A Theory of Goal-Oriented Communication*

Oded Goldreich Brendan Juba Madhu Sudan

September 17, 2009

Abstract

We put forward a general theory of *goal-oriented communication*, where communication is not an end in itself, but rather a means to achieving some *goals* of the communicating parties. The goals can vary from setting to setting, and we provide a general framework for describing any such goal. In this context, "reliable communication" means overcoming the (potential) initial misunderstanding between parties towards achieving a given goal.

We identify a main concept, which we call sensing, that captures the party's ability to check whether progress is made towards achieving the goal. We then show that if sensing is available, then the gap between a priori mutual understanding and lack of it can be bridged. For example, if providing the parties with an adequate interpreter allows them each to achieve their (possibly different) goals, then they can achieve their goals also without such an interpreter (although they may misunderstand each other and err at the beginning). Or, if each server (in a predetermined class of servers) can help some user (who understands the server) achieve its goal, then there exists a user strategy that achieves the goal no matter with which server it communicates.

*An early version of this work has appeared as an ECCC report [10].

Contents

1 Introduction

The traditional perception of the "problem of communication" in EE and CS is dominated by Shannon's highly influential work [15], which focuses on communicating data over a noisy channel and sets aside the meaning of this data. Shannon's work assumes that the communicating parties are a priori in agreement on the communications protocol they are about to employ. Thus, the question of what the meaning of the intended communication is, is deemed irrelevant and so entirely dismissed.[1]

However, the meaning of information does start to become relevant (even to the engineering problem of designing communication systems), whenever there is diversity in the communicating parties, or when the parties are themselves evolving over time. Take, for example, the mundane "printing problem:" Here a *computer* attempts to communicate with a *printer*, to print some image, but does not know the format used by the printer (aka the "printer driver"). As a second example consider the "computational (delegation) problem:" Here a weak computer (a laptop) would like to outsource some computational task to a powerful supercomputer, but does not know the language in which computational tasks may be described to the supercomputer. In both examples, the bottlenecks today (empirically) seem to be not in the reliability of the communication channel, but rather misunderstanding at the endopints.

This leads us to consider a problem "complementary" to Shannon's. We consider communication in the setting where parties do not, a priori, agree on a communications protocol (i.e., on a language). Indeed, these parties may not a priori *understand* each other, and so the question of meaning arises; that is, what does each of the parties want? what does each expect? what does each hope to achieve? and can they cooperate in a way that benefits each of them?

The foregoing questions are rooted in the thesis that communication has a goal[2] (or that each of the communicating parties has a goal that it wishes to achieve). That is, following a dominant approach in twentieth century philosophy (see Section 1.5), we associate the meaning of communication with the goal achieved by it.

1.1 Our work at a glance

Our main contribution is in suggesting a mathematical theory of goal-oriented communication. Starting with a formal definition of communicating parties as mathematical (computational) objects, we give a general definition of goals of communication. In particular we give a single schema by which all goals of communication can be described (including the "printing" goal and the "computational" goal above). In particular, our schema allows us to capture communicating parties, each of which may wish to achieve a goal, but may not understand each other (i.e., there is no a priori agreement on a protocol). Goals in our setting are "perpetual." One such goal, for example, is to print an infinite sequence of images on a printer; we would be happy if our computer manages to print all but a finite number correctly.

We identify a main concept, called sensing, that captures the party's ability to check whether progress is made towards achieving the goal (i.e., whether it is "on the right track"). We then show that if sensing is available, then the gap between a priori mutual understanding and lack of it can be bridged. For example, if providing the parties with an adequate interpreter allows them

[1]Specifically, Shannon [15] asserts "Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem."

[2]In the printing example above, the goal of the computer is to ensure the image is printed properly; while in the computational example, the goal of the weak computer is to ensure that the computational task is executed correctly.

each to achieve their (possibly different) goals, then they can achieve their goals also without such an interpreter, although they may misunderstand each other and err finitely many times at the beginning.

We stress the generality of the foregoing statements (which go far beyond the "computational goals" considered by Juba and Sudan [9]).[3] Indeed, our definition of goal also includes goals that can be achieved without any communication, but our actual focus is on goals that do require communication. We find it instructive to consider an asymmetric situation in which the two communicating parties are a *user* and a *server*. (In the printing example, the user is the computer that wishes to print, and the server is the printer.) The user has some goal but needs the server's help in order to achieve it. The server is willing to help and it would indeed be helpful for the "right" user, but the actual user does not know the strategy used by the "right" user. (Again in the printing example, the printer does print images correctly for the computers with the right driver, but "our" computer does not know this driver.) Put in other words, the server is one of several possible helpful servers (i.e., each such server is helpful to some user), but the user does not know which server it is communicating with. Our main result says that with the help of sensing, the user may achieve its goal nevertheless.

1.2 Focus on the user–server setting

On top of the intellectual interest in definitions and results regarding general concepts such as the "meaning of communication" and goal-oriented collaborations, we wish to highlight the potential relevance of the theory of goal-oriented communication to computer practice. In a variety of settings, users are trying to obtain various services from designated servers (e.g., printers, web-servers, etc), while not necessarily having full understanding of the functionality of the servers. The users may be humans or computers, and the same applies to the servers. Either way, in many cases, the user does not know the server well, and consequently each party may send messages that the other party does not understand or, possibly worse, misunderstands.

Note that in these situations the user has some goal, whereas the server is just put there to help users. Still the issue at hand is that the server may not understand all (or most) users, and the individual users don't necessarily know how to communicate with the server. A theory that tells us under what conditions this initial lack of understanding can be overcome, how, and at what expense, is thus of interest.

For example, one of our results asserts that if the user can sense whether the server is making progress towards the desired goal, then it is possible to achieve the goal when communicating with an arbitrary (unknown to the user!) server as long as this server is helpful at all (i.e., may assist some other user). Thus, the fact that a server is helpful to somebody implies that it can also help us (although we may not know a priori which server we are trying to use). This is achieved by using a "universal strategy" (i.e., one that works whenever some other strategy works). For example, if a printer can be used by some machine, which uses a specific format, then we can use it too (although we may not know a priori which format the printer expects).

Essentially, our solution, which is based on sensing, is "try and check" (i.e., try all possible formats and rely on the ability to check whether the printer responses at all and what it prints). Thus, we confirm a very natural and often used human paradigm.

[3] Indeed, the work of Juba and Sudan [9] was motivated by such general questions and provided the first step in that direction (as well as many of the ideas we use in the current work). However, the restriction to computational goals and the use of PSPACE-complete server strategies led to some skepticism among other researchers, on the possibility of providing a theory of general goals along their lines. We hope and believe that this skepticism will be resolved by the current work.

Of course, we do not recommend implementing the solutions we provide. These are to be thought of as feasibility results and as first steps in a study of conditions under which initial lack of understanding can be overcome, and where the "expense" of overcoming the lack of understanding can even be quantified.

1.3 Confirming various practices and other perspectives

Our results confirm a number of common beliefs and practices. One example, which was already mentioned above, is the practice of acting before reaching full and/or certain understanding of the situation. Indeed, if we are to wait without actions until achieving certainty, then we will never achieve any progress. This common wisdom is reflected in the design of the universal strategy that just acts based on its current hypothesis and changes the hypothesis once it sees evidence against it. Such false hypotheses cause only a bounded amount of damage, whereas waiting for certainty would have resulted in waiting forever and making no progress at all. Indeed, humans act as universal users, and computers may be designed to do the same.

The aforementioned universal users rely on sensing (i.e., the ability to sense progress towards our goals). Indeed, this work confirms the benefit in being able to obtain feedback on the effect of our actions. In particular, intelligible feedback from the environment about the progress we are making towards achieving our goal gives rise to a trivial sensing process, which in turn yields a universal user strategy.

This work also offers a perspective on the notion of language translation, which may be viewed as a syntactic way of overcoming misunderstandings. Specifically, we view languages as arbitrary sets of strings and translators (or interpreters) as efficiently computable and invertible functions mapping one language to another. Now, consider such a function f, a user strategy U and a server S, and let U_f be the strategy that applies f to each outgoing message determined by U and applies f^{-1} to each incoming message (before feeding it to U). Then, if U_f achieves the goal G when communicating with the server S, then we may say that f is a good translator (or interpreter) for U with respect to interacting with S towards achieving the goal G.

1.4 Relation to prior work of Juba and Sudan [9, 10]

This work greatly extends the scope of the research direction initiated by Juba and Sudan [9, 10]. Specifically, the study in [9] is restricted to ("one shot") computational goals, whereas we consider arbitrary goals and extend the results of [9] to this general case. Furthermore, their main results refers to servers that employ PSPACE-complete strategies, whereas our results apply also to polynomial-time implementable servers.

We proclaim [10] to be an early version of the current work.[4] We mention that the current exposition introduces a natural formalization that is both more expressive and more transparent than the one used in [10], and in a future version of this work, we will show how the present formalization can be adapted to more cleanly capture all the problems considered in [10].

The exposition in [10] is restricted to one-shot goals (a.k.a "finite" goals) and thus associates termination with achieving the goal. In contrast, we consider infinite goals (including multi-session goals) and decouple achieving the goal from the simultaneous awareness of the user of progress on its goal. This allows the user to take risks – that is, make assumptions regarding the world (including the server) that may prove wrong (and be corrected in the future) – and benefit in the

[4]We stress that the only "publication" of [10] is as an ECCC report.

meanwhile rather than waiting to a point, which may never occur, in which it may act without risk.

1.5 Relation to work in other fields

Given the general scope of our investigation of goal-oriented communication, it is hardly surprising that similar questions were studied in other fields. Before reviewing some of these studies, we note that the surprising fact is that these types of questions were not studied before (i.e., before [9]) in computer science. We attribute this phenomenon to the immense influence of Shannon's prior study of communication [15].

As stated above, the semantics (or meaning) of communication was irrelevant to the problem Shannon studied. Indeed at the time, ignoring the problem of semantics was perhaps the most appropriate choice, given the much larger problem of (lack of) reliability of the communication channel. In the subsequent years, the resulting research has addressed this problem adequately enough that the "lesser" problem of semantics of information now seems deserving of study and forms the core of the problem that we address.

Shannon's theory assumes the compatibility of the communicating parties (i.e., full, a priori, mutual understanding) and studied their ability to overcome external interference, whereas our study aims at overcoming the possible incompatibility of the communicating parties. Thus, we are led to consider a model of communication that relates to the *meaning of communication*, where we associate meaning with achieving goals. Similar approaches to modeling communication have been considered in Philosophy (see next).

Related approaches in Philosophy. In the 1920s, various philosophers independently concluded that communication should be regarded as a means to an end, and that the "meaning" of the communication should be taken as no more and no less than the ends achieved via communication. For example, Dewey stated such a view in 1925 [5], and in a later revision of his work observed that a similar view had been independently expressed some years earlier in an essay by Malinowski [12]. Subsequently, such views were adopted by Wittgenstein, and played a central role in some of the most influential work in philosophy of the twentieth century [16, 17].[5]

In these works, Wittgenstein introduced his views of communication by means of "language games," scenarios of limited scope in which the utterances serve some definite purpose. For example, one language game he considered featured a primitive language used by a builder and his assistant, where the builder calls out, e.g., "brick!" or "slab!" and the assistant brings the corresponding object. Our model of goal-oriented communication resemble these language games. We note, however, that Wittgenstein's reference to these language games is mainly descriptive and illustrative (primarily by examples).[6]

[5]By and large, work in Linguistics or Semiotics, though influenced by these views, has too narrow a scope to be of relevance to us. Specifically, these fields assume communication that is structured (as terms in a grammar) and a human conceptual scheme, whereas we make no prior assumptions about the syntax of the communication nor about the conceptual schemes employed by the communicating parties. In other words, our focus is on the effect of communication rather than on the languages or symbols used.

[6]Indeed, Wittgenstein did not provide a generic abstract formalization of language games – for his purposes, it was enough to only consider simple examples. The closest he comes to giving definitions of language games is on p. 81 of [16] and in Remarks 2–7 of [17] (cf. also Remarks 23 and 130): He defines a language game as a complete system of (human) communication, i.e., one that could be taken as a primitive language. Remark 23 lists examples of activities where language is used, and asserts that there is a language game corresponding to each of these activities (and this makes it clear that each goal of communication we consider corresponds to a language game); Remark 130 clarifies that his purpose in considering language games is to obtain idealized models of language usage, somewhat

Our contribution is thus in providing a clear and rigorous definition of goal-oriented communication. Furthermore, this definition is suitable as a basis for study of various qualitative and quantitative questions that arise naturally. Indeed, using this formalism, one may ask when and to what extent meaningful (i.e., goal-oriented) communication is possible. Moreover, our formalism incorporates the computational aspects of communication, which both permits us to formulate computational goals for communication and moreover permits us to consider the computational feasibility of various schemes for communication.

Related work in AI. It is not surprising that a model of goal-oriented, computationally limited agents has also been considered in AI. In particular, the Asymptotic Bounded Optimal Agents, introduced by Russell and Subramanian [13], bear some similarity to the universal communicators we consider here. The similarity to our work is merely in the attempt to capture the notion of a goal and in the related definitions of "optimal" achievement of goals, while the crucial difference is that they only consider a single player: in their work the goal is achieved by a user (called an agent) that acts directly on the environment and obtains no help from a server (with whom it may need to communicate, while establishing an adequate level of mutual understanding) and so no issues analogous to incompatibilities with the server ever arise. Indeed, the question of the meaning of communication (i.e., understanding and misunderstanding) does not arise in their studies.[7]

Our universal communicators are also similar to the universal agents considered by Hutter [8]. Like Russell and Subramanian, Hutter considers a single agent that interacts with the environment, and so there is no parallel to our interest in the communication with the server. In addition, Hutter's results are obtained in a control-theoretic, reinforcement learning setting, that is, a model in which the environment is assumed to provide the value of the agent's actions explicitly as feedback. Although we sometimes consider such settings, in general we assume that the user needs to decide for itself whether or not communication is successful.

2 Overview

In this section we provide a high-level overview of the main contents of our work. We try to avoid any technical details, and hope that the result will be sufficiently clear without them. We warn, however, that the actual treatment has to deal with various technical difficulties and subtleties, which we pushed under the carpet in the current section.

2.1 The general framework: notions and results

In this section we overview of the general conceptual framework of our work and the type of results we obtain. The corresponding detailed technical treatment can be found in Sections 3 and 4.

The basic setting. In the basic setting, we model goal-oriented communication between *two entities*, by studying *three* entities. We describe these entities below, but first we note that for our

akin to "ignoring friction" in physics problems.

[7] Indeed, the "task-environments" of Russell and Subramanian are related to our notion of goals, though they use real-valued utilities instead of our Boolean-valued predicates reflecting success. But the crucial difference is that while Russell and Subramanian consider a goal-interested agent interacting with an environment, these interactions are actually actions, and communication per se (let alone its meaning) is not a major concern. By contrast, in our model there are two entities, a user and a server, and we typically consider goals where (intelligible) communication between these entities is essential for achieving the goal. Even in the context of modeling goals for a solitary agent, there are significant differences in the formalism, but these are minor in comparison to the above.

purpose an entity is mathematically a (possibly randomized, or non-deterministic) function from the current state and current input signals (coming from other entities) to a new state and new output signals. The state as well as the signals are defined to come from a discrete, but possibly countably infinite set.

Our starting entity is ourselves, that is, *users*. Users wish to affect the environment in a certain way or obtain something from the environment, making this *environment* our second entity. To achieve the desired effect on (or information from) the environment, we may need help from somebody else, called a *server*. Thus, the definition of a goal involves three entities: a user, a server, and the environment (or the world).

The communication between the user (resp., server) and the environment reflects actions or feedback that the user (resp., server) can directly perform in the environment or obtain from it. The difference between the action and feedback capacities of the user and server with respect to the environment is the reason that the user may want to get help from the server, and towards this end these two parties must communicate (and understand one another). Indeed, *the communication between the user and the server is the focus of our study.* This communication carries (symbolic) text that reflects control commands and/or information, and the purpose of this communication is to coordinate some effect and/or obtain some information on/from the environment. Since the user–server communication is symbolic in nature, the question of its meaning and intelligibility (w.r.t the goal at hand) arises.

Jumping ahead, we mention that the problem we face (regarding the communication between the user and the server) is that *the user and server do not know one another and may not understand each other's language*, where a language may mean either as a natural language or a "computer language" that specifies actions according to a predetermined formal protocol (or system). From our point of view (as users), the server is one of several possible servers; that is, it is selected arbitrarily in a class of possible servers, where each server in the class is potentially helpful but may use a different language. Thus, in order to benefit from interacting with the server, we must (get to) know its language. We shall return to these issues later on.

A general notion of a goal. The notion of a goal refers to the way we (the users) wish to effect the environment and/or to the information we wish to obtain from it. Without loss of generality, we can incorporate the information that the user obtains in the state of the environment (i.e., the environment may record that certain information was communicated to the user and the user can communicate to the environment whatever it has inferred, possibly, from communication with the server). Thus, we formalize the notion of a *goal* by focusing on the *evolution of (the state of) the environment*, which may be viewed as an execution of the user–server–environment system. The goal is captured by two mathematical objects: The first is a Boolean predicate which determines if an infinite evolution of the states of the environment satisfies the goal. The second object captures all that is known (or postulated) about the operation of the environment; that is, the way that the environment reacts to various actions of the user and/or server. (Recall that these actions and reactions are modeled as communication between the user/server and the environment.)

