Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 75 (2009)

A Theory of Goal-Oriented Communication[*]

Oded Goldreich Brendan Juba Madhu Sudan

December 18, 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. Focusing on goals provides a framework for addressing the problem of potential "misunderstanding" during communication, where the misunderstanding arises from lack of initial agreement on what protocol and/or language is being used in communication. In this context, "reliable communication" means overcoming any initial misunderstanding between parties towards achieving a given goal. Despite the enormous diversity among the goals of communication, we propose a simple model that captures *all* goals.

In the simplest form of communication we consider, two parties, a *user* and a *server*, attempt to communicate with each other in order to achieve some goal of the user. We show that *any goal* of communication can be modeled mathematically by introducing a third party, which we call the *referee*, who hypothetically monitors the conversation between the user and the server and determines whether or not the goal has been achieved. Potential *misunderstanding* between the players is captured by allowing each player (the user/server) to come from a (potentially infinite) class of players such that each player is unaware which instantiation of the other it is talking to. We identify a main concept, which we call *sensing*, that allows goals to be achieved even under misunderstanding. Informally, *sensing* captures the user's ability (potentially using help from the server) to simulate the referee's assessment on whether the communication is achieving the goal. We show that when the user can sense progress, the goal of communication can be achieved despite initial misunderstanding. We also show that in certain settings sensing is necessary for overcoming such initial misunderstanding.

Our results significantly extend the scope of the investigation started by Juba and Sudan (STOC 2008) who studied the foregoing phenomenon in the case of a *single* specific goal. Our study shows that their main suggestion, that misunderstanding can be detected and possibly corrected by focusing on the goal, can be proved in full generality.

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

ISSN 1433-8092

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 the meaning of the intended communication 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 communication 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 1.4), we associate the meaning of communication with the goal achieved by it.

1.1 Our work at a glance

Our general contribution is in suggesting a mathematical theory of goal-oriented communication. We note that this line was initiated in prior work of Juba and Sudan [9] who considered one *specific goal* of communication and showed how misunderstandings could be overcome in their setting to achieve the goal. In this work we vastly generalize their work by addressing completely *generic goals* of communication. At this level of generality it is not possible to expect misunderstandings to be overcome in every instance. Indeed, the focus of our work is on when can misunderstanding be overcome and how. A central contribution of this work is in identifying and highlighting a concept, which we call *sensing*, that suffices for resolving misunderstandings (and seems effectively necessary for such resolution). The concept of sensing thus provides a design principle, derived from a theoretical study, for communication systems; that is, a well-designed system should enable the user to sense whether or not progress is made towards its goal. Following is an outline of the

[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 foregoing printing example, the goal of the computer is to have the image printed properly; while in the computational example, the goal of the weak computer is to obtain a correct answer to the computational task that it has delegated.

main conceptual steps in this work.

The main setting we focus on involves the interaction between two entities, a *user* and a *server*, where the user is trying to achieve some goal and the server is trying to help the user. (The treatment of the general case, in which each of the communicating parties may have its own goal, is postponed to Section 7; interestingly, the treament of this general case can be reduced to the treatment of the asymmertic (user–server) model.)

Misunderstanding in this setting is modeled by postulating that the user is selected arbitrarily from a *class* of potential users, and similarly the server is selected arbitrarily from a *class* of potential servers. In principle the class may contain all the "linguistic cousins" of a single user (or server). The user does not know which server from the class it is talking to and vice versa, leading to potential misunderstandings between them.

Our first main contribution is the modeling of a *generic goal*. Note that this is not trivial since neither the user nor the server is fixed (and hence neither is the meaning of the bits flowing along the communication channel between them). So how can one model a fixed goal in the middle of all this variability? We do so by introducing a *third entity* that we refer to as the *referee/environment*, who monitors the conversation between the user and the server by obtaining information that the user and server choose to pass to it, and using this information determines whether or not the interaction is achieving the goal.[3] Thus, each such referee models a (potentially) different goal, and conversely we postulate (as a thesis) that every goal of communication can be modeled by such a referee, along with an appropriate choice of a class of users and servers.

The presence of this third entity also provides a simple way to model the "environment" in which the communicating players interact. In particular, it allows us to model any other shared resources or correlated signals that the user and server share. In what follows, we use either of the phrases "referee" or "environment" to describe this third entity, where the usage is supposed to reflect the role being played by the entity (whether it is testing the communication, or just part of the modeling of the world).

Armed with this formal definition of generic goals we proceed to analyze the conditions under which misunderstandings can be *detected* and *resolved*. To this end we propose the notion of *sensing*, a concept that captures the ability of the *user* to sense progress towards achieving the goal. The user may be *un*able to sense progress due to the unavailability of information stored in the environment (e.g., some messages from the server), or due to its computational resources (which may be less than those of the environment). However if (and roughly only if) the user is able to sense such progress, or more precisely the lack of progress, then it can *detect* misunderstanding.

Thus, the *goal* specifies a (communication) problem, while the *sensing* function is a key ingredient in solving such problems. Indeed, as in many other cases, detecting trouble is a first step towards resolving it.

We then show that *sensing* also suffices to yield methods for *correcting* misunderstandings or at least achieving the goals with respect to the class of all helpful servers, where a server is called *helpful* if *some* user may achieve the goal when communicating with it. Specifically, one of our results asserts that if the user can sense whether it is making progress towards the desired goal (when communicating with any server), then it can 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

[3]Note that since the referee itself is fixed, there is no issue of misunderstanding with respect to communicating with it.

(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).

Our universal strategy, which is based on sensing, enumerates all possible "interpretations" of what the server may be doing, and rules out incorrect interpretations (by using sensing to detect misunderstandings).[4] Hence, our solution is essentially just "try and check" (e.g., in the printer case, we try all possible formats and rely on the ability to check whether the printer responds at all and what it prints). Indeed, this confirms a very natural and often used human paradigm.

One may argue that the "try and check" paradigm is straightforward and that so is the aforementioned enumeratation. Our point, however, is that once one abstracts the concept of sensing, tools (possibly several tools) become available to tackle the problem of resolving misunderstanding towards achieving given goals. Indeed, a straightforward enumeration is one tool, and in general (as we show) one may not do any better, but in some cases more efficient solutions may be possible.

We also prove that *in some settings sensing is a necessary condition for the existence of universal strategies*. Specifically, this is the case in the *finite executions* model, but not in the case of *infinite executions*.

We stress the generality of the foregoing statements (which go far beyond the "computational goals" considered by Juba and Sudan [9]; see discussion in Section 1.3).

1.2 A second look at the problem and our results

1.2.1 The problem

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. It is an obvious fact that modern computers communicate incessantly. It is also a fact that the protocols for communication, as well as the parties they are communicating with, are extremely diverse and furthermore are perpetually evolving. The principles, protocols, and assumptions regarding this reality of diverse and evolving communicating-computers are currently determined on a completely ad-hoc basis. This work attempts to provide a theoretical framework for addressing this reality.

Our focus on the two-party setting, especially the user-server setting, is not a major restriction. It captures most common forms of communication. In a variety of settings, users are trying to obtain various services from designated servers (e.g., printing on printers, obtaining drivers, visiting web-servers, etc). Furthermore, the user-server setting captures the essence of misunderstanding. Indeed, in all the foregoing examples, the users are interacting with servers without 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.

Our model focuses on the fact that in all such situations the user has some goal and wishes to achieve it. We view the server as just put there to help users. Still the issue at hand is that the

[4]The use of enumeration and other similarities between this part of our work to "Inductive Inference" or work in AI, notably Hutter's "Universal AI" [8], has confused some reviewers into believing the problems tackled are the same. We stress that this is a similarity in solutions, and not in the problem that we are hoping to tackle.

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.

1.2.2 On the nature of our results

We readily admit that we do not present solutions to all communication problems. In fact, our framework is so rich that it also includes goals that cannot be achieved even when the user communicates with a server such that they perfectly understand one another. Needless to say, our focus is on goals that are achievable in such a case (i.e., when the user is helped by a suitable server), and we ask whether such goals can be achieved also in the absence of initial mutual understanding. Indeed, our results address the question of when can the gap between initial mutual understanding and the lack of it be bridged, and how. The "when" part is addressed by providing sufficient and necessary conditions, whereas the "how" part refers to the presentation of design principles.

For example, the aforementioned result asserting that sensing allows for bridging the gap between initial mutual understanding and the lack of it (aka "universal user strategies") provides both a sufficient condition and a design principle. That is, this result is a strong incentive for designers of communication protocols to keep sensing in mind (i.e., provide mechanisms by which the user can sense progress). Additional examples appear in Section 1.2.4.

1.2.3 Two disclaimers (or rather clarifications)

We stress that our framework does not prescribe any particular goal over others, or any particular measure of success over others. There seems to be enough diversity in practice to suggest that any such prescriptions will be too restrictive and will simply be ignored. Instead we focus on creating a definitional framework that encompasses as broad a class of goals as possible. For instance, consider a setting where a user wishes to print a sequence of pages on a printer. It may be happy if it manages to print all of them perfectly correctly, or may be more flexible and be happy if they all print correctly after a few (fixed number of) initial mistakes, or even if mistakes occur only with small constant probability. Our study allows each one of these goals to be captured formally, thereby letting the designers choose their preferred goal.

We do not recommend actually implementing the solutions we provide. Our solutions 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.

1.2.4 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 a full and/or perfectly reliable understanding of the situation. Indeed, if we are to wait without actions until reaching 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.

By distinguishing the aspects of the model system that influence the referee's verdict, our model also presents a natural taxonomy of various kinds of goals. For example, if the referee's verdict only depends on the actions of the server, then we describe such a goal as "control-oriented," since the user must control the server's actions to achieve such a goal. Similarly, if the referee's verdict only depends on messages it exchanges with the user (and, for example, the server may have no means to interact directly with the environment), then we refer to such a goal as an "intellectual goal." In such a case, although the server may aid the user in helping it compute an appropriate message, a sufficiently powerful user could succeed without communicating with the server. This stands in contrast to "information gathering goals," where the user must send an appropriate message to the referee that is *not* determined by the user-referee communication, but rather depends on the server's view of the environment. This list is not exhaustive, but rather meant to give a taste of the variety of goals and distinctions that our model can capture.

Turning back to our results, we mention that a couple of them confirm the benefit in being able to reset the server. Indeed, resetting the server offers a fast way of recovering from damage caused by incompatible prior communication, since it guarantees that the damage is confined to the past (rather than being propagated to the future).

Another result confirms the value of "exploration," which in our case refers to invoking the server on "instances" (i.e., possible system configurations) that we don't really need to handle just in order to sense whether the current communication strategy being employed by us is adequate. The advantage in trying such instances is that they can be picked to be very small, and hence cause a negligible amount of overhead of all types. However, if the communication strategy being employed is inadequate, then this is due to some fixed instance, and if the actual instances are very large in comparison to it then we may gain by trying to find this small instance and rule out the current strategy at a lower cost than by using the actual instances.

1.3 Relation to related works of Juba and Sudan [9, 10]

The current work greatly extends the scope of the research direction initiated by Juba and Sudan [9]. For concretness we review their work here (in our language) and compare the conclusions.

In [9] the authors study the specific case of ("one shot") computational goals. Such a goal is specified by a decision problem D such that the user's goal is to decide D on arbitrary instances (which may be viewed as selected by the environment). The user, who cannot solve the problem D on its own, seeks help from a powerful server that can solve D, but the user and server may not understand one another. Juba and Sudan showed that, for any $D \in \mathcal{PSPACE}$, this initial misunderstanding can be overcome (provided that the server is helpful and can solve $\mathcal{PSPACE}$-complete problems), and that this is not possible if $D \notin \mathcal{PSPACE}$.

