few representative examples of meta-goals (which may not be achievable as stated) to illustrate their versatility. We will also introduce a simple taxonomy of goals in Section 3.

Example 3 (Printer problem) *Here Bob B is a computer and Alice A is a printer. Bob's goal is to ensure that Alice enters a state from which every time Bob says "Print $\langle X \rangle$", Alice responds with "$\langle X \rangle$". Formally, the goal is given by $G(A, \gamma) = 1$ if $\forall x, A(\gamma, \text{Print } x) = (\gamma', x)$ for some γ'. A weaker goal, specific to Bob's current printing challenge x_0, would be $G'(A, \gamma, x_0) = 1$ if $A(\gamma, \text{Print } x_0) = (\gamma', x_0)$ for some γ', where x_0 denotes the* private *input of B.*

Example 4 (Computation) *In this setting, B wishes to compute some (hard) function f. The goal is simply defined so that $G(A_\gamma \leftrightarrow B_\sigma(x), n) = 1$ if Bob's last private output signal is $f(x)$.*

Example 5 (Turing test) *We recall Turing's famous test [18], which we can model as follows. We consider two classes of Alices, computers and humans, which we denote as $\mathcal{A}_1$ and $\mathcal{A}_2$. Env consists of a pair of agents, (A, A'), such that if $A \in \mathcal{A}_1$ then $A' \in \mathcal{A}_2$ and vice-versa. The goal is now defined so that $G((A, A')_\gamma \leftrightarrow B_\sigma, n) = 1$ if Bob's last private output signal is i where $A \in \mathcal{A}_i$.*

Helpful Alices and Universal Bobs: We now turn to our main objective, which is to model universal communication. This is the setting where Bob B wishes to communicate with Alice A, when he is not certain who she is, but knows or believes she comes from a broad class $\mathcal{A}$. We will give a minimal condition – "helpfulness" – that each Alice in $\mathcal{A}$ must satisfy for Bob to be able to successfully communicate with her, and we will give a definition of a "universal" Bob who can successfully communicate with every Alice in $\mathcal{A}$.

To formalize what it means to "communicate" we simply fix Bob's goal for communication, and seek conditions under which it can be reliably and universally achieved. The ability to communicate

universally thus depends on the collection of Alices $\mathcal{A}$ that Bob wishes to communicate with, and his goal G. For any given Alice $A \in \mathcal{A}$ it may be impossible for any Bob to be able achieve the goal G when communicating with her. (For instance, an Alice who refuses to communicate may not be very helpful to Bob, for any non-trivial goal.) Thus one needs to ensure that Alice is helpful. Defining this is somewhat subtle, since one should avoid definitions that bluntly force her to be helpful to our Bob B.

In the "universal setting" we model this as follows. From Alice's perspective, she is willing to talk to some Bob B_A and help him achieve his goal G. Furthermore, this Bob B_A is in some sense not more resourceful than our own Bob B. Generically, this is modeled by saying the $B_A \in \mathcal{B}$ for some (restricted) class $\mathcal{B}$.

We now have all the ingredients, namely G, $\mathcal{A}$ and $\mathcal{B}$, that go into the definition of helpfulness and universality. We give the formal definitions below.

Definition 6 ($(G, \mathcal{B})$-Helpful) *Alice A is said to be* $(G, \mathcal{B})$-helpful *for a goal G and class of Bobs $\mathcal{B}$ if there exists $B = B^A \in \mathcal{B}$ (with initial state σ_0) such that for every state γ of Alice and every private input x, there exists an n_0 such that with probability at least $1 - \mathrm{negl}(n_0)$ $\forall n \geq n_0$ $G(A_\gamma \leftrightarrow B_{\sigma_0}(x), n) = 1$*

Note that we require something stronger than just the fact that Alice helps Bob. The quantifiers on γ above require Alice to help Bob, effectively, no matter what her initial state is. Also, while the definition does not require n_0 to be small with respect to, say $|x|$, one can enforce such restrictions by picking the class $\mathcal{B}$ appropriately.

Ideally, we desire an efficent protocol for the goal, the interactive analogue of an algorithm—a well-defined sequence of steps of reasonable length that achieves the goal under any circumstances, when followed. In general, though, we wish to consider classes $\mathcal{A}$ for which there is no a priori bound on the resources required to communicate with the various members of $\mathcal{A}$. Thus, we model efficiency by introducing a class $\mathcal{B}$ of "efficient" algorithms (which should, in particular, reach a halting state for every input x) and require in each case that the behavior of our agent B should be simulateable by some member $B' \in \mathcal{B}$. We use a particularly strong notion of simulation, so that the requirement that B' simulates B restricts the resources available to B when B' comes from some computationally limited class $\mathcal{B}$. The precise notion that we use is described in Appendix A.

Definition 7 ($(G, \mathcal{B}, \mathcal{A})$-Universal Protocol) *Bob B is said to implement a* $(G, \mathcal{B}, \mathcal{A})$-universal protocol *if for every $A \in \mathcal{A}$ there exists a $B' \in \mathcal{B}$ (with initial state σ_0) such that for every private input x and every starting state γ of A*

- *there exists an n such that B halts after n rounds and $G(A_\gamma \leftrightarrow B_{\sigma_0}(x), n) = 1$ with probability at least $1 - \mathrm{negl}(|x|)$.*
- *B' simulates B with probability at least $1 - \mathrm{negl}(|x|)$.*

Note that we don't explicitly require the class of Alices to be helpful. Of course, we expect that Alice must be helpful to some agent in $\mathcal{B}$ for the universal protocol to succeed, and generally we will only consider $(G, \mathcal{B}, \mathcal{A})$-universal protocols for classes $\mathcal{A}$ of Alices who are $(G, \mathcal{B})$-helpful. We might, however, wish to demand that Alice helps agents in some class $\mathcal{B}'$ on which we impose additional

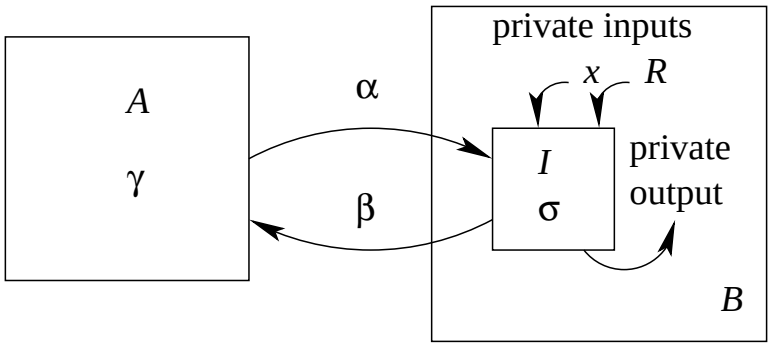

Figure 3: A controller B using an interpreter I to interact with an agent A

restrictions. In such cases, it will still be useful to know that B is not too powerful relative to the class $\mathcal{B}'$ and not too inefficient in the number of rounds of communication. In our theorems and examples in later sections we will comment on the efficiency of the universal protocol.

We use the following notation for "typical" classes of Alices.

Definition 8 (Typical classes) *For a goal G and class of Bobs $\mathcal{B}$, the "universal class" (denoted $\mathcal{A}_{G,\mathcal{B}}$) is defined to be $\{A : A \text{ is } (G,\mathcal{B})-\text{helpful}\}$. The class of all agents is denoted $\mathcal{A}_0$.*

2.2 Syntactic Goals: Interpreters and Verifiability

The general definition of goals of communication are too broad to be able to achieve it universally with a broad class of Alices. For instance, the goals may involve elements of Alice's state that are totally independent of Alice's communication. Many other barriers exist to universal communication and in this section we attempt to give a more effective description of a goal of communication – a "syntactic goal for communication" – and consider when it can be used by Bob to achieve a meta-goal.

For example, some goals obviously cannot be reliably achieved. Consider the following goal:

Example 9 (Guessing a coin toss) *Alice tosses a coin and randomly enters either state γ_1 or γ_2. $G(A,\gamma,\epsilon,\beta) = 1$ iff $\beta = i$ such that $\gamma = \gamma_i$. Any guess β made by Bob is wrong with probability at least $1/2$.*

This goal is particularly unfair to Bob in the sense that Alice does not even tell Bob which way the coin came up. Although Bob might have some hope at predicting the behavior of an efficient, deterministic Alice (rather than one who chooses her state by tossing coins), it is just as possible that, if Bob is also deterministic, Alice could compute the same function and transition to the opposite state—and in this case, he would be wrong with probability 1.

Interpreters and Feedback: In the setting we model, Bob wishes to vary his behavior so that he can successfully interact with Alice. It thus naturally turns out that it is useful for him to be able to compute some feedback, indicating whether or not his attempts are successful, using a "verification predicate."

In particular, generally we will wish to consider Bob's "various behaviors" to be taken from some efficiently enumerable class of algorithms; we refer to the specific algorithms as *interpreters*, and

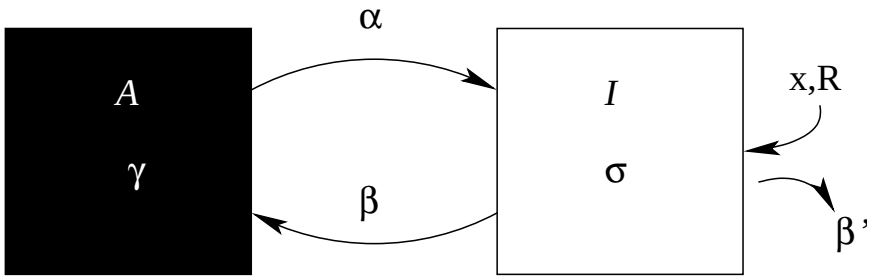

Figure 4: **Bob's Syntactic Goal** is characterized by a verification predicate that is a function of everything except Alice's internal workings.

we envision a separation of Bob into two entities, the interpreter and a controller (see Figure 3). Bob uses the interpreter to interact with Alice, while monitoring the conversation and state of the interpreter as the controller. Thus, we wish to allow the controller to be able to compute a function – the verification predicate – that provides feedback on the status of the goal from the history of computation and interaction of the interpreter. We will let $\mathsf{hist}(I_\sigma(x;R)|A_\gamma)$ denote the computation history of the (interpreter) agent I starting from state σ with private input x and randomness R when the environment consists of the agent A starting in state γ.