We stress that the environment may model also other processes that are executed by the party that invokes the user and/or server strategies. Indeed, such processes may affect our goals, but they are external to the (user) strategy that we employ in order to achieve the current goal, and therefore we view them as part of the environment. Thus, the notion of an environment does not necessarily reflect an external physical environment (although it may indeed incorporate one), but rather captures all that is external to the strategies that we employ towards achieving our goal. (The same holds also with respect to the server.)

As usual, it is instructive to consider a couple of examples. The first example refers to using a printer; that is, our goal is to print some document using a printer, which is viewed as a server. In this case, we model the document that we wish to print as a message coming from the environment (since indeed the documents we wish to print come from some other process that we are involved in), and the printed document is modeled as a message of the server to the environment. Indeed, the "printing goal" is an archetypical case of an effect we wish to have on the environment, where this effect can be performed by the server. In contrast, there are goals that are centered at obtaining information from the environment. Consider, for example, a web-server provided weather forecast, and our goal of deciding whether or not to take an umbrella. Note that in this case, our decision is modeled by a message that the user sends the environment, specifying whether or not it decided to take an umbrella. In both examples, we need to communicate with the server in order to achieve our goal, and thus we need to understand its language (at least at a level that suffices for that communication).

At this point we note that our actual modeling is not confined to a single performance of such a printing job or a weather-based decision, but rather allows to model an infinite sequence of such jobs. This modeling reflects the fact that we are actually interested in multiple instances of the same type, and that we may be willing to tolerate failure on few of these instances. Indeed, a natural class of goals consists of "multi-session goals" that correspond to an infinite sequence of similar sub-goals. Such goals are an important motivating case of compact goals, discussed next.

Compactness. The foregoing description of a goal possesses no restrictions on the corresponding predicate that determines whether or not the goal is satisfied (i.e., we are successful) in a specific execution. Consequently, in general, the set of successful executions may be arbitrary and thus not measurable with respect to the natural probability measure of executions. To this end we identify a natural subclass of goals which are more amenable to analysis, and tend to reflect most natural goals. We note that natural goals (and in particular multi-session goals) are "compact" in the sense that the success of infinite executions can be reflected in the "tentative success" (or "progress") that is associated with all finite execution prefixes. Specifically, an execution of the system (associated with a compact goal) is successful if and only if all but finitely many prefixes look fine (i.e., show progress towards the goal or at least no severe regression with respect to it). Compactness is the key not only to measurability of the set of successful executions but also to our (as users) "sensing" whether the executions is progressing well. Before discussing this notion of sensing, we note that not all goals are achievable.

Achievable goals and user–server communication. A goal is achievable if there exists a user strategy that achieves it (i.e., yields a successful execution with probability 1) when interacting with a suitable server. In trivial cases, where the user can achieve the goal without even communicating with the server, any server will do. But if the server's help is required, then the question of whether the user and sever understand one another arises. Such an understanding is required if the user relies on the server for either affecting the environment or for obtaining some information from the environment. For example, in the printing goal, if the user wishes to print some image, then it must communicate the image to the printer in an adequate format. In the weather goal, if the user wishes to obtain a weather forecast, then it must understand the language used by the web-server.

As stated already, *our focus is on situations in which the user interacts with a server that is selected arbitrarily among a class of possible servers.* The user is only guaranteed that each of these servers is helpful in the sense that when using an adequate strategy (e.g., the right file format in the case of the printing goal or the right language in the case of the web-server) the user may achieve the goal (via communication with the server). However, the user does not know the identity of the server *a priori* and/or does not know which strategy (e.g., format or language) to use. In this case, a good idea for the user is to try *some* strategy, and see what happens.

Sensing. Trying our luck seems like a good idea we (i.e., the users) can *sense* whether our choice is a good one, i.e., if we can sense whether our current strategy leads to progress towards achieving our goal. Formally, a sensing function is just a Boolean predicate computable by the user. Loosely speaking, this sensing function is "safe" if whenever the execution leads to no progress, the sensing function evaluates to 0 and the user obtains a negative indication. (We also allow sensing functions that have some delay built into them and only detect lack of progress after some time.) The complementary notion is "viability", which means that the user always obtains a positive indication when interacting with *some* server. Indeed, if the sensing process is both safe and viable, then the user achieves the goal when interacting with the latter server. Furthermore, when interactive with a different server that causes the user to fail (in achieving the goal), the user senses this misfortune after a finite amount of time.

Helpful servers. As hinted before, our (as users) ability to achieve our goals depends on our ability to communicate with an adequate server. A minimal requirement is that this server is helpful in the sense that there exists a user strategy that achieves the said goal when interacting with this server.

Access to a helpful server does not suffice – it is only a necessary requirement: we (as users) need to be able to effectively communicate with this server, which means communicating in a way that the server understands what we say and/or we understand the server's answers. A key point here is that the user is only guaranteed access to some helpful server, whereas the class of helpful server contains a large variety of servers, which use different communication languages (or formats or protocols). Not knowing a priori with which server it communicates, the user has to cope with the communication problem that is at the core of the current work: *how to conduct a meaningful communication with alien* (to it) *entities.*

Universal user strategies. A strategy is called universal with respect to a given goal if it can overcome the said communication problem with respect to that goal. That is, this strategy achieves the goal when communicating with an *arbitrary* helpful server. In other words, if *some* user (strategy) achieves the goal when communicating with the given server, then the universal user (strategy) also achieves the goal when communicating with this server. Thus, a universal user strategy is able to conduct a meaningful communication with any helpful server, where the communication is called meaningful (with respect to a given goal) if it allows the user to achieve the goal.

The design of good (i.e., safe and viable) sensing processes is the key to the design of universal user strategies. Specifically, having access to a helpful server and employing a "sufficiently good" sensing process allows the user to be universal. Actually, we need a combination of the helpfulness and viability condition: The combined condition requires that, for every helpful server, there exists a user that employs a safe sensing process and achieves the goal when interacting with this server and *while obtaining positive indication* (of success) all along. We say that this server satisfies enhanced helpfulness.

Theorem 2.1 (main result, loosely stated): *Let G be a goal and suppose that $\mathcal{S}$ a class of servers that are enhancedly helpful with respect to G. Then, there exists a user strategy U that is universal* *with respect to the server class $\mathcal{S}$ and the goal G; that is, the strategy U achieves the goal when interacting with any server in $\mathcal{S}$.*

Note that the theorem holds trivially when $\mathcal{S}$ contains a single server, but our focus is on the case that $\mathcal{S}$ contains numerous different servers that are all (enhancedly) helpful.

Essentially, Theorem 2.1 is proved by showing that having access to a helpful server and employing a good sensing process allows the user to try all possible communication strategies and abandon each such strategy as soon as it senses that this strategy leads to no progress. The amount of damage caused by bad strategies is proportional to the quality of the sensing process as well as to the index of the adequate strategy in the enumeration of all possible strategies.

The reader may be disappointed by the fact that the universal strategy just tries all possible user strategies and criticize the overhead (in terms of damage and/or delay) caused by this approach. The answer to these sentiments is three-fold.

1. Theorem 2.1 is merely a first step in a new direction. It establishes a general feasibility result, and opens the door to further study (see the third item).

2. The overhead of Theorem 2.1 is actually the best possible within the general framework in which it is stated. Specifically, one of our secondary results is a proof that for a natural class of servers no universal user strategy can have a significantly smaller overhead than the one offered by Theorem 2.1.

3. In light of the previous two items, Theorem 2.1 calls for future study of the possibility of improvement in a variety of natural *special cases*. Specifically, we conjecture that there exists natural classes of servers for which universality holds with overhead that is proportional to the logarithm of the index of the actual server (rather than to the index itself).

We note that we also establish refined versions of Theorem 2.1 in which the overhead (i.e., amount of damage or delay) is tightly related to the quality of the sensing process.

We stress that Theorem 2.1 applies to *any* class of (enhancedly) helpful servers and not only to the class of *all* (enhancedly) helpful servers. We consider this point important. On the one hand, the wider the class of servers for which universality holds, the better. But, on the other hand, generality comes with a cost, while well-motivated restrictions of the class of the helpful servers may offer better quantitative results (i.e., lower overhead and/or more efficient procedures).

2.2 Ramifications

Our general framework and basic ideas facilitate numerous ramifications, some of them are explored in Sections 3 and 4 and more will appear in a future version of this work. These ramifications include several variants of the basic universality result, the identification of numerous special cases and their initial study, and proofs of the inherent limitations on the ability to achieve certain goals. A few examples are discussed below.

The effect of size. The foregoing discussions made no reference to the size of the "instances" (i.e., system configurations) that arise in an execution. However, a more refined study may seek to allow various quantities (e.g., complexities, delays, number of errors) to depend on the size of the instances at hand. Our basic treatment in Sections 3 and 4 supports such a possibility, but (for sake of simplicity) this treatment postulates that the size of instances is fixed throughout the execution. The general case of varying sizes will be treated in a future version of this work.

Resettable servers. One natural special case of servers that we consider is the class of servers that can be reset by the user. In the context of solving computational problems, we note that such servers correspond to memoryless programs (and so sensing functions with respect to them correspond to program checkers [4]), whereas general servers correspond to potentially cheating provers in interactive proof systems [7]. Given the widely-believed separation between the power of these two models, our results confirm the benefit in being able to reset the server.

One-shot goals. The foregoing discussions refer to reactive systems and to goals that are defined in terms of infinite executions. In terms of the natural special case of multi-session goals, this means an infinite number of (bounded-length) sessions and our definitions allow to ignore a finite number of them. In contrast, one may be interested in a single (bounded-length) session, which means that achieving the goal requires full awareness of success (before termination). We call such goals one-shot, and note that they are the framework studied in [9, 10]. In a future version of this work, we shall provide a treatment of one-shot goals using the much more transparent modeling of the current exposition.

3 Goals: Parties, Compactness, Achievability, and Sensing

3.1 The parties

We consider three types of parties: a user, which represents "us" (or "our point of view"), a server (or a set of servers), which represents "other entities" (the help of which "we" seek), and a world, which represents the environment in which the user and server(s) operate. The world may provide the user and server with feedback on the effect of their actions on the environment (where the actions are modeled as messages sent to the world), and it may also model the way the environment changes in response to these actions. The world will also determine whether a goal was achieved (which is also a feedback that the world may, but need not, communicate to the user and server). The interaction among these (three types of) parties will be represented by strategies.

Strategies. We prefer to present strategies as explicitly updating the party's internal state (as well as determining its outgoing messages). The set of states in which the system may be in is denoted Ω (indeed, we may assume that $\Omega = \{0,1\}^*$). The state of the system (a.k.a the global state) at any point in time is the concatenation of the internal states of the various parties and the messages that are in transit among the parties. Indeed, the internal state of each party (resp., the message in transit between a pair of parties) is merely a projection of the global state. Fixing the number of parties to m (e.g., $m = 3$ is the most common case), for every $i \in [m] \stackrel{\rm def}{=} \{1, ..., m\}$, we denote the internal state of the $i^{\rm th}$ party when the system is in (global) state $\sigma \in \Omega$ by $\sigma^{(i)}$, and denote the set of possible internal states of the $i^{\rm th}$ party by $\Omega^{(i)}$ (i.e., $\Omega^{(i)} = \{\sigma^{(i)} : \sigma \in \Omega\}$. The canonical system that we consider consists of a world player, denoted $\mathtt{w}$, a user denoted $\mathtt{u}$, and a single server, denoted $\mathtt{s}$; see Figure 1. Likewise, the message in transit from the $i^{\rm th}$ party to the $j^{\rm th}$ party is denoted $\sigma^{(i,j)}$ (and the corresponding set of possible messages is denoted $\Omega^{(i,j)}$). We refer to a synchronous model of communication in which, at each round, each party sends messages to all other parties.

Definition 3.1 (strategies): *A strategy of the $i^{\rm th}$ party in* (an m-party system) *is a function from $\Omega^{(i)} \times (\times_{j\neq i}\Omega^{(j,i)})$ to $\Omega^{(i)} \times (\times_{j\neq i}\Omega^{(i,j)})$ which represents the actions of the party in the current communication round. That is, the argument to the function represents the party's internal state*

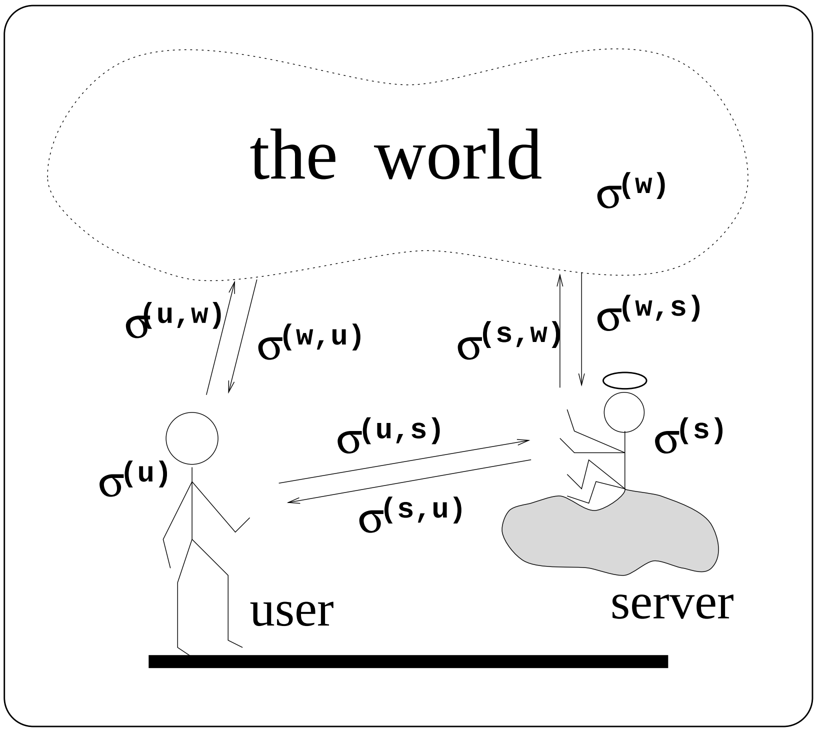

Figure 1: The canonical system: the world, user, and server.

and the $m-1$ messages it has received in the previous round, and the function's value represents its updated internal state and the $m-1$ messages that it sends in the current round.

Indeed, such a strategy modifies the global state such that the change only depends on the corresponding local (internal) state (and the relevant messages in transit), and its effect is restricted in an analogous manner. Still, to simplify our notation, we will often write strategies as if they are applied to the (entire) global state and update the (entire) global state.

The world. Intuitively, the user's goal is to have some effect on the environment. Furthermore, also effects on the server (or on the user itself) can be modeled as effects of the environment (e.g., by letting these parties communicate their internal states to the environment/world). Thus, part of the world's internal state indicates whether the desired goal has been (temporarily) achieved. Actually, we will consider a more general notion of achieving goals, a notion that refers to an infinite execution of the system. Intuitively, this may capture reactive systems whose goal is to repeatedly achieve an infinite sequence of sub-goals. Thus, we augment the world with a referee, which rules whether such an infinite execution (actually, the corresponding sequence of the world's local states) is successful.

Definition 3.2 (referees and successful executions): *A referee R is a function from infinite executions to a Boolean value; that is, $R:\Omega^\omega \to \{0,1\}$* (or, actually, $R:(\Omega^{(\mathtt{w})})^\omega \to \{0,1\}$)*. Indeed, the value of $R(\sigma_1,\sigma_2,...)$ only depends on $\sigma_1^{(\mathtt{w})},\sigma_2^{(\mathtt{w})},...$* (and it may be written as $R(\sigma_1^{(\mathtt{w})},\sigma_2^{(\mathtt{w})},...)$)*. We say that the infinite execution $\overline{\sigma}=(\sigma_1,\sigma_2,...)\in\Omega^\omega$ is* successful (w.r.t R) *if $R(\overline{\sigma})=1$.*

The combination of the world's strategy and a referee gives rise to a notion of a goal. Intuitively, the goal is to affect the world (environment) in a way that is deemed successful by the referee.

Probabilistic and non-deterministic strategies. So far, our formalism has referred to deterministic strategies. We wish, however, to consider also probabilistic strategies for all parties. Generalizing Definition 3.1, such a strategy is a *randomized process that maps pairs consisting of the party's current state and the* $m-1$ *messages that it has received in the previous round to a distribution over pairs representing the party's updated internal state and the* $m-1$ *messages that it sends in the current round.* On top of probabilistic strategies, we wish to model also arbitrary changes in the environment that are independent of the interaction among the players; that is, external (to the interaction) events that change the environment (i.e., the world's internal state). Such changes only depend on the world's current state and they they are confined to several predetermined possibilities. Indeed, such changes can be modeled by non-deterministic steps of the world. Assuming that the world never returns to the same state, such on-line non-deterministic choices (or steps) can can be modeled by an off-line non-deterministic choice of a probabilistic strategy for the world (chosen from a set of predetermined possibilities).[8]

Definition 3.3 (the world's strategy, revisited): *The* world's (non-deterministic) strategy *is defined as a set of probabilistic strategies, and the* actual world's strategy *is an element of the former set.*

Having revised our definitions of strategies, we are ready to formally define goals and executions.

Definition 3.4 (goals): *A* goal *is a pair consisting of a* (non-deterministic) *world strategy and a referee.*

Indeed, the non-deterministic world strategy describes the possible behavior of the environment in which we operate (including the way it interacts with the user and server), whereas the referee determines what executions are deemed successful.

