Towards a Unification of Logic and Information Theory

Luis A. Lastras[1], Barry Trager[1], Jonathan Lenchner[1], Wojciech Szpankowski[2], Chai Wah Wu[1], Mark S. Squillante[1] and Alexander Gray[3]

[1]IBM T.J. Watson Research Center
[2]Purdue University
[3,2]Centaur AI Institute

January 6, 2025

Abstract

Today, the vast majority of the world's digital information is represented using the fundamental assumption, introduced by Claude Shannon in 1948, that "...the semantic aspects of communication are irrelevant to the engineering problem (of the design of communication systems)...". It is hard to overestimate the extraordinary positive impact of this assumption, which has allowed the design of flexible efficient and reliable communication systems that operate regardless of the intended meaning of our messages.

Consider, nonetheless, the observation that we, individuals, as well as our computing devices, often combine a received message with other information in order to deduce new facts (and hopefully make better decisions), thereby expanding the value of the originally received message. It is noteworthy that to-date, no rigorous theory of communication has been put forth which postulates the existence of deductive capabilities on the receiver's side.

The purpose of this paper is to present a proposal that combines information theory and logic at a fundamental level. We formally model such deductive capabilities using logic reasoning, and present a rigorous theory which covers the following generic scenario: Alice and Bob each have knowledge of some logic sentence, and they wish to communicate as efficiently as possible with the shared goal that, following their communication, Bob should be able to deduce a particular logic sentence that Alice knows to be true, but that Bob currently cannot prove. Many variants of this general setup are considered in this article; in all cases we are able to provide sharp upper and lower bounds phrased in terms of an entropy-like function that we call Λ, in reference to its apparent connection to problems of communication involving logic. Our contribution includes the identification of the most fundamental requirements that we place on a logic and associated logical language for all of our results to apply; an example is Propositional Logic over a finite number of propositions. Practical algorithms that are in some cases asymptotically optimal are provided, and we illustrate the potential practical value of the design of communication systems that incorporate the assumption of deductive capabilities at the receiver using experimental results that suggest significant possible gains compared to classical systems.

1 Introduction and summary of contributions

It is well known that a significant contribution of Shannon [64] was to provide a definition for information that is independent of the semantics of the message being conveyed. This abstraction is one of the most successful concepts in the computing and communication revolution, as it has allowed us to build flexible machines that process and communicate information in a standardized way, while the messages and intentions behind those messages remain removed from the operation of those machines. This aspect sometimes feels counter to one's intuition of what we think of as information, as articulated in 1949 by Warren Weaver, then Research Director of the Rockefeller Foundation, who breached the subject of semantics in information theory in an oft-cited commentary [74]. To capture some intuitive feel for the difference between Shannon's classical information and some desired notion of "semantic" information, we borrow from [16]: "As a very simple

example, pressing the keys of a computer keyboard at random generates a message that has a high syntactic information, because the generated symbols are approximately independent and uniformly distributed, so that their entropy (average information in Shannon's sense) is maximum. However, most likely, the generated message carries zero semantic information, as it does not carry any meaningful content."

Although Shannon's original theory is semantics-free, it was not because he had not been thinking about semantics. Six years prior to his 1948 paper, Shannon and fellow Bell Labs mathematician John Riordan considered the question of how compactly one could express a given Boolean function on n-bit inputs in Propositional Logic [61]. We therefore know that Shannon was thinking about the transmission of semantic information even before he developed his theory of communication.

Shortly after Shannon's seminal 1948 paper, debate began about what might constitute a satisfactory theory that addressed the semantic content of communication. An elegant proposal for incorporating semantics in a theory of information was made by Carnap and Bar-Hillel [7], which carries particular gravitas due to Carnap's status as one of the pioneers of the modern formal treatment of semantics via mathematical logic [17], along with the likes of Tarski [71] and Kripke [46]. This work systematically identified desirable properties for what semantic information should be, including how to measure it, and is the most seminal reference in all subsequent treatments of this problem.

Another early example of interest on the subject of semantic information can be found in a little-known paper due to Shannon himself [63]. To motivate his viewpoint, Shannon observed that a sentence may be encoded in multiple different ways, each recoverable from the other; for example, imagine a sentence and its translation to Morse code. Shannon then observed that "For most purposes of communication, any of these forms is equally good and may be considered to contain the same information". Shannon then concluded that "Thus we are led to define the actual information of a stochastic process as that which is common to all stochastic processes which may be obtained from the original by reversible encoding operations". In both [7, 63] we find the idea that if a sentence can be *deduced* from the sentences one already knows, it conveys zero information, but the more general implications of this observation to communication systems were left unexplored by these authors [70]. Later on, Shannon introduced Rate-Distortion theory [65], a general extension of information theory that allows for lossy compression. Although rate-distortion theory has had most of its practical influence on the compression of media such as images, video and audio, it is an extraordinarily general theory and correspondingly, it has found itself at the center at most of the efforts to extend classic information theory to semantics [50, 49, 66, 33, 69, 34, 55]. Additionally, the fundamental observation in [63] that information may admit partial ordering has received renewed interest [83, 84].

To illustrate the main driving point of view in our article, we will rely on an unrelated, but insightful quote. At the beginning of his Lectures on Physics [29], Feynman posed a hypothetical situation where all scientific knowledge is destroyed, and a sentence that has the most information in the fewest words needs to be chosen. His choice was "all things are made of atoms", which assumes that scientists would be able, through experimentation, induction and deduction, to reconstruct vast amounts of our scientific knowledge from the one sentence. In the absence of a deductive process, effectively conveying all scientific knowledge would seem to require a large quantity of bits to be transmitted. Yet Feynman's sentence can be transmitted with just a few dozen bits. In this paper we aim to provide the theoretical foundations for understanding this phenomenon – how something that can be so succinctly described can have such a profound consequence when paired with deductive reasoning.

Despite the success of classical information theory, pragmatic concerns have revived interest in the possibility that focusing on transmitting meaning accurately might offer savings over transmitting bits accurately [70], [44]. Motivated by the continued massive increase in the world's data and the need for next-generation network systems to somehow keep up, influential vision papers such as [16] have set off a recent explosion of interest in the promise of "semantic communication". However, surprisingly few have leveraged the deep original insights of Carnap/Bar-Hillel and arguably Shannon himself by invoking the power of *deduction*. This is likely due to the deep cross-disciplinarity needed to do so, requiring both sufficient depth in information-theoretic tools and in the formalisms of logic. The fact that mathematical logic serves as a foundation for much of computer science [77] and mathematics [78] can serve as some testament to its depth of development and difficulty to penetrate for the casual non-expert. Similarly, information theory is heavily developed mathematically and reliant on very different ideas, rooted in probability theory. A fundamental challenge in writing this paper has been to somehow make it accessible to its multiple possible audiences.

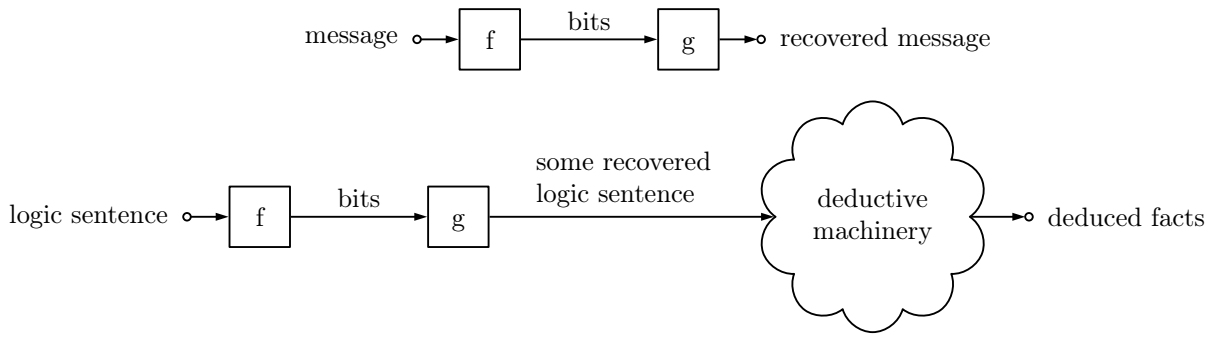


Figure 1: In the top, Shannon's original digital communication model. In the bottom, a sketch of our proposed extension of Shannon's model.

1.1 Our contributions

If nothing else, we ask the reader to remember this article through Figure 1, where at the top, we illustrate the famous digital communication model that Shannon introduced in [64], and in the bottom we sketch a version of this model where messages are replaced with logic sentences and where we assume the existence of a mechanism that allows the receiver to derive additional conclusions from whatever has been received. The remainder of the paper can be seen as a proposal for rigorously using logic to mathematically model such deductive mathematical machinery and how to derive Shannon-style bounds for the communication cost under a variety of different goals for the communication.

We begin our investigation connecting information theory to logic by thinking about a scenario where there are two parties, a sender (Alice) and a receiver (Bob), each of whom have in general different logic sentences, but such that the Alice's sentence entails that of Bob's. Furthermore, Alice wishes to convince Bob of the truth of a given logic sentence, that Bob cannot, with the information known to him, prove by himself. Furthermore, the problem is for Alice to send the *minimum* amount of information (in terms of bits sent) so that the Bob can prove what needs to be proved. A number of variants of the setting above are treated in this article – for example, Alice and Bob may share some common logic sentences, or each may know something the other doesn't know, or the goal of the communication may be for Bob to only prove a subset of what Alice can prove. In yet another aspect we treat, there may be logical inconsistencies between Alice and Bob's sentences, leading to a rudimentary model of misinformation.

Building on top of the foundations established by Carnap and Bar-Hillel [7], Shannon's Rate-Distortion theory [65], and the theories of source coding with side information due to Slepian-Wolf [68] and Wyner-Ziv [81] coding, we provide, for the first time, a rigorous theory that incorporates deductive reasoning directly in the communication process, providing sharp upper and lower bounds on communication cost under a wide variety of scenarios often showing significant efficiency gains compared to classic approaches. We also provide preliminary evidence of practical systems realizing a fraction of these possible gains. A pattern we found and subsequently used to guide our problem selection is the realization that the solution to all these problems had in common a simple scaled version of conditional entropy (see beginning of Section 3).

Our article is intended to serve as a bridge between the traditionally separate communities of information theory and logic. As a result, we devote special attention to the identification of the basic definitions that we found play a role in bridging between these fields, especially in logic and, in particular, in the sub-field of logic known as model theory. In an effort to provide as general conditions as possible for our information-theoretic results to apply, we first define the abstract notion of a *logic* – something that is rarely assayed in textbooks on the subject. We then introduce the notion of a *Logic System*, which is a logic taken together with a set of models of the different sentences of the logic, along with two maps, one taking a logical sentence to a set of models, and the other taking a set of models to a sentence – with the requirement that the latter map behave like a pseudo-inverse of the former map. Our development of the notions of a logic and of a logic system has some similarities to the treatment of model theoretic logics, introduced by Barwise and others in the 1980s [8]. However, our definitions are also unique and idiosyncratic since they are tailored to achieving our goal

of providing the minimum set of conditions for our information-theoretic results to apply. As part of that goal, we needed for Propositional Logic and First-Order Logic to be united under a common umbrella, again something that is rarely done, and something that required us to introduce a non-standard vocabulary for Propositional Logic. Further, the Propositional Proof System we provide is slightly different from the usual ones, and we need to make distinctions that are not always made between soundness and strong soundness, and between completeness and strong completeness. In the latter case we are led to introduce a new finitary variant of strong completeness that we have dubbed ω-strong completeness.

Finally, before settling on the traditional information-theoretic concept of bits as the communication efficiency metric, we studied other paradigms, including the idea of communicating to a receiver only logic sentences that he could not derive himself already; this then posed the interesting problem of how one could construct such sentences. To solve this problem, we explore connections beyond information theory and logic to include a third area of mathematics, which is the algebra of multivariate polynomials with variables and coefficients belonging to a finite field. Our insight in doing so is that Propositional Logic sentences can be represented using polynomials, which in turn provides us access to powerful mathematical tools such as Gröbner bases. We exploit these mathematical tools to establish fundamental results that support our general approach for synchronizing the knowledge between a receiver and a sender in a communication optimal way by having the receiver add new non-trivial sentences in its knowledge base of logic sentences (without changing its old sentences). More specifically, our general approach consists of first converting the original logic sentences into polynomials, then exploiting the foregoing mathematical tools to perform reduction and decomposition transformations in the polynomial domain, and finally converting the resulting polynomials back to logic expressions. We note that the basic technique of our general mathematical framework is also very general and has applicability beyond this article.

1.2 Relation to other treatments of semantic information

While reliance on mathematical logic is not universally leveraged in the field of semantic information, many authors do start with Carnap/Bar-Hillel's *logical probability*, one of many concepts introduced in their seminar work [7]. For the purposes of establishing contrast, here we too start with it assuming that the reader is familiar with the generalities of Propositional Logic. By means of example, assume two binary-valued propositions $\mathtt{X}_1$ and $\mathtt{X}_2$ and consider the logical sentence

$$\mathtt{s} = \neg \mathtt{X}_1 \vee \mathtt{X}_2. \tag{1}$$

There are four possible choices for these propositional variables $\mathcal{M} = \{00, 01, 10, 11\}$ but only for three of those does $\mathtt{s}$ happen to be true, namely $\{00, 01, 11\}$. In general, the subset of $\mathcal{M}$ where a sentence $\mathtt{s}$ is true is defined in [7] to be the *range* of that sentence; in our article, a generalization of this concept to general logics will be called the *kernel* of $\mathtt{s}$ and will be denoted by $\kappa(\mathtt{s})$. Assume a distribution P_μ over $\mathcal{M}$ and let μ be drawn according to such a distribution. Then the *logical probability* of $\mathtt{s}$, relative to the distribution of μ, is given by

$$P_\mu\left([\mu \in \kappa(s)]\right), \tag{2}$$

which informally is sometimes referred to as "the probability that the logic sentence is true", and a measure of the semantic information in $\mathtt{s}$, denoted in [7] by $\mathtt{inf}$, is defined by

$$\mathtt{inf}(\mathtt{s}) \stackrel{\Delta}{=} \log_2\left(\frac{1}{P_\mu\left([\mu \in \kappa(s)]\right)}\right). \tag{3}$$

In some treatments of semantic information (see, for example, Bao, Basu, et al. [4, 5, 10]), each element of $\mathcal{M}$ is regarded as a possible "meaning"; in our earlier example (1), "01" is such a meaning. In this article *we make no such choice*. For us, one has conveyed the semantic content of a logic sentence $\mathtt{s}$ if a receiver is able to infer from whatever is conveyed exactly the same that a sender can infer; as we will develop rigorously in this article, this can only be done if and only if $\kappa(\mathtt{s})$ is reproducible by the receiver; in this case, we state that $\kappa(\mathtt{s}) = \{00, 01, 11\}$.

For now, informally, define $\mathcal{L}$ to be the set of all possible sentences. Instead of assuming a distribution P_μ over $\mathcal{M}$, assume a distribution P over *subsets* of $\mathcal{M}$. Then, an optimal code length (in bits) for sending

such essentials in $\mathtt{s}$ in the above sense, relative to P, is given by

$$\log_2 \left(\frac{1}{P(\kappa(\mathtt{s}))} \right). \tag{4}$$

This quantity has a strong operational significance in the sense of measuring the optimal cost of transmission if one wishes to send to a receiver the information necessary for it to be able to infer whatever the sender can infer from $\mathtt{s}$ (Theorem 2), and is a reasonable example of an early (but not the only) idea in our paper. It is important to note that *no other such definition will carry this strong operational significance.* By taking the approach of defining the semantics of a sentence via kernels, we have been able to rigorously derive a large collection of results where a receiver is able to reproduce all or a subset of the mathematical facts that a sender can infer, under a variety of assumptions of what prior logic sentences each has access to, including even settings where the sender is unaware of what the receiver knows.

In contrast, a large segment of the community approaches the problem of semantic information without a direct linkage to logic. For us, a particularly relevant set of prior works are those who use Shannon's rate-distortion theory to set up problems involving semantics; in fact one could reasonably argue that Shannon's original foray into lossy coding [65] is an early example of exploiting semantics in compression. This theory is so general that almost any conception of semantics can be retrofitted to it. The work of Liu et al. [49, 50] as well as the follow-up work by Stavrou and Kontouris [69] and Guo et al. [33], for example, explicitly model a data source as comprising *intrinsic* (unobservable) and *extrinsic* (observable) components and proceed to derive information-theoretic bounds for desired approximations to either of these using general distortion measures. Another example is Shao et al. [66] who model an end-to-end semantic communication process that starts with some intended meaning which is (stochastically) transformed into some expressed language, which may then experience a form of semantic noise as it is received; the authors then argue for the use of joint source-channel coding techniques for designing optimal communication systems. In contrast our work is singularly focused on exploring in rigorous depth communication assuming the existence of reasoning engines. Recently, a general theory of semantic information that draws parallels to Shannon's lossless source, channel and lossy source coding theorems has been proposed by Niu and Zhang [54, 55]. Our work can be thought of as a depth-first, rather than breadth-first, work where the angle being explored is characterized by the possibility of deductive inference at the receiver's side. Our approach is rewarded with very sharp insight in this context, including the discovery of the role of the Λ function in the characterization of a broad set of problems involving various communication paradigms.

In this last respect, the reader will notice that for some of our work involving settings where Alice is not fully aware of what logic sentence Bob possesses, we rely on the idea of multiple rounds of communication. This setup has similarities to problems involving communication complexity, famously introduced by Yao [82] in 1979; see also Papadimitriou and Sipser [59] and related work on interactive communication, for example Orlitsky [57]. To establish contrast, we note that in our work, there is no pre-agreed function that Alice (or Bob) wants to compute.

A rather different take on the problems above can be found in the work of Juba and Sudan [42], who consider the problem of communication between sender and receiver when there has been no previous agreement on protocol and where the main difference between them lies on their computational power. Compared to [42], our work follows much more closely the usual conventions in information theory where sender and receiver do agree on elementary matters such as how information will be encoded and decoded.

In this subsection, we have only covered a subset of relevant works. We refer the reader to Gündüz et al. [34] for a broad survey on the subject.

1.3 Outline of the remainder of the paper

The Mathematical Preliminaries (Section 2) provide the fundamental notion of a Logic System, fully discussed in the Logical Underpinnings (Section 6), and also include the basic definitions of how the communication system is set up. We then provide a summary of our information-theoretic contributions (Section 3), leaving the formal statement and proof of these results to the latter Section 5. Theorem 2 contains the result supporting the discussion above surrounding (4); for clarity of exposition, the rest of the theorems in Section 3 solely contain the simpler single-letter upper bounds which are tight when additional i.i.d. assumptions are made. We develop practical methods, including linear and nonlinear codes, in Section 4, where we also

include experiments on synthetic data demonstrating significant possible gains over classical systems. We then provide a treatment of logic and a form of efficient communication from the standpoint of the algebra of polynomials on finite fields in Section 7. Speculative future directions are included in Section 8, followed by concluding thoughts in Section 9.

2 Mathematical preliminaries

2.1 Logic Systems

Suppose we are given a logic L (for example, Propositional Logic on a fixed set of $\mathtt{m}$ variables, or the First-Order Logic of graphs). We shall give a more complete definition of what we mean by a logic in Section 6, but for the time being it suffices to note that a logic L specifies a syntax, or set of rules for constructing valid logic sentences starting from a particular logical vocabulary, τ, and also provides a proof system, or rules of inference, for deducing the truth of new sentences assuming the truth of other sentences. Let us denote the set of well-formed sentences for the logic L by $\mathcal{L}$. We call $\mathcal{L}$ the *language* associated with the logic L. We will use the `typewriter font` to denote a logic sentence $\mathtt{s} \in \mathcal{L}$. The logic L may come equipped with a set of axioms, σ, or sentences of $\mathcal{L}$, that are assumed to be true without proof. For two logic sentences $\mathtt{s}_1, \mathtt{s}_2 \in \mathcal{L}$, we write $\mathtt{s}_1 \vdash \mathtt{s}_2$ if, assuming the truth of $\mathtt{s}_1$, it is possible to prove $\mathtt{s}_2$ in the logic L, possibly with the assistance of some of the sentences in σ. The empty sentence is considered to be well-formed and always true. We therefore write $\vdash \mathtt{s}$ if and only if (iff) s can be proven directly in L, starting from the axioms σ. Then $\vdash$ is a relation defined among pairs of elements of $\mathcal{L}$. We call $\vdash$ the "entailment" relation for the logic L. The definition of entailment extends naturally to *sets* of sentences $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{L}$ so that $\mathcal{S}_1 \vdash \mathcal{S}_2$ iff it is possible to prove every sentence $\mathtt{s} \in \mathcal{S}_2$ assuming the truth of every sentence in $\mathcal{S}_1$. Thus, $\vdash$ is both a relation among pairs of sentences in $\mathcal{L}$ and among pairs of *sets* of sentences in $\mathcal{L}$.

Definition 1 *Let $\lambda = (L, \mathcal{M}, \kappa, \ell)$, where L is a logic with associated language $\mathcal{L}$ and entailment relation $\vdash$, $\mathcal{M}$ is a set, $\mathcal{P}(\mathcal{M})$ is its power set (i.e., set of all subsets), $\kappa : \mathcal{L} \to \mathcal{P}(\mathcal{M})$ and $\ell : \mathcal{P}(\mathcal{M}) \to \mathcal{L}$ are both functions. We call λ a **Logic System** if κ, ℓ, and $\vdash$ additionally satisfy the conditions that, for all $M \subseteq \mathcal{M}$ and for all $\mathtt{s}_1, \mathtt{s}_2 \in \mathcal{L}$, one has*

$$\kappa(\ell(M)) = M, \tag{5}$$

$$\mathtt{s}_1 \vdash \mathtt{s}_2 \text{ if and only if } \kappa(\mathtt{s}_1) \subseteq \kappa(\mathtt{s}_2). \tag{6}$$

*We refer to κ as the **kernel function** associated with λ, and for a given sentence $\mathtt{s} \in \mathcal{L}$, we call $\kappa(\mathtt{s})$ the **kernel** of $\mathtt{s}$.*

This article deals exclusively with cases where $\mathcal{M}$ is finite, and thus $\mathcal{P}(\mathcal{M})$ is a finite set[1]. For the vast majority of the results of this article, we shall not need to assume anything further about the Logic Systems with which we work. The underlying logical vocabularies can be arbitrary and need not include any of the usual logical operators, so long as conditions (5) and (6) are satisfied. For Theorem 8 in Subsection 5.7, however, we will have to assume the presence of the standard logical operators $\vee, \wedge$ and $\neg$, and, moreover, that they have a certain natural set-theoretic behavior with respect to the kernel function κ.

Definition 2 *A Logic System $\lambda = (L, \mathcal{M}, \kappa, \ell)$ is said to be **proper** if the logical vocabulary of L includes the operators $\vee, \wedge$ and $\neg$, and, moreover, for every $\mathtt{s}, \mathtt{t} \in \mathcal{L}$, where $\mathcal{L}$ is the language associated with L, the following hold:*

1. $\kappa(\mathtt{s} \vee \mathtt{t}) = \kappa(\mathtt{s}) \cup \kappa(\mathtt{t})$,
2. $\kappa(\mathtt{s} \wedge \mathtt{t}) = \kappa(\mathtt{s}) \cap \kappa(\mathtt{t})$,
3. $\kappa(\neg \mathtt{s}) = \kappa(\mathtt{s})^c$.

[1] It is also possible to consider the case where $\mathcal{M}$ is not a set but rather a *proper class*, in which case $\mathcal{P}(\mathcal{M})$ is then called the *power class*. See the footnote to Theorem 9 for a brief discussion.

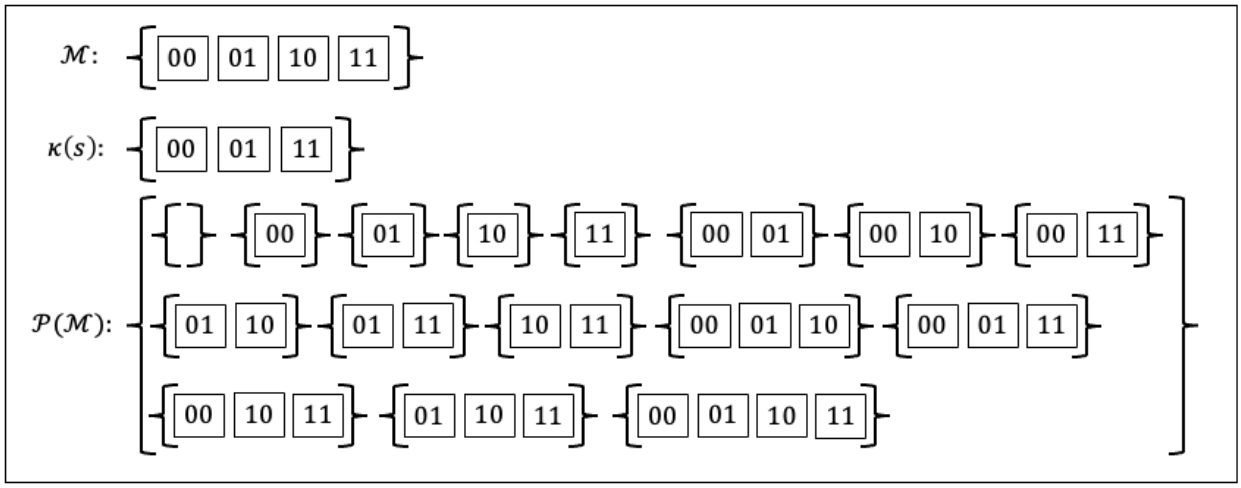


Figure 2: Pictorial representation of the objects $\mathcal{M}, \kappa(\mathtt{s})$ and $\mathcal{P}(\mathcal{M})$ for the case of Propositional Logic on 2 variables. The displayed kernel, $\kappa(\mathtt{s})$, is for the sentence $\mathtt{s} = \neg \mathtt{X}_1 \vee \mathtt{X}_2$. The notation $\boxed{01}$, for example, corresponds to the truth value assignment $\mathtt{X}_1 = \text{False}, \mathtt{X}_2 = \text{True}$.

Example 1 (Propositional Logic) *Let us consider the case where L is (classical) Propositional Logic on a fixed number,* $\mathtt{m}$*, of propositional variables. The vocabulary* τ *consists of the logical connectives* $\vee, \wedge$*, and* $\neg$*, the* $\mathtt{m}$ *propositional variables* $\mathtt{X}_1, ..., \mathtt{X}_\mathtt{m}$*, as well as parentheses* $(\cdot)$ *to aid in grouping. Any single standalone propositional variable* $\mathtt{X}_\mathtt{i}$ *is considered to be a well-formed sentence. We also consider the empty sentence, denoted alternatively by* $\top$*, to be a well-formed sentence. If* $\mathtt{s}, \mathtt{t}$ *are well-formed sentences, then so are* $\mathtt{s} \vee \mathtt{t}, \mathtt{s} \wedge \mathtt{t}, \neg \mathtt{s}$*, and* $(\mathtt{s})$*. We typically write* $\bot$ *in lieu of* $\neg\top$*. The symbols* $\vee$ *and* $\wedge$ *are understood to apply in left-to-right order. In other words, the sentence* $\mathtt{r} \vee \mathtt{s} \wedge \mathtt{t}$ *is syntactically equivalent to* $(\mathtt{r} \vee \mathtt{s}) \wedge \mathtt{t}$*.*

In this case, we let the set $\mathcal{M}$ *be the set of the* $2^\mathtt{m}$ *different truth-value assignments to the* $\mathtt{m}$ *propositional variables. Further, we let* $\kappa : \mathcal{L} \to \mathcal{P}(\mathcal{M})$ *be the function that maps each sentence* $\mathtt{s} \in \mathcal{L}$ *to the set of truth-value assignments that make* $\mathtt{s}$ *true. In Subsection 6.3 we will see that one can define a function* $\ell : \mathcal{P}(\mathcal{M}) \to \mathcal{L}$ *such that, for all* $M \subseteq \mathcal{M}$*, we have* $\kappa(\ell(M)) = M$*, and thus (5) holds. We will also see that for* κ *as we have defined it, as long as we equip L with a standard propositional proof system (look ahead to Definition 20), condition (6) holds. Furthermore, from the definition of* κ*, it is an elementary exercise to verify that conditions 1–3 of Definition 2 all hold, so that Propositional Logic on a fixed number of variables can thus be turned into a proper Logic System.*

Figure 2 depicts several of the objects described in the above example for the case of Propositional Logic on 2 variables and the sentence $\mathtt{s} = \neg \mathtt{X}_1 \vee \mathtt{X}_2$. The set $\mathcal{M}$ consists of all truth value assignments to the variables $\mathtt{X}_1$ and $\mathtt{X}_2$. The kernel of $\mathtt{s}, \kappa(\mathtt{s})$, is the set of all truth value assignments to $\mathtt{X}_1$ and $\mathtt{X}_2$ making $\mathtt{s}$ true, and $\mathcal{P}(\mathcal{M})$ is the set of all subsets of $\mathcal{M}$, in other words, the set of all possible kernels of sentences in the two variables $\mathtt{X}_1$ and $\mathtt{X}_2$.

The basic preliminaries on logic systems in this subsection are sufficient for the majority of what follows. As previously noted, in Section 6 we provide a more complete definition of logic systems. Subsection 6.1 describes more formally what we mean by a *logic*. Subsection 6.2 provides a brief introduction to the branch of logic known as Model Theory and describes what it means for a mathematical object to be a model of a given set of logic sentences. Using just a small amount of model-theoretic formalism, we will then be able to show that the condition (6) for being a Logic System is satisfied by virtually all logics we care about. Lastly, Subsections 6.3 and 6.4 provide examples of proper Logic Systems, first for Propositional Logic and then for First-Order Logic.

2.2 Elementary information-theoretic definitions and notation

For a scalar $0 \leq p \leq 1$, we denote Shannon's binary entropy by

$$H_{\text{bin}}(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

For a random variable X on any arbitrary *discrete* alphabet, governed by a distribution p_X, we define

$$H(X) = E_X \left[\log_2 \frac{1}{p_X(X)} \right],$$

where p_X denotes the probability mass function of the discrete random variable X. In either case, entropy is expressed in bits, as we are using the logarithm base 2. Given two discrete random variables X, Y, we define conditional entropy and mutual information as

$$\begin{aligned} H(X|Y) &= E_{X,Y} \left[\log_2 \frac{1}{p_{X|Y}(X|Y)} \right], \\ I(X;Y) &= E_{X,Y} \left[\log_2 \frac{p_{X|Y}(X|Y)}{p_Y(Y)} \right]. \end{aligned}$$

It is also the case that

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y), \\ &= H(X) - H(X|Y) = H(Y) - H(Y|X) \end{aligned}$$

when the corresponding individual entropies are finite.

For three random variables A, B, C we say that they from a Markov chain if given B, A and C are statistically independent, and we write

$$A \to B \to C.$$

If $A \to B \to C$ then the data processing inequality states that

$$I(A;B) \geq I(A;C).$$

In addition,

$$H(A|BC) = H(A|B). \tag{7}$$

2.3 Communication setup

In this subsection, we describe the fundamentals of our communication setup, also summarized in Figure 3.

2.3.1 Notation

We use the letter $\mathtt{s}$ to denote a logic sentence known to Alice, the sender. Similarly, we use $\mathtt{r}$ to denote a logic sentence known to Bob, the receiver. In many, but not all, of our setups $\mathtt{r}$ is also known to Alice. Once Bob decodes whatever information he receives from Alice, possibly in combination with $\mathtt{r}$, Bob deduces $\hat{\mathtt{s}}$, which he uses for the purpose of deducing $\mathtt{q}$, the problem that tests the success of this communication endeavor. To model various kinds of uncertainties in what Alice and Bob know about each other's knowledge, we will introduce random versions of the logic sentences by using upper case notation $\mathtt{S}, \mathtt{R}, \mathtt{Q}$.

For any given kernel $k \in \mathcal{M}$, we let $|k|$ denote the size of the set; obviously $0 \leq |k| \leq |\mathcal{M}|$. We will often refer in our article to the *normalized expected kernel size* of some random sentence, defined, in this example, as follows:

$$\frac{1}{|\mathcal{M}|} E\left[|\kappa(\mathtt{S})| \right]. \tag{8}$$

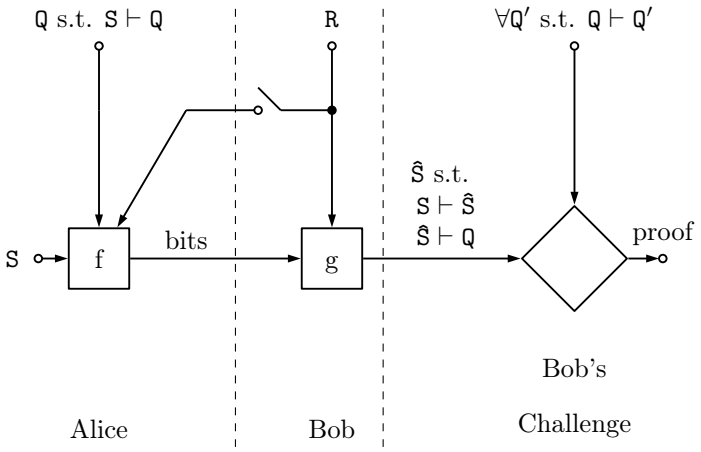


Figure 3: General communication diagram treated in this work.

2.3.2 Initial meeting

Alice and Bob meet ahead of time, and settle on a Logic System (see Definition 1). They agree on the general conditions of a future communication: that Alice will have access to $\mathtt{S}$, whether Bob will have access to $\mathtt{R}$; if so, whether Alice herself will have access to it as well. They also agree that the goal is for Bob to prove the truth of a logic sentence $\mathtt{Q} \in \mathcal{L}$ using information that Alice will provide employing pre-agreed upon encoding and decoding functions, and that the output of the decoding function, called $\hat{\mathtt{S}}$, must be entailed by $\mathtt{S}$. The sentences $\mathtt{S}, \mathtt{R}, \mathtt{Q}$ are not known at the time of this meeting, and will be revealed to the relevant parties later. We will describe these functions shortly. It is important to note that any logic sentences that can be deduced within the Logic System are assumed to be known to be true by both Alice and Bob as a result of this meeting. No communication cost whatsoever is levied against any exchange that happens during this meeting.

2.3.3 Correlated world observations

After the initial meeting, Alice and Bob go their own ways; Alice, the sender, obtains knowledge about the world summarized in a logic sentence $\mathtt{S} \in \mathcal{L}$ whereas Bob, the receiver, obtains $\mathtt{R} \in \mathcal{L}$. We consider both settings where Alice knows and doesn't know $\mathtt{R}$. We assume that there is consistency between Alice's and Bob's observations but that Alice has a potentially sharper view of the world:

$$\mathtt{S} \vdash \mathtt{R}. \tag{9}$$

There is one exception to this assumption when we treat a misinformation scenario, which will be clear during that discussion. We assume that the query $\mathtt{Q}$ that Bob will be able to prove after the communication takes place is provable using Alice's knowledge:

$$\mathtt{S} \vdash \mathtt{Q}. \tag{10}$$

This is universally true in all of our results, including those of misinformation. Finally, we make an assumption that is more technical in nature:

$$\mathtt{Q} \vdash \mathtt{R}. \tag{11}$$

In the case that Alice knows $\mathtt{R}$, the assumption above is justified in light of the following result.

Lemma 1 *Given a Logic System $(L, \mathcal{M}, \kappa, \ell)$, for $\mathtt{s}, \mathtt{q}, \mathtt{r} \in \mathcal{L}$, if $\mathtt{s} \vdash \mathtt{q}$ and $\mathtt{s} \vdash \mathtt{r}$, then there exists a $\mathtt{q}' \in \mathcal{L}$, given by $\mathtt{q}' = \ell(\kappa(\mathtt{q}) \cap \kappa(\mathtt{r}))$, such that $\mathtt{q}' \vdash \mathtt{q}$, $\mathtt{s} \vdash \mathtt{q}'$ and $\mathtt{q}' \vdash \mathtt{r}$.*

The Lemma follows from the definition of a Logic System. Thus, in the case both Alice and Bob share $\mathtt{R}$, without loss of essential generality, the query that Alice is attempting to ensure Bob can prove can be assumed to satisfy (11).

The case that Alice does not know $\mathtt{R}$ splits in two cases. In one case, $\mathtt{Q} = \mathtt{S}$ and (11) simply reduces to (9); this is a very interesting setting in practice. If in general $\mathtt{Q}$ is a weaker sentence than $\mathtt{S}$, then the assumption (11) is too strong since it does not reduce to (9) and thus we believe it to be of reduced practical interest. For mathematical completeness, we do provide a result (Theorem 8) under such an assumption but do not rely on it to make the main points of our paper.

2.3.4 Communication

To communicate, Alice and Bob rely on the functions agreed upon during the initial meeting. The nature of these functions depend on the nature of the specific situation Alice and Bob have planned for. In the simplest of settings, neither Bob nor Alice have access to $\mathtt{R}$ and Alice will be communicating to Bob a message that ensures he prove all that she can prove.

The encoding function is generally denoted f, and in this simple case it maps $\mathtt{S}$ to a finite sequence of bits:

$$f : \mathcal{L} \to \{0,1\}^*,$$

where the notation $\{0,1\}^*$ is meant to signify the set of finite binary strings. In turn a receiver will decode the information being send by the sender using a decoding function denoted by g:

$$g : \{0,1\}^* \to \mathcal{L}.$$

The output of g is generally denoted by $\hat{\mathtt{S}}$. More complex situations augment the arguments that f can take on to include $\mathtt{Q}, \mathtt{R}$ as relevant; similarly g may also depend on $\mathtt{R}$. In even more complex situations, the communication involves a conversation where sender and receiver take turns.

The function f in any of the settings under consideration (Figure 5) is capable of producing a variable number of bits, as this is a more flexible setting than assuming a fixed number of bits. However, an additional complication is that it may not be easy to determine when these bits start and finish in an otherwise arbitrary bit sequence. To resolve this matter, we will rely on the standard concept from information theory of prefix-free codes. Let $\mathcal{C} \subseteq \{0,1\}^*$ be a set of codewords. We say that $\mathcal{C}$ is prefix-free if, for all distinct $c_1, c_2 \in \mathcal{C}$, c_1 is not a prefix of c_2. We will assume that the image of the encoder is prefix free; this will be mathematically explicit when the theorems are stated and proved.

Finally, we will use the expected number of transmitted bits as a performance metric for any proposed system. In the simple example described above, such a metric is

$$E_{\mathtt{S}}[\mathbf{len}(f(\mathtt{S}))],$$

where $\mathbf{len}()$ is the function that maps a finite string to its length.

2.3.5 Challenge and deduction

After the communication takes place, Bob is challenged with any sentence that can be proven by $\mathtt{Q}$ (including possibly $\mathtt{Q}$ itself), and Bob is able to produce a proof for that query starting from the logic sentence $\hat{\mathtt{S}}$, which in turn we assume is entailed by $\mathtt{S}$. Mathematically, for the system to have succeeded, it must be the case these conditions hold: $\mathtt{S} \vdash \hat{\mathtt{S}}$, $\hat{\mathtt{S}} \vdash \mathtt{Q}$.

2.3.6 Probabilistic model

We will present theoretical results in the form of upper and lower bounds on the total number of expected bits. Our upper bounds are applicable to any possible distribution over $\mathtt{S}, \mathtt{Q}, \mathtt{R}$ as long as the entailment conditions described in Subsection 2.3.3 are met, and are phrased in terms of normalized expected kernel sizes (see Equation (8)). We next make a definition that we will rely on when we formally state our theorems:

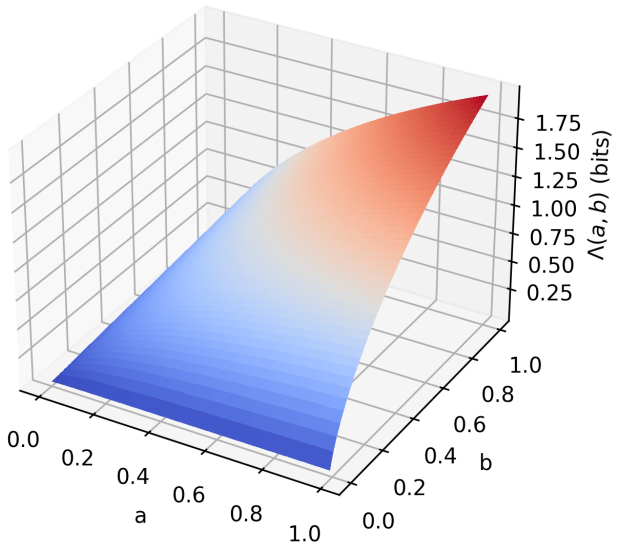


Figure 4: Illustration of $\Lambda(a, b)$

Definition 3 (probability laws for kernels) *We say that the random logic sentence* $\mathtt{A} \in \mathcal{L}$ *has a kernel that follows a* p_a*-law if* $|\mathcal{M}|^{-1} E\left[|\kappa(\mathtt{A})|\right] = p_a$*. We say that the random logic sentences* $\mathtt{A}, \mathtt{B} \in \mathcal{L}$ *have kernels that follow a* (p_a, p_b)*-law, with* $p_b \geq p_a$*, if* $\mathtt{A}$ *has a kernel that follows a* p_a*-law,* $\mathtt{B}$ *has a kernel that follows a* p_b*-law, and* $\mathtt{A} \vdash \mathtt{B}$*. We say that the random logic sentences* $\mathtt{A}, \mathtt{B}, \mathtt{C} \in \mathcal{L}$ *have their kernels follow a* (p_a, p_b, p_c)*-law, with* $p_c \geq p_b \geq p_a$*, if* $\mathtt{A}$ *follows a* p_a*-law,* $\mathtt{B}$ *follows a* p_b*-law,* $\mathtt{C}$ *follows a* p_c*-law,* $\mathtt{A} \vdash \mathtt{B}$ *and* $\mathtt{B} \vdash \mathtt{C}$*.*

For our lower bounds, we use stronger assumptions that involve independent and identically distributed (i.i.d.) random variables. Throughout the paper, the reader will notice that whenever we use the i.i.d. assumption, which is otherwise never assumed in the upper bounds, the upper and lower bounds will match asymptotically. This pattern follows similar patterns in information theory, where "the i.i.d. source is the hardest to compress" (amongst all sources with the same marginal statistics).