Definition 10 ($(G,\mathcal{I},\mathcal{A})$-Verification Predicate) *We say that V is a verification predicate for G with respect to $\mathcal{A}$ or, a* $(G,\mathcal{I},\mathcal{A})$-verification predicate *if for every $A \in \mathcal{A}$, $I \in \mathcal{I}$, and private input x, whenever $V(\mathsf{hist}(I_\sigma(x;R)|A_\gamma)) = 1$, at the final round n of the interaction, $G(A_\gamma \leftrightarrow I_\sigma(x;R), n) = 1$ with probability $1 - \mathrm{negl}(|x|)$ over R.*

The difference between a verification predicate and the predicate G defining the goal is not merely that V is "weaker" (in the sense that $V = 1 \Rightarrow G = 1$ but $G = 1 \not\Rightarrow V = 1$) but that while G is blind to the internal workings of the agent B, V sees everything about the interpreter agent I, but is rather blind to the states and transitions of A. (Contrast Figure 4 with Figure 2.)

For V to be of any practical use, we must restrict it even further, so that it can be computed by the controller. Thus, we finally arrive at the definition of a syntactic goal, our main definition in this work:

Definition 11 ($(G,\mathcal{B},\mathcal{A})$-Syntactic Goal) *A syntactic goal for communication for G with respect to $\mathcal{B}$ and $\mathcal{A}$, or* $(G,\mathcal{B},\mathcal{A})$-syntactic goal *is given by a pair $(V,\mathcal{I})$ such that*

- *$\mathcal{I}$ is an efficiently enumerable class of algorithms*
- *V is a $(G,\mathcal{I},\mathcal{A})$-verification predicate*
- *V is computable from the efficient enumeration simulation of agents in $\mathcal{I}$ by agents in $\mathcal{B}$.*

The precise definitions of efficient enumerability and our computability requirement are given in Appendix A.

We note that the verification predicate used in a syntactic goal and a universal protocol are both algorithms that can be run by Bob. The difference between the two is, naturally, that a universal

protocol tells Bob whether to continue the interaction and what to say next if so, whereas the verification predicate merely tells Bob whether or not G was achieved by some interpreter I.

Helpful Alices for Syntactic Goals: Just as we encountered earlier when considering meta-goals, whether or not Bob has any hope of succeeding with a class of Alices $\mathcal{A}$ depends on his choice of syntactic goal $(V, \mathcal{I})$. This raises the need for a definition of "well-designed" syntactic goals which do not eliminate the chance of success with any Alices in $\mathcal{A}$.

We know that for a particular $(V, \mathcal{I})$, it may be impossible for any interpreter $I \in \mathcal{I}$ to satisfy the verification predicate V with some given Alice $A \in \mathcal{A}$. The situation is particularly acute here, since asking Alice to satisfy some $(G, \mathcal{I}, \mathcal{A})$-verification predicate V is asking more than that she can merely satisfy G. (For example, our goal might be to find new theorems and V might require proofs of all of her assertions, which she might be unable or unwilling to provide.) This motivates extending the definition of "helpful" Alices to syntactic goals below:

Definition 12 ($(V, \mathcal{I})$-Helpful) *Alice A is said to be* $(V, \mathcal{I})$-helpful *for a predicate V and class of agents $\mathcal{I}$ if there exists $I = I^A \in \mathcal{I}$ (with initial state σ_0) such that for every state γ of Alice and every private input x, $V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma)) = 1$ with probability at least $1 - \mathrm{negl}(n)$.*

Since a Bob in $\mathcal{B}$ can succeed using a $(G, \mathcal{B}, \mathcal{A})$-syntactic goal $(V, \mathcal{I})$ whenever some $A \in \mathcal{A}$ is $(V, \mathcal{I})$-helpful by simulating I^A, we know that $(V, \mathcal{I})$-helpfulness for such a syntactic goal immediately implies $(G, \mathcal{B})$-helpfulness. A Bob who wishes to achieve his goal with every $(G, \mathcal{B})$-helpful Alice (i.e., with every member of $\mathcal{A} \cap \mathcal{A}_{G,\mathcal{B}}$) using a particular $(V, \mathcal{I})$ would need the two conditions to be equivalent, though—which is thus when we call the syntactic goal "well designed."

Definition 13 (Well-designed Syntactic Goal) *We will speak of a $(G, \mathcal{B}, \mathcal{A})$-syntactic goal $(V, \mathcal{I})$ as* well-designed *if $\forall A \in \mathcal{A}$ $(G, \mathcal{B})$-helpfulness implies $(V, \mathcal{I})$-helpfulness.*

Naturally, we would like to have well-designed syntactic goals, but it is not obvious when they would exist or how to construct them. Surprisingly, we will see next that it is a consequence of our main theorem that whenever a $(G, \mathcal{B}, \mathcal{A}_{G,\mathcal{B}})$-universal protocol exists for a "standard" class $\mathcal{B}$, a well-designed syntactic goal exists (Corollary 15).

2.3 Main Theorems

We are now ready to state our main theorems outlining the relationship between universal protocols and syntactic goals. Roughly, we show two things:

1. A syntactic goal is sufficient for construction of a universal protocol (Theorem 14).
2. A syntactic goal is necessary for a universal protocol; in particular, if the protocol is designed to work with a sufficiently broad class of Alices, then the goal must be verifiable with respect to *all* Alices, i.e., even malicious ones (Theorem 18).

A "standard" class of agents is one that satisfies a few properties typical of classes of algorithms defined by their asymptotic computational complexity—in particular, the class of polynomial-time agents, denoted $\mathcal{P}$, and the class of polynomial-time agents that only use logspace, denoted $\mathcal{L}$, are both examples of "standard" classes. (we give the precise definition in Appendix A).

Theorem 14 (Equivalence) *For any standard class of agents $\mathcal{I}$, there is a $(G, \mathcal{I}, \mathcal{A})$-universal protocol iff there is a $(G, \mathcal{I}, \mathcal{A})$-syntactic goal $(V, \mathcal{I})$ such that every $A \in \mathcal{A}$ is $(V, \mathcal{I})$-helpful.*

The intuition underlying this theorem is that for correctness, a protocol needs to "know" when the goal has been achieved since it is only allowed to halt before achieving its goal with negligible probability. On the one hand, this allows us to extract some verification predicate from the protocol; since the protocol works for every $A \in \mathcal{A}$, every $A \in \mathcal{A}$ is helpful for the V we construct in this way. On the other hand, standard classes are efficiently enumerable, and the only thing preventing an efficient enumerator for $\mathcal{I}$ from being a universal protocol (when it exists) is that it needs to stop enumerating, which is achieved by the feedback from V. The full proof appears in Appendix B.

We also obtain the following immediate corollary for $(G, \mathcal{B})$-helpful Alices (the class $\mathcal{A}_{G,\mathcal{B}}$):

Corollary 15 (Sufficiency) *For any standard class of agents $\mathcal{B}$ if there is a $(G, \mathcal{B}, \mathcal{A}_{G,\mathcal{B}})$-universal protocol then a well-designed $(G, \mathcal{B}, \mathcal{A}_{G,\mathcal{B}})$-syntactic goal exists.*

We now recall the definition of a "semantically closed class" of Alices from prior work [11] since it will be useful to us as our notion of "sufficient broadness."

Definition 16 (Semantically closed class) *We say that a class of Alices $\mathcal{A}$ is* semantically closed *if for every injective function f where f and f^{-1} are computable in logspace and for every logspace Alice A', for each member $A \in \mathcal{A}$, the following incarnation $A'^{f}_{A'}$ is also in $\mathcal{A}$: for state (γ_1, γ_2) (where γ_1 is a state of A and γ_2 is a state of A') and β is in the range of f, if $A(\gamma_1, f^{-1}(\beta)) = (\gamma_1', \alpha)$, then $A'^{f}_{A'}((\gamma_1, \gamma_2), \beta) = ((\gamma_1', \gamma_2), f(\alpha))$. For β outside the range of f, if $A'(\gamma_2, \beta) = (\gamma_2', \alpha)$, then $A'^{f}_{A'}((\gamma_1, \gamma_2), \beta) = ((\gamma_1, \gamma_2'), \alpha)$.*

We also introduce one more definition (a technical requirement):

Definition 17 (Simulation closed goal) *We say that a goal G is* simulation closed *if whenever A simulates A' and $G(A_\gamma \leftrightarrow B_\sigma, n) = 1$, also $G(A'_\gamma \leftrightarrow B_\sigma, n) = 1$.*

Any G which treats Alice as a black box is clearly simulation closed, but, for example, one which aims to test her running time would not be.

Theorem 18 (Verifiability) *For any $(G, \mathcal{B}, \mathcal{A})$-syntactic goal $(V, \mathcal{I})$, if G is simulation closed and $\mathcal{A}$ is semantically closed and nonempty then $(V, \mathcal{I})$ is a $(G, \mathcal{B}, \mathcal{A}_0)$-syntactic goal.*

This is the generalization of our limitation result from previous work: roughly, if some malicious $\tilde{A} \in \mathcal{A}_0$ could trick V into outputting 1 when G was not satisfied on some instance with high probability, then we can construct some A' in the semantic closure of $\mathcal{A}$ that also achieves this, so V would not be a verification predicate. Again, the full proof appears in Appendix B.

Note now, the "universal classes" of Alices are sufficiently broad, i.e.,

Proposition 19 *For any standard class $\mathcal{B}$, $\mathcal{A}_{G,\mathcal{B}}$ is semantically closed.*

This is so since for any Alice $A \in \mathcal{A}_{G,\mathcal{B}}$ helping some $B^A \in \mathcal{B}$, the Alice $A^f_{A'}$ helps an B' who applies f to each outgoing message and f^{-1} to each incoming message. We can then obtain the following strong version of Corollary 15

Corollary 20 (Strong sufficiency) *For any standard class of agents $\mathcal{B}$, simulation closed G, and nonempty $\mathcal{A}_{G,\mathcal{B}}$, if there is a $(G, \mathcal{B}, \mathcal{A}_{G,\mathcal{B}})$-universal protocol then a well-designed $(G, \mathcal{B}, \mathcal{A}_0)$-syntactic goal exists.*

This, together with Corollary 15, motivates a study of semantic communication where the semantics are *defined by* the verifiable conditions achieved. To this end, we next show how a variety of natural communication problems can be formulated in our framework.