Specific criticisms of their work included concern that such "computational goals" do not capture all goals, and that requiring the server to use a $\mathcal{PSPACE}$-complete strategy seems severely limit the applicability of the result. In this work we address all these criticisms. Firstly, our thesis is that no goals of communication are excluded by our framework. Secondly, our results and techniques do not require that the user and server have vastly different computational resources (as in case of $\mathcal{BPP}$ versus $\mathcal{PSPACE}$); the server may even have less computing power than the user, and the gain in interacting with the server may stem from the server having a different interface with the environment. Indeed, as long as the user has something to gain from the server, either in the form of knowledge or in the form of changes to the environment that the user cannot effect on its own, the user can overcome misunderstanding so as to achieve the desired effects (provided that the user can sense the achievement of these effects). Thus, overcoming initial misunderstanding via interaction and sensing is possible regardless of the relative computational power of the user and the server—in fact, without any reference to it.

We proclaim [10] to be an early version of the current work.[5] However, these versions differ significantly in technical aspects. The bottom-line is that the current exposition subsumes [10], while introducing a natural formalization that is both more expressive and more transparent than the one used in [10]. The improvement is due to two main definitional ingrediants that are introduced in the current version:

1. The introduction of a third party (i.e., the referee/environment) for modeling goals

   The use of this third party allows a much more transparent and clear formulation of various goals of communication. This is also beneficial in the special case of one-shot goals.

2. The introduction of infinite goals.

   In contrast, the exposition in [10] is restricted to one-shot goals (a.k.a "finite" goals) and thus associates termination with achieving the goal. Here we consider also 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 meanwhile rather than waiting to a point, which may never occur, in which it may act without risk.

The new framework raises natural question that are less apparent in the formulation of [10]. In fact, the current version contains many results that did not appear in [10].

1.4 Relationship to works 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

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

study of communication [15]. As stated earlier, 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.

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].[6]

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 resembles these language games. We note, however, that Wittgenstein's reference to these language games is mainly descriptive and illustrative (primarily by examples).[7]

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

[6]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.

[7]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 akin to "ignoring friction" in physics problems.

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.[8]

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 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

[8]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.

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 affect 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 interacting 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 ability (as users) 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. In any such case, we obtain a user strategy that achieves the goal no matter with which of these servers it interacts.

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 last assertion implies that it is indeed good practice to use an enumeration (of possible communication strategies) that reflects the user's a priori beliefs regarding the server. For example, if the user is quite sure that the server uses a specific communication format, then it better place the corresponding communication strategy as first in this enumeration. In this case, if the user's guess is correct, then it suffers no damage and/or incurs no overhead, and otherwise it still achieves the goal (but at the cost of limited damage and/or overhead). Thus, using the universal strategy may be viewed as a safeguard for the case that the user's beliefs are wrong.

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. 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. Some more substantial extensions, discussed below, are considered in Sections 5 and 6.

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 is treated in Section 5.1.

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, described in Section 5.3, 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]. We provide a treatment of one-shot goals using the much more transparent modeling of the current exposition in Section 6.

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 $i \in [m] \stackrel{\text{def}}{=} \{1, ..., m\}$, we denote the internal state of the i^{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^{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 3.1. Likewise, the message in transit from the i^{th} party to the

j^{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.

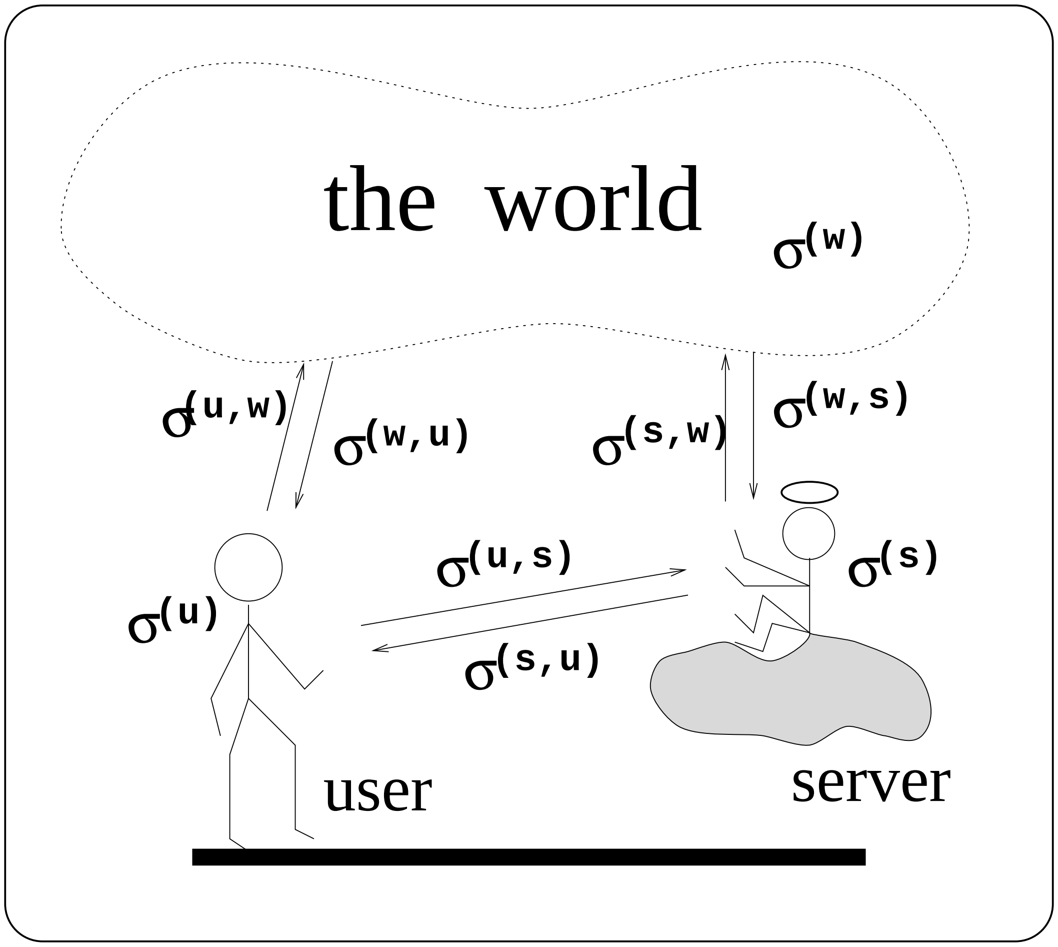

Figure 1: The canonical system: the world, user, and server.

Definition 3.1 (strategies): *A strategy of the* i^{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 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 also consider 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 also model 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 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).[9]

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

[9]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).

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)[10] *world 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 communicates 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.

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 3.1). A goal in which the server's help is required, regardless of computational resources, follows.

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

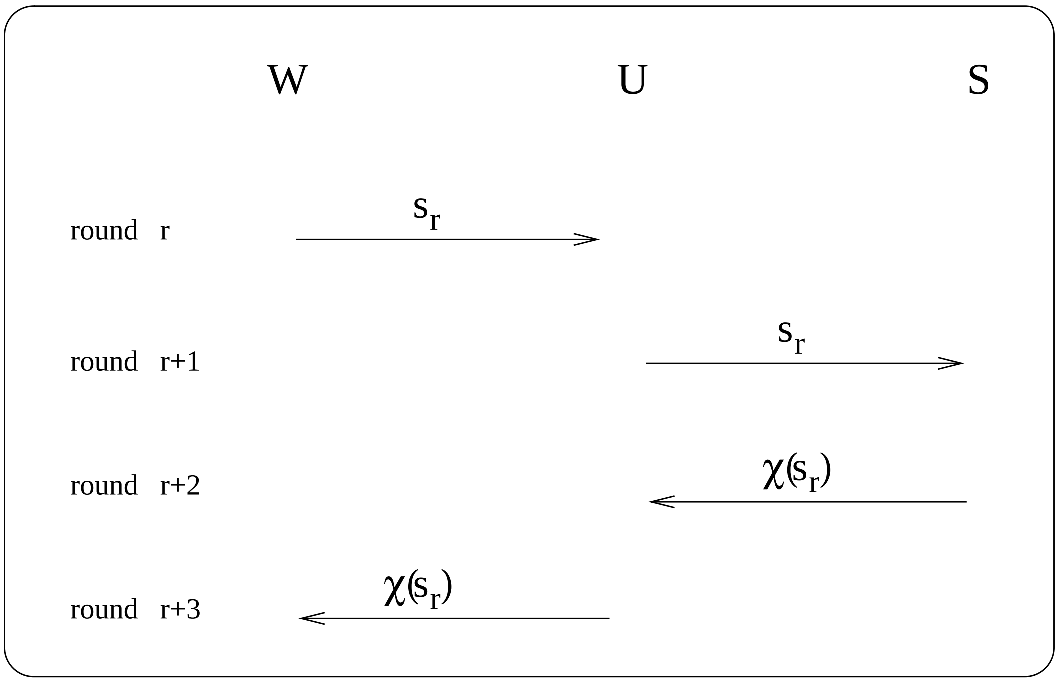

Figure 2: The time-line for getting the server's help in deciding instances of D.

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., server) 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.[11] 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.

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 $R'(\sigma_{t'}) \neq \perp$.)[12]

Multi-session goals. Examples 3.6–3.8 actually belong to a natural subclass of compact goals, which we call multi-session goals.[13] 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

[11] 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.

[12] 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.

[13] 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 $3i, 3i+1, 3i+2$.

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 (an integer representing the index of the session) *and a* contents (representing the state of the actual execution of the session).[14] *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.*

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.)

[14]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).

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 sessions.[15]

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 $W(i, \sigma') \equiv W(1, \sigma')$.*[16]

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, is postponed to §5.2.1 (cf. Definition 5.4). Note that Examples 3.7 and 3.8 fit this general formulation without any modification (cf. Footnote 13).

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.*

[15]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.

[16]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).

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 reflects 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. A more refined definition, which quantifies over a subset of the possible states is postponed to Section 5 (see Definition 5.10). Most importantly, this refined definition allows to consider the (natural case of the) set of all global in which the user's local state is reset to some initial value. (Indeed, in contrast to resetting the world, resetting the user seems feasible in many cases, and seems less demanding than resetting the server.)

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 also be helpful to the user 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*[17] *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 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*

[17] 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).

size of the various states remains invariant throughout the execution; the general case (in which the size varies) will be dealt with in Section 5.1. Anyhow, we assume that the size of the current state is known to the user in the present development.

Our formulation will be further simplified by observing that the quantification over all initial states (which also takes place in Definition 3.16) allows us 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 $t \in [B(\mathsf{s}(\sigma_1)) + 1, 2B(\mathsf{s}(\sigma_1))]$ 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).[18] 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.

[18]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 $B : \mathsf{N} \to \mathsf{N}$ 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 $t \in [B(\mathsf{s}(\sigma_1)) + 1, 2B(\mathsf{s}(\sigma_1))]$ 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).[19] 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 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). *In some contrast, for each version of viability, we say that U' is* viable with respect to U and the server class $\mathcal{S}$ *if there exists $S \in \mathcal{S}$ such that U' is viable with respect to (U, S).*

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.

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

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 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 interpret 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).[20] 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*

[20] 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.

class $\mathcal{S}$ (and G).[21] *Furthermore, the mapping $U \mapsto (U', B)$ is computable,*[22] *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).

Indeed, Condition 1 (which implies weak sensing as per Definition 3.16)[23] 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 $\overline{\sigma}$, 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 t_i, and b_i be as above. Then, by the strong viability condition, for every $t > t_i + b_i$, it holds that $U'_i(\sigma_t) = 1$. ■

[21] See Definition 3.19.