3 Summary of information-theoretic contributions

In this section, we provide a summary of the core information-theoretic results that we have obtained, intended as a guide to understand the actual formal result statements and proofs found in Section 5, and also the practical results in Section 4.5.

All of our information-theoretic results are expressed using a function with two arguments $\Lambda(a, b)$, defined this way: for any $a, b \geq 0$,

$$\Lambda(a, b) = a \log_2 \left(\frac{a+b}{a} \right) + b \log_2 \left(\frac{a+b}{b} \right) = (a+b) H_{\text{bin}} \left(\frac{a}{a+b} \right) = (a+b) H_{\text{bin}} \left(\frac{b}{a+b} \right),$$

where we additionally define $\Lambda(0, b) = \Lambda(a, 0) = 0$. The greek letter Λ is chosen for this function in reference to its apparent emergence in problems involving logic. This function is illustrated in Figure 4.

The reader is not expected to appreciate, at the present moment, the intuition behind why the $\Lambda(a, b)$ function is relevant to our problem. The way we first encountered this function was as the solution to a

variational problem that is at the core of the proof of Theorem 4, which in turn is the basis for how Theorem 5 is proven. Subsequently, we noticed that all of our information-theoretic results could be rewritten in terms of this function. This expression satisfies the following basic properties, which are proved in the Appendix.

Lemma 2 (Elementary properties of $\Lambda(a,b)$) *The function $\Lambda(a,b)$ is concave over the domain $[0,+\infty)\times[0,+\infty)$. If $\Delta_a,\Delta_b\geq 0$ with at least one of them being strictly positive, then $\Lambda(a+\Delta_a,b+\Delta_b)>\Lambda(a,b)$. If $a+b<1$, then for any mixture parameter $\lambda\in[0,1]$, $\Lambda(a,b)<H_{bin}(\lambda a+(1-\lambda)b)$. For any ξ, $\xi\Lambda(a,b)=\Lambda(\xi a,\xi b)$.*

We believe that this article is the first to point out the relevance of this particular form of entropy to communication problems involving logic. To this end, given a Logic System $(L,\mathcal{L},\mathcal{M},\kappa,\ell,\vdash)$ and a chosen ordering of the elements of $\mathcal{M}=\{\mu_1,\ldots,\mu_{|\mathcal{M}|}\}$, we define $\vec{\kappa}:\mathcal{L}\to\{0,1\}^{|\mathcal{M}|}$ via

$$\vec{\kappa}(\mathtt{s})_i \triangleq \begin{cases} 1 & \text{if } \mu_i\in\kappa(\mathtt{s}),\\ 0 & \text{otherwise;}\end{cases} \tag{12}$$

$$|\vec{\kappa}(\mathtt{s})| \triangleq |\kappa(\mathtt{s})|. \tag{13}$$

Note that the function $\vec{\kappa}$ is just another way of thinking about the function κ, e.g., as an indicator function. Similarly, define the function

$$\vec{\ell}:\{0,1\}^{|\mathcal{M}|}\to\mathcal{L} \tag{14}$$

that accepts a set indicator vector, recovers the corresponding subset of $\mathcal{M}$, and then passes that subset to ℓ.

Our main result, to be interpreted in the context of a given Logic System $(L,\mathcal{L},\mathcal{M},\kappa,\ell,\vdash)$ and the communication setup in Section 2 (as illustrated in Figure 3), is stated next.

Theorem 1 *Given a Logic System $(L,\mathcal{M},\kappa,\ell)$, for any distribution over $(\mathtt{S},\mathtt{Q},\mathtt{R})$ meeting the entailment conditions $\mathtt{S}\vdash\mathtt{Q}$ and $\mathtt{Q}\vdash\mathtt{R}$, if the corresponding kernels have normalized sizes p_s,p_q,p_r, respectively, then an algorithm exists with a normalized average cost in total bits exchanged that is upper bounded by $\Lambda(p_s,p_r-p_q)+O(|\mathcal{M}|^{-1}\log_2|\mathcal{M}|)$. If additionally the random variables $\{(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q}),\vec{\kappa}(\mathtt{R}))_j\}_{j=1}^{|\mathcal{M}|}$ are i.i.d. and $\mathtt{R}\to\kappa(\mathtt{R})\to(\kappa(\mathtt{S}),\kappa(\mathtt{Q}))$, then the normalized average cost of any such algorithm is lower bounded by $\Lambda(p_s,p_r-p_q)$. The theorem statement holds true regardless of whether Alice knows $\mathtt{R}$ or not.*

We remark that this theorem is a consequence of Theorems 7 and 8, and in fact, we will not provide a direct proof for it. Nonetheless, the mathematical machinery developed to address the scenarios addressed by those theorems is unnecessarily complex and thus for didactic purposes, we chose to gradually build the sophistication of our results over a series of theorems so as to allow the key ideas to settle more easily and firmly.

For the following discussion, the reader is referred to Figures 5, 6 and 7. The first figure is in essence a case-by-case expansion of Figure 3, which explicitly links each case of interest to its assumptions and the corresponding Theorem that treats it. While in Figure 5 we emphasize the end-to-end nature of our setup, where Alice and Bob experience sentences from the language $\mathcal{L}$ and where Bob ends up with an updated sentence after the communication takes place, much of our work is predicated on the analysis of the underlying kernels. Figure 6 illustrates the assumptions of each notable result as set relations between the various kernels involved in the communication setup, including kernels that each of Alice and Bob know separately at the time of communication as well as the kernel that Bob has in his hands after the communication takes place. Figure 7 elaborates upon Figure 4 by replacing the 3-dimensional plot with a contour plot and illustrating how changes in the p_s,p_q,p_r values result in different theoretical bounds based on $\Lambda(p_s,p_r-p_q)$. The point of this last figure is to "put it all together" – in spite of the different set of assumptions that we will walk the reader through next, at the end all the results can be expressed in terms of $\Lambda(p_s,p_r-p_q)$.

3.1 Full ignorance – Theorem 2

In this setup (Figures 5-a, 6-a) the goal is for Bob to be able to prove any mathematical sentence that Alice can prove, whilst Bob has access to no logic sentence, and hence the reference to "full ignorance", understood here as Bob's state relative to Alice's knowledge. A single parameter determines the results that we have in this scenario, namely the normalized expected kernel size of Alice's sentence, denoted by $p_s = |\mathcal{M}|^{-1} E|\kappa(\mathtt{S})|$. In the context of our more general result (Theorem 1), this scenario corresponds to the setting where $p_r = 1, p_q = p_s$. In Figure 7 we illustrate a contour plot of $\Lambda(p_s, p_r - p_q)$; full ignorance is then represented by the top-most negative 1 slope line. We note that $\Lambda(p_s, 1-p_s) = H_{\text{bin}}(p_s)$, and thus this result agrees with the intuition that the optimal cost in this case is the entropy of Alice's kernel. A smaller kernel, in our setup, is associated with a more informative logic sentence, since it has ruled out more of $\mathcal{M}$ as impossible. However, for $p_s < 0.5$, smaller kernels are in fact *cheaper* to send, contradicting the intuition that they somehow correspond to "more knowledge". The opposite happens nonetheless for $p_s > 0.5$, where smaller kernels are indeed more expensive to send. The lesson here is that one should not necessarily equate the notion of the amount of knowledge facts with information bits.

We stress that this result, in and by itself, is not particularly surprising given how we have defined the Logic System and the kernel of a sentence, but it is a useful baseline to understand our general result as well as our proof techniques.

We also reinforce that $\Lambda(p_s, 1-p_s)$ *is loose* when the special additional i.i.d. conditions in the theorem are not met. In fact Theorem 2 includes a generally tight bound which states that, not surprisingly,

$$H(\kappa(\mathtt{S})) \tag{15}$$

is the ultimate compression bound; this is to be connected to our early discussion leading to (4). This type of strong, ultimate bound calculation is presently not provided for the rest of the Theorems in an effort to emphasize so-called "single letter results", such as the bound given by $\Lambda(p_s, 1-p_s)$, which are tight under special conditions and which are often held in special esteem in the information theory field as they are much simpler to state and reason about, and thus yield more early insight.

3.2 Partial ignorance – Theorem 3

A straightforward way to improve upon the full-ignorance setting is to assume that at the time of communication a sentence `r` is revealed to both Alice and Bob, in addition to `s` being revealed to Alice only. This is represented by the two new arrows in Figure 5-b as well as the the new kernel with a rectangular shape in Figure 6-b. Note that this rectangular kernel shows up in both the sender and receiver diagrams, as it is available to both during the communication act. As disclosed earlier, we assume that $\mathtt{s} \vdash \mathtt{r}$ and thus `r` does not allow Alice to prove any more sentences than she could with `s` alone, however it gives her significant context to what Bob is aware of, thus reducing the total cost of communication. In this setting, two parameters determine the scenario: $p_s \leq p_r$, and in the more general context of Theorem 1, the additional condition is that $p_q = p_s$, since it is still the goal for Bob to be able to prove anything that Alice can. In Figure 7 this scenario is illustrated with negative 1 sloped lines that are strictly below the top-most such line.

In the spirit of highlighting Bob's state with respect to that of Alice's, we say that Bob is partially ignorant. The corresponding bound in this scenario is $\Lambda(p_s, p_r - p_s)$. We remind the reader that Λ is monotonically increasing on either of its two arguments, and therefore as p_r decreases while keeping p_s fixed, the bound strictly decreases. As we discussed earlier, a smaller kernel is associated with a more informative logic sentence and thus unlike in the full ignorance case, in this case the result does agree with intuition: the more informative is the logic sentence that is shared by Alice and Bob, the lower the communication cost.

3.3 Less is More – Theorem 5

For this scenario, we return to the full-ignorance setting, but add a twist: the goal is not for Bob to prove all that Alice can, but rather, to prove a more targeted query `q` that can be derived from Alice's `s`, but in general is not logically equivalent to `s`. In the context of Theorem 1, this scenario corresponds to the scenario $p_q > p_s$ and $p_r = 1$, with the bound being $\Lambda(p_s, 1-p_q)$. We introduce it in Figure 5-c with a query `q` that is given to Alice at the time of communication, but not to Bob. Correspondingly, in Figure 6-c we introduce an oval shaped kernel which includes that of Alice's logic sentence, but which is unavailable to Bob.

The reader's first instinct may be, why don't we either send the kernel of $\mathtt{q}$ or the kernel of $\mathtt{s}$, whatever is least expensive? The normalized average cost for this strategy is

$$\min\{H_{\text{bin}}(p_s), H_{\text{bin}}(p_q)\} > \Lambda(p_s, 1-p_q), \tag{16}$$

where this last inequality is a consequence of Lemma 2. Thus this strategy is strictly speaking suboptimal. We give the insight as to why. In reference to the third column of Figure 6-c, notice that any $\hat{\mathtt{s}}$ with the property that

$$\kappa(\mathtt{s}) \subseteq \kappa(\hat{\mathtt{s}}) \subseteq \kappa(\mathtt{q}) \tag{17}$$

will allow Bob to prove $\mathtt{q}$ with an $\hat{\mathtt{s}}$ with the property that $\mathtt{s} \vdash \hat{\mathtt{s}}$; this is a consequence of Definition 1 of a Logic System. Thus Alice has many more options to meet this goal than sending the kernel of $\mathtt{q}$ or that of $\mathtt{s}$, and it is possible to create an efficient listing of those options to cover all the possibilities for $\mathtt{q}$ and $\mathtt{s}$. We illustrate this in Figure 8, which illustrates what we call the "less is more" paradox. Notice that for Bob to be able to prove $\mathtt{q}$, the bit cost was smaller than either sending the kernels of $\mathtt{q}$ or $\mathtt{s}$ (Less...). But notice that in general $\kappa(\hat{\mathtt{s}})$ may be a strict subset of $\kappa(\mathtt{q})$. Because it is still the case that $\kappa(\mathtt{s}) \subseteq \kappa(\hat{\mathtt{s}})$, it follows that Bob is able to prove even more facts than he needed to prove (...is more).

Of note, this also has potential implications for security – being as efficient as one can to allow Bob to prove $\mathtt{q}$ using facts consistent with Alice's $\mathtt{s}$ results in revealing more than $\mathtt{q}$. One may say that one needs to say more to say less.

3.4 No need to know – Theorem 6

We now return to the Partial-ignorance setting, but eliminate Alice's ability to directly observe $\mathtt{r}$ (see the difference between Figures 5-b and 5-d). Since Alice doesn't know $\mathtt{r}$, she cannot use the strategy that we described under Partial ignorance which leverages the kernel of $\mathtt{r}$, illustrated as a rectangle in the first and third columns of Figure 6-b, in order to reduce the total bit cost (observe the absence of the rectangular kernel in the first column of Figure 6-d). We stress that for simplicity reasons, Figure 5-d only shows a single turn of communication where Alice is the sender and Bob is the receiver. In our work, the communication pattern is more complex – multiple turns are allowed. To keep evaluation as consistent as possible, the total sum of the average bits exchanged in any direction is the figure of merit in this setup.

The surprising result here is that exactly the same achievable limit as in the case of Partial ignorance applies – i.e., $\Lambda(p_s, p_r - p_s)$ – hence the "Alice does *not have a need to know*" reminder in the title of this subsection. The main proof mechanism borrows from the theories of Slepian/Wolf [68] and Wyner/Ziv [81] the idea of hashing, which in this case is applied to the kernel of $\mathtt{s}$; however, specialized arguments are introduced in this article that allow us to prove an upper bound under very general assumptions on how $\mathtt{S}, \mathtt{R}$ are distributed, and a lower bound is also introduced that accounts for the potential of multiple turns as well. The details can be found in Section 5.

3.5 Misinformation – Theorem 7

To discuss this subsection, we depart from the Partial-ignorance setup, and replace the assumption that $\mathtt{s} \vdash \mathtt{r}$ with an assumption that, instead, the sentences $\mathtt{s}$ and $\mathtt{r}$ are logically inconsistent – $\kappa(\mathtt{s}) \cap \kappa(\mathtt{r}) = \emptyset$ – see Figure 6-e. Bob still wants to be able to prove all that Alice can, and thus we regard Bob as being in a state of misinformation, albeit in a highly cooperative situation.

The fundamental limit in this situation is in fact quite easy to derive from the arguments for Partial ignorance, or as a direct consequence of the much more general setup of the subsequent Subsection 3.6, and for this reason no separate proof of it is provided in this article. The corresponding bound, under the assumption that $p_s \leq 1 - p_r$, is given by

$$\Lambda(p_s, 1 - p_r - p_s),$$

which is quite intuitive since it simply replaces p_r with $1 - p_r$ in the bound for Partial ignorance. Some thought provoking ideas can be derived from the result above. Consider the ratio of the cost of misinformation

vs. ignorance

$$\frac{\Lambda(p_s, p_r - p_s)}{\Lambda(p_s, 1 - p_r - p_s)},$$

which makes sense only under the restrictions $p_s \leq 1 - p_r, p_s \leq p_r, p_s \leq 1/2$. In Figure 10 we plot this ratio for the case $p_s = 0.1$. Note that the curve is monotonically decreasing with increasing p_r. Also note that as $p_r \longrightarrow p_s = 0.1$ from the right, the curve diverges to infinity. We may then colloquially state that, in a cooperative misinformation setting, the relative cost of correcting misinformation vs. correcting ignorance grows unbounded as the receiver becomes more opinionated. This agrees with one's intuition of what we would expect should happen.

Figure 9 illustrates contour plots of $\Lambda(p_s, p_r - p_s)$ and $\Lambda(p_s, 1 - p_r - p_s)$, under the restriction $p_s \leq p_r$, $p_s \leq 1 - p_r$, $p_s \leq 1/2$. These functions are obviously symmetric under the transformation $p_r \leftrightarrow 1 - p_r$. Notice that, as the cost of misinformation is kept constant and one approaches the regime where Bob is highly opinionated (p_r close to p_s), one "cuts" through contours for the ignorance case that are ever decreasing in bit cost value. This helps explain the unbounded growth shown in Figure 10.

Notice also that exactly the same behavior occurs with roles reversed in the upper part of Figure 9. Even though near the upper part (the line $p_r = 1 - p_s$) one might feel tempted to regard $\mathtt{r}$ as the most uninformative for a given p_s, this in fact is not true: in this case, the complement of the kernel of $\mathtt{r}$ is close to the kernel of $\mathtt{s}$ (said differently, Bob can simply negate $\mathtt{r}$ and thus obtain a sentence that is close, logically, to $\mathtt{s}$). Thus in reality the most uninformative sentences $\mathtt{r}$ are those associated with $p_r = 1/2$.

3.6 General setup when Alice knows what Bob knows – Theorem 7

In this subsection, we present the most general result we have been able to obtain in the case where both Alice and Bob share knowledge of $\mathtt{r}$. For this setting, we drop the assumption $\mathtt{s} \vdash \mathtt{r}$; correspondingly this is shown as stricken in illustrated in Figure 5-e. We do allow for any query $\mathtt{q}$ with the property that it is provable using $\mathtt{s}$ ($\mathtt{s} \vdash \mathtt{r}$).

The additional ways in which various kernels may relate to each other when $\mathtt{s} \vdash \mathtt{r}$ is dropped (but we keep $\mathtt{s} \vdash \mathtt{q}$) are illustrated in Figure 6-e, where the rectangular kernel may only be partially overlapping the kernel of $\mathtt{s}$ and the kernel of $\mathtt{q}$. Correspondingly, the result is more complex, with a sum of two terms involving the Λ function. The reader is not expected to immediately understand how to interpret these bounds, since additional notation has been introduced that is only discussed in Section 5. Having said this, in spite of all its apparent complexity, for the upper bound all that is really happening here is that two communication paths are being established: one to address ignorance, and one to address misinformation, and by themselves, these actually do not introduce fundamentally new ideas beyond those already introduced in our other results. This result completely subsumes the result on misinformation described in Subsection 3.5.

3.7 General setup when Alice does not know what Bob knows – Theorem 8

To complete our set of results, in this subsection we present a result where the conditions $\mathtt{s} \vdash \mathtt{q}$, $\mathtt{q} \vdash \mathtt{r}$ (and hence $\mathtt{s} \vdash \mathtt{r}$) are assumed. Critically, we do not assume that Alice knows $\mathtt{r}$. Thus this setting can be seen as a general result that subsumes those of Subsections 3.3 and 3.4. The mathematical aspects of this result are the most complex in the paper, incorporating every technique we developed elsewhere. Yet, one should be cautious in interpreting the practical significance of this result beyond what we have already argued in Subsections 3.3 and 3.4. We have already made this observation in the discussion subsequent to the sentence of Lemma 1; in a more practical version of this setting, the assumption $\mathtt{q} \vdash \mathtt{r}$ is dropped altogether. We conjecture that in that case, the estimate $\Lambda(p_s, p_r - p_q)$ is too low – the actual bit cost is in effect higher.

4 Practical algorithms and experimental results

In this section, we develop two practical algorithms that are components of our proposed logical semantic communication system, aimed at the targeted query scenario of Theorem 5 and the scenario where Alice does not know what Bob knows addressed in Theorem 6. These algorithms are based on linear coding concepts

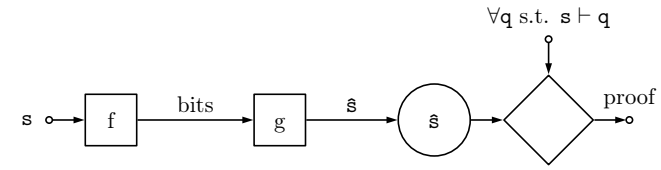
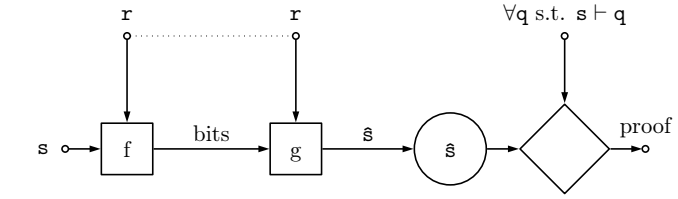
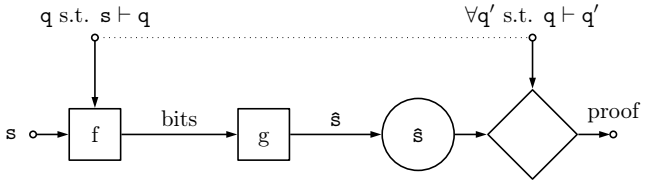
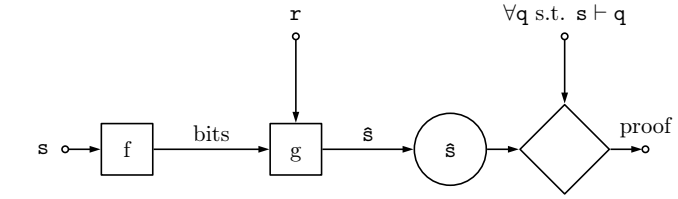
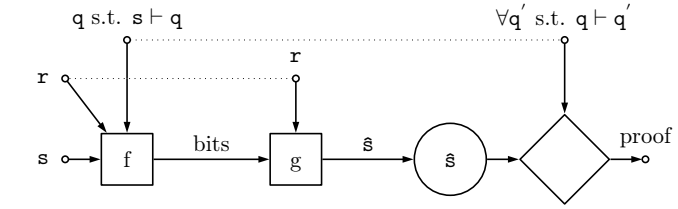
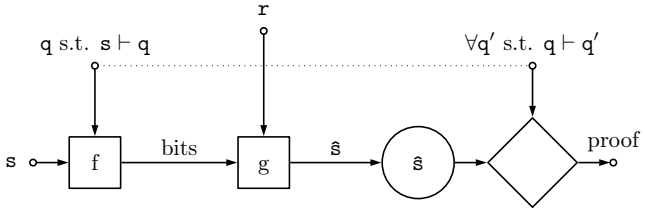
	Communication diagram	Assumptions	Achievable Shannon limit
(a)		○ r is available to Bob ○ r is available to Alice ○ q is available to Alice ● s entails r	$\Lambda(p_s, 1-p_s)$ Theorem 2 (full ignorance)
(b)		● r is available to Bob ● r is available to Alice ○ q is available to Alice ● s entails r	$\Lambda(p_s, p_r-p_s)$ Theorem 3 (partial ignorance)
(c)		○ r is available to Bob ○ r is available to Alice ● q is available to Alice ● s entails r	$\Lambda(p_s, 1-p_q)$ Theorem 5 (less is more)
(d)		● r is available to Bob ○ r is available to Alice ○ q is available to Alice ● s entails r	$\Lambda(p_s, p_r-p_s)$ (accounting for all bi-directional communication) Theorem 6 (no need to know)
(e)		● r is available to Bob ● r is available to Alice ● q is available to Alice ○ s entails r	$\Lambda(p_{s^*}, p_r-p_{q^*})+$ $\Lambda(p_{s^{**}}, 1-p_r-p_{q^{**}})$ Theorem 7 (Bob's sentence may not be entailed by Alice's)
(f)		● r is available to Bob ○ r is available to Alice ● q is available to Alice ● s entails r	$\Lambda(p_s, p_r-p_q)$ (accounting for all bi-directional communication) Theorem 8 (General setup when Alice does not know what Bob knows)

Figure 5: Communication diagrams for the scenarios covered in this article. The circle stands for the knowledge that the receiver has after the transmission has taken place. The diamond represents a computational device that given $\hat{\mathtt{s}}$ and a query $\mathtt{q}$, is capable of producing a proof of $\mathtt{q}$ as long as $\hat{\mathtt{s}} \vdash \mathtt{q}$. Throughout all the diagrams, we assume that if $\mathtt{q}$ is such that $\mathtt{s} \vdash \mathtt{q}$, then $\mathtt{s} \vdash \hat{\mathtt{s}}$ and $\hat{\mathtt{s}} \vdash \mathtt{q}$. For (b,d,f) we additionally assume that $\mathtt{s} \vdash \mathtt{r}$ and $\mathtt{q} \vdash \mathtt{r}$; these assumptions are crucially omitted in (e).

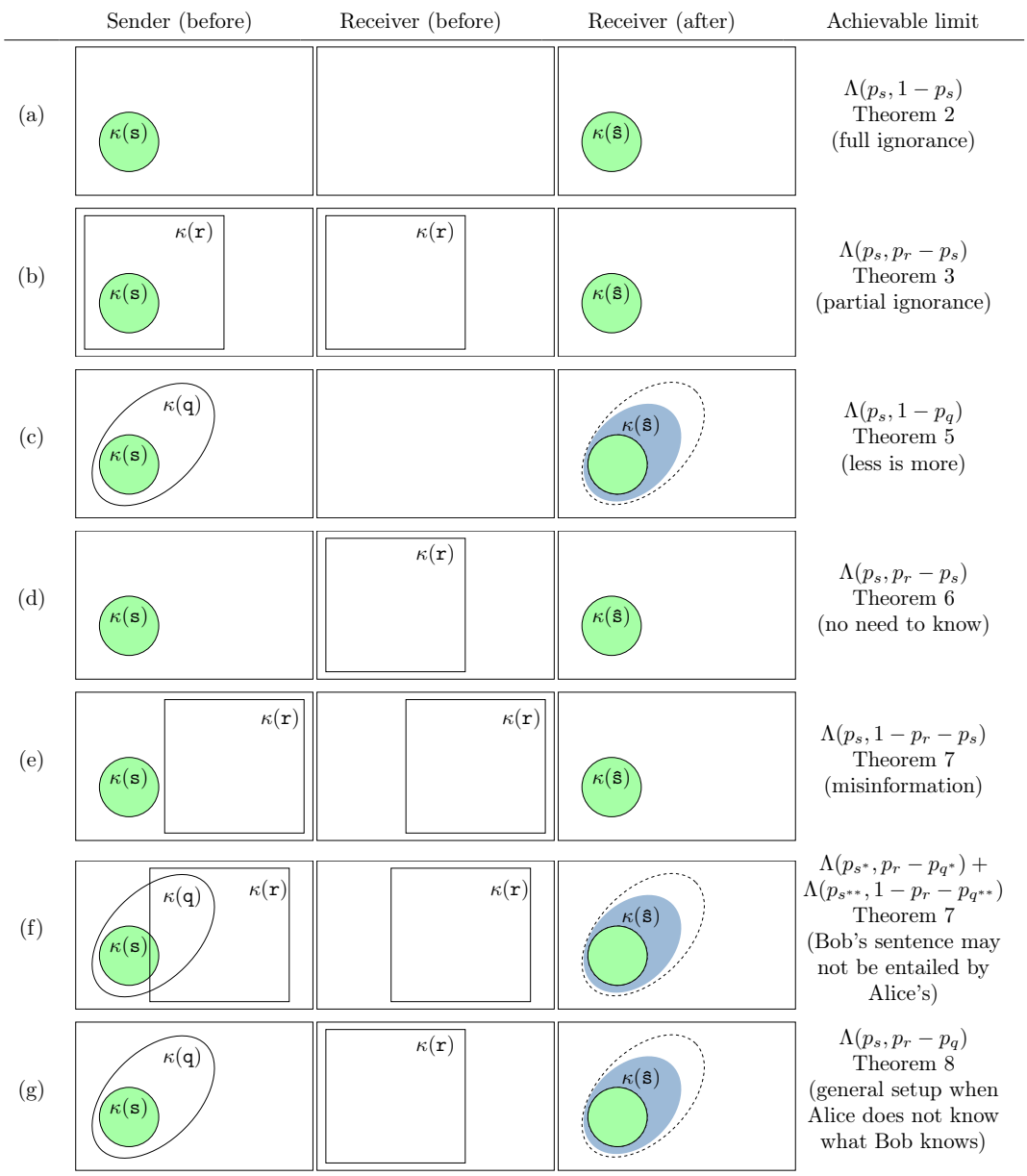


Figure 6: Kernel relationships for the communication scenarios considered in this paper.

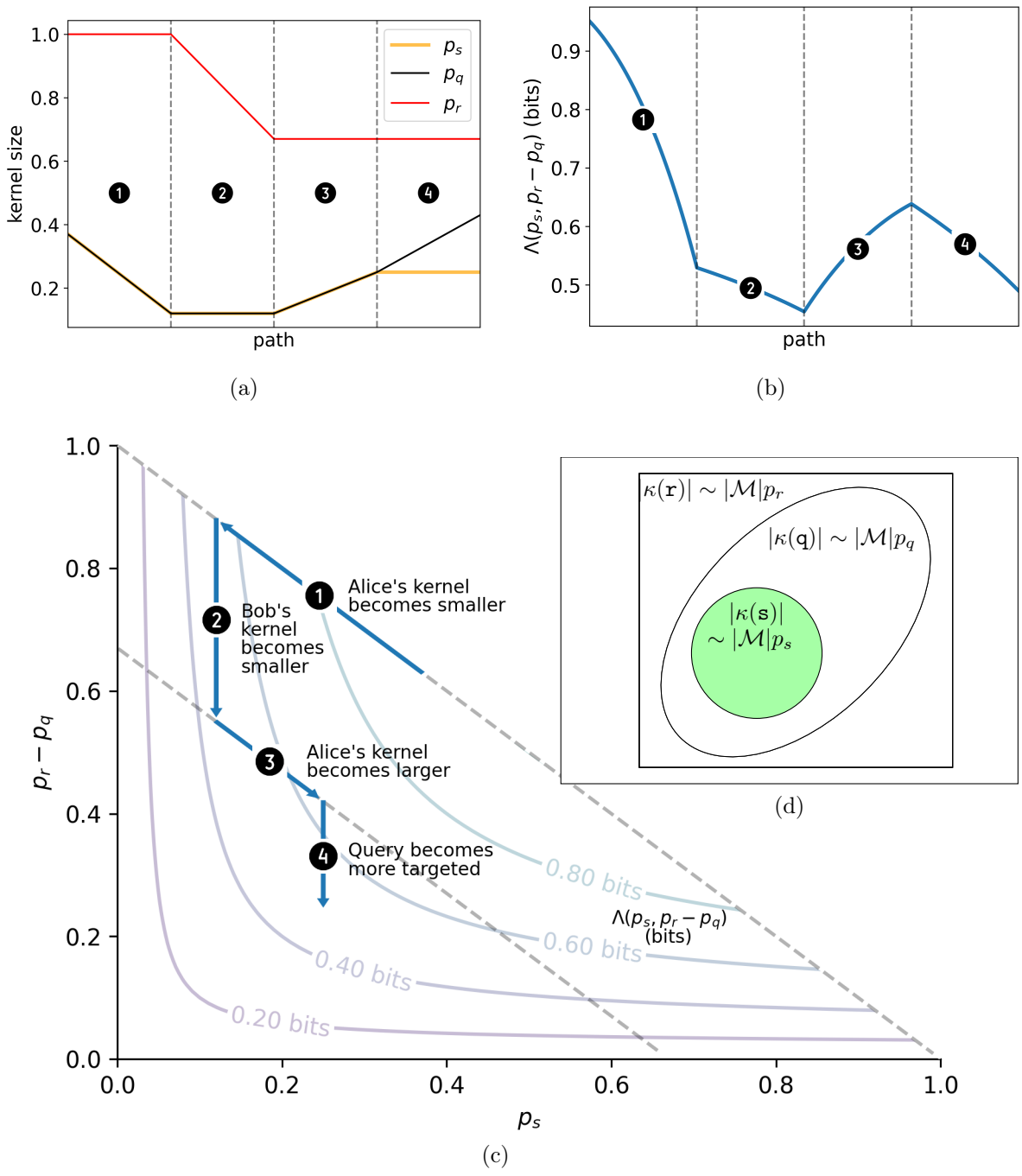


Figure 7: An illustration of several scenarios where Alice's knowledge entails that of Bob's. The contours are those of $\Lambda(p_s, p_r - p_q)$. As the parameters p_s, p_q, p_r are changed (a), the bound changes in value (b), also seen as as paths on a contour plot (c). The general relationship between the kernels is illustrated in (d). In all cases, Alice need not know $\mathbf{r}$ for the result to hold.

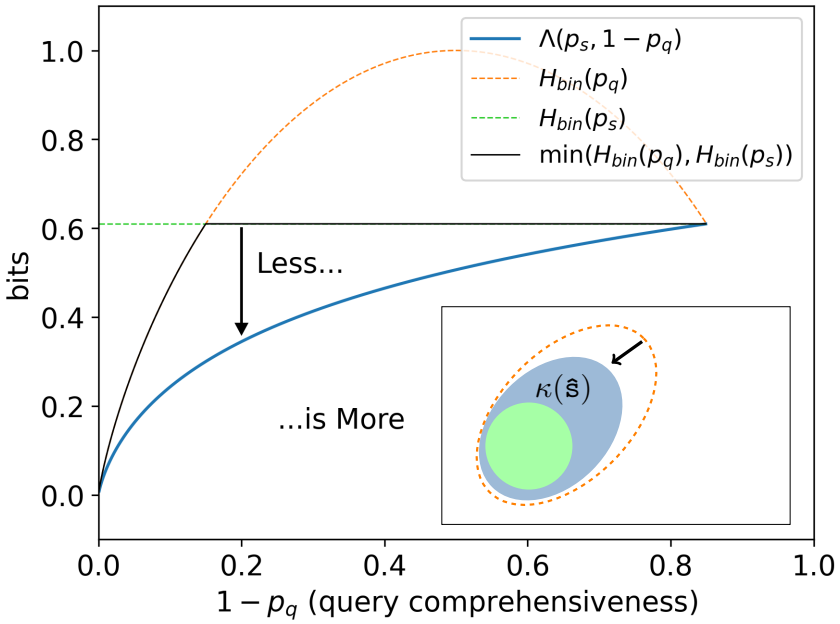


Figure 8: In blue, the ultimate communication limit Λ for the case $p_r = 1$, as the query ranges from trivial ($p_q = 1$) to coinciding with the sender's information ($p_q = p_s = 0.15$). Λ is cheaper (Less...) than the two obvious strategies, yet the kernel size received by Bob is smaller than that of the query, showing Bob can prove even more things (is More...) than required. A similar picture will hold for any p_r.

over the Galois Field with two elements $\{0, 1\}$, denoted $GF(2)$, and are shown to be optimal on specific settings. We then present experimental results comparing the performance of an ensemble of methods, including the ones developed here, and contrast them with the Λ bound as well as optimized methods that treat the logic expressions as strings in the classic information-theoretic sense.

4.1 Linear codes for targeted queries

Let $X_1^n \in \{0, 1, 2\}^n$ denote a random vector that represents three sets: $\{i : X_i = 0\}$, $\{i : X_i = 1\}$ and $\{i : X_i = 2\}$. The problem is to transmit to a receiver a vector $\hat{X}_1^n \in \{0, 1\}^n$ which agrees with X_1^n on all the positions $\{i : X_i \in \{0, 1\}\}$ as efficiently as possible, where efficiency is quantified as expected number of bits transmitted. One way to see this is that we are trying to efficiently send a partition that splits the sets $\{i : X_i = 0\}$, $\{i : X_i = 1\}$. The connection to the problem of semantic logic communication for the targeted queries scenario (Theorem 5) emerges from the observation that sending a partition that splits the sets $\kappa(\mathtt{S})$ and $\kappa(\mathtt{Q})^c$ is the key step in that problem.

We now present a general algorithm for partition compression based on linear codes over $GF(2)$. This algorithm can be applied even when we do not have any statistical model of the underlying sets for which we are creating a partition. In the special case where the entries of X_1^n are drawn i.i.d. from $\{0, 1, 2\}$ using probability masses $(p_0, p_1, 1 - p_0 - p_1)$, we will show that this algorithm is asymptotically optimal if $p_0 = p_1$. The attractiveness of linear codes stems from the fact that they are easier to implement in practice.

For ease of analysis, we assume that the encoder and decoder share a random matrix with i.i.d. entries drawn from $GF(2)$ uniformly at random, and with n columns and an infinite number of rows, which we shall refer to as G; nonetheless, the number of rows we will effectively be using is only slightly larger than $n(p_0 + p_1)$. Define

$$\Psi = \{i : X_i \in \{0, 1\}\}. \tag{18}$$

Let G_r denote the matrix obtained by extracting the first r rows from G, and for any given vector

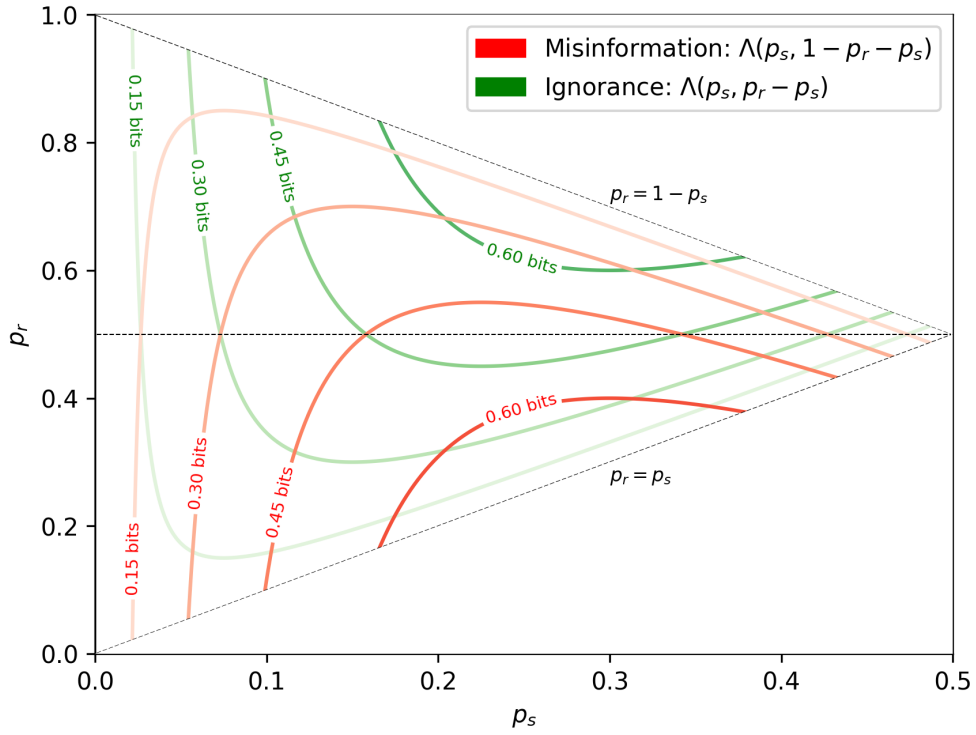


Figure 9: Contour plots for $\Lambda(p_s, p_r - p_s)$ (ignorance) and $\Lambda(p_s, 1-p_r-p_s)$ (misinformation). The plot accentuates the symmetry of these around $p_r = 1/2$; what is cheap for ignorance is expensive for misinformation and vice versa.

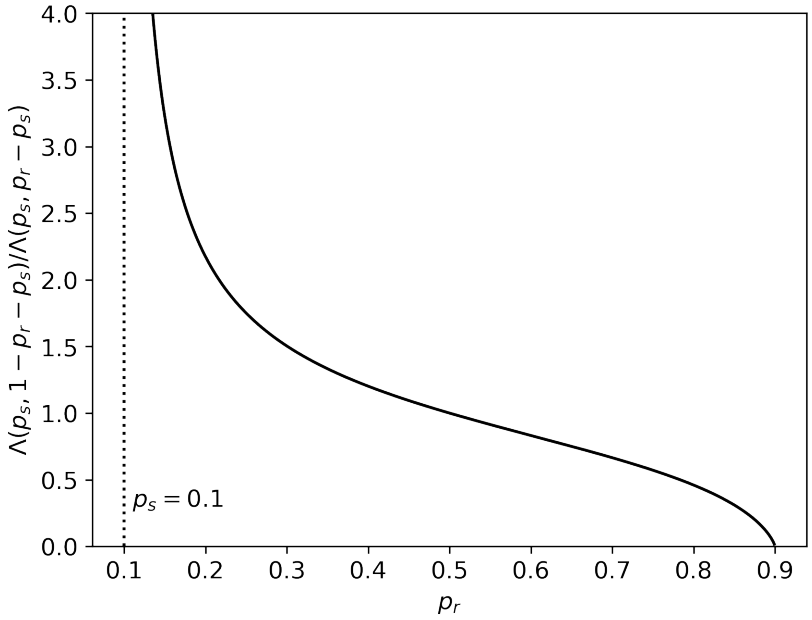


Figure 10: Illustration of the ratio of the cost of misinformation over ignorance in a cooperative setting, which grows as p_r becomes smaller (Bob becomes more opinionated), and tends to infinity as $p_r \longrightarrow p_s = 0.1$.

$x \in \{0,1\}^n$ and set of indices $\xi \subset \{0,1,\ldots,n-1\}$, let x_ξ denote the $|\xi|$-long vector obtained by extracting from x only the indices given by ξ. The algorithm starts by finding the smallest positive integer J such that the equation

$$[M \cdot G_J]_\Psi = [X]_\Psi \tag{19}$$

can be solved for some $M \in GF(2)^{1\times J}$. Exploiting δ Elias coding [26], the sender then sends the integer J to the receiver using

$$\mathbf{len}(\mathrm{elias}_\delta(J))$$

bits, followed by the J bits in the message M. The receiver then decodes J and computes $M \cdot G_J$ to retrieve the partition.

We now analyze the expected performance of this algorithm:

$$\begin{aligned} n^{-1}E_J\left[\mathbf{len}(\mathrm{elias}_\delta(J))\right] &\leq E_J\left[J + \log_2 J + 2\log_2(\log_2 J) + 3\right] \\ &= n^{-1}E_\Psi\left[E_J\left[J + \log_2 J + 2\log_2(\log_2 J) + 3|\Psi\right]\right] \\ &\leq n^{-1}E_\Psi\left[E_J\left[J|\Psi\right] + \log_2 E_J\left[J|\Psi\right] + 2\log_2(\log_2 E_J\left[J|\Psi\right]) + 3\right]. \end{aligned} \tag{20}$$

We next upper bound $E_J\left[J|\Psi\right]$. A sufficient condition to be able to solve Equation (19) is that $[G_J]_\Psi$ has full-row rank, that is, the dimension of the space spanned by the rows of $[G_J]_\Psi$ is exactly $|\Psi|$.