3 Syntactic goals for communication

We now turn to the problem of constructing syntactic goals for communication corresponding to meta-goals that formalize a host of natural communications problems. Recall that we use $\mathcal{P}$ to denote the class of polynomial-time agents and $\mathcal{L} \subset \mathcal{P}$ to denote the class of polynomial-time agents that only use logspace. (Note that these are standard classes in the sense we require: see Section A for details.)

We will use the following simple taxonomy of goals for communication to guide our discussion:

Definition 21 (Intellectual goal) *We say that a goal G is an* intellectual goal *if it is a function of Bob's private outputs. It is* purely intellectual *if it is* only *a function of these private inputs and and outputs.*

Definition 22 (Physical goal) *We say that a goal G is* physical *if it is a function of either the states of Env or inputs from Env. It is* purely physical *if it is independent of Bob's private outputs.*

So, the computational goal in Example 4 is a *purely intellectual* goal, whereas the goal of printing in Example 3 is *purely physical*. Finally, the goal of the Turing test examiner in Example 5 is neither (it is both intellectual and physical). Note that purely intellectual goals are necessarily simulation closed. Physical goals may not be simulation closed, but the examples of purely physical goals we consider next also turn out to satisfy this property.

We are interested in goals for which Bob obtains some benefit from Alice's help. In the interest of formulating this notion precisely, we will let Φ denote the Alice who always responds with the empty string. Since this Alice can be simulated easily by Bob, she will be our canonical example of a "trivial" Alice in the following sense:

Definition 23 (Nontrivial goal) *We say that a goal G is* nontrivial *with respect to a class of agents $\mathcal{B}$ if for every Bob in $\mathcal{B}$ there are infinitely many private inputs x such that the probability that Bob achieves G on private input x when interacting with Φ is at most* $1/3$.

We will use nontriviality primarily as a "sanity check" on our definitions of intellectual goals.

3.1 Computational goals

We first consider computational meta-goals, as introduced in Example 4. We begin by showing how results from interactive proofs allow us to easily obtain the results from prior work [11] which characterizes the class of computational meta-goals that can be achieved by polynomial time bounded agents. We then show an application to delegating polynomial-time computations to unknown servers by logspace agents, and conclude by discussing some challenges to obtaining a similar characterization of the class of computational meta-goals that can be achieved by logspace agents.

3.1.1 Polynomial time agents seeking wisdom

Returning to the fantastic scenario sketched in prior work [11], we suppose that Bob has been contacted by some computationally powerful extraterrestrial Alice, and we ask what Bob can learn by communicating with her. We do not limit Alice's computational power a priori: we imagine that she can solve *any* computational problem, and thus see how Bob's *lack of understanding* places inherent limitations on the communication. Again, we model the lack of understanding by considering a *class* of Alices, $\mathcal{A}$, defined by the existence of a suitable communications protocol (i.e., "language"), and we ask that Bob's protocol work with *every* member of the class $\mathcal{A}$. In this case, since it turns out that verifiability the goal is necessary and sufficient, the inability for Bob to verify solutions to problems outside PSPACE prevents Bob from using Alice's help to solve such problems, but interactive proofs allow Bob to use Alice's help to solve PSPACE-complete problems. We will show how these results can be recovered in the framework of this paper.

Technically, we proceed as follows. We start by recalling the computational meta-goals associated with a class of computational problems, and invoke Theorem 14 to cast the existence of protocols in terms of the existence of syntactic (verifiable) goals. We then show, on the one hand, how Theorem 18 leads to a computational limitation result, and on the other how the existing protocols for PSPACE [13, 16] lead to the consrtruction of a syntactic goal. Finally, we show that the goal is well-designed, thus yielding a universal protocol for the computational meta-goal.

Main definitions in this setting: Let the class of Bobs $\mathcal{B} = \mathcal{P}$, and recall the computational goal G_Π from Example 4 for a decision/promise problem Π. Recall that $\mathcal{A}_{G_\Pi,\mathcal{P}}$ is the class of $(G_\Pi, \mathcal{P})$-helpful Alices: those Alices A for which there exists some $B^A \in \mathcal{P}$ who reliably achieves G_Π (i.e., computes Π on its private input) when interacting with A. Observe that, as long as we assume that all states of Alice can be reached by some message history, this is exactly the same as the class of Alices who are "Π-helpful," as defined in prior work [11]: if Alice helps some Bob compute Π with probability 2/3, she helps some agent B' – who simulates $O(n)$ interactions with Alice and takes a majority vote – compute Π with probability at least $1 - 2^{-n}$.

We wish to know, for what class of problems can we hope to design a $(G_\Pi, \mathcal{P}, \mathcal{A}_{G_\Pi,\mathcal{P}})$-universal protocol: a protocol that, for every $(G_\Pi, \mathcal{P})$-helpful A, achieves G_Π and is simulated by some polynomial-time agent. Since Bob achieves G_Π precisely when A helps him decide Π, notice that such a Bob is essentially a Π-universal Bob, as defined in prior work[2].

Theorem 14 tells us that it is necessary and sufficient that we exhibit a $(G_\Pi, \mathcal{P}, \mathcal{A}_{G_\Pi,\mathcal{P}})$-syntactic goal $(V, \mathcal{P})$ that is "well-designed," which consists of a $(G_\Pi, \mathcal{P}, \mathcal{A}_{G_\Pi,\mathcal{P}})$-verification predicate V that

[2]It is similarly not hard to see that a Π-universal Bob can have its success probabilities amplified by repetitions with a majority vote to construct a $(G_\Pi, \mathcal{P}, \mathcal{A}_{G_\Pi,\mathcal{P}})$-universal protocol; the only difference is that a $(G_\Pi, \mathcal{P}, \mathcal{A}_{G,\mathcal{P}})$-universal protocol runs in polynomial time with $1 - \text{negl}(n)$ probability, whereas a Π-universal Bob was defined to run in expected polynomial time.

can be evaluated for our enumeration of agents in $\mathcal{P}$ by agents in $\mathcal{P}$. In particular, V should look at the computation history of the current agent in the enumeration and only accept when that agent has achieved G_Π (since it is a verification predicate), but should be able to accept for some agent in $\mathcal{P}$ whenever A is $(G_\Pi, \mathcal{P})$-helpful (i.e., it is well-designed).

Characterization of problems with polynomial-time universal protocols: So, the existence of our universal protocols is equivalent to the existence of well-designed $(G_\Pi, \mathcal{P}, \mathcal{A}_{G_\Pi,\mathcal{P}})$-syntactic goals. Theorem 18, on the other hand, tells us that such a syntactic goal has a very strong verifiability requirement: it is actually a $(G_\Pi, \mathcal{P}, \mathcal{A}_0)$-syntactic goal, where $\mathcal{A}_0$ is the class of *all* Alices. (As in the soundness of an interactive proof system.) This will allow us to obtain the impossibility result stated in prior work:

Theorem 24 *Let Π be a decision problem that is not in PSPACE. Let $\mathcal{A}$ be a nonempty semantically closed class of Π-helpful Alices. Then for every probabilistic algorithm B, there exists an $A \in \mathcal{A}$ such that B fails to decide Π with the help of A.*

Proof Fix a nonempty semantically closed class of Π-helpful Alices $\mathcal{A}$, and a $(G_\Pi, \mathcal{P}, \mathcal{A})$-universal B. We will show that this forces $\Pi \in$ PSPACE.

By Theorem 14, there exists a $(G_\Pi, \mathcal{P}, \mathcal{A})$-syntactic goal, $(V, \mathcal{P})$. On the other hand, since satisfying this goal depends only on Bob's private outputs which are identical regardless of whether Bob converses with some Alice A or some Alice A' who simulates A, this goal is simulation closed and hence by Theorem 18, $(V, \mathcal{P})$ must also be a $(G_\Pi, \mathcal{P}, \mathcal{A}_0)$-syntactic goal. Therefore, in PSPACE, we can compute the probabilities over the coins R used by the universal protocol that V accepts (polynomial-length) computation histories for input x in which the agent outputs 0 or 1 in the final round. By the correctness of the verification predicate for $\mathcal{A}_0$, and nonemptiness of $\mathcal{A}$, the value of $\Pi(x)$ will be much more likely. This gives a scheme for computing $\Pi(x)$ in PSPACE. ■

We *can* design a syntactic goal for problems in PSPACE, though. Informally, our syntactic goal is to find not only whether or not "$x \in \Pi$" holds, but also obtain an interactive proof (i.e., for Shamir's verifier [16, 13]). Moreover, if Π is also PSPACE-complete, we claim that this syntactic goal is well-designed. (Clearly such a G_Π is also nontrivial unless PSPACE = BPP.) Formally, this yields the main theorem from our prior work:

Theorem 25 *For every PSPACE complete problem Π, there is a Bob that is $(G_\Pi, \mathcal{P}, \mathcal{A}_{G_\Pi,\mathcal{P}})$-universal.*

Proof We will begin by observing that the sum-check protocol [16, 13], as a *public-coin protocol*, readily yields a well-designed syntactic goal. (This observation simplifies our analysis considerably but is unessential, cf. Section 3.1.3, where this distinction is important).

Construction: For any PSPACE-complete Π, let $\Pi' = \{(x, b) : b = \Pi(x)\}$, and let V_Π be the public-coin verifier for an interactive proof for membership in Π' with perfect completeness and negligible soundness error, guaranteed to exist by Shamir's Theorem [16].

$V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma))$ accepts if, when we simulate $V_\Pi(x, 0)$ and $V_\Pi(x, 1)$ in sequence using the private outputs of I as the messages from the prover, and using the coin tosses of I following each output as the verifier's randomness in the corresponding round, if the final private output of I is b, $V_\Pi(x, b)$ accepts.