[22] Recall that we view the sensing function U' as an augmentation of the strategy U, and the same applies to the bound B. Indeed, an alternative formulation of the conditions is obtained by replacing the current mapping and enumeration requirement by requiring that the set of triples (U, U', B) such that $U \in \mathcal{U}$ (and (U', B) are the corresponding sensing function and strong viability bound) is enumerable.

[23] See Footnote 19.

Claim 4.6.2 *Suppose that (U, S) does not robustly achieve 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 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"). ■

On the computational complexity of the universal strategy. Note that Theorem 4.6 asserts that, for every server $S \in \mathcal{S}$, the computational complexity of the universal strategy (when interacting with S) is comparable to the computational complexity of some user strategy in $\mathcal{U}$ (when interacting with S). Thus, if $\mathcal{U}$ denotes the class of user strategies that are implementable in probabilistic polynomial-time when interacting with any fixed $S \in \mathcal{S}$ (where the polynomial may depend on S), then the universal strategy resides in $\mathcal{U}$. Indeed, this stands in contrast to standard universality results in complexity theory, where the universal machine does not reside in the class for which it is universal.[24] The reason that the universal strategy of Theorem 4.6 escapes this fate is that its complexity is measured with respect to the server that it interacts with, and so it may afford to spend a different amount of time when emulating each of the corresponding user strategies.

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

[24]Note that completeness results avoid this fate by padding the instances.

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 the success to be sensed 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[25] 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 $W(\sigma)^{(\mathtt{w,u})}$ equals $R'(\sigma)$.*

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

We note that compact goals that allow $\perp$-runs of a priori bounded length (as in Footnote 12) can be converted to (functionally equivalent) compact goals that allow no $\perp$-values (w.r.t R').[26]

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. ■

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.

[26]That is, for R' as in Definition 3.9, we assume here the existence of a function $B:\mathsf{N}\to\mathsf{N}$ such that $R(\overline{\sigma})=1$ only if for every $t>0$ there exists $t'\in[t+1,t+B(\mathsf{s}(\sigma_1))]$ 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 $i<B(\mathsf{s}(\sigma))$.

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 non-deterministically selects 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).[27] *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 memoryless strategies* (i.e., strategies that maintain no local state)[28] *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 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, and the verdict of the program checker determines the verdict of U'. Note that this U' is strongly viable with respect to U and the D-solver, and safe with respect to U and the class $\mathcal{F}$, where the crucial point is that the strategies in $\mathcal{F}$ are memoryless. Furthermore, the bound in the strong viability condition is the constant 1, since a solver is correct in each round.

  Recall that program checkers exist for $\mathcal{PSPACE}$-complete and $\mathcal{EXP}$-complete problems (cf. [11, 14] and [1], respectively).[29]

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.

[27]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.

[28]Recall that strategies map pairs consisting of the current local state and the incoming messages to pairs consisting of an updated local state and outgoing messages. In case of memoryless strategies there is no local state (or, equivalently, the local state is fixed).

[29]See also [2].

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 non-deterministically selecting 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).

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).[30]

[30] 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

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 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).

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").

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 $b \overset{\text{def}}{=} B(\mathsf{s}(\sigma_1))$.*
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 and U' 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 $1 - \epsilon(\mathsf{s}(\sigma_1))$, either $R_B(\overline{\sigma}) = 1$ 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 $1 - \epsilon(\mathsf{s}(\sigma_1))$, either $R_B(\overline{\sigma}) = 1$ or for some $t \in [B(\mathsf{s}(\sigma_1)) + 1, 2B(\mathsf{s}(\sigma_1))]$ it holds that $U'(\sigma_t) = 0$, where here the probability refers to the execution suffix $(\sigma_{t+1}, \sigma_{t+2}, ...,)$.*

*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 $1-\epsilon(\mathsf{s}(\sigma_1))$, for every $t \geq B(\mathsf{s}(\sigma_1))$ 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.[31]

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 $B_{i+1} < B_i/2\epsilon$.*

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 $\mathcal{U}$.*

[31]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.

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 $\{t \in [t_i, t_{i+1}] : (\forall t' \in [t, t+b_i])\ R'(\sigma_{t'}) = \perp\}$ 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 $|\{t \in \mathsf{N} : (\forall t' \in [t, t+b])\ R'(\sigma_{t'}) = \perp\}| > b$). 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 $R'(\sigma_{t'}) = \perp$. 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 $b_i + (t'' - t) \leq 3b_i$. 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 $r \geq 1$ 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 $\sum_{r\geq 0} \epsilon^r \cdot \sum_{j\in[i+r]} 6B_j$. 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 goal 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 may incur $\Theta(\Delta)$ 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 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 and U' 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 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.

Indeed, while helpfulness with B errors refers to the expected number of errors, the notion of $(B, \cdot)$-viability refers to the probability that the number of failure indications exceeds B. Needless to say, the latter bound is easily related to an upper bound on the expected number of failures.

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 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 $(m - b_i)/b_i \le b_i$. Thus, Claim 4.20.1 follows. As for Claim 4.17.2, it is modified as follows.

Claim 4.20.2 *Let $i \ge 1$ 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 $3B_i^2(\mathtt{s}(\sigma_1))$.*

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 $j \le i$, 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 $3\sum_{j\in[i]} B_j^2(\mathtt{s}(\sigma_1))$. By the induction hypothesis, the expected number of errors that occur before round t_i is at most $3\sum_{j\in[i-1]} B_j^2(\mathtt{s}(\sigma_1))$, and so at least $3b_i^2$ errors occur after round t_i, where $b_i = B_i(\mathtt{s}(\sigma_1))$. 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 $\epsilon \le 1/3$ (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.*[32]

[32]In fact, strong safety holds with respect to all possible servers. This fact follows from the unconditional soundness of the interactive proof system (i.e., soundness holds no matter which strategy is used for the cheating prover).

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 $\mathcal{PSPACE}$-complete and some problems in $\mathcal{SZK}$ that are believed not to be in $\mathcal{P}$ (cf. [11, 14] and [6], respectively).[33]

- *By invoking Theorem 4.20 we obtain an $\mathcal{S}$-universal user strategy, where $\mathcal{S}$ denotes 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,

[33]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].

for sake of robustness (both intuitive and technical)[34] 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 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 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).[35]

[34]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.

[35]Indeed, also the trivial server that prevents R' from ever evaluating to 1 may be viewed by Definition 4.15 as

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.

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 non-triviality 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 strongly safe with respect to $\mathcal{S}$, which may contain only $\mathcal{U}$-helpful servers, then U' is strongly safe with respect to any server (including servers that are not helpful to any user). Thus, *a strongly safe sensing function cannot be trivial.*

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-

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 $\sum_{i=1}^{\ell/2} 2^i = O(2^{\ell/2})$.

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 servers): *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 $0^{i-1}1$ 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 $2\epsilon(\mathsf{s}(\sigma_1))$, the random execution $\overline{\sigma}$ is such that $U'(\sigma_i) = 1$ for all $i \leq B(\mathsf{s}(\sigma_1))$ 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 hybrid strategy $\widetilde{S}^*$ such that $\widetilde{S}^*$ behaves like S^* on messages of length at most i and behaves more like S otherwise. Specifically, upon receiving a message of length greater than i, the strategy $\widetilde{S}^*$ omits the first $i+1$ bits, feeds the result to S, and answers as S does. 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. ■

5 Extensions

In this section, we discuss various natural augmentations of our basic model including the treatment of varying state sizes (see Section 5.1), generalizations of multi-session goals (see Section 5.2), and notions of resettable servers (see Section 5.3) and partial robustness (see Section 5.4).

5.1 Varying state sizes

Our basic treatment, provided in Sections 3 and 4, postulates that the size of the various states remains invariant throughout the execution. This postulate was made mainly for sake of simplicity, and we waive it here both for sake of generality and because the generalization seems essential to an appealing result that appears in §5.3.2.

5.1.1 Extending the definitional treatment

Recall that the size of the various states in the execution is used only as a basis for defining various bounds, which are stated as functions of the state's size. Given that the state's size may change, the question that we face is how to express these bounds in such a case.[36] Recall that we use bounds of two types.

1. *Bounds that determine the length of various intervals*, including the length of intervals in which the temporal decision is suspended (i.e., $R' = \perp$) or the delay of sensing (e.g., in the definition of safety). For example, both types of delays appear in (Item 2 of) Definition 3.18 (which refers to strong sensing).

   Since such bounds refer to some "transient" events (i.e., a state in which $R' = 0$ or the first state in a $\perp$-run under R'), it is natural to keep them expressed in terms of the size of the corresponding state (in which the event occurs).

2. *Bounds that determine the total number of various events*, including the number of allowed errors and/or detection failures (as in Definition 4.15 and Definition 4.19, respectively).

   Since these bounds are "global" in nature, it makes no sense to associate them with any single event (or state or size). Instead, we may view each individual bad event (i.e., an error and/or detection failure) as contributing to a general pool, and weight its contribution with reference to the relevant size. (See Definition 5.1.)

In accordance with the foregoing discussion, the definitions of sensing functions (i.e., Definition 3.18 (and, needless to say, Definition 3.16) remain intact (although the size of the various states in an execution may vary). We stress that, since our universal strategies refer to these bounds, it is

[36]While we believe that the definitional choices made here (i.e., in Section 5.1) are reasonable, we are far from being convinced that they are the best possible.

important to maintain our postulation by which *the user knows the size of the current* (global) *state.*[37]

We now turn to the treatment of global bounds, like the bounds on the total number of errors (in Definition 4.15). Recall that Item 1 in Definition 4.15 states that the expected cardinality of $\{t \in \mathsf{N} : R'(\sigma_t) = 0\}$ is at most $B(\mathsf{s}(\sigma_1))$, for every initial state σ_1. However, when the size of states may vary, it makes little sense to bound the number of errors with reference to the size of the initial state. Instead, we may consider an error at state σ_t as contributing an amount proportional to $1/B(\mathsf{s}(\sigma_t))$ (towards the violation of the "error bound") and say that a violation occurs if the sum of all contributions exceeds 1.

Definition 5.1 (varying size version of Definition 4.15): *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 value of the sum $\sum_{t \in \mathsf{N}: R'(\sigma_t)=0} \frac{1}{B(\mathsf{s}(\sigma_t))}$ is at most 1.*

2. *The expected value of the sum $\sum_{t \in \mathsf{N}: (\forall t' \in [t, t+B(\mathsf{s}(\sigma_t))])\ R'(\sigma_{t'})=\perp} \frac{1}{B(\mathsf{s}(\sigma_t))}$ is at most 1.*

If $\overline{\sigma}$ is an execution in which the bounds corresponding to the foregoing conditions are both satisfied, then we write $R_B(\overline{\sigma}) = 1$. Finally, if $\overline{\sigma}$ is an execution such that $R'(\sigma_t) = 0$ or $(\forall t' \in [t, t + B(\mathsf{s}(\sigma_t))])\ R'(\sigma_{t'}) = \perp$, then we say that $\overline{\sigma}$ contains an error in round t.

Note that each individual bad event in Item 2 is defined with respect to the size at the corresponding time, and (like in Item 1) its contribution is defined with respect to the size at the corresponding time. However, in both items, the condition refers to the aggregate contribution, where each event may contribute a different amount to this sum. Observe that in the case that the state remains fixed throughout the execution, Definition 5.1 coincides with Definition 4.15.

A similar modification should be applied to the definition of robustly achieving a goal. Consequently, the refined definition of strong safety (i.e., Definition 4.16) is updated by merely postulating that R_B is as defined in Definition 5.1 (rather than as in Definition 4.15). Lastly, we generalized the definition of relaxed viability (i.e., Definition 4.19)[38] analogously to the foregoing modification (i.e., that yielded Definition 5.1).