Let W_i be the smallest integer such that the row subspace spanned by $[G_{W_i}]_\Psi$ has dimension i. We can then write

$$W_{|\Psi|} = W_1 + \sum_{i=1}^{|\Psi|-1} W_{i+1} - W_i. \tag{21}$$

The probability that a vector drawn uniformly from $GF(2)^{|\Psi|}$ is nonzero, and therefore spans a space of dimension 1, is $1 - 2^{-|\Psi|}$; hence,

$$E\left[W_1|\Psi\right] = \frac{1}{1-2^{-|\Psi|}}. \tag{22}$$

We now analyze the difference $E[W_{i+1} - W_i|\Psi]$. Note that, by definition, $[G_{W_i}]_\Psi$ spans a subspace of dimension i, which must consist of exactly 2^i elements. If one chooses, uniformly at random, an element from $GF(2)^{|\Psi|}$, the probability that it lies within the subspace spanned by $[G_{W_i}]_\Psi$ is $2^{i-|\Psi|}$. As a consequence, the probability that the subspace spanned by the rows of $[G_{W_i}]_\Psi$ with such a random vector having dimension $i+1$ is $1 - 2^{i-|\Psi|}$, and thus

$$E[W_{i+1} - W_i|\Psi] = \frac{1}{1-2^{i-|\Psi|}}. \tag{23}$$

Observing that $J \leq W_{|\Psi|}$, we obtain

$$E_J\left[J|\Psi\right] \leq \sum_{i=0}^{|\Psi|-1} \frac{1}{1-2^{i-|\Psi|}} \leq |\Psi| + 2.$$

Continuing from inequality (20), we obtain an upper bound on performance as

$$n^{-1}E_\Psi\left[|\Psi| + \log_2(|\Psi|+2) + 2\log_2\left(\log_2(|\Psi|+2)\right) + 5\right].$$

To understand how good this bound is, we assume that the entries of X_1^n are drawn i.i.d. from $\{0,1,2\}$ using probability masses $(p_0, p_1, 1-p_0-p_1)$ and therefore $E\left[|\Psi|\right] = n(p_0+p_1)$, which results in the following upper bound on average performance:

$$p_0 + p_1 + \frac{\log_2\left(n(p_0+p_1)+2\right)}{n} + 2\frac{\log_2\left(\log_2\left(n(p_0+p_1)+2\right)\right)}{n} + \frac{5}{n}. \tag{24}$$

Now assume that $p_0 = p_1$. The achievable Shannon limit for this setting is given by

$$\Lambda(p_0,p_1) = \Lambda(p_0,p_0) = 2p_0,$$

which is the same performance as in (24) asymptotically as n grows. If $p_0 \neq p_1$, then $\Lambda(p_0,p_1) < p_0 + p_1$ and thus our linear code construction is not optimal.

4.2 Nonlinear codes for targeted queries

We next present an example of a small nonlinear code for the case $n = 6$ that is optimal. In this setting, we assume that $X_1^n \in \{0, 1, 2\}$ with exactly one entry of X_1^n equal to 0, and exactly one entry equal to 1.

One way to send a partition that separates the (single entry) sets $\{i : X_i = 0\}$ and $\{i : X_i = 1\}$ is to send an integer in the set $\{1, 2, 3, 4, 5, 6\}$ identifying the one element in, say, the first set. The cost of this is $\log_2 6$ bits. Alternately, one can scan each of the 4 codewords below to find one that matches with $X_0X_1X_2X_3X_4X_5$ on the 0s and 1s, regarding the 2s as "don't care":

sender vector ($X_i \in \{0, 1, 2\}$)	X_0	X_1	X_2	X_3	X_4	X_5
first codeword	0	0	0	1	1	1
second codeword	0	1	1	0	1	0
third codeword	1	0	1	1	0	0
fourth codeword	1	1	0	0	0	1

. (25)

We observe that any two columns of this binary matrix contain at least one row with the pattern "0 1" and one row with the pattern "1 0". For example, if $X_0 = 1, X_1 = 2, X_2 = 2, X_3 = 2, X_4 = 0, X_5 = 2$ the third and fourth codewords are valid codewords.

Therefore, one can always find one such codeword, which can be specified using $2 < \log_2 6$ bits.

This small example can in fact be extended easily. Notice that the columns of the matrix are exactly the set of all binary patterns with 2 ones (in the parlance of coding theory, the columns have weight 2), and thus this poses the interesting question of what the properties are of matrices whose column weights are a constant. One such property is easy to deduce, as stated in the following result.

Lemma 3 (Constant column weight codes) *Let c be a $t \times n$ binary matrix where every column has exactly the same weight w, and any two columns are different. Then the result code partitions any two sets each comprising exactly one (but different) integer in the set $\{0, \ldots, n-1\}$.*

Proof. Let i, j be the indices of any two distinct columns of the matrix c. The problem is to demonstrate that there is a row k such that $[c_{k,i}, c_{k,j}] = [0, 1]$ and that there is another row k' where $[c_{k',i}, c_{k',j}] = [1, 0]$. Supposing that neither of these conditions is true, then we deduce that $[c_{k,i}, c_{k,j}] \in \{[0, 0], [1, 1]\}$ for all $0 \leq k < t$ and thus necessarily the two columns are identical, which contradicts the assumption of the lemma. Suppose that, say, the first condition is true, but not the second one. Then it must be the case that the second column indexed by j has a strictly larger weight than the column indexed by i, which is also a contradiction of the assumptions in the lemma. The case where the second condition is true but not the first one is dealt with similarly. □

It is possible to obtain, for any given desired length n, a crude bound on the minimum number of rows t in a $t \times n$ binary matrix c that partitions two sets each comprising exactly one non-overlapping integer in $\{0, \ldots, n-1\}$. Take any one row of the matrix c, and assume it has n_0 zeros and n_1 ones. The number of patterns with exactly one 0, one 1, and the rest don't cares, that can be handled by any one row is at most $n_0 n_1 \leq n^2/4$. Therefore the entire matrix t can handle at most $tn^2/4$ patterns. There are a total of $n(n-1)$ patterns that we need to handle, and therefore the following relation must always hold:

$$t \geq 4\left(1 - \frac{1}{n}\right).$$

Rounding up (since t is an integer), we see that at least 4 rows are needed for any value of $n > 1$, showing that our 4×6 code is optimal in this sense. This bound is obviously too loose for anything other than $n = 6$.

4.3 Linear codes for the the case Alice does not know what Bob knows

Let $X_1^n, Y_1^n \in GF(2)^n$ represent two random vectors with the property that $\{i : X_i = 1\} \subseteq \{i : Y_i = 1\}$. We assume that Alice knows X_1^n but not Y_1^n (other than the condition above), and that Bob knows Y_1^n. The goal is to efficiently transmit X_1^n to Bob. The connection to the problem alluded to in the title of this subsection arises by identifying X_1^n, Y_1^n with the kernels of `S` and `R`, respectively.

In what follows we show a practical method based on linear codes. Let $\Delta > 0$ be an integer that is a design parameter. For convenience, we assume that Alice and Bob share a matrix G with dimensions $(n+\Delta) \times n$ and entries drawn uniformly and independently at random from $GF(2)$.

Let $\Psi = \{i : Y_i = 1\}$. The method starts with Bob transmitting to Alice the integer $|\Psi|$. At this point both Alice and Bob will keep only the first $|\Psi| + \Delta$ rows of G, denoted $G_{|\Psi|+\Delta}$. Alice sends to Bob the bits resulting from the multiplication $G_{|\Psi|+\Delta} X_1^n$, where X_1^n is interpreted as a column vector. Let $[G_{|\Psi|+\Delta}]_\Psi$ denote the matrix obtained by extracting from $G_{|\Psi|+\Delta}$ the columns implied by the indices Ψ; note that this matrix is computable by Bob but not Alice. Bob then attempts to solve the equation

$$[G_{|\Psi|+\Delta}]_\Psi z = G_{|\Psi|+\Delta} X_1^n \tag{26}$$

for a unique z. If such a unique z exists, Bob can retrieve X_1^n by making use of the fact that the only entries of X_1^n that could possibly be equal to 1 must have an index contained in Ψ and the fact that

$$[X_1^n]_\Psi = z,$$

where $[X_1^n]_\Psi$ denotes the entries of X_1^n subset to the indices in Ψ. Therefore X_1^n can be recovered from z by lifting the latter using the indices Ψ.

By construction, (26) has at least one solution. If the $|\Psi|$ columns of $[G_{|\Psi|+\Delta}]_\Psi$ are linearly independent, then that solution must be unique. The probability that these columns are linearly independent is given by $\Pi_{i=0}^{|\Psi|-1}(1 - 2^{i-|\Psi|-\Delta})$ and can be lower bounded in this manner:

$$\begin{aligned}
\Pi_{i=0}^{|\Psi|-1}(1 - 2^{i-|\Psi|-\Delta}) &= \exp\left(\sum_{i=0}^{|\Psi|-1} \log(1 - 2^{i-|\Psi|-\Delta})\right) \\
&\geq \exp\left(\sum_{i=0}^{|\Psi|-1} -\frac{2^{i-|\Psi|-\Delta}}{1 - 2^{i-|\Psi|-\Delta}}\right) \\
&= \exp\left(\sum_{i=1}^{|\Psi|} -\frac{2^{-i-\Delta}}{1 - 2^{-i-\Delta}}\right) \\
&\geq \exp\left(\sum_{i=1}^{|\Psi|} -2^{-i-\Delta+1}\right) \\
&\geq \exp\left(-2^{-\Delta+1} \sum_{i=1}^{|\Psi|} 2^{-i}\right) \\
&\geq \exp\left(-2^{-\Delta+1}\right).
\end{aligned}$$

Bob can signal to Alice success or failure in finding a unique z with a single bit, and in the case of failure, Alice can simply send X_1^n verbatim by sending n bits.

The normalized expected number of bits transmitted in either direction is then upper bounded as follows:

$$\begin{aligned}
&\left(1 - \exp\left(-2^{-\Delta+1}\right)\right) + \frac{1}{n}\left(E\left[|\Psi|\right] + \Delta + E\left[\mathbf{len}(\mathrm{elias}_\delta(|\Psi|))\right] + 1\right) \\
&\quad\leq\ 2^{-\Delta+1} + O\left(2^{-2\Delta}\right) + \frac{1}{n}\left(E\left[|\Psi|\right] + \Delta + \log_2 E\left[|\Psi|\right] + \log_2 \log_2 E\left[|\Psi|\right] + 4\right).
\end{aligned}$$

Define $q_1 \stackrel{\Delta}{=} n^{-1} E\left[|\Psi|\right]$. With this definition, then the upper bound may be summarized as

$$q_1 + O\left(\frac{\log_2 n}{n}\right) + 2^{-\Delta+1} + O\left(2^{-2\Delta}\right), \tag{27}$$

where the $O(\cdot)$ terms are to be interpreted with respect to the $n \to \infty$ and $\Delta \to 0$ limits, respectively.

To understand how good the above bound is, assume temporarily that Y_1^n has entries drawn i.i.d. from $\{0,1\}$ using probability masses $\{1-q_1, q_1\}$, and that each X_i is drawn using the conditional distribution

$$\begin{aligned} P(X_i = 1|Y_i = 1) &= p_1/q_1, \\ P(X_i = 0|Y_i = 0) &= 1. \end{aligned}$$

Under these assumptions for X_1^n, Y_1^n, a lower bound is given by $n^{-1}H(X_1^n|Y_1^n) = q_1 H_{\text{bin}}(p_1/q_1)$. Comparing this lower bound to (27), we see that if $q_1 = 2p_1$, n is sufficiently large and Δ is sufficiently small, then the linear coding algorithm can arbitrarily approach the lower bound. For other choices of q_1, p_1, nonetheless the algorithm is not optimal.

4.4 Experimental setup

The goal of this section is to illustrate the possible gains that one may expect from a practical semantic communication system compared to a classical one, and to also show the existing gap between such semantic communication system with respect to the ultimate bound given by Λ.

A first problem is the fact it is possible to portray classical communication systems to be nearly arbitrarily inefficient when compared to semantic ones. The reason is the multiple different ways in which sentences can express the same underlying semantic content; classic compression systems must be faithful to the original sentence itself whereas semantic systems as regarded in this article can take advantage of the intended meaning of the symbols within the sentence. To illustrate this problem, note that given $H(\kappa(\mathtt{S})|\mathtt{S}) = 0$,

$$H(\mathtt{S}) = H(\mathtt{S}|\kappa(\mathtt{S})) + H(\kappa(\mathtt{S})) \geq H(\kappa(\mathtt{S})). \tag{28}$$

The gap $H(\mathtt{S}|\kappa(\mathtt{S}))$ can be very large; consider, for example, propositional logic in two variables X_1, X_2 and note that

$$\begin{gathered} (X_1 \wedge X_2) \vee \neg X_3 \\ \neg(\neg(X_1 \wedge X_2) \wedge X_3) \\ \neg((\neg X_1 \vee \neg X_2) \wedge X_3) \\ \neg((\neg X_1 \wedge X_3) \vee (\neg X_2 \wedge X_3)) \\ \neg((\neg X_1 \wedge X_3) \vee (\neg X_2 \wedge X_3)) \wedge (X_1 \vee \neg X_1) \\ \neg((\neg X_1 \wedge X_3) \vee (\neg X_2 \wedge X_3)) \wedge (X_1 \vee \neg X_1) \wedge (X_1 \vee \neg X_1) \wedge (X_1 \vee \neg X_1) \wedge (X_1 \vee \neg X_1) \wedge (X_1 \vee \neg X_1) \end{gathered}$$

are all logically equivalent sentences. To be clear, one can legitimately expect semantic communication systems to take advantage of the observation (28), but one must exercise caution and not overstate this. Our paper's theoretical results not only leverage the phenomenon (28), but go beyond and exploit more delicate findings on how targeted queries may admit unusually efficient representations or how, in some occasions, communication is surprisingly just as efficient when Alice doesn't know Bob's sentence $\mathtt{R}$ as when she does know it.

Our experimental evaluation methodology follows the philosophy of always choosing a stronger "classical" (non-semantic) baseline to compare against whenever a choice is on the table, even if these baselines start to become more semantic in nature. In particular:

- On purpose, we will forego the type of advantage that stems from observation (28) even though we can legitimately claim it.
- Logic sentences will be optimized so that they can be represented more compactly by "classical" compressions systems.

4.4.1 Scenarios

We demonstrate two contrasting scenarios, derived from Theorem 7 and Theorem 6, respectively:

- **(Targeted query with a shared statement)** After the communication, Bob will be able to prove $\mathtt{Q}$, which satisfies $\mathtt{S} \vdash \mathtt{Q}$. Both Alice and Bob know $\mathtt{R}$; only Alice knows $\mathtt{S}$; we simplify to the setting $\mathtt{S} \vdash \mathtt{R}$.
- **(Alice doesn't know what Bob knows)** After the communication, Bob will be able to prove all that Alice can. Only Bob knows $\mathtt{R}$, and only Alice knows $\mathtt{S}$. We assume $\mathtt{S} \vdash \mathtt{R}$.

4.4.2 Test case generation

The reader shall recall that all of our theorems have an upper bound that holds in significant generality, and a corresponding lower bound that holds under additional assumptions. We have chosen distributions for $\mathtt{S}, \mathtt{Q}$ and $\mathtt{R}$ that meet these additional assumptions to enable comparisons with both upper and lower bounds. In particular, their distribution is chosen so that:

- $\mathtt{S} \vdash \mathtt{Q}$, $\mathtt{Q} \vdash \mathtt{R}$ (and therefore $\mathtt{S} \vdash \mathtt{R}$);
- The random tuples $\{(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}), \vec{\kappa}(\mathtt{R}))_i\}$ are i.i.d.;
- The Markov chain $\mathtt{R} \to \kappa(\mathtt{R}) \to (\kappa(\mathtt{S}), \kappa(\mathtt{R}))$ holds.

A simple way to construct $\mathtt{S}, \mathtt{Q}, \mathtt{R}$ with these properties is to choose their underlying kernels first, and then construct sentences from those kernels. If the reader has, instead, a collection of sentences meeting the entailment constraints which are believed to come from an otherwise unknown distribution, then the reader may still obtain value from our paper by considering only the upper bounds.

We generated random kernels meeting all the three constraints above on sets of 10 variables to test our methods in practice. For each of the kernels we produced a generalized decision tree with the given kernel as its set of satisfying truth value assignments. We then post-processed each sentence by writing it using postfix notation, compressing variable names and removing spaces. For the targeted query scenario we have chosen $p_r = 0.5$, $p_s = 0.075$ and p_q by sampling the range $[p_s, p_r)$ uniformly. For the scenario where Alice does not know what Bob knows, we set $p_s = p_q = 0.075$ and let p_r range in $(p_s, 0.5]$. We generated 1000 test cases for each choice of (p_s, p_q, p_r) considered. This data could be used, through appropriate subselection, to illustrate all of the theorems in this article in fullness with the exception of Theorem 7. In this theorem the assumption $\mathtt{S} \vdash \mathtt{R}$ is simply omitted. Therefore only those subcases where it is true can be demonstrated with this data.

4.4.3 Method of Generating Kernels

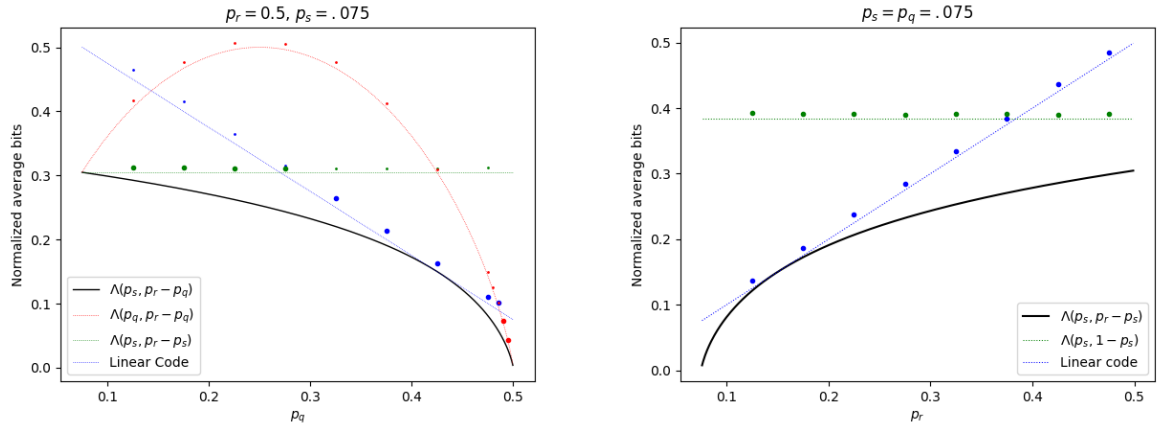
Since the sender's kernel is a subset of the receiver's kernel, and, moreover, the query is provable by the sender but not the receiver, it follows that the sender's kernel is contained in the kernel associated with the query, which in turn is contained in the receiver's kernel. It then follows that we can go through all the possible truth value assignments for the 10 variables and, for each assignment μ, generate a random number $\eta \in [0, 1]$ where, if $\eta \leq p_r$, we place μ in the receiver's kernel. If additionally $\eta \leq p_q$, then we can place μ in the query kernel, and if additionally to that $\eta \leq p_s$, then we can place μ in the sender's kernel. Pseudocode for this simple procedure is given in Algorithm 1 in Appendix C.1.

4.4.4 Method of Generating Generalized Decision Tree Sentences

The gist of the decision tree approach is to recursively call a method $\mathtt{GENERATE\_GDT\_FOR\_KERNEL}(\cdot)$ to find a most balanced variable $\mathtt{X_b}$ among the variables $\mathtt{X_1}, ..., \mathtt{X_n}$ – meaning a variable where $\mathtt{X_b} = 1$ and $\mathtt{X_b} = 0$ as close to equally as possible among the kernel elements – and then output

$$(\mathtt{X_b} \wedge \mathtt{GENERATE\_GDT\_FOR\_KERNEL(pos\_kernel)}) \vee$$
$$(\neg \mathtt{X_b} \wedge \mathtt{GENERATE\_GDT\_FOR\_KERNEL(neg\_kernel)}),$$

where $\mathtt{pos\_kernel}$ is the reduced kernel on the $\mathtt{n-1}$ variables $\mathtt{X_1}, ..., \mathtt{X_n}$ but with $\mathtt{X_b}$ excluded, and in the original kernel $\mathtt{X_b} = 1$, while $\mathtt{neg\_kernel}$ is the reduced kernel on the $\mathtt{n-1}$ variables $\mathtt{X_1}, ..., \mathtt{X_n}$ but with $\mathtt{X_b}$ excluded, and in the original kernel $\mathtt{X_b} = 0$. The generalized version of this heuristic is a bit more nuanced so that variables that are effectively constant with respect to the kernel (in other words, there is a variable $\mathtt{X_i}$ such that either $\mathtt{X_i} = 1$ or $\mathtt{X_i} = 0$ for every kernel element) are efficiently split out, and when there are $\mathtt{k}$ variables remaining and the kernel size is either $2^{\mathtt{k}}$ or 0 the routine immediately terminates, outputting an empty string (equivalent to outputting True) in the former case, and outputting a False indicator in the latter case. The pseudocode for implementing the generalized decision tree algorithm, given a kernel, along with an accompanying detailed description and worked example are provided in Appendix C.2. There is a considerable literature on the use of decision trees to represent Boolean functions. See, for example, [14, 52, 56].



(a) Semantic communication systems for targeted queries with a shared sentence with $p_r = 0.5$, $0.075 < p_q \leq 0.5$

(b) Semantic communication systems for when Alice doesn't know what Bob knows with $p_s = p_q = 0.075$

Figure 11: Comparison of practical semantic communication methods against the Shannon bound $\Lambda(p_s, p_r - p_q)$ for two scenarios.

4.5 Experimental results

4.5.1 Semantic communication systems

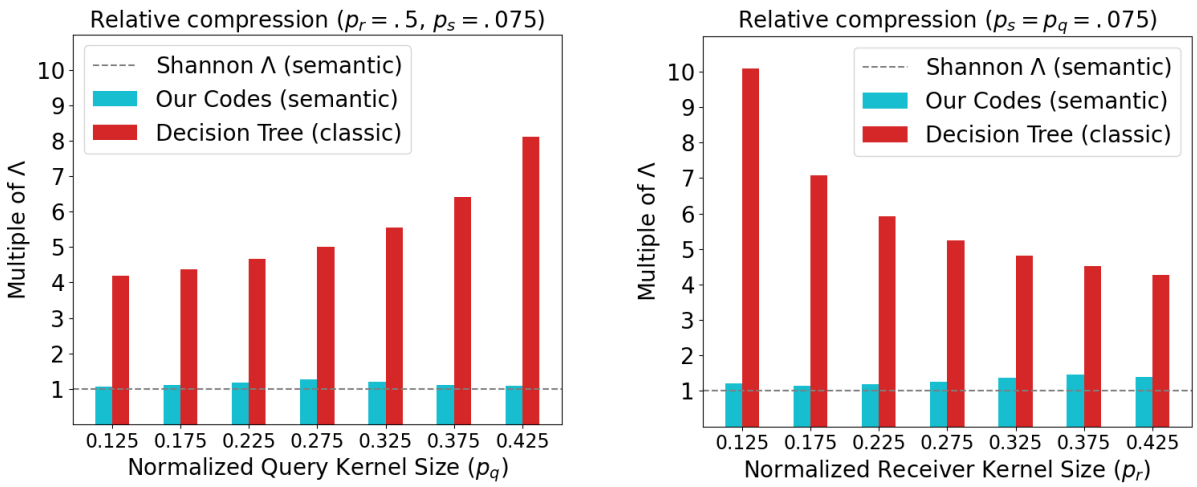
In a first set of experiments, we aim to demonstrate the performance of our practical semantic communication techniques against the new theoretical bounds. We have implemented the linear codes described in Subsections 4.1 and 4.3 which are used to implement the two scenarios in Subsection 4.4.1. Due to the fact that these linear codes are not always optimal, we augmented these systems with "näive" semantic communication strategies based on efficient lossless transmission of kernels. In certain parameter ranges these are better than the linear codes so in those cases, we can use them instead. In the case of targeted queries, the task of allowing Bob to prove $\mathtt{Q}$ can be alternately accomplished by sending to him $\kappa(\mathtt{Q})$ or $\kappa(\mathtt{S})$, whichever is cheapest. The results, averaged across all 1000 test cases for each choice of parameters, can be found in Figure 11a, where in blue, we illustrate the performance of linear codes, and in green and red we illustrate the performance of sending $\kappa(\mathtt{S})$ and $\kappa(\mathtt{Q})$ using enumerative source codes [20], respectively. In each of these three practical systems, we include a line of the same color that is a lower bound on performance for the specific technique. The bolder dots are those that are closest to the limiting Shannon bound, which is the bold, lowest plot shown in the figure. Similarly, in the case that Alice doesn't know what Bob knows, the task of allowing Bob to prove $\mathtt{S}$ can be alternately accomplished by sending to him $\kappa(\mathtt{S})$, also using enumerative source coding [20]. The results of this experiment can be found in Figure 11b. It can be appreciated that our practical codes can be quite efficient in some scenarios, but in general, more work is needed to develop practical codes that meet the Shannon bound in general.

4.5.2 Classic communication systems

In a second set of experiments, we want to contrast practical classic systems with semantic ones. In these experiments, we will reuse the best results from practical methods in Subsection 4.5.1 and compare those with classic compression systems.

We define classic compressions systems as ones that may only use the sentence as presented to the communication system, and not perform operations on it with awareness of its underlying semantic content. These are examples of such systems:

- **(Targeted query with a shared statement)** Using the Decision Tree representation of a sentence, employ a standard compression algorithm (in particular, gzip, lzma, bzip2) to compress $\mathtt{S}$ and $\mathtt{Q}$; choose the best representation and send that one.



(a) Targeted queries with a shared sentence ($p_r = .5$, $p_s = .075$)

(b) Alice does not know Bob's sentence($p_s = p_q = .075$)

Figure 12: Contrasting classic and semantic communication systems relative to the semantic Shannon limit

- **(Alice doesn't know what Bob knows)** The same as above, but considering only $\mathtt{S}$ as in this scenario, $\mathtt{Q} = \mathtt{S}$.

We provide an additional advantage to the type of classic communication systems used above: we compress all 1000 samples simultaneously, which allow the compression algorithms described above to leverage patterns that only emerge when more data is available. In contrast, the semantic communication systems are compressing only one instance at a time, which is a much more difficult target. We reiterate that by using Decision Trees, we have already used semantic concepts to benefit the classical baseline. The results of these experiments can be found in Figures 12a and 12b. These plots normalize the performance of the practical semantic or classical systems against the corresponding Shannon bound $\Lambda(p_s, p_r - p_q)$ and $\Lambda(p_s, p_r - p_s)$, respectively. We acknowledge that in the realm of classical communication systems we should consider Slepian-Wolf compression algorithms [68] as a baseline. This is particularly difficult to do as we are not aware of practical instances of such algorithms that would be applicable to the complex statistical distributions present in $\mathtt{S}, \mathtt{R}$; we leave this comparison as an open item for future research.

5 Formal information-theoretic results

The purpose of this section is to formally state and prove each of the information-theoretic results we offer in this paper.

5.1 Full ignorance

In this first problem, Alice is in possession of a logic sentence and wishes to communicate to Bob with the goal that whatever logic inferences Alice can make starting from that sentence, Bob, who otherwise knows nothing, can do the same. Crucially, Bob is not required to reproduce the particular way in which Alice's logic sentence is represented in her mind, but rather it just needs to be able to retrieve a functionally equivalent sentence. The intention of this result is mainly to introduce notation and concepts as we build up to more interesting cases.

As with all our theorems, Theorem 2 has a lower bound on performance, as well as an algorithm to achieve that performance. We sketch next that algorithm, and then formally prove both upper and lower bounds. Alice and Bob meet in preparation for a future where $\mathtt{s}$ is revealed to Alice. Given that any possible query that is entailed by $\mathtt{s}$ must be proved by Bob (see Figure 13, which illustrates the kernels of two possible queries $\mathtt{q}'$ and $\mathtt{q}'$ with the property that $\mathtt{s} \vdash \mathtt{q}'$ and $\mathtt{s} \vdash \mathtt{q}'$), and given that this must be done as efficiently

as possible, they agree that the sensible goal is not to send her sentence $\mathtt{s}$ verbatim, but to send the kernel $\kappa(\mathtt{s})$ instead. From this, Bob can use the function ℓ to recover a functionally equivalent sentence $\hat{\mathtt{s}}$.

To implement this, our suggested algorithm is for Alice to first send to Bob the *size* of the kernel. Once this size is transmitted, all possible kernels can be enumerated by both Alice and Bob and then Alice can simply send the index of the kernel that she has in her possession. To implement the idea above we need two tools: a means for efficiently sending integers (to encode the size) and a way to efficiently encode the kernel indices. The same needs will recur in all of our results, so we pause here to introduce two widely known tools to accomplish this.

For integer encoding, we use δ Elias coding [26]. This code assigns codewords to each integer, where the length of a codeword for the integer $n \geq 1$ is given by

$$\begin{aligned} \mathbf{len}(\text{elias}_\delta(n)) &= \lfloor \log_2(n) \rfloor + 2\lfloor \log_2(1 + \lfloor \log_2(n) \rfloor) \rfloor + 1 \text{ bits} \\ &\leq \log_2(n) + 2\log_2\log_2(n) + 3 \text{ bits}. \end{aligned}$$

Our rationale for choosing this particular code is that it is both convenient and sufficient to prove asymptotically good results. For the problem of sending a kernel index, techniques for enumerating members of a set and efficiently sending and receiving indices from such enumeration can be found in the work of Cover on enumerative source coding [20]. This overall strategy is described at the bottom of Figure 13.

We now formally present our main result for this case followed by its proof.

Theorem 2 (Bob can prove all Alice is able to) *Let $(L, \mathcal{M}, \kappa, \ell)$ be a Logic System. Let $\mathtt{S} \in \mathcal{L}$ have a kernel that follows a p_s-law. Let the encoder f and decoder g be functions*

$$\begin{aligned} f : \mathcal{L} &\rightarrow \{0,1\}^*, \\ g : \{0,1\}^* &\rightarrow \mathcal{L}, \end{aligned}$$

respectively. Then,

$$\min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S}}\left[\mathbf{len}(f(\mathtt{S}))\right] \leq |\mathcal{M}|^{-1} H(\kappa(\mathtt{S})) + \frac{1}{|\mathcal{M}|} \leq H_{bin}(p_s) + O\left(\frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|}\right),$$

where the minimization is over f, g such that the image of f is prefix-free and such that, if $\mathtt{s} \vdash \mathtt{q}$, then $\mathtt{s} \vdash g(f(\mathtt{s}))$ and $g(f(\mathtt{s})) \vdash \mathtt{q}$. Under the same assumptions for f, g, the following lower bound holds:

$$|\mathcal{M}|^{-1} H(\kappa(\mathtt{S})) \leq \min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S}}\left[\mathbf{len}(f(\mathtt{S}))\right]. \tag{29}$$

Furthermore, if we additionally have that the $\{\vec{\kappa}(\mathtt{S})_j\}_{j=1}^{|\mathcal{M}|}$ are i.i.d., then

$$\Lambda(p_s, 1-p_s) = H_{bin}(p_s) \leq \min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S}}\left[\mathbf{len}(f(\mathtt{S}))\right].$$

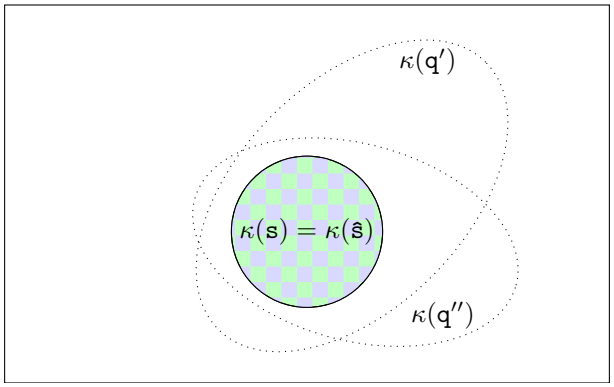
Proof. We begin with the proof of the lower bound. Let f, g satisfy the conditions for the minimization. The starting point for this proof is the classical result from information theory proved using Kraft's inequality as follows.

Lemma 4 *Let $\{l_i\}$ be the codeword lengths of a binary code that is prefix-free. Assume a distribution over these codewords, and let C be a random codeword drawn according to that distribution. Then,*

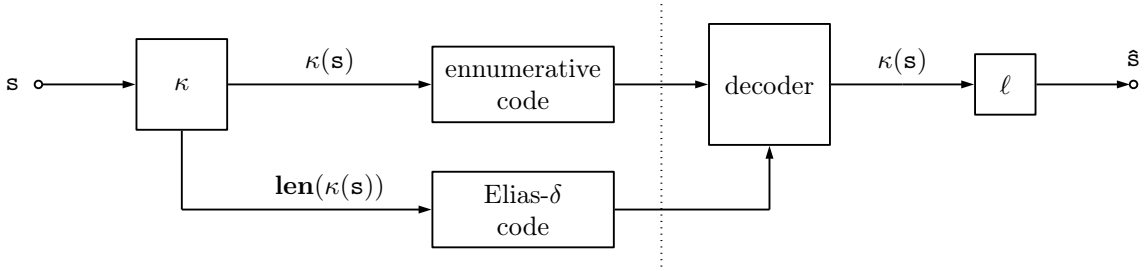
$$E_C\left[\mathbf{len}(C)\right] \geq H(C).$$

Using this result together with the assumption that the image of f is prefix-free, we can write

$$E_{\mathtt{S}}\left[\mathbf{len}(f(\mathtt{S}))\right] = E_{f(\mathtt{S})}\left[\mathbf{len}(f(\mathtt{S}))\right] \geq H(f(\mathtt{S})) \geq H(g(f(\mathtt{S}))) \geq H(\kappa(g(f(\mathtt{S})))) = H(\kappa(\hat{\mathtt{S}})), \tag{30}$$



(a)



(b)

Figure 13: Proof strategy for Theorem 2.

where the last two inequalities follow from the fact that deterministic functions of random variables cannot increase entropy.

For any $\mathtt{s}, \mathtt{q}$ such that $\mathtt{s} \vdash \mathtt{q}$, we are assuming that $\mathtt{s} \vdash g(f(\mathtt{s}))$, $g(f(\mathtt{s})) \vdash \mathtt{q}$ and therefore using the definition of a Logic System, we have

$$\kappa(\mathtt{s}) \subseteq \kappa(g(f(\mathtt{s}))) \subseteq \kappa(\mathtt{q}). \tag{31}$$

In particular, choosing $\mathtt{q} = \mathtt{s}$, we obtain $\kappa(\mathtt{s}) = \kappa(g(f(\mathtt{s}))) = \kappa(\hat{\mathtt{s}})$. This is geometrically described in Figure 13-a, which illustrates the kernels of two queries satisfying $s \vdash q'$, $s \vdash q''$ (depicting the choice of q in the right hand side of (31)), as well as the conclusion that $\kappa(\mathtt{s}) = \kappa(\hat{\mathtt{s}})$. Substituting this in the right hand side of (30), we conclude

$$\begin{aligned} E_{\mathtt{S}}\left[\mathbf{len}(f(\mathtt{S}))\right] &\overset{(a)}{\geq} H(\kappa(\mathtt{S})) \\ &\overset{(b)}{=} H(\vec{\kappa}(\mathtt{S})_1, \ldots, \vec{\kappa}(\mathtt{S})_{|\mathcal{M}|}) \\ &\overset{(c)}{=} \sum_{i=1}^{\mathcal{M}} H(\vec{\kappa}(\mathtt{S})_i) \\ &\overset{(d)}{=} |\mathcal{M}| H_{\text{bin}}\left(p_s\right), \end{aligned}$$

where (a) already proves (29), (b) follows from the definition of $\vec{\kappa}(\cdot)$ in (13), (c) follows from the independence assumption, and (d) follows from the p_s-law assumption combined with the "identically distributed" assumption. This completes the proof of the lower-bound result.

To prove the upper bound, we construct a code as follows. First, the sentence $\mathtt{s}$ is mapped to its kernel, $\kappa(\mathtt{s})$; let P_K denote the probability distribution governing $\kappa(\mathtt{S})$. Then we use a Shannon code to encode this

kernel, which uses a code length of

$$\left\lceil \log_2 \left(\frac{1}{P_K(\kappa(\mathtt{s}))} \right) \right\rceil \leq \log_2 \left(\frac{1}{P_K(\kappa(\mathtt{s}))} \right) + 1. \tag{32}$$

We point the reader to the early discussion of this quantity in (4). When substituting a random $\mathtt{S}$ in lieu of $\mathtt{s}$, the expectation of the right hand side of the expectation of (32) is

$$H(\kappa(\mathtt{S})) + 1. \tag{33}$$

One then writes

$$\begin{aligned}
H(\kappa(\mathtt{S})) &\overset{(a)}{=} H(\vec{\kappa}(\mathtt{S})_1, \ldots, \vec{\kappa}(\mathtt{S})_{|\mathcal{M}|}) \\
&\overset{(b)}{\leq} \sum_{i=1}^{\mathcal{M}} H(\vec{\kappa}(\mathtt{S})_i) \\
&\overset{(c)}{=} \sum_{i=1}^{\mathcal{M}} H_{\text{bin}}\left(P(\vec{\kappa}(\mathtt{S})_i = 1)\right) \\
&\overset{(d)}{\leq} |\mathcal{M}| H_{\text{bin}}\left(|\mathcal{M}|^{-1} \sum_{i=1}^{\mathcal{M}} P(\vec{\kappa}(\mathtt{S})_i = 1) \right) \\
&\overset{(e)}{=} |\mathcal{M}| H_{\text{bin}}\left(p_s\right) = |\mathcal{M}| \Lambda(p_s, 1 - p_s),
\end{aligned}$$

where (a) follows from the definition of $\vec{\kappa}(\cdot)$ in (13), (b) is a standard upper bound on the entropy of joint variables, (c) follows from the fact that $\vec{\kappa}(\mathtt{S})_i$ is a binary random variable, (d) follows from the concavity of entropy, and (e) follows from the assumption of $\mathtt{S}$ following a p_s-law.

We note that this result already proves the tighter upper bound of the Theorem and that such a tighter upper bound can be included in the results of our subsequent theorems, as explained in the discussion around (15). We include an additional argument nonetheless, which only proves the looser upper bound, but that agrees in style with the rest of the results in the article; see Figure 13. We first specify the encoder f. For a given size ξ of a kernel, there are a total of

$$\binom{|\mathcal{M}|}{\xi} \leq 2^{|\mathcal{M}| H_{\text{bin}}\left(\frac{\xi}{|\mathcal{M}|}\right)}$$

possible kernels of the same size. Let $\text{enum}_\xi : \{k \subseteq \mathcal{M} : |k| = \xi\} \to \{0,1\}^*$ be a function that maps each possible kernel of size ξ to a fixed-length binary encoding of the integers $\left\{1, \ldots, \binom{|\mathcal{M}|}{\xi}\right\}$, which is an integer that uniquely determines such a set. Hence, in particular,

$$\mathbf{len}(\text{enum}_{|\kappa(\mathtt{S})|}(\kappa(\mathtt{S}))) \leq |\mathcal{M}| H_{\text{bin}}\left(\frac{|\kappa(\mathtt{S})|}{|\mathcal{M}|} \right) + 1,$$

where in the above the binary entropy function $H_{\text{bin}}(p) = -p \log_2(p) - (1-p)\log_2(1-p)$ is evaluated on the random variable $|\kappa(\mathtt{S})|/|\mathcal{M}|$, and thus the result of the evaluation is also a random variable.

For a sentence kernel $\mathtt{s} \in \mathcal{L}$, we define the encoder f as a concatenation of two separate encodings:

$$f(\mathtt{s}) = \text{elias}_\delta(|\kappa(\mathtt{s})|)\text{enum}_{|\kappa(\mathtt{s})|}(\kappa(\mathtt{s})), \tag{34}$$

where the above is the result of the concatenation of two codes. Since both of the codes implied by each encoding are prefix-free, the concatenation is also prefix-free. Finally, we let g be the decoder that recovers the kernel $\kappa(\mathtt{s})$ from the output of f, and evaluates ℓ on that kernel.

Note that, by construction, if $\mathtt{q}$ is such that $\mathtt{s} \vdash \mathtt{q}$, then since $\kappa(\mathtt{s}) = \kappa(g(f(\mathtt{s})))$, we conclude $g(f(\mathtt{s})) \vdash \mathtt{q}$ as well, and thus we have met the conditions of the Theorem.

The estimate for the overall cost of the encoding can be done by separately estimating the length of the two encodings in (34), namely

$$|\mathcal{M}| H_{\text{bin}}\left(\frac{|\kappa(\mathtt{S})|}{|\mathcal{M}|}\right) + \log_2|\kappa(\mathtt{S})| + 2\log_2\left(\log_2|\kappa(\mathtt{S})|\right) + 4$$

bits. Taking the expectation with respect to $\mathtt{S}$, using the concavity $\cap$ of the logarithm and entropy functions, and normalizing by $|\mathcal{M}|$, we obtain a normalized upper estimate of

$$H_{\text{bin}}(p_s) + \frac{\log_2(p_s|\mathcal{M}|)}{|\mathcal{M}|} + 2\frac{\log_2\left(\log_2(p_s|\mathcal{M}|)\right)}{|\mathcal{M}|} + \frac{4}{|\mathcal{M}|}.$$

Note that, in this upper bound, at no point did we assume the more restrictive condition involving the i.i.d. assumptions of the lower bound, thus completing the proof of the Theorem. This pattern of the upper bound holding under much more general conditions will repeat throughout our other proofs. □

5.2 Partial Ignorance

In this Theorem, both Alice and Bob have access to the same background logic sentence $\mathtt{R}$. It is important to emphasize that in this setup, such background information is *not* known at the initial meeting between Alice and Bob, and rather, it is the result of both independently collecting the same such logic sentences from the environment. We refer the reader to Figure 14, which extends the corresponding Figure 13 by incorporating such background information. In some respects, the effect that the presence of the background $\mathtt{R}$ has in the problem is rather elementary: we still want to send somehow $\kappa(\mathtt{S})$, however we can now leverage the fact that both Alice and Bob know that such a kernel must be contained within $\kappa(\mathtt{R})$. In the same figure we show an updated strategy: both Alice and Bob first compute $\kappa(\mathtt{R})$, and then Alice uses it to enumerate all possible subsets of $\kappa(\mathtt{R})$ of size $\mathbf{len}(\kappa(\mathtt{S}))$, and sends $|\kappa(\mathtt{S})|$ followed by the index of the kernel she has in her possession. Bob then recovers $\kappa(\mathtt{S})$ and uses ℓ to recover a logic sentence.