When defining executions, we fix an actual world's strategy that is consistent with the world's (non-deterministic) strategy (i.e., an element of the latter set). Fixing probabilistic strategies to all parties gives rise to a sequence of random variables that represents the distribution over the possible sequences of (global) states of the system.

Definition 3.5 (executions): *An* execution *of a system consisting of the* m *probabilistic strategies, denoted* $P_1, ..., P_m$*, is an infinite sequence of random variables* $X_1, X_2, ...$ *such that for every* $t \geq 1$ *and every* $i \in [m]$ *it holds that*

$$(X_{t+1}^{(i)}, X_{t+1}^{(i,\cdot)}) \leftarrow P_i(X_t^{(i)}, X_t^{(\cdot,i)}),$$

where $X_t^{(\cdot,i)} = (X_t^{(j,i)})_{j \neq i}$ *and* $X_{t+1}^{(i,\cdot)} = (X_{t+1}^{(i,j)})_{j \neq i}$*. Unless it is explicitly stated differently, the execution starts at the system initial state* (i.e., X_1 equals a fixed initial global state). *An* execution *of the system* $P_1, ..., P_m$ *starting in an arbitrary global state* σ_1 *is define similarly, except that* X_1 *is set to equal* σ_1.

When we wish to consider an arbitrary value in the support of the sequence $\overline{X} = (X_1, X_2, ...)$, we shall use the term an actual execution. For example, we say that the execution $\overline{X}$ succeeds with probability p if the probability that $\overline{X}$ belongs to the set of (actual) successful executions equals p. Referring to the foregoing framework, let us consider a few examples.

Example 3.6 (predicting the world coins): *A simple, but impossible to achieve, goal is predicting the world's coin tosses. This goal may be formulated by considering a* (single actual)[9] *world strategy*

[8] Indeed, the latter set of possible probabilistic strategies may be isomorphic to the set of reals. Our treatment of probabilistic and non-deterministic choices is intentionally different: it facilitate fixing the non-deterministic choices and considering the distribution of the execution of the residual probabilistic system (which consists of probabilistic strategies).

[9] Indeed, in this example, the world's non-deterministic strategy is a singleton, containing a single actual strategy.

that, at each round, tosses a single coin and sets its local state according to the coin's outcome, and a referee that checks whether (at each round) *the message sent by the user to the world equals the world's current state. Since this world's actual strategy does not communicate any information to the user, no user strategy may succeed with positive probability* (since the number of rounds exceeds the logarithm of the reciprocal of any positive number).

Note that in this example no server can help the user to achieve its goal (i.e., succeed with positive probability). In contrast, if the world communicate its state to the server, and the referee checks whether the message sent by the user to the world (at each round) equals the world's state two rounds before, then an adequate server may help the user succeed with probability 1.

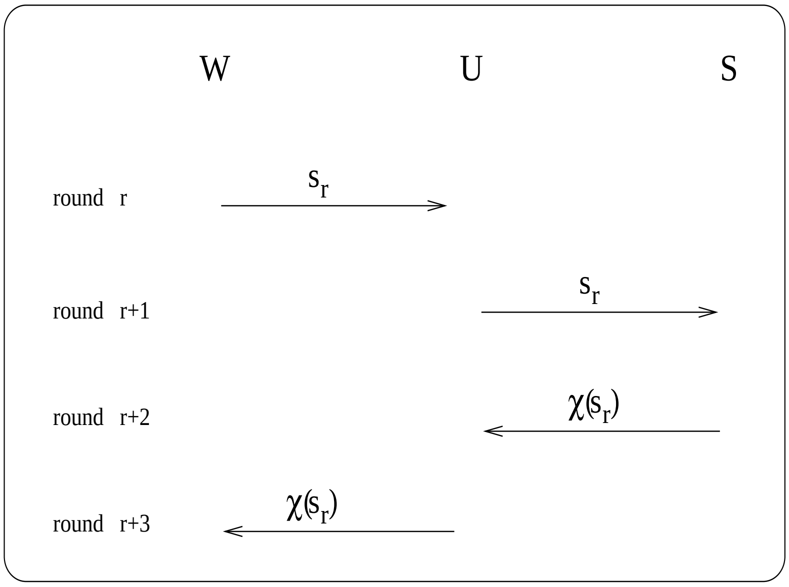

Figure 2: The time-line for getting the server's help in deciding instances of D.

Example 3.7 (solving computational problems posed by the world): *For a fixed decision problem D, consider a non-deterministic world strategy that in round r generates an arbitrary r-bit string, denoted s_r, and communicates it to the user, and a referee that checks whether, for every $r > 2$, the message sent by the user to the world at round r equals $\chi_D(s_{r-2})$, where $\chi_D(s) = 1$ if and only if $s \in D$. Indeed, this goal can be achieved by the user if and only if in round $r+1$ it has computational resources that allow for deciding membership in $D \cap \{0,1\}^r$.*

Note that also in this example no server can help the user, since the user obtains the "challenge" at round r and needs to answer at round $r+2$ (which does not allow for communicating the challenge to the server and obtaining the server's answer in time). In contrast, if the goal is modified such that the referee checks the user's message in round r against the world's message of round $r-3$, then communicating with a server that has computing power that exceeds the user's power may be of help. Indeed, in this modified goal, communication between the user and the server allows the user to obtain computational help from the server (see Figure 2). A goal in which the server's help is required, regardless of computational resources, follows.

Example 3.8 (printing): *Think of the server as a printer that the user wishes to use in order to print text that is handed to it by the environment. That is, consider a non-deterministic world* *strategy that at each round r generates an arbitrary bit $b_r \in \{0,1\}$ and communicates b_r to the user, and a referee that checks whether, for every $r > 2$, the message sent by the sender to the world at round r equals b_{r-2}.*

Indeed, the only way that a user can achieve this goal is by transmitting b_r to the server in time $r+1$, and counting on the server to transmit this bit to the world in round $r+2$.

The computational complexity of strategies. Since strategies are essentially functions, it is natural to define their complexity as the complexity of the corresponding functions. We follow this convention with two modifications (adaptations):

1. We define complexity with respect to the *size* of the current state (rather than with respect to the length of its description), where size is an adequate function of the state that need not equal the length of its description. Nevertheless, typically, the size will be polynomially related to the length, but this relation need not be fixed a priori.

2. We define the complexity of a (user) strategy with respect to the specific party (i.e., sever) with which it interacts. This convention facilitates reflecting the phenomenon that some servers allow the user to "save time"; that is, the complexity of the user is lower when interacting with such servers.

3.2 Compact goals

Examples 3.6–3.8 belong to a natural class of goals, which we call compact. In compact goals success can be determined by looking at sufficiently long (but finite) prefixes of the actual execution. Indeed, this condition refers merely to the referee's predicate, and it guarantees that the set of successful executions is measurable with respect to the natural probability measure (see Appendix). Furthermore, the compactness condition also enables the introduction of the notion of user-sensing of success (see Section 3.4).

By incorporating a record of all (the relevant information regarding) previous states in the current state, it suffices to take a decision based solely on the current state.[10] As in the case of the referee function R, the temporary decision captured by R' is actually a function of the world's local state (and not of the entire global state).

Definition 3.9 (compactness): *A referee $R : \Omega^\omega \to \{0,1\}$ is called* compact *if there exists a function $R' : \Omega \to \{0,1,\perp\}$* (or, actually, $R' : \Omega^{(\mathtt{w})} \to \{0,1,\perp\}$) *such that for every $\overline{\sigma} = (\sigma_1, \sigma_2, ...) \in \Omega^\omega$ it holds that $R(\overline{\sigma}) = 1$ if and only if the following two conditions hold*

1. The number of failures is finite:

   There exists T such that for every $t > T$ it holds that $R'(\sigma_t) \neq 0$ (or, actually, $R'(\sigma_t^{(\mathtt{w})}) \neq 0$).

2. There are no infinite runs of $\perp$:

   For every $t > 0$ there exists $t' > t$ such that $R'(\sigma_{t'}) \neq \perp$.

The function R' is called the temporal decision function.

[10] That is, consider a definition analogous to Def. 3.9, where $R' : \Omega^* \to \{0,1,\perp\}$ and the conditions refer to $R'(\sigma_1, \sigma_2, ..., \sigma_i)$ rather than to $R'(\sigma_i)$. Then, using $(\sigma_1, \sigma_2, ..., \sigma_i)$ as the i^{th} state, allows to move to the formalism of Def. 3.9. Furthermore, in typical cases it suffices to include in the i^{th} state only a "digest" of the previous $i-1$ states.

Indeed, the special symbol $\perp$ is to be understood as suspending decision regarding the current state. Definition 3.9 asserts that an execution can be deemed successful only if (1) failure occurs at most a finite number of times and (2) decision is not suspended for an infinite number of steps. (A stronger version of (Condition 2 of) Definition 3.9 may require that there exists B such that for every $t > 0$ there exists $t' \in [t+1, t+B]$ such that $R'(\sigma_{t'}) \neq \perp$.)[11]

Multi-session goals. Examples 3.6–3.8 actually belong to a natural subclass of compact goals, which we call multi-session goals.[12] Intuitively, these goals consists of an infinite sequence of sub-goals, where each sub-goal is to be achieved in a finite number of rounds, which are called the current session. Furthermore, the world's state is (non-deterministically) reset at the beginning of each session (indeed, as in Example 3.7). We further restrict such goals in the following definition, where these restrictions are aimed to capture the intuitive notion of a multi-session goal.

Definition 3.10 (multi-session goals): *A goal consisting of a non-deterministic strategy $\mathcal{W}$ and a referee R is called a* multi-session goal *if the following conditions hold.*

1. The world's states: *The local states of the world are partitioned into three non-empty sets consisting of* start-session *states,* end-session *states, and* (intermediate) session *states. Each of these states is a pair consisting of an* index (representing the index of the session) *and a* contents (representing the actual execution of the session).[13] *The initial local state corresponds to the pair $(0, \lambda)$, and belongs to the set of end-session states.*

2. The referee suspends verdict till reaching an end-session state: *The referee R is compact. Furthermore, the corresponding temporal decision function R' evaluates to $\perp$ if and only if the current state is not an end-session state.*

3. Starting a new session: *When being in an end-session state, the world moves non-deterministically to a start-session state while increasing the index. Furthermore, this move is independent of the actual contents of the current end-session state. That is, for each actual world strategy $W \in \mathcal{W}$, the value of W is invariant over all possible end-session states that have the same index* (i.e., for every two end-session state (i, σ') and (i, σ''), it holds that $W(i, \sigma')^{(\mathtt{w})} = W(i, \sigma'')^{(\mathtt{w})} \in \{i+1\} \times \Omega$, and similarly for $W(i, \cdot)^{(\mathtt{w}, \cdot)}$).

   Optional: *The world can also notify the user that a new session is starting, and even whether or not the previous session was completed successfully* (i.e., with R' evaluating to 1). *Analogous notifications can also be sent to the server.*

[11] It is tempting to suggest even a stronger version of Definition 3.9 in which both T and B are absolute constants, rather than quantities determined by the sequence $\overline{\sigma}$; however, such a stronger definition would have violated some of our intuitive desires. For example, we wish to focus on "forgiving" goals that are achieved even if the user adapts a good strategy only at an arbitrary late stage of the execution, and so we cannot afford to have T be execution invariant. Also, for an adequate notion of "size" (of the current state), we wish to allow the user to achieve the goal by interacting with a server for a number of rounds that depends on this size parameter (and suspend decision regarding success to the end of such interactions). In fact, we even "forgive" infinite runs of $\perp$'s if they result from a permanent increase in the size parameter.

[12] Actually, to fit Examples 3.7 and 3.8 into the following framework we slightly modify them such that the world generates and sends challenges only at rounds that are a multiple of three. Thus, the i^{th} session consists of rounds $3i, 3i+1, 3i+2$.

[13] The states are augmented by an index in order to allow for distinguishing the same contents when it occurs in different sessions. This is important in order to allow different non-deterministic choices in the different sessions (cf. Condition 3).

4. Execution of the current session: *When being in any other state, the world moves probabilistically while maintaining the index of the state* (i.e., for every $W \in \mathcal{W}$ and such state (i, σ'), it holds that $W(i, \sigma') = (i, \cdot)$). *Furthermore, the movement is independent of the index as well as of the actual world strategy; that is, for every* $W_1, W_2 \in \mathcal{W}$ *and every* $i_1, i_2 \in \mathsf{N}$ *and* $\sigma', \sigma'' \in \Omega$, *it holds that* $\Pr[W_1(i_1, \sigma') = (i_1, \sigma'')]$ *equals* $\Pr[W_2(i_2, \sigma') = (i_2, \sigma'')]$.

The execution of a system that corresponds to Def. 3.10 consists of a sequence of sessions, where each session is a sequence of states sharing the same index. Indeed, all the states in the i^{th} such sequence have index i, and correspond to the i^{th} session. The temporal decision function R' determines the success of each session based solely on the state reached at the end of the session (which includes also the session's index), and it follows that the entire execution is successful if and only if all but finitely many sessions are successful. We stress that, except for the index, the world's local state carries no information about prior sessions. Furthermore, with the exception of the initial move into a start-session state, the world's actions during the session are oblivious of the session's index. (In contrast to the world's action, the strategies of the user and server may maintain arbitrary information across sessions, and their actions in the current session may depend on this information.)

Repetitive (multi-session) goals. A special type of multi-session goals consists of the case in which the world repeats the non-deterministic choices of the first session in all subsequent sessions. We stress that, as in general multi-session goals, the world's probabilistic choices in each session are independent of the choices made in other choices.[14]

Definition 3.11 (repetitive goals): *A multi-session goal consisting of a non-deterministic strategy* $\mathcal{W}$ *and a referee* R *is called a* repetitive *if its non-deterministic choice is independent of the index; that is, for every* $W \in \mathcal{W}$ *and every* $i \in \mathsf{N}$ *and* $\sigma' \in \Omega$, *it holds that* $W(i, \sigma') \equiv W(1, \sigma')$.[15]

Indeed, any multi-session goal using a world strategy that makes no non-deterministic choices (cf., e.g., Example 3.6) is a repetitive goal. An example of a repetitive goal that does involve non-deterministic choices follows.

Example 3.12 (repeated guessing with feedback): *Consider a non-deterministic world strategy that generates an integer i and proceeds in sessions. Each session consists of two rounds, where in the first round the user sends a guess to the world, and in the second round the world notifies the user whether or not its guess was correct* (i.e., whether or not the message sent by the user in the first round equals i). *The referee deems a session successful if the user sent the correct message i. Indeed, by recording all previous failed attempts, the user can eventually succeed in a single session, be informed about it, and repeat this success in all subsequent sessions.*

Indeed, the feedback provided by the world is essential for the user's ability to (eventually) succeed in guessing the world's initial choice.

Generalized multi-session goals. Our formulation of multi-session goals mandates that the current session must end before any new session can start (see Definition 3.10). A more general formulation, which allows concurrent sessions, will appear in a future version of this work.

[14]Indeed, a stronger notion, which we do not consider here, requires that the world also repeats the probabilistic choices of the first session in all subsequent sessions. We note that this stronger notion cannot be captured in the current formalism.

[15]We used $X \equiv Y$ to indicate that the random variables X and Y are identically distributed. Note that if σ' is an end-session state, then $W(i, \sigma')$ and $W(1, \sigma')$ are actually fixed strings (and they must be equal).

3.3 Achieving Goals

We have already touched on the notion of achieving a goal, but now we turn to define it formally, while assuming that the corresponding referee is compact (as per Definition 3.9). As detailed in the Appendix, the compactness assumption implies that the set of successful executions is measurable (with respect to the natural probability measure). The basic definition of achieving a goal is as follows.

Definition 3.13 (achieving goals): *We say that a pair of user-server strategies, (U, S),* achieves the goal *$G = (\mathcal{W}, R)$ if, for every $W \in \mathcal{W}$, a random execution of the system (W, U, S) is successful with probability 1, where success is as in Def. 3.2.*

Recall that by Definition 3.5, our convention is that (unless stated differently) the execution starts at the system's (fixed) initial global state. However, in the sequel we will be interested in what happens when the execution starts in an arbitrary state, which might have been reached before the actual execution started. This reflect the fact that the environment (or world) is not initialized each time we (users) wish to achieve some goal, and the same may hold with respect to the servers that we use. Thus, a stronger notion of achievable goals arises.

Definition 3.14 (robustly achieving goals): *We say that a pair of user-server strategies, (U, S),* robustly achieves the goal *$G = (\mathcal{W}, R)$ if for every $W \in \mathcal{W}$ and every global state σ_1 a random execution of the system (W, U, S) starting in state σ_1 is successful with probability 1.*

Indeed, this notion of robust achievability is "forgiving" of an initial portion of the execution that may be carried on by inadequate user and/or server strategies.

Proposition 3.15 (robustness allows ignoring execution prefixes): *Let U_t* (resp., *S_t*) *be a user* (resp., server) *strategy that plays the first t rounds using the user strategy U_0* (resp., server strategy *S_0*) *and plays all subsequent rounds using the user strategy U* (resp., server strategy *S*)*. Then, if (U, S) robustly achieves the goal $G = (\mathcal{W}, R)$, then so does (U_t, S_t).*

The proof only uses the hypothesis that (W, U, S) is successful when started in a state that may be reached by an execution of W with an arbitrary pair of user and server strategies. Indeed, for all practical purposes, the definition of robust achievability may be confined to such initial states (i.e., in Definition 3.14, we may quantify only over states σ_1 that can be reached in some execution of the system (W_0, U_0, S_0), where $W_0 \in \mathcal{W}$ and (U_0, S_0) is an arbitrary user–server pair).