Definition 5.2 (varying size version of Definition 4.19): *Let $G = (\mathcal{W}, R)$, S, U and U' be as in Def. 3.18. For $B : \mathsf{N} \to \mathsf{N}$ and $\epsilon \in [0, 1/3]$, 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 $1 - \epsilon$, the value of the sum $\sum_{t \in \mathsf{N}: U'(\sigma_t)=0} \frac{1}{B(\mathsf{s}(\sigma_t))}$ is smaller than 1. If $\epsilon = 0$, the we say that U' is* B-viable.

As commented in §4.4.1, the foregoing definitions are simplified versions of more general definitions that use different bounding functions for the various bounds that underly these definitions.

[37]Indeed, we relied on this postulation also in the fixed-size case, since it was used there in the same way. However, in the current context the state may change all the time, and the user should be aware of these changes (at least whenever it needs to determine the values of these bounds).

[38]In order to avoid a possible controversy, we state Definition 5.2 only for constant values of ϵ, whereas Definition 4.19 allowed any $\epsilon : \mathsf{N} \to [0, 1/3]$. Note that $\epsilon = 0$ and $\epsilon = 1/3$ are indeed the most important cases (cf. Definition 3.18). Nevertheless, we mention that we believe that when allowing ϵ to vary, it is most natural to apply it to the initial state (indeed, as in Definition 4.19).

5.1.2 Extending the (fixed-size) universality results

The universality results stated in Section 4 can be generalized to the context of varying sizes, where we refer to the generalized definitions presented in §5.1.1. Actually, we can prove these generalized results only for goals in which the state size does not change dramatically from one round to the next one. For simplicity, we only consider one concrete case, in which the size can change at each round by at most one unit. (Note that goals that allow arbitrary changes in the state sizes can be emulated by the foregoing restricted goals by introducing an adequate number of dummy rounds.) Similarly, we consider only small bounding functions, while larger bounds can be handled by artificially increasing the size measure (which is an arbitrary function of the states).

Theorem 5.3 (varying size version of Theorem 4.20): *Let $G = (\mathcal{W}, R)$, $\mathcal{U}$, $\mathcal{S}$, ϵ and $\mathcal{B}$ be as in Theorem 4.20, except that here we refer to the varying-size generalization of the notions of achieving and sensing* (and in particular to replacing Definition 4.19 by Definition 5.2). *Suppose that G is such that in each execution $\overline{\sigma} = (\sigma_1, \sigma_2, ...)$ and at every time t it holds that $|\mathsf{s}(\sigma_{t+1}) - \mathsf{s}(\sigma_t)| \leq 1$. Further suppose that for every $B \in \mathcal{B}$ it holds that $B(s+d) \leq B(s) + (d/2)$, and that for every two functions in $\mathcal{B}$ it holds that one of them dominates the other* (i.e., for every $B_1, B_2 \in \mathcal{B}$ and $s, s' \in \mathsf{N}$, if $B_1(s) < B_2(s)$, then $B_1(s') \leq B_2(s')$). *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 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 $\mathcal{U}$.*

Recall that by saying that a "goal is achieved with a certain number of errors" we mean that the expected contribution of all errors is bounded by 1, where the contribution of each error is "normalized" with respect to the relevant size (as per Definition 5.1).

Proof Sketch: We follow the outline of the proof of Theorem 4.20, while adapting the "accounting of failure indications". Recall that, in that proof (and while in the case of $\epsilon \equiv 0$), we introduced a universal user strategy U that switches from the user strategy U_i to the next strategy (i.e., U_{i+1}) after seeing sufficiently many failure indications, where "sufficiently many failures" meant a number that exceeds a predetermined bound that was expressed in terms of the fixed size (of states). Here, sufficiently many failures will mean an accumulated contribution that exceeds 1, where each contribution is normalized with respect to the relevant size (as per Definition 5.2). Specifically, suppose that U starts emulating U_i at round t_i, then *strategy U emulates U_i until a time t_{i+1} such that the sum $\sum_{t \in (t_i, t_{i+1}] : U'_i(\sigma_t) = 0} \frac{1}{B_i(\mathsf{s}(\sigma_t))}$ exceeds 1, and switches to U_{i+1} once the latter event occurs.*

We shall show that Claims 4.20.1 and 4.20.2 remain essentially valid, when modified in accordance to the relevant new measures. Specifically, Claim 4.20.1 is modified as follows.

Claim 5.3.1 *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, letting E denote the set of rounds containing errors* (as in Definition 5.1) *it holds that $\sum_{t \in E : t > t_i} \frac{1}{4B_i^2(\mathsf{s}(\sigma_t))} \leq 1$, where t_i is as above.*

Proof sketch: As in the case of Claims 4.17.1 and 4.20.1, the main part only relies on B_i-viability and follows from the construction of U. The furthermore part follows by observing that B_i-viability mandates a upper bound on the contribution of failure indications (w.r.t U'_i), whereas the B_i-strong safety condition translates the total contribution of errors (w.r.t R') to a lower bound on the the contribution of failure indications (w.r.t U'_i). Specifically, suppose towards the contradiction that $\sum_{t\in E:t>t_i}\frac{1}{B_i^2(\mathsf{s}(\sigma_t))}>4$. Consider $b>t_i$ such that both $\sum_{t\in E\cap(t_i,b]}\frac{1}{B_i^2(\mathsf{s}(\sigma_t))}>2$ and $\sum_{t\in E:t>b}\frac{1}{B_i^2(\mathsf{s}(\sigma_t))}>1$ hold, and let $E'=\{t\in E: t\in(t_i,b]\}$. Now, by the B_i-strong safety condition, for every $t'\in E'$ there exists $t''\in[t',t'+B_i(\mathsf{s}(\sigma_{t'}))]$ such that $U'_i(\sigma_{t''})=0$ (because $R_{B_i}(\sigma_{t'})$ evaluates to 0). Let us denote the corresponding mapping by $\pi: E'\to(\mathsf{N}\setminus[t_i])$; that is, for every $t'\in E'$ it holds that $\pi(t')\in[t',t'+B_i(\mathsf{s}(\sigma_{t'}))]$ and $U'_i(\sigma_{\pi(t')})=0$. Then:

$$\begin{aligned}\sum_{t>t_i:U'_i(\sigma_t)=0}\frac{1}{B_i(\mathsf{s}(\sigma_t))} &\geq \sum_{t\in\pi(E')}\frac{1}{B_i(\mathsf{s}(\sigma_t))}\\ &= \sum_{t'\in E'}\frac{1}{|\pi^{-1}(\pi(t'))|}\cdot\frac{1}{B_i(\mathsf{s}(\sigma_{\pi(t')}))}\\ &\geq \sum_{t'\in E'}\frac{1}{2B_i^2(\mathsf{s}(\sigma_{\pi(t')}))}\end{aligned}$$

where the last inequality follows from the fact that $|\pi^{-1}(t'')|\leq 2B_i(\mathsf{s}(\sigma_{t''}))$, which in turn follows by combining $\pi^{-1}(t'')\subseteq\{t'\in[t'']: t'+B_i(\mathsf{s}(\sigma_{t'}))\geq t''\}$ with $|\{t'\in[t'']: t'+B_i(\mathsf{s}(\sigma_{t'}))\geq t''\}|\leq 2B_i(\mathsf{s}(\sigma_{t''}))$, where the last fact relies on two of the technical conditions of the theorem.[39] Using $\sum_{t'\in E'}\frac{1}{2B_i^2(\mathsf{s}(\sigma_{t'}))}>1$, we infer that $\sum_{t>t_i:U'_i(\sigma_t)=0}\frac{1}{B_i(\mathsf{s}(\sigma_t))}>1$, which causes the execution to switch to U_{i+1}, in contradiction to the main part. The furthermore part follows. ■

Regarding Claim 4.20.2, it is modified as follows.

Claim 5.3.2 *Let $i\geq 1$ and suppose that (U,S) does not robustly achieve the goal with $6i\max_{j\in[i]}B_j^2$ errors.*[40] *That is, letting E be as in Claim 5.3.1, it holds that the expected value of $\sum_{t\in E}\frac{1}{\max_{j\in[i]}\{B_j^2(\mathsf{s}(\sigma_t))\}}$ exceeds $6i$. Consider a random execution of (W,U,S), and let t_i,t_{i+1} be as above. Then, recalling that each U'_j is B_j-strongly safe* (w.r.t (U_j,S) *it holds that the expected value of $\sum_{t\in E:t\in(t_i,t_{i+1}]}\frac{1}{B_i^2(\mathsf{s}(\sigma_t))}$ is at most* 6.

Combining Claims 5.3.1 and 5.3.2 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 $B(s)=6i\max_{j\in[i]}\{B_j^2(s)\}$.

Proof sketch: Following the proof of Claim 4.20.2, we proceed by induction on i. Let σ_1 be a global state such that the expected value of $\sum_{t\in E}\frac{1}{\max_{j\in[i]}\{B_j^2(\mathsf{s}(\sigma_t))\}}$ exceeds $6i$. By the induction hypothesis, the expected value of $\sum_{t\in E:t\leq t_i}\frac{1}{\max_{j\in[i-1]}\{B_j^2(\mathsf{s}(\sigma_t))\}}$ is at most $6(i-1)$, and so the

[39]Specifically, using $|\mathsf{s}(\sigma_{t'})-\mathsf{s}(\sigma_{t''})|\leq|t'-t''|$ and $B_i(s+d)\leq B_i(s)+(d/2)$, we upper-bound $|\{t'\in[t'']: t'+B_i(\mathsf{s}(\sigma_{t'}))\geq t''\}|$ by $|\{t'\in[t'']: t'+B_i(\mathsf{s}(\sigma_{t''}))+(t''-t')/2\geq t''\}|$.

[40]Here we use the technical hypothesis by which for every two functions in $\mathcal{B}$ it holds that one of them dominates the other. Hence, $\max_{j\in[i]}\{B_j^2\}$ is well defined, and if $B_1(s)<B_2(s)$ for some $s\in\mathsf{N}$ then $B_1(s')\leq B_2(s')$ holds for all $s'\in\mathsf{N}$.

expected value of $\sum_{t\in E:t\in(t_i,t_{i+1}]}\frac{1}{B_i^2(\mathsf{s}(\sigma_t))}$ is at least 6. By B_i-strong safety, for each of these $t\in E\cap(t_i,t_{i+1}]$ there exists $t'\in[t,t+B_i(\mathsf{s}(\sigma_t))]$ such that $U_i'(\sigma_{t'})=0$, and by an accounting similar to the one in the proof of Claim 5.3.1 it follows that $\sum_{t\in(t_i,t_{i+1}]:U_i'(\sigma_t)=0}\frac{1}{B_i(\mathsf{s}(\sigma_t))}>3$, which causes U to switch to emulating U_{i+1} before t_{i+1} (in contradiction to the definition of t_{i+1}). The claim follows. ■

As in the proof of Theorem 4.17, we need to extend the analysis to the general case in which $\epsilon\le 1/3$ (rather than $\epsilon=0$). The extension is analogous to the original one, where (as in the proof of Theorem 4.20) each repetition causes an overhead of $O(B^2)$ errors. ■

5.2 Generalized multi-session goals

In this section we consider two (mostly orthogonal) generalizations of the notion of multi-session goals. The first generalization, which refers to concurrent sessions (and appears in §5.2.1), is presented merely as a framework towards future study. The second generalization, which supports "user exploration" sessions (and appears in §5.2.2), seems essential to the result presented in §5.3.2. For sake of simplicity, we do not combine these two generalizations (although such a combination makes sense), but rather present each generalization with reference to the basic definition of multi-session goals (i.e., Definition 3.10).