Analysis: It is easy to see that V is a $(G_\Pi, \mathcal{P}, \mathcal{A}_0)$-verification predicate: if the final private output of I is $b \neq \Pi(x)$, then V accepting implies that $V_\Pi(x, b)$ accepted the preceding transcript as an interactive proof of $(x, b) \in \Pi'$. Since the coin tosses on each round were independent of the prover's message in the prior round, the probability that I generates a transcript that V accepts is equal to the probability that V_Π would accept in the interaction with a prover who sends the outputs of I as its messages. Thus, by the assumed soundness of V_Π, this probability is negligible, and V is a $(G_\Pi, \mathcal{P}, \mathcal{A}_0)$-verification predicate. Since $\mathcal{P}$ is efficiently enumerable and each round of V_Π can be simulated in polynomial time, V can also be computed from the efficient enumeration with only an additional polynomial factor overhead by simulating one check by V_Π following each output by $I \in \mathcal{P}$, so $(V, \mathcal{P})$ is a syntactic goal.

On the other hand, by the perfect completeness of V_Π, for every sequence of coin tosses of I, there is some sequence of messages such that $V_\Pi(x, \Pi(x))$ accepts. Since the optimal prover's next message can be computed from the current history and x in polynomial space, there is a polynomial time reduction from the optimal prover's messages to instances of Π, where these messages lead V to accept with probability 1. Therefore, if A helps some $B' \in \mathcal{P}$ decide Π, there is some polynomial time I^A such that $V(\mathsf{hist}(I^A_{\sigma_0}(x)|A_\gamma))$ accepts, as claimed, so $(V, \mathcal{P})$ is a well-designed $(G_\Pi, \mathcal{P}, \mathcal{A}_0)$-syntactic goal. It now follows from Theorem 14 that a universal protocol exists (for the universal class $\mathcal{A}_{G_\Pi, \mathcal{P}} \subset A_0$). ■

3.1.2 Logspace agents delegating computation

Section 3.1.1 demonstrated how any agent who can compute PSPACE can communicate the results to a polynomial-time universal agent. If we believe that PSPACE-complete problems cannot be solved by any agent in a reasonable amount of time, though, it is hard to see what practical value the resulting protocol has, if any. In light of this, the more interesting question to ask is if, in a similar way, polynomial-time agents can communicate the results of "hard" computations to some weaker universal agent; in this section, we will envision the weaker universal agent "delegating" the problem to the polynomial-time agent. For example, we might imagine wishing to use a weak handheld wireless device to solve some hard problem, e.g., solving a linear program, using the aid of some foreign server that happens to be within its range. Our objective here will be to design a protocol for the weak device that allows it to use *any* server which can be accessed by a lightweight communications protocol. (We will investigate another variant of this question, about the sharing of knowledge among polynomial-time agents – as opposed to delegating computation to more powerful agents – in Section 3.3.)

Technically, we will use $\mathcal{L}$, the class of logspace and polynomial-time agents as our class of weak devices, and we will see that we can use a recent result of Goldwasser et al. [9] to scale down the syntactic goal of Section 3.1.1 to an interesting well-designed goal verifiable in logspace. In short, we use their verifier to construct a syntactic goal $(V, \mathcal{L})$ and invoke Theorem 14 to obtain a protocol. We then show that the syntactic goal is well-designed, so the protocol we thus obtain is a universal protocol (for the class of all Alices who help some agent in $\mathcal{L}$).

Let $\Pi \in \mathrm{P}$ be given; again, for the corresponding computational meta-goal G_Π of Example 4, recalling that $\mathcal{A}_{G_\Pi, \mathcal{L}}$ is the class of all Alices who help some $B \in \mathcal{L}$ achieve G_Π, we wish to construct a $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi, \mathcal{L}})$-universal protocol. Note that such a G_Π is clearly nontrivial unless $\mathrm{BPL} = \mathrm{P}$. The following theorem asserts that this is possible:

Theorem 26 *For any P-complete* Π *(under logspace reductions), if* G_Π *is the corresponding computational meta-goal, then there is a* $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$*-universal protocol.*

Proof Theorem 14 tells us that this is equivalent to constructing a $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$-syntactic goal $(V, \mathcal{L})$, which is how we proceed:

Construction: Let $\Pi' = \{(x, b) : b = \Pi(x)\}$; observe that Π' is also P-complete. Goldwasser et al. show that there is a public-coin interactive proof system (P_Π, V_Π) for Π' such that V_Π runs in logspace. We can assume wlog (by sequential repetition) that this proof system has negligible soundness error. Now, our verification predicate $V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma))$ accepts if V_Π would accept (x, b) using the private outputs of I as the messages from the prover and the coin tosses in $\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma)$ following each private output as its random moves in that round, and I outputs b as its final private output.

Analysis: As in the proof of Theorem 25, the key is that the coin tosses for each round are independent of the prover's prior message, since I tosses these coins after it outputs its prover message. Therefore V accepting implies that V_Π accepted the preceding interaction as an interactive proof of $(x, b) \in \Pi'$, so the probability that V accepts the messages generated by I is equal to the probability that V_Π would accept (x, b) when interacting with a prover who sends the private outputs of I as its messages. Since we assumed that V_Π only accepts a "proof" that x takes some other value than $\Pi(x)$ with negilgible probability, an interpreter who satisfies V only outputs $b \neq \Pi(x)$ with negligible probability. It follows that V is a $(G_\Pi, \mathcal{L}, \mathcal{A}_0)$-verification predicate. Moreover, since V_Π uses only $O(\log n)$ space, we can simulate its computation in parallel with any logspace interpreter I in $O(\log n)$ space overall, and thus this is a $(G_\Pi, \mathcal{L}, \mathcal{A}_0)$-syntactic goal.

There is a reduction due to Condon and Ladner [6] that shows how, for a public-coin interactive proof system with a logspace verifier, the next bit from an optimal prover can be computed by a reduction to linear programming. Since the reduction from Condon and Ladner shows that the next bit from the optimal prover can be computed in polynomial time given x and the current state of V_Π (which can be maintained in logspace), since Π is P-complete under logspace reductions, we will now see that V is well-designed. By the completeness of the proof system (V_Π, P_Π), there is some prover strategy P_Π that makes V_Π accept. Since there is a logspace reduction from the problem of computing the next bit from the optimal prover to Π, there is also a logspace interpreter I^A such that $V(\mathsf{hist}(I^A_{\sigma_0}(x)|A_\gamma))$ accepts for any A who helps some logspace B' decide Π. Thus, $(V, \mathcal{L})$ is a well-designed $(G_\Pi, \mathcal{L}, \mathcal{A}_0)$-syntactic goal, and by Therorem 14, we therefore have a $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$-universal protocol. ■

3.1.3 Classifying the computational capabilities of logspace universal protocols

Of course, our objective in Section 3.1.1 was not just to construct a universal protocol, but to give a characterization of the class of problems which have polynomial-time universal protocols. We might also wish characterize the class of computational goals that can be achieved by logspace universal protocols. That is, we might again imagine that Bob is communicating with an all-powerful Alice, and we might again wish to know how his lack of understanding limits what he can learn from Alice—with the twist now that Bob is additionally restricted to run in logspace. Formally, this corresponds to finding the class of problems Π for which there exist $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$-universal protocols.

We note that since $\mathcal{A}_{G_\Pi,\mathcal{L}}$ is semantically closed and nonempty, and $\mathcal{L} \subset \mathcal{P}$, Theorem 24 shows one limitation: any such Π is contained in PSPACE. This is the best limitation result we know; it is possible that this is tight, but we do not know how to show this. One cannot, for example, insert the reduction of Condon and Ladner [6] in the proof of Theorem 24 and obtain a tighter result since this reduction only applies to public-coin protocols, whereas a universal protocol is, in general, a private-coin protocol.

By contrast, there is a construction due to Condon [5] that can be used to convert a polynomial-time verifier into a verifer that runs in polynomial-time and logspace, e.g., given $\Pi \in$ PSPACE, we would let V_Π be the verifier for an interactive proof for membership in $\Pi' = \{(x,b) : \Pi(x) = b\}$ as before. We can use Condon's construction to convert V_Π into V'_Π; now, $V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma))$ checks that

1. I is a particular syntactic composition of V'_Π and I' for some I' (i.e., so I simulates $V'_\Pi \leftrightarrow I'$)
2. V'_Π enters an accepting state in the simulation
3. I outputs b such that V'_Π accepted (x,b)

By the soundness of V'_Π, this is a verification predicate, and thus $(V,\mathcal{L})$ is a $(G_\Pi,\mathcal{L},\mathcal{A}_0)$-syntactic goal. Although the existence of the honest prover guarantees that the class of $(V,\mathcal{L})$-helpful Alices is nonempty, it isn't clear whether or not this is a well-designed syntactic goal, and hence it isn't clear whether or not a universal protocol exists for the class $\mathcal{A}_{G_\Pi,\mathcal{L}}$ for a PSPACE-complete Π. We can show that the key issue here is not just the efficiency of the verifier in an interactive proof for PSPACE but also how efficient the prover can be for such an efficiently verifiable proof system.

Competitive interactive proofs: Competitive interactive proofs were introduced by Bellare and Goldwasser [3] to study the complexity of the prover in an interactive proof system. In their terminology, (P,V) was a competitive interactive proof system for a decision problem Π if P was a probabilistic polynomial-time oracle machine such that (P^Π,V) was an interactive proof system for Π. 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 P and V using an oracle for Π allows one to generate "proofs" in polynomial time given the ability to decide Π.

In a similar spirit, we introduce the following modification of their definition to logspace-bounded machines:

Definition 27 (Logspace competitive interactive proof) *Let P be a probabilistic logspace and polynomial-time oracle machine, and let V be a probabilistic logspace and polynomial-time machine. We say that (P,V) is a* logspace competitive interactive proof system for a problem Π *if*

1. *For every $x \in \Pi$, the probability that V accepts when interacting with P^Π on common input x is at least $2/3$*
2. *For every $x \notin \Pi$, and every interactive function $\tilde{P}$, the probability that V accepts when interacting with $\tilde{P}$ on common input x is at most $1/3$*

This definition has similar virtues to its polynomial-time counterpart—if it exists, then one can generate a proof in logspace given the ability to decide Π. This is the main observation involved in the following theorem:

Theorem 28 *There is a logspace competitive interactive proof system for problems* Π *and* $\bar{\Pi}$ *(*$\bar{\Pi}(x) = \neg\Pi(x)$*) iff there is a* $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$*-universal protocol.*

The proof of this theorem is straightforward; it is contained in Appendix B for completeness. Of course, a logspace competitive interactive proof system for any PSPACE-complete problem implies an interactive proof system for the language's complement since PSPACE is closed under complement. (We can observe such a step in action in the proofs of Theorem 25 and Theorem 26) Unfortunately, we don't know of such a proof system – for example, in particular it isn't clear how we could simulate the prover in Condon's proof system [5] without using polynomial space – and the capabilities of logspace universal protocols remains open.