Proof: The proposition follows by considering the execution of the system (W, U, S) starting at the state, denoted σ_1, that is reached after t rounds of the system (W, U_0, S_0). (Indeed, σ_1 may be a distribution over such states.) By combining the robust achievability hypothesis (which refers to the execution of (W, U, S) started at σ_1) and the compactness hypothesis (which allows to discard the t first steps of (W, U_t, S_t)), we conclude that the execution of (W, U_t, S_t) (started at any state σ_1') is successful with probability 1. ■

Achievable goals. We may say that a goal $G = (\mathcal{W}, R)$ is achievable (resp., robustly achievable) if there exists a pair of user-server strategies that achieve (resp., robustly achieve) G. Indeed, as hinted before, predicting the world's coins (i.e., Example 3.6) is an unachievable goal, whereas the goals of Examples 3.7 and 3.8 are (robustly) achievable. Note, however, that the printing goal (i.e., Example 3.8) is achievable by a very simple user–server pair, whereas solving the computational problems posed by the world (i.e., Example 3.7) is achievable only by a sufficiently powerful user (i.e., one that can decide membership in D). Thus, achievable goals are merely our starting point; indeed, *starting with such a goal* G, we shall ask what should be required of a user–server pair that achieves G and what should be required of a user that can achieve this goal when paired with any server that is taken from a reasonable class.

3.4 Sensing

The achievability of the goal of "repeated guessing with feedback" (i.e., Example 3.12) relies on the feedback provided to the user (regarding its success in previous sessions). In general, such feedback is reasonable to assume in the context of many multi-session goals, and (as we shall see) such feedback can be helpful to the user also in non-repetitive goals.

Intuitively, we shall consider world strategies that allow the user to sense its progress towards achieving the goal, where this sensing should satisfy adequate safety and viability conditions. Loosely speaking safety means that if the user gets a positive indication (i.e., senses progress) almost all the time, then the goal is actually achieved, whereas viability means that when the goal is achieved the user gets positive indication almost all the time. Thus, infinitely many negative indications should occur if and only if the execution fails. (As usual, we will represent a positive indication by the value 1, and a negative indication by 0.)

The aforementioned indication is provided by a function, denoted U', of the user's current state. We stress that this function U' is tailored for the corresponding user strategy U, and should be viewed as an *augmentation of the user strategy* U. The function U' is required to be viable and safe. Note that viability is not meaningful without safety, and vice versa; for example, under any reasonable definition, the all-zero function is (trivially) safe, whereas the all-one function is (trivially) viable. Although we will be interested in safety and viability with respect to classes of possible servers, we find it useful to define restricted notions of safety and viability that refer to a fixed server strategy.

Definition 3.16 (user sensing function, weak version): *Let $G = (\mathcal{W}, R)$ be a compact*[16] *goal and S be a server strategy. The predicate $U' : \Omega \to \{0,1\}$* (or rather $U' : \Omega^{(\mathrm{u})} \to \{0,1\}$) *is* safe *with respect to (U,S)* (and G) *if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, letting $\overline{\sigma}$ denote a random execution of the system (W,U,S) starting at state σ_1, with probability 1, it holds that if $R(\overline{\sigma}) = 0$ then for infinitely many t it holds that $U'(\sigma_t) = 0$. The predicate U' is* viable *with respect to (U,S) if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, with probability 1, it holds that $U'(\sigma_t) = 0$ holds for finitely many t.*

Indeed, if U' is viable and safe with respect to (U,S) (and G), then (U,S) robustly achieves the goal G, because viability implies that a random execution yields finitely many negative indications, whereas safety implies that in such a case the goal is achieved. In particular, if U' is safe with respect to (U,S), then, with probability 1, if U' evaluates to 0 finitely many times, then the corresponding temporal decision function R' evaluates to 0 finitely many times.

The foregoing reference to the temporal decision function R' suggests stronger (i.e., quantified) notions of sensing. Intuitively, we seek a stronger notion of (safe) sensing in which failure (as per R') is sensed after a bounded number of steps (rather than eventually). Similarly, a stronger notion of viability should guarantee a positive indication after a bounded number of steps (rather than

[16] Actually, the current definition does not refer to the compactness condition (and is applicable also w.r.t non-compact goals). The compactness condition was added here for consistency with the following definitions, which do refer to it (or rather to the temporal decision function provided by it).

eventually). That is, in both cases, the "grace period" (of bad sensing) is explicitly bounded rather than merely postulated to be finite. This bound will be stated in terms of an adequate notion of "size" (of the current state), denoted $\mathsf{s}(\sigma)$, thus allowing the grace period to depend on the "complexity" (or rather the "size") of the relevant states. For simplicity, *we assume here that the size of the various states remains invariant throughout the execution*; the general case (in which the size varies) will be treated in a future version of this work. Our formulation will be further simplified by observing that the quantification over all initial states (which also takes place in Definition 3.16) allows to focus on grace periods that start at time 1 (rather than considering grace periods that start at time t for any $t \in \mathsf{N}$). These considerations lead to the following definition, which is a straightforward strengthening of Definition 3.16.

Definition 3.17 (user sensing function, very strong version): *Let $G = (\mathcal{W}, R)$, S, U, and U' be as in Def. 3.16, and let $\mathsf{s} : \Omega \to \mathsf{N}$ be the aforementioned size function. We say that U' is* very strongly safe *with respect to (U, S)* (and G) *if there exists a function $B : \mathsf{N} \to \mathsf{N}$ such that, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, the following two conditions hold.*

1. *If $R'(\sigma_1) = 0$, then, with probability at least $2/3$, for some $t \leq B(\mathsf{s}(\sigma_1))$ it holds that $U'(\sigma_t) = 0$, where σ_t denotes the system's state after t rounds.*
2. *If for every $i \in [B(\mathsf{s}(\sigma_1))]$ it holds that $R'(\sigma_i) = \perp$, then, with probability at least $2/3$, for some $t \in [i+1, i+B(\mathsf{s}(\sigma_1))]$ it holds that $U'(\sigma_t) = 0$, where σ_i, σ_t are as above.*

Analogously, U' is strongly viable *with respect to (U, S) if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, with probability at least $2/3$, for every $t \geq B(\mathsf{s}(\sigma_1))$ it holds that $U'(\sigma_t) = 1$. We say that strong viability holds perfectly if the foregoing holds with probability 1* (i.e., for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, with probability 1, it holds that $U'(\sigma_t) = 0$ holds for finitely many t).

We note that satisfying the first safety condition of Definition 3.17 implies that, for every $W \in \mathcal{W}$ and $\sigma_1 \in \Omega$ and every $T > 0$, if $R'(\sigma_T) = 0$ then, with probability at least $2/3$, for some $t \in [T, T + B(\mathsf{s}(\sigma_T))]$ it holds that $U'(\sigma_t) = 0$, where σ_i denotes the system's state after i rounds. Analogous statements apply to the second safety condition and to the viability condition (of Definition 3.17). It follows that very strong safety (resp., viability) as in Definition 3.17 implies weak safety (resp., viability) satisfying Definition 3.16 (because infinitely many sensing failures imply infinitely many disjoint B-long intervals containing sensing failure).[17] All of this will apply also to the following definition (which is a relaxation of Definition 3.17).

In order to motivate the following definition, note that Definition 3.17 requires that failure be detected even if the execution has recovered from it. For example, the first safety condition requires that U' senses that $R'(\sigma_1) = 0$ (i.e., $U'(\sigma_t) = 0$ for some $t \leq B(\mathsf{s}(\sigma_1))$) even if $R'(\sigma_i) = 1$ for every $i > 1$. Insisting on detection of an old (initial) failure that is no longer relevant seems unnecessary, and it may make the design sensing functions (unnecessarily) harder. The following (relaxed w.r.t Def. 3.17) definition requires detection of an initial failure only in the case that the entire execution has failed. In other words, if the sensing function "believes" that the possible initial failure is no longer relevant, then it is not required to signal an alarm.

[17]The foregoing sketchy justification seems to suffice for the case of strong viability that holds perfectly, but even in such a case a more rigorous argument is preferable. Indeed, suppose that weak viability as per Definition 3.16 is violated. This implies that, with positive probability, for a random execution $\overline{\sigma}$ there exist infinitely many $t \in \mathsf{N}$ such that $U'(\sigma_t) = 0$. But this contradicts strong viability (even in the general, non-perfect, sense) as per Definition 3.17, because for every $T \in \mathsf{N}$ with probability at least $2/3$ it holds that $U'(\sigma_t) = 0$ for every $t \leq T + B(\sigma_T)$. Dealing with the safety conditions is somewhat more complicated. One has to show that the very strong safety condition implies that the probability that a random execution $\overline{\sigma}$ is unsuccessful (i.e., $R(\overline{\sigma}) = 0$) and yet $\{t \in \mathsf{N} : U'(\sigma_t) = 0\}$ is finite is zero.

Definition 3.18 (user sensing function, strong version): *Let $G = (\mathcal{W}, R)$, S, U, and U' be as in Def. 3.16. We say that U' is* strongly safe *with respect to* (U, S) (and G) *if there exists a function $B : \mathbb{N} \to \mathbb{N}$ such that, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, the following conditions hold.*

1. *If $R'(\sigma_1) = 0$, then, with probability at least 2/3, either $R(\overline{\sigma}) = 1$ or for some $t \leq B(\mathsf{s}(\sigma_1))$ it holds that $U'(\sigma_t) = 0$, where $\overline{\sigma} = (\sigma_1, \sigma_2, ...,)$ denotes a random execution of the system (W, U, S).*
2. *If for every $i \in [B(\mathsf{s}(\sigma_1))]$ it holds that $R'(\sigma_i) = \perp$, then, with probability at least 2/3, either $R(\overline{\sigma}) = 1$ or for some $t \in [i+1, i+B(\mathsf{s}(\sigma_1))]$ it holds that $U'(\sigma_t) = 0$.*

The strong viability condition is exactly as in Def. 3.17.

We mention that the strong sensing version (i.e., as per Definition 3.18) implies the weak one (i.e., as per Definition 3.16).[18] We will refer mainly to the weak and strong versions (i.e., Definitions 3.16 and 3.18, respectively); the very strong version (i.e., Definition 3.17) was presented mainly for clarification.

Safety with respect to classes of servers. Sensing is crucial when the user is not sure about the server with whom it interacts. Recall that Section 3.3 ended with a declared focus on achievable goals; but this only means that the adequate user U can be sure that *it achieves the goal when it interacts with an adequate server.* But this user may not be aware that the server is actually not the designated one, and in such a case and if interaction with this server is not leading to success, then the user may wish to be notified of this failure. For this reason, we will be interested in sensing functions U' that are viable with respect to some (U, S_0) and satisfy the safety condition with respect to (U, S) for every S in a set of servers $\mathcal{S}$.

Definition 3.19 (safety w.r.t classes of servers). *For each version of safety, we say that U' is* safe with respect to U and the server class $\mathcal{S}$ (and the goal G) *if for every $S \in \mathcal{S}$ it holds that U' is safe with respect to* (U, S) (and G).

4 On Helpful Servers and Universal Users

Our focus is on the cases in which the user and server need to collaborate in order to achieve the goal. Indeed, in order to collaborate, the user and server may need to communicate. Furthermore, they need to understand one another. The latter requirement is non-trivial when the server may be selected arbitrarily within some class of helpful servers, where a server is helpful if it can be coupled with some user so that this pair achieves the goal. That is, at best, we can expect to achieve the goal when communicating with a server S for which there exists a user strategy U such that (U, S) achieves the goal. But even in this case, the mere existence of a suitable user strategy U does not suffice, because we may not know this strategy. Still, we start with the assumption that such a user strategy U exists, which leads to the definition of a helpful server.

Helpful servers. Fixing an arbitrary (compact) goal $G = (\mathcal{W}, R)$, we say that a server S is helpful if there exists a user strategy U such that (U, S) achieves the goal. We strengthen this helpfulness requirement in two ways. Firstly, we will require that (U, S) robustly achieves the goal, rather than merely achieves it. This strengthening reflects our interest in executions that

[18]This requires a proof; cf. Footnote 17.

start at an arbitrary state, which might have been reached before the actual execution started (cf. Definition 3.14). Secondly, at times, we may require that the user strategy U (for which (U, S) robustly achieves the goal) belongs to some predetermined class of strategies $\mathcal{U}$ (e.g., a class of efficient strategies).

Definition 4.1 (helpfulness): *A server strategy S is $\mathcal{U}$-helpful* (w.r.t the goal G) *if there exists a user strategy $U \in \mathcal{U}$ such that (U, S) robustly achieves the goal G.*

When $\mathcal{U}$ is not specified, we usually mean that helpfulness holds with respect to the class of all recursive user strategies.

4.1 Universality and guarded helpfulness

When allowed to interact with a known (to us) helpful server, we may achieve the goal (if we use the strategy U that is guaranteed by Definition 4.1). But *what happens when we are allowed to interact with a server that is selected arbitrarily among several helpful servers*? Specifically, suppose that both S_1 and S_2 are $\mathcal{U}$-helpful, does this mean that there exists a user strategy U (let alone in $\mathcal{U}$) such that both (U, S_1) and (U, S_2) achieve the goal? As shown next, the answer may be negative.

Example 4.2 (using one out of two different printers): *In continuation to Example 3.8, for every $i \in \{0, 1\}$, consider a printer S_i such that, in each round, upon receiving the message b from the user, the printer S_i sends the message $b \oplus i$ to the world. That is, if at round $r + 1$ the server S_i receives b, then at round $r + 2$ it sends the message $b \oplus i$ to the world. Note that each of these two server strategies is $\{U_0, U_1\}$-helpful, where U_i is a user strategy that at round $r+1$ sends $b_r \oplus i$ to the server, where $b_r \in \{0, 1\}$ denotes the message sent by the world to the user in round r. However, there exists no user strategy U such that both (U, S_0) and (U, S_1) achieve the goal.*

Indeed, one may think of U_1 and S_1 as using, for communication among them, a different language than the one used by the world (i.e., they interprets 0 as 1, and 1 as 0). This is not so odd if we bear in mind that the communication between the various pairs of parties represents communication over vastly different media; for example, the user obtains email (from the world), which the user sends to the printer in some adequate format, while the printer produces an image (in the world). Thus, Example 4.2 can be made more realistic by saying that there exists two text formating functions, denoted f_0 and f_1 (e.g., PostScript and PDF) such that the following holds: if, at round r, user U_i receives the email text T_r (from the world), then it sends $f_i(T_r)$ to the server in round $r + 1$, whereas when server S_j receives the message M from the user it prints an image of $f_j^{-1}(M)$ (i.e., it sends the message $f_j^{-1}(M)$ to the world).

Example 4.3 (two printers, modified): *In continuation to Example 4.2, we consider a modified goal in which the world sends in each round a pair of bits (b, s) such that b is as above* (i.e., as in Examples 3.8 and 4.2) *and s indicates whether the referee is satisfied with the last message received by the server. In this case, there exists a simple user strategy U such that both (U, S_0) and (U, S_1) achieve the goal. Specifically, U first behaves as U_0, and if it gets an indication* (in round 3) *that printing failed, then it switches to use U_1.*

Indeed, in this case the world's messages suggest a user sensing function that is both safe and viable (w.r.t the server class $\{S_0, S_1\}$). This sensing function allows the user to recover from failure (by learning with which server it interacts and acting accordingly).

Universal users. The user strategy U of Example 4.3 achieves the corresponding goal when coupled with any server strategy in the class $\mathcal{S} \stackrel{\text{def}}{=} \{S_0, S_1\}$. Thus, we may say that U is $\mathcal{S}$-universal (in the sense defined next).

Definition 4.4 (universality): *A user strategy U is $\mathcal{S}$-*universal (w.r.t the goal G) *if for every server strategy $S \in \mathcal{S}$ it holds that (U, S) robustly achieves the goal G.*

Needless to say, if U is $\mathcal{S}$-universal, then every $S \in \mathcal{S}$ must be $\mathcal{U}$-helpful for any $\mathcal{U}$ that contains U. Thus, *we cannot have $\mathcal{S}$-universal users whenever the server class $\mathcal{S}$ contains unhelpful strategies.* In fact, a stronger statement holds.

Proposition 4.5 (universal strategies have trivial user-sensing functions): *If U is $\mathcal{S}$-universal, then there exists a sensing function U' such that U' is strongly viable and* (weakly) *safe with respect to (U, S) for any $S \in \mathcal{S}$.*

Indeed, as its title indicates, the user-sensing function provided by the proof of Proposition 4.5 is rather trivial (and is based on the hypothesis that for every $S \in \mathcal{S}$ it holds that (U, S) achieves the goal).[19] Still Proposition 4.5 is meaningful as a *necessary condition for the design of $\mathcal{S}$-universal users*; that is, *we must be able to design a user-sensing function that is both safe and viable for the class of servers $\mathcal{S}$.*

Proof: Let U' be identically 1, and consider any $S \in \mathcal{S}$. Then, viability of U' (under any version) holds trivially. The weak version of safety (i.e., Def. 3.16) holds vacuously for U' (w.r.t (U, S)), because for every $W \in \mathcal{W}$ a random execution of (W, U, S) starting at any state σ_1 is successful with probability 1. ■

Proposition 4.5 provides a necessary condition for the design of universal users, but what we actually seek are sufficient conditions. The following theorem states a seemingly general sufficient condition for the existence of a $\mathcal{S}$-universal user: The main condition (i.e., the main part of Condition 1) is closely related to saying that every $S \in \mathcal{S}$ is $\mathcal{U}$-helpful in a strong sense; specifically, S can be used by a user strategy that is augmented with a sensing function that is viable with respect to S and safe with respect to the server class $\mathcal{S}$.