Theorem 3 (Alice and Bob share a logic sentence) *Let $(L, \mathcal{M}, \kappa, \ell)$ be a Logic System. Let $\mathtt{S}, \mathtt{R} \in \mathcal{L}$ represent the sender's logic sentence and the shared logic sentence, with the property that $\mathtt{S} \vdash \mathtt{R}$ and, in particular, $\mathtt{S}, \mathtt{R}$ have kernels that follow a (p_s, p_r)-law. Let the encoder f and decoder g be functions*

$$\begin{aligned} f : \mathcal{L}^2 &\to \{0,1\}^*, \\ g : \{0,1\}^* \times \mathcal{L} &\to \mathcal{L}. \end{aligned}$$

Then

$$\min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{R}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{R}))\right] \leq \Lambda(p_s, p_r - p_s) + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right),$$

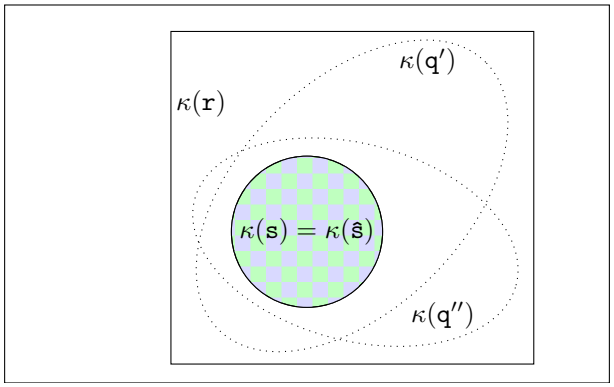
where the minimization is over f, g such that the image of $f(\cdot, \mathtt{r})$ is prefix-free for any choice of $\mathtt{r}$ and such that, if $\mathtt{s} \vdash \mathtt{r}$ and $\mathtt{s} \vdash \mathtt{q}$, then $\mathtt{s} \vdash g(f(\mathtt{s},\mathtt{r}),\mathtt{r})$ and $g(f(\mathtt{s},\mathtt{r}),\mathtt{r}) \vdash \mathtt{q}$. Furthermore if additionally the random variables $\{(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{R}))_j\}_{j=1}^{|\mathcal{M}|}$ are i.i.d. and $\mathtt{R} \to \kappa(\mathtt{R}) \to \kappa(\mathtt{S})$, then

$$\Lambda(p_s, p_r - p_s) \leq \min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{R}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{R}))\right].$$

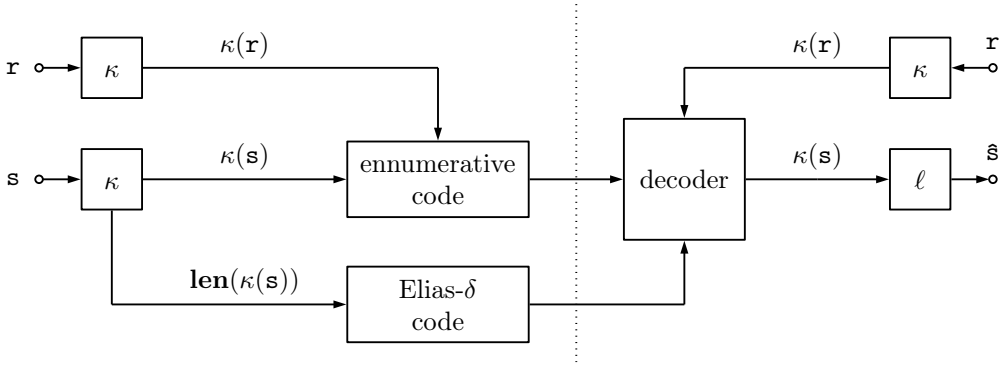
Proof. As with Theorem 2, we begin with the proof of the lower bound. Let f, g satisfy the conditions for the minimization. Paralleling the definition (13), for any given $k \subseteq \mathcal{M}$, let

$$k_j = \begin{cases} 1 & \text{if } \mu_j \in k \\ 0 & \text{otherwise} \end{cases}.$$

For the proof of the lower bound, we will rely on the following Lemma, which is provided without proof since it is elementary.



(a)



(b)

Figure 14: Proof strategy for Theorem 3.

Lemma 5 *For given $0 < p_a \leq p_b \leq 1$, let the random variables $A, B \in \{0,1\}$ be such that $EA = p_a$ and $EB = p_b$, and furthermore, $A \leq B$. Then conditioned on $B = 1$, the random variable A is distributed according to $\{1 - p_a/p_b, p_a/p_b\}$.*

We now derive

$$\begin{aligned}
E_{\mathtt{S},\mathtt{R}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{R}))\right] &= E_{f(\mathtt{S},\mathtt{R})}\left[\mathbf{len}(f(\mathtt{S},\mathtt{R}))\right] \\
&\overset{(a)}{=} E_{\mathtt{R}}\left[E_{f(\mathtt{S},\mathtt{R})|\mathtt{R}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{R}))|\mathtt{R}\right]\right] \\
&\overset{(b)}{\geq} H(f(\mathtt{S},\mathtt{R})|\mathtt{R}) \\
&\overset{(c)}{\geq} H(\kappa(g(f(\mathtt{S},\mathtt{R}),\mathtt{R}))|\mathtt{R}) \\
&= H(\kappa(\hat{\mathtt{S}})|\mathtt{R}) \\
&\overset{(d)}{=} H(\kappa(\mathtt{S})|\mathtt{R}) \\
&\overset{(e)}{=} H(\kappa(\mathtt{S})|\kappa(\mathtt{R})) \\
&= \sum_{k \subseteq \mathcal{M}} P(\kappa(\mathtt{R}) = k) H(\kappa(\mathtt{S})|\kappa(\mathtt{R}) = k) \\
&\overset{(f)}{=} \sum_{k \subseteq \mathcal{M}} P(\kappa(\mathtt{R}) = k) \sum_j H(\vec{\kappa}(\mathtt{S})_j | \vec{\kappa}(\mathtt{R})_j = k_j)
\end{aligned}$$

$$
\begin{aligned}
&\overset{(g)}{=} \sum_{k\subseteq\mathcal{M}} P(\kappa(\mathtt{R})=k) \sum_{j:k_j=0} H(\vec{\kappa}(\mathtt{S})_j|\vec{\kappa}(\mathtt{R})_j=0) \\
&\quad + \sum_{k\subseteq\mathcal{M}} P(\kappa(\mathtt{R})=k) \sum_{j:k_j=1} H(\vec{\kappa}(\mathtt{S})_j|\vec{\kappa}(\mathtt{R})_j=1) \\
&\overset{(h)}{=} \sum_{k\subseteq\mathcal{M}} P(\kappa(\mathtt{R})=k) \sum_{j:k_j=1} H(\vec{\kappa}(\mathtt{S})_j|\vec{\kappa}(\mathtt{R})_j=1) \\
&\overset{(i)}{=} \sum_{k\subseteq\mathcal{M}} P(\kappa(\mathtt{R})=k)|k|H_{\text{bin}}\left(p_s/p_r\right) \\
&= |\mathcal{M}|p_r H_{\text{bin}}\left(p_s/p_r\right) \\
&\overset{(j)}{=} |\mathcal{M}|\Lambda(p_s, p_r-p_s),
\end{aligned} \tag{35}
$$

where (a) follows from the law of total expectation, (b) follows from the assumption that the image of $f(\cdot,\mathtt{r})$ is prefix-free for any choice of $\mathtt{r}$ as well as the definition of conditional entropy, and (c) follows from the fact that deterministic functions of random quantities cannot increase entropy. To see (d), note that $\hat{\mathtt{s}} = g(f(\mathtt{s},\mathtt{r}),\mathtt{r})$ and by assumption $\mathtt{s}\vdash\hat{\mathtt{s}}$, and for any $\mathtt{q}$ such that $\mathtt{s}\vdash\mathtt{q}$, we have $\hat{\mathtt{s}}\vdash\mathtt{q}$. As a consequence, using the property (6) of a Logic System, we conclude

$$\kappa(\mathtt{s})\subseteq\kappa(\hat{\mathtt{s}})\subseteq\kappa(\mathtt{q}).$$

By choosing $\mathtt{q}=\mathtt{s}$, we obtain the statement that $\kappa(\mathtt{s})=\kappa(\hat{\mathtt{s}})$. Since $\kappa(\mathtt{S})=\kappa(\hat{\mathtt{S}})$, the conditional distribution of either given $\mathtt{R}$ is the same, establishing (d). Step (e) follows from the assumption that $\mathtt{R}\to\kappa(\mathtt{R})\to\kappa(\mathtt{S})$, and then utilizing the result (7). Step (f) follows from the assumption of independence. Step (g) is a simple splitting of the summation in (f). To justify (h), we use the assumption that $\mathtt{S}\vdash\mathtt{R}$ which in particular means that

$$\kappa(\mathtt{S})\subseteq\kappa(\mathtt{R}).$$

Thus necessarily if $\vec{\kappa}(\mathtt{R})_j=0$ then $\vec{\kappa}(\mathtt{S})_j=0$ and the corresponding conditional entropy is zero. To justify (i), we invoke the "identically distributed" property, together with the assumption that $|\mathcal{M}|^{-1}E|\kappa(\mathtt{S})|=p_s$, $|\mathcal{M}|^{-1}E|\kappa(\mathtt{R})|=p_r$, as well as Lemma 5. The final step (j) follows from the definition of Λ. This establishes the lower bound.

The upper bound is proved with a simple variant of the proof of Theorem 2, where $\mathcal{M}$ is substituted with $\kappa(\mathtt{R})$. We write the proof here for completeness. For a given size ξ of a kernel, there are a total of

$$\binom{|\kappa(\mathtt{R})|}{\xi} \leq 2^{|\kappa(\mathtt{R})|H_{\text{bin}}\left(\frac{\xi}{|\kappa(\mathtt{R})|}\right)}$$

possible kernels of the same size, since both sender and receiver share knowledge of $\mathtt{R}$. Let $\text{enum}_{\xi,\psi}:\{k_s\subseteq\mathcal{M}:|k_s|=\xi\}\times\{k_r\subseteq\mathcal{M}:|k_r|=\psi\}\to\{0,1\}^*$ be a function that maps each possible pair (k_s,k_r) of kernels respectively of size ξ and ψ such that $k_s\subseteq k_r$, to a fixed-length binary encoding of the integers $\left\{1,\ldots,\binom{\psi}{\xi}\right\}$, which is an integer that uniquely determines k_s as a subset of k_r. Then, in particular,

$$
\begin{aligned}
\mathbf{len}(\text{enum}_{|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|}(\kappa(\mathtt{S}),\kappa(\mathtt{R}))) &\leq |\kappa(\mathtt{R})|H_{\text{bin}}\left(\frac{|\kappa(\mathtt{S})|}{|\kappa(\mathtt{R})|}\right)+1 \\
&= \Lambda(|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|-|\kappa(\mathtt{S})|)+1.
\end{aligned}
$$

The last step follows from the definition of Λ; note that the resulting expression is a random variable. For given sentences $\mathtt{s},\mathtt{r}\in\mathcal{L}$, we define the encoder f as a concatenation of two separate encodings:

$$f(\mathtt{s},\mathtt{r})=\text{elias}_\delta(|\kappa(\mathtt{s})|)\text{enum}_{|\kappa(\mathtt{s})|,|\kappa(\mathtt{R})|}(\kappa(\mathtt{s}),\kappa(\mathtt{r})). \tag{36}$$

Since both of the codes implied by each encoding are prefix-free, the concatenation is also prefix-free. Finally, we let g be the decoder that recovers the kernel $\kappa(\mathtt{s})$ from the output of f, and evaluates ℓ on that kernel.

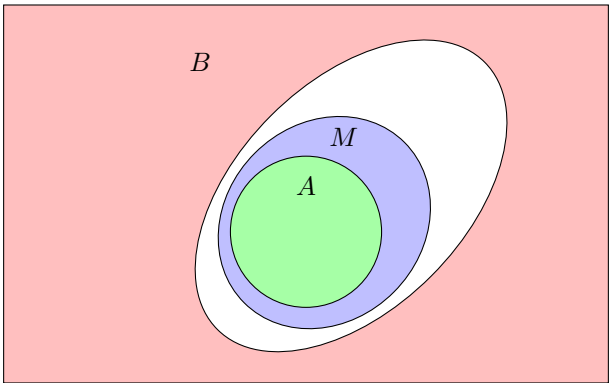


Figure 15: An example of a partition $\{M, M^c\}$ that separates A from B.

We note that, by construction, if $\mathtt{q}$ is such that $\mathtt{s} \vdash \mathtt{q}$, then $g(f(\mathtt{s},\mathtt{r}),\mathtt{r}) \vdash \mathtt{q}$ since $\kappa(\mathtt{s}) = \kappa(g(f(\mathtt{s},\mathtt{r}),\mathtt{r}))$, and thus we have met the conditions of the Theorem. The estimate for the overall cost of the encoding can be done by separately estimating the length of the two encodings in (36), namely

$$\Lambda(|\kappa(\mathtt{S})|, |\kappa(\mathtt{R})| - |\kappa(\mathtt{S})|) + \log_2 |\kappa(\mathtt{S})| + 2\log_2 (\log_2 |\kappa(\mathtt{S})|) + 4.$$

Taking the expectation with respect to $\mathtt{S}, \mathtt{R}$, using the concavity $\cap$ of Λ (see Lemma 2) as well as of the logarithm, and normalizing by $|\mathcal{M}|$, we obtain an upper estimate of

$$\Lambda(p_s, p_r - p_s) + \frac{\log_2 (p_s|\mathcal{M}|)}{|\mathcal{M}|} + 2\frac{\log_2 (\log_2 (p_s|\mathcal{M}|))}{|\mathcal{M}|} + \frac{4}{|\mathcal{M}|}.$$

As before, the upper bounds holds under the more general assumption that $\mathtt{S}, \mathtt{R}$ have kernels that follow a (p_s, p_r)-law. This completes the proof of the Theorem.

5.3 The partition compression problem

In this subsection, we treat a lossy data compression problem of central relevance to the "targeted query" settings of Theorems 5, 7 and 8. We refer the reader to Figure 15. Imagine one has n items as well as two non-intersecting subsets of those n items, which we call A and B, and one is interested in efficiently sending to a receiver a subset M which contains A but excludes B; or, alternatively stated, a partition that separates A from B. One solution is to send $M = A$ or $M = B$, whichever is cheapest to send, but it turns out that there is generally a better solution. Let $X^n \in \{0,1,2\}^n$ be a vector with $X_i = 1$ if the ith element is inside of A, $X_i = 0$ if the ith element is inside of B, and $X_i = 2$ if the ith element is neither in A nor in B. Assume that the entries of X^n meet the following conditions:

$$\begin{aligned} n^{-1}E|\{i : X_i = 0\}| &= p_b, \\ n^{-1}E|\{i : X_i = 1\}| &= p_a, \\ n^{-1}E|\{i : X_i = 2\}| &= 1 - p_a - p_b. \end{aligned} \tag{37}$$

Define a distortion metric to be a function

$$\rho : \{0,1,2\} \times \{0,1\} \to \{0,1\} \tag{38}$$

using this matrix

	0	1
0	0	1
1	1	0
2	0	0

. (39)

An encoder and decoder are functions $f_n : \{0,1,2\}^n \to \{0,1\}^*$ and $g_n : \{0,1\}^* \to \{0,1\}^n$, respectively. The problem is to find a good upper bound for

$$\min_{f_n, g_n} n^{-1} E_{X^n} \left[\mathbf{len}(f_n(X^n)) \right]$$

subject to the condition that, for any $x_1^n \in \{0,1,2\}^n$,

$$\sum_{i=0}^{n-1} \rho(x_i, g_n(f_n(x_1^n))_i) = 0. \tag{40}$$

The solution to this problem can be obtained by an application of Shannon's Rate-Distortion theory.

Theorem 4 (Shannon bounds for partition compression) *For any $n \geq 1$, let $X_1^n \in \{0,1,2\}^n$ be a random vector with the property (37). Let f_n, g_n be encoder and decoder functions as defined earlier. Then*

$$\min_{f_n, g_n} n^{-1} E_X \left[\mathbf{len}(f_n(X_1^n)) \right] \leq \Lambda(p_a, p_b) + 2 \frac{\log_2 \left(n \Lambda(p_a, p_b) \right)}{n} + \frac{3}{n},$$

where the minimization is over f_n, g_n, such that the image of f_n is prefix-free and that condition (40) is met. Furthermore, if the random variables $\{X_i\}$ are additionally i.i.d., we have

$$\Lambda(p_a, p_b) \leq \min_{f_n, g_n} n^{-1} E_X \left[\mathbf{len}(f_n(X_1^n)) \right].$$

Remark: From Lemma 2, if $p_a + p_b < 1$, then we can deduce that $\Lambda(p_a, p_b) < \min\{H_{\text{bin}}(p_a), H_{\text{bin}}(p_b)\}$, and thus this result predicts the existence of partition compression techniques which are more efficient than the "naive" solution of sending the cheapest of the sets (A or B).

Proof. We provide a proof of the upper bound that is simple and includes a useful second order upper bound which will play a role in the proofs of Theorems 5, 7. As we will show, the expected performance of a random matrix with the density of ones appropriately tuned will asymptotically approach the Shannon limit $\Lambda(p_a, p_b)$. This implies, as per the classic random coding argument of Shannon, the existence of a deterministic code with performance at least as good as the expected performance of the random code.

Let $X_1, \ldots, X_n \in \{0,1,2\}^n$ be the input random vector, and let $\Psi \subseteq \{1, \ldots, n\}$ denote the random positions where X_i is taking on 0 or 1, thus where we want to enforce a bit pattern. For any arbitrary $z \in \{0,1\}^n$, let $[z]_\Psi$ denote the $1 \times |\Psi|$ vector obtained by extracting from z the columns indexed by Ψ.

Let C be a random binary matrix with an infinite number of rows $\{C_1, C_2, \ldots\}$ and each with n columns. Assume its entries are chosen i.i.d. according to the distribution

$$P(C_{i,j} = 0) = \frac{p_a}{p_a + p_b}.$$

The sender scans the matrix C from top to bottom until it finds the first row J that satisfies the following condition:

$$[C_J]_\Psi = X_\Psi, \tag{41}$$

and sends the index J of that row. The receiver then recovers the row from the index. Let $N_0 = \{i : X_i = 0\}$ and $N_1 = \{i : X_i = 1\}$. The probability of a row of C satisfying (41), conditional on Ψ, is given by

$$P([C_i]_\Psi = X_\Psi | \Psi) = \left(\frac{p_a}{p_a + p_b} \right)^{N_0} \left(\frac{p_b}{p_a + p_b} \right)^{N_1}.$$

Note that independence of the entries in the vector X was not necessary to assert this, and instead, the way C is constructed is sufficient. It is now easy to calculate

$$P(J = j) = (1 - P([C_i]_\Psi = X_\Psi | \Psi))^{j-1} P([C_i]_\Psi = X_\Psi | \Psi)$$

since J is the *first* occurrence of the pattern Ψ. As a consequence,

$$E\left[J|\Psi\right] = 1/P([C_i]_\Psi = X_\Psi|\Psi) = 2^{\left(-N_0 \log_2 \frac{p_a}{p_a+p_b} - N_1 \log_2 \frac{p_b}{p_a+p_b}\right)}.$$

To send the index J we will be using δ Elias coding. Note that $E_\Psi\left[\log_2\left(E\left[J|\Psi\right]\right)\right] = n\Lambda(p_a, p_b)$. We can then upper bound the performance of the code as

$$\begin{aligned} E_J\left[\mathbf{len}(\text{elias}_\delta(J))\right] &= E_\Psi\left[E_J\left[\mathbf{len}(\text{elias}_\delta(J))|\Psi\right]\right] \\ &\leq E_\Psi\left[E_J\left[\log_2 J + 2\log_2(\log_2 J) + 3\ |\Psi\right]\right] \\ &\leq E_\Psi\left[\log_2 E_J\left[J|\Psi\right] + 2\log_2(\log_2 E_J\left[J|\Psi\right]) + 3\right] \\ &\leq E_\Psi\left[\log_2 E_J\left[J|\Psi\right]\right] + 2\log_2(E_\Psi\left[\log_2 E_J\left[J|\Psi\right]\right) + 3] \\ &\leq n\Lambda(p_a, p_b) + 2\log_2(n\Lambda(p_a, p_b)) + 3, \end{aligned}$$

where in addition to the estimate of the performance of δ Elias coding, we used the concavity $\cap$ of the logarithm. The lower bound, which is only valid in the case X_1^n is an i.i.d. random vector, is a straightforward consequence of existing $R(D)$ literature [12] so it is omitted.

5.4 Less is more

In this pattern the purpose of Alice is not to communicate to Bob enough to prove anything she can prove; rather it is to communicate to him the minimal amount of information needed to prove a *specific sentence* $\mathtt{q}$ (and of course, having achieved that, any other sentence that is entailed by $\mathtt{q}$).

We refer the reader to the top of Figure 16. In here we illustrate $\kappa(\mathtt{s})$ in green, and $\kappa(\mathtt{q})$ in white. One fairly obvious strategy to solve this problem would be to send to Bob enough information to reconstruct $\kappa(\mathtt{q})$; then Bob will be able to prove anything that can be proved starting from $\mathtt{q}$. Alternately, one could send to Bob enough information to reconstruct $\kappa(\mathtt{s})$ itself, which would allow Bob to prove potentially even more logic sentences. For the purposes of quantifying the cost of either of these two strategies, let us assume that $\mathtt{S}, \mathtt{Q}$ have kernels that follow a (p_s, p_q)-law (with $p_s < p_q$), then using the ideas behind Theorem 2, and choosing the best from either of these two strategies, we would be spending a normalized average total of

$$\min\{H_{\text{bin}}\left(p_s\right), H_{\text{bin}}\left(p_q\right)\} \tag{42}$$

bits (neglecting asymptotically vanishing terms). Yet, as we will soon show, this strategy is in general *suboptimal*. This was a surprising revelation to us, so we want to equip the reader with the insight used to prove this result. From the top of Figure 16, it should be apparent that not only could Alice use either of the two strategies above to communicate to Bob; in fact she has the freedom to send any possible $\kappa(\hat{\mathtt{s}})$ that satisfies $\kappa(\mathtt{s}) \subseteq \kappa(\hat{\mathtt{s}}) \subseteq \kappa(\mathtt{q})$; one such example $\kappa(\hat{\mathtt{s}})$ is illustrated in blue in the Figure. However, crucially, note that a set $\kappa(\hat{\mathtt{s}})$ that is bigger than $\kappa(\mathtt{s})$, even though it may feel more complex to describe, can actually be good for many choices for $\mathtt{s}$ and $\mathtt{q}$, *and thus we may not need that big a pre-agreed collection of those.* This insight means we can beat the estimate (42). In the same figure, we show an architecture with a special form of encoding which was introduced in the proof of Theorem 4.

Theorem 5 (Goal is a targeted query) *Let $(L, \mathcal{M}, \kappa, \ell)$ be a Logic System. Let $\mathtt{S}, \mathtt{Q} \in \mathcal{L}$ represent the sender's and query logic sentences, respectively, with the property that $\mathtt{S} \vdash \mathtt{Q}$ and in particular, $\mathtt{S}, \mathtt{Q}$ have kernels that follow a (p_s, p_q)-law. Let the encoder f and decoder g be functions*

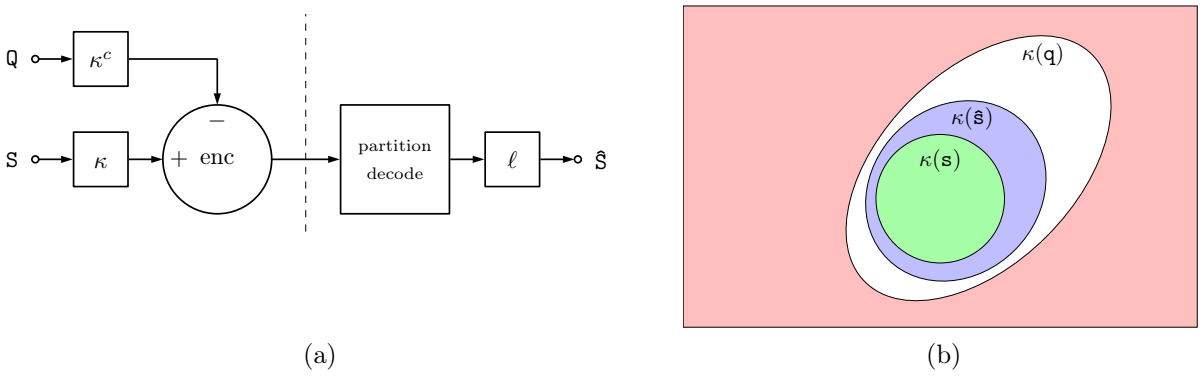
$$\begin{aligned} f : \mathcal{L}^2 &\to \{0,1\}^*, \\ g : \{0,1\}^* &\to \mathcal{L}. \end{aligned}$$

Then

$$\min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{Q}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q}))\right] \leq \Lambda(p_s, 1-p_q) + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right),$$

where the minimization is over f, g such that the image of f is prefix-free and such that, if $\mathtt{s} \vdash \mathtt{q}$, then $\mathtt{s} \vdash g(f(\mathtt{s},\mathtt{q}))$ and $g(f(\mathtt{s},\mathtt{q})) \vdash \mathtt{q}$. If additionally, $\{(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}))_j\}_{j=1}^{|\mathcal{M}|}$ are i.i.d., then

$$\Lambda(p_s, 1-p_q) \leq \min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{Q}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q}))\right]$$



(a) (b)

Figure 16: Proof strategy for Theorem 5

Proof. As with Theorem 2, we start by invoking Kraft's inequality through Lemma 4 and write

$$E_{\mathtt{S},\mathtt{Q}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q}))\right] \geq H(f(\mathtt{S},\mathtt{Q})) \geq H(g(f(\mathtt{S},\mathtt{Q}))).$$

The direction of the proof now diverges with respect to that of Theorem 2, by deriving

$$\begin{aligned} H(g(f(\mathtt{S},\mathtt{Q}))) &\overset{(a)}{=} H(g(f(\mathtt{S},\mathtt{Q}))) - H(g(f(\mathtt{S},\mathtt{Q}))|\mathtt{S},\mathtt{Q}) \\ &\overset{(b)}{=} I(\mathtt{S},\mathtt{Q}; g(f(\mathtt{S},\mathtt{Q}))) \\ &\overset{(c)}{\geq} I(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q}); \vec{\kappa}(g(f(\mathtt{S},\mathtt{Q})))) \end{aligned}$$

where (a) follows from the fact that discrete entropy is zero when conditioning on all randomness, (b) is from the definition of mutual information, and (c) follows from the data processing inequality.

Referencing Figure 16, we now construct a function ψ that accepts $\vec{\kappa}(\mathtt{S})$ and $\vec{\kappa}(\mathtt{Q})$ and produces a vector of length $|\mathcal{M}|$ with entries in $\{0,1,2\}$ indicating whether each entry is in the red (0), green (1) or white (2) regions. Let $\mathtt{v},\mathtt{w} \in \{0,1\}^{|\mathcal{M}|}$. The construction is as follows:

$$\psi(\mathtt{v},\mathtt{w}) = 0 \times (\underline{1} - \mathtt{w}) + 1 \times \mathtt{v} + 2 \times (\underline{1} - \mathtt{v})\mathtt{w}, \tag{43}$$

where the $\times$ operator is multiplying a scalar times a vector element-wise, and $\underline{1}$ represents a vector of all ones, and the product of vectors in the last term is element-wise. Of course, the leftmost term is always zero, but we included it to ensure that the connection to Figure 16 is clear. Next define

$$\begin{aligned} X_1^{|\mathcal{M}|} &= \psi(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})), \\ Z_1^{|\mathcal{M}|} &= 1 \times \vec{\kappa}(g(f(\mathtt{S},\mathtt{Q}))). \end{aligned}$$

The relationship between $X_1^{|\mathcal{M}|}$ and $Z_1^{|\mathcal{M}|}$ is in general very complex, as we have few assumptions on f and g. However, some key assertions can be made. First, recall that $\mathtt{S},\mathtt{Q}$ have kernels that follow a (p_s,p_q)-law; see Definition 3, and in particular recall that $\mathtt{S} \vdash \mathtt{Q}$. Using elementary probability,

$$\begin{aligned} P(X_i = 0) &= P([\vec{\kappa}(\mathtt{S})_i = 0] \cap [\vec{\kappa}(\mathtt{Q})_i = 0]) \\ &= P([\vec{\kappa}(\mathtt{S})_i = 0]) + P([\vec{\kappa}(\mathtt{Q})_i = 0]) - P([\vec{\kappa}(\mathtt{S})_i = 0] \cup [\vec{\kappa}(\mathtt{Q})_i = 0]) \\ &= P([\vec{\kappa}(\mathtt{Q})_i = 0]) \end{aligned}$$

where the last equality uses (6) from the fundamental Definition 1, to conclude that $\vec{\kappa}(\mathtt{S}) \leq \vec{\kappa}(\mathtt{Q})$ where the inequality is to be interpreted element-wise. Now using the "identically distributed" assumption from the (and not yet independence), we obtain

$$P(X_i = 0) = 1 - p_q$$

With a far simpler argument, also without using independence and only the identical distributed assumption, we obtain

$$P(X_i = 1) = P([\vec{\kappa}(\mathtt{S})_i = 1]) = p_s$$

Finally we add the independence assumption, thus concluding that the $\{X_i\}$ are i.i.d. according to the distribution

$$\begin{aligned} P(X_i = 0) &= 1 - p_q, \\ P(X_i = 1) &= p_s, \\ P(X_i = 2) &= p_q - p_s. \end{aligned}$$

To continue, the assumption that $\mathtt{s} \vdash g(f(\mathtt{s}, \mathtt{q}))$, $g(f(\mathtt{s}, \mathtt{q})) \vdash \mathtt{q}$ can be used to establish a useful relation between X_i and Z_i. Recall the definition of the distortion metric ρ from Equations (38) and (39). Then, the assumption implies that, for all $1 \leq i \leq |\mathcal{M}|$,

$$\rho(X_i, Z_i) = 0. \tag{44}$$

This is an important fact that will be used shortly. We now apply the data processing inequality once more, taking advantage of the definitions for X_i and Z_i, and continue the proof with a pattern commonly found in Rate-Distortion theory:

$$\begin{aligned} I(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}); \vec{\kappa}(g(f(\mathtt{S}, \mathtt{Q})))) &\overset{(d)}{\geq} I(X_1, \ldots, X_{|\mathcal{M}|}; Z_1, \ldots, Z_{|\mathcal{M}|}) \\ &= H(X_1, \ldots, X_{|\mathcal{M}|}) - H(X_1, \ldots, X_{|\mathcal{M}|} | Z_1, \ldots, Z_{|\mathcal{M}|}) \\ &\overset{(e)}{=} \sum_{i=1}^{|\mathcal{M}|} H(X_i) - H(X_i | Z_1, \ldots, Z_{|\mathcal{M}|}, X_1, \ldots X_{i-1}) \\ &\overset{(f)}{\geq} \sum_{i=1}^{|\mathcal{M}|} H(X_i) - H(X_i | Z_i) \\ &= \sum_{i=1}^{|\mathcal{M}|} I(X_i; Z_i) \end{aligned} \tag{45}$$

where (d) follows from the data processing inequality, (e) follows from the fact that the $\{X_i\}$ are independent and from the chain rule for entropy, and (f) follows from the fact that conditioning cannot increase entropy.

We now pause to observe that for any given random variables V, W, the mutual information $I(V; W)$ is an expression that can be entirely computed from p_V and $Q_{W|V}$ as these two completely determine the distribution of V, W. Thus we could write $I(V, W) = \iota(p_V, Q_{W|V})$ for some function ι and in particular the step (45) can be rewritten as

$$\sum_{i=1}^{|\mathcal{M}|} \iota(p_{X_i}, Q_{Z_i|X_i}) = \sum_{i=1}^{|\mathcal{M}|} \iota(p_{X_1}, Q_{Z_i|X_i}) \tag{46}$$

where this last equality follows from the fact that the marginal for X_i is identical for all i, but the conditionals $Q_{Z_i|X_i}$ are in general different. It is known that the function $\iota(p_V, Q_{W|V})$ is convex $\cup$ on $Q_{W|V}$ and therefore by Jensen's inequality, the expression (46) can be lower bounded by

$$\iota\left(p_{X_1}, \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} Q_{Z_i|X_i}(z|x)\right).$$

Let X', Z' be distributed according to the marginal for X_1 and the averaging of conditional distribution above. Because of (44), it is the case that

$$E_{X',Z'} \rho(X', Z') = 0.$$

Therefore the following bound holds

$$\sum_{i=1}^{|\mathcal{M}|} I(X_i; Z_i) \geq |\mathcal{M}| I(X'; Z') \geq |\mathcal{M}| \min_{P_{X,Z} \in \mathcal{D}} I(X; Z),$$

where the domain $\mathcal{D}$ for the minimization is defined by joint distributions for X, Z with $X \sim (1-p_q, p_s, p_q-p_s)$ and $E[\rho(X, Z)] = 0$. The fact that such a minimization results in $\Lambda(p_s, 1-p_q)$ can be checked using standard variational methods. This concludes the proof of the lower bound.

To prove the upper bound, we construct a code as follows. Define

$$X_1^{|\mathcal{M}|} = \psi(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q})).$$

We note that by construction, under the weaker assumption that $\mathtt{S}, \mathtt{Q}$ have kernels that follow a (p_s, p_q)-law.

$$\begin{aligned} n^{-1} \sum_{i=0}^{n-1} P(X_i = 0) &= 1 - p_q, \\ n^{-1} \sum_{i=0}^{n-1} P(X_i = 1) &= p_s, \\ n^{-1} \sum_{i=0}^{n-1} P(X_i = 2) &= p_q - p_s. \end{aligned}$$

In reference to (37), we next invoke the upper bound of Theorem 4, which guarantees the existence of $\hat{f}$, $\hat{g}$ such that

$$\begin{aligned} |\mathcal{M}|^{-1} E_{X_1^{|\mathcal{M}|}} \left[\mathbf{len}(\hat{f}(X_1^{|\mathcal{M}|})) \right] &\leq \Lambda(p_s, 1-p_q) + 2 \frac{\log_2 \left(|\mathcal{M}| \Lambda(p_s, 1-p_q) \right)}{|\mathcal{M}|} + \frac{3}{|\mathcal{M}|}, \\ \sum_{i=1}^{|\mathcal{M}|} \rho(X_i, \hat{g}(\hat{f}(X_1^{|\mathcal{M}|}))_i) &= 0. \end{aligned}$$

We define our encoder then as

$$f(\mathtt{s}, \mathtt{q}) = \hat{f}(\psi(\vec{\kappa}(\mathtt{s}), \vec{\kappa}(\mathtt{q}))).$$

Recall that $\hat{g}(\text{codeword}) \in \{0,1\}^{|\mathcal{M}|}$ and thus it can be regarded also as a kernel via the dual notation where kernels are depicted by the indicator vector of the corresponding subset of $\mathcal{M}$. To conclude, we define the decoder g as

$$g(\text{codeword}) = \vec{\ell}(\hat{g}(\text{codeword})),$$

where codeword $\in \{0,1\}^*$ is meant to be precisely $f(\mathtt{s}, \mathtt{q})$ when the encoder and decoder are being used simultaneously. We conclude the proof by noting that, by construction, if $\mathtt{s} \vdash \mathtt{q}$, we have $\mathtt{s} \vdash g(f(\mathtt{s}, \mathtt{q}))$, $g(f(\mathtt{s}, \mathtt{q})) \vdash \mathtt{q}$ and furthermore

$$|\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{Q}} \left[\mathbf{len}(f(\mathtt{S}, \mathtt{Q})) \right] \leq \Lambda(p_s, 1-p_q) + 2 \frac{\log_2 \left(|\mathcal{M}| \Lambda(p_s, 1-p_q) \right)}{|\mathcal{M}|} + \frac{3}{|\mathcal{M}|}.$$

□

5.5 No need to know

In this problem, Bob's logic sentence $\mathtt{R}$ is unavailable to Alice, and the goal of the communication is for Bob to be able to prove anything that Alice can. To address the problem, we will need to introduce a more complex communication pattern, where Alice and Bob are able to exchange messages over several rounds,

Alice's private context	common context	Alice's function	direction	Bob's function	common context	Bob's private context
$\mathtt{s}$		g^0	$\xleftarrow{b^0}$	f^0		$\mathtt{r}$
$\mathtt{s}$	b^0	f^1	$\xrightarrow{a^1}$	g^1	b^0	$\mathtt{r}$
$\mathtt{s}$	b^0, a^1	g^2	$\xleftarrow{b^2}$	f^2	b^0, a^1	$\mathtt{r}$
$\mathtt{s}$	b^0, a^1, b^2	f^3	$\xrightarrow{a^3}$	g^3 $\downarrow$ $\hat{\mathtt{s}}$ ($\forall \mathtt{q}$ s.t. $\mathtt{s} \vdash \mathtt{q}$, $\hat{\mathtt{s}} \vdash \mathtt{q}$)	b^0, a^1, b^2	$\mathtt{r}$

Table 1: A 4-turn code for full synchronization. It is assumed that $\mathtt{s} \vdash \mathtt{r}$. The actual step of Bob proving queries is not shown.

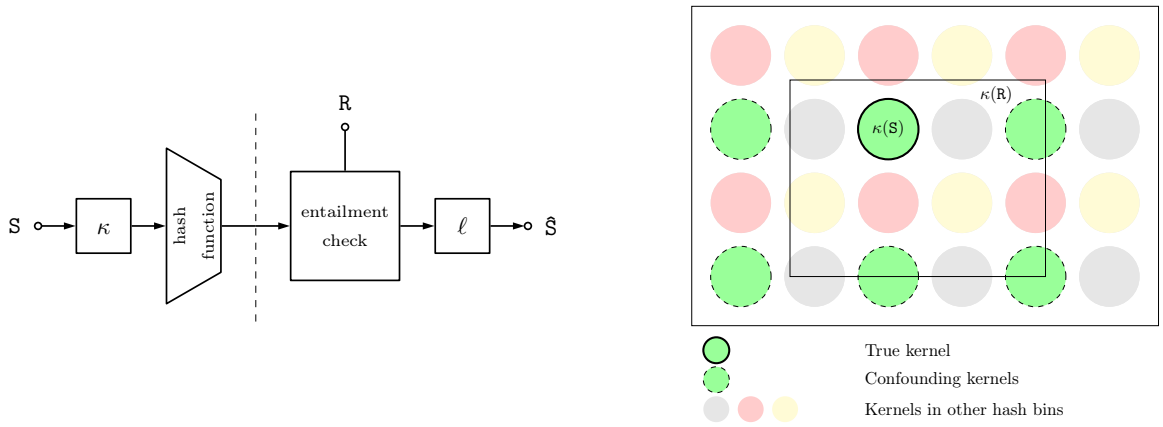
taking turns on who is sending and who is receiving. As before, we will use f to denote an encoder and g to denote a decoder, but the agent doing the encoding and decoding will change depending on the turn.

Table 1 is an exemplary representation of the conversational paradigm that we are now entering. In this table, the context to which either Alice or Bob are privy is shown, split into whether the context is either common or private. The context informs what the encoder or decoder functions are allowed to depend on; in addition, every decoder is allowed to depend on the output of the corresponding encoder, which is denoted by the letter on the top of each arrow indicating the direction of the communication. For example, the f^3 encoder, whose output is a^3, may depend on $\mathtt{s}, b^0, a^1$ and b^2. In all cases, the image for each of the f^i encoders is a subset of $\{0,1\}^*$. The communication task finishes when g^3 computes its output $\hat{\mathtt{s}}_{\mathtt{m}} \in \mathcal{L}$ which can now be used to prove any $\mathtt{q}$ with the property that $\mathtt{s} \vdash \mathtt{q}$. To measure the efficiency of communication, we define the total normalized average cost:

$$\frac{1}{|\mathcal{M}|} E_{\mathtt{S},\mathtt{R}} \left[\mathbf{len}(B^0) + \mathbf{len}(A^1) + \mathbf{len}(B^2) + \mathbf{len}(A^3) \right]$$

where B^0, A^1, B^2, A^3 are random versions of b^0, a^1, b^2, a^3 when in the interaction the sender experiences $\mathtt{S}$ and the receiver experiences $\mathtt{R}$. The reader may recall that in previous results, we assumed that the image of an encoder was a prefix-free code. In a similar manner, we assume that the image of encoder f^0 is prefix-free. In turn 1, both parties have b^0 as common context, and therefore the assumption that we have is that the image of f^1, when the value of b^0 is kept fixed, is prefix-free. Similarly, we assume that the image of f^2, when the value of b^0, a^1 are kept fixed, is prefix-free and finally, we assume that the image of f^3 when b^0, a^1, b^2 are kept fixed is prefix-free. We call any code with the properties illustrated by Table 1 and this discussion a *4-turn code* for full synchronization.