In relation to the treatment of state sizes in Section 5.1, it is natural to comment of the state sizes in multi-session goals. It is natural to postulate that, in the basic formulation (i.e., Definition 3.10), the state size remains invariant during each session, and is thus determined by the start-session state of this session. In the case of concurrent sessions (see §5.2.1) it is natural to define size as a function of the sizes associated with all active sessions (e.g., the maximum size and the sum of sizes seem natural choices).

5.2.1 Concurrent sessions

Our basic formulation of multi-session goals (see Definition 3.10) mandates that the current session ends before any new session can start. A more general formulation, which allows concurrent sessions, follows.

Definition 5.4 (concurrent multi-session goals, sketch): *A goal consisting of a non-deterministic strategy $\mathcal{W}$ and a referee R is called a* concurrent multi-session goal *if the following conditions hold.*

1. The world's states: *The local states of the world consist of* (non-empty) *sequences of pairs, where each pair is called a* session state *and has a form as postulated in the first condition of Definition 3.10; that is, each session state is a pair consisting of an* index *and a* contents*, and belongs to one of three sets of session states called* start-session states*,* end-session states*, and* intermediate session states*. The initial local state corresponds to the single pair* $(0,\lambda)$*, and belongs to the set of end-session states.*

2. The referee verdict depends only on the end-session states: *The referee R is compact. Furthermore, the corresponding function R' evaluates to $\perp$ if and only if the current state contains no end-session state. Otherwise, the value of R' is a conjunction of the values of some Boolean predicate R'' that is applied to all the end-session states that appear in the current state.*

3. Starting new sessions: *At any round, the world may start an arbitrary number of new sessions. This is done by moving non-deterministically to a state that contains a list of start-session states, each having an index that does not appear in the previous list of session states.*[41] *The contents of each of these new session states is determined by the world based solely on the indices of the existing sessions* (and is invariant of their contents; cf. Condition 3 of Definition 3.10)*.*

4. Execution of the current active sessions: *In addition to the above, the world takes a move in each of the active sessions that are listed in the current state, where a session is called* active *if it is not in an end-session state. The world's movement in each such session is probabilistic and is independent of the index as well as of the actual world strategy* (cf. Condition 4 of Definition 3.10)*. Furthermore, this movement maintains the index of the session.*

Note that the state maintains the list of all non-active sessions (i.e., sessions that reached a end-session state). An alternative formulation may omit the non-active sessions from the state, and maintain instead an explicit counter that represents the number of sessions started so far.

5.2.2 Exploration sessions and other effects on the world

Our basic formulation of multi-session goals (see Definition 3.10) mandates that the world determines the initial state of new sessions obliviously of any previous actions of the parties (i.e., actions of these parties in prior sessions). This is an artifact of the postulate that the world's move at an end-session state only depends on the index of that state (i.e., the index of the last session), which means that whatever effects the user and server have had on the world (during the last session) is dissolved at the beginning of a new session. This somewhat stringent postulate was made in order to develop a notion of sessions that are independent of one another (from the world's point of view). In contrast, at the extreme, allowing the world's actions (in each session) to depend on the entire history collapses such multi-session goals to general compact goals. An intermediate case, which seems very appealing, refers to multi-session executions in which the dependence of the world's actions on the history of prior sessions is limited. Specifically, we restrict the dependence of the world's actions on the history of prior sessions to the selection of the contents of the initial state in new sessions. Furthermore, we introduce a framework that allows to consider cases where the selection of the initial state (of the new session) is further restricted.

In addition to the general appeal of the new framework, it facilitates the introduction of the notion of "exploration sessions:" these are sessions that are actually initialized by the user with the aim of assisting it to later perform better in "real" sessions that are invoked by the world, as usual. Note that such sessions can be easily modeled (by the new framework) by having the user indicate at the end of the current session that it wishes to perform such an exploration and even have the world behave as if it has selected a specific contents for the initial state of the next session.[42]

Definition 5.5 (history-dependent multi-session goals, sketch): *Let $\mathcal{H}$ be a family of functions, representing the allowed history-dependent actions of the world when initiating a new session. A*

[41] Without loss of generality, this index may be the smallest integer that does not appear in that list.

[42] Indeed, such an effect can also be captured by the original formulation (i.e., of Definition 3.10) by an awkward modification of the world's strategy. However, capturing such exploration sessions via Definition 5.5 seems more transparent.

goal consisting of a non-deterministic strategy $\mathcal{W}$ and a referee R is called an $\mathcal{H}$-dependent multi-session goal *if the following conditions hold.*

1. The world's states: *As in Definition 3.10, 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 a* record (representing a digest of the history of prior sessions) *and a* contents (representing the actual state within the execution of the current session).[43] *The initial local state corresponds to the pair (λ,λ), and belongs to the set of end-session states.*

2. The referee behaves like in Definition 3.10; *that is, 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. Furthermore, like in Definition 3.10, this move is independent of the actual contents of the current end-session state. That is, for each actual world strategy $W \in \mathcal{W}$, W is invariant over all possible end-session states that have the same record, and it fits some function in $\mathcal{H}$; that is, for each $W \in \mathcal{W}$ there exists $h \in \mathcal{H}$ such that for every end-session state $(r,\sigma') \in \{0,1\}^* \times \Omega$, it holds that $W(r,\sigma') = h(r) \in \{0,1\}^* \times \Omega$.*[44]

   Optional (as in Definition 3.10): *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.*

4. Execution of the current session: *When being in any other state, the world moves probabilistically, while possibly updating the record, but its behavior is independent of the actual world strategy. That is, for every $W_1, W_2 \in \mathcal{W}$ and every non-end-session state (r,σ'), the random variables $W_1(r,\sigma')$ and $W_2(r,\sigma')$ are identically distributed. Furthermore, the contents of that state changes obliviously of the record; that is, for every $W \in \mathcal{W}$ and pair of non-end-session states $((r_1,\sigma'),(r_2,\sigma'))$, the second element of $W(r_1,\sigma')$ is distributed identically to the second element of $W(r_2,\sigma')$* (i.e., for every $\sigma'' \in \Omega$ it holds that $\sum_{r'} \Pr[W(r_1,\sigma') = (r',\sigma'')]$ equals $\sum_{r'} \Pr[W(r_2,\sigma') = (r',\sigma'')]$)*.*

Indeed, Definition 3.10 is a special case of Definition 5.5, obtained by requiring that for every $W \in \mathcal{W}$ and every state (r,σ') the first element of $W(r,\sigma')$ equals $r+1$ if the state is an end-session state and equals r otherwise. Needless to say, here we are interested in other cases.

One natural type of history-dependence that is captured by Definition 5.5 is the dependence of (the initial contents) of the next session on the record of failures and successes of prior sessions. Another natural case, which refers to the aforementioned (multi-session goals with) exploration sessions, is defined next. In this case, the record maintains the number of sessions completed so far and possibly also an exploration request, denoted e, sent by the user (say, at the very end of the last session).

Definition 5.6 (exploration sessions, sketch): *A* multi-session goal with exploration sessions *is an $\mathcal{H}$-dependent multi-session goal as in Definition 5.5 where for every $h \in \mathcal{H}$ it holds that $h(r) = (i+1, e)$*

[43] Indeed, the index of the current session (used in Definition 3.10) is a special case of the record (of prior sessions).

[44] Note that Condition 3 in Definition 3.10 can be stated in this manner, when r equals the current index i, and $\mathcal{H}$ is the set of all functions h over Ω such that $h(i) = (i+1, h'(i))$ for some $h' : \mathbb{N} \to \Omega$. Alternatively, we can state the condition in Definition 3.10 by postulating that h only depends on the number of prior sessions recorded in r.

if $r = (i, e)$, *and otherwise the world replaces the record* $r = i$ *by* $i + 1$ (where $r = i$ and $i + 1$ are viewed as integers). *Lastly, during the execution of a session, the record* $r = i$ *remains intact unless* (at the session's end) *the world receives a special exploration request with contents* e (from the user), *which sets the record to the value* (i, e).

Indeed, $r = (i, e)$ encodes the event that session i ended with the user requesting exploration with contents e, and otherwise the record is viewed as merely encoding the number of sessions completed so far. In the latter case, the new session is initialized with a contents that only depends on the world's non-deterministic choice (and on the number of sessions completed so far).

The advantage of exploration sessions will be demonstrated in §5.3.2, but in the next few pages we shift gears (again) and consider a seemingly orthogonal notion (of resettable servers). Actually, this notion plays a major role in the aforementioned demonstration, which explains the organization of our exposition.

5.3 Resettable Servers

A natural feature that many servers have is resettability: that is, a guarantee that these servers can be simply reset to a predetermined initial state. This resetting is performed in response to a simple predetermined user command (or message). Indeed, we distinguish the case in which the "resetting command" (or "resetting message") is known a priori from the case in which this command (or message) may not not be known a priori. Needless to say, it is more advantageous to have servers of the first type, but we mention (see discussion below) that it is also beneficial to just have servers of the second type. Formally, we capture the difference by considering classes of resettable servers that respond to the same resetting command.

Definition 5.7 (user-resettable servers): *A server strategy is called* user-resettable (or just resettable) *if upon receiving a special* (resetting) *message from the user, it moves to a predetermined state, which coincides with its initial local state. A class of resettable servers is called* uniformly resettable *if all servers in the class respond to the same resetting message.*

Note that we do not assume that the servers in a uniformly resettable class have the same initial local state, nor do we assume any other common features (beyond resetting upon receiving the same message).

Uniformly vs non-uniformly resettable servers. In the rest of this section, we will refer to classes of uniformly resettable servers. As we shall see uniform resettability can play a role in constructing universal user strategies, while it seems that non-uniformly resettable servers cannot play this role. Still, non-uniform resettability can be beneficial for achieving various goals, and this benefit may be inherited by universal strategies. Specifically, if a server tends to get stuck (or damaged by some effect of the environment), then being able to reset it (even by a server-specific message) is very beneficial.

5.3.1 One benefit of uniform resettability

The benefit of using uniformly resettable servers is demonstrated by considering the gap between Examples 4.12 and 4.21. Recall that both examples refer to solving computational problems posed by the world; specifically, instances of a decision problem D_0. In Example 4.12, we showed that *solvers for arbitrary computationally equivalent problems* (i.e., equivalent to D_0) can be used for solving D_0, by relying on a program checker for D_0. In Example 4.21 we showed that we can do better if both D_0 and its complement have interactive proof systems (with relatively efficient provers); in such a case, we can solve D_0 by using *any server that is helpful for solving* D_0. That is, we can benefit from a considerably wider class of servers.

Recall that interactive proof systems as required in Example 4.21 yields a program checker (for the same decision problem), but the opposite direction is widely believed to be false (because it would have implied $\mathcal{EXP} \subseteq \mathcal{PSPACE}$). This means that, in this context, benefiting from arbitrary helpful servers is harder than benefiting from all $\mathcal{D}$-solvers, where $\mathcal{D}$ is the class of problems that are computationally equivalent to D_0.

Here we note that the result of Example 4.12 can be extended to *any class of helpful servers that is uniformly resettable*. That is, we show if D_0 has a program checker, then D_0 can be solved by using *any resettable server that is helpful for solving* D_0. Indeed, this extends the result in Example 4.12, because the class of all $\mathcal{D}$-solvers is a very restricted class of helpful servers that are uniformly resettable.