Theorem 4.6 (on the existence of universal strategies): *Let $G = (\mathcal{W}, R)$ be a compact goal, $\mathcal{U}$ be a set of user strategies and $\mathcal{S}$ a set of server strategies such that the following two conditions hold.*

1. *For every $S \in \mathcal{S}$ there exists a user strategy $U \in \mathcal{U}$ and a user sensing function U' such that U' is strongly viable with respect to (U, S) and is weakly safe with respect to U and the server class $\mathcal{S}$* (and G).[20] *Furthermore, the mapping $U \mapsto (U', B)$ is computable, where B is the bounding function guaranteed by the strong viability condition.*

2. *The set $\mathcal{U}$ is enumerable.*

Then, there exists an $\mathcal{S}$-universal user strategy (w.r.t G). *Furthermore, if the* (strong) *viability condition holds perfectly, then, for every $S \in \mathcal{S}$, the complexity of the universal user strategy when interacting with S is upper-bounded by the complexity of some fixed strategy in $\mathcal{U}$* (when interacting with S).

[19]Interestingly, strong safety does not seem to follow because the discrepancy between the bounded nature of the strong safety condition and the unbounded nature of the definition of achieving a goal. This discrepancy is eliminated in Section 4.4.

[20]See Definition 3.19.

Indeed, Condition 1 (which implies weak sensing as per Definition 3.16)[21] implies that every $S \in \mathcal{S}$ is $\mathcal{U}$-helpful; in fact, it implies that every $S \in \mathcal{S}$ is $\mathcal{U}$-helpful in a strong sense (to be defined in Definition 4.7). Also note that only weak safety is required in Condition 1. We mention that there is an intuitive benefit in having strong safety, but this benefit is not reflected by the statement of the theorem. We shall return to this issue in Section 4.4.

Proof: We construct a user strategy, denoted U, that operates as follows. The strategy U enumerates all $U_i \in \mathcal{U}$, and emulates each strategy U_i as long as it (via U_i') obtains no proof that U_i (coupled with the unknown server $S \in \mathcal{S}$) fails to achieve the goal. Once such a proof is obtained, U moves on to the next potential user strategy (i.e., U_{i+1}). If this "proof system" is sound, then U will never be stuck with a strategy U_i that (coupled with the unknown server $S \in \mathcal{S}$) does not achieve the goal. On the other hand, the completeness of this "proof system" and the hypothesis that every $S \in \mathcal{S}$ is $\mathcal{U}$-helpful imply that there exist a U_i that (once reached) will never be abandoned.

Needless to say, the foregoing argument depends on our ability to construct an adequate "proof system" (for evaluating the performance of various $U_i \in \mathcal{U}$). Let B_i be the bounding function guaranteed by the strong viability condition of U_i'; that is, viability guarantees that (with an adequate S) the sensing function U_i' will indicate success after at most $B_i(\mathsf{s}(\cdot))$ rounds. Thus, a good strategy is to wait for the system to recover (from potential past failures) for $B_i(\mathsf{s}(\cdot))$ rounds, and abandon the current U_i whenever U_i' indicates failure after this grace period. A more accurate description follows.

Let us first analyze the case where the (strong) viability condition hold perfectly; that is, with probability 1 (rather than with probability 2/3, as in the main part of Definition 3.17). Suppose that U starts emulating U_i at round t_i, and denote the system's state at this round by σ_{t_i}. Then, *for the first $b_i \leftarrow B_i(\mathsf{s}(\sigma_{t_i}))$ rounds strategy U just emulates U_i, and in any later round $t > t_i + b_i$ strategy U switches to U_{i+1} if and only if $U_i'(\sigma_t) = 0$.*

Claim 4.6.1 *Suppose that U_i' is strongly and* perfectly *viable with respect to (U_i, S), and consider a random execution of (W, U, S). Then, if this execution ever emulates U_i, then it never switches to U_{i+1}.*

Proof: Let $\overline{\sigma}$, t_i, and b_i be as above. Then, by the strong viability condition, for every $t > t_i + b_i$, it holds that $U_i'(\sigma_t) = 1$. □

Claim 4.6.2 *Suppose that (U, S) does not robustly achieves the goal and consider a random execution of (W, U, S). Then, recalling that each U_i' is* (weakly) *safe* (w.r.t (U_i, S), *this execution emulates each U_i for a finite number of rounds.*

Combining the foregoing two claims with the hypothesis that for every $S \in \mathcal{S}$ there exists a user strategy $U_i \in \mathcal{U}$ and a user-sensing function U_i' such that U_i' is strongly viable with respect to (U_i, S), it follows that (U, S) robustly achieves the goal.

Proof: Let σ_1 be a global state such that a random execution of the system starting at σ_1 fails with positive probability, and let $\overline{\sigma}$ be such an execution (i.e., $R(\overline{\sigma}) = 0$). Let t_i and b_i be as above. Then, by the (weak) safety of U_i' w.r.t (any) $S \in \mathcal{S}$ (cf., Definition 3.16), for some $t'' > t_i + b_i$ (actually for infinitely many such t's), it holds that $U_i'(\sigma_{t''}) = 0$, which causes U to switch to emulating U_{i+1}. □

The above analysis assumes perfect (strong) viability, which may not hold in general. In order to cope with imperfect viability (i.e., a strong viability condition that holds with probability 2/3) we

[21] See Footnote 18.

need to modify our strategy U. Specifically, we will use a "repeated enumeration" of all machines such that each machines appears infinitely many times in the enumeration. Furthermore, for every i and t there exists an n such that U_i appears t times in the first n steps of the enumeration (e.g., use the enumeration $1, 1, 2, 1, 2, 3, 1, 2, 3, 4, ...$). Using a modified version of Claim 4.6.1 that asserts that if the execution starts emulating U_i then it switches to U_{i+1} with probability at most $1/3$ (equiv., stays with U_i forever), we derive the main claim of the theorem (because after finitely many R'-failures, strategy U returns to emulating U_i).

Regarding the furthermore claim, we note that the complexity of U (when interacting with S) is upper bounded by the maximum complexity of the strategies $U_1, ..., U_i$, where i is an index such that (U_i, S) robustly achieves the goal. Note that the complexity of the enumeration can be absorbed in the complexity of the emulation itself (by using tricks such as "lazy enumeration"). ■

Guarded helpfulness. The hypothesis of Theorem 4.6 (specifically, Condition 1) refer to servers that are not only helpful but rather satisfy a stronger condition, which we call guarded helpfulness. Recall that a server S is called helpful if it allows for achieving the goal (by employing an adequate strategy U). Loosely speaking, guarded helpfulness means that the strategy U that can use S to achieve the goal is coupled with a sensing function U' that "protects" U in case the server strategy is not helpful to it (i.e., when interacting with $\widetilde{S}$ such that $(U, \widetilde{S})$ does not achieve the goal. That is, fixing a class of servers $\mathcal{S}$, we say that a server S (possibly in $\mathcal{S}$) is helpful in an enhanced (i.e., guarded) sense if S allows achieving the goal by employing a user strategy U that is coupled with a sensing function U' that is both (1) viable with respect to (U, S), and (2) safe with respect to the server class $\mathcal{S}$. Thus, S not only allows for achieving the goal but also allows to sense the success via a function that is safe with respect to the server class $\mathcal{S}$. That is, S is helpful to a user strategy U that has a sensing function U' that is viable (w.r.t (U, S)) and safe (w.r.t all strategies in $\mathcal{S}$). We may say that U' is "guarded w.r.t $\mathcal{S}$" (and so the helpfulness of S is "$\mathcal{S}$-guarded").

Definition 4.7 (enhanced (or guarded) helpfulness): *Let $G = (\mathcal{W}, R)$ be a compact goal, $\mathcal{U}$ be a set of user strategies and $\mathcal{S}$ a set of server strategies. A server strategy S is* $\mathcal{S}$-guarded $\mathcal{U}$-helpful (w.r.t the goal G) *if there exists a user strategy $U \in \mathcal{U}$ and a user sensing function U' such that*

1. *U' is strongly viable with respect to (U, S), and*
2. *U' is weakly safe with respect to U and the server class $\mathcal{S}$* (and G).

Recall that the hypothesis that U' is viable and safe (even only in a weak sense) with respect to (U, S) (and G) implies that (U, S) robustly achieves the goal G, which in turn implies that S is $\mathcal{U}$-helpful. We stress that guarded helpfulness is somewhat weaker than Condition 1 in Theorem 4.6 (i.e., the mapping of $U \mapsto (U', B)$ is not necessarily computable).

Note that there may be a difference between $\mathcal{S}_1$-guarded $\mathcal{U}$-helpfulness and $\mathcal{S}_2$-guarded $\mathcal{U}$-helpfulness, for $\mathcal{S}_1 \neq \mathcal{S}_2$, because a user-sensing function U' may be (viable and) safe with respect to $(U, \mathcal{S}_1)$ but not with respect to $(U, \mathcal{S}_2)$. This fact reflects the relation of guarded helpfulness to universality, discussed next.

4.2 From helpfulness to guarded helpfulness

Proposition 4.5 and Theorem 4.6 relate universality with guarded helpfulness. Specifically, Proposition 4.5 asserts that, if U is $\mathcal{S}$-universal, then every $S \in \mathcal{S}$ is $\mathcal{S}$-guarded $\{U\}$-helpful. On the other hand, Theorem 4.6 (essentially) asserts that, if every $S \in \mathcal{S}$ is helpful[22] in an $\mathcal{S}$-guarded manner, then there exists an $\mathcal{S}$-universal user strategy. Indeed, both $\mathcal{S}$-universality and $\mathcal{S}$-guarded helpfulness become harder to achieve when $\mathcal{S}$ becomes more rich (equiv., are easier to achieve when $\mathcal{S}$ is restricted, of course, as long as it contains only helpful servers).

Since plain helpfulness is self-evident in many settings, the actual issue is moving from it to guarded helpfulness. That is, the actual issue is transforming user strategies that witness the plain helpfulness of some class of servers $\mathcal{S}$ to user strategies that support $\mathcal{S}$-guarded helpfulness (of the same class of servers). A simple case when such a transformation is possible (and, in fact, is straightforward) is presented next.

Definition 4.8 (goals that allow trivial user-sensing): *We say that a compact goal $G = (\mathcal{W}, R)$* allows trivial user-sensing *if, at each round, the corresponding temporal decision function R' evaluates to either 0 or 1, and the world notifies the user of the current R'-value; that is, for every $W \in \mathcal{W}$ and every $\sigma \in \Omega$, it holds that the first bit of $W(\sigma)^{(\mathtt{w},\mathtt{u})}$ equals $R'(\sigma)$.*

We note that compact goals that allow $\perp$-runs of a priori bounded length (as in Footnote 11) can be converted to (functionally equivalent) compact goals that allow no $\perp$-values (w.r.t R').[23]

By letting U' output the first bit it receives from the world (i.e., $U'(\sigma)$ equals the first bit of $\sigma^{(\mathtt{w},\mathtt{u})}$), we obtain a user-sensing function that is strongly safe with respect to any pair (U, S) and is strongly viable with respect to any (U, S) that robustly achieves the goal. Thus, we obtain:

Proposition 4.9 (trivial sensing implies guarded helpfulness): *Let $G = (\mathcal{W}, R)$ be a compact goal that allows trivial user-sensing, and $\mathcal{U}$ be a class of users. If a server strategy S is $\mathcal{U}$-helpful w.r.t G, then, for every class of server strategies $\mathcal{S}$, the strategy S is $\mathcal{S}$-guarded $\mathcal{U}$-helpful w.r.t G.*

By combining Proposition 4.9 and Theorem 4.6, we obtain

Theorem 4.10 (trivial sensing implies universality): *Let G and $\mathcal{U}$ be as in Proposition 4.9, and suppose that $\mathcal{U}$ is enumerable and that $\mathcal{S}$ is a class of server strategies that are $\mathcal{U}$-helpful w.r.t G. Then, there exists an $\mathcal{S}$-universal user strategy* (w.r.t G). *Furthermore, for every $S \in \mathcal{S}$, the complexity of the universal user strategy is upper-bounded by the complexity of some fixed strategy in $\mathcal{U}$.*

Proof: The sensing function U' that arises from Definition 4.8 satisfies Condition 1 of Theorem 4.6 (i.e., U' is fixed, $B = 1$, and the viability and safety conditions hold perfectly and in a very strong sense). Condition 2 of Theorem 4.6 holds by the extra hypothesis of the current theorem, which now follows by applying Theorem 4.6. ■

---

[22]Indeed, ($\mathcal{S}$-guarded) helpfulness here means ($\mathcal{S}$-guarded) $\mathcal{U}$-helpfulness for some (unspecified) class of user strategies $\mathcal{U}$.

[23]That is, for R' as in Definition 3.9, we assume here the existence of a function $B : \mathbb{N} \to \mathbb{N}$ such that $R(\overline{\sigma}) = 1$ only if for every $t > 0$ there exists $t' \in [t+1, t+B(\mathtt{s}(\sigma_1))]$ such that $R'(\sigma_{t'}) \neq \perp$. In such a case, the goal can be modified as follows. The states of the modified world will consist of pairs $(\sigma^{(\mathtt{W})}, i)$ such that $\sigma^{(\mathtt{W})}$ is the state of the original world and i indicates the number of successive $\perp$-values (w.r.t R') that preceded the current state. Thus, the index i is incremented if $R'(\sigma^{(\mathtt{W})}) = \perp$ and is reset to 0 otherwise. The modified temporal decision function evaluates to 1 on input $(\sigma^{(\mathtt{W})}, i)$ if and only if either $R'(\sigma^{(\mathtt{W})}) = 1$ or $i < B(\mathtt{s}(\sigma))$.

A variant on allowing trivial user-sensing. One natural case that essentially fits Definition 4.8 is of a *transparent world*, which intuitively corresponds to the case that the user sees the entire state of the environment. Formally, a transparent world is defined as a world that communicates its current state to the user (at the end of each round). Thus, ability to compute the corresponding temporal decision function R' puts us in the situation of a goal that allows trivial user-sensing. Consequently, analogously to Theorem 4.10, we conclude that

Theorem 4.11 (transparent world implies universality): *Let G be a compact goal with a transparent world. Suppose that $\mathcal{U}$ is an enumerable class of user strategies and that $\mathcal{S}$ is a class of server strategies that are $\mathcal{U}$-helpful w.r.t G. Then, there exists an $\mathcal{S}$-universal user strategy* (w.r.t G). *Furthermore, for every $S \in \mathcal{S}$, the complexity of the universal user strategy is upper-bounded by the complexity of some fixed strategy in $\mathcal{U}$ and the complexity of the temporal decision function R'.*

Beyond trivial user-sensing. Going beyond goals that allow trivial user-sensing, we note that a viable and safe user-sensing function may arise from the interaction between the user and the server (and without any feedback from the world). An instructive example of such a case, which can be traced to the first work of Juba and Sudan [9], is reformulated next using our terminology.

Example 4.12 (solving computational problems, revised): *In continuation to Example 3.7, we consider a multi-session goal that refers to a decision problem, D_0. In each session, the world selects non-deterministically a string and sends it to the user, which interacts with the server for several rounds, while signaling to the world that the session is still in progress. At some point, the user terminates the session by sending an adequate indication to the world, along with a bit that is supposed to indicate whether the initial string is in D_0, and the referee just checks whether or not this bit value is correct. Indeed, a simple two-round interaction with a server that decides D_0 yields a user-server pair that achieves this goal, where the user strategy amounts to forwarding messages between the world and the server. But what happens if a probabilistic polynomial-time user can interact with a server that decides D, where D is an arbitrary decision problem that is computationally equivalent to D_0? That is, we say that a server is a D-*solver *if it answers each user-message z with a bit indicating whether or not $z \in D$, and we ask whether we can efficiently solve D_0 when interacting with a D-solver for an arbitrary D that is computationally equivalent to D_0.*

- *Clearly, for every $D \in \mathcal{D}$, any D-solver is $\mathcal{U}$-helpful, where $\mathcal{D}$ denotes the class of decision problems that are computationally equivalent to D_0, and $\mathcal{U}$ denotes the class of probabilistic polynomial-time user strategies* (strategies that in each session run for a total time that is upper-bounded by a polynomial in the length of the initial message obtained from the world).[24] *Specifically, such a user may just employ the polynomial-time reduction of D_0 to D.*

- *More interestingly, as shown implicitly in [9], if D_0 has a program checker [4], then for every $D \in \mathcal{D}$ the D-solver is $\mathcal{F}$-guarded $\mathcal{U}$-helpful, where $\mathcal{F}$ is the class of all strategies and $\mathcal{U}$ is as above.*

  Showing that any such D-solver is $\mathcal{F}$-guarded $\mathcal{U}$-helpful amounts to constructing an adequate user strategy U along with a sensing function U' such that U attempts to answer the initial

[24] Indeed, our definition of $\mathcal{U}$ restricts both the complexity of the user strategy as a function and the number of rounds in which the user may participate in any session.

message obtained from the world by forwarding it to the server and verifies the correctness of the answer by running the program checker for D_0. Specifically, U emulates a potential program for D_0 by using the hypothetical D-solver via the reduction of D_0 to D. Note that this U' is viable with respect to U and the D-solver, and safe with respect to U and the class $\mathcal{F}$.