Before presenting the result we introduce the main intuition behind how the upper bound of this result is proved. Alice and Bob meet ahead of time, and they agree on a family of hash functions that can be used to map sets $\kappa(\mathtt{S})$ of various cardinalities to a bin in a many-to-one fashion. After the receiver is presented with $\mathtt{R}$ and the sender is presented with $\mathtt{S}$, in Turn 0, the receiver informs the sender of the cardinality of $\mathtt{R}$ set, and as part of Turn 1, the sender does the same but for $\mathtt{S}$. As a result of this calibration, also as part of Turn 1, the sender sends the index of a hash bin where $\kappa(\mathtt{S})$ has been mapped. Aided by $\mathtt{R}$, the receiver is able to recover $\kappa(\mathtt{S})$ from the hash bin the majority of time. Turns 2 and 3 are there to address the possible exception where the receiver fails to recover $\kappa(\mathtt{S})$ in Turn 1. The communication costs in Turns 0, 2 and 3 are asymptotically negligible and thus Turn 1 dominates the total communication cost. The main task is to demonstrate that asymptotically, the normalized logarithm of the number of such bins approaches $\Lambda(p_s, p_r - p_s)$ and to demonstrate that all other communication cost is negligible.



(a) (b)

Figure 17: Illustration of the general strategy for the upper bound in the proof of Theorem 6, based on sending the index of a hash bin containing the kernel of the sender's sentence, and then resolving ambiguity by testing whether R is entailed.

Theorem 6 (Alice doesn't know what Bob knows, a.k.a. No Need to Know) *Let $(L, \mathcal{M}, \kappa, \ell)$ be a Logic System. Let $\mathtt{S}, \mathtt{R} \in \mathcal{L}$ represent the sender's logic sentence and the receiver's logic sentence, assuming $\mathtt{S}, \mathtt{R}$ have kernels that follow a (p_s, p_r)-law (and thus $\mathtt{S} \vdash \mathtt{R}$). Then*

$$\begin{aligned} &\min_{\{f^i, g^i\}_{i=0}^3} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{R}} \left[\mathbf{len}(B^0) + \mathbf{len}(A^1) + \mathbf{len}(B^2) + \mathbf{len}(A^3) \right] \\ &\leq \quad \Lambda(p_s, p_r - p_s) + O\left(\frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|} \right). \end{aligned}$$

where the minimization is over all 4-turn codes for full synchronization, as defined by Table 1 and the corresponding explanation of it. Furthermore, the if $\{(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{R}))_i\}_{i=1}^{|\mathcal{M}|}$ are i.i.d., and furthermore, $\mathtt{R} \to \kappa(\mathtt{R}) \to \kappa(\mathtt{S})$, then

$$\Lambda(p_s, p_r - p_s) \quad \leq \quad \min_{\{f^i, g^i\}_{i=0}^3} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{R}} \left[\mathbf{len}(B^0) + \mathbf{len}(A^1) + \mathbf{len}(B^2) + \mathbf{len}(A^3) \right].$$

Proof. The lower bound is proved as follows. We start from from a general 4-turn code for full synchronization $\{f^i, g^i\}_{i=0}^3$, and let $B^0, A^1, B^2, A^3 \in \{0,1\}^*$ be the binary strings output by the encoders f^0, f^1, f^2, f^3 that are used by Bob and Alice, when the interaction is operating over the random quantities $\mathtt{S}$ and $\mathtt{R}$. The expected communication cost is then lower bounded as follows:

$$\begin{aligned} E\left[\mathbf{len}(B^0) + \mathbf{len}(A^1) + \mathbf{len}(B^2) + \mathbf{len}(A^3)\right] &\overset{(a)}{=} E\left[\mathbf{len}(B^0)\right] + E\left[E\left[\mathbf{len}(A^1)|B^0\right]\right] \\ &+ E\left[E\left[\mathbf{len}(B^2)|B^0, A^1\right]\right] + E\left[E\left[\mathbf{len}(A^3)|B^0, A^1, B^2\right]\right] \\ &\overset{(b)}{\geq} H(B^0) + H(A^1|B^0) + H(B^2|B^0, A^1) + H(A^3|B^0, A^1, B^2) \\ &= H(B^0, A^1, B^2, A^3) \\ &\geq H(B^0, A^1, B^2, A^3|\mathtt{R}) \\ &\geq H(\hat{\mathtt{S}}|\mathtt{R}). \end{aligned} \tag{47}$$

Step (a) is a straightforward application of the law of total expectation. Step (b) follows from the application of Lemma 4 and the prefix-free assumption in the theorem. The argument continues as follows:

$$H(\hat{\mathtt{S}}|\mathtt{R}) \quad \geq \quad H(\kappa(\hat{\mathtt{S}})|\mathtt{R})$$

$$\begin{aligned} &\overset{(c)}{=} && H(\kappa(\mathtt{S})|\mathtt{R}) \\ &\overset{(d)}{\geq} && |\mathcal{M}|\Lambda(p_s, p_r - p_s). \end{aligned}$$

Step (c) is justified using arguments similar to those of Subsection 5.2 substituting $\hat{\mathtt{s}} = g^3(\mathtt{r}, b^0, a^1, b^2, a^3)$ instead. Step (d) is justified by exactly the same derivation found in (35), starting from step (e). This establishes the lower bound.

We now turn our attention to the upper bound. First we establish the precise domain/image of the encoder and decoders for our 4-turn mode:

$$\begin{aligned} f^0 : \mathcal{L} &\rightarrow \{0,1\}^*, \quad g^0 : \{0,1\}^* \rightarrow \mathbb{Z}, \\ f^1 : \mathcal{L} \times \mathbb{Z} &\rightarrow \{0,1\}^*, \quad g^1 : \{0,1\}^* \times \mathcal{L} \rightarrow \mathcal{L} \times \{\text{success}, \text{failure}\}, \\ f^2 : \{\text{success}, \text{failure}\} &\rightarrow \{0,1\}, \quad g^2 : \{0,1\} \rightarrow \{\text{success}, \text{failure}\}, \\ f^3 : \mathcal{L} &\rightarrow \{0,1\}^*, \quad g^3 : \{0,1\}^* \times \{\text{success}, \text{failure}\} \times \mathcal{L} \rightarrow \mathcal{L}. \end{aligned}$$

Alice and Bob first interact to establish a family of hash functions. How this is done will be described shortly. The transmission protocol is as follows:

1. The receiver sends $|\kappa(\mathtt{R})|$ to the sender using δ Elias coding, which is decoded at the sender side. This step defines f^0 and g^0.

2. The sender sends $|\kappa(\mathtt{S})|$ to the receiver using δ Elias coding, which is decoded at the receiver side. This only partly defines f^1 and g^1 .

3. At this point, the receiver and sender independently can compute

$$\text{RATE} = \Lambda\left(\frac{|\kappa(\mathtt{S})|}{|\mathcal{M}|}, \frac{|\kappa(\mathtt{R})|}{|\mathcal{M}|} - \frac{|\kappa(\mathtt{S})|}{|\mathcal{M}|}\right) + \frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|}.$$

   The same hash function is then selected by both parties independently, which matches the selected rate and depends on $\mathcal{M}$, $|\kappa(\mathtt{S})|$ and $|\kappa(\mathtt{R})|$.

4. The sender sends the bin index $\text{BIN}_{|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|}(\kappa(\mathtt{S}))$ to the receiver using $2^{|\mathcal{M}|\text{RATE}}$ bits. This step completes the definition of f^1.

5. The receiver attempts to decode $\kappa(\mathtt{S})$ using the bin index, its knowledge of $\kappa(\mathtt{R})$ and the algorithm described below. By leveraging the function ℓ, the outcome of this attempt results in an element of $\mathcal{L}$ and a "success" or a default, dummy element from $\mathcal{L}$ and a "failure". This completes the definition of g^1.

6. The success/failure of the attempt is signaled back to the sender using a single bit. This defines both f^2 and g^2.

7. If successful, the sender has nothing to do anymore, as the receiver expects no further communication. The receiver simply outputs the element from $\mathcal{L}$ computed by g^1, designating it as the output $\hat{\mathtt{S}}$, partly defining g^3.

8. If unsuccessful, the sender sends $\kappa(\mathtt{S})$ as a binary vector of length $|\mathcal{M}|$, which is decoded by the receiver and by passing it through ℓ, becomes $\hat{\mathtt{S}}$, the designated output of the protocol, completing the definition of f^3 and g^3.

The reader may appreciate that the algorithm guarantees that the receiver at the end will have in possession enough information to reproduce a message $\hat{\mathtt{S}}$ that is functionally equivalent to $\mathtt{S}$. The problem that remains is demonstrating that the overall communication efficiency of this protocol is such that the normalized total bidirectional cost in bits is asymptotically $\Lambda(p_s, p_r - p_s)$.

Prior to stating the protocol we disclosed that Alice and Bob met before to agree on a family of hash functions. We describe next how this agreement is arrived to. For each possible combination of potential sender and receiver kernel sizes $0 \leq s \leq r \leq |\mathcal{M}|$ with corresponding coding rate $\text{rate}(s,r) \triangleq$

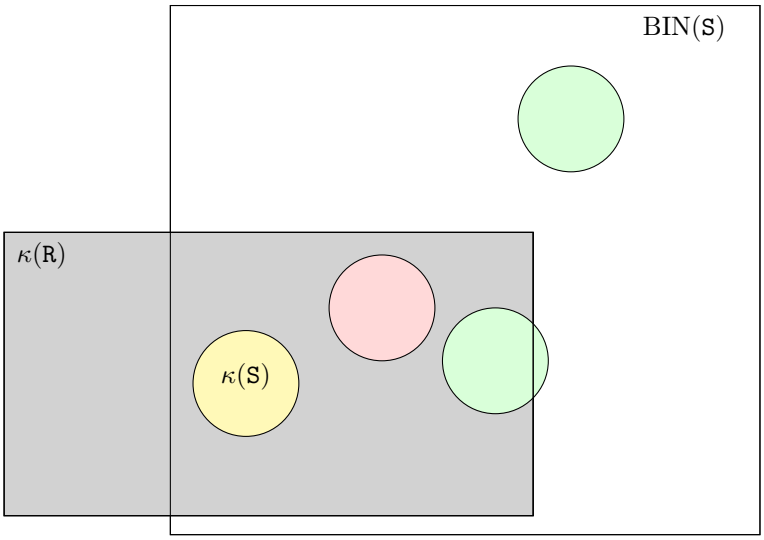


Figure 18: An example of a hash bin collision in Theorem 6, where the receiver is unable to decide between the true kernel in yellow and a confounding one in red.

$\Lambda\left(\frac{s}{|\mathcal{M}|}, \frac{r}{|\mathcal{M}|} - \frac{s}{|\mathcal{M}|}\right) + \frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|}$, we will build a random hash function for $\{\mathtt{s} \in \mathcal{L} : |\kappa(\mathtt{s})| = s\}$. For each element of this set we choose a bin index independently and uniformly at random from the set $\{1, \ldots, \lceil 2^{|\mathcal{M}| \cdot \mathrm{rate}(s,r)} \rceil\}$; call the resulting function $\mathrm{BIN}_{s,r} : \{0,1\}^{|\mathcal{M}|} \to \{1, \ldots, \lceil 2^{|\mathcal{M}| \cdot \mathrm{rate}(s,r)} \rceil\}$. We use the upper case notation BIN to remind the reader that this is a random hash function. The expected, normalized cost for the transmissions associated with steps 1,2,4 admit the following upper bounds, obtained using convexity arguments and the assumption that $(\mathtt{S}, \mathtt{R})$ have kernels that follow a (p_s, p_r)-law:

$$|\mathcal{M}|^{-1} \left(\log_2 \left(|\mathcal{M}| p_s\right) + 2 \log_2 \log_2 \left(|\mathcal{M}| p_s\right) + 3\right), \tag{48}$$

$$|\mathcal{M}|^{-1} \left(\log_2 \left(|\mathcal{M}| p_r\right) + 2 \log_2 \log_2 \left(|\mathcal{M}| p_r\right) + 3\right), \tag{49}$$

$$\Lambda(p_s, p_r - p_s) + \frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|} + \frac{1}{|\mathcal{M}|}. \tag{50}$$

The last term uses Lemma 2. We now describe the decoding algorithm behind step 5. As of step 2, the receiver has decoded $|\kappa(\mathtt{S})|$ and step 4, the receiver has decoded the bin index $\mathrm{BIN}_{|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|}(\kappa(\mathtt{S}))$. Next he would like to recover $\kappa(\mathtt{S})$ itself. Recall that by assumption, $\mathtt{S} \vdash \mathtt{R}$ and therefore the receiver knows that irrespective of what $\mathtt{S}$ is, the following relation must hold:

$$\kappa(\mathtt{S}) \subseteq \kappa(\mathtt{R}).$$

Define the hypotheses set as

$$\mathrm{hypotheses}(\mathtt{S}, \mathtt{R}) \triangleq \left\{\alpha \subseteq \mathcal{M} : \alpha \subseteq \kappa(\mathtt{R}), |\alpha| = |\kappa(\mathtt{S})|, \mathrm{BIN}_{|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|}(\alpha) = \mathrm{BIN}_{|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|}(\kappa(\mathtt{S}))\right\}.$$

If the hypotheses set has cardinality exactly one, g^1 outputs the result of mapping the one element to the space of logic sentences using ℓ and additionally outputs "success"; here ℓ refers to the function that maps a kernel to a logic expression (c.f. Definition 28). Otherwise, g^1 outputs a dummy element from $\mathcal{L}$ (doesn't matter which) and additionally outputs "failure". We illustrate an example of a decoding failure in Figure 18. The box in white represents $\mathrm{BIN}_{|\kappa(\mathtt{S})|,|\kappa(\mathtt{R})|}(\mathtt{S})$, and the box in gray represents $\kappa(\mathtt{R})$. The various circles represent kernels that have been mapped to the same bin. The yellow circle represents the true kernel $\kappa(\mathtt{S})$ that we want Bob to recover. The kernels illustrated through the green color can be eliminated by the receiver from consideration, since they are not fully included in $\kappa(\mathtt{R})$. The kernel illustrated in red cannot be discerned from the kernel in yellow by the receiver, and thus in this example, we have a decoding failure.

To estimate the failure probability, we upper bound the probability of this event:

$$[|\text{hypotheses}(\mathtt{S},\mathtt{R})| \geq 2]\,.$$

Let I_α be equal to 1 if $\alpha \in \text{hypotheses}(\mathtt{S},\mathtt{R})$ and 0 otherwise. Then we can upper bound the error probability as

$$\begin{aligned} P\left([|\text{hypotheses}(\mathtt{S},\mathtt{R})| \geq 2]\right) &= P\left(\sum_\alpha I_\alpha \geq 2\right) \\ &= E_{\mathtt{S},\mathtt{R}}\left[P\left(\sum_\alpha I_\alpha \geq 2 | \mathtt{S},\mathtt{R}\right)\right]. \end{aligned} \tag{51}$$

We then write

$$P\left(\sum_\alpha I_\alpha \geq 2|\mathtt{S},\mathtt{R}\right) = 1 - P\left(\sum_{\alpha:\alpha\neq\kappa(\mathtt{S}),\alpha\subseteq\kappa(\mathtt{R}),|\alpha|=|\kappa(\mathtt{S})|} I_\alpha = 0|\mathtt{S},\mathtt{R}\right) \tag{52}$$

$$= 1 - \Pi_{\alpha\neq\kappa(\mathtt{S}),\alpha\subseteq\kappa(\mathtt{R}),|\alpha|=|\kappa(\mathtt{S})|} P(I_\alpha = 0|\mathtt{S},\mathtt{R}) \tag{53}$$

$$= 1 - \Pi_{\alpha\neq\kappa(\mathtt{S}),\alpha\subseteq\kappa(\mathtt{R}),|\alpha|=|\kappa(\mathtt{S})|} \left(1 - 2^{-|\mathcal{M}|\text{RATE}}\right)$$

$$\leq 1 - \left(1 - 2^{-|\mathcal{M}|\text{RATE}}\right)^{\left(\begin{array}{c} |\kappa(\mathtt{R})| \\ |\kappa(\mathtt{S})| \end{array}\right)}. \tag{54}$$

We describe the rationale behind this critical derivation. By construction, the hypotheses set always contains at least one element (the true kernel for the sentence in possession by Alice), and thus the first equality (52) focuses on estimating the failure probability by estimating instead the success probability, where every kernel other than $\kappa(\mathtt{S})$ is not mapped to the hypotheses set. The second equality (53) follows by recognizing that under the given conditioning, the terms identified in the summation are statistically independent and hence the overall probability can be reduced to a product of individual probabilities. The third equality uses the specifics on how we constructed the random hash function to compute the probability that a kernel is not mapped to the bin to which $\kappa(\mathtt{S})$ belongs. Finally (54) uses combinatorial counting arguments to obtain an estimate of the product. We bound the combinatorial as follows:

$$\begin{aligned} \left(\begin{array}{c} |\kappa(\mathtt{R})| \\ |\kappa(\mathtt{S})| \end{array}\right) &\leq 2^{|\mathcal{M}|\frac{|\kappa(\mathtt{R})|}{|\mathcal{M}|}H\left(\frac{|\kappa(\mathtt{S})|}{|\kappa(\mathtt{R})|}\right)} \\ &= 2^{|\mathcal{M}|\Lambda\left(\frac{|\kappa(\mathtt{S})|}{|\mathcal{M}|},\frac{|\kappa(\mathtt{R})|-|\kappa(\mathtt{S})|}{|\mathcal{M}|}\right)} \\ &= 2^{|\mathcal{M}|\text{RATE}-\log_2|\mathcal{M}|}. \end{aligned} \tag{55}$$

Recall now Bernoulli's inequality:

$$(1-a)^n \geq 1 - an.$$

Combining this inequality with (54), (55), and (51), we obtain

$$P\left([|\text{hypotheses}(\mathtt{S},\mathtt{R})| \geq 2]\right) \leq \frac{1}{|\mathcal{M}|}.$$

With this estimate, we can account for the remainder of the bits in the protocol (going on either direction) with the expression

$$\left(1 + P\left([|\text{hypotheses}(\mathtt{S},\mathtt{R})| \geq 2]\right)|\mathcal{M}|\right)/|\mathcal{M}| \leq 2/|\mathcal{M}|.$$

Using this estimate together with (48), (49), (50), we obtain that the cost in bits is at most

$$\Lambda(p_s, p_r - p_s) + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right).$$

The proof of the upper bound is completed by noting that since the above is the expected performance of a random hash function, there must exist at least one hash function with an error probability not worse than the average. □

5.6 Bob's sentence may not be entailed by Alice's

In Figure 19-a, we illustrate the same situation that we had described in Theorem 5, but with the addition of background information $\mathtt{r}$ that is known to both Bob and Alice, and with the assumption $\mathtt{s} \vdash \mathtt{r}$. Now suppose that it is no longer the case that $\mathtt{s} \vdash \mathtt{r}$, then what we obtain is the more general setup in Figure 19, where as it can be appreciated, $\kappa(\mathtt{r})$ no longer contains $\kappa(\mathtt{s})$ fully and where the complement of $\kappa(\mathtt{r})$ is patterned with a "dotted" fill. This more general setup is the subject of this subsection.

In order to derive results for this setting, we will need to make additional assumptions. For the first time in the article, we will use a *proper* Logic System (see Definition 2), where we assume the availability of the standard logic operators $\vee$, $\wedge$, $\neg$ as well as the set theoretic implications of the kernels resulting from such operations; see Equations (1-3).

We introduce a more complex set of measurements that need to be made on the probability laws of $\mathtt{S}, \mathtt{Q}, \mathtt{R}$. We keep the assumption $\mathtt{S} \vdash \mathtt{Q}$ in (10) but drop the assumptions $\mathtt{S} \vdash \mathtt{R}, \mathtt{Q} \vdash \mathtt{R}$ in (9),(11). We measure the following expected kernel sizes:

$$\begin{aligned} p_r &\triangleq |\mathcal{M}|^{-1} E|\kappa(\mathtt{R})|, \\ p_{s^*} &\triangleq |\mathcal{M}|^{-1} E|\kappa(\mathtt{S} \wedge \mathtt{R})|, \\ p_{q^*} &\triangleq |\mathcal{M}|^{-1} E|\kappa(\mathtt{Q} \wedge \mathtt{R})|, \\ p_{s^{**}} &\triangleq |\mathcal{M}|^{-1} E|\kappa(\mathtt{S} \wedge \neg\mathtt{R})|, \\ p_{q^{**}} &\triangleq |\mathcal{M}|^{-1} E|\kappa(\mathtt{Q} \wedge \neg\mathtt{R})|. \end{aligned}$$

Our upper bounds will be phrased in terms of these expected sizes. Together with the assumption that $\mathtt{S} \vdash \mathtt{Q}$, we say that the kernels of $\mathtt{S}, \mathtt{Q}, \mathtt{R}$ follow a $(p_r, p_{s^*}, p_{q^*}, p_{s^{**}}, p_{q^{**}})$-law.

As with previous results, we briefly introduce the strategy for proving the upper bound for this result is illustrated in Figure 19. While the figure appears formidable, upon further examination its elements are quickly decomposed into elements that should be familiar to the reader now. At the highest level, the problem is simply split in two: because both Alice and Bob know $\mathtt{r}$, they can create a dual strategy: one to handle sending whatever piece of $\kappa(\mathtt{s})$ and $\kappa(\mathtt{q})$ that will intersect with $\kappa(\mathtt{r})$, and the other one to handle the same but that intersects with $\kappa(\mathtt{r})^c$; this explains why there is vertical symmetry on the figure. Then, focusing on, say, only the top half of the diagram, we realize that the resulting system is in essence a combination of the strategies used in Theorems 3 and 5. As a result, the fundamental device for efficiently sending partitions, denoted by a circle with the $+$ and $-$ hooks (and fully addressed in Theorem 4) is used twice.

Theorem 7 (Bob's sentence may not be entailed by Alice's) *Let $(L, \mathcal{M}, \kappa, \ell)$ be a proper Logic System. Let $\mathtt{S}$, $\mathtt{Q}$, $\mathtt{R}$ represent the sender, query and receiver logic sentences, respectively, which we assume have kernels that follow a $(p_r, p_{s^*}, p_{q^*}, p_{s^{**}}, p_{q^{**}})$-law. Let the encoder f and decoder g be functions*

$$\begin{aligned} f : \mathcal{L}^3 &\rightarrow \{0,1\}^* \\ g : \{0,1\}^* \times \mathcal{L} &\rightarrow \mathcal{L}. \end{aligned}$$

Then

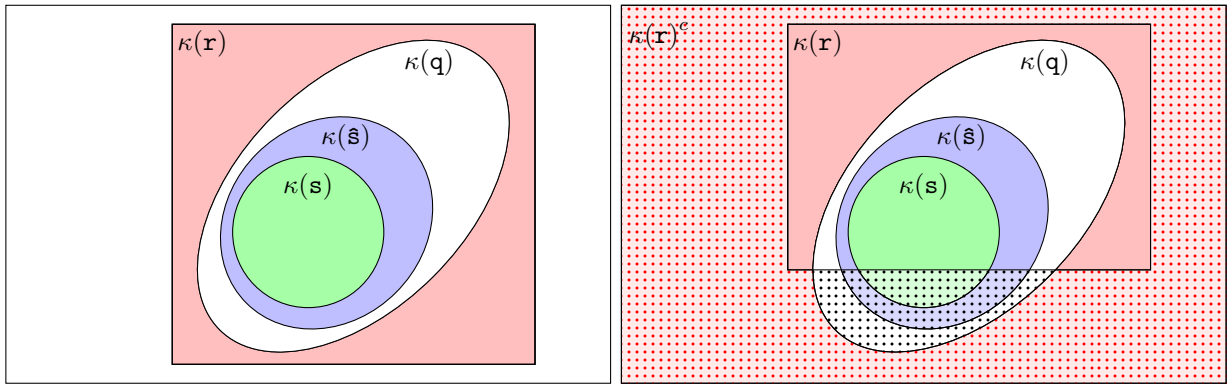
$$\min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{Q},\mathtt{R}} [\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{R}))] \leq \Lambda(p_{s^*}, p_r - p_{q^*}) + \Lambda(p_{s^{**}}, 1 - p_r - p_{q^{**}}) + O\left(\frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|}\right)$$

where the minimization is over f, g such that the image of $f(\cdot,\cdot,\mathtt{r})$ is prefix-free for any choice of $\mathtt{r}$, and such that if $\mathtt{s} \vdash \mathtt{q}$ then $\mathtt{s} \vdash g(f(\mathtt{s},\mathtt{q},\mathtt{r}),\mathtt{r})$, and $g(f(\mathtt{s},\mathtt{q},\mathtt{r}),\mathtt{r}) \vdash \mathtt{q}$. Furthermore, if $\{(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}), \vec{\kappa}(\mathtt{R}))_i\}_{i=1}^{|\mathcal{M}|}$ are i.i.d. and $\mathtt{R} \rightarrow \kappa(\mathtt{R}) \rightarrow (\kappa(\mathtt{S}), \kappa(\mathtt{Q}))$, then

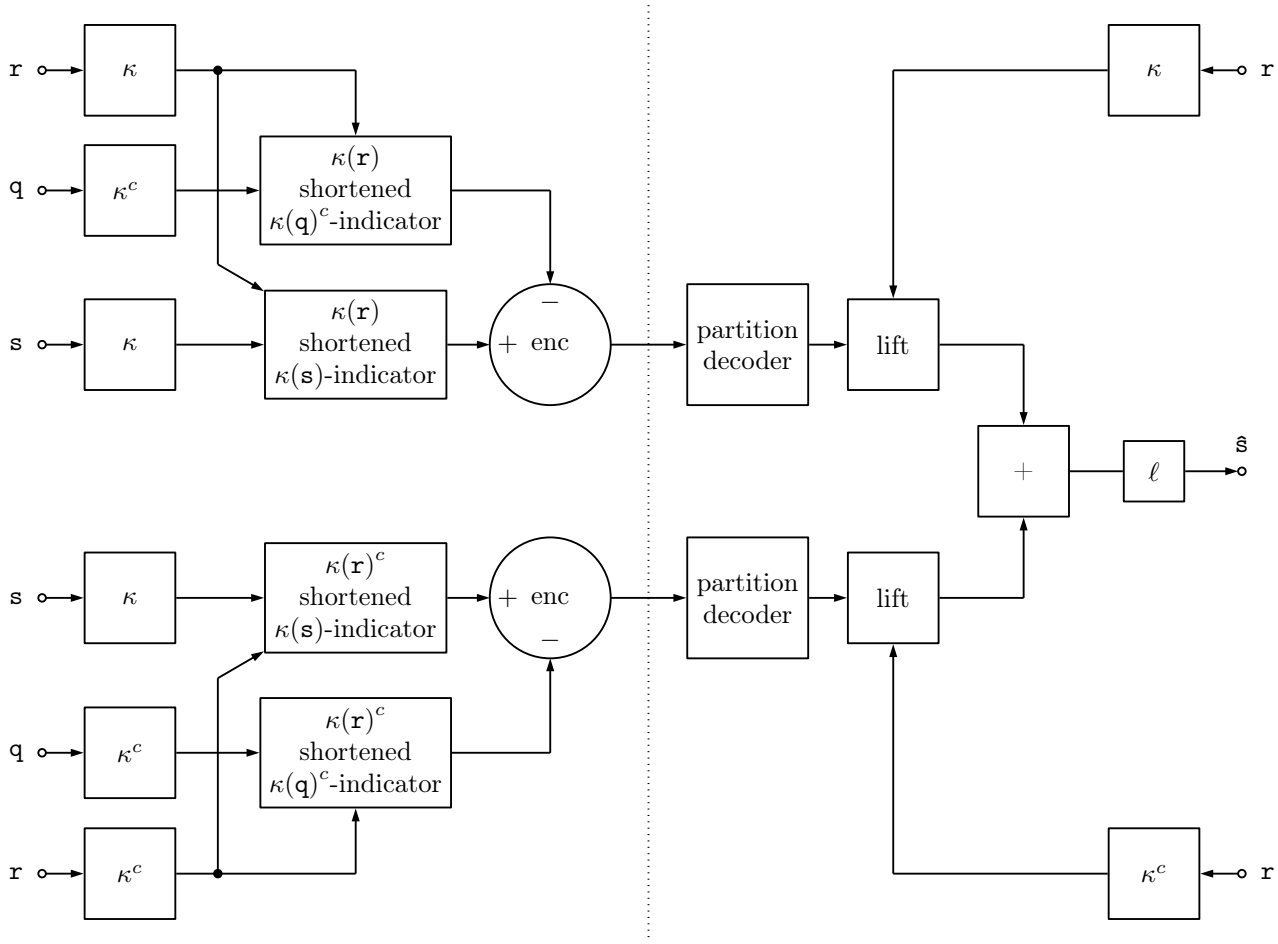
$$\Lambda(p_{s^*}, p_r - p_{q^*}) + \Lambda(p_{s^{**}}, 1 - p_r - p_{q^{**}}) \leq \min_{f,g} |\mathcal{M}|^{-1} E_{\mathtt{S},\mathtt{Q},\mathtt{R}} [\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{R}))]$$

Proof. The proof of this result builds upon the ideas in the proofs of Theorem 3 and 5. We start using the law of total expectation to write

$$E_{\mathtt{S},\mathtt{Q},\mathtt{R}} [\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{R}))] = E_{\mathtt{R}} [E_{\mathtt{S},\mathtt{Q}} [\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{R}))|\mathtt{R}]]$$



(a) (b)



(c)

Figure 19: Proof strategy for Theorem 7

We invoke again Kraft's inequality through Lemma 4 and write

$$\begin{aligned} E_{\mathtt{S},\mathtt{Q}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{r}))|\mathtt{R}=\mathtt{r}\right] &= E_{f(\mathtt{S},\mathtt{Q},\mathtt{r})|\mathtt{R}=\mathtt{r}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{r}))|\mathtt{R}=\mathtt{r}\right] \\ &\geq H(f(\mathtt{S},\mathtt{Q},\mathtt{r})|\mathtt{R}=\mathtt{r}) \end{aligned}$$

where we have used the assumption that $f(\cdot,\cdot,\mathtt{r})$ is prefix-free for any choice of $\mathtt{r}$. From here, we apply the same ideas as in Theorem 5 to obtain

$$E_{\mathtt{S},\mathtt{Q}|\mathtt{R}=\mathtt{r}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{r}))|\mathtt{R}=\mathtt{r}\right] \geq I(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q});\vec{\kappa}(g(f(\mathtt{S},\mathtt{Q},\mathtt{r}),\mathtt{r}))|\mathtt{R}=\mathtt{r}).$$

Recall the definition of ψ in (43) and write

$$\begin{aligned} X_1^{|\mathcal{M}|} &= \psi(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})), \\ Z_1^{|\mathcal{M}|} &= 1 \times \vec{\kappa}(g(f(\mathtt{S},\mathtt{Q},\mathtt{r}),\mathtt{r})). \end{aligned}$$

Similarly, as before note that the assumptions that $\mathtt{s} \vdash g(f(\mathtt{s},\mathtt{q},\mathtt{r})$ and $\mathtt{r})$, $g(f(\mathtt{s},\mathtt{q},\mathtt{r}),\mathtt{r}) \vdash \mathtt{q}$ imply that for all $1 \leq i \leq |\mathcal{M}|$,

$$\rho(X_i, Z_i) = 0.$$

We continue applying the ideas in the proof of Theorem 5 and obtain

$$\begin{aligned} & I(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q});\vec{\kappa}(g(f(\mathtt{S},\mathtt{Q},\mathtt{r}),\mathtt{r}))|\mathtt{R}=\mathtt{r}) \\ &\geq I(X_1^{|\mathcal{M}|};Z_1^{|\mathcal{M}|}|\mathtt{R}=\mathtt{r}) \\ &= H(X_1^{|\mathcal{M}|}|\mathtt{R}=\mathtt{r}) - H(Z_1^{|\mathcal{M}|}|X_1^{|\mathcal{M}|},\mathtt{R}=\mathtt{r}) \\ &\overset{(a)}{=} H(X_1^{|\mathcal{M}|}|\vec{\kappa}(\mathtt{R})=\vec{\kappa}(\mathtt{r})) - H(Z_1^{|\mathcal{M}|}|X_1^{|\mathcal{M}|},\mathtt{R}=\mathtt{r}) \\ &\overset{(b)}{\geq} H(X_1^{|\mathcal{M}|}|\vec{\kappa}(\mathtt{R})=\vec{\kappa}(\mathtt{r})) - H(Z_1^{|\mathcal{M}|}|X_1^{|\mathcal{M}|},\vec{\kappa}(\mathtt{R})=\vec{\kappa}(\mathtt{r})) \\ &= I(X_1^{|\mathcal{M}|};Z_1^{|\mathcal{M}|}|\vec{\kappa}(\mathtt{R})=\vec{\kappa}(\mathtt{r})) \end{aligned}$$

where (a) follows from the assumption that the following Markov chain holds:

$$\mathtt{R} \to \vec{\kappa}(\mathtt{R}) \to (\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})),$$

and (b) follows from the fact that conditional entropy can only increase under weaker conditioning.

We will next argue that conditioned on $\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})$, the $\{X_i\}$ are independent, and distributed according to at most two distributions, which follow from the assumptions of the distribution of $\mathtt{S},\mathtt{R},\mathtt{Q}$.

Lemma 6 *Let the random variables $A, B, C \in \{0,1\}$ and assume that $A \leq B$. Then the distribution of $\psi(A,B)$ given $C = c$*

$$\{P(B=0|C=c), P(A=1|C=c), 1-P(B=0|C=c)-P(A=1|C=c)\}. \tag{56}$$

Proof. Recall that

$$\psi(A,B) = A + 2(1-A)B$$

and therefore, the the following event equivalences hold:

$$\begin{aligned} [\psi(A,B)=1] &= [A=1] \\ [\psi(A,B)=0] &= [A=0,B=0] = [B=0] \end{aligned}$$

where the last equality follows from the assumption that $A \leq B$. Then given $C = c$,

$$P(\psi(A,B)=1|C=c) = P(A=1|C=c)$$

$$P(\psi(A,B)=0|C=c) \quad = \quad P(B=0|C=c)$$

□

We now write

$$\begin{aligned}
P(X_i=0|\vec{\kappa}(\mathtt{R})=\vec{\kappa}(\mathtt{r})) &\overset{(a)}{=} P(X_i=0|\vec{\kappa}(\mathtt{R})_i=\vec{\kappa}(\mathtt{r})_i)\\
&\overset{(b)}{=} P(\vec{\kappa}(\mathtt{Q})_i=0|\vec{\kappa}(\mathtt{R})_i=\vec{\kappa}(\mathtt{r})_i)\\
&\overset{(c)}{=} P(\vec{\kappa}((\mathtt{Q}\wedge\mathtt{R})\vee(\mathtt{Q}\wedge\neg\mathtt{R}))_i=0|\vec{\kappa}(\mathtt{R})_i=\vec{\kappa}(\mathtt{r})_i)\\
&\overset{(d)}{=} P([\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]|\vec{\kappa}(\mathtt{R})_i=\vec{\kappa}(\mathtt{r})_i)\\
&\overset{(e)}{=} \begin{cases} P([\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]|\vec{\kappa}(\mathtt{R})_i=1) & \text{if } \vec{\kappa}(\mathtt{r})_i=1\\ P([\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]|\vec{\kappa}(\mathtt{R})_i=0) & \text{if } \vec{\kappa}(\mathtt{r})_i=0\end{cases}\\
&\overset{(f)}{=} \begin{cases} 1-p_{q^*}/p_r & \text{if } \vec{\kappa}(\mathtt{R})_i=1\\ 1-p_{q^{**}}/(1-p_r) & \text{if } \vec{\kappa}(\mathtt{R})_i=0\end{cases}
\end{aligned}$$

where (a) follows from the Definition $X_i=\psi(\vec{\kappa}(\mathtt{S})_i,\vec{\kappa}(\mathtt{Q})_i)$ and the independence assumption and (b) follows from Lemma 6. To go from (b) to (c) we establish that $\kappa(\mathtt{Q})=\kappa((\mathtt{Q}\wedge\mathtt{R})\vee(\mathtt{Q}\wedge\neg\mathtt{R}))$ using the conditions for being a proper Logic System (Definition 2). By condition 2.3, $\kappa(\neg\mathtt{R})=\kappa(\mathtt{R})^c$. Thus,

$$\begin{aligned}
\kappa(\mathtt{Q}) &= \kappa(\mathtt{Q})\cap(\kappa(\mathtt{R})\cup\kappa(\neg\mathtt{R}))\\
&= (\kappa(\mathtt{Q})\cap\kappa(\mathtt{R}))\cup\kappa((\mathtt{Q})\cap\kappa(\neg\mathtt{R})) &(57)\\
&= \kappa(\mathtt{Q}\wedge\mathtt{R})\cup\kappa(\mathtt{Q}\wedge\neg\mathtt{R}) &(58)\\
&= \kappa((\mathtt{Q}\wedge\mathtt{R})\vee(\mathtt{Q}\wedge\neg\mathtt{R})), &(59)
\end{aligned}$$

where (57) follows by the DeMorgan Laws for sets, (58) follows by condition 2.2, and (59) follows by condition 2.1. Then, to go from (c) to (d), note that

$$\begin{aligned}
\kappa(\neg((\mathtt{Q}\wedge\mathtt{R})\vee(\mathtt{Q}\wedge\neg\mathtt{R}))) &= \kappa((\mathtt{Q}\wedge\mathtt{R})\vee(\mathtt{Q}\wedge\neg\mathtt{R}))^c &(60)\\
&= (\kappa(\mathtt{Q}\wedge\mathtt{R})\cup\kappa(\mathtt{Q}\wedge\neg\mathtt{R}))^c &(61)\\
&= \kappa(\mathtt{Q}\wedge\mathtt{R})^c\cap\kappa(\mathtt{Q}\wedge\neg\mathtt{R})^c &(62)\\
&= \kappa(\neg(\mathtt{Q}\wedge\mathtt{R}))\cap\kappa(\neg(\mathtt{Q}\wedge\neg\mathtt{R})), &(63)
\end{aligned}$$

where equality (60) follows from condition 2.3, (61) follows by condition 2.1, (62) follows by DeMorgan, and (63) follows by condition 2.3. Putting these together, it follows that $\vec{\kappa}((\mathtt{Q}\wedge\mathtt{R})\vee(\mathtt{Q}\wedge\neg\mathtt{R}))_i=0$ iff $[\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]$. Here we use the fact that if $\mathtt{s}\in\mathcal{L}$ is a sentence, then $\vec{\kappa}(\mathtt{s})_i=0$ iff $\vec{\kappa}(\neg\mathtt{s})_i=1$, which follows from condition 2.1, $\kappa(\neg\mathtt{s})=\kappa(\mathtt{s})^c$, since taking the complement of the kernel corresponds to flipping 1s to 0s and 0s to 1s in the vector notation. To obtain (e), note that

$$\begin{aligned}
[\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{R})_i=1] &= [\vec{\kappa}(\mathtt{R})_i=1]\\
[\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{R})_i=0] &= [\vec{\kappa}(\mathtt{R})_i=0]
\end{aligned}$$

From this observation one then may trivially write

$$\begin{aligned}
[\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{R})_i=1] &= [\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{R})_i=1]\\
[\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{R})_i=0] &= [\vec{\kappa}(\mathtt{Q}\wedge\neg\mathtt{R})_i=0]\cap[\vec{\kappa}(\mathtt{R})_i=0]
\end{aligned}$$

from which the assertion follows. Finally (f) follows from the expected average kernel sizes as given in the theorem, in conjunction with the "identically distributed" assumption. Specifically,

$$\begin{aligned}
P([\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=0|\vec{\kappa}(\mathtt{R})_i=1]) &= 1-P([\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=1|\vec{\kappa}(\mathtt{R})_i=1])\\
&= 1-\frac{P([\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=1])P(\vec{\kappa}(\mathtt{R})_i=1|\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=1)}{P(\vec{\kappa}(\mathtt{R})_i=1)}\\
&= 1-\frac{P([\vec{\kappa}(\mathtt{Q}\wedge\mathtt{R})_i=1])}{P(\vec{\kappa}(\mathtt{R})_i=1)}
\end{aligned}$$

$$= \quad 1 - p_{q^*}/p_r$$

with a parallel derivation applicable for the other equation in (f).

Similarly, one can write

$$\begin{aligned}
P(X_i = 1|\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) &\overset{(a)}{=} P(X_i = 1|\vec{\kappa}(\mathtt{R})_i = \vec{\kappa}(\mathtt{r})_i) \\
&\overset{(b)}{=} P(\vec{\kappa}(\mathtt{S})_i = 1|\vec{\kappa}(\mathtt{R})_i = \vec{\kappa}(\mathtt{r})_i) \\
&\overset{(c)}{=} P(\vec{\kappa}((\mathtt{S} \wedge \mathtt{R}) \vee (\mathtt{S} \wedge \neg\mathtt{R}))_i = 1|\vec{\kappa}(\mathtt{R})_i = \vec{\kappa}(\mathtt{r})_i) \\
&\overset{(d)}{=} P([\vec{\kappa}(\mathtt{S} \wedge \mathtt{R})_i = 1] \cup [\vec{\kappa}(\mathtt{S} \wedge \neg\mathtt{R})_i = 1]|\vec{\kappa}(\mathtt{R})_i = \vec{\kappa}(\mathtt{r})_i) \\
&\overset{(e)}{=} P(\vec{\kappa}(\mathtt{S} \wedge \mathtt{R})_i = 1|\vec{\kappa}(\mathtt{R})_i = \vec{\kappa}(\mathtt{r})_i) + P(\vec{\kappa}(\mathtt{S} \wedge \neg\mathtt{R})_i = 1|\vec{\kappa}(\mathtt{R})_i = \vec{\kappa}(\mathtt{r})_i) \\
&\overset{(f)}{=} \begin{cases} P([\vec{\kappa}(\mathtt{S} \wedge \mathtt{R})_i = 1]|\vec{\kappa}(\mathtt{R})_i = 1) & \text{if } \vec{\kappa}(\mathtt{r})_i = 1 \\ P([\vec{\kappa}(\mathtt{S} \wedge \neg\mathtt{R})_i = 1]|\vec{\kappa}(\mathtt{R})_i = 0) & \text{if } \vec{\kappa}(\mathtt{r})_i = 0 \end{cases} \\
&\overset{(g)}{=} \begin{cases} p_{s^*}/p_r & \text{if } \vec{\kappa}(\mathtt{r})_i = 1 \\ p_{s^{**}}/(1-p_r) & \text{if } \vec{\kappa}(\mathtt{r})_i = 0 \end{cases}
\end{aligned}$$

Steps (a-d) can be justified using arguments very similar to those of the previous derivation. Step (e) follows from the observation that

$$[\vec{\kappa}(\mathtt{S} \wedge \mathtt{R})_i = 1] \cap [\vec{\kappa}(\mathtt{S} \wedge \neg\mathtt{R})_i = 1] = \emptyset$$

Step (f) can be seen by noting that if $\vec{\kappa}(\mathtt{r})_i = 1$, then the second term of (e) is zero, and if $\vec{\kappa}(\mathtt{r})_i = 0$, then the first term of (e) is zero. Step (g) follows from the expected average kernel sizes and the identically distributed assumption, using arguments similar to those of in step (f) of the earlier derivation.