Proposition 5.8 *Suppose that the decision problem D_0 has a program checker. Let $\mathcal{U}$ denote the class of user strategies that run in probabilistic polynomial-time, and $\mathcal{S}_0$ denote an enumerable class of uniformly resettable servers that are all $\mathcal{U}$-helpful* (with a bounded number of errors)[45] *for deciding D_0. Furthermore, suppose that the mapping $S \mapsto B$ is computable, where B is the error bounding function guaranteed for S. Then, there exists an $\mathcal{S}_0$-universal user strategy such that, for every $S \in \mathcal{S}_0$, when interacting with S, this universal strategy runs in probabilistic polynomial-time.*

Indeed, the additionally required mapping exists trivially in the case that $\mathcal{S}_0$ is a class of uniformly resettable servers that all have the same helpfulness-error bound.

Proof: The proof is a hybrid of the arguments used in Examples 4.12 and 4.21. As in both cases, we reduce the construction of a $\mathcal{S}_0$-universal strategy to the construction of an adequate user strategy for each server $S \in \mathcal{S}_0$. Furthermore, as in both cases, each such user strategy U is coupled with an adequate sensing function U' that is viable with respect to S and safe with respect to $\mathcal{S}_0$. In our construction we use a program checker for D_0 (just as done in Example 4.12), but only provide a relaxed viability guarantee (as in Example 4.21), because we use a wider class of helpful servers that (unlike in Example 4.12) includes servers that cause a bounded number of errors. Consequently, as in Example 4.21, we shall be using Theorem 4.20 (rather than Theorem 4.6, which was used in Example 4.12).

We mention that the enumeration of user strategies (required by Theorem 4.20) holds by definition of $\mathcal{U}$, whereas the mapping of users to the index of the corresponding server (which allows to obtain the corresponding error bound) can be obtained by replacing each possible user strategy U with the sequence $(U,1),(U,2),....$ Thus, we focus on constructing an adequate user strategy U and an adequate sensing function U' for every $S \in \mathcal{S}_0$, while assuming that we know the corresponding error bound (as well as the uniformly resetting message). This is done by following the approach of Example 4.12, which relies on using a program checker for D_0. In fact, following this approach, it suffices to show how to transform a resettable server into a memoryless strategy (i.e., a member of $\mathcal{F}$)[46] such that S is transformed into a D_0-solver.

[45] Here we refer to the notion of refined helpfulness, as defined at the end of §4.4.1.

[46] As in Example 4.12, $\mathcal{F}$ denotes the class of all memoryless strategies (i.e., strategies that maintain no local state).

The transformation, which amount to emulating a memoryless strategy by using a resettable strategy, proceed as follows. Let us assume first that this resettable strategy S is helpful without any errors, and consider the corresponding user strategy $u(S)$ that uses it. Recall that we are trying to emulate a memoryless strategy that is supposed to be a D_0-solver, while the messages that we attempt to answer come from the program checker (which the strategy U invokes, on input a message received from the world). Upon receiving a new message (which is an instance of D_0), we reset the server, and start a new communication session using the said message as if it were received from the world. (Note that we know the corresponding resetting message.) When we (or rather the corresponding user $u(S)$) detect that the communication session is completed (i.e., that $u(S)$ has determined its answer to the world), we use the corresponding answer (i.e., the answer that $u(S)$ would have sent to the world) as our response to the original message. We stress that all this activity is oblivious towards the world; that is, we create sessions that do not exist with respect to the world, while the server is unaware of their "unreal" nature (since by this goal's definition the world only communicates with the user, whereas the world neither sends messages to the server nor expects any messages from it).

Indeed, the foregoing transformation converts any resettable server strategy into a memoryless strategy, because in each emulation of the latter we invoke the former on the same initial local state (via resetting) and communicate with it using the same strategy $u(S)$. Furthermore, if $S \in \mathcal{S}_0$ makes no errors when communicating with $u(S)$, then S is transformed into a D_0-solver.

It is still left to deal with the case that $S \in \mathcal{S}_0$ makes a bounded number of errors when communicating with $u(S)$. The problem is that we only used the first session in the interaction of S with $u(S)$, whereas this session may always produce a useless (e.g., random) answer. The solution is to let $u(S)$ engage in many sessions with S, all regarding the same instance, and rule by majority, where the number of such sessions is significantly larger than the (expected) error bound, denoted b. Specifically, in our i^{th} emulation we reset the server $O(\log i)$ times, and after each resetting we run $3b$ ("unreal") sessions, all regarding the same instance that equals the i^{th} orignal message of the world (which we aim to answer). As our answer, we take the majority vote among these $O(\log i)$ trials, where each trial takes a majority vote among its $3b$ ("unreal") sessions. Thus, the probability that we err in our i^{th} emulation is at most $1/(i+1)^2$, because each of the $O(\log i)$ trials errs with probability at most $1/3$. It follows that the *expected* total number of errors that the emulated D_0-solver commits is bounded by a constant (i.e., indeed, the constant 1 will do).

This completes the presentation of the transformation of every $S \in \mathcal{S}_0$ into a memoryless strategy, which is used by a corresponding user strategy, denoted U. Like in Example 4.12, we couple U with a corresponding sensing function U', which uses the checker's output in the corresponding invocation (which corresponds to a real session that is initiated with a D_0-instance, selected by the world). This U' is $O(1)$-viable with respect to (U, S), and is safe with respect to U and the class of all resettable servers (which contains $\mathcal{S}_0$). This would suffice for a version of Theorem 4.20 that requires relaxed strong viability and weak safety (i.e., viability as in Theorem 4.20 and safety as in Theorem 4.6). The current proposition follows.[47] ■

[47]Note that U' is not quite strongly safe (with respect to the latter class), because the number of rounds used in each real session grows with its index (since we use increasingly more "unreal" sessions in our emulations), and for that reason we cannot apply Theorem 4.20 as is. The benefit in using Theorem 4.20 is that it provides an error bound for the universal strategy, a bound not stated in the current result. We mention that we could have obtained stronger results in a variety of natural cases. For example, if the size of the session's initial state (i.e., the length of the instance) grows such that the i^{th} real session refers to size $\Omega(\log i)$ and if every bound B is polynomial, then we

Digest. Proposition 5.8 demonstrates the benefit of (uniform) resetting. The effect of resetting occurs at two levels. Most conspicuously, resetting is used to emulate a memoryless server strategy, and the benefit is that a memoryless strategy may cause less harm than an arbitrary strategy. In particular, the damage caused by improper communication with a memoryless strategy is confined to the period of improper communication, and does not propagate to the future.

5.3.2 On the benefit of exploration

In this section we demonstrate the benefit of exploration (or rather exploration sessions as defined in §5.2.2) in the context of resettable servers. Indeed, one may also view the following result as another demonstration of the benefit of resettability, so it seems fair to credit both features with the following demonstration.

Recall that all prior (quantitative) universality results upper bound the number of errors as a function of the state size (indeed, see Theorems 4.17, 4.20, and 5.3). The corresponding universal strategies switch away from a failing user strategy (which they emulate) as soon as they sense many errors, where these errors occur with respect to the actual goal that the user attempts to achieve. Instead, it would have been nice to cause less errors with respect to the actual goal, even at the expense of causing errors in some "side experiments" (indeed, explorations), while not slowing down progress on the actual goal by too much. We shall actually do even better with the strategy presented below.

The observation that underlies the following universal strategy is that the failure of a specific user strategy (with respect to a fixed server) can be accounted to some fixed state size (i.e., the minimal state size that causes failure). So if we can experiment with various state sizes, in parallel to doing our "real" activity, then we may be able to abandon a bad user strategy while causing a number of errors that is related to this fixed size (rather than being related to the potentially larger size with which we are actually concerned). These parallel attempts are performed in exploration sessions, and the formalism of Definition 5.6 guarantees that, from the world's point of view, these explorations do not effect the "real" sessions and vice versa. Furthermore, resetting will be used so that all sessions (real or exploratory) look the same to the server, and so the server behaves in these explorations exactly as it would have behaved in real sessions.

Theorem 5.9 (universality via exploration): *Let $G = (\mathcal{W}, R)$, $\mathcal{U}$, $\mathcal{S}$, ϵ and $\mathcal{B}$ be as in Theorem 4.20, except that here we refer to the varying-size generalization as in Theorem 5.3. Suppose that G is a multi-session goal with exploration sessions, that $\mathcal{S}$ is a set of uniformly resettable servers, and that each strategy in $\mathcal{U}$ resets the server at the beginning of each new session. Further suppose that the number of rounds in a session of G is monotonically non-decreasing with the relevant size, and that the number of start-session states of any specific size is finite. Then, there exists an $\mathcal{S}$-universal user strategy U such that for every $S \in \mathcal{S}$ there exists a constant b such that (U, S) robustly achieves the goal G with b errors, where here we refer to error counts in the simple sense as in Definition 4.15* (and, e.g., in Theorem 4.17). *Furthermore, the number of rounds spent in exploration sessions never exceeds a constant fraction of the total number of rounds.*

The constant b depends on the smallest size, denoted s, for which each of the prior user strategies tried for interacting with the server S fails. Specifically, b is proportional to the number of initial states (i.e., initial contents of start-session states) that have size at most s. We note that while

can obtain a polynomial safety bounds.

in many settings the said number is exponential in s, there are settings in which the number is polynomial in s (e.g., the world may ask us to solve decision problems that relate to a sparse set). We also note that the helpfulness of the servers in $\mathcal{S}$ holds with respect to a class of users that reset the server at the beginning of each session. While this class of user strategies is somewhat restricted, it seems natural to expect "helpful" severs to be helpful also with respect to that class.

Proof Sketch: We focus on the special case in which the user strategies (in $\mathcal{U}$) carry no state across sessions and the world never chooses the same initial state for two sessions. This case is quite natural, and allows us to present all the essential ideas.

Recall that the universal strategies used in all our proofs proceed by enumerating all strategies in $\mathcal{U}$ and emulating each U_i until encountering a sufficient number of failure indications. We do essentially the same, except that we try to maintain a balance between the number of rounds used in real sessions and the number of rounds used in exploration sessions. In a sense, our strategy can be described in terms of an imaginary world strategy that introduces such exploration sessions. Furthermore, the exploration sessions are invoked such that all possible initial states (i.e., initial contents of start-session states) of a certain size are used before we use states of larger size. We stress that this activity is done in parallel to the execution of real sessions that are initiated by the real world. Specifically, upon terminating the execution of a real session (initiated by the real world), the imaginary world (or rather our user) initiates new exploration sessions until the number of rounds used by all exploration sessions exceeds half the total number of rounds so far.

Noting that the foregoing execution necessarily refers to varying state sizes, we adopt the switching criterion used in the proof of Theorem 5.3. That is, we switch from emulating U_i to emulating U_{i+1} as soon as reach a round t_{i+1} such $\sum_{t\in(t_i,t_{i+1}]:U'_i(\sigma_t)=0}\frac{1}{B_i(\mathsf{s}(\sigma_t))}$ exceeds 1. We stress that this acounting (and corresponding switch) is done across all sessions, real sessions and exploration sessions alike.

A key observation regarding this interleaved execution of real sessions and exploration sessions is that each of these (partial) executions is oblivious of the other. This follows by the fact that (by hypothesis) each strategy $U_i \in \mathcal{U}$ starts each session by resetting the server. Thus, we can decouple the aforementioned interleaved execution into a real execution (containing no explorations) and an auxiliary executions that consists of all exploration sessions. Furthermore, the auxiliary execution is totally determined as a sequence of all possible explorations, ordered according to their initial states (where states of smaller size appear before states of larger size).

Another key observation is that if (U_i, S) does not achieve the goal G (or rather its real part), then there exists a constant s_i such that the contribution of exploration sessions having (initial state of) size at most s_i to the sum $\sum_{t\in\mathsf{N}:U'_i(\sigma_t)=0}\frac{1}{B_i(\mathsf{s}(\sigma_t))}$ exceeds 1. This follows from the fact that events that would occur in a real execution (of real sessions) will also occur in the auxiliary execution (of exploration sessions) whereas each of these executions consists of a sequence of mutually oblivious sessions, together with our assumption that the world only uses each initial state at most once, so the corresponding (finite) set of initial states witnesses the failure of U_i with S.