3.2 Control-oriented goals

We next consider some purely physical goals. Designing syntactic goals for these meta-goals is straightforward, especially if Bob only wants to manipulate his input signals, as we will see. Both examples follow the same rough outline: we define an appropriate meta-goal, and use Theorem 14 to reduce the problem of constructing a universal protocol for the meta-goal to the problem of constructing a well-designed syntactic goal, which is then easily achieved.

3.2.1 Control/function evaluation

We again let the class of Bobs $\mathcal{B} = \mathcal{P}$. We return to the weaker printing goal G' from Example 3, and generalize it slightly to a goal G'' as follows. The informal goal is that Bob wants Alice to print a string x on her output tape (modeling the task of printing) or, more generally, wants her to print $f(x)$ where f is some polynomial-time computable function, and f and x are given as private inputs to Bob. If $f(x)$ is a nonempty string for infinitely many x, it is easy to see that this goal is nontrivial.

Recall that $A_{G'',\mathcal{P}}$ is defined to be the class of $(G'', \mathcal{P})$-helpful Alices: those Alices A for which there is some $B^A \in \mathcal{P}$ which can reliably get A to print any $f(x)$ on its output. We wish to design a $(G'', \mathcal{P}, \mathcal{A}_{G'',\mathcal{P}})$-universal protocol, which for every $A \in \mathcal{A}_{G'',\mathcal{P}}$ would get A to print any $f(x)$ from any state γ, and would be simulateable by some agent in $\mathcal{P}$. Theorem 14 tells us that we can, equivalently, attempt to construct a well-designed $(G'', \mathcal{P}, \mathcal{A}_{G'',\mathcal{P}})$-syntactic goal $(V, \mathcal{P})$, i.e., where V should be a function computable by agents in $\mathcal{P}$ that looks at the computation histories of agents in $\mathcal{P}$, tells us if an agent has achieved G'', and which can succeed for every $A \in \mathcal{A}_{G'',\mathcal{P}}$ with some $I^A \in \mathcal{P}$ on every input (f, x) and state of Alice γ.

This is easy to do, though, since G'' is a simple function of Alice's output. Formally, $V(\mathsf{hist}(I_{\sigma_0}(f,x)|A_\gamma))$ first computes $f(x)$ and then accepts if some message from A in $\mathsf{hist}(I_{\sigma_0}(f,x)|A_\gamma)$ is $f(x)$. Since f is assumed to be polynomial time computable, V is clearly also polynomial time computable. It is easy to see that $(V, \mathcal{P})$ is a well-designed $(G'', \mathcal{P}, \mathcal{A}_{G'',\mathcal{P}})$-syntactic goal since G'' is satisfied iff V is.

3.2.2 Searching

Suppose that Bob is searching for an object in *Env* satisfying some $O(n^k)$-time verifiable property, V_O, and wishes to enlist Alice's help in the search. We will suppose that Bob can examine a location ℓ (labeled by a string) in his environment by sending *Env* the message ℓ; we denote the response of *Env* by $Env(\ell)$ (ignoring the state).

Formally now, we fix *Env* and the class of Bobs $\mathcal{B} = \mathcal{P}$, and define both $G((A_\gamma, Env) \leftrightarrow I_{\sigma_0}(V_O), n)$ and $V(\mathsf{hist}(I_{\sigma_0}(V_O)|(A, Env))$ to accept if for some message ℓ sent to *Env* by I, $V_O(Env(\ell))$ accepts, and note that V can be evaluated in time $O(n^{k+1})$. Thus, it is immediate that $(V, \mathcal{P})$ is a well-designed $(G, \mathcal{P}, \mathcal{A}_{G,\mathcal{P}})$-syntactic goal. By Theorem 14, we therefore have a $(G, \mathcal{P}, \mathcal{A}_{G,\mathcal{P}})$-universal protocol, which efficiently finds suitable objects with the help of an Alice A whenever some $B^A \in \mathcal{P}$ could. Furthermore, depending on *Env*, the goal may be nontrivial, for example if the environment does not initially contain a string satisfying V_O and Bob cannot introduce such a string into the environment on his own.

We also remark that there is a close connection between our model of searching here and the model of computational awareness proposed by Devanur and Fortnow [7]. In their model, Bob is endowed with an enumeration procedure M with oracle access to an environment and some input context, which in our case would be a description of V_O. We note that our model extends their formal model slightly, as we take *Env* to be an entity that is interacting with B (and A, so that B and A can modify *Env*—we note that this is also implicitly done by Devanur and Fortnow, but they still formally model *Env* as an oracle). Recall that Devanur and Fortnow *define* the unawareness of an object with respect to I to be the time for I to print it, so V is accepting iff I has unawareness of a satisfactory ℓ (with Alice's help) that is less than its time bound.

3.3 Intellectual curiosity

Not all intellectual goals involve computing a hard function on a specific instance. In particular, we may wish to grant Alice the latitude to suggest a context (e.g., where she knows how to do something) as in the following example: suppose Bob wishes to learn something "interesting" from Alice, e.g., a proof of a "deep" theorem. Informally, we would like to obtain a theorem that no efficient Bob lacking prior knowledge would be able to prove and a proof of that theorem. Our formalization of this will be in terms of computational depth [2], and we minimally wish that B outputs a theorem x followed by an output containing a proof of x such that any proof of x has t-time bounded depth at least $f(k, t) = \Theta(\log k + \log t)$ conditioned on x, so in this sense the theorem is "hard." We want a little bit more than this, though, so that B does not ignore Alice and try to generate theorems together with proofs on his own.

Formally, we take the class of Bobs $\mathcal{B} = \mathcal{P}$ and $G(A_\gamma \leftrightarrow B_{\sigma_0}(0^k; R), n) = 1$ if by the nth round, B outputs a theorem x followed by an output containing a proof of x such that there is some interactive algorithm $B' \in \mathcal{P}$ with encoding $\langle B' \rangle$ that outputs x in state ω when interacting with A_γ on randomness R, outputs a proof of x t steps later, for which x does not have proofs of t-time bounded depth $f(k, t)$ conditioned on x and $\langle B' \rangle$ (i.e., B' does not a priori "know" a proof), and does not have proofs of t-time bounded depth $f(k, t)$ conditioned on x, ω, and $\langle B' \rangle$ (so B' does not already have the proof stored in ω). Properly, we should also give B' Bob's "prior knowledge" as an auxillary input and condition on this as well, but we will suppress this in the discussion below. Conditioning can increase or decrease the depth of a string substantially – additional auxillary inputs can, respectively, either provide context that makes a string deep, or provide immediate access to a string that makes it shallow – so each of these properties is, in general, quite different.

A result of Antunes et al. [1] shows that, assuming pseudorandom generators against polynomial time with logarithmic seeds exist, if $B'_{\sigma_0}(0^k) \leftrightarrow \Phi$ outputs a theorem x and a proof at most t steps later, then the t-time bounded depth of the proof conditioned on x, $\langle B' \rangle$, and the state of B' when it outputs x is at most $O(\log t)$ and hence the goal is nontrivial for an appropriate choice of f.

To design V for G, we use an efficient probabilistic algorithm Π such that $\Pi(x; 0^t, 0^k|y)$ either outputs a proof of x or $\perp$, which we interpret as "I don't know." Antunes et al. also show that there exists a choice of Π that finds proofs of any theorem of depth $f(t,k) = O(\log k + \log t)$ with high probability. For this choice of Π, V accepts if the following holds: in $\mathsf{hist}(I_{\sigma_0}(0^k)|A_\gamma)$, I has some private output x, followed by a private output containing a proof of x at most t steps later, such that if ω is the state of I when it outputs x, then $\Pi(x; 0^t, 0^k|\langle I\rangle) = \perp$ and $\Pi(x; 0^t, 0^k|\omega, \langle I\rangle) = \perp$. Note that for our choice of Π, with high probability V (only) accepts theorems with proofs of depth $f(k,t)$, so $(V, \mathcal{P})$ is a $(G, \mathcal{P}, \mathcal{A}_0)$-syntactic goal. Moreover, whenever a $B' \in \mathcal{P}$ exists that produces a sufficiently deep theorem and proof with A, it is easy to see that V will accept $\mathsf{hist}(B'_{\sigma_0}(0^k)|A_\gamma)$ so $(V, \mathcal{P})$ is "well-designed" (for a slightly restricted class of Alices, or up to some constant factors in the depth required).

3.4 Tests

We have seen ways we can design syntactic goals (and hence universal protocols) for meta-goals that are purely intellectual or purely physical in our taxonomy; we now turn to some of the most interesting kinds of goals, that are both intellectual and physical, i.e., tests.

We recall the goal of the Turing test examiner, as considered in Example 5. There, we had two possible classes of Alices, $\mathcal{A}_1$ and $\mathcal{A}_2$, which corresponded to computers and humans, respectively, and the goal was that given a pair (A_1, A_2) where one $A_i \in \mathcal{A}_1$ and the other was from $\mathcal{A}_2$, to decide whether $A_1 \in \mathcal{A}_1$. So, the examiner wished to determine some property of the *Env*, specifically some property of an Alice he was conversing with. (Of course, the *computer* would be said to pass the Turing test if no such examiner could succeed at this goal better than chance; we only consider the examiner's goal here.) We won't attempt to formalize the Turing test further, and it isn't clear how this could ever be captured by a syntactic goal. Instead we will exhibit a test of *computational ability* (irrespective of whether or not this corresponds to "intelligence") which will illustrate several important issues in the design of such tests.