By construction, the X_is are mutually independent conditional on $\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})$. Using this fact, we can now write

$$\begin{aligned}
&I(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}); \vec{\kappa}(g(f(\mathtt{S}, \mathtt{Q}, \mathtt{r}), \mathtt{r}))|\mathtt{R} = \mathtt{r}) \\
&\quad \geq \quad I(X_1^{|\mathcal{M}|}; Z_1^{|\mathcal{M}|}|\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) \\
&\quad = \quad \sum_i H(X_i|\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) - \sum_i H(X_i|Z_1^{|\mathcal{M}|}X_1^{i-1}, \vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) \\
&\quad \geq \quad \sum_i I(X_i; Z_i|\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) \\
&\quad = \quad \sum_{i:\vec{\kappa}(\mathtt{r})_i = 1} I(X_i; Z_i|\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) + \sum_{i:\vec{\kappa}(\mathtt{r})_i = 0} I(X_i; Z_i|\vec{\kappa}(\mathtt{R}) = \vec{\kappa}(\mathtt{r})) \\
&\quad \geq \quad |\vec{\kappa}(\mathtt{r})| \min_{X \sim (1-p_{q^*}/p_r, p_{s^*}/p_r, p_{q^*}/p_r - p_{s^*}/p_r), Q_{Z|X}: E[\rho(X,Z)]=0} I(X;Z) \\
&\qquad + |\vec{\kappa}(\mathtt{r})^c| \min_{X \sim (1-p_{q^{**}}/(1-p_r), p_{s^{**}}/(1-p_r), p_{q^{**}}/(1-p_r) - p_{s^{**}}/(1-p_r)), Q_{Z|X}: E[\rho(X,Z)]=0} I(X;Z) \\
&\quad \geq \quad |\vec{\kappa}(\mathtt{r})|\Lambda(p_{s^*}/p_r, 1 - p_{q^*}/p_r) + |\vec{\kappa}(\mathtt{r})^c|\Lambda(p_{s^{**}}/(1-p_r), 1 - p_{q^{**}}/(1-p_r)).
\end{aligned}$$

Finally note that $E_{\mathtt{R}}[|\vec{\kappa}(\mathtt{R})|] = |\mathcal{M}|p_r$ and thus putting everything together

$$\begin{aligned}
&E_{\mathtt{S},\mathtt{Q},\mathtt{R}}[\mathbf{len}(f(\mathtt{S}, \mathtt{Q}, \mathtt{R}))] \\
&\quad \geq \quad |\mathcal{M}|p_r\Lambda(p_{s^*}/p_r, 1 - p_{q^*}/p_r) + |\mathcal{M}|(1-p_r)\Lambda(p_{s^{**}}/(1-p_r), 1 - p_{q^{**}}/(1-p_r)) \\
&\quad = \quad |\mathcal{M}|\Lambda(p_{s^*}, p_r - p_{q^*}) + |\mathcal{M}|\Lambda(p_{s^{**}}, 1 - p_r - p_{q^{**}}).
\end{aligned}$$

We now prove the upper bound, which we emphasize, will be developed under the weaker assumption that $\mathtt{S}, \mathtt{Q}, \mathtt{R}$ follow a $(p_r, p_{s^*}, p_{s^{**}}, p_{q^*}, p_{q^{**}})$-law. Let f and g be the encoder and decoder functions that we will be constructing. Let $\mathtt{s}, \mathtt{q}, \mathtt{r} \in \mathcal{L}$ and define

$$\mathtt{s}^* \quad = \quad \mathtt{s} \wedge \mathtt{r}$$

$$\begin{aligned}
\mathtt{s}^{**} &= \mathtt{s}\wedge\neg\mathtt{r} \\
\mathtt{q}^{*} &= \mathtt{q}\wedge\mathtt{r} \\
\mathtt{q}^{**} &= \mathtt{q}\wedge\neg\mathtt{r}.
\end{aligned}$$

We similarly define random versions of the above by replacing the lower case letters with upper case letters. We can compute, for the random versions, the corresponding normalized expected kernel sizes:

$$\begin{aligned}
p_r &= |\mathcal{M}|^{-1}E|\kappa(\mathtt{R})| \\
p_{s^*} &= |\mathcal{M}|^{-1}E|\kappa(\mathtt{S}^*)| \\
p_{s^{**}} &= |\mathcal{M}|^{-1}E|\kappa(\mathtt{S}^{**})| \\
p_{q^*} &= |\mathcal{M}|^{-1}E|\kappa(\mathtt{Q}^*)| \\
p_{q^{**}} &= |\mathcal{M}|^{-1}E|\kappa(\mathtt{Q}^{**})|.
\end{aligned}$$

Also define

$$\begin{aligned}
j_0 &= \{i : \vec{\kappa}(\mathtt{r})_i = 0\} \\
j_1 &= \{i : \vec{\kappa}(\mathtt{r})_i = 1\} \\
x_1^* \cdots x_{|\vec{\kappa}(\mathtt{r})|}^* &= [\psi(\vec{\kappa}(\mathtt{s}^*), \vec{\kappa}(\mathtt{q}^*))]_{j_1} \\
x_1^{**} \cdots x_{|\mathcal{M}|-|\vec{\kappa}(\mathtt{r})|}^{**} &= [\psi(\vec{\kappa}(\mathtt{s}^{**}), \vec{\kappa}(\mathtt{q}^{**}))]_{j_0}
\end{aligned}$$

and as before, define random versions by replacing lower case letters with upper case letters.

Using Theorem 4 twice we can guarantee the existence of f^*, g^*, f^{**}, g^{**} such that

$$\begin{aligned}
E\left[\mathbf{len}(f^*(X_1^* \cdots X_{|\vec{\kappa}(\mathtt{R})|}^*))|\vec{\kappa}(\mathtt{R})\right] &\leq |\vec{\kappa}(\mathtt{R})|\Lambda(p_{s^*}/p_r, 1-p_{q^*}/p_r) + \\
&\quad +2\log_2\left(|\vec{\kappa}(\mathtt{R})|\Lambda(p_{s^*}/p_r, 1-p_{q^*}/p_r)\right) + 3 \\
\sum_{i=1}^{|\vec{\kappa}(\mathtt{R})|} \rho(X_i^*, g^*(f^*(X_1^* \cdots X_{|\vec{\kappa}(\mathtt{R})|}^*))_i) &= 0 \\
E\left[\mathbf{len}(f^{**}(X_1^{**} \cdots X_{|\mathcal{M}|-|\vec{\kappa}(\mathtt{R})|}^{**}))|\vec{\kappa}(\mathtt{R})\right] &\leq (|\mathcal{M}| - |\vec{\kappa}(\mathtt{R})|)\Lambda(p_{s^{**}}/(1-p_r), 1-p_{q^{**}}/(1-p_r)) \\
&\quad +2\log_2\left((|\mathcal{M}| - |\vec{\kappa}(\mathtt{R})|)\Lambda(p_{s^{**}}/(1-p_r), 1-p_{q^{**}}/(1-p_r))\right) + 3 \\
\sum_{i=1}^{|\mathcal{M}|-|\vec{\kappa}(\mathtt{R})|} \rho(X_i^{**}, g^{**}(f^{**}(X_1^{**} \cdots X_{|\mathcal{M}|-|\vec{\kappa}(\mathtt{R})|}^{**}))_i) &= 0.
\end{aligned}$$

Next we assemble an encoder and decoder for our setup. Given two binary strings $c_1, c_2 \in \{0,1\}^*$, let c_1c_2 denote the string resulting from concatenating the two individual strings. We define our encoder as

$$f(\mathtt{s},\mathtt{q},\mathtt{r}) = f^*(x_1^* \cdots x_{|\vec{\kappa}(\mathtt{r})|}^{**}) f^{**}(x_1^{**} \cdots x_{|\mathcal{M}|-|\vec{\kappa}(\mathtt{r})|}^{**}).$$

From our earlier estimates,

$$\begin{aligned}
E_{\mathtt{S},\mathtt{Q},\mathtt{R}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{R}))|\vec{\kappa}(\mathtt{R})\right] &\leq |\vec{\kappa}(\mathtt{R})|\Lambda(p_{s^*}/p_r, 1-p_{q^*}/p_r) \\
&\quad +(|\mathcal{M}| - |\vec{\kappa}(\mathtt{R})|)\Lambda(p_{s^{**}}/(1-p_r), 1-p_{q^{**}}/(1-p_r)) + 6 \\
&\quad +2\log_2|\vec{\kappa}(\mathtt{R})|\Lambda(p_{s^*}/p_r, 1-p_{q^*}/p_r) \\
&\quad +2\log_2\left((|\mathcal{M}| - |\vec{\kappa}(\mathtt{R})|)\Lambda(p_{s^{**}}/(1-p_r), 1-p_{q^{**}}/(1-p_r))\right)
\end{aligned}$$

and using the law of total expectations, and the concavity $\cap$ of the logarithm, we obtain

$$\begin{aligned}
|\mathcal{M}|^{-1}E_{\mathtt{S},\mathtt{Q},\mathtt{R}}\left[\mathbf{len}(f(\mathtt{S},\mathtt{Q},\mathtt{R}))\right] &\leq p_r\Lambda(p_{s^*}/p_r, 1-p_{q^*}/p_r) \\
&\quad +(1-p_r)\Lambda(p_{s^{**}}/(1-p_r), 1-p_{q^{**}}/(1-p_r)) + 6|\mathcal{M}|^{-1} \\
&\quad +2|\mathcal{M}|^{-1}\log_2\left(p_r|\mathcal{M}|\Lambda(p_{s^*}/p_r, 1-p_{q^*}/p_r)\right)
\end{aligned}$$

$$+2|\mathcal{M}|^{-1}\log_2\left((1-p_r)|\mathcal{M}|\Lambda(p_{s^{**}}/(1-p_r),1-p_{q^{**}}/(1-p_r))\right).$$

Now we construct the decoder. We define a function *lift*, on two arguments: a *filter* and a *pattern*. The function lifts a binary *pattern* to into a larger binary vector using the positions indicated by *filter*, maintaining the order of the bits in the *pattern*. For a vector $\mathit{filter} \in \{0,1\}^{|\mathcal{M}|}$ and $\mathit{pattern} \in \{0,1\}^{|\mathit{filter}|}$ (where $|\mathit{filter}|$ indicates the number of ones in the vector *filter*), let a_i, for $i = 1,\ldots,|\mathit{filter}|$, be the positions such that $\mathit{filter}_{a_i} = 1$. We then define

$$\mathit{lift}(\mathit{filter},\mathit{pattern})_{a_i} = \mathit{pattern}_i$$

and define *lift* to be 0 in all other indices.

The code that Theorem 4 guarantees existence of is prefix free, and thus the output of $f(\mathtt{S},\mathtt{Q},\mathtt{R})$ can be decoded sequentially. Define

$$\begin{aligned}\mathbf{a} &= \mathit{lift}\left(\vec{\kappa}(\mathtt{r}), g^*\left(f^*\left(x_1^*\cdots x^*_{|\vec{\kappa}(\mathtt{R})|}\right)\right)\right)\\ \mathbf{b} &= \mathit{lift}\left(1-\vec{\kappa}(\mathtt{r}), g^{**}\left(f^{**}\left(x_1^{**}\cdots x^{**}_{|\mathcal{M}|-|\vec{\kappa}(\mathtt{r})|}\right)\right)\right).\end{aligned}$$

Now, assume that $\mathtt{q}$ is such that $\mathtt{s} \vdash \mathtt{q}$. Then by we know that

$$\begin{aligned}\vec{\kappa}(\mathtt{s})\vec{\kappa}(\mathtt{r}) \le \mathbf{a} \le \vec{\kappa}(\mathtt{q})\vec{\kappa}(\mathtt{r}),\\ \vec{\kappa}(\mathtt{s})(1-\vec{\kappa}(\mathtt{r})) \le \mathbf{b} \le \vec{\kappa}(\mathtt{q})(1-\vec{\kappa}(\mathtt{r}))\end{aligned}$$

where the products and inequalities are interpreted element-wise. And as a consequence

$$\vec{\kappa}(\mathtt{s}) \le \mathbf{a}+\mathbf{b} \le \vec{\kappa}(\mathtt{q}).$$

The output of the decoder g is defined to be

$$\vec{\ell}(\mathbf{a}+\mathbf{b}).$$

This ensures that the requirement $g(f(\mathtt{s},\mathtt{q},\mathtt{r})) \vdash \mathtt{q}$ is satisfied. □

5.7 General setup when Alice does not know what Bob knows

To conclude our sequence of theorems, we study a situation where Alice does not know the specific sentence $\mathtt{R}$ knows, and is trying to ensure that Bob knows how to prove a targeted query $\mathtt{Q}$. This setup can be seen as a combination of the setups in Theorem 5 and Theorem 6, and it is the most complex situation that we analyze from the standpoint of the mathematical machinery used to prove it.

In a manner similar to the discussion leading to Theorem 6, we rely on Table 2 in order to define the communication pattern that we analyze. A total of 6 turns are allowed, starting with Bob making the first communication and concluding with Bob being able to prove a targeted query $\mathtt{Q}$. The outputs for all the encoders are denoted by $B^0, A^1, B^2, A^3, B^4, A^5$. The cost of communication is given by

$$\frac{1}{|\mathcal{M}|}E_{\mathtt{S},\mathtt{Q},\mathtt{R}}\left[\mathbf{len}(B^0)+\mathbf{len}(A^1)+\mathbf{len}(B^2)+\mathbf{len}(A^3)+\mathbf{len}(B^4)+\mathbf{len}(A^5)\right].$$

The image of all of the encoders is assumed to be prefix-free, assuming that whatever is the common context is kept fixed when considering the image. For a more detailed discussion of this, note the comments associated with Table 1. A 6-turn code $\{f^i,g^i\}_{i=0}^5$ as defined by Table 2 and this discussion is called a 6-turn code for targeted queries. The reason for the word "targeted" is because the goal for the communication can be restricted to be a subset of Alice's knowledge, rather than its entirety.

Theorem 8 *Let $(L,\mathcal{M},\kappa,\ell)$ be a Logic System. Let $\mathtt{S},\mathtt{Q},\mathtt{R} \in \mathcal{L}$ represent the sender, query and receiver logic sentences, with $\mathtt{S},\mathtt{Q}$ only known to Alice and $\mathtt{R}$ only known to Bob. Assume that $\mathtt{S},\mathtt{Q},\mathtt{R}$ have kernels that follow a (p_s,p_q,p_r)-law. Then*

$$\min_{\{f^i,g^i\}_{i=0}^5}\frac{1}{|\mathcal{M}|}E_{\mathtt{S},\mathtt{Q},\mathtt{R}}\left[\mathbf{len}(B^0)+\mathbf{len}(A^1)+\mathbf{len}(B^2)+\mathbf{len}(A^3)+\mathbf{len}(B^4)+\mathbf{len}(A^5)\right]$$

Alice's private context	common context	Alice's function	direction	Bob's function	common context	Bob's private context
$\mathtt{s},\mathtt{q}$		g^0	$\xleftarrow{b^0}$	f^0		$\mathtt{r}$
$\mathtt{s},\mathtt{q}$	b^0	f^1	$\xrightarrow{a^1}$	g^1	b^0	$\mathtt{r}$
$\mathtt{s},\mathtt{q}$	b^0,a^1	g^2	$\xleftarrow{b^2}$	f^2	b^0,a^1	$\mathtt{r}$
$\mathtt{s},\mathtt{q}$	b^0,a^1,b^2	f^3	$\xrightarrow{a^3}$	g^3	b^0,a^1,b^2	$\mathtt{r}$
$\mathtt{s},\mathtt{q}$	b^0,a^1,b^2,a^3	g^4	$\xleftarrow{b^4}$	f^2	b^0,a^1,b^2,a^3	$\mathtt{r}$
$\mathtt{s},\mathtt{q}$	b^0,a^1,b^2,a^3,b^4	f^5	$\xrightarrow{a^5}$	g^3	b^0,a^1,b^2,a^3,b^4	$\mathtt{r}$
				$\downarrow$		
				$\hat{\mathtt{s}}$		
				$\mathtt{s}\vdash\mathtt{q}$		

Table 2: A 6-turn code for targeted queries. It is assumed that $\mathtt{s}\vdash\mathtt{q}$ and $\mathtt{q}\vdash\mathtt{r}$. The actual step of Bob proving queries is not shown.

$$\leq \quad \Lambda(p_s, p_r - p_q) + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right),$$

where the minimization is for all 6-turn codes for targeted queries (c.f. Table 2). Furthermore, if $\{(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q}),\vec{\kappa}(\mathtt{R}))_i\}_{i=1}^{|\mathcal{M}|}$ *are i.i.d. and* $\mathtt{R}\to\kappa(\mathtt{R})\to(\kappa(\mathtt{S}),\kappa(\mathtt{Q}))$ *then*

$$\Lambda(p_s,p_r-p_q) \quad\leq\quad \min_{\{f^i,g^i\}_{i=0}^5}\frac{1}{|\mathcal{M}|}E_{\mathtt{S},\mathtt{Q},\mathtt{R}}\left[\mathbf{len}(B^0)+\mathbf{len}(A^1)+\mathbf{len}(B^2)+\mathbf{len}(A^3)+\mathbf{len}(B^4)+\mathbf{len}(A^5)\right]$$

Proof. We start the proof using the beginning arguments from the proof of the lower bound of Theorem 6. We assume a 6-turn code for targeted queries $\{f^i,g^i\}_{i=0}^5$, and let $B^0,A^1,B^2,A^3,B^4,A^5\in\{0,1\}^*$ be the binary strings output by the encoders f^0,f^1,f^2,f^3,f^4,f^5 that are employed by Bob and Alice. Using the same type of argument that led to (47), we have

$$E_{\mathtt{S},\mathtt{Q},\mathtt{R}}\left[\mathbf{len}(B^0)+\mathbf{len}(A^1)+\mathbf{len}(B^2)+\mathbf{len}(A^3)+\mathbf{len}(B^4)+\mathbf{len}(A^5)\right]\geq H(\hat{\mathtt{S}}|\mathtt{R}).$$

We now continue by using a more complex variant of the arguments found in the proof of Theorem 5. We write

$$\begin{aligned}
H(\hat{\mathtt{S}}|\mathtt{R}) &\overset{(a)}{\geq} H(\vec{\kappa}(\hat{\mathtt{S}})|\mathtt{R})\\
&\overset{(b)}{=} H(\vec{\kappa}(\hat{\mathtt{S}})|\mathtt{R})-H(\vec{\kappa}(\hat{\mathtt{S}})|\mathtt{S},\mathtt{Q},\mathtt{R})\\
&\overset{(c)}{=} I(\mathtt{S},\mathtt{Q};\vec{\kappa}(\hat{\mathtt{S}})|\mathtt{R})\\
&\overset{(d)}{\geq} I(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q});\vec{\kappa}(\hat{\mathtt{S}})|\mathtt{R})\\
&\overset{(e)}{=} H(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})|\mathtt{R})-H(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})|\mathtt{R},\vec{\kappa}(\hat{\mathtt{S}}))\\
&\overset{(f)}{=} H(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})|\vec{\kappa}(\mathtt{R}))-H(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})|\mathtt{R},\vec{\kappa}(\hat{\mathtt{S}}))\\
&\overset{(g)}{\geq} H(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})|\vec{\kappa}(\mathtt{R}))-H(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q})|\vec{\kappa}(\mathtt{R}),\vec{\kappa}(\hat{\mathtt{S}}))\\
&\overset{(h)}{=} I(\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q});\vec{\kappa}(\hat{\mathtt{S}})|\vec{\kappa}(\mathtt{R}))
\end{aligned}$$

where (a) follows from the fact that deterministic functions of random quantities cannot increase entropy, (b) holds because in the second term, the conditioning is over all randomness, rendering the corresponding

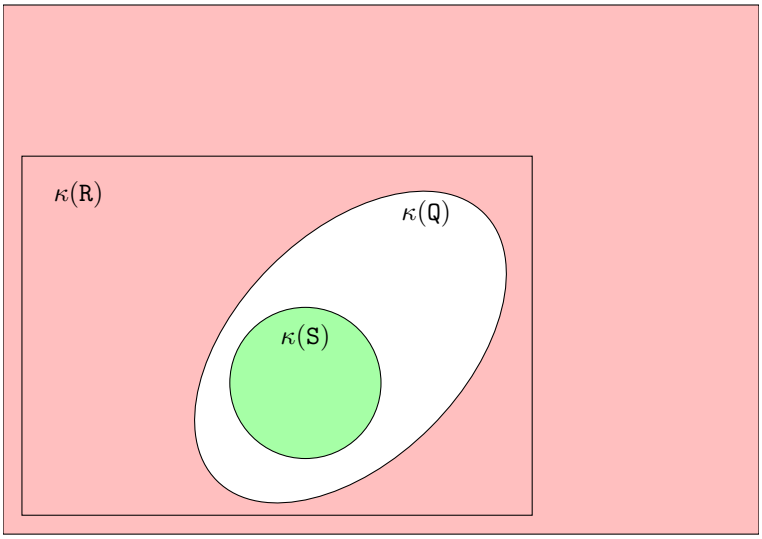


Figure 20: Reference diagram for the proof of the lower bound in Theorem 8

entropy equal to zero, (c) is the definition of mutual information, (d) follows from the data processing inequality, (e) expands mutual information back into a difference of entropies, (f) follows from the assumption that

$$\mathtt{R} \to \vec{\kappa}(\mathtt{R}) \to (\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}))$$

as well as the data processing inequality, (g) follows from the fact that a weaker conditioning cannot reduce conditional entropy, and (h) is from the definition of mutual information.

In reference to the proof of Theorem 5, recall that for a given $\mathtt{a} \in \mathcal{L}$, $\vec{\kappa}(\mathtt{a})$ can be regarded as an indicator vector for the kernel $\kappa(\mathtt{a})$. Additionally, we use the same definition for $\psi(\cdot,\cdot)$:

$$\psi(\mathtt{a},\mathtt{b}) = 0 \times (\underline{1} - \mathtt{b}) + 1 \times \mathtt{a} + 2 \times (\underline{1} - \mathtt{a}) \times \mathtt{b}.$$

where $(\underline{1} - \mathtt{a}) \times \mathtt{b}$ is interpreted to be product element-wise. We now define

$$\begin{aligned} X_1^{|\mathcal{M}|} &= \psi(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q})) \\ Z_1^{|\mathcal{M}|} &= \vec{\kappa}(\hat{\mathtt{S}}) \\ Y_1^{|\mathcal{M}|} &= \vec{\kappa}(\mathtt{R}). \end{aligned}$$

To interpret the first definition we refer the reader to Figure 20, which is a counterpart to Figure 16-b where we have added $\kappa(\mathtt{R})$. As in that figure, the red region is assigned "0", the green region "1" and the white region "2".

The next sequence of arguments is a simplified version of those in Theorem 7. For any kernel $k \in \{0,1\}^{|\mathcal{M}|}$:

$$\begin{aligned} P(X_i = 0 | Y_1^{|\mathcal{M}|} = k) &\overset{(a)}{=} P(X_i = 0 | Y_i = k_i) \\ &\overset{(b)}{=} P(\vec{\kappa}(\mathtt{Q})_i = 0 | Y_i = k_i) \\ &= 1 - P(\vec{\kappa}(\mathtt{Q})_i = 1 | Y_i = k_i) \\ &= 1 - \frac{P(\vec{\kappa}(\mathtt{Q})_i = 1) P(Y_i = k_i | \vec{\kappa}(\mathtt{Q})_i = 1)}{P(Y_i = k_i)} \\ &\overset{(c)}{=} \begin{cases} 1 & \text{if } k_i = 0 \\ 1 - p_q/p_r & \text{if } k_i = 1 \end{cases} \end{aligned} \tag{64}$$

where (a) follows from the Definition $X_i = \psi(\vec{\kappa}(\mathtt{S})_i, \vec{\kappa}(\mathtt{Q})_i)$ and the independence assumption, (b) follows from Lemma 6, and (c) follows from the identically distributed assumption and the (p_s, p_q, p_r)-law. Similarly, one can deduce

$$P(X_i = 1 | Y_1^{|\mathcal{M}|} = k) = P(X_i = 1 | Y_i = 1) = p_s/p_r \tag{65}$$

and therefore

$$P(X_i = 2 | Y_1^{|\mathcal{M}|} = k) = P(X_i = 2 | Y_i = 1) = p_q/p_r - p_s/p_r \tag{66}$$

One consequence of this analysis is that, conditional on $Y_1^{|\mathcal{M}|} = k$, the $\{X_i\}$ are independent although not generally identically distributed. Recall we are assuming that whenever using the 6-turn code for targeted queries, $\mathtt{S} \vdash \hat{\mathtt{S}}$ and that $\hat{\mathtt{S}} \vdash \mathtt{Q}$ and recall the definition of the distortion metric ρ from Equations (38) and (39). Then for all $1 \leq i \leq |\mathcal{M}|$,

$$\begin{aligned} P(Z_i = 0 | X_i = 0, Y_1^{|\mathcal{M}|} = k) &= 1 \\ P(Z_i = 1 | X_i = 1, Y_1^{|\mathcal{M}|} = k) &= 1 \end{aligned}$$

and therefore as a consequence, for all $1 \leq i \leq |\mathcal{M}|$, under the conditioning $Y_1^{|\mathcal{M}|} = k$, with probability 1,

$$\rho(X_i, Z_i) = 0. \tag{67}$$

Following in the footsteps of the proof of the lower bound for Theorem 5, we write:

$$\begin{aligned} I(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}); \vec{\kappa}(\hat{\mathtt{S}}) | \vec{\kappa}(\mathtt{R})) &= I(\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q}); Z_1^{|\mathcal{M}|} | Y_1^{|\mathcal{M}|}) \\ &\overset{(i)}{\geq} I(X_1, \ldots, X_{|\mathcal{M}|}; Z_1, \ldots, Z_{|\mathcal{M}|} | Y_1^{|\mathcal{M}|}) \\ &\overset{(j)}{\geq} \sum_k P(Y_1^{|\mathcal{M}|} = k) I(X_1, \ldots, X_{|\mathcal{M}|}; Z_1, \ldots, Z_{|\mathcal{M}|} | Y_1^{|\mathcal{M}|} = k). \end{aligned}$$

The step (i) follows from the fact that $X_1^{|\mathcal{M}|}$ is a deterministic function of $\vec{\kappa}(\mathtt{S})$ and $\vec{\kappa}(\mathtt{Q})$, and thus conditional on $Y_1^{|\mathcal{M}|}$, the following Markov chain holds:

$$X_1^{|\mathcal{M}|} \to (\vec{\kappa}(\mathtt{S}), \vec{\kappa}(\mathtt{Q})) \to Z_1^{|\mathcal{M}|}.$$

Step (j) follows from the definition of conditional mutual information. We then focus on each individual term:

$$\begin{aligned} & I(X_1, \ldots, X_{|\mathcal{M}|}; Z_1, \ldots, Z_{|\mathcal{M}|} | Y_1^{|\mathcal{M}|} = k) \\ &= H(X_1, \ldots, X_{|\mathcal{M}|} | Y_1^{|\mathcal{M}|} = k) - H(X_1, \ldots, X_{|\mathcal{M}|} | Z_1, \ldots, Z_{|\mathcal{M}|}, Y_1^{|\mathcal{M}|} = k) \\ &\overset{(k)}{=} \sum_{i=1}^{|\mathcal{M}|} H(X_i | X_1^{i-1}, Y_1^{|\mathcal{M}|} = k) - H(X_i | Z_1, \ldots, Z_{|\mathcal{M}|}, X_1^{i-1}, Y_1^{|\mathcal{M}|} = k) \\ &\overset{(l)}{=} \sum_{i=1}^{|\mathcal{M}|} H(X_i | Y_i = k_i) - H(X_i | Z_1, \ldots, Z_{|\mathcal{M}|}, X_1^{i-1}, Y_1^{|\mathcal{M}|} = k) \\ &\overset{(m)}{\geq} \sum_{i=1}^{|\mathcal{M}|} H(X_i | Y_i = k_i) - H(X_i | Z_i, Y_i = k_i) \\ &= \sum_{i=1}^{|\mathcal{M}|} I(X_i; Z_i | Y_i = k_i) \end{aligned}$$

where (k) follows from the chain rule for entropy, (l) follows from the fact that the $|\mathcal{M}|$ tuples $\{(X,Y)_i\}$ are statistically independent as per the Theorem's assumption. Finally (m) follows from the fact that eliminating conditioning random variables cannot decrease entropy. We continue by splitting the last summation in two:

$$\begin{aligned}\sum_{i=1}^{|\mathcal{M}|} I(X_i;Z_i|Y_i=k_i) &= \sum_{i:k_i=0} I(X_i;Z_i|Y_i=0) + \sum_{i:k_i=1} I(X_i;Z_i|Y_i=1) \\ &= \sum_{i:k_i=1} I(X_i;Z_i|Y_i=1) \end{aligned} \tag{68}$$

where we have made the observation that conditioning on $Y_i=0$, X_i is constant (and equal to zero) and therefore the corresponding mutual information is zero.

As before, to complete the lower bound, we note that (68) is an averaging of mutual informations where the marginal for X_i is identical for all $\{i:k_i=1\}$, as given by Equations (64,65,66), but the conditionals $Q_{Z_i|X_i,Y_i=1}$ are in general different. Assume temporarily that $|k|=|\{i:k_i=1\}|\geq 1$. Define a conditional distribution by averaging all those conditionals:

$$Q_{Z'|X'}(z|x) \triangleq \frac{1}{|k|}\sum_{\{i:k_i=1\}} Q_{Z_i|X_i,Y_i}(z|x,y=1). \tag{69}$$

In reference to Equations (64,65,66), let X' be distributed according to $\{1-p_q/p_r, p_s/p_r, p_q/p_r-p_s/p_r\}$ and let the joint X',Z' be defined by having Z' be the effect of passing X' through the channel defined by (69). Because of (67), it is the case that

$$E_{X',Z'}\left[\rho(X',Z')\right]=0.$$

Temporarily imagine $I(X_i;Z_i|Y_i=1)$ as a function of two distributions: a distribution on X_i given $Y_i=1$ and a conditional distribution $Q_{Z_i|X_i,Y_i}(z|x,y=1)$. Given our earlier arguments, in (68), the distribution of X_i given $Y_i=1$ is identical for all those $\{i:k_i=1\}$, and precisely equal to that of X'. Furthermore, it is known that mutual information is convex $\cup$ if one keeps the marginal distribution of X fixed as one varies the conditional distribution of Z given X and therefore the following bound holds:

$$\sum_{\{i:k_i=1\}} I(X_i;Z_i|Y_i=1) \geq |k| I(X';Z') \geq |k| \min_{P(X,Z)\in\mathcal{D}} I(X;Z) \tag{70}$$

where the domain $\mathcal{D}$ for the minimization is defined by joint distributions for X,Z with $X\sim(1-p_q/p_r, p_s/p_r, p_q/p_r-p_s/p_r)$ and $E[\rho(X,Z)]=0$. Also note that even though we assumed $|k|\geq 1$, the bound (70) is trivially true for $|k|=0$ due to the nonnegativity of mutual information. Such minimization, which can be obtained using standard variational methods, results in the following expression

$$|k|\Lambda(p_s/p_r, 1-p_q/p_r).$$

We now bring back the distribution for k, and write

$$\sum_k P(Y_1^{|\mathcal{M}|}=k)|k|\Lambda(p_s/p_r,1-p_q/p_r) = p_r\Lambda(p_s/p_r,1-p_q/p_r) = \Lambda(p_s,p_r-p_q)$$

where the last equality follows from basic properties of Λ (Lemma 2). This concludes the proof of the lower bound.

The proof of the upper bound is adapted from the proof of Theorem 6. We stress that this result will hold under the very general assumption that $\mathtt{S},\mathtt{Q},\mathtt{R}$ follow a (p_s,p_q,p_r)-law with $p_s\leq p_q\leq p_r$. As before, we establish the precise domain/image of the encoder and decoders for our 6-turn mode:

$$\begin{aligned} f^0:\mathcal{L} &\to \{0,1\}^*, \quad g^0:\{0,1\}^*\to\mathbb{Z}, \\ f^1:\mathcal{L}^2 &\to \{0,1\}^*, \quad g^1:\{0,1\}^*\to\mathbb{Z}^2, \\ f^2:\mathbb{Z} &\to \{0,1\}^*, \quad g^2:\{0,1\}^*\to\mathbb{Z}, \end{aligned}$$

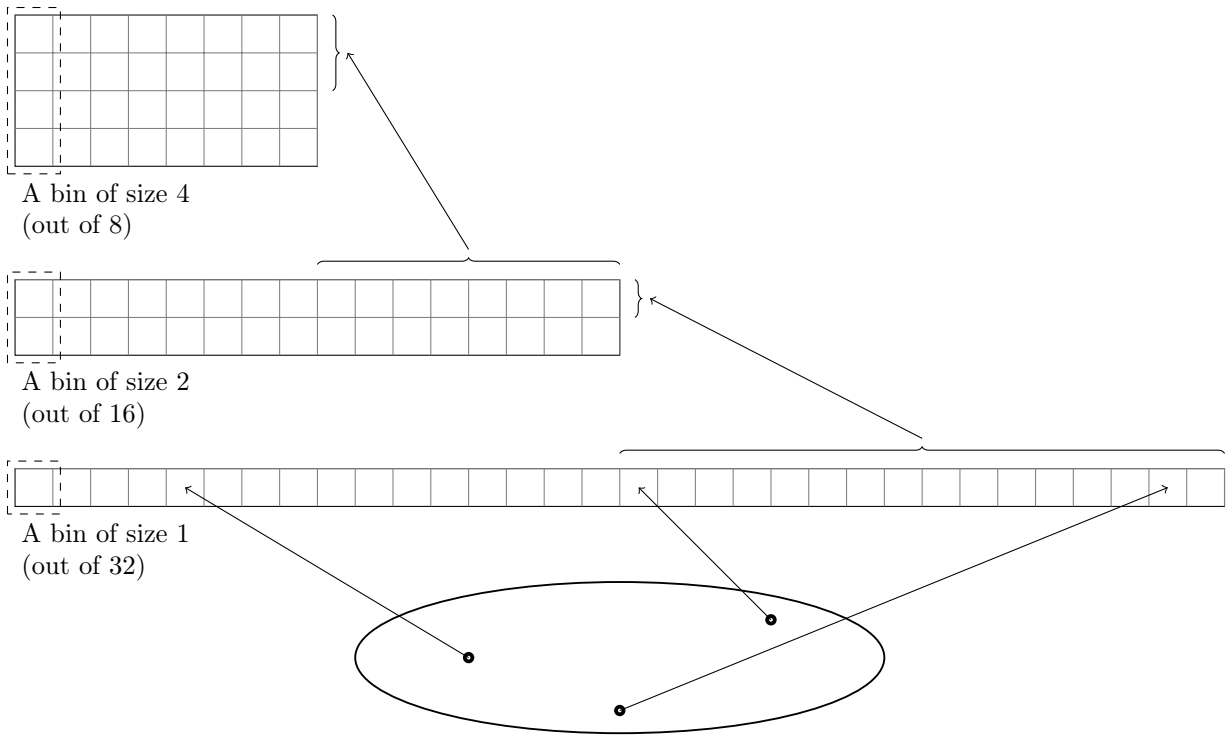


Figure 21: An adaptive rate hash function. An initial hash function, depicted at the bottom, can be used to create a family of hash functions by stacking halves iteratively. A total of three hash functions are shown; an exemplary bin for each is illustrated using a dashed box.

$$\begin{aligned} f^3 : \mathcal{L}^2 \times \mathbb{Z}^3 &\rightarrow \{0,1\}^*, \quad g^3 : \{0,1\}^* \times \mathcal{L} \rightarrow \mathcal{L} \times \{\text{success}, \text{failure}\}, \\ f^4 : \{\text{success}, \text{failure}\} &\rightarrow \{0,1\}, \quad g^4 : \{0,1\} \rightarrow \{\text{success}, \text{failure}\}, \\ f^5 : \mathcal{L} &\rightarrow \{0,1\}^*, \quad g^5 : \{0,1\}^* \times \{\text{success}, \text{failure}\} \times \mathcal{L} \rightarrow \mathcal{L}. \end{aligned}$$

In reference to the proof of Theorem 6, a key step is the definition of a hash function. For the present result we will require an *adaptive rate* hash function which allows the number of bins to be flexibly changed in response to specific requirements. We refer the reader to Figure 21 for this construction. A hash function with 2^i bins for some integer i, requiring i bits to specify a specific bin, can also behave as a hash function with 2^{i-1} bins by stacking the second half of the bins for the former on top of the left half, creating half the bins with twice the number of elements and thus requiring one bit less to fully specify a bin. This stacking procedure can be continued, creating a family of hash functions with fewer bit requirements from a single hash function. Given any desired bit rate, provided that it is equal or lower than the maximum bit rate that the hash function supports, the adaptive hash function will be able match that desired bit rate within at most 1 bit of inefficiency. Define

$$T(w) = \{c \in \{0,1\}^{|\mathcal{M}|} : \ |c| = w\}.$$

Alice and Bob first interact to agree on the following functions for integers $0 < s \leq q \leq r$

$$\begin{aligned} \text{weight}(s,q,r) &= \left\lceil \frac{rs}{s+r-q} \right\rceil \\ \text{code\_size}(s,q,r) &= \left\lceil \log_2 |\mathcal{M}| \left(\begin{array}{c} |\mathcal{M}| \\ \text{weight}(s,q,r) \end{array} \right) / \left(\begin{array}{c} q-s \\ \text{weight}(s,q,r)-s \end{array} \right) \right\rceil \end{aligned} \tag{71}$$

as well as

- a set of codewords $\text{code}(s,q,r) \subseteq T(\text{weight}(s,q,r))$, indexed using integers $1,\ldots,\text{code\_size}(s,q,r)$
- an adaptive rate hash function $\text{bin}_{s,q,r,\text{rate}} : \text{code}(s,q,r) \to \mathbb{Z}$.

From this definition it should be clear that a codeword is an element of $\{0,1\}^{|M|}$. In our paper, a codeword is equivalent to a kernel, since it can be seen as the indicator vector of the kernel. Thus a code can be seen as a set of kernels. How these are constructed will be described shortly. For two vectors a, b of the same length, we will write $a \leq b$ whenever $a_i \leq b_i$ for every i. The protocol is then as follows:

1. The receiver sends $|\vec{\kappa}(\mathtt{R})|$ to the sender, which she decodes. This step defines f^0 and g^0.
2. The sender sends $|\vec{\kappa}(\mathtt{S})|$ and $|\vec{\kappa}(\mathtt{Q})|$, which are decoded at the receiver side. This defines f^1 and g^1 .
3. Both sender and receiver compute a target weight and a code book size using the functions (71) evaluated on $|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|$, and $|\vec{\kappa}(\mathtt{R})|$.
4. The receiver calculates the average bit rate it wants from the receiver using the expression

$$\text{RATE} = \frac{1}{|\mathcal{M}|} \log_2 \lceil | \{a \in \text{code}(|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|, |\vec{\kappa}(\mathtt{R})|) : a \leq \vec{\kappa}(\mathtt{R})\} | \rceil + \frac{\log_2 |\mathcal{M}|}{|\mathcal{M}|}. \tag{72}$$

   Note the "round up" operator inside of the logarithm. This implies, in particular, that $2^{|\mathcal{M}|\text{RATE}}$ will be a proper power of 2, and hence the resulting rate can be used in the context of the adaptive rate hash function. The integer $|\mathcal{M}|\text{RATE}$ is sent back to the sender, defining f^2 and g^2. At this point they both agree on target weight, code book size and target rate to be used in the adaptive rate hash function.
5. The sender attempts to find $A \in \text{code}(|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|, |\vec{\kappa}(\mathtt{R})|)$ with the property that

$$\vec{\kappa}(\mathtt{S}) \leq A \leq \vec{\kappa}(\mathtt{Q}). \tag{73}$$

   If this fails, then the sender signals the failure to the receiver and executes step 9. If it succeeds, the sender signals success and sends to the receiver the bin to which A was mapped, for the hash function with the agreed upon rate.
6. The receiver attempts to retrieve A using the bin index, its knowledge of $\vec{\kappa}(\mathtt{R})$ and the algorithm described below. The outcome of this attempt results in an element of $\mathcal{L}$ and a "success" or a default, dummy element from $\mathcal{L}$ and a "failure". Together with the previous step, this completes f^3, g^3.
7. The success/failure of the attempt is signaled back to the sender using a single bit. This defines both f^4 and g^4.
8. If successful, the sender has nothing to do anymore, as the receiver expects no further communication. The receiver simply outputs the element from $\mathcal{L}$ computed by g^3, designating it as the output $\hat{\mathtt{S}}$, partly defining g^5.
9. If unsuccessful, the sender sends $\vec{\kappa}(\mathtt{S})$ as a binary vector of length $|\mathcal{M}|$, which is decoded by the receiver and becomes $\hat{\mathtt{S}}$, the designated output of the protocol, completing the definition of f^5 and g^5.

The algorithm guarantees that the receiver at the end will have in possession enough information to reproduce a message $\hat{\mathtt{S}}$ that can be used to prove $\mathtt{Q}$. We now demonstrate that the normalized total bidirectional cost in bits is asymptotically $\Lambda(p_s, p_r - p_q)$.

Instead of constructing a specific $\text{code}(s,q,r)$ and hash function $\text{bin}_{s,q,r,\text{rate}}$ we will define a probability distribution over each of these, evaluate the corresponding expected performance, and demonstrate that in average the performance as desired, implying the existence of deterministic versions of these with at least the same performance. We shall refer to the random constructions as $\text{CODE}(s,q,r)$ and $\text{BIN}_{s,q,r,\text{rate}}$, respectively.