Recall that program checkers exist for PSPACE-complete and EXP-complete problems (cf. [11, 14] and [1], respectively).[25]

By invoking Theorem 4.6, we obtain an $\mathcal{S}$-universal user strategy, where $\mathcal{S}$ denote the class of all D-solvers for $D \in \mathcal{D}$. Furthermore, for every $S \in \mathcal{S}$, when interacting with S this universal strategy can be implemented in probabilistic polynomial-time.

Example 4.12 provides a rather generic class of goals that have $\mathcal{S}$-universal user strategies, where $\mathcal{S}$ is a class of "adequate solvers" (and furthermore these universal strategies are efficient). This class of multi-session goals refers to solving computational problems that have program checkers, and universality holds with respect to the class of servers that solve all computationally equivalent problems. We stress that the world strategies underlying these goals provides no feedback to the user, which indeed stands in sharp contrast to the goals that allow trivial user-sensing (of Definition 4.8).

We mention that the class of "adequate solvers" $\mathcal{S}$ considered in Example 4.12 is actually a strict subset of the class of all $\mathcal{U}$-helpful servers, where $\mathcal{U}$ is as in Example 4.12. Juba and Sudan [9] have actually established a stronger result, which can be reformulated as referring to the class of all servers that are helpful in a strong sense that refers to achieving the goal with a bounded number of errors. (Recall that a general helpful server may cause a finite number of sessions to fail, whereas the aforementioned solvers do allow achieving the goal without making any errors.) For details, see Section 4.4.2.

4.3 Universality without feedback

While Theorem 4.10 and Example 4.12 provide universal users based on user-sensing functions that rely on feedback either from the world or from the server (respectively), we note that universality may exist in a meaningful way also without any feedback. Below, we identify a class of goals for which this is possible.

Example 4.13 (multi-session "forgiving" communication goals): *For any function $f : \{0,1\}^* \to \{0,1\}^*$, we consider the multi-session goal in which each session consists of the world sending a message, denoted x, to the user and expecting to obtain from the server the message $f(x)$. That is, the world starts each session by selecting non-deterministically some string, x, and sending x to the user, and the session ends when the user notifies the world so. The session is considered successful if during it, the world has obtained from the server the message $f(x)$.* (Indeed, this notion of success is forgiving in the sense that it only requires that a specific message arrived during the session, and does not require that other messages did not arrive during the same session.) *The entire execution is considered successful if at most a finite number of sessions are not successful. Note that this goal is non-trivial* (i.e., it cannot be achieved when using a server that does nothing), *and yet it can be achieved by some coordinated user-server pairs* (e.g., a user that just forwards x to the server coupled with a server that applies f to the message it receives and forwards the result to the world).

[25] See also [2].

Proposition 4.14 (a universal strategy for Example 4.13): *Let $G = (\mathcal{W}, R)$ be a goal as in Example 4.13, and $\mathcal{U}$ an arbitrary enumerable class of user strategies. Let $\mathcal{S}$ be a class of server strategies such that for every $S \in \mathcal{S}$ there exists $U \in \mathcal{U}$ and an integer n such that in any execution of (U, S), starting at any state, all sessions, with possible exception of the first n ones, succeed. Then, there exists an $\mathcal{S}$-universal user strategy for G.*

Note that the hypothesis regarding $\mathcal{S}$ is stronger than requiring that every server in $\mathcal{S}$ be $\mathcal{U}$-helpful (which only means that for some $U \in \mathcal{U}$ the pair (U, S) robustly achieves the goal).[26]

Proof: For simplicity, we first assume that $n = 0$ (for all $S \in \mathcal{S}$). In this case, the universal strategy, denoted U, will emulate in each session a growing number of possible user strategies, and will notify the world that the session is completed only after completing all these emulations. We stress that in all these emulations we relay messages between the emulated user and the server, but we communicate with the world only at the beginning and end of each session. Specifically, in the i^{th} session, U emulates the first i strategies in the enumeration, denoted $U_1, ..., U_i$. For every $j = 1, ..., i$, we start the emulation of U_j by feeding U_j with the initial message obtained from the world in the current (i.e., i^{th}) session (as if this is the first session). (Thus, in the i^{th} real session we only emulate the first session of each of the U_j's.) When emulating U_j, for $j < i$, we use U_j's notification (to the world) that the session is over in order to switch to the emulation of the next strategy (i.e., U_{j+1}). When the emulation of U_i is completed (i.e., when U_i notifies the world that the session is over), we notify the world that the session is over.

Suppose that U interacts with the server $S \in \mathcal{S}$, and let j denote the index of a user strategy U_j such that (U_j, S) achieves the goal (in the strong sense postulated in the hypothesis). Then, for every $i \geq j$, considering the time $t_{i,j}$ in the i^{th} session in which we start emulating U_j, we note that the subsequent execution with S yields the adequate server message to the world, regardless of the state in which S was at time $t_{i,j}$. Thus, with a possible exception of the first $j - 1$ sessions, the pair (U, S) will be successful in all sessions, and hence (U, S) robustly achieves the goal.

We now turn to the general case, where n may not be zero (and may depend on $S \in \mathcal{S}$). In this case, we modify our emulation such that in the i^{th} real session we emulate each of the user strategies (i.e., $U_1, ..., U_i$) for i sessions (from each U_j's point of view), where we use the message we received in the real i^{th} session as the message sent to U_j in each of the emulated sessions. That is, let x_i denote the message that U receives from the world at the beginning of the i^{th} real session. Then, for $j = 1, ..., i$, the modified strategy U emulates i sessions of the interaction between U_j and the server (but, as in the case $n = 0$, does not notify the world of the end of the current session before all emulations are completed). Each of these i emulated sessions (in which U_j is used) starts with feeding U_j the message x_i (as if this were the message sent by the world in the currently emulated session).

For the modified strategy U and every $S \in \mathcal{S}$, with a possible exception of the first $\max(j-1, n)$ sessions, the pair (U, S) will be successful in all sessions, where j is as before and n is the bound guaranteed for S. ■

Digest. Proposition 4.14 asserts that there exists universal user strategies for (non-trivial) goals in which no feedback whatsoever is provided to the user. These goals, however, are very forgiving of failures; that is, they only require that during each session some success occurs, and they do

[26] This only means that for every $S \in \mathcal{S}$ there exists $U \in \mathcal{U}$ such that, in any execution of (U, S) starting at any state, there exists an integer n such that all sessions, with possible exception of the first n ones, succeed. In the hypothesis of Proposition 4.14, the order of quantification is reversed (from "for every execution there exists an n" to "there exists an n that fits all executions").

not require that there are no failures during the same session. Hence, we have seen three types of universal users. The first type exist for goals that allow trivial user-sensing (as in Definition 4.8), the second type rely on sensing through interaction with the server (as in Example 4.12 (following [9])), and the third type exists for multi-session goals that allow failures in each session (see Proposition 4.14).

4.4 Universality, revisited

In this section we present two refined versions of Theorem 4.6. The first one is merely a quantified version of the original, where the quantification is on the number of errors, which relies on the quality of the sensing functions in use. The second version introduces a more flexible universal user, which uses a relaxed notion of viability in which only the total number of negative indications (rather than the length of the time interval in which they occur) is bounded.

4.4.1 A quantified version (bounding the number of errors)

As stated in Section 4.1, the universal user strategy asserted in Theorem 4.6 does not benefit from the potential *strong safety* of user-sensing functions. The intuitive benefit in such user-sensing functions is that they may allow the universal strategy to switch earlier from a bad user strategy, thus incurring less errors. Indeed, this calls for a more refined measure of achieving goals, presented next.

Definition 4.15 (achieving goals (Definition 3.13), refined): *Let $G = (\mathcal{W}, R)$ be a compact goal and $R' : \Omega \to \{0, 1, \perp\}$ be as in Def. 3.9. For $B : \mathsf{N} \to \mathsf{N}$, we say that a pair of user-server strategies, (U, S),* achieves the goal G with B errors *if, for every $W \in \mathcal{W}$, a random execution $\overline{\sigma} = (\sigma_1, \sigma_2, ...)$ of the system (W, U, S) satisfies the following two conditions:*

1. *The expected cardinality of $\{t \in \mathsf{N} : R'(\sigma_t) = 0\}$ is at most $b \stackrel{\rm def}{=} B(\mathsf{s}(\sigma_1))$.*
2. *The expected cardinality of $\{t \in \mathsf{N} : (\forall t' \in [t, t+b])\ R'(\sigma_{t'}) = \perp\}$ is at most b.*

When B is understood from the context, we say that the execution $\overline{\sigma}$ contains an error in round t *if either $R'(\sigma_t) = 0$ or for every $t' \in [t, t + B(\mathsf{s}(\sigma_1))]$ it holds that $R'(\sigma_{t'}) = \perp$. If $\overline{\sigma}$ contains at most $B(\mathsf{s}(\sigma_1))$ errors, then we write $R_B(\overline{\sigma}) = 1$.*

Note that Definition 4.15 strengthens Definition 3.13, which (combined with Definition 3.9) only requires conditions analogous to the above where B may depend on the execution $(\sigma_1, \sigma_2, ...)$. Intuitively, whereas Definition 3.13 only requires that the number of errors in a random execution be finite, Definition 4.15 requires a bound on the number of errors such that this bound holds uniformly over all executions (as a function of the size of the initial state). A similar modification should be applied to the definition of robustly achieving a goal. Lastly, we refine the definition of strong sensing functions (i.e., Definition 3.18), by replacing all references to R by references to R_B (and specifying the relevant bound B in the terminology). (We also seize the opportunity and replace the fixed error-probability bound of $1/3$ by a general bound, denoted ϵ.)

Definition 4.16 (strong sensing (Definition 3.18), refined): *Let $G = (\mathcal{W}, R)$, S, U, U' and B be as in Def. 3.18. For $B : \mathsf{N} \to \mathsf{N}$ and $\epsilon : \mathsf{N} \to [0, 1/3]$, we say that U' is (B, ϵ)*-strongly safe *with respect to (U, S)* (and G) *if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, the following conditions hold.*

1. *If $R'(\sigma_1) = 0$, then, with probability at least $1 - \epsilon(\mathtt{s}(\sigma_1))$, either $R_B(\overline{\sigma}) = 1$ or for some $t \leq B(\mathtt{s}(\sigma_1))$ it holds that $U'(\sigma_t) = 0$, where $\overline{\sigma} = (\sigma_1, \sigma_2, ...,)$ denotes a random execution of the system (W, U, S).*

2. *If for every $i \in [B(\mathtt{s}(\sigma_1))]$ it holds that $R'(\sigma_i) = \perp$, then, with probability at least $1-\epsilon(\mathtt{s}(\sigma_1))$, either $R_B(\overline{\sigma}) = 1$ or for some $t \in [i+1, i+B(\mathtt{s}(\sigma_1))]$ it holds that $U'(\sigma_t) = 0$.*

*Analogously, U' is (B, ϵ)-*strongly viable *with respect to (U, S) if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, with probability at least $1 - \epsilon(\mathtt{s}(\sigma_1))$, for every $t \geq B(\mathtt{s}(\sigma_1))$ it holds that $U'(\sigma_t) = 1$. We say that strong viability* (resp., safety) *holds* perfectly *if $\epsilon \equiv 0$ holds in the viability* (resp., safety) *condition, and in such a case we say that U' is B-*strongly viable (resp., B-strongly safe).

Note that the existence of a B-strongly safe and viable sensing function w.r.t (U, S) (as in Definition 4.15) implies that (U, S) robustly achieves the goal with $2B$ errors (as in Definition 4.16). Intuitively, B errors result from the delay of the viability condition, and another B from the safety condition (i.e., the allowance to fail sensing if $R_B = 1$). If the sensing function is only $(B, 1/3)$-strongly safe and viable, then (U, S) robustly achieves the goal with $O(B)$ errors.

We comment that the foregoing definitions are simplified version of more appropriate definitions that we only sketch here. For starters, note that the bounding function B is used in Definition 4.15 in three different roles, which may be seperated: (1) bounding the expected number of errors of Type 1 (in Item 1), (2) bounding the expected number of errors of Type 2 (in Item 2), and (3) determining the length of $\perp$-runs that is considered an error of Type 3. Thus, R_B should be repalced by R_{B_1,B_2,B_3}, where B_1, B_2, B_3 are the three seperated bounding functions. In Definition 4.16, the bounding function B is used in six different roles: three roles are explicit in the two items analogously to the roles in Definition 4.15 and three implicit in the use of R_B (which should be replaced by R_{B_1,B_2,B_3}). Seperating all these bounding functions is conceptually right, since the various quantaties are fundamentally different. Still we refrained from doing so for sake of simplicity.[27]

With the foregoing definitions in place, we are ready to present a refined version of Theorem 4.6. The universal strategy postulated next achieves the goal with a bounded number of errors, where the bound depends on the bounds provided for the strong user-sensing functions.

Theorem 4.17 (universal strategies (Theorem 4.6), revisited): *Let $G = (\mathcal{W}, R)$ be a compact goal, $\mathcal{U}$ be an* enumerable *set of user strategies, $\mathcal{S}$ be a set of server strategies, and $\epsilon : \mathsf{N} \to [0, 1/3]$ such that the following two conditions hold:*

1. *For every $S \in \mathcal{S}$ there exists a user strategy $U \in \mathcal{U}$, a user sensing function U' and a bounding function B such that U' is (B, ϵ)-strongly viable with respect to (U, S) and (B, ϵ)-strongly safe with respect to U and the server class $\mathcal{S}$* (i.e., for every $\widetilde{S} \in \mathcal{S}$ it holds that U' is (B, ϵ)-strongly safe with respect to $(U, \widetilde{S})$ (and G))*. Furthermore, the mapping $U \mapsto (U', B)$ is computable.*

   Let $\mathcal{B}$ denote the set of bounds that appear in the image of this mapping; that is, $\mathcal{B} = \{B_i : i \in \mathsf{N}\}$, where B_i is the bound associated with the i^{th} user strategy in $\mathcal{U}$.

2. *One of the following two conditions hold.*

   (a) *The* (strong) *viability condition holds perfectly* (i.e., $\epsilon \equiv 0$).

   (b) *For every i, it holds that $B_{i+1} < B_i / 2\epsilon$.*

[27] Likewise, it is conceptually correct to replace R_B (and actually also R) in Definition 4.16 (resp., Definition 3.18) by a more strict condition that requires no errors at all after time B. Again, this was avoided only for sake of simplicity.

Then, there exists an $\mathcal{S}$-universal user strategy U such that for every $S \in \mathcal{S}$ there exists $B \in \mathcal{B}$ such that (U, S) robustly achieves the goal G with $O(B)$ errors, where the constant in the O-notation depends on S. Furthermore, if the (strong) *viability condition holds perfectly and the complexity of the enumeration is negligible* (when compared to the complexity of the strategies in $\mathcal{U}$), *then, for every $S \in \mathcal{S}$, the complexity of U is upper-bounded by the complexity of some fixed strategy in $\mathcal{U}$.*

Proof: Following the proof of Theorem 4.6, we first consider the case in which *both* the (strong) viability and safety conditions hold perfectly; that is, $\epsilon \equiv 0$ in (both the viability and safety conditions of) Definition 4.16. Recall that the universal user strategy U enumerates all $U_i \in \mathcal{U}$, and consider the corresponding pairs $(U_i^!, B_i)$, where $U_i^!$ is B_i-strongly safe (w.r.t U_i and $\mathcal{S}$). Specifically, suppose that U starts emulating U_i at round t_i, and denote the system's state at this round by σ_{t_i}. Then, *for the first $b_i \leftarrow B_i(\mathsf{s}(\sigma_{t_i}))$ rounds, strategy U just emulates U_i, and in any later round $t > t_i + b_i$ strategy U switches to U_{i+1} if and only if $U_i^!(\sigma_t) = 0$.*

Note that Claims 4.6.1 and 4.6.2 remain valid, since we maintained the construction of U. However, we seek a stronger version of Claim 4.6.2. Let us first restate Claim 4.6.1.

Claim 4.17.1 *Suppose that $U_i^!$ is B_i-strongly viable and safe with respect to (U_i, S), and consider a random execution of (W, U, S). Then, if this execution ever emulates U_i, then it never switches to U_{i+1}. Furthermore, in this case, for t_i and b_i as above, it holds that the number of errors* (w.r.t the bound b_i) *occuring after round t_i is at most $2b_i$.*

The furthermore part follows by observing that B_i-strong viability implies that for every $t \geq t_i + b_i$ it holds that $U_i^!(\sigma_t) = 1$, whereas the B_i-strong safety implies that the number of errors (w.r.t the bound b_i) occuring after round $t_i + b_i$ is at most b_i (because otherwise R_{B_i} evaluates to 0, and so $U_i^!(\sigma_t) = 0$ must holds for some $t > t' > t_i + b_i$, where t' is some time in which such a fault occurs).

Claim 4.17.2 *Let $i \geq 1$ and suppose that (U, S) does not robustly achieves the goal with $4\sum_{j\in[i]} B_j$ errors. Consider a random execution of (W, U, S), and, for $j \in \{i, i+1\}$, let t_j denote the round in which U started emulating U_j. Then, recalling that each $U_j^!$ is B_j-strongly safe* (w.r.t (U_j, S)), *the expected number of errors* (w.r.t the bound B_i) *that occur between round t_i and round t_{i+1} is at most $4b_i$ where $b_i \stackrel{\text{def}}{=} B_i(\mathsf{s}(\sigma_1))$. In particular,*

1. *The expected cardinality of $\{t \in [t_i, t_{i+1}] : R'(\sigma_t) = 0\}$ is at most $4b_i$.*
2. *The expected cardinality of $\{t \in [t_i, t_{i+1}] : (\forall t' \in [t, t + b_i])\ R'(\sigma_{t'}) = \bot\}$ is at most $4b_i$.*

Combining the foregoing two claims with the hypothesis that for every $S \in \mathcal{S}$ there exists a user strategy $U_i \in \mathcal{U}$ and a user-sensing function $U_i^!$ such that $U_i^!$ is B_i-strongly viable and safe with respect to (U_i, S), it follows that (U_i, S) robustly achieves the goal with B errors, where $B(s) = 2B_i(s) + 4\sum_{j\in[i-1]} B_j(s)$. Note that indeed $B(s) = O(B_j(s))$ for some $j \leq i$, where the constant in the O-notation depends on i (and hence on S).

Proof: We proceed by induction on i (using a vacuous base case of $i = 0$). Let σ_1 be a global state such that the expected number of errors produced by a random execution of the system starting at σ_1 exceeds $b = 4\sum_{j\in[i]} B_j(\mathsf{s}(\sigma_1))$ (i.e., either $|\{t \in \mathsf{N} : R'(\sigma_t) = 0\}| > b$ or $|\{t \in \mathsf{N} : (\forall t' \in [t, t+b])\ R'(\sigma_{t'}) = \bot\}| > b$). By the induction hypothesis, the expected number of errors that occur before round t_i is at most $4\sum_{j\in[i-1]} B_j(\mathsf{s}(\sigma_1))$, and some errors (w.r.t the bound b_i) occur after round $t_i + b_i$, where $b_i = B_i(\mathsf{s}(\sigma_1))$. That is, there exists $t > t_i + b_i$ such that either $R'(\sigma_t) = 0$ or for every $t' \in [t, t + b_i]$ it holds that $R'(\sigma_{t'}) = \bot$. In the first case the first (B_i-strong) safety condition (w.r.t $S \in \mathcal{S}$) implies that for some $t'' \in [t, t+b_i]$ it holds that $U_i^!(\sigma_{t''}) = 0$, whereas in the second case the second (B_i-strong) safety condition implies that for some $t'' \in [t'+1, t'+b_i] \subset [t+1, t+2b_i]$ it holds that $U_i^!(\sigma_{t''}) = 0$. In both cases, the fact that $U_i^!(\sigma_{t''}) = 0$ (for $t'' > t_i + b_i$) causes U to switch to emulating U_{i+1} at round $t''+1$ (if not before). Hence, if $t > t_i + b_i$ is set to the first round that contains an error (following round $t_i + b_i$), then the number of errors (w.r.t the bound b_i) during the emulation of U_i is at most $b_i + (t'' - t) \le 3b_i$. The claim follows. □

The foregoing analysis applies also in case the (strong) safety condition holds only with probability $1-\epsilon$, where $\epsilon = \epsilon(\mathsf{s}(\sigma_1))$, because there are many opportunities to switch from U_i, and each one is taken with probability at least $1-\epsilon$. More precisely, except for the first $b_i + 4\sum_{j\in[i-1]} B_j(\mathsf{s}(\sigma_1))$ errors, each error yields an opportunity to switch from U_i soon, and each such opportunity is accounted for by at most $2b_i$ errors. Thus, in addition to the $3b_i$ errors that occur when we have perfectly strong safety, we may incur $j \cdot 2b_i$ additional errors with probability at most ϵ^j, which gives an expected number of additional errors that is upper-bounded by $\sum_{j\in\mathsf{N}} \epsilon^j \cdot 2b_i j < 2b_i$. Hence, Claim 4.17.2 holds also in the general case, when replacing $4\sum_{j\in[i]} B_j$ by $6\sum_{j\in[i]} B_j$.

In contrast, in order to cope with imperfect (strong) viability (i.e., a strong viability condition that holds with probability $1-\epsilon$), we need to modify our strategy U. We use the same modification (i.e., "repeated enumeration") as at the end of the proof of Theorem 4.6. Since each additional repetition occurs with probability at most ϵ, the expected number of failures will remain bounded. Specifically, if U_i is repeated $r \ge 1$ additional times, then the expected number of errors is at most $\sum_{j\in[i+r]} 6B_j$, and so the expected number of errors is bounded by $\sum_{r\ge 0} \epsilon^r \cdot \sum_{j\in[i+r]} 6B_j$. Using the hypothesis $B_{j+1} < (2\epsilon)^{-1} \cdot B_j$, which implies $B_{i+r} < (2\epsilon)^{-r} \cdot B_i$, we upper-bound this sum by $12\sum_{j\in[i]} B_j$, and the main claim follows.

Regarding the furthermore claim, we note that the complexity of U is upper bounded by the maximum complexity of the strategies $U_1, ..., U_i$, where i is an index such that (U_i, S) robustly achieves the goal. Indeed, by an extra hypothesis, the complexity of the enumeration is dominated by the complexity of the emulation. ■

Theorem 4.17 versus Theorem 4.6. Indeed, Theorem 4.17 utilizes strongly safe user-sensing functions, whereas Theorem 4.6 only utilizes weakly safe user-sensing functions, but the conclusion of Theorem 4.17 is much more appealing: Theorem 4.17 provides an absolute (in terms of state size) upper bound on the number of errors incurred by the universal strategy, whereas Theorem 4.6 only asserts that each infinite execution of the universal strategy incurs finitely many errors. We stress that a user strategy that incurs (significantly) less errors should be preferred to one that incurs more errors. This is demonstrated next.

Example 4.18 (goals with delayed feedback): *Consider a goals G and classes of users and servers as in Theorem 4.17, and suppose that $\mathcal{B}$ is a class of moderately growing functions* (e.g., constant functions or polynomials). *Suppose that, for some huge function $\Delta : \mathsf{N} \to \mathsf{N}$* (e.g., an exponential function), *for every execution $\overline{\sigma}$ and every $t \in \mathsf{N}$, the user can obtain $R'(\sigma_t)$ at round $t + \Delta(\mathsf{s}(\sigma_t))$. This implies a very simple universal strategy via a simple adaptation of the principles underlying the proof of Theorem 4.10, but this strategy incurs $\Theta(\Delta)$ errors. In contrast, recall that the universal strategy provided by Theorem 4.17 incurs $O(B)$ errors, for some $B \in \mathcal{B}$.*

Refined helpfulness. The refined (or rather quantified) notion of achieving a goal suggests a natural refinement of the notion of helpful servers. This refinement is actually a restriction of the class of helpful servers, obtained by upper-bounding the number of errors caused by the server (when helping an adequate user). That is, for any bounding function $B : \mathsf{N} \to \mathsf{N}$, we may consider servers S that are not only $\mathcal{U}$-helpful but can rather be coupled with some $U \in \mathcal{U}$ such that (U, S) robustly achieves the goal with B errors. We say that such servers are $\mathcal{U}$-helpful with B errors.

4.4.2 Using relaxed viability

The latter notion of helpfulness with an explicitly bounded number of errors is not compatible with our current notion of bounded viability (cf. Definition 4.16). The point is that B-strong viability allows failure indications to occur only till time B, whereas helpfulness with B errors refers to the total number of errors. Wishing to utilize such helpful servers, we relax the notion of strong viability accordingly.

Definition 4.19 (a relaxed notion of strong viability): *Let $G = (\mathcal{W}, R)$, S, U, U' and B be as in Def. 3.18. For $B : \mathsf{N} \to \mathsf{N}$ and $\epsilon : \mathsf{N} \to [0, 1/3]$, we say that U' is (B, ϵ)-viable with respect to (U, S)* (and G) *if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Omega$, with probability at least $1 - \epsilon(\mathsf{s}(\sigma_1))$, the expected cardinality of $\{t \in \mathsf{N} : U'(\sigma_t) = 0\}$ is at most $B(\mathsf{s}(\sigma_1))$. If $\epsilon \equiv 0$, the we say that U' is B-viable.*

Theorem 4.20 (Theorem 4.17, revisited): *Let $G = (\mathcal{W}, R)$, $\mathcal{U}$, $\mathcal{S}$, ϵ and $\mathcal{B}$ be as in Theorem 4.17, except that each sensing function U'_i is (B_i, ϵ)-viable* (as per Definition 4.19) *rather than (B_i, ϵ)-strongly viable* (as per the viability condition in Definition 4.16). *Then, there exists an $\mathcal{S}$-universal user strategy U such that for every $S \in \mathcal{S}$ there exists $B \in \mathcal{B}$ such that (U, S) robustly achieves the goal G with $O(B^2)$ errors, where the constant in the O-notation depends on S. Furthermore, if B-viability holds* (i.e., the sensing function U'_i is $(B_i, 0)$-viable) *and the complexity of the enumerating $\mathcal{U}$ is negligible* (when compared to the complexity of the strategies in $\mathcal{U}$), *then, for every $S \in \mathcal{S}$, the complexity of U is upper-bounded by the complexity of some fixed strategy in $\mathcal{U}$.*

Proof Sketch: Following the proof of Theorem 4.17, we first consider the case in which *both* the viability and safety conditions hold perfectly (i.e.,, $\epsilon \equiv 0$ both in the viability condition of Definition 4.19 and in the safety condition of Definition 4.16). We modify the universal user strategy U used in the proofs of Theorems 4.6 and 4.17 such that it switches to the next strategy after seeing sufficiently many failure indications (rather than when seeing a failure indication after sufficiently much time). Specifically, suppose that U starts emulating U_i at round t_i, and denote the system's state at this round by σ_{t_i}. Then, *strategy U emulates U_i till it encounters more than $b_i \leftarrow B_i(\mathsf{s}(\sigma_{t_i}))$ rounds $t > t_i$ such that $U'_i(\sigma_t) = 0$ holds, and switches to U_{i+1} once it encounters the $b_i + 1^{\text{st}}$ such round.*

We shall show that Claims 4.17.1 and 4.17.2 remain essentially valid, subject to some quantitative modifications. Specifically, Claim 4.17.1 is modified as follows.

Claim 4.20.1 *Suppose that U'_i is B_i-viable and B_i-strongly safe with respect to (U_i, S), and consider a random execution of (W, U, S). Then, if this execution ever emulates U_i, then it never switches to U_{i+1}. Furthermore, in this case, for t_i and b_i as above, it holds that the number of errors* (w.r.t the bound b_i) *occuring after round t_i is at most $b_i + b_i^2$.*

The furthermore part follows by observing that B_i-viability implies that $|\{t > t_i : U'(\sigma_t) = 0\}| \leq b_i$, whereas the B_i-strong safety implies that if more than b_i errors occur after round $t' > t_i$, then $U'_i(\sigma_{t''}) = 0$ must holds for some $t'' \in [t', t' + b_i]$ (since in this case R_{B_i} evaluates to 0). Thus, if errors appear at rounds $t'_1, ..., t'_m > t_i$ such that $t'_1 < t'_2 < \cdots < t'_m$, then failure indications must occur in rounds $t''_1, t''_2, ..., t''_{m-b_i} > t_i$ such that $t''_j \in [t'_j, t'_j + b_i]$ (for every $j \in [m - b_i]$). Since at most b_i of these intervals may have a non-empty intersection, it follows that $(m - b_i)/b_i \leq b_i$. Thus, Claim 4.20.1 follows. As for Claim 4.17.1, it is modified as follows.

Claim 4.20.2 *Let $i \geq 1$ and suppose that (U,S) does not robustly achieve the goal with $3\sum_{j\in[i]} B_j^2$ errors. Consider a random execution of (W,U,S), and, for $j \in \{i,i+1\}$, let t_j denote the round in which U started emulating U_j. Then, recalling that each U_j' is B_j-strongly safe* (w.r.t (U_j,S), *the expected number of errors* (w.r.t the bound B_i) *that occur between round t_i and round t_{i+1} is at most $3B_i^2(\mathsf{s}(\sigma_1))$.*

Combining Claims 4.20.1 and 4.20.2 with the hypothesis that for every $S \in \mathcal{S}$ there exists a user strategy $U_i \in \mathcal{U}$ and a user-sensing function U_i' such that U_i' is B_i-viable and B_i-strongly safe with respect to (U_i,S), it follows that (U_i,S) robustly achieves the goal with B errors, where $B(s) = 2B_i^2(s) + 3\sum_{j\in[i-1]} B_j^2(s)$. Note that indeed $B(s) = O(B_j^2(s))$ for some $j \leq i$, where the constant in the O-notation depends on i (and hence on S).

Proof sketch: Following the proof of Claim 4.17.2, we proceed by induction on i. Let σ_1 be a global state such that the expected number of errors produced by a random execution of the system starting at σ_1 exceeds $3\sum_{j\in[i]} B_j^2(\mathsf{s}(\sigma_1))$. By the induction hypothesis, the expected number of errors that occur before round t_i is at most $3\sum_{j\in[i-1]} B_j^2(\mathsf{s}(\sigma_1))$, and so at least $3b_i^2$ errors occur after round t_i, where $b_i = B_i(\mathsf{s}(\sigma_1))$. The first $(b_i+1)b_i$ errors must (by B_i-strong safety) cause more than b_i failure indications (i.e., rounds $t > t_i$ such that $U'(\sigma_t) = 0$), which causes U to switch to emulating U_{i+1} as soon as $b_i + 1$ such indications are encountered, which occurs at most another b_i rounds after the last detected error (again by B_i-strong safety). Hence, the number of errors (w.r.t the bound b_i) during the emulation of U_i is at most $3b_i^2$, and the claim follows. □

As in the proof of Theorem 4.17, we need to extend the analysis to the general case in which $\epsilon \leq 1/3$ (rather than $\epsilon = 0$). The extension is analogous to the original one, where here each repetition causes an overhead of $O(B^2)$ (rather than $O(B)$) errors. ■

Example 4.21 (solving computational problems, revised again): *We consider the same goal as in Example 4.12, but here we consider the possibility of achieving this goal when interacting with an arbitrary server that is $\mathcal{U}$-helpful with a polynomially bounded number of errors* (rather than interacting with an arbitrary D-solver). *Recall that we consider the multi-session goal of solving instances* (selected non-deterministically by the world) *of a decision problem, D_0, and $\mathcal{U}$ denotes the class of probabilistic polynomial-time user strategies. This is a multi-session version of the goal studied by Juba and Sudan [9], and their solution can be nicely cast in the current framework. Specifically:*

- *As shown implicitly in [9], if both D_0 and its complement have interactive proof systems* (cf. [7]) *in which the designated prover strategy can be implemented by a probabilistic polynomial-time oracle machine with access to D_0 itself, then, for some polynomial B, there exists a sensing function that is B-viable and $(B, 1/3)$-strongly safe with respect to the class of $\mathcal{U}$-helpful with B errors servers.*

  The proof of the foregoing claim amounts to the user invoking the interactive proof system, while playing the role of the verifier and using the helpful server in order to implement the designated prover strategy. For details, see [9].

  Recall that adequate interactive proof systems exists for PSPACE-complete and some problems in $\mathcal{SZK}$ that are believed not to be in $\mathcal{P}$ (cf. [11, 14] and [6], respectively).[28]

[28] Recall that we need interactive proof systems in which the designated prover strategy is relatively efficient in the sense that it can be implemented by a probabilistic polynomial-time oracle machine with access to the problem itself. Such interactive proof systems are known, e.g., for Graph Isomorphism problem [6], but it seems unlikely that *all* problems in $\mathcal{IP}$ (or even $\mathcal{NP}$) have such proof systems [3].

- *By invoking Theorem 4.20 we obtain an $\mathcal{S}$-universal user strategy, where $\mathcal{S}$ denote the class of all $\mathcal{U}$-helpful servers. Furthermore, for every $S \in \mathcal{S}$, when interacting with S this universal strategy can be implemented in probabilistic polynomial-time.*

Analogous reasoning can be applied to other classes of user strategies; for example log-space implementable strategies. Details will appear in a future version.

4.5 On the limitations of universal users and related issues

In this section we justify some of the limitations of the positive results presented in prior sections. Specifically, we address the overhead in Theorem 4.17 and the fact that the strong sensing functions of Example 4.12 are also safe with respect to non-helpful servers.

4.5.1 On the overhead in Theorem 4.17

Recall that the number of errors incurred by the universal user asserted in Theorem 4.17 (as well as in Theorem 4.6) is at least linear in the index of the server that it happens to use (with respect to a fixed ordering of all servers in the class). Thus, the number of errors is exponential in the length of the description of this server (i.e., the length of its index). We shall show that this overhead (w.r.t a user tailored for this server) is inherent whenever the universal user has to achieve any non-trivial goal with respect to a sufficiently rich class of servers.

Loosely speaking, a goal is nontrivial if it cannot be achieved without the help of some server. Since our basic framework always includes a server, we model the absence of a "real" server by referring to the notion of a *trivial server* (i.e., a server that sends empty messages in each round).

Definition 4.22 (nontrivial goals): *Let T denote a server, called* trivial*, that sends empty messages in each round. We say that a compact goal $G = (\mathcal{W}, R)$ is* nontrivial *w.r.t. a class of users $\mathcal{U}$ if for every user $U \in \mathcal{U}$ there is a $W \in \mathcal{W}$ such that the temporal decision function R' never outputs 1 in the execution (W, U, T).*

Note that the notion of nontrivial is more restricted than the requirement that (U, T) does not achieve the goal. Nevertheless, the stronger requirement, which asserts that the temporal decision function R' never rules that the execution is tentatively successful, is very natural.