We may now invoke Theorem 5.3, while considering only the exploration sessions. While these partial executions do not necessarily satisfy the additional technical requirements regarding $\mathcal{B}$, the reasoning underlying the proof of Theorem 5.3 applies, and so we may infer that the accumulated contribution of the rounds containing errors (in which exploration occurs) is upper-bounded by a constant that depends on $s = \max_{j\in[i-1]}\{s_j\}$, where (U_i, S) achieves the goal (and the s_j's are as above). Specifically, the number of errors in these exploration sessions is bounded by i times the number of different initial states of size at most s, because errors occur only at the end of sessions.

We need to show that the number of errors that occur in real sessions can also be bounded in terms of s.

The last claim is shown by noting that the number of real sessions of size at least s that took place in the said period does not exceed the number of exploration sessions. This follows by the monotonicity hypothesis (which implies that the number of rounds taken by each session of size at least s is no less than the number of rounds taken by any session of size at most s). Thus, the total number of errors (both in exploration and real sessions) is bounded by twice the aforementioned upper bound (on the number of errors in exploration sessions). The theorem follows for this special case (i.e., where $\mathcal{U}$ consists of strategies that, unlike the universal strategy that we presented, carry no state across session).

When dealing with the general case the failures at the various sessions are not fully determined by the initial state of this session (and the strategies employed) but can be affected by the history of the execution at the user's end. (Indeed, the world's behavior is independent of the history and the same holds with respect to the server that is being reset by the user at the beginning of each session, but the user strategy may depend on previous sessions.) The solution to this problem is to enumerate finite sequences of sessions with start-session state having size that does not exceed a specific bound. We treat each such finite sequence as we treated a single session before; that is, our exploration sessions now consist of sequences of sessions taken from this enumeration. We stress that the emulated user strategy is reset at the begining of each such sequence of exploration sessions, but the real execution maintains state across the real sessions. The enumeration guarantees that any finite sequence of sessions that causes too many failures with respect to startegy U will be encountered in finite time and cause the universal strategy to abandon U after a finite number of sessions. The theorem follows. ■

5.4 Partial robustness

As hinted in Section 3.3, the notion of robustly achieving a goal (see Definition 3.14) is too strong for the study of one-shot goals. Recall that this definition mandates that the goal is achieved no matter which global state the system is initiated in. In general, a more refined definition that quantifies over a subset of all possible global states is desirable, because it corresponds to natural settings and offers greater definitional flexibility. Most importantly, this refined definition allows to consider the (natural case of the) set of all global states in which the user's local state is reset to some initial value. (Indeed, in contrast to resetting the world, resetting the user seems feasible in many cases, and seems less demanding than resetting the server.)

We relax (and generalize) the notion of robustly achieving a goal by quantifying over a predetermined set of states (rather than over all states). We also allow an explicit specification of the success probability (rather than insisting that the success probability equals 1).

Definition 5.10 (robustly achieving goals, revised): *Let $\Theta \subseteq \Omega$ and $p : \mathsf{N} \to [0,1]$. We say that a pair, (U,S), of user-server strategies (Θ,p)*-robustly achieves the goal *$G = (\mathcal{W},R)$ if for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Theta$ a random execution of the system (W,U,S) starting in state σ_1 is successful with probability at least $p(\mathsf{s}(\sigma_1))$.*

Definition 3.14 is obtained as a special case of Definition 5.10 by letting $\Theta = \Omega$ and $p \equiv 1$.

6 One-Shot Goals

Our focus so far has been on goals that refer to infinite executions. Note, however, that the natural case of multi-session goals implicitly refers to "sub-goals" that should be achieved within a finite number of rounds (i.e., the corresponding session). In this section, we focus on such sub-goals, which when viewed in isolation are called "one-shot" goals. We highlight the fact that, in this context (i.e., w.r.t "one-shot" goals), sensing is necessary for achieving goals (or, in fact, sensing is actually implicity in achieving goals).

6.1 Definitional treatment

The most natural definition of one-shot goals refers to finite (i.e., terminating) executions, and thus does not fit our main terminology (as presented in Section 3.1). Nevertheless, one-shot goals can be viewed as a special case of general goals (see §6.1.1), where this case is closely related to (but different from) to a special case of multi-session goals. In addition, in §6.1.2 we provide a direct (stand-alone) treatment of one-shot goals.

6.1.1 One-shot goals as a special case of general goals

Definition 6.1 (one-shot goals): *A goal $G = (\mathcal{W}, R)$ is called a* one-shot goal *if the following conditions hold.*

1. The world's states: *The local states of the world are partitioned into two non-empty sets consisting of* non-terminating *states,* terminating *states. The initial local state belongs to the set of non-terminating states.*
2. The referee suspends its verdict until reaching a terminating state: *The referee R is compact. Furthermore, the corresponding function R' evaluates to $\perp$ if and only if the current state is a non-terminating state.*
3. Termination: *When being in a terminating state, the world just maintains its state; that is, for each actual world strategy $W \in \mathcal{W}$, and each terminating state σ it holds that $W(\sigma) = \sigma$.*

When being in any non-terminating state, the world follows its strategy as usual.

Thus, a typical execution of a system that refers to a one-shot goal consists of an actual finite execution that enters a terminating state, which is artificially propagated by an infinite sequence of repetitions of this state. It follows that *an execution is successful if and only if it enters a terminating state that evaluates to 1* (under R').

Robustly achieving one-shot goals. As hinted in Section 3.3, the notion of robustly achieving a goal (see Definition 3.14) is too strong for the study of one-shot goals, because no execution that starts in a terminating state that evaluates to 0 (under R') can be successful.[48] Thus, we must relax the notion of robustly achieving a goal such that starting in such states is not considered. Hence, our starting point is Definition 5.10, which offers a general refined notion of robustly achieving

[48]The same holds for any global state σ that causes the world to immediately enter such a terminating state due to the messages currently in transit. We consider this case specifically since the world usually terminates the session in response to a message that the user sends.

a goal that quantified over a predetermined set of states rather than over all states. Indeed, the flexibility provided by Definition 5.10 provides a good basis for defining robustly achievable one-shot goals. Specifically, we let Θ consist of all global states in which the current user's local state is empty and the world's next local state is non-terminating. (That is, we wish to avoid not only states σ such that $\sigma^{(\mathtt{w})}$ is terminating, but also states σ that lead the world to a terminating state in the next move, due to messages in transit.)

Definition 6.2 (robustly achieving one-shot goals): *For a one-shot goal $G = (\mathcal{W}, R)$, we say that a global state σ is* doomed *if for every $W \in \mathcal{W}$ it holds that $W(\sigma)^{(\mathtt{w})}$ is terminating, and we assume that G has states that are not doomed. Letting μ denote an unspecified negligible function, we say that a pair of user-server strategies, (U, S),* robustly achieves the one-shot goal *$G = (\mathcal{W}, R)$ if it $(\Theta, 1-\mu)$-robustly achieves the goal $G = (\mathcal{W}, R)$ for Θ that contains all global states in which the user's local state is empty and the world's local state is not doomed. If $\mu \equiv 0$, then we say that (U, S) robustly achieves the one-shot goal $G = (\mathcal{W}, R)$* in a perfect manner*. We stress that if $\Theta = \emptyset$, then the goal G is not* (robustly) *achievable.*

The foregoing adaptation of robust achievability to one-shot goals supports the following adaptation of Proposition 3.15.

Proposition 6.3 *Let U, S and S_t be as in Proposition 3.15, and let U_t be a user strategy that plays the first t rounds using the user strategy U_0, then resets its local state to empty and plays all subsequent rounds using the user strategy U. Then, if (U, S) robustly achieves the one-shot goal $G = (\mathcal{W}, R)$, then so does (U_t, S_t).*

The proof proceeds as in the case of Proposition 3.15, while relying on the modified robust achievability hypothesis (which matches the modified construction of U_t).

User-sensing in the context of one-shot goals. The notion of strongly *viable* user-sensing (cf. Definitions 3.17 and 3.18) is adapted to the current context in a way analogous to the adaptation applied to the notion of robustly achieving (cf. Definition 6.2 versus Definition 3.14). That is, a user-sensing function for a one-shot goal is considered strongly viable if the condition in Definition 3.17 holds for every $\sigma_1 \in \Theta$ (rather than for every $\sigma_1 \in \Omega$) and with probability $1-\mu$ (rather than with probability $2/3$).

As for the *safety* condition, its formulation is greatly simplified by the special features of one-shot goals (i.e., the fact that the world's state does not change once it enters a termination state). In particular, the difference between Definitions 3.18 and 3.17 disappears, and it suffices to refer to the value of R' at termination states. As a consequence of safety referring to the value of R' on terminating states though, the delay period for sensing must be eliminated, or else a "safe" sensing function could provide a failure indication *after* the session has terminated; actually, quite contrary to allowing any delays in sensing, recalling that the user often terminates a session by sending the world a message, we see that in order for sensing to be of use, the sensing function must *predict* the value of R' on the subsequent round (e.g., from a doomed state). Finally, for consistency with the foregoing adaptation, we also adapt the strong safety condition (cf. Definition 3.18) such that it holds with probability $1-\mu$ (rather than with probability $2/3$). The modified definitions are presented in their entirety in Definition 6.4 (for the direct definitional treatment of one-shot goals presented in §6.1.2).

Deriving multi-session versions of one-shot goals. For every one-shot goal, we can derive a natural multi-session version by letting the new world initiate a new session each time the original (one-shot) world enters a terminating state. Note that, in the derived multi-session goal, we may only expect to succeed in a $1-\mu$ fraction of the sessions (where μ is the error probability allowed in Definition 6.2), whereas achieving a multi-session goal requires succeed in all but finitely many sessions. In order to overcome this difficulty, we extend the definition of one-shot goals by allowing the user to control the success probability. Formally, we consider a uniform family of user strategies, $\overline{U} = \{U_i\}_{i\in\mathsf{N}}$, along with a uniform family of negligible functions, $\overline{\mu} = \{\mu_i\}_{i\in\mathsf{N}}$ (e.g., $\mu_i(n) = \mu(n)/i^2$ for some negligible function μ), and require that (U_i, S) (Θ, μ_i)-robustly achieves the goal (for every $i \in \mathsf{N}$). An analogous adaptation will be applied to the user-sensing function.

6.1.2 A direct definition of one-shot goals

A direct definitional treatment of one-shot (finite) goals is derived by degenerating the definitional treatment of infinite goals provided in Section 3. Thus, we consider finite executions of the system (consisting of a user, a server, and the world), which raises the question of when does such an execution terminate.

Termination. Recall that termination was defined in §6.1.1 (see Def. 6.1), but that notion of termination was a labeling of the states of the world that was used for the purpose of defining the temporal decision R', while the actual execution never stops formally (but does get "frozen" from the world's point of view). Here, instead, we consider an actual termination, and so it is most natural to define termination according to the user state. We may assume, without loss of generality, that the user notifies the world whenever it is about to terminate (i.e., the last message sent by the user indicates its termination), and so the world may terminate in the same round. (Note that we cannot assume that the server terminates in the same round, since it may not perfectly understand the user even if the goal has been achieved with its help.)

Thus, the notions of strategies and executions remain intact, except that now executions are typically finite. That is, we consider goals in which, for each choice of the actual world strategy, with high probability the corresponding system halts in a finite number of rounds. This refers to all definitions in Section 3.1.