3.4.1 A test of computational ability

We fix the following parameters. Let $t(n)$ be the polynomial time bound we wish to test for, and let $k(n) = O(\log n)$ be a bound on the number of bits for the maximum size interactive Turing machines we wish to consider for Alice. Let $\mathcal{A}_1$ be the class of $t(n)$-time bounded machines. We wish to rule out the possibility that Alice is some $k(n)$-bit member of $\mathcal{A}_1$. (This description bound is critical, since it rules out the possibility that Alice has a look-up table for her responses "at length n;" incidentally, an essentially similar modification to the actual Turing test is stressed by Shieber [17] in response to essentially this issue as raised by Block [4].) Ideally, we would like to distinguish $\mathcal{A}_1$ from $\mathcal{A}_0 \setminus \mathcal{A}_1$, but this is clearly impossible, since some Alices in $A' \in \mathcal{A}_0$ are merely inefficient simulations of members of $A \in \mathcal{A}_1$, in the sense that A' may have the same behavior as A when interacting with Bob, but the internal state history of A' cannot be computed efficiently.

We show how to approach a still more restricted "promise" version of the goal: namely, fix a polynomial time bound $t'(n)$ and let $\mathcal{A}_2$ be the class of Alices A such that for any problem S that can be decided in time $t(n) + t'(n)$ on a four-tape Turing machine, for the goal G_S of computing S (as in Example 4) and class $\mathcal{I}_{t'}$ of $t'(n)$-time bounded interactive Turing machines, A is $(G_S, \mathcal{I}_{t'})$-helpful. That is, for every S decidable in time $t(n) + t'(n)$ on a four-tape Turing machine, A has

some interpreter I^S running in time $t'(n)$ that computes S when interacting with A from any state γ. We will also bound the description lengths of the interpreters we search over by some function $b(n) = O(\log n)$.

A meta-goal for testing ability: We will describe a test that works for the class $\mathcal{A}_1 \cup \mathcal{A}_2$. Precisely, the meta-goal $G_{1,2}$ here is that if an $A \in \mathcal{A}_1$ has no $k(\ell)$-bit descriptions, or an $A \in \mathcal{A}_2$ is not helpful to any $b(\ell)$-bit $I \in \mathcal{I}_{t'}$, then always $G_{1,2}(A_\gamma \leftrightarrow B_{\sigma_0}(0^\ell), n) = 1$ since we don't care what Bob does in this case. Otherwise, for $A \in \mathcal{A}_i$, $G_{1,2}(A_\gamma \leftrightarrow B_{\sigma_0}(0^\ell), n) = 1$ if the last output of B is "i."

Obviously, every $A \in \mathcal{A}_1 \cup \mathcal{A}_2$ is trivially $(G_{1,2}, \mathcal{P})$-helpful (to the appropriate Bob who prints "i"); the entire problem is in *selecting* the correct output, which, it is easy to see, would be performed by a $(G_{1,2}, \mathcal{P}, \mathcal{A}_1 \cup \mathcal{A}_2)$-universal protocol. Theorem 14 merely formalizes common sense here, when it tells us that we should attempt to construct a $(G_{1,2}, \mathcal{P}, \mathcal{A}_1 \cup \mathcal{A}_2)$-syntactic goal.

Theorem 29 *There is a $(G_{1,2}, \mathcal{P}, \mathcal{A}_1 \cup \mathcal{A}_2)$-universal protocol*

Proof Consider the predicate V that checks that on input 0^n, if the interpreter has a $b(n) = O(\log n)$ bit description and runs in time t', the jth private output of the interpreter is the negation of the jth output of the jth $O(b+k)$-bit four-tape Turing machine (the actual function of b and k will be clear later), restricted to run in $t(n) + t'$ steps. If this is true for every such machine and I outputs "2," V accepts. Note that there are a polynomial number of $O(b(n) + k(n)) = O(\log n)$-bit Turing machines. If, on the other hand, the interpreter simulates every $b(n)$-bit $t'(n)$-time bounded interpreter, none correctly compute this diagonal language, and I outputs "1," then V accepts. In all other cases, V rejects.

Observe that a four-tape $O(b+k(n))$-bit Turing machine can simulate the interaction between a b-bit t'-time bounded interpreter and a $k(n)$-bit $t(n)$-time bounded Alice in time $t(n)+t'$. Therefore, since we know that our interpreter has a b-bit description and runs in time t', if V accepts a "2," Alice must either need more than $k(n)$ bits or run for more than $t(n)$ steps. On the other hand, if V accepts a "1," we witnessed the failure of every $t'(n)$-time $b(n)$-bit interpreter, so either $A \notin \mathcal{A}_2$ or the interpreter needs more than $b(n)$ bits. Thus, V is a $(G_{1,2}, \mathcal{P}, \mathcal{A}_1 \cup \mathcal{A}_2)$-verification predicate.

Of course, some polynomial time interpreter does simulate every $t'(n)$-time bounded $b(n)$-bit Turing machine and witnesses its failure to find the diagonal language with $A \in \mathcal{A}_1$, so we will correctly identify any sufficiently small $A \in \mathcal{A}_1$. Furthermore, if there is a $b(n)$-bit interpreter running in time $t'(n)$ such that Alice helps this interpreter decide this diagonal set, then V will accept, so any $A \in \mathcal{A}_2$ will pass this test with some interpreter in $\mathcal{P}$ on every n. Therefore, we see that this is a well-designed $(G_{1,2}, \mathcal{P}, \mathcal{A}_1 \cup \mathcal{A}_2)$-syntactic goal, as needed. The result now follows immediately from Theorem 14. ■

Promises and verifiability: Our restriction of the class of Alices to $\mathcal{A}_1 \cup \mathcal{A}_2$ seems to play a substantial role in our success at desigining a protocol for this goal: Theorem 14 tells us that we shouldn't expect to give a protocol unless we can verify its correctness, and our earlier observations about the indistinguishability of certain members of $\mathcal{A}_0 \setminus \mathcal{A}_1$ from members of $\mathcal{A}_1$ would seem to dash our hopes. It is only by assuming an Alice who exhibits her powers – i.e., a member of $\mathcal{A}_2$, playing a role analogous to that of $\mathcal{A}_{G,\mathcal{B}}$ preivously – that we can achieve success.

Moreover, observe that our usual restrictions on the goals and classes of Alices that we employ in our limitation results fail to hold here: on the one hand, the goal we would ideally wish to achieve

– distinguishing $\mathcal{A}_1$ from the rest of $\mathcal{A}_0$ – is a prototypical example of one that is *not* simulation closed; on the other hand, the semantic closure of $\mathcal{A}_1$ in particular contains many Alices who fall neither in $\mathcal{A}_1$ (since, for strings outside their language, they might take time greater than $t(n)$) nor in $\mathcal{A}_2$ (since they may be unhelpful to $t'(n)$-time bounded interpreters), so $\mathcal{A}_1 \cup \mathcal{A}_2$ is not semantically closed. Thus, at least, the usual strong limitations (Theorem 18) do not apply here. Upon reflection, it's the latter change – the lack of semantic closure – that makes Bob's success possible; the lack of simulation-closedness of this goal is part of what makes Bob's task difficult!

More broadly, one could envision situations where restrictions on the class of environments Bob faces could permit him to succeed at goals that would otherwise be impossible—we could variously assume kinds of "commonality" (common knowledge, etc.) between Alice and Bob, and these would be reflected in restrictions on the class of environments (class of Alices) that the protocol is designed to work with.

4 Discussion

In this paper we asserted that any form of communication should be "goal-oriented." Under this assumption, we showed that it is possible for any task to be achieved by two players, without any common language or background, if and only if the goal is verifiable by the individual players. The insistence that communication ought to be goal-oriented seems to be totally defensible. Yet formulating explicit goals for natural scenarios is a non-trivial task and we provided a few examples on how goals may be formulated to model natural motivations for communication.

We can even go so far as to claim that we capture *all* significant semantics of communication under our definitions—that achieving one's own selfish goals *is* understanding. This claim that such goal-oriented communication is really "semantic communication" is probably controversial, but we mention that modern philosophers such as Quine [14] often take recourse to such a view of communication and thus implicitly (and independently) assert it, and more recently such a view of communication was explicitly asserted by Gauker [8] to solve some problems of reference. From a philosophical standpoint, our contribution is a rigorous, quantitative (algorithmic) modeling of semantic communication – phenomena that had previously been only qualitatively described – and an investigation of the consequences of this model, specifically what it says about the possibilities for and limitations on meaningful communication in the absence of a common background. We further remark that since the notion of a "goal" for communication certainly applies in essentially all practical contexts, (i.e., all contexts falling within the scope of engineering) one does not need to accept the thesis to appreciate our resuts.

Supposing we accept this thesis for now, does this mean we don't need (common) languages, and can simply proceed without them? We believe not, and suggest that language is a means to achieving efficiency in communication over the long term. This raises the question, "what is the resource whose usage language is trying to optimize?" This seems to be an interesting question to explore further. Some possible candidates are that language tries to reduce the number of bits exchanged by Alice and Bob or tries to minimize the number of rounds of communication. At the moment we are not sure which, if any, of these resources best models the use of language, or if language offers asymptotic improvements on the efficiency. But the idea that language can potentially be understood by the tools of computational complexity seems quite promising.

4.1 Open problems: restricting $\mathcal{A}$

In prior work [11], we showed a limitation result for universal protocols for the goal outlined in Section 3.1.1, which roughly asserted that an exponential dependence in the running time on the length of the interpreter we needed for communication was essentially necessary. Nothing about the proof of this theorem depended on the details of this goal, except that it was nontrivial (under the assumption that PSPACE $\neq$ BPP) and $\mathcal{I}$ was a well-behaved class (i.e., polynomial-time interactive Turing machines). We can generalize it as follows:

Theorem 30 *Let $\mathcal{I}$ be $\mathcal{P}$ or $\mathcal{L}$, and let G be a nontrivial, simulation closed goal. Let $\mathcal{A}$ be a semantically closed class of $(G,\mathcal{I})$-helpful Alices, and let $\mathcal{A}|_t \subset \mathcal{A}$ be the subclass that help some protocol running in time $O(t(n))$. Then a $(G,\mathcal{I},\mathcal{A})$-universal protocol B must run for a number of rounds that is exponential in the description length of the shortest protocol that is helped by Alice in time $O(t(n))$.*

The proof remains essentially the same: we construct an infinite family of Alices $\{A_\sigma\}_{\sigma\in\{0,1\}^*}$, each helping some interpreter of description length $|\sigma| + O(1)$. Since Alice is a black-box to Bob, a subexponential-round Bob does not have enough messages to try every σ, and must achieve the goal without Alice's help. (The full proof appears in Appendix B.)