We construct $\text{CODE}(s,q,r)$ by drawing uniformly and independently at random from $T(\text{weight}(s,q,r))$ a total of $\text{code\_size}(s,q,r)$ codewords. We construct $\text{BIN}_{s,q,r,\text{rate}}$ by mapping every element of $\text{CODE}(s,q,r)$

uniformly and independently at random to an integer in the range $\{1, \ldots, 2^{\lceil \log_2 |\mathcal{M}| \rceil}\}$ (each representing a bin), and then iteratively stacking halves of these bins to produce an adaptive rate hash function, as illustrated by Figure 21.

For brevity, define

$$W = \text{weight}(|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|, |\vec{\kappa}(\mathtt{R})|)$$
$$\mathcal{C} = \text{CODE}(|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|, |\vec{\kappa}(\mathtt{R})|).$$

In the algorithm above, there are two key failure events for which we want to estimate probabilities. First we focus on the probability of a failure in step 5. The probability that any one element of $\mathcal{C}$ satisfies (73) is

$$P_{succ} = \frac{\left(\begin{array}{c} |\vec{\kappa}(\mathtt{Q})| - |\vec{\kappa}(\mathtt{S})| \\ W - |\vec{\kappa}(\mathtt{S})| \end{array} \right)}{\left(\begin{array}{c} |\mathcal{M}| \\ W \end{array} \right)}.$$

The probability that none of the elements of $\mathcal{C}$ meet the condition (73) is then

$$\begin{aligned} (1 - P_{succ})^{|\mathcal{C}|} &\leq \exp\left(-P_{succ}|\mathcal{C}|\right) \\ &\leq \frac{1}{|\mathcal{M}|} \end{aligned}$$

where we used the definition (71) to obtain the latter. Now we focus on the failure probability in step 6. Define

$$\text{hypotheses}(\mathtt{S}, \mathtt{Q}, \mathtt{R}) \stackrel{\Delta}{=} \left\{\alpha \in \mathcal{C} : \alpha \leq \vec{\kappa}(\mathtt{R}), \text{BIN}_{|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|, |\vec{\kappa}(\mathtt{R})|}(\alpha) = \text{BIN}_{|\vec{\kappa}(\mathtt{S})|, |\vec{\kappa}(\mathtt{Q})|, |\vec{\kappa}(\mathtt{R})|}(A)\right\}. \tag{74}$$

To estimate the failure probability, we upper bound the probability of this event:

$$P\left([|\text{hypotheses}(\mathtt{S}, \mathtt{Q}, \mathtt{R})| \geq 2]\right).$$

Let I_α be equal to 1 if $\alpha \in$ hypotheses$(\mathtt{S}, \mathtt{Q}, \mathtt{R})$ and 0 otherwise. Then we can upper bound the error probability as

$$P\left([|\text{hypotheses}(\mathtt{S}, \mathtt{Q}, \mathtt{R})| \geq 2]\right) = P\left(\sum_\alpha I_\alpha \geq 2\right).$$

We note that when conditioning on $\mathtt{S}, \mathtt{Q}, \mathtt{R}, A, \mathcal{C}$, the only randomness that remains in (74) is that of the randomness of bin assignments. We then write

$$\begin{aligned} P\left(\sum_\alpha I_\alpha \geq 2 | \mathtt{S}, \mathtt{Q}, \mathtt{R}, A, \mathcal{C}\right) &= 1 - P\left(\sum_{\alpha \in \mathcal{C}, \alpha \neq A, \alpha \leq \vec{\kappa}(\mathtt{R})} I_\alpha = 0 | \mathtt{S}, \mathtt{Q}, \mathtt{R}, A, \mathcal{C}\right) & (75) \\ &= 1 - \Pi_{\alpha \in \mathcal{C}, \alpha \neq A, \alpha \leq \vec{\kappa}(\mathtt{R})} P(I_\alpha = 0 | \mathtt{S}, \mathtt{Q}, \mathtt{R}, A, \mathcal{C}) & (76) \\ &= 1 - \Pi_{\alpha \in \mathcal{C}, \alpha \neq A, \alpha \leq \vec{\kappa}(\mathtt{R})} \left(1 - 2^{-|\mathcal{M}|\text{RATE}}\right) \\ &\leq 1 - \Pi_{\alpha \in \mathcal{C}, \alpha \leq \vec{\kappa}(\mathtt{R})} \left(1 - 2^{-|\mathcal{M}|\text{RATE}}\right) \\ &= 1 - \left(1 - 2^{-|\mathcal{M}|\text{RATE}}\right)^{|\{\alpha \in \mathcal{C}, \alpha \leq \vec{\kappa}(\mathtt{R})\}|} \\ &\leq 2^{-|\mathcal{M}|\text{RATE}} |\{\alpha \in \mathcal{C}, \alpha \leq \vec{\kappa}(\mathtt{R})\}| \\ &= 2^{-|\mathcal{M}|\text{RATE} + \log_2 |\{\alpha \in \mathcal{C}, \alpha \leq \vec{\kappa}(\mathtt{R})\}|} \\ &= \frac{1}{|\mathcal{M}|}. & (77) \end{aligned}$$

As in the proof of Theorem 6, the derivation above is the essence of the theorem. The most delicate step is the one that leads from (75) to (76), where independence of the $\{I_\alpha\} \setminus \{A\}$ events under the given conditioning

is invoked to rewrite the probability of the event as a product of probabilities. The core reason why this independence holds is because under the given conditioning, the bins to which any one α is mapped are chosen independently from each other over the set of possible bins. Since these steps are very similar to those in Theorem 3, we skip most explanations and simply point out that the final expression (77) comes from the definition (72). In summary, we now have that

$$P\left([|\text{hypotheses}(\mathtt{S},\mathtt{Q},\mathtt{R})| \geq 2]\right) \quad \leq \quad \frac{1}{|\mathcal{M}|}.$$

We can now account for all of the bits sent in the protocol. The biggest contribution to the bit rate is that of step 5:

$$\begin{aligned} E\left[\text{RATE}\right] &\leq E\left[\frac{1}{|\mathcal{M}|}\log_2 |\left\{A \in \text{CODE}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|) : A \leq \vec{\kappa}(\mathtt{R})\right\}|\right] + \frac{1}{|\mathcal{M}|} + \frac{\log_2|\mathcal{M}|}{|\mathcal{M}|} \\ &= E\left[E\left[\frac{1}{|\mathcal{M}|}\log_2 |\left\{A \in \text{CODE}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|) : A \leq \vec{\kappa}(\mathtt{R})\right\}| \,\middle|\, \mathtt{S},\mathtt{Q},\mathtt{R}\right]\right] + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right) \\ &\leq E\left[\frac{1}{|\mathcal{M}|}\log_2 E\left[|\left\{A \in \text{CODE}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|) : A \leq \vec{\kappa}(\mathtt{R})\right\}| \,||\, \mathtt{S},\mathtt{Q},\mathtt{R}\right]\right] + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right). \end{aligned}$$

We recall that the way $\text{CODE}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|)$ is constructed is by choosing uniformly at random

$$\text{code\_size}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|) \tag{78}$$

elements from $T(W)$. Next note that

$$|\{A : A \leq \vec{\kappa}(\mathtt{R})\} \cap T(W)| = \left(\begin{array}{c} |\vec{\kappa}(\mathtt{R})| \\ W \end{array}\right). \tag{79}$$

Any one element of $T(W)$ has a probability of

$$\left(\begin{array}{c} |\mathcal{M}| \\ W \end{array}\right)^{-1} \tag{80}$$

The average number of elements of $\{A : A \leq \vec{\kappa}(\mathtt{R})\}$ chosen to be part of $\text{CODE}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|)$, conditional on $\vec{\kappa}(\mathtt{S}),\vec{\kappa}(\mathtt{Q}),\vec{\kappa}(\mathtt{R})$ is then given by the product of (78), (79) and (80):

$$\begin{aligned} &\log_2 E\left[|\left\{A \in \text{CODE}(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{Q})|,|\vec{\kappa}(\mathtt{R})|) : A \leq \vec{\kappa}(\mathtt{R})\right\}| \,||\, \mathtt{S},\mathtt{Q},\mathtt{R}\right] \\ &= \log_2\left(\log_2|\mathcal{M}|\left(\begin{array}{c} |\vec{\kappa}(\mathtt{R})| \\ W \end{array}\right) / \left(\begin{array}{c} |\vec{\kappa}(\mathtt{Q})| - |\vec{\kappa}(\mathtt{S})| \\ W - |\vec{\kappa}(\mathtt{S})| \end{array}\right)\right) \\ &\leq (|\vec{\kappa}(\mathtt{S})| + |\vec{\kappa}(\mathtt{R})| - |\vec{\kappa}(\mathtt{Q})|)H_{\text{bin}}\left(|\vec{\kappa}(\mathtt{S})|/(|\vec{\kappa}(\mathtt{S})| + |\vec{\kappa}(\mathtt{R})| - |\vec{\kappa}(\mathtt{Q})|)\right) + \log_2\log_2|\mathcal{M}| \\ &= \Lambda(|\vec{\kappa}(\mathtt{S})|,|\vec{\kappa}(\mathtt{R})| - |\vec{\kappa}(\mathtt{Q})|) + \log_2\log_2|\mathcal{M}|. \end{aligned}$$

Further taking the expectation over the remaining randomness, using the concavity $\cap$ of the Λ function (see Lemma 2), and normalizing, we obtain the estimate

$$E\left[\text{RATE}\right] \leq \Lambda(p_s, p_r - p_q) + O\left(\frac{\log_2|\mathcal{M}|}{|\mathcal{M}|}\right).$$

The cost of steps 1, 2, 4 can be verified to be $O(|\mathcal{M}|^{-1}\log|\mathcal{M}|)$ by using δ Elias coding. Finally, in the case of failure when trying to find an A meeting (73) the cost is $O(|\mathcal{M}|^{-1})$ given (74). □

6 Formal logical underpinnings

In this section we formally define what we mean by a logic, something that we only introduced informally in the description preceding Definition 1 of a Logic System. To make the definition of a logic rigorous, we

introduce several additional concepts, including: what a structure is (Definition 9), what it means for a structure to model a set of logic sentences (Definition 15), what it means for a logic sentence to be true (Definition 16), the notions of a logic being sound and complete (Definitions 21 and 23), and several subtle variations on these latter two notions (Definitions 22, 24, and 25). Two key results of this section are Theorem 9, which shows that condition (6) of Definition 1 is equivalent to strong soundness and a type of strong completeness of the underlying logic, and Lemma 7, which gives general conditions under which a Logic System can be guaranteed to satisfy conditions 1-3 of Definition 2, and hence be deemed a *proper* Logic System.

With the aim of developing as all-encompassing a notion of what constitutes a logic as possible, we will establish a set of definitions that set us up well for First-Order Logic and logics that "extend" First-Order Logic, such as second and higher-order logics [73][2]. Towards the end of our development (Subsection 6.3), we will see that one can fit Propositional Logic and other quantifier-free logics into this model as well. Among other things, at the end of this section we will be able to conclude that all the results of this paper apply to Propositional Logic on a fixed number of variables as well as the First-Order Logic of structures of fixed finite sizes.

6.1 The notion of a Logic

By a *logic*, $L = (\tau, \$, \mathcal{T}, \sigma, \mathcal{P})$, we mean (i) a vocabulary, or set of symbols, τ, (ii) a syntax, $\$$, or set of rules for combining the elements of τ together to form "well-formed sentences", (iii) a definition of truth, $\mathcal{T}$, or set of rules describing how truth propagates through logical symbols, such as $\wedge, \vee, \neg, \forall$ and $\exists$, that are part of the vocabulary, (iv) a set of axioms, σ (with, possibly, $\sigma = \varnothing$), or sentences that are assumed to be true without proof, and finally (v) a proof system, $\mathcal{P}$, comprising a set of rules of inference, for establishing when certain well-formed sentences follow from other well-formed sentences. As in the definition of a Logic System, we refer to the set of well-formed sentences over τ, given the syntax of L, by $\mathcal{L}$. We denote the entailment operator associated with the proof system $\mathcal{P}$ by $\vdash$ (or by $\vdash_{\mathcal{P}}$ if the associated proof system is not entirely clear). The meta-mathematical expression $\mathtt{s} \vdash \mathtt{t}$ means that the sentence $\mathtt{t}$ can be proved from $\sigma \cup \{\mathtt{s}\}$ using $\mathcal{P}$. The symbol $\vdash$ is not part of any logical vocabulary and hence the expression $\mathtt{s} \vdash \mathtt{t}$ is not a sentence of any logical language – hence our assertion that the expression $\mathtt{s} \vdash \mathtt{t}$ is a "meta-mathematical expression". As noted in Subsection 2.1, we can apply the entailment operator, $\vdash$, to sets of sentences as well as to individual sentences.

We described the vocabulary and syntax of classical Propositional Logic in Subsection 2.1. A second important logic that we will consider is First-Order Logic. Although the vocabulary of First-Order Logic includes the logical connectives and grouping parentheses of Propositional Logic, the vocabularies of these two logics are otherwise quite distinct. The vocabulary of First-Order Logic does *not* include propositional variables, but adds the universal and existential quantifiers, $\forall$ and $\exists$, as well as a countable number of variables, x_i, that can be associated with the quantifiers. The vocabulary of First-Order Logic also includes function and relation symbols of different arities, $f_i^{a_i}$ and $R_i^{\alpha_i}$, as well as constant symbols, c_i. As is customary, we will always assume that there is a distinguished binary equality relation symbol $=(\cdot,\cdot)$, and per convention, write $x = y$ for $=(x, y)$ and $x \neq y$ for $\neg=(x, y)$. A vocabulary that includes the logical connectives of Propositional Logic, parentheses, a countable number of variable symbols, the quantifiers $\forall$ and $\exists$, as well as a particular set of function, relation and constant symbols (the relations symbols necessarily including $=(\cdot,\cdot)$), is called a **First-Order vocabulary**. When it is understood that a vocabulary, τ, is First-Order, one typically writes $\tau = (\{f_i^{a_i}\}, \{R_i^{\alpha_i}\}, \{c_i\})$, specifying just the non-logical symbols that distinguish τ from other First-Order vocabularies. One of the simplest First-Order vocabularies is the vocabulary of directed graphs. In addition to the built-in equality symbol, this vocabulary consists of a single binary relation symbol, $E(\cdot,\cdot)$. In this case we would write $\tau = (\varnothing, \{E(\cdot,\cdot)\}, \varnothing)$, or, equivalently, $\tau = (\varnothing, \{E^2\}, \varnothing)$. In some cases when we work with graphs, we may be considering, say, shortest path, or connectivity questions between specified nodes. In this case it may be convenient to expand the vocabulary to include constants, which are typically denoted s and t (for "source" and "target"). The vocabulary would then be denoted $\tau = (\varnothing, \{E(\cdot,\cdot)\}, \{s, t\})$.

[2]Logics that we will, however, not consider formally, other than to say that they extend First-Order Logic in the sense of Definition 11.

The concept of a "variable" in First-Order Logic is completely different from the concept of a variable in Propositional Logic, as is illustrated by their vastly different syntax, or rules for sentence formation. To understand the syntax of First-Order Logic, we build up a set of definitions.

Definition 4 *A **term** is defined inductively as follows:*

- *Each variable or constant symbol is a term;*
- *If f^m is an m-ary function symbol, and $t_1, \ldots, t_m$ are terms, then $f^m(t_1, \ldots, t_m)$ is also a term.*

Definition 5 *A **primitive formula**[3] in First-Order Logic is a string of symbols of the form $R^m(t_1, \ldots, t_m)$ where R^m is an m-ary relation symbol (possibly the binary equality symbol $=(t_1, t_2)$) and $t_1, \ldots, t_m$ are terms. Analogously, we refer to a primitive formula that does not contain variables as a **primitive sentence**.*

Definition 6 *A string of symbols is a **formula** of First-Order Logic if it is either a primitive formula or can be constructed from primitive formulas by repeated application of the following rules:*

R1. If ϕ is a formula, then so is $\neg\phi$;

R2. If ϕ and ψ are formulas, then so are $\phi \vee \psi$ and $\phi \wedge \psi$;

R3. If ϕ is a formula, then so are $\exists x\phi$ and $\forall x\phi$.

The same symbol in a First-Order vocabulary can appear in multiple positions within a given formula. We refer to each appearance of the same symbol as an **occurrence** of the symbol.

Definition 7 *We recursively define the notion of an occurrence of a variable being either **free** (equivalently, **unbound**) or **bound** to a quantifier within a formula. If an occurrence of x in a formula ϕ is free, the variable x is also said to **occur freely** in ϕ. In a formula ϕ without quantifiers, all occurrences of all variables appearing in ϕ are considered to be free. Then we have the following:*

R1. If a variable occurrs freely in ϕ then it also occurrs freely in $\neg\phi$;

R2. If a variable occurrs freely in ϕ or ψ, then it occurrs freely in $\phi \vee \psi$ and $\phi \wedge \psi$;

R3. If a variable x occurrs freely in ϕ, then it is bound in the formula $\exists x\phi$ and in the formula $\forall x\phi$. If the variable x does not occur in ϕ, we also say that x is bound in the formula $\exists x\phi$ and in the formula $\forall x\phi$.

Given a logic L, with language $\mathcal{L}$, if ϕ is a formula with free variables compatible with the syntax of L, with some abuse of notation we shall sometimes write $\phi \in \mathcal{L}$. A formula can have both free and bound occurrences of the same variable. Suppose the variable x appears freely in both the formulas ϕ and ψ. Then there are both free and bound occurrences of x in the formula $\phi \vee \exists x\psi$.

Definition 8 *A **sentence** in First-Order Logic is a formula having no free variables.*

Remark 1 *Now that we have the full definitions of what it means for a string of logical symbols to be either a formula or a sentence, we can say that a primitive formula (respectively, primitive sentence) is a formula (respectively, sentence) that contains neither logical connectives nor quantifiers.*

Definitions 4 through 8 provide a specification of the syntax of First-Order Logic. Since it is conceivable for other logics beside First-Order Logic to have this syntax, we call this syntax **First-Order syntax**. There are of course many equivalent ways to specify the same syntax – meaning that starting with the same vocabularies one would apply the somewhat different rules and arrive at the same language. We will loosely refer to any such set of rules as First-Order syntax. A **First-Order language** is a language obtained from a First-Order vocabulary by applying First-Order syntax.

Truth in every logic is just defined for the sentences of the logic. To understand how truth is defined for a logic, we will first need to make an excursion into the branch of logic known as *model theory*. We do so in the next subsection.

[3]Sometimes referred to as an atomic formula.

6.2 Models of logic sentences and Truth in Models

Definition 9 *In mathematical logic, a* ***structure*** $\mu = (\mathcal{U}, \{f_i^{a_i}\}, \{R_i^{\alpha_i}\}, \{c_i\})$ *consists of a* non-empty *set* $\mathcal{U}$ *(sometimes called a "universe" or "universe of discourse"), together with collections (possibly empty) of functions* $\{f_i^{a_i}\}$ *and relations* $\{R_i^{\alpha_i}\}$*, each of finite arity defined on the elements of* $\mathcal{U}$ *(each function* $f_i^{a_i}$ *also having its image in* $\mathcal{U}$ *so that* $f_i^{a_i} : \mathcal{U}^{a_i} \to \mathcal{U}$*, and each relation* $R_i^{\alpha_i}$ *defining a subset of* $\mathcal{U}^{\alpha_i}$*, with* $R_i^{\alpha_i}()$ *returning True for a given* α_i*-tuple of elements of* $\mathcal{U}$ *if the particular* α_i*-tuple is in the given subset, and returning False otherwise), and again, optionally, a collection of constants* $\{c_i\}$*, which may be thought of as functions of* 0*-arity picking out individual elements of* $\mathcal{U}$*. When we talk about the "elements" of the structure* μ*, we mean the elements of the set* $\mathcal{U}$*.*

In First-Order Logic every structure is assumed to include an equality relation which is associated with the equality symbol, $=(\cdot,\cdot)$, and is required to behave in the accustomed manner so that two elements of the structure are equal iff they are the same element. An example of a structure in First-Order Logic is a directed graph viewed as a set of nodes together with (the equality relation and) a single binary relation, $E(\cdot,\cdot)$, defined on the nodes, which is true iff there is a directed edge going from the first node to the second node. A graph may also have constants defined, such as a specially designated "source" and "terminal" nodes, typically denoted s and t, respectively. We say that a structure "interprets" the function, relation and constant symbols of a given logical vocabulary if it contains functions and relations of the same arity as each of the function and relation symbols, and has elements, that we call constants, that it associates with each of the constant symbols in the vocabulary.

Definition 10 *Given a logical vocabulary* τ *that includes some number of function, relation, and constant symbols, by a* τ***-structure*** *one means a structure that interprets each of the function, relation and constant symbols in* τ *via concrete functions and relations of the same arity and constants.*

We are now going to take up the subject of defining *truth* in structures, which will become the basis of what we mean, more generally, by *truth* in a logic. Recall that every logic must contain a definition of truth, $\mathcal{T}$, which comprises a set of rules that describe how truth propagates through the logical symbols, such as $\vee, \wedge, \neg, \exists, \forall$, in its vocabulary. It will take us some time to develop this subject, but when a logic sentence, $\mathtt{s}$, is found to be true in a given structure, μ, we shall designate this fact with the symbology $\mu \models \mathtt{s}$. If the sentence $\mathtt{s}$ is *not* true in μ, we will instead write $\mu \not\models \mathtt{s}$.

In First-Order Logic with some vocabulary τ, consider the terms without free variables – such terms are called **ground terms**. Note that if τ has no constant symbols then there are no ground terms. If τ *does* contain constant symbols, then in any structure μ that interprets the function, relation and constant symbols of τ, the ground terms resolve to a specific element of μ. If we then consider an m-ary relation symbol[4] $R^m \in \tau$, and the expression $R^m(t_1, \ldots, t_m)$ where $t_1, \ldots, t_m$ are all ground terms, then either $R^m(t_1, \ldots, t_m)$ or $\neg R^m(t_1, \ldots, t_m)$ holds in μ, and we analogously write, $\mu \models R^m(t_1, \ldots, t_m)$ or $\mu \not\models R^m(t_1, \ldots, t_m)$ in these two cases.

Definition 11 *We say that one logic,* $L' = (\tau', \mathbb{S}', \mathcal{T}', \sigma', \mathcal{P}')$ *with associated language* $\mathcal{L}'$*,* ***extends*** *another logic,* $L = (\tau, \mathbb{S}, \mathcal{T}, \sigma, \mathcal{P})$ *with associated language* $\mathcal{L}$*, if (i)* $\tau \subseteq \tau', \mathcal{L} \subseteq \mathcal{L}', \sigma \subseteq \sigma'$*, (ii) the rules for truth preservation through the common logical symbols of* τ *and* τ' *(in other words, through the logical symbols of* τ*) are the same, and (iii) every sentence* $\mathtt{s} \in \mathcal{L}$ *that is provable in* L *is provable in* L'*, in other words, for every* $\mathtt{s} \in \mathcal{L}, \vdash_{\mathcal{P}} \mathtt{s}$ *implies* $\vdash_{\mathcal{P}'} \mathtt{s}$*. The logic* L' *is then said to be an* ***extension*** *of the logic* L*.*

The First-Order Logic of directed graphs with vocabulary $\tau' = (\varnothing, \{E(\cdot,\cdot)\}, \{s, t\})$ thereby extends the First-Order Logic of directed graphs with vocabulary $\tau = (\varnothing, \{E(\cdot,\cdot)\}, \varnothing)$. Further, the First-Order Logic of Undirected Graphs can be viewed as an extension of the First-Order Logic of Directed Graphs, if in both logics we use the same edge relation symbol $E(\cdot,\cdot)$ and to the First-Order Logic of Undirected Graphs we add the single axiom

$$\forall x \forall y (E(x, y) \Rightarrow E(y, x)). \tag{81}$$

Let us return now to the question of defining truth in a given logic. The following definition, though somewhat refined since its original conception in 1935, is credited to the Polish logician Alfred Tarski and

[4]Note that there is at least one such relation symbol, since we assume $=(\cdot,\cdot)$ is part of every First-Order vocabulary.

often referred to as Tarski's *theory of truth* [72, 80]. It is a definition of truth that is based on the notion of truth in structures. The rules for truth propagation through the logical symbols $\vee, \wedge$ and $\neg$ are very simple and require little explanation, but things get a bit more complicated when we get to the rules for truth propagation through the quantifiers, $\forall, \exists$ of First-Order Logic. To completely prescribe these rules, we need a couple of definitions.

Definition 12 *Given a logic $L = (\tau, \mathbb{\$}, \mathcal{T}, \sigma, \mathcal{P})$, and τ-structure μ with universe $\mathcal{U}$, denote the set of variables in τ by $\overline{X} = \{x_1, \ldots\}$. Then a **variable assignment** is a mapping $\alpha : \overline{X} \to \mathcal{U}$, taking each variable to a specific element of the universe, $\mathcal{U}$.*

Definition 13 *Suppose we have a formula ϕ with free variables $x_1, \ldots, x_k$, and a structure μ that interprets each of the function, relation and constant symbols in ϕ, and let $a_1, \ldots, a_k$ denote elements of the universe of μ. Then we write $\phi(x_1, \ldots, x_k)[x_1/a_1, \ldots, x_k/a_k]$, or, alternatively, $\phi(x_1, \ldots, x_k)[x_1/a_1], \ldots, [x_k/a_k]$, to denote the mapping of each of the freely occurring variables x_i to corresponding elements a_i under a particular variable assignment α, such that $\alpha(x_i) = a_i$. We refer to $\phi(x_1, \ldots, x_k)[x_1/a_1, \ldots, x_k/a_k]$ in this case as a **fully assigned formula**.*

It is important to note that in Definition 13 the elements of the structure μ that we have designated by $a_1, \ldots, a_k$ are *not* part of any logical language (and, importantly, not constants) and hence a fully assigned formula is *not* a sentence. However, due to their resemblance to sentences, we shall typically use Roman lettering and the `typewriter font` to denote fully assigned formulas, e.g., we will denote, say $\mathtt{r} = \phi(x_1, \ldots, x_k)[x_1/a_1, \ldots, x_k/a_k]$. For succinctness, we sometimes will write $\phi[x_1/a_1, \ldots, x_k/a_k]$ in lieu of $\phi(x_1, \ldots, x_k)[x_1/a_1, \ldots, x_k/a_k]$. Further, as long as $\phi(x_1, \ldots, x_k) \in \mathcal{L}$ we will say, with analogous abuse of notation, that $\mathtt{r} \in \mathcal{L}$. We will also have the need to consider formulas, where all of the variables but one are assigned, and the remaining variable is free. We denote such a formula using notation $\phi(x)[x_1/a_1, \ldots, x_k/a_k]$, with x denoting the single unassigned free variable.

Note that given a term t and a structure $\mu = (\mathcal{U}, \{f_i^{a_i}\}, \{R_i^{\alpha_i}\}, \{c_i\})$ that interprets each of the function and constant symbols appearing in t (recall that terms do *not* contain relation symbols), if we replace each variable appearing in t by an element of $\mathcal{U}$, the result evaluates to an element of $\mathcal{U}$. Then, given an $\mathtt{m}$-ary relation $R^{\mathtt{m}}$ and terms $t_1, \ldots, t_m$, whose collective set of free variables is $\{x_{i_1}, \ldots, x_{i_k}\}$, we say that $\mu \models R^{\mathtt{m}}(t_1, \ldots, t_m)[x_{i_1}/a_1, \ldots, x_{i_k}/a_k]$, if when we replace each variable x_{i_j} by the respective element a_j, the $\mathtt{m}$-tuple of elements of $\mathcal{U}$ given by $(t_1[x_{i_1}/a_1, \ldots, x_{i_k}/a_k], \ldots, t_m[x_{i_1}/a_1, \ldots, x_{i_k}/a_k])$ is an element of $R^{\mathtt{m}}$ (in other words, $R^{\mathtt{m}}$ evaluates to True on this tuple). Combining definitions 5 and 13 we say that $R^{\mathtt{m}}(t_1, \ldots, t_m)[x_{i_1}/a_1, \ldots, x_{i_k}/a_k]$ is a **primitive fully assigned formula**.

Our goal will be to define truth in a given structure for all fully assigned formulas and, in so doing, define truth for all sentences, starting with the primitive fully assigned formulas. Just like we did for sentences, we use the notation $\mu \models \mathtt{r}$ to denote the fact that the fully assigned formula $\mathtt{r}$ is true in μ, or instead write $\mu \not\models \mathtt{r}$, if $\mathtt{r}$ is not true (equivalently, is false) in μ. Although we have defined what we mean by the individual fully assigned formulas, $\mathtt{r} = \phi[x_{i_1}/a_{i_1}, \ldots, x_{i_k}/a_{i_k}]$ and $\mathtt{s} = \psi[x_{j_1}/a_{j_1}, \ldots, x_{j_\ell}/a_{j_\ell}]$, we will also need to define what we mean by expressions like $\neg\mathtt{r}, \mathtt{r} \vee \mathtt{s}$ and $\mathtt{r} \wedge \mathtt{s}$. The extension to the logical not operator, $\neg\mathtt{r} = \neg\phi[x_{i_1}/a_{i_1}, \ldots, x_{i_k}/a_{i_k}]$ is obvious. However, for $\mathtt{r} \vee \mathtt{s}$ if we were to write $(\phi \vee \psi)[x_{i_1}/a_{i_1}, \ldots, x_{i_k}/a_{i_k}, x_{j_1}/a_{j_1}, \ldots, x_{j_\ell}/a_{j_\ell}]$, this expression could be ambiguous if some of the variables used in ϕ, namely the $x_{i_1}, \ldots, x_{i_k}$ overlap with some of the variables used in ψ, namely the $x_{j_1}, \ldots, x_{j_\ell}$, but the associated assigned values of $\mathcal{U}$ are different. Thus, to express $\mathtt{r} \vee \mathtt{s}$ as a fully assigned formula, we sequentially change each variable in ψ that also appears in ϕ to a not yet used variable in *either* ϕ or ψ so that at the end we get a new expression with different variables. Let us designate by $\psi' = \psi'(x'_{j_1}, \ldots, x'_{j_\ell})$ the formula that is otherwise identical to ψ, but where the new set of variables, $\{x'_{j_1}, \ldots, x'_{j_\ell}\}$, is disjoint from the set of variables in ϕ. We then take $\mathtt{r} \vee \mathtt{s}$ to be the fully assigned (and unambiguous) formula $(\phi \vee \psi')[x_{i_1}/a_{i_1}, \ldots, x_{i_k}/a_{i_k}, x'_{j_1}/a_{j_1}, \ldots, x'_{j_\ell}/a_{j_\ell}]$. The fully assigned formula $\mathtt{r} \wedge \mathtt{s}$ is defined analogously.

Definition 14 ***Rules of Truth Preservation in Structures for First-Order Logic.*** *Suppose we are given a logic $L = (\tau, \mathbb{\$}, \mathcal{T}, \sigma, \mathcal{P})$ that extends First-Order Logic, together with an associated logical language $\mathcal{L}$. Let μ be a τ-structure and let $\mathtt{r}$ and $\mathtt{s}$ be two fully assigned formulas of $\mathcal{L}$. Then we have:*

i. If $\mathtt{r}$ is a primitive fully assigned formula that holds in μ, then $\mu \models \mathtt{r}$;

ii. $\mu \models \neg \mathtt{r}$ *iff* $\mu \not\models \mathtt{r}$*;*

iii. $\mu \models \mathtt{r} \vee \mathtt{s}$ *iff* $\mu \models \mathtt{r}$ *or* $\mu \models \mathtt{s}$*;*

iv. $\mu \models \mathtt{r} \wedge \mathtt{s}$ *iff* $\mu \models \mathtt{r}$ *and* $\mu \models \mathtt{s}$*;*

v. If $\mathtt{r} = \phi[x_{i_1}/a_1, \ldots, x_{i_k}/a_k], \mathtt{s} = \phi'[x'_{i_1}/a_1, \ldots, x'_{i_k}/a_k]$ *and the formulas* ϕ *and* ϕ' *are identical up to a renaming of their variables, then* $\mu \models \mathtt{r}$ *iff* $\mu \models \mathtt{s}$*.*

For every formula $\phi(x) = \phi(x)[x_{i_1}/a_1, \ldots, x_{i_k}/a_k]$ *with a single unassigned free variable* x*, we have:*

vi. $\mu \models \forall x(\phi(x))$ *iff for every element* $a \in \mu$*,* $\mu \models \phi(x)[x/a]$*;*

vii. $\mu \models \exists x(\phi(x))$ *iff for some element* $a \in \mu$*,* $\mu \models \phi(x)[x/a]$*.*

By virtue of the sentence construction rules (a.k.a. syntax) of First-Order Logic, rules i. through vii. are sufficient to define truth for all fully assigned formulas, and hence all sentences, with an arbitrary number of quantifiers – one simply argues by induction on the quantifier rank[5] of the associated fully assigned formula. Suppose, for example, that we have the sentence $\mathtt{s} = \exists x(\forall y E(x,y) \vee \forall y E(y,x))$, from the First-Order Logic of graphs, which is of quantifier rank 2, and we are trying to determine the truth or falsity of this sentence for a given directed graph μ. Rules i. through vii. directly describe how to assign truth to all fully assigned formulas of quantifier rank 1. By rule vii., $\mu \models \mathtt{s}$ iff there is some $a \in \mathcal{U}$, such that $\mu \models (\forall y E(x,y) \vee \forall y E(y,x))[x/a]$. We are now down to the quantifier rank 1 case and rule iii. applies, telling us that $\mu \models (\forall y E(x,y) \vee \forall y E(y,x))[x/a]$ iff $\mu \models \forall y E(x,y)[x/a]$ or $\mu \models \forall y E(y,x)[x/a]$. Rule vi. now applies to each of these pieces, giving that $\mu \models \forall y E(x,y)[x/a]$ iff for every $b \in \mathcal{U}, \mu \models E(x,y)[x/a, y/b]$ and $\mu \models \forall y E(y,x)[x/a]$ iff for every $b \in \mathcal{U}, \mu \models E(y,x)[x/a, y/b]$.

Definition 15 *Given a logic* L *with associated language* $\mathcal{L}$*, suppose we are given a set,* $\mathcal{S} \subseteq \mathcal{L}$*, of sentences. Then a structure* μ *is said to be a* ***model*** *of, or for,* $\mathcal{S}$ *if the function, relation and constant symbols used in* $\mathcal{L}$ *are interpreted in* μ*, and each of the sentences of* $\mathcal{S}$ *are true in* μ *(equivalently, for every* $\mathtt{s} \in \mathcal{S}, \mu \models \mathtt{s}$*). The set of sentences* $\mathcal{S}$ *is also said the be* ***modeled*** *by* μ*.*

Consider the First-Order Logic of directed graphs and the sentence:

$$\exists x(\neg E(x,x)). \tag{82}$$

The logical sentence (82) says that the relation designated by $E(\cdot,\cdot)$ is not reflexive. A model of this sentence is any directed graph that has a single node without a self-loop (i.e., an edge starting and terminating at the same node). Quantification is over the vertices of the graph and $E(\cdot,\cdot)$ is interpreted as the directed edge relation between vertices. Consider the subtly different sentence:

$$\forall x(\neg E(x,x)). \tag{83}$$

This sentence says that the relation designated by $E(\cdot,\cdot)$ *never* holds for one and the same element. A model of this sentence is any directed graph such that *no* node has a self-loop.

Definition 16 *Given a logic* $L = (\tau, \mathbb{\$}, \mathcal{T}, \sigma, \mathcal{P})$ *with associated language* $\mathcal{L}$*, a sentence* $\mathtt{s} \in \mathcal{L}$ *is said to be* ***true*** *in* L *if it is true in all* τ*-structures that are models of* σ*. Analogously, the sentence* $\mathtt{s} \in \mathcal{L}$ *is said to be* ***false*** *in* L *if the sentence* $\neg\mathtt{s}$ *is true in all models of* σ *(equivalently, if* $\mathtt{s}$ *is false in all models of* σ*).*

Remark 2 *Suppose we are given a logic* $L = (\tau, \mathbb{\$}, \mathcal{T}, \sigma, \mathcal{P})$ *that extends First-Order Logic and its associated language* $\mathcal{L}$*. Consider a logic sentence* $\mathtt{s} \in \mathcal{L}$ *and a* τ*-structure* μ*. By Condition 14.ii. for Truth Preservation, either* $\mathtt{s}$ *is true in* μ *or false in* μ*. Though the sentence* $\mathtt{s}$ *is necessarily either true or false in* μ*, it may be* neither *true nor false in* L*, since to be true in* L *it must be true in all* τ*-structures that are models of* σ*, and similarly, to be false in* L*, it must be false in all such* τ*-structures. Despite the fact that the sentence* $\mathtt{s}$ *may be neither true nor false in* L*, it is still the case that the sentence* $\mathtt{s} \vee \neg\mathtt{s}$ *is true in* L *since in every* τ*-structure either* $\mathtt{s}$ *or* $\neg\mathtt{s}$ *is true and hence in every* τ*-structure,* $\mathtt{s} \vee \neg\mathtt{s}$ *holds.*

[5] Also known as quantifier nesting depth.

To make Remark 2 concrete, consider the First-Order Logic of graphs and let $\mathtt{s}$ be the sentence (83) that says that there are no self-loops. Given any particular graph μ, either the graph has a vertex with a self-loop or it does not, and thus either $\mathtt{s}$ or $\neg\mathtt{s}$ holds in μ. While neither $\mathtt{s}$ nor $\neg\mathtt{s}$ holds for all graphs, certainly, $\mathtt{s} \vee \neg\mathtt{s}$ does.

Definition 17 *Given a logic $L = (\tau, \mathbb{S}, \mathcal{T}, \sigma, \mathcal{P})$, a **proof system**, $\mathcal{P}$, is a set of rules, known as* rules of inference*, each rule specifying conditions under which, given the truth of certain known sentences, one can conclude the truth of one or more other sentences. Now, let $\mathcal{L}$ be the language associated with L, and let $\mathcal{S} \subseteq \mathcal{L}$, and $\mathtt{s} \in \mathcal{L}$. A **proof** of the sentence $\mathtt{s}$, under the assumption of the truth of the set of sentences $\mathcal{S}$ comprises a finite sequence of steps, including for each step, (i) a rule of inference from the proof system for L, (ii) a sentence or set of sentences to which the rule is applied, such sentence(s) either coming from the axioms, the set $\mathcal{S}$, or the conclusion of prior steps, and (iii) a concluded sentence, and where the final concluded sentence is the sentence $\mathtt{s}$. The sentence $\mathtt{s}$ is then said to be **proved** or **provable** from $\mathcal{S}$ in L, which we denote using the notation $\mathcal{S} \vdash \mathtt{s}$.*

Definition 18 *Given a logic L with associated language, $\mathcal{L}$, a **theorem** of L is a sentence $\mathtt{s} \in \mathcal{L}$ that is provable without additional assumptions (i.e., where $\mathcal{S} = \varnothing$ in Definition 17). By the **theorems of** L, one means the set of all such sentences.*

We have defined what we mean by a proof system but, thus far, have not given an example of one. Our immediate objective is to give such an example for Propositional Logic. Before proceeding to that, let us give a preliminary definition.

Definition 19 *In Propositional Logic on some number $\mathtt{m}$ of variables, a sentence $\mathtt{s} \in \mathcal{L}_{\mathtt{m}}$, is said to be a **tautology** or a **propositional tautology** if $\mathtt{s}$ is true for all truth-value assignments to the $\mathtt{m}$ propositional variables $\mathtt{X}_1, \ldots, \mathtt{X}_{\mathtt{m}}$.*

A key idea that motivates the upcoming definition, is that if we have a propositional tautology $\mathtt{s} \in \mathcal{L}_{\mathtt{m}}$ and now consider another logic L, with language $\mathcal{L}$, that contains the logical connectives of Propositional Logic and propagates truth through these connectives in the same way as Propositional Logic (in other words, in accordance with rules i.-iv. of Definition 14), then if we replace the propositional variables $\mathtt{X}_1, \ldots, \mathtt{X}_{\mathtt{m}}$ by arbitrary sentences $\mathtt{s}_1, \ldots, \mathtt{s}_{\mathtt{m}} \in \mathcal{L}$, we will get a new sentence $\mathtt{s}'$ that is necessarily true in L.

There are many so-called *propositional proof systems*, aimed at codifying a set of rules of inference sufficient for proving all the propositional tautologies [15]. Often these systems are studied from the vantage point of establishing the polynomial-time provability of every propositional tautology from a finite set of rules of inference [19] – a fundamental open problem in theoretical computer science. In this work, we are not interested in questions involving polynomial time provability, or of finding a minimum set of rules of inference, and are just concerned with being able to derive all propositional tautologies from an arbitrary finite set of rules of inference. We therefore adopt the following definition.

Definition 20 *A logic L, with associated language $\mathcal{L}$ and a vocabulary that includes the logical symbols $\vee, \wedge$ and $\neg$, is said to include a **classical propositional proof system** if the following conditions hold. For every triple of sentences $\mathtt{s}, \mathtt{s}_1, \mathtt{s}_2 \in \mathcal{L}$ one has:*

i. *If $\mathtt{s} \vdash \mathtt{s}_1$ then $(\mathtt{s} \vdash \mathtt{s}_1 \vee \mathtt{s}_2$ and $\mathtt{s} \vdash \mathtt{s}_2 \vee \mathtt{s}_1)$,*

ii. *$\mathtt{s} \vdash \mathtt{s}_1 \wedge \mathtt{s}_2$ iff $(\mathtt{s} \vdash \mathtt{s}_1$ and $\mathtt{s} \vdash \mathtt{s}_2)$,*

iii. *$(\mathtt{s}_1 \vdash \mathtt{s}$ and $\mathtt{s}_2 \vdash \mathtt{s})$ iff $\mathtt{s}_1 \vee \mathtt{s}_2 \vdash \mathtt{s}$,*

iv. *If $(\mathtt{s}_1 \vdash \mathtt{s}$ or $\mathtt{s}_2 \vdash \mathtt{s})$ then $\mathtt{s}_1 \wedge \mathtt{s}_2 \vdash \mathtt{s}$,*

v. *If $(\vdash \neg\mathtt{s}_1 \vee \mathtt{s}_2)$ then $\mathtt{s}_1 \vdash \mathtt{s}_2$,*

vi. *$\vdash \mathtt{s} \vee \neg\mathtt{s}$.*

It is common to include the symbol $\Rightarrow$ in the logical vocabulary of L, where upon the condition $\neg \mathtt{s}_1 \vee \mathtt{s}_2$ of 20.v. is typically replaced by $\mathtt{s}_1 \Rightarrow \mathtt{s}_2$, and known as modus ponens. Condition 20.vi. is known as the Law of the Excluded Middle.