Since we are going to define achieving a goal and sensing it according to the value of the referee at the time of termination, we do not need the notion of compactness and the corresponding temporal decision function (see Section 3.2).[49]

Robustly achieving one-shot goals and sensing it this context. The relevant definitions are very similar to those presented in §6.1.1, except that here we refer to finite executions. This is straightforward with respect to the definition of robustly achieving one-shot goals (i.e., Def. 6.2). The only thing to bear in mind is that *success refers to the world's state at termination time.* Turning to user-sensing in the context of one-shot goals, there is more to say. The most important thing is that while the following definition has a "strong" flavor (as Definitions 3.17 and 3.18), it directly refers to the referee (and does not refer to the non-existing temporal decision function). For sake of self containment, we spell out this definition.

[49]Alternatively, one may say that all finite goals are compact and that the corresponding temporal decision function R' is derived from the referee R such that $R'(\sigma) = R(\sigma)$ if σ is a terminating state and $R'(\sigma) = \perp$ otherwise.

Definition 6.4 (user sensing in the one-shot context): *Let $G = (\mathcal{W}, R)$ be a one-shot goal, S be a server strategy, and Θ be as in Definition 6.2. The predicate $U' : \Omega \to \{0,1\}$ is $(1-\mu)$-*safe *with respect to (U,S)* (and G) *if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Theta$, letting $\overline{\sigma}$ denote a random execution of the system (W,U,S) starting at state σ_1, with probability at most $\mu(\mathsf{s}(\sigma_1))$, it holds that both $R(\sigma_t) = 0$ and $U'(\sigma_t) = 1$, where σ_t is the state at termination time. The predicate U' is $(1-\mu)$-*viable *with respect to (U,S) if, for every $W \in \mathcal{W}$ and every $\sigma_1 \in \Theta$, with probability at least $1 - \mu(\mathsf{s}(\sigma_1))$, it holds that $U'(\sigma_t) = 1$ holds.*

We will usually take μ to be a negligible function, and talk of safety and viability instead of $(1-\mu)$-safety and $(1-\mu)$-viability, respectively. Indeed, if U' is viable and safe with respect to (U,S) (and G), then (U,S) robustly achieves the goal G.

Deriving multi-session versions of one-shot goals. This is done as in §6.1.1, except that here the modification applies to an execution that actually terminated rather than to one that is only considered as terminating.

6.2 A tighter relation between sensing and universality

Universality results analogous to those presented in the previous sections (e.g., Theorem 5.3) can be proved for one-shot goals; in fact, the proofs are simpler in the current context. Note that these universality results assume a stronger notion of sensing (as introduced here) and thus may yield a stronger universal strategy, which is "aware" of its success in achieving the goal. This awareness is reflected in the fact that, with high probability, when the user strategy halts it must be the case that the (one-shot) goal is achieved. Thus, in the one-shot goal, there is a tighter relation between sensing and universality.

Firstly, we note that, any user strategy U that is universal with respect to a server class $\mathcal{S}$, yields a sensing function that is safe and viable with respect to U and $\mathcal{S}$. Specifically, this sensing function is identical to the halting predicate that underlines U (i.e., the predicate that determines whethere or not U halts after each round). Secondly, by employing an argument analogous to (but simpler than) the one used in the proof of Theorem 4.26, we may infer that this sensing function is safe with respect to all servers (assuming that the original server class $\mathcal{S}$ includes all helpful servers).

7 The Symmetric Model

Recall that our main treatment of goals was confined to the asymmetric case in which the user has a goal, while the server is only there to help it. That is, the goals treated so far are actually the "user's goals," whereas the server had no goal of its own (beyond, maybe, assisting the user). In this section we consider the general case in which both parties may have their "individual goals;" for example, even if we think of users and servers, the vanilla setting studied so far can be extended to account for the possibility that the server wishes to obtain some compensation for its help (i.e., the server also has some individual goal).

More generally, we consider a pair of parties having arbitrary individual goals. Indeed, it may be impossible to simultaneously achieve both these individual goals, but recall that also in the asymmetric (user–server) setting there are goals that cannot be achieved. As in the latter setting, our focus is on goals that can be achieved (when both parties use adequate strategies), and the question that we study is whether and under what conditions these goals can be achieved via interaction between parties that are initially incompatible (i.e., do not understand each other a priori). In this section we outline a treatment of this question.

Parties, goals, compactness, and achieving. The basic framework remains unchanged, except that the parties (other than the world) are now treated symmetrically. In general, we may have $n \overset{\text{def}}{=} m-1 \geq 2$ such parties. More importantly, the definition of the referee R is extended such that it maps global states into n-ary Boolean vectors, where the i^{th} coordinate refers to the personal goal of the i^{th} party. The same applies to the temporal decision function R' (underlying the definition of compactness). An n-tuple achieves the i^{th} personal goal if a random execution of the corresponding system evaluates (under R) to a vector having 1 in the i^{th} coordinate, and the global goal is achieved if all n personal goals are achieved.

Recall that we avoid the question of which global goals are achievable, and focus on (personal and global) goals that are achievable, investigating the possibility of achieving them in the presence of initial misunderstandings. Note that we did exactly the same in the asymmetric setting, but there this attitude felt less alarming because there was no room for conflict between the personal goals of the parties.[50] Thus, our focus is again on the communication between the n parties, and the possibility of meaningful communication.

Sensing. Again, the basic approach remains unchanged, and is merely extended to n parties. Needless to say, when dealing with the i^{th} party, the result of its sensing function is compared against the i^{th} coordinate of the temporal decision function R'. Note that safety and viability are defined with respect to a tuple of $n-1$ strategies (for the other $n-1$ parties), and with respect to a class of such tuples.

Helpfulness. This notion generalizes naturally too, provided that we confine ourselves to helpfulness with respect to achieving the personal goals of individual parties. To clarify the issue, let us focus on the case of two parties (i.e., $n=2$), and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be classes of strategies for these two parties. For $i=1,2$, we say that $P \in \mathcal{P}_{3-i}$ is $(i,\mathcal{P}_i)$-helpful if there exists $Q \in \mathcal{P}_i$ such that (P,Q) achieves the personal goal of Party i. However, it is not necessarily the case that Q is $(3-i,\mathcal{P}_{3-i})$-helpful. Furthermore, even if $P_{3-i} \in \mathcal{P}_{3-i}$ is $(i,\mathcal{P}_i)$-helpful and $P_i \in \mathcal{P}_i$ is $(3-i,\mathcal{P}_{3-i})$-helpful, it is not necessarily the case that (P_1,P_2) achieves the personal goal of any of the parties (let alone achieving the global goal). Nevertheless, all these reservations become irrelevant with respect to our results regarding universal strategies.

Universal strategies. Remaining in the confine of two parties (i.e., $n=2$), we say that a strategy for party i is universal with respect to its personal goal if it can achieve this goal when communicating with any party that is helpful (for that personal goal). In other words, if *some* strategy for Party i achieves the goal when communicating with the other party, then the universal strategy also achieves the goal when communicating with this party. That is, for $i=1,2$, we say that U is $(i,\mathcal{P}_{3-i})$-universal if it achieves the personal goal of Party i when communicating with any

[50]Recall that in the asymmetric setting the server has no personal goal (and the user's personal goal is identified with the global goal, which was just called 'the goal'). Still, the question of whether the user's goal is achievable arises in that setting too, and we have just focused on achievable goals. We do exactly the same here.

strategy $P \in \mathcal{P}_{3-i}$, where we definitely confine ourselves to classes $\mathcal{P}_{3-i}$ of $(i, \mathcal{P}_i)$-helpful strategies (for Party $3-i$).[51]

Theorem 7.1 (symmetric result, loosely stated): *Let G be a goal, $i \in \{1,2\}$, and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be classes of strategies for the two parties such that every strategy in $\mathcal{P}_{3-i}$ is $(i, \mathcal{P}_i)$-helpful* (in an enhanced sense).[52] *Then, there exists an $(i, \mathcal{P}_{3-i})$-universal strategy.*

Indeed, Theorem 7.1 is merely a restating of Theorem 2.1: Theorem 7.1 refers to achieving the personal goal of Party i. It is just that for a fixed goal G, the party that Theorem 7.1 refers to is not predetermined (as in Theorem 2.1). This opens the door to applying Theorem 7.1 to both parties (see next).

Note that if the hypothesis of Theorem 7.1 holds for both $i \in \{1,2\}$, then we obtain a pair of strategies (U_1, U_2) such that each U_i is $(i, \mathcal{P}_{3-i})$-universal. It follows that if $(U_1, U_2) \in \mathcal{P}_1 \times \mathcal{P}_2$, then (U_1, U_2) achieves the global goal. Thus, in such a case, if each party looks after its own interest (of achieving its personal goal), which calls for using a universal strategy, then the global goal is achieved as well unless one party chooses a universal strategy that is not helpful to the other party.

We stress that the aforementioned achievability of the global goal relies on the hypothesis that the universal strategies reside in the class of strategies that are helpful for the other party (i.e., for every $i \in \{1,2\}$, the $(i, \mathcal{P}_{3-i})$-universal strategy U_i is $(3-i, \mathcal{P}_{3-i})$-helpful). In other words, there is no free lunch here; the key to achieving the global goal is the parties' use of strategies that are helpful for the (goal of the) other party. The point is that this choice of a helpful strategy suffices; that is, the party need not know (a priori) the strategy employed by the other party.

Indeed, a natural question arises: *For which classes of strategies $\mathcal{P}_1$ and $\mathcal{P}_2$ as in Theorem 7.1 is it possible to have universal strategies that are helpful to the other party* (i.e., for every i, there exists an $(i, \mathcal{P}_{3-i})$-universal strategy that is $(i, \mathcal{P}_i)$-helpful).

Extension to any $n \geq 2$. Towards such an extension we need to generalize the notions of helpfulness and universality, where in both cases we will consider $(i-1)$-tuples of strategies for the other parties. Specifically, for $i \in [n]$, we say that $(P_1, ..., P_{i-1}, P_{i+1}, ..., P_n) \in \overline{\mathcal{P}}_{-i}$ is $(i, \mathcal{P}_i)$-helpful if there exists $Q \in \mathcal{P}_i$ such that $(P_1, ..., P_{i-1}, Q, P_{i+1}, ..., P_n)$ achieves the personal goal of Party i. Similarly, we say that U is $(i, \overline{\mathcal{P}}_{-i})$-universal if it achieves the personal goal of Party i when communicating with any tuple $(P_1, ..., P_{i-1}, P_{i+1}, ..., P_n) \in \overline{\mathcal{P}}_{-i}$.

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.

[51]Note that the index i in the definitions of $(i, \cdot)$-helpfulness and $(i, \cdot)$-universality refers to the index of the party who's personal goal is being achieved.

[52]The enhancement referred to here is analogous to the one arising from the various universality theorems in the asymmetric case.

[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 isomorphic 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 equals $\cap_{i>t} S'_i$, where $S'_i = \{\overline{\sigma} : R'(\sigma_i) = 1\}$ is a countable union of $E_{(i,\sigma)}$ such that $R'(\sigma) = 1$.

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 the complement of a countable union of the sets F_t, where F_t is the set of executions such that R' always evaluates to $\perp$ after time t (i.e., $\overline{\sigma} \in F_t$ if for every $i > t$ it holds that $R'(\sigma_i) = \perp$). On the other hand, F_t equals $\cap_{i>t} F'_i$, where $F'_i = \{\overline{\sigma} : R'(\sigma_i) = \perp\}$ is a countable union of $E_{(i,\sigma)}$ such that $R'(\sigma) = \perp$. ■

ECCC ISSN 1433-8092

http://eccc.hpi-web.de