With the exception of the goal of testing computational ability (in Section 3.4, which succeeds when it identifies $\Phi \in \mathcal{A}_1$), all of the goals we considered were nontrivial in our sense. The point here is that enumeration is optimal for these natural classes of interpreters when we require that $\mathcal{A}$ is semantically closed, and this effect is widespread.

Since this exponential constant factor in the running time of a universal protocol is extremely undesirable, we need to explore means of restricting $\mathcal{A}$ so that it is *not* semantically closed but is still broad enough to yield useful protocols. This result suggests that mere restrictions on computational resources cannot suffice for these purposes; intuitively, we need some definition that rules out this degenerative "hiding" behavior of the A_σ (and allows an appropriate interpreter to be efficiently found). Alternatively, we might hope to find a measure of the "degeneracy" of Alice and we might then hope to find a protocol for which the efficiency scales appropriately with this quantity.

Acknowledgements

We would like to thank Oded Goldreich and Ryan Williams for their input on earlier revisions of this work, Or Meir for discussing alternatives to some definitions with us, and Guy Rothblum for helpful discussions on space-bounded interactive proofs.

References

[1] Luis Antunes, Lance Fortnow, Alexandre Pinto, and Andre Souto. Low depth witnesses are easy to find. In *Proc. 22nd Conference on Computational Complexity*, 2007.

[2] Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Computational depth: concept and applications. *Theor. Comput. Sci.*, 354(3):391–404, 2006.

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

[4] Ned Block. Psychologism and behaviorism. *Philosophical Review*, 90(1):5–43, 1981.

[5] Anne Condon. Space-bounded probabilistic game automata. *J. ACM*, 38(2):472–494, 1991.

[6] Anne Condon and Richard Ladner. Probabilistic game automata. *JCSS*, 36(3):452–489, 1988.

[7] Nikhil R. Devanur and Lance Fortnow. A computational theory of awareness and decision making. Technical Report TR08-046, ECCC, 2008.

[8] Christopher Gauker. *Words Without Meaning*. MIT Press, Cambridge, 2003.

[9] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proc. 40th STOC*, 2008.

[10] Marcus Hutter. *Universal Artificial Intelligence*. Springer, Berlin, 2004.

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

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

[13] 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.

[14] Willard Van Orman Quine. *Word and Object*. MIT Press, Cambridge, 1960.

[15] Stuart Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Prentice Hall, New Jersey, 1995.

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

[17] Stuart M. Shieber. The Turing test as interactive proof. *Noûs*, 41(4):686–713, December 2007.

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

A Technical definitions

In this section, we will define the technical notions underlying the definitions in Section 2, which will be necessary to give formal proofs of our main theorems in Appendix B. In particular, we will describe a variety of properties of classes of algorithms that we will require. Unfortunately, we are not aware of any existing definitions that would allow us to state our main theorem in the generality we desire, so although the properties we demand will be familiar, the definitions are not standard.

Naturally, in this section we will consider interactive Turing machines as our models of algorithms computing the functions defining our agents. Sometimes we will say that an interactive Turing machine B "sets a bit." We mean that if B has state set Q, there is some subset of these states, $R \subset Q$ where the bit is considered to be 1, and for any $q \notin R$, the bit is considered to be 0. So, for example, "setting the bit to 1" means that B enters some state in R as it continues with its computation.

Definition 31 (Simulation) *We will say that* B' simulates B *in a collection of computation histories if there is a surjective mapping* σ *from states of* B' *to states of* B *and a projection* π *mapping the tape configuration of* B' *to a tape configuration of* B *such that for all histories in the collection,* B' *and* B *produce the same outputs; the mapping* ψ *from configurations of* B' *to configurations of* B *obtained from* σ *and* π *is surjective on the set of configurations of* B *occurring in the history; and if for configurations* c_i *and* c_{i+1} *of* B' $c_i \to c_{i+1}$ *(on a coin toss* b*), then it is either the case that* $\psi(c_i) = \psi(c_{i+1})$ *or* $\psi(c_i) \to \psi(c_{i+1})$ *(on a coin toss* b*).*

Typical examples of such simulations are the standard "clocked" simulations. Notice that if B' simulates B, B' takes at least as many steps on each history in the collection, uses at least as much space, and uses at least as many coin tosses. Note also that if B sets a bit to 1 in some states R, and B' simulates B, we can say B' also sets this bit to 1 in states $\sigma^{-1}(R)$.

Definition 32 (Efficiently enumerable class) *We say that a class of interactive algorithms* $\mathcal{I}$ *is* $\mathcal{B}$-efficiently enumerable *(or, if* $\mathcal{I} = \mathcal{B}$*, simply* efficiently enumerable*) if there is some* efficient enumerator U *such that, for each* $I \in \mathcal{I}$ *there is some machine* $U_I \in \mathcal{B}$ *that for all* x *and all response histories,* $U_I(x)$ *simulates some prefix of* $U(x)$ *and* $U_I(x)$ *simulates* $I(x)$ *in some suffix.*

Definition 33 (Comparable class) *We say that a class of efficiently enumerable interactive algorithms* $\mathcal{I}$ *is* comparable *if, for any interactive algorithm* I^1 *which, for every environment, is simulated by some* $I^2 \in \mathcal{I}$ *then there is some efficient enumerator* U' *such that whenever* U' *simulates a machine* $I \in \mathcal{I}$ *that halts,* U' *sets a bit (initially 0) indicating whether* I *and* I^1 *would have produced the same outputs if* I^1 *had been given the same coins and inputs on each round.*

Definition 34 (Logspace-closed class) *We say that a class of agents* $\mathcal{I}$ *is* logspace-closed *if given any pair of logspace functions* f *and* g *and an agent* $I \in \mathcal{I}$*, there is an agent* $I' \in \mathcal{I}$ *that simulates* I *with* f *applied to each of its inputs from Env and* g *applied to each of its outputs to Env.*

Definition 35 (Monotone class) *We say that a class of agents* $\mathcal{I}$ *is* monotone *if, for any* $M \in \mathcal{I}$*, if* M' *is constructed by replacing some states of* M *with halting states, then* $M' \in \mathcal{I}$ *as well.*

Definition 36 (Standard class) *We say that a class of agents* $\mathcal{I} \supseteq \mathcal{L}$ *is* standard *if it is efficiently enumerable, comparable, logspace-closed, monotone, and all members halt with probability* $1 - \mathrm{negl}(|x|)$ *on every private input* x*.*

All of the classes we consider are defined in terms of the asymptotic resource usage of the machines involved, and so trivially will satisfy these properties. The nontrivial part – the efficient enumerability of polynomial time – is originally due to Levin [12].

Lemma 37 *The class of polynomial-time interactive Turing machines is standard.*

Lemma 38 *For space constructible* $s(n) = \Omega(\log n)$*, the class of space-*$s(n)$ *and polynomial time bounded interactive Turing machines is standard.*

In addition, when we vary the class $\mathcal{I}$, we will need to further restrict the verification predicate to satisfy the following definition:

Definition 39 (Verifiable in $\mathcal{B}$) *We say that a verification predicate $(V,\mathcal{I})$ is* verifiable in $\mathcal{B}$ *if there is a $\mathcal{B}$-efficient enumerator U_V for $\mathcal{I}$ with a special "accept" bit such that for each $I \in \mathcal{I}$ and each x, $V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma)) = 1$ iff $U_V(x)$ sets the accept bit to 1 after simulating I on private input x and interacting with A with probability at least $1 - \mathrm{negl}(n)$.*

B Proofs of Theorems

In this section, we give the proofs our our main theorems (stated in Section 2.2) and the limitation result stated in Section 4.1. The proofs depend crucially on the technical notions introduced in Appendix A, in addition to the various definitions of Section 2.

Proof (of Theorem 14)

($\Rightarrow$): Suppose that there is a universal protocol B for the goal with respect to $\mathcal{A}$ and $\mathcal{I}$. Let V test if I simulates B (using the same coin tosses) until it halts in $I_{\sigma_0}(x) \leftrightarrow A_\gamma$, and note that when B is simulated by $B^A \in \mathcal{I}$, B halts. Moreover, since B^A simulates B with probability at least $1 - \mathrm{negl}(n)$ and $B_{\sigma_0}(x) \leftrightarrow A_\gamma$ achieves G on x with probability at least $1 - \mathrm{negl}(n)$, if I produces the same outputs as B until it halts, then I achieves the goal with probability at least $1 - \mathrm{negl}(n)$, and we have that V is a verification predicate for G with respect to $\mathcal{I}$ and $\mathcal{A}$.

Since $\mathcal{I}$ is efficiently enumerable we know there is an efficient enumerator U, and since it is comparable, we know that there is a machine U_V that simulates U and B and compares the outputs of each I and B, so V is verifiable in $\mathcal{I}$, and hence $(V,\mathcal{I})$ is a $(G,\mathcal{I},\mathcal{A})$-syntactic goal of communication.

It only remains to show that every $A \in \mathcal{A}$ is $(V,\mathcal{I})$-helpful. This is so, since for every $A \in \mathcal{A}$ we have the $B^A \in \mathcal{I}$ that $\forall x$ and states of Alice γ, B simulates to completion with probability at least $1 - \mathrm{negl}(n)$, for which V then clearly accepts $(\mathsf{hist}(B^A_{\sigma_0}(x)|A_\gamma))$.

($\Leftarrow$): Suppose that there is a $(G,\mathcal{I},\mathcal{A})$-syntactic goal of communication $(V,\mathcal{I})$ such that every $A \in \mathcal{A}$ is $(V,\mathcal{I})$-helpful.

Since this means that V is verifiable in $\mathcal{I}$, there is an efficient enumerator U_V for $\mathcal{I}$ that maintains an "accept" bit that is set after simulating $I \in \mathcal{I}$ on x with probability $1-\mathrm{negl}(|x|)$ iff V would accept $\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma)$. Let B be the following modification of U_V: whenever U_V would set the accept bit, B halts.