With the symbol $\Rightarrow$ in the logical vocabulary, a more common propositional proof system is to use the following set of named rules: *And-introduction*: If $\mathtt{s} \vdash \mathtt{s}_1$ and $\mathtt{s} \vdash \mathtt{s}_2$ then $\mathtt{s} \vdash \mathtt{s}_1 \wedge \mathtt{s}_2$, *And-elimination*: If $\mathtt{s} \vdash \mathtt{s}_1 \wedge \mathtt{s}_2$ then $\mathtt{s} \vdash \mathtt{s}_1$ and $\mathtt{s} \vdash \mathtt{s}_2$, *Or-introduction*: If $\mathtt{s} \vdash \mathtt{s}_1$ or $\mathtt{s} \vdash \mathtt{s}_2$ then $\mathtt{s} \vdash \mathtt{s}_1 \vee \mathtt{s}_2$, *Negation-elimination*: $\neg\neg \mathtt{s} \vdash \mathtt{s}$, *Modus ponens*: If $\vdash \mathtt{s}_1 \Rightarrow \mathtt{s}_2$ then $\mathtt{s}_1 \vdash \mathtt{s}_2$ and *Resolution*: If $\vdash \mathtt{s} \vee \mathtt{s}_1$ and $\vdash \neg \mathtt{s} \vee \mathtt{s}_2$ then $\vdash \mathtt{s}_1 \vee \mathtt{s}_2$. Note, however, that i.–iv. give us the And/Or introduction and elimination rules, v. is Modus ponens, and vi. (together with the other rules) allow us to prove Resolution and Negation-elimination.

Although there are logics with proof systems that don't include a classical propositional proof system (e.g., Intuitionistic Logic [36], which does not include vi.), including a classical propositional proof system may be regarded as the bare minimum requirement of any proof system for doing classical mathematics.

With these definitions in hand, we have the following important notions:

Definition 21 *A logic L, with axioms σ and language $\mathcal{L}$, is said to be **sound** if all of its theorems are true in all models of σ.*

Definition 22 *Suppose we are given a logic L, with axioms σ and language $\mathcal{L}$. The logic L is said to be **strongly sound** if for every sentence $\mathtt{s} \in \mathcal{L}$ and every set, $\mathcal{S} \subseteq \mathcal{L}$, of sentences, if the sentence $\mathtt{s}$ is provable from $\mathcal{S}$, then $\mathtt{s}$ is true in all models of $\sigma \cup \mathcal{S}$.*

Definition 23 *A logic L, with axioms σ and language $\mathcal{L}$, is said to be **complete** if every sentence that is true in all models of σ is a theorem of L.*

Definition 24 *Suppose we are given a logic L, with axioms σ and language $\mathcal{L}$. The logic L is said to be **strongly complete** if for every sentence $\mathtt{s} \in \mathcal{L}$ and every set, $\mathcal{S} \subseteq \mathcal{L}$, of sentences, if $\mathtt{s}$ is true in all models of $\sigma \cup \mathcal{S}$, then $\mathtt{s}$ can be proven from $\mathcal{S}$ in L.*

In the case of completeness, we will actually need a notion that is in between completeness and strong completeness as follows.

Definition 25 *Suppose we are given a logic L, with axioms σ and language $\mathcal{L}$. The logic L is said to be **ω-strongly complete** if for every sentence $\mathtt{s} \in \mathcal{L}$ and every* finite *set, $\mathcal{S} \subseteq \mathcal{L}$, of sentences, if $\mathtt{s}$ is true in all models of $\sigma \cup \mathcal{S}$, then $\mathtt{s}$ can be proven from $\mathcal{S}$ in L.*

Definition 26 *Given a logic L with associated language $\mathcal{L}$, a set $\mathcal{S} \subseteq \mathcal{L}$ of sentences is said to be **consistent** if there is a structure μ in which all of the sentences of $\mathcal{S}$ are true. The set of sentences $\mathcal{S}$ is said to be **inconsistent** otherwise.*

For the statement of the fundamental equivalence theorem (Theorem 9) that is one of the center-pieces of this subsection, we will need one additional definition.

Definition 27 *Suppose we are given a logical vocabulary τ and two τ-structures μ_1, μ_2, such that $\mu = \{\mathcal{U}, f_1^{a_1}, \ldots, f_k^{a_k}, R_1^{\alpha_1}, \ldots, R_\ell^{\alpha_\ell}, c_1, \ldots, c_m\}$ and $\mu' = \{\mathcal{U}', f_1'^{a_1}, \ldots, f_k'^{a_k}, R_1'^{\alpha_1}, \ldots, R_\ell'^{\alpha_\ell}, c_1', \ldots, c_m'\}$, where $\mathcal{U}$ and $\mathcal{U}'$ are the underlying universes, $f^{a_i}, R_i^{\alpha_i}(\cdot), c_i$ and $f_i'^{a_i}, R_i'^{\alpha_i}(\cdot), c_i'$ are the function, relation and constants associated with the same function, relation and constant symbols, and a_i, α_i the associated arities of those functions/function symbols and relations/relation symbols. Then μ and μ' are said to be **isomorphic** (equiv. **τ-isomorphic**) iff there is a function, relation and constant preserving bijection $\pi : \mathcal{U} \to \mathcal{U}'$. In other words, there is a bijective map π such that:*

i. For every function symbol $f_i^{a_i} \in \tau$ and every a_i-tuple of elements $(e_1, \ldots, e_{a_i})$ from $\mathcal{U}_1$, one has $\pi(f_i(e_1, \ldots, e_{a_i})) = f_i'(\pi(e_1), \ldots, \pi(e_{a_i}))$,

ii. For every relation symbol $R_i^{\alpha_i} \in \tau$ and every α_i-tuple of elements $(e_1, \ldots, e_{\alpha_i})$ from $\mathcal{U}_1$, one has $R_i(e_1, \ldots, e_{\alpha_i})$ iff $R_i'(\pi(e_1), \ldots, \pi(e_{\alpha_i}))$.

iii. For $1 \leq i \leq m, \pi(c_i) = c_i'$.

Theorem 9 (fundamental equivalence) *Let $L = (\tau, \$, \mathcal{T}, \sigma, \mathcal{P})$ be a logic with associated language $\mathcal{L}$, and, moreover, such that the vocabulary τ contains the logical symbol $\wedge$, the following hold:*

1. *Given sentences $\mathtt{r}, \mathtt{s} \in \mathcal{L}$, the syntax $\$$ supports the formation of the sentence $\mathtt{r} \wedge \mathtt{s}$.*
2. *Truth propagates through the symbol $\wedge$ in accordance with how truth propagates through this symbol in Propositional Logic (Definition 14.iv.), in other words, for every τ-structure μ, and every pair of sentences $\mathtt{r}, \mathtt{s} \in \mathcal{L}$, $\mu \models \mathtt{r} \wedge \mathtt{s}$ iff both $\mu \models \mathtt{r}$ and $\mu \models \mathtt{s}$,*
3. *The proof system of L is sufficient to utilize rules ii. and iv. of being a classical Propositional Proof System. In other words, for sentences, $\mathtt{s}_1, \mathtt{s}_2, \mathtt{s} \in \mathcal{L}$, (i) $\mathtt{s} \vdash \mathtt{s}_1 \wedge \mathtt{s}_2$ iff ($\mathtt{s} \vdash \mathtt{s}_1$ and $\mathtt{s} \vdash \mathtt{s}_2$), and (ii) if $\mathtt{s}_1 \vdash \mathtt{s}$ or $\mathtt{s}_2 \vdash \mathtt{s}$, then $\mathtt{s}_1 \wedge \mathtt{s}_2 \vdash \mathtt{s}$.*

Suppose further that $\mathcal{M}$ consists of all τ-structures up to the isomorphism. Let $\kappa : \mathcal{L} \to \mathcal{P}(\mathcal{M})$ map each sentence to the collection of structures in $\mathcal{M}$ that satisfy (i.e., model) it as well as satisfy each of the sentences of σ[6]. Then, for all $\mathtt{s}_1, \mathtt{s}_2 \in \mathcal{L}$,

$$\mathtt{s}_1 \vdash \mathtt{s}_2 \text{ implies } \kappa(\mathtt{s}_1) \subseteq \kappa(\mathtt{s}_2) \tag{84}$$

iff L is strongly sound. Further, for all $\mathtt{s}_1, \mathtt{s}_2 \in \mathcal{L}$,

$$\kappa(\mathtt{s}_1) \subseteq \kappa(\mathtt{s}_2) \text{ implies } \mathtt{s}_1 \vdash \mathtt{s}_2 \tag{85}$$

iff L is ω-strongly complete.

Proof. Suppose first that L is strongly sound. Then, under the assumption that $\mathtt{s}_1 \vdash \mathtt{s}_2$, in any structure in which $\{\mathtt{s}_1\} \cup \sigma$ is true, $\mathtt{s}_2$ is true. It follows that $\kappa(\mathtt{s}_1) \subseteq \kappa(\mathtt{s}_2)$. On the other hand, suppose L is *not* strongly sound. Then there is some set of sentences $\mathcal{S}$ from which L can prove a sentence $\mathtt{s}$ even though $\mathtt{s}$ is not true in all models of $\sigma \cup \mathcal{S}$. However, by virtue of the rules for what constitutes a proof in a logic (Definition 17), in the purported proof of $\mathtt{s}$, only finitely many sentences $\mathtt{s}_1, \ldots, \mathtt{s}_n \in \mathcal{S}$ are used. Hence, starting from just the axioms of L, we can write $\{\mathtt{s}_1, \ldots, \mathtt{s}_n\} \vdash \mathtt{s}$. Then, since $\mathtt{s}_1 \vdash \mathtt{s}_1$, by repeated application of condition 3.(ii) in the statement of the lemma, we get that $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n \vdash \mathtt{s}_1$. Since the same argument can be made for each the sentences $\mathtt{s}_i$, it follows that $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n \vdash \{\mathtt{s}_1, \ldots, \mathtt{s}_n\}$. Hence, be transitivity of the entailment relation, $\vdash$[7], it follows that $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n \vdash \mathtt{s}$. But $\mathtt{s}_1, \ldots, \mathtt{s}_n \in \mathcal{S}$, so the sentences $\mathtt{s}_1, \ldots, \mathtt{s}_n$ are true in every model of $\sigma \cup \mathcal{S}$, and so too, by condition 2 in the statement of the lemma, $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n$ is true in every model of $\sigma \cup \mathcal{S}$. On the other hand, $\mathtt{s}$ is *not* true in every model of $\sigma \cup \mathcal{S}$ and so we have $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n \vdash \mathtt{s}$ but $\kappa(\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n) \not\subseteq \kappa(\mathtt{s})$. The first part of the lemma, involving implication (84) and strong soundness is therefore established.

Next, assume L is ω-strongly complete and suppose $\kappa(\mathtt{s}_1) \subseteq \kappa(\mathtt{s}_2)$. Then the sentence $\mathtt{s}_2$ is true in every structure in which σ and $\mathtt{s}_1$ are true. It follows, by ω-strong completeness, that $\mathtt{s}_1 \vdash \mathtt{s}_2$. Suppose, on the other hand, that L is *not* ω-strongly complete. There is then some *finite* set of sentences $\mathcal{S} = \{\mathtt{s}_1, \ldots, \mathtt{s}_n\} \subseteq \mathcal{L}$ and a sentence $\mathtt{s}$ such that $\mathtt{s}$ is true in all models of $\sigma \cup \mathcal{S}$ but for which we don't have $\mathcal{S} \vdash \mathtt{s}$. Since $\mathcal{S} \vdash \mathtt{s}_i$ for $1 \leq i \leq n$, by condition 3.(i) it follows that $\mathcal{S} \vdash \mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n$. Hence, by the transitivity of $\vdash$, it must be that $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n \nvdash \mathtt{s}$. By condition 2 of the lemma, $\kappa(\mathcal{S}) = \kappa(\{\mathtt{s}_1, \ldots, \mathtt{s}_n\}) = \kappa(\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n) \subseteq \kappa(\mathtt{s})$. Hence, if L is not ω-strongly complete, condition (85) does not hold with respect to the pair of sentences $\mathtt{s}_1 \wedge \cdots \wedge \mathtt{s}_n$ and $\mathtt{s}$. This establishes the second half of the lemma and the overall proof is complete. □

In addition to First-Order Logic, a logic that satisfies the assumptions of Theorem 9 is All Positive First-Order Logic, a fragment of First-Order Logic that has no negation symbol and no universal quantifier [45].

Returning to our previously established terminology around the word *kernel* (Definition 1 from Section 2), we see that under the mapping defined in Theorem 9, the kernel of a sentence $\mathtt{s}$, i.e., $\kappa(\mathtt{s})$, is the collection of structures in $\mathcal{M}$ which are models of $\mathtt{s}$, equivalently, the collection of structures in $\mathcal{M}$ for which $\mathtt{s}$ is true.

[6] It is possible that $\mathcal{M}$ is not actually a set but rather a *proper class*, as would be the case if $\mathcal{M}$ were the set of all directed graphs up to isomorphism. Then $\mathcal{P}(\mathcal{M})$ would not be a set either, and is properly referred to as the *power class* of $\mathcal{M}$, rather than the power set of $\mathcal{M}$.

[7] The fact that the meta-mathematical relation $\vdash$ is transitive follows from how proofs are defined in Definition 17.

Lemma 7 (kernel relations) *Let $\lambda = (L, \mathcal{M}, \kappa, \ell)$, be a Logic System with associated vocabulary τ and language $\mathcal{L}$, and suppose λ satisfies the following conditions:*

i. The vocabulary τ contains the logical symbols $\vee, \wedge$ and $\neg$, and truth propagates through these symbols in accordance with how truth propagates in classical Propositional Logic (in other words, in accordance with rules i.-iv. of Definition 14).

ii. The collection $\mathcal{M}$ consists of an arbitrary collection of τ-structures.

iii. The function $\kappa : \mathcal{L} \to \mathcal{P}(\mathcal{M})$ maps each sentence, $s \in \mathcal{L}$, to the collection of all structures in $\mathcal{M}$ that model it.

Then for every $\mathtt{s}, \mathtt{t} \in \mathcal{L}$, the following hold:

1. $\kappa(\mathtt{s} \vee \mathtt{t}) = \kappa(\mathtt{s}) \cup \kappa(\mathtt{t})$,
2. $\kappa(\mathtt{s} \wedge \mathtt{t}) = \kappa(\mathtt{s}) \cap \kappa(\mathtt{t})$,
3. $\kappa(\neg\mathtt{s}) = \kappa(\mathtt{s})^C$.

Thus λ is a proper Logic System.

Proof. By truth preservation rule 14.iii., for every τ-structure μ, $\mu \models \mathtt{s} \vee \mathtt{t}$ iff $\mu \models \mathtt{s}$ or $\mu \models \mathtt{t}$. Given the definition of κ, equality 1. immediately follows. Analogously, by truth preservation rule 14.iv., for every *tau*-structure μ, $\mu \models \mathtt{s} \wedge \mathtt{t}$ iff $\mu \models \mathtt{s}$ *and* $\mu \models \mathtt{t}$, so equality 2. follows. Finally, by rule 14.ii., $\mu \models \neg\mathtt{s}$ iff $\mu \not\models \mathtt{s}$, which gives equality 3. □

6.3 Propositional Logic

In this section we show that Propositional Logic on a fixed number, $\mathtt{m}$, of variables, can fit within the same basic framework for being a logic as those logics that extend First-Order Logic, with the exception that Propositional Logic does not admit quantifiers. Fulfilling our promise from Subsection 2.1, we will also show that with appropriate choices for the set $\mathcal{M}$ and the functions κ, ℓ, the Propositional Logic of $\mathtt{m}$ variables can be turned into a proper Logic System.

The fundamental difficulty in trying to fit Propositional Logic into the framework we have elucidated for extensions of First-Order Logic is that there is no analog of the propositional variables $\mathtt{X}_1, \ldots, \mathtt{X}_\mathtt{m}$ in the vocabularies of logics that extend First-Order Logic, and the accepted terminology of calling the $\mathtt{X}_\mathtt{i}$ "variables" adds a certain amount of confusion. It is therefore useful to think of the following non-standard vocabulary, τ_{PL}, for Propositional Logic. In addition to the usual logical connectives $\vee, \wedge$ and $\neg$, τ_{PL} includes a unary relation symbol $One(\cdot)$, a binary relation symbol $<(\cdot,\cdot)$, and a set of $\mathtt{m}$ constant symbols $c_1, \ldots, c_\mathtt{m}$. There are no variable symbols and no quantifiers. Per convention, we write $c_i < c_j$ in lieu of $<(c_i, c_j)$. In addition, the formal proscription of this logic includes the following set of axioms, σ_{PL}. For every i, j with $1 \leq i, j \leq \mathtt{m}$,

$$c_i < c_j \qquad \text{if i} < \text{j}, \tag{86}$$

$$\neg(c_i < c_j) \qquad \text{otherwise.} \tag{87}$$

We may then regard the propositional variable $\mathtt{X}_\mathtt{i}$ as shorthand for the expression $One(c_i)$. The rules of sentence formation are now just the same as in First-Order Logic (Definitions 4 – 8) with the exception that there is no rule R3 in Definition 6. The only terms are the constant symbols, and every formula is a sentence. There are no variables, so Definition 7 does not apply. The primitive sentences are the sentences $One(c_i)$ for $\mathtt{1} \leq i \leq \mathtt{m}$ and sentences are just Boolean combinations of these, in other words Boolean combinations of the $\mathtt{X}_\mathtt{i}$ – as we'd expect.

Recall that a logic, $L = (\tau, \$, \mathcal{T}, \sigma, \mathcal{P})$, consists of a vocabulary τ, a syntax $\$$, or rules for sentence formation, rules, $\mathcal{T}$, for truth propagation through the logical symbols of τ, a set of axioms, σ, and a proof system $\mathcal{P}$. For Propositional Logic we have described $\tau, \$$ and σ. The rules of $\mathcal{T}$ are just rules i.-iv. of Definition 14, and the proof system, $\mathcal{P}$, of Propositional Logic is just the classical propositional proof system given in Definition 20.

The axioms $\sigma_{\rm PL}$ (the family of sentences (86) and (87), completely determine how the relation $<(\cdot,\cdot)$ is defined on the constants, so, with one caveat that we shall get to in a moment, $\tau_{\rm PL}$-structures – the structures of Propositional Logic on $\mathtt{m}$ variables – are completely determined by how the relation $One(\cdot)$ is defined on the $\mathtt{m}$ constants $c_1,\ldots,c_{\mathtt{m}}$. Any particular definition of $One(\cdot)$ corresponds to a choice of which values on $One(c_i)$ are true, and hence to truth value assignment to the $\mathtt{m}$ propositional variables $\mathtt{X}_1,\ldots,\mathtt{X}_{\mathtt{m}}$, in accordance with how we defined the set $\mathcal{M}$ back in Example 1 of Subsection 2.1. Although, as we have pointed out, the accepted terminology of calling the expressions $\mathtt{X}_{\mathtt{i}}$ (a.k.a. $One(c_i)$) "propositional variables" is misleading, we shall by and large keep to the accepted terminology. Note that a structure $\mu=(\mathcal{U},\varnothing,\{One(\cdot),<(\cdot,\cdot)\},\{c_1,\ldots,c_{\mathtt{m}}\})$ can have a universe, $\mathcal{U}$, with more than the $\mathtt{m}$ elements associated with the constants $c_1,..,c_{\mathtt{m}}$ – this is the caveat that we alluded to earlier. These elements, however, can have no bearing on the truth or falsity of any sentence, since they cannot be addressed, and we call them **atoms**. There are no atoms in First-Order Logic. The possibility of atoms requires just one change to the definitions we have assembled in the prior section, namely to the definition of what it means for two structures to be isomorphic (τ-isomorphic) – Definition 27. Instead of demanding that the function, relation and constant-preserving map $\pi:\mathcal{U}\rightarrow\mathcal{U}'$ be a bijection, we must require instead that the map π be a bijection with respect to the *non-atoms* in $\mathcal{U}$ and $\mathcal{U}'$.

Let us now revert to the more customary way of thinking about Propositional Logic with $\mathtt{m}$ ordered propositional variables $\mathtt{X}_1,\ldots,\mathtt{X}_{\mathtt{m}}$ and $\mathcal{M}=\{\mathtt{False},\mathtt{True}\}^{\mathtt{m}}$ defined to be the set of all truth value assignments to the $\mathtt{m}$ ordered propositional variables. Given a logic sentence $\mathtt{s}\in\mathcal{L}_{\mathtt{m}}$, we define the kernel function $\kappa:\mathcal{L}_{\mathtt{m}}\rightarrow\mathcal{P}(\mathcal{M})$ such that $\kappa(\mathtt{s})$ is the subset of $\mathcal{M}$ that makes $\mathtt{s}$ true. For brevity, in what follows, we will generally write 1 for $\mathtt{True}$ and 0 for $\mathtt{False}$. As an illustrative example, suppose the propositional variables are properties of objects in the popular game of twenty questions. For simplicity, let us take $\mathtt{m}=3$ and let $\mathtt{X}_1$ denote the property 'is a country', let $\mathtt{X}_2$ denote the property 'is a place', and let $\mathtt{X}_3$ denote the property 'has a population of over one million people'. Sample sentences of the associated Propositional language $\mathcal{L}_3$ are $\mathtt{X}_1\vee\neg\mathtt{X}_2$ and $\neg\mathtt{X}_3$. One then has $\kappa(\mathtt{X}_1\vee\neg\mathtt{X}_2)=\{000,001,100,101,110,111\}$ and $\kappa(\neg\mathtt{X}_3)=\{000,010,100,110\}$.

The following Lemma formalizes the fact that Propositional Logic on a fixed number of variables, with $\mathcal{M}$ and κ defined as above, satisfies the condition (6) for being a Logic System (Definition 1).

Lemma 8 (entailment and kernels) *Suppose L is Propositional Logic on* $\mathtt{m}$ *variables. Then, given two logic sentences,* $\mathtt{a},\mathtt{b}\in\mathcal{L}_{\mathtt{m}}$, $\mathtt{a}\vdash\mathtt{b}$ *if and only if* $\kappa(\mathtt{a})\subseteq\kappa(\mathtt{b})$.

Proof. The lemma follows immediately from Theorem 9 by virtue of the fact that $\mathcal{M}$ contains a model of every consistent sentence in $\mathcal{L}_{\mathtt{m}}$ and Propositional Logic is both strongly sound and strongly complete (and therefore ω-strongly complete). See [79] for details concerning the soundness and completeness of Propositional Logic. □

As an illustration, one can easily check that the kernel subset relation predicted by the Lemma works for the example

$$(\mathtt{X}_1\vee\neg\mathtt{X}_2)\wedge\neg\mathtt{X}_3\vdash\mathtt{X}_1\vee\neg\mathtt{X}_2.$$

Fundamental to the results in this manuscript is the idea that one can recover from a kernel $\kappa(\mathtt{s})$ a logic sentence $\hat{\mathtt{s}}$ that is functionally equivalent to $\kappa(\mathtt{s})$, in the sense that $\mathtt{s}\vdash\hat{\mathtt{s}}$ and $\hat{\mathtt{s}}\vdash\mathtt{s}$. Since $\kappa(\cdot)$ is a many-to-one function, it doesn't have an inverse, but we will show that a sentence in Disjunctive Normal Form (DNF) can be canonically constructed from the kernel. Denote by $\mathcal{P}(\{0,1\}^{\mathtt{m}})$ the collection of all possible sets of $\mathtt{m}$-bit strings – equivalently, the set of all possible kernels of logic sentences on $\mathtt{m}$ propositional variables. Note that since there are $2^{\mathtt{m}}$ possible $\mathtt{m}$-bit strings there are $2^{2^{\mathtt{m}}}$ sets of $\mathtt{m}$-bit strings.

Definition 28 (from kernels to logic sentences and back) *We define* $\ell:\mathcal{P}(\{0,1\}^{\mathtt{m}})\rightarrow\mathcal{L}_{\mathtt{m}}$ *to be the function that maps an element* $M\in\mathcal{P}(\{0,1\}^{\mathtt{m}})$ *to a canonical DNF sentence that has* M *as its kernel. Let* $M=\{e_1,e_2,\ldots,e_{\mathtt{k}}\}$, *with each* $e_{\mathtt{i}}=\langle\epsilon_{\mathtt{i}1},\ldots,\epsilon_{\mathtt{i}\mathtt{m}}\rangle\in\{0,1\}^{\mathtt{m}}$. *Then to obtain* $\ell(M)$, *for each* $e_{\mathtt{i}}$ *we create a conjunctive clause including* $\mathtt{X}_{\mathtt{j}}$ *whenever* $\epsilon_{ij}=1$ *and including* $\neg\mathtt{X}_{\mathtt{j}}$ *whenever* $\epsilon_{ij}=0$. *We then take* $\ell(M)$ *to be the disjunction of all these conjunctive clauses.*

We generally write an m-bit string more compactly as $\epsilon_{\mathtt{i}1}\cdots\epsilon_{\mathtt{i}\mathtt{m}}$ rather than $\langle\epsilon_{\mathtt{i}1},\ldots,\epsilon_{\mathtt{i}\mathtt{m}}\rangle$, so 101 rather than $\langle1,0,1\rangle$. With this more compact notation, an example of the above definition is: $\ell(\{000,100,110\})=(\neg\mathtt{X}_0\wedge\neg\mathtt{X}_1\wedge\neg\mathtt{X}_2)\vee(\mathtt{X}_0\wedge\neg\mathtt{X}_1\wedge\neg\mathtt{X}_2)\vee(\mathtt{X}_0\wedge\mathtt{X}_1\wedge\neg\mathtt{X}_2)$.

Lemma 9 *For every $M \in \mathcal{P}(\{0,1\}^{\mathtt{m}})$, the functions κ and ℓ satisfy the relation $\kappa(\ell(M)) = M$.*

Proof. Let $M \in \mathcal{P}(\{0,1\}^{\mathtt{m}})$ with $M = \{e_1, e_2, \ldots, e_{\mathtt{k}}\}$ and each $e_{\mathtt{i}} = \langle \epsilon_{\mathtt{i}1}, \ldots, \epsilon_{\mathtt{i}\mathtt{m}} \rangle \in \{0,1\}^{\mathtt{m}}$. It is clear by construction that $M \subseteq \kappa(\ell(M))$. So let $e \in \{0,1\}^{\mathtt{m}}$ be such that $e \notin M$. We must show that $e \notin \kappa(\ell(M))$, in other words that e is not satisfied by the DNF sentence given by $\ell(M)$. Since $e \notin M$, e differs from each $e_{\mathtt{i}} = \langle \epsilon_{\mathtt{i}1}, \ldots, \epsilon_{\mathtt{i}\mathtt{m}} \rangle$ in at least one position. But it then follows that e does not satisfy any of the clauses in $\ell(M)$ and therefore e is not satisfied by $\ell(M)$ – the disjunction of these clauses. The lemma follows. □

Lemmas 8 and 9 enable us to conclude that:

Corollary 1 *Propositional Logic on a fixed set of m variables, where $\mathcal{M}$ is the set of all truth-value assignments to the m variables, κ is the function taking each sentence to the set of truth-value assignments for which the sentence is true, and ℓ is the function defined in Definition 28, is a well defined Logic System.*

Henceforth, we shall call the Logic System for Propositional Logic on m variables that includes the definitions of $\mathcal{M}, \kappa$, and ℓ, as given in the above corollary, the *standard Logic System* for Propositional Logic. Since $\mathcal{M}$ and κ as defined in Corollary 1 satisfy the conditions of Lemma 7, by the conclusion of that lemma, we immediately have:

Corollary 2 *The standard Logic System for Propositional Logic on a fixed set of m variables is a proper Logic System.*

6.3.1 Propositional Logic: Synopsis of Notation and Terminological Conventions

Suppose we are considering Propositional Logic on $\mathtt{m}$ variables, with vocabulary τ, language $\mathcal{L}_{\mathtt{m}}$ and $\mathtt{s} \in \mathcal{L}_{\mathtt{m}}$. Let $\lambda = (L, \mathcal{M}, \kappa, \ell)$ denote the **standard Logic System** for this logic. Then we have the following:

- μ = structure = truth-value assignment to the $\mathtt{m}$ propositional variables $\mathtt{X}_1, \ldots, \mathtt{X}_{\mathtt{m}}$. May be thought of as a bit-string of length $\mathtt{m}$. More formally, $\mu = (\mathcal{U}, <(\cdot,\cdot), One(\cdot), c_1, \ldots, c_{\mathtt{m}})$ for a particular choice of the unary relation $One(\cdot)$. $\mathtt{X}_{\mathtt{i}}$ is then shorthand for $One(c_i)$.
- $\mathcal{M}$ = set of all structures up to isomorphism = set of all truth-value assignments to the $\mathtt{m}$ propositional variables.
- $\kappa(\mathtt{s})$ = "kernel" of the sentence $\mathtt{s}$ = set of truth value assignments to the $\mathtt{m}$ variables that make the sentence $\mathtt{s}$ true = structures in which $\mathtt{s}$ is true.
- $\vec{\kappa}(\mathtt{s})$ = indicator vector representation for the kernel of $\mathtt{s}$. We enumerate the $2^{\mathtt{m}}$ possible truth-value assignments to the $\mathtt{m}$ variables as binary strings and indicate which assignments make the sentence $\mathtt{s}$ true using a length-$2^{\mathtt{m}}$ indicator vector.
- $\mathcal{P}(\mathcal{M})$ = all possible sets of truth-value assignments = all possible kernels. There are $2^{2^{\mathtt{m}}}$ such sets. This is the image of the function κ in the definition of a Logic System (Definition 1).

Figure 22 depicts the objects described above for the case of Propositional Logic on 3 variables and the sentence $\mathtt{s} = (\mathtt{X}_1 \vee \neg \mathtt{X}_2) \wedge \mathtt{X}_3$.

6.4 First-Order Logic and Models of Bounded Size

In this subsection we consider the First-Order Logic (FOL) of directed graphs on a fixed set of m vertices and show that, by picking $\mathcal{M}, \kappa$ and ℓ appropriately, this logic can be turned into a well defined proper Logic System. In order to define the inference rules for FOL above and beyond those for Propositional Logic, we need some additional definitions.

Definition 29 *Suppose we are given a FO language $\mathcal{L}$ with associated FO vocabulary τ. Let x be a variable in τ, t a term comprised of symbols from τ, and let $\phi \in \mathcal{L}$ be a formula with a single free variable x. Then by* SUBST$(\{x/t\}, \phi)$*, we mean the result of replacing every* free *occurrence of the variable x in ϕ by t.*

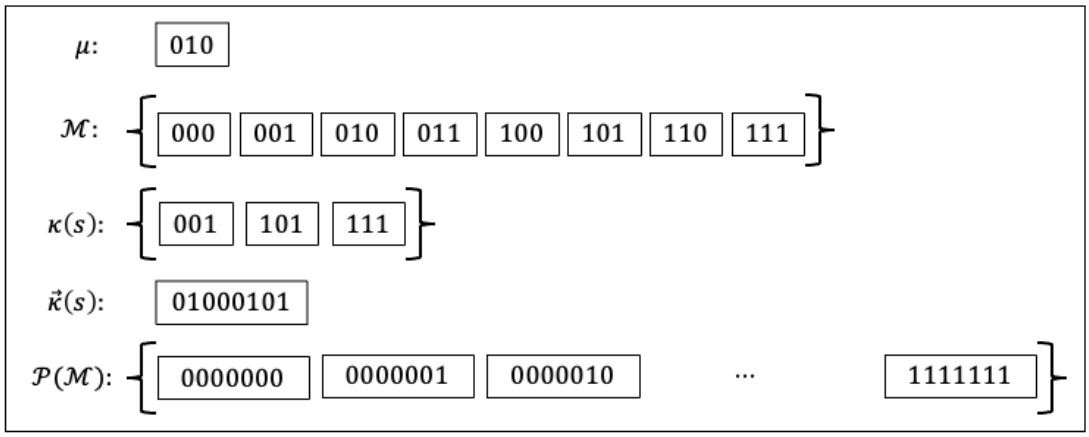


Figure 22: Pictorial representation of the objects $\mu, \mathcal{M}, \kappa(\mathtt{s}), \vec{\kappa}(\mathtt{s})$ and $\mathcal{P}(\mathcal{M})$ for the case of Propositional Logic on 3 variables. The displayed kernels, $\kappa(\mathtt{s})$ and $\vec{\kappa}(\mathtt{s})$ are for the sentence $\mathtt{s} = (\mathtt{X}_1 \vee \neg \mathtt{X}_2) \wedge \mathtt{X}_3$.

Note the slight subtlety in the above definition, that the formula ϕ may have bound occurrences of the variable x in addition to the free occurrences. The substitution called out in $\text{SUBST}(\{x/t\}, \phi)$ just replaces the free occurrences of x with t.

We then have the following:

Definition 30 *Suppose we are given a logic $L = (\tau, \$, \mathcal{T}, \sigma, \mathcal{P})$ such that τ is a FO vocabulary, $\$$ is FO Syntax, so that the associated language, $\mathcal{L}$, is a FO language. When we say that L contains a **proof system for FOL**, we mean that the proof system, $\mathcal{P}$, includes a classical propositional proof system and, in addition, includes the following rules of inference. For every formula $\phi \in \mathcal{L}$ with one free variable,*

i. ***Universal Instantiation.*** *For every ground term g, if $\vdash \forall x \phi(x)$ then $\vdash \text{SUBST}(\{x/g\}, \phi(x))$,*

ii. ***Existential Instantiation.*** *Suppose there is a constant symbol $k \in \tau$ that is not mentioned in any sentence of σ or, if this is not the case, augment τ with an additional constant symbol, k, and extend the associated FO language, $\mathcal{L}$, accordingly. If $\vdash \exists x \phi(x)$, then $\vdash \text{SUBST}(\{x/k\}, \phi(x))$,*

iii. ***Universal Generalization.*** *If, for an arbitrary constant symbol $c \in \tau$ that is not mentioned in any sentence of σ, $\vdash p(c)$ then $\vdash \forall x \phi(x)$,*

iv. ***Existential Generalization.*** *If for some constant symbol $c \in \tau$, and for a sentence σ in which c appears, suppose $\vdash \sigma$. Then, replacing some (possibly all) occurrences of c by a variable x not appearing in σ, we have that $\vdash \exists x \sigma(x)$.*

Henceforth, in this section, by a graph we mean a directed graph. Our logic, L, is the FOL of such graphs, $\mathcal{L}$ is the associated language, having, in addition to the equality relation symbol $= (\cdot, \cdot)$, the single additional relation symbol $E(\cdot, \cdot)$, and there is a single axiom, $\sigma_{\mathtt{m}}$, saying that a graph must contain exactly m vertices:

$$\sigma_{\mathtt{m}} = \exists \mathtt{x}_1 \cdots \exists \mathtt{x}_{\mathtt{m}} \forall \mathtt{y} \Big(\bigwedge_{1 \le i \ne j \le m} \mathtt{x}_{\mathtt{i}} \ne \mathtt{x}_{\mathtt{j}} \ \wedge \bigvee_{1 \le i \le \mathtt{m}} \mathtt{y} = \mathtt{x}_{\mathtt{i}} \Big). \tag{88}$$

This sentences says, firstly, that there are some $\mathtt{m}$ distinct nodes, and, secondly, that any additional node must be equal to one of the $\mathtt{m}$ distinct nodes. The collection, $\mathcal{M}$, of structures, is the set of all directed graphs on m vertices, and, following Definition 30, $\vdash$ is the usual logical entailment in FOL, but for which we can additionally assume the truth of the sentence $\sigma_{\mathtt{m}}$. We will show that we can define the functions κ and ℓ in such a way that $\lambda = (L, \mathcal{M}, \kappa, \ell)$ becomes a proper Logic System.

For the function κ, and $\mathtt{s} \in \mathcal{L}$, we define $\kappa(\mathtt{s})$ to be the set M of all $\mathtt{m}$ vertex graphs satisfying $\mathtt{s}$. For a given graph $G = (V, E) \in \mathcal{M}$ we next exhibit a sentence $\mathtt{s}_G$ that is true for G but false for every other graph in $\mathcal{M}$. For this purpose, let $V = \{v_i\}_{i=1}^{\mathtt{m}}$. Then write:

$$\mathtt{s}_G = \exists \mathtt{x}_1 \cdots \exists \mathtt{x}_\mathtt{m} \Big(\bigwedge_{i \neq j} \mathtt{x_i} \neq \mathtt{x_j} \ \wedge \bigwedge_{(v_i, v_j) \in E} E(\mathtt{x_i}, \mathtt{x_j}) \ \wedge \bigwedge_{(v_i, v_j) \notin E} \neg E(\mathtt{x_i}, \mathtt{x_j}) \Big). \tag{89}$$

Now, given $M \subseteq \mathcal{M}$, let

$$\ell(M) = \bigvee_{G \in M} \mathtt{s}_G. \tag{90}$$

Lemma 10 *With κ and ℓ as defined above, and $M \subseteq \mathcal{M}$ an arbitrary set of m vertex graphs, the functions κ and ℓ satisfy the relation $\kappa(\ell(M)) = M$.*

Proof. To prove the lemma we show that the sentence $\ell(M)$, as given by (90), is satisfied by every graph $G \in M$ but by no other m-vertex graph. Each sentence $\mathtt{s}_G$, as in (89), satisfies some $G \in M$, so their disjunction, $\bigvee_{G \in M} \mathtt{s}_G$, is satisfied by *every* $G \in M$. On the other hand, if $G \notin M$ then G satisfies none of the $\mathtt{S}_G$ for $G \in M$, and so G does not satisfy the disjunction of all these sentences, which is $\ell(M)$, and so the lemma is established. □

Lemma 11 *Let L be the FOL of $\mathtt{m}$-node directed graphs and let $\mathcal{L}$ be the associated FO language. Given $\mathtt{a}, \mathtt{b} \in \mathcal{L}$, we have that $\mathtt{a} \vdash \mathtt{b}$ if and only if $\kappa(\mathtt{a}) \subseteq \kappa(\mathtt{b})$.*

Proof. A sentence in the First-Order theory of graphs given $\sigma_\mathtt{m}$ is true iff it is true for all m-vertex graphs. The rules of inference of FOL above and beyond those of Propositional Logic, are easily seen to preserve truth, so FOL is strongly sound. Strong completeness (and hence ω-strong completeness) follows as a consequence of Gödel's Completeness Theorem [67][8]. The result therefore follows by Theorem 9. □

Theorem 10 *Consider $\lambda = (L, \mathcal{M}, \kappa, \ell)$, where L is the First-Order Logic of Directed Graphs on a fixed number, m, of vertices, $\mathcal{M}$ is the set of all directed graphs on m vertices, κ is the function mapping each sentence in $\mathcal{M}$ to the m-vertex directed graphs that satisfy it, and ℓ is defined as in (89), (90). Then λ is a proper Logic System.*

Proof. By lemmas 10 and 11, λ gives rise to a well defined Logic System. By Lemma 7, λ is also a proper Logic System. □

The same result (Theorem 10) holds for undirected graphs on m vertices. We simply add the sentence (81) as an additional axiom. Analogous to Propositional Logic, we refer to the Logic System for the First-Order Logic of Directed Graphs described in Theorem 10 as the **standard Logic System for Directed Graphs**, and if we add sentence 81, it is then the **standard Logic System for Undirected Graphs**.

We make three further observations: (1) We could just as well have considered graphs of size less than or equal to a fixed m, rather than just those of size exactly m. (2) There is no set of FO sentences such that if we were to take these sentences as axioms, the set of models would be precisely the set of all finite graphs (a simple consequence of the Compactness Theorem of FOL [67]). (3) We could consider the generic (non-size-limited) theory of graphs and take $\mathcal{M}$ to be the set of all finite graphs, but then there would be some subsets of $\mathcal{M}$ (of countably infinite cardinality) that one could not capture with a sentence from the First-Order language of graphs, and hence we wouldn't be able to establish condition (5) for being a Logic System for such a system.

[8] Though strong completeness can be established considerably more simply in this case via the method of quantifier elimination. See, for example, [40].

6.4.1 First-Order Logic of Directed Graphs: Synopsis of Notation and Terminological Conventions

Suppose we are considering the FOL of Directed Graphs on $\mathtt{m}$ vertices, with vocabulary τ, language $\mathcal{L}$ and $\mathtt{s} \in \mathcal{L}$. Let $\lambda = (L, \mathcal{M}, \kappa, \ell)$ denote the **standard Logic System** for this logic. Then we have the following:

- μ = structure = an $\mathtt{m}$-vertex directed graph.
- $\mathcal{M}$ = set of all $\mathtt{m}$-vertex directed graphs.
- $\kappa(\mathtt{s})$ = "kernel" of the sentence $\mathtt{s}$ = set of all $\mathtt{m}$-vertex directed graphs for which the sentence $\mathtt{s}$ is true.
- $\vec{\kappa}(\mathtt{s})$ = indicator vector representation for the kernel of $\mathtt{s}$. We enumerate the possible directed graphs on the $\mathtt{m}$ vertices and indicate which of the graphs make the sentence $\mathtt{s}$ true using an indicator vector.
- $\mathcal{P}(\mathcal{M})$ = all possible sets of directed $\mathtt{m}$-vertex graphs = all possible kernels. This is the image of the function κ in the definition of a Logic System (Definition 1).

Figure 23 depicts the objects described above for the case of the FOL of Directed Graphs on 2 vertices and the sentence $\mathtt{s} = \forall x(\neg E(x, x))$, which says there are no self-loops.