As for the class of "sufficiently rich" class of servers, we consider here one such possible class (or rather a type of classes). Specifically, we consider servers that become helpful (actually stop sending empty messages) only as soon as they receive a message from the user that fits their password. Such "password protected" servers are quite natural in a variety of settings. Actually, for sake of robustness (both intuitive and technical)[29] we postulate that the password be check at every round (rather than only in the first round). That is, in each round, the server will check that the message received is prepended with a string that matches its password.

Definition 4.23 (password-protected servers and password closure): *For every server strategy S and string $x \in \{0,1\}^*$, the* password-protected version of S with password x (x-protected version of S)*, denoted S^x, is the server strategy that upon receiving a message of the form xy, updates its*

[29] In order for user strategies to robustly achieve goals with password-protected servers, the user must be ready to provide the password when started from any arbitrarily chosen state (as required by Definition 3.14). The most straightforward and natural way to ensure this is for the user to send the password on every message to the server. Thus, a natural type of password-protected servers that permits users to robustly achieve their goals consists of servers that expect all messages to be prepended by their password.

state and sends messages as S would upon receiving y. Otherwise, S^x sends the empty messages to all parties, like the trivial server would, and does not update the state.

Roughly speaking, the reason password-protected servers demonstrate the need for substantial overhead is that, when the user does not know the password, the user has no choice but to try all possible passwords, which implies a lower bound on the number of errors. For this demonstration (of overhead) to be meaningful, we should show that password-protected versions of helpful servers are essentially as helpful as their unprotected counterparts. Indeed, for starters, we establish the latter claim, where this holds with respect to classes of user strategies that are closed under a simple transformation (i.e., prepending of adequate passwords).

Proposition 4.24 (password-protected versions of helpful servers are helpful): *Let $\mathcal{U}$ be a class of user strategies such that, for any $U \in \mathcal{U}$ and any string $x \in \{0,1\}^*$, there there exists a strategy $U^x \in \mathcal{U}$ that acts as U except that it appends x to the beginning of each message that it sends to the server. Then, for every $\mathcal{U}$-helpful server S and every password $x \in \{0,1\}^*$, the x-protected version of S, denoted S^x, is $\mathcal{U}$-helpful. Furthermore, if (U,S)* (robustly) *achieves the goal, then (U^x,S^x)* (robustly) *achieves the goal with the same number of errors as (U,S).*

Proof: Then, since S is $\mathcal{U}$-helpful, there exists $U \in \mathcal{U}$ such that (U,S) robustly achieves the goal. Since (U^x,S^x) send the same messages to the world as (U,S), it holds that (U^x,S^x) also robustly achieves the goal and incurs precisely the same number of errors as (U,S). Since $U^x \in \mathcal{U}$, it follows that S^x is $\mathcal{U}$-helpful. ■

Having established the helpfulness of password-protected versions (of helpful servers), we prove a lower bound on the number of errors incurred when achieving (nontrivial) goals by interacting with such servers.

Theorem 4.25 (on the overhead of achieving nontrivial goals with password-protected servers): *Let $G = (\mathcal{W},R)$ be a nontrivial compact goal and S be helpful with respect to G. Then, for every user U and integer ℓ, there exists an ℓ-bit string x such that (U,S^x) does not achieve G in less than $2^{(\ell-3)/2}$ errors, where S^x denotes the x-protected version of S.*

Note that the fact that the lower bound has the form $\Omega(2^{\ell/2})$ (rather than $\Omega(2^{\ell})$) is due to the definition of errors (cf. Definition 4.15).[30]

Proof: Let any user strategy U be given and let T be a trivial server. Since G is nontrivial, there exists $W \in \mathcal{W}$ such that the temporal decision function R' never evaluates to 1 in a random execution of (W,U,T). For starters, we assume (for simplicity) that in such random executions R' always evaluates to 0. Consider, a random execution of (W,U,S^x), when x is uniformly selected in $\{0,1\}^\ell$, Then, with probability at least $1 - m \cdot 2^{-\ell}$, the user U did not prepend the string x to any of the messages it sent in the first m rounds. In this case, the m-round execution prefix of (W,U,S^x) is distributed identically to the m-round execution prefix of (W,U,T), which means that it generates m errors. Using $m = 2^{\ell-1}$ it follows that, for a uniformly selected $x \in \{0,1\}^\ell$, the expected number of errors in a random execution of (W,U,S^x) is at least $2^{\ell-2}$. Hence, there exists a string $x \in \{0,1\}^\ell$ such that (U,S^x) does not achieve G in less than $2^{\ell-2}$ errors.

[30] Indeed, also the trivial server that prevents R' from ever evaluating to 1 may be viewed by Definition 4.15 as making only $\sqrt{2^\ell}$ errors (for some adequate R'). In particular, we may consider the following behavior of R' for the case that the server never sends a message to the world. For every $i = 1,2,...,$ and $j \in [2^{2i-2}, 2^{2i}]$, in round j the value of R' equals 0 if j is a multiple of 2^i and equals $\perp$ otherwise. Then, for every even ℓ, the first 2^ℓ rounds contain no $2^{\ell/2}$-long run of $\perp$, whereas the total number of zeros in these rounds is $\sum_{i=1}^{\ell/2} 2^i = O(2^{\ell/2})$.

In the general case (i.e., when considering $\perp$-values for R'), we may infer that there exists a string $x \in \{0,1\}^\ell$ such that, with probability at least $1 - m \cdot 2^{-\ell}$, the temporal decision function R' does not evaluate to 1 in the first m rounds of a random execution of (W, U, S^x). In this case, this execution prefix contains at least $\sqrt{m}$ errors (see the two items of Definition 4.15), and the theorem follows (by setting $m = 2^{\ell-1}$). ■

Discussion. Combining Theorem 4.25 and Proposition 4.24, we demonstrate the necessity of the error overhead incurred by the universal strategy of Theorem 4.17. Specifically, the latter strategy must work for server and user classes that are derived via Proposition 4.24. Now, Theorem 4.25 asserts that this class of 2^ℓ servers contains a server that causes an overhead that is exponential in ℓ, which in turn is closely related to the length of the description of most servers in this class.

4.5.2 On the requirements of strong sensing functions

Recall that the existence of a $\mathcal{S}$-universal strategy implies the existence of a sensing function that is safe with respect to $\mathcal{S}$ (see Proposition 4.5). However, this sensing function is trivial (i.e., it is identically 1), and its safety with respect to $\mathcal{S}$ just follows from the fact that the $\mathcal{S}$-universal strategy achieves the goal when coupled with any server in $\mathcal{S}$. Clearly, this safety property may no longer hold with respect to servers outside $\mathcal{S}$, and specifically with respect to servers that are not helpful at all. We believe that sensing functions that are safe also with respect to a wider class of servers are desirable. Also, it is desirable to have sensing functions that are strongly safe, because such functions offer bounds on the number of errors made by the universal strategy (see Theorem 4.17).

Turning to the cases in which we designed nontrivial strong sensing functions – i.e., those in Example 4.12 – we observe that these sensing functions were actually safe with respect to any server. We show now that this is no coincidence: It turns out that a strong sensing function with respect to a sufficiently rich class of helpful servers is actually safe with respect to any server. In other words, if U' is safe with respect to $\mathcal{S}$, which may contain only $\mathcal{U}$-helpful servers, then U' is safe with respect to any server (including servers that are not helpful to any user).

As we observed in our discussion of password-protected servers, an individual password-protected server poses no particular challenge to the right user strategy. The "challenge" of password-protection only arises when the user is faced with an entire *class* of servers, which use different possible passwords; the user is then forced to search through all possible strings until it hits upon the right one. The phenomenon that forces strong sensing functions to guard against all servers (including totally unhelpful ones) is similar. Considering a class of helpful servers that are each helpful when they communicate with users that send sufficiently long messages and may behave arbitrarily otherwise, we show that (strong) safety with respect to this class implies (strong) safety with respect to all servers. Specifically, for each user strategy U, we will consider the class $\mathtt{pad}(U)$ of all user strategies that prepend messages of U by a sufficiently long prefix, and show that (strong) safety with respect to the class of all $\mathtt{pad}(U)$-helpful servers implies (strong) safety with respect to all servers.

Theorem 4.26 (strong safety w.r.t helpful servers implies same w.r.t all server): *Let $G = (\mathcal{W}, R)$ be a compact goal, which is achievable by the pair (U, S). Let $\mathtt{pad}_i(U)$ denote a user strategy that prepends 0^i to each message sent by U, and suppose that U' is (B, ϵ)-strongly safe with respect to U and each $\{\mathtt{pad}_i(U) : i \in \mathsf{N}\}$-helpful server* (and G). *Then, U' is $(B, 2\epsilon)$-strongly safe with respect to the user U and every server* (and G).

Proof: Suppose, towards the contrary, that there exists an arbitrary server S such that U' is not $(B, 2\epsilon)$-strongly safe with respect to (U, S) and G. The strong safety property implies that the sensing failure of U' is witnessed by finite prefixes of the relevant executions. Specifically, for some $W \in \mathcal{W}$ and some initial state σ_1, with probability greater than 2ϵ, a random execution of (W, U, S) starting at σ_1 contains a finite prefix that witnesses the sensing failure. Recall that there are two cases depending on whether $R'(\sigma_1) = 0$ or $R'(\sigma_1) = \perp$.

Starting with the first case, we note that with probability greater than $2\epsilon(\mathtt{s}(\sigma_1))$, the random execution $\overline{\sigma}$ is such that $U'(\sigma_i) = 1$ for all $i \leq B(\mathtt{s}(\sigma_1))$ and $R_B(\sigma) = 0$. Note that the first event depends only on the B-long prefix of σ, denoted $\sigma_{[1,B]}$. Thus, with probability at least 2ϵ, this prefix is such that (1) U' is identically 1 on all its states, and (2) with positive probability this prefix is extended to a random execution that is unsuccessful (per R_B). Fixing any such prefix, we note that event (2) is also witnessed by a finite prefix; that is, with positive probability, a random extension of this prefix contains a (longer) prefix that witnesses the violation of R_B. Using the fact that the latter event refers to a countable union of fixed prefix events, we conclude that there exists $\ell \in \mathsf{N}$ such that with positive probability the said violation is seen in the ℓ-step prefix. Furthermore, by viewing the probability of the former event as a limit of the latter events, we can make the probability bound retain 90% of its original value. The same process can be applied across the various B-long prefixes, and so we conclude that there exists an $\ell \in \mathsf{N}$ such that, with probability at least 1.5ϵ, a violation is due to the ℓ-long prefix of a random execution. Similar considerations apply also to the second aforementioned case (where $R'(\sigma_1) = \perp$).

Next, we note that we can upper bound the length of the messages that are sent by U in the first ℓ steps of most of these random executions. That is, there exists an $i \in \mathsf{N}$ such that, with probability at least ϵ, the sensing function U' fails in a random ℓ-step execution prefix during which U sends messages of length at most i. At this point we are ready to define a helpful server that also fails this sensing function.

Firstly, we consider the strategy $\widetilde{u} = \mathtt{pad}_{i+1}(U)$, and define a strategy $\widetilde{s} = \mathtt{upad}_{i+1}(S)$, where $\mathtt{upad}_j(S)$ denote a server strategy that omits the first j bits of each incoming message (from the user) and feeds the result to S. Clearly, $(\widetilde{u}, \widetilde{s})$ achieves the goal G, and so $\widetilde{s}$ is $\mathtt{pad}(U)$-helpful. On the other hand, by the foregoing argument, it is the case that U' fails with probability at least ϵ in a random execution of $(W, U, \widetilde{s})$. Thus, U' is not (B, ϵ)-strongly safe with respect to U and $\widetilde{s}$ (and G), which contradicts our hypothesis regarding safety with respect to all helpful servers (or rather all $\{\mathtt{pad}_j(U) : j \in \mathsf{N}\}$-helpful servers). The theorem follows. ■

References

[1] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, Vol. 1, No. 1, pages 3–40, 1991.

[2] L. Babai, L. Fortnow, N. Nisan and A. Wigderson. BPP has Subexponential Time Simulations unless EXPTIME has Publishable Proofs. *Complexity Theory*, Vol. 3, pages 307–318, 1993.

[3] M. Bellare and S. Goldwasser. The Complexity of Decision Versus Search. *SICOMP*, Vol. 23, No. 1, pages 91–119, 1994.

[4] M. Blum and S. Kannan. Designing Programs that Check their Work. In *21st STOC*, pages 86–97, 1989.

[5] J. Dewey. *Experience and Nature*. Norton, New York, 1929. *First edition*, 1925.

[6] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38, No. 3, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[7] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SICOMP*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[8] M. Hutter. *Universal Artificial Intelligence*. Springer, Berlin, 2004.

[9] B. Juba and M. Sudan Universal Semantic Communication I. In *40th STOC*, pages 123–132, 2008.

[10] B. Juba and M. Sudan Universal Semantic Communication II: A Theory of Goal-Oriented Communication. *ECCC*, TR08-095, October 2008.

[11] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *JACM*, Vol. 39, No. 4, pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.

[12] B. Malinowski. The Problem of Meaning in Primitive Languages. In *The Meaning of Meaning*, C. K. Ogden and I. A. Richards. Harcourt Brace Jovanovich, New York, 1923.

[13] S. Russell and D. Subramanian. Provably Bounded-Optimal Agents. *JAIR*, Vol. 2, pages 575–609, 1995. Preliminary version with R. Parr in *13th IJCAI*, 1993.

[14] A. Shamir. IP = PSPACE. *JACM*, Vol. 39, No. 4, pages 869–877, 1992. Preliminary version in *31st FOCS*, 1990.

[15] C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, Vol. 27, pages 379–423, 623–656, 1948.

[16] L. Wittgenstein. *The Blue and Brown Books: Preliminary Studies for the 'Philosophical Investigations'*. Harper & Row, New York, 1958.

[17] L. Wittgenstein. *Philosophical Investigations*. Basil Blackwell, Malden, 2001. *First edition*, 1953.

Appendix: On the measurability of various sets of executions

In general (i.e., for a general refree that is not compact), the set of successful executions may not be measurable (with respect to the natural probability measure that assigns each prefix of a random execution a measure that corresponds to the probability that they occur). This follows from the fact that an arbitrary referee gives rise to an arbitrary subset of the set of all executions, whereas the set of executions is isomoprphic to the set of real numbers. The compactness condition imposes a structure on the set of successful executions, and thus guarantees that this set is measurable (with respect to the natural probability measure).

Recall that a probability measure is defined with respect to a sigma-algebra that contains the sets of interest , which in our case is the set of successful executions (as well as other related sets). A sigma-algebra is a pair (X, Σ), where X is a set and $\Sigma \subseteq 2^X$, such that $\Sigma \neq \emptyset$ is closed under complementation and countable unions (i.e., $S \in \Sigma$ implies $X \setminus S \in \Sigma$ and $S_1, S_2, ... \in \Sigma$ implies $\cup_{i\in\mathsf{N}} S_i \in \Sigma$). The natural probability measure arises from a sigma-algebra that corresponds to all execution prefixes.

Definition A.1 (the natural probability measure of executions): *For a system (W, U, S), we consider the sigma-algebra (X, Σ) such that X is the set of all possible executions of the system (W, U, S) and Σ equals the closure of the family of sets $\{E_{(i,\sigma)} : i \in \mathsf{N}, \sigma \in \Omega\}$ under complementation and countable union, where $E_{(i,\sigma)}$ denotes the set of executions $\overline{\sigma} = (\sigma_1, \sigma_2 ...)$ such that $\sigma_i = \sigma$. The* natural probability measure of executions, *denoted μ, is obtained by assigning each prefix of a random execution a measure that corresponds to the probability that it occurs.*

Note that the mapping μ is indeed a measure for the foregoing sigma-algebra Σ, because it is (1) non-negative, (2) assigns zero to the empty set, and (3) satisfies sigma-additivity (i.e., for any countable collection of pairwise disjoint sets $S_1, S_2, ... \in \Sigma$ it holds that $\mu(\cup_{i\in\mathsf{N}} S_i) = \sum_{i\in\mathsf{N}} \mu(S_i)$). As we shall see, for compact referees, the set of successful executions can be expressed as a countable union of sets in Σ.

Proposition A.2 *For any compact referee R, the set of successful executions is measurable with respect to the natural probability measure of executions.*

Proof: Let R' be the temporal decision function associated with R (by the compactness hypothesis), and assume for simplicity that R' never assumes the value $\perp$. In this case, the set of successful executions is a countable union of the sets S_t, where S_t is the set of executions in which no failures occur after time t (i.e., $\overline{\sigma} \in S_t$ if for every $i > t$ it holds that $R'(\sigma_i) = 1$). On the other hand, S_t is a countable intersection of the sets $E_{(i,\sigma)}$ such that $R'(\sigma) = 1$ (and $i \in \mathsf{N}$).

To handle the case that R' may assume the value $\perp$, we show that the set of executions containing no infinite runs of $\perp$ is measurable. The latter set is a countable union of the sets S'_t, where S_t is the set of executions of executions having $\perp$-runs of length at most t. On the other hand, the complement of each such set is a countable intersection of the sets $\cap_{j=0}^{t} E_{(i+j,\sigma)}$ such that $R'(\sigma) = \perp$ (and $i \in \mathsf{N}$). ■