Let any $A \in \mathcal{A}$ be given. Since every $A \in \mathcal{A}$ is $(V,\mathcal{I})$-helpful, for any such A we know that there exists some $I^A \in \mathcal{I}$ such that for all $x \in \{0,1\}^n$ and states of Alice γ, with probability at least $1-\mathrm{negl}(n)$, $V(\mathsf{hist}(I^A_{\sigma_0}(x)|A_\gamma)) = 1$. Since U_V is an efficient enumeration of $\mathcal{I}$, there is some $U_A \in \mathcal{I}$ such that $\forall x$ $U_A(x)$ simulates some prefix of $U_V(x)$ and $U_A(x)$ simulates $I^A(x)$ in some suffix. Since U_A simulates U_V, it also has an accept bit. Let B^A be the following modification of U_A: whenever U_A would set the accept bit, B^A halts, and note that since $\mathcal{I}$ is monotone, $B^A \in \mathcal{I}$ as well.

We now show that B and B^A are as required for a universal protocol:

– Since V is a $(G, \mathcal{I}, \mathcal{A})$-verification predicate, whenever V accepts $\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma)$, I has achieved G on x with A with probability $1 - \text{negl}(n)$; since $B^A(x)$ simulates some $I(x)$ such that V accepts $\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma))$ with probability at least $1 - \text{negl}(n)$, they produced the same outputs, and hence B^A achieves G on x with A with probability at least $1 - \text{negl}(n)$.

– Note that B^A has states and configurations identical to U_A (which simulates U_V) until U_A would set the accept bit, and B has states and configurations identical to U_V, again until U_A would set the accept bit, and when this happens, B also halts. So, in any history where U_A sets the accept bit, we can use the mappings from U_A to U_V to find that B^A simulates B until it halts. Since V accepts $\mathsf{hist}(I^A(x)|A_\gamma)$ with probability at least $1 - \text{negl}(n)$, and U_A simulates I^A in some suffix of its computation, U_A sets the accept bit with probability at least $1 - \text{negl}(n)$, as needed.

■

Proof (of Theorem 18) Let a semantically closed and nonempty $\mathcal{A}$ be given, and suppose that $(V, \mathcal{I})$ is not a $(G, \mathcal{B}, \mathcal{A}_0)$-syntactic goal. Then, V must not be a $(G, \mathcal{I}, \mathcal{A}_0)$-verification predicate, and we find that for any negligible failiure probability $f(n)$, there is some $A \in \mathcal{A}_0$ and $I \in \mathcal{I}$ such that there exists a private input x and a finite set of coin tosses occurring with probability greater than $f(n)$ for which I halts at step n' when interacting with A, $V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma) = 1$ and $G(A_\gamma \leftrightarrow I_{\sigma_0}(x), n') \neq 1$. Let ℓ be the length of the longest message sent by $I_{\sigma_0}(x)$ on these coins when interacting with A.

We now convert such an Alice A into $A'_x \in \mathcal{A}$: let $\tilde{A}$ be any Alice in $\mathcal{A}$. If A'_x answers messages of the form $0^{\ell+1} \circ y$ as $\tilde{A}$ answers y padded with $0^{\ell+1}$, and computes the same function as A does on prefixes of the finite set of histories where I fails, then A'_x is contained in the semantic closure of $\tilde{A}$ since this finite function can be computed in $\mathcal{L}$ and likewise the padding can be performed in logspace, and hence $A'_x \in \mathcal{A}$ since it is semantically closed. Yet, since G is simulation closed, A'_x simulates A, and $G(A_\gamma \leftrightarrow I_{\sigma_0}(x), n') \neq 1$ with probability greater than $f(n)$, we also find that with probability greater than $f(n)$, $G((A'_x)_{(\gamma,\tilde{\gamma})} \leftrightarrow I_{\sigma_0}(x), n') \neq 1$ but

$$V(\mathsf{hist}(I_{\sigma_0}(x)|(A'_x)_{(\gamma,\tilde{\gamma})}) = V(\mathsf{hist}(I_{\sigma_0}(x)|A_\gamma) = 1$$

so V was not a $(G, \mathcal{I}, \mathcal{A})$-verification predicate with any such correctness probability $1 - f(n)$, and we find that $(V, \mathcal{I})$ could not have been a $(G, \mathcal{B}, \mathcal{A})$-syntactic goal. ■

Proof (of Theorem 30) Let any $A \in \mathcal{A}|_t$ be given. We start by constructing a family of $(G, \mathcal{I})$-helpful Alices $\{A_\sigma\}_\sigma$ for every $\sigma \in \{0,1\}^*$, where A_σ behaves as follows: for A_ϕ which always responds with the empty string and $f_\sigma(y) = \sigma \circ y$, $A_\sigma = A^{f_\sigma}_{A_\phi}$.

Clearly, for the Bob B^* that is helped by A and runs in time $O(t(|x|))$, the Bob B^*_σ who converts each query y to $\sigma \circ y$, ignores the prefix σ of Alice's reponses, and otherwise computes the same function as B^* has description length $|\sigma| + O(1)$, runs in time $O(t(|x|))$, and has the same space requirements. Furthermore, every A_σ is in the semantic closure of A, and thus is clearly also in $\mathcal{A}|_t$.

Now suppose there is a $(G, \mathcal{I}, \mathcal{A})$-universal Bob B whose probabilistic running time when interacting with any A_σ on input x is $O(|x|^k)$, and runs in a number of rounds that is subexponential in $|\sigma|$;

since Bob runs for at least one step on each round, this is at most $2^{o(|\sigma|)}|x|^k$ rounds in total. Notice that if B runs for less than $2^{2k \lg |x|}$ rounds, then there is some σ of length $2k \lg |x|$ such that A_σ is consistent with this set of responses, since Bob sends at most one such σ on each round. Since B runs in $O(|x|^{1.5k})$ rounds for all such A_σ, for all sufficiently large x B must halt without receiving a response from Alice other than the empty string. Since all $A_\sigma \in \mathcal{A}$ help Bob, Φ is simulated by some A_σ in this case, and G is simulation closed, Φ helps Bob as well contradicting our assumption that G is nontrivial. ■

Proof (of Theorem 28)

($\Leftarrow$): Given a $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$-universal protocol B, let P^Π respond to any message x with $\Pi(x)$, and let V on input x run B on private input x, and accept iff B would halt with 1 as the final private output of B. We claim that this is a logspace competitive interactive proof system.

Clearly P can be implemented in deterministic logspace (given oracle access to Π) and P^Π is $(G_\Pi, \mathcal{L})$-helpful. In turn, this guarantees that the interaction between B and P^Π can be simulated in probabilistic logspace and polynomial-time, so V can be implemented in probabilistic logspace and polynomial-time. It is easy to see that V is complete: if $x \in \Pi$, then since B achieves G_Π with high probability, B will output 1 before it halts with probability greater than 2/3, and hence V will accept with this probability. To see that V is sound, notice that in transcripts where B halts, V would have simulated B faithfully, and hence we would have satisfied the verification predicate constructed in Theorem 14. Since this goal is simulation closed and $\mathcal{A}_{G_\Pi,\mathcal{L}}$ is semantically closed and nonempty, Theorem 18 says that this verification predicate is valid for $\mathcal{A}_0$; in particular, no matter what prover $\tilde{P}$ B would converse with, B can only output 1 and halt when $x \notin \Pi$ with negligible probability. Thus, when $x \notin \Pi$, for every $\tilde{P}$, V accepts with probability less than 1/3 for sufficiently large inputs, which is sufficient.

The proof system for $\bar{\Pi}$ is similar: V accepts iff B would output 0 and halt.

($\Rightarrow$): Suppose that our logspace competitive interactive proof systems are given by (P_Π, V_Π) and $(P_{\bar{\Pi}}, V_{\bar{\Pi}})$. Our verification predicate is then as follows: V checks that I repeatedly simulates n runs of V_Π followed by n runs of $V_{\bar{\Pi}}$, that I produces an output after its final simulation, and that if the final private output of I is a 1, then V_Π accepted in the majority of the n preceding simulations, or else if the final private output of I is a 0, then $V_{\bar{\Pi}}$ accepted in the majority of the n preceding simulations. By the assumed soundness of V_Π and $V_{\bar{\Pi}}$, no matter what $A \in \mathcal{A}_0$ causes I to output, the probability that a majority of the runs accept when $x \notin \Pi$ or $x \in \Pi$, respectively, is exponentially small, so an interpreter can only satisfy V when it outputs $\Pi(x)$ except with negligible probability, and hence V is a verification predicate for G_Π that is sound against $\mathcal{A}_0$.

Since V_Π and $V_{\bar{\Pi}}$ can be implemented in probabilistic polynomial time and logspace, these checks can also be performed in parallel in polynomial time and logspace (given the coin tosses of I). Therefore, $(V, \mathcal{L})$ is a $(G_\Pi, \mathcal{L}, \mathcal{A}_0)$-syntactic goal.

To see that $(V, \mathcal{L})$ is a well-designed $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi,\mathcal{L}})$-syntactic goal, observe that whenever $A \in \mathcal{A}_{G_\Pi,\mathcal{L}}$, there is a $B_A \in \mathcal{L}$ such that $B_A(x)$ outputs $\Pi(x)$ with high probability when interacting with A; wlog, we can assume that this probability is $1 - \text{negl}(n)$. We also know that P_Π and $P_{\bar{\Pi}}$ run in logspace and polynomial time, so there is an interpreter $I_\Pi \in \mathcal{L}$ that

simulates P_Π and $P_{\bar{\Pi}}$, using B_A as a subroutine to simulate an oracle for Π. Since on any query x', the response from B_A agrees with $\Pi(x')$ with probability $1 - \text{negl}(n)$, I_Π simulates P_Π^Π and $P_{\bar{\Pi}}^\Pi$ correctly with probability $1 - \text{negl}(n)$ on each interaction, and hence satisfies V with probability $1 - \text{negl}(n)$, so A is also $(V, \mathcal{L})$-helpful, and so $(V, \mathcal{L})$ is well-designed, as claimed. Theorem 14 therefore shows that a $(G_\Pi, \mathcal{L}, \mathcal{A}_{G_\Pi, \mathcal{L}})$-universal protocol exists.

■

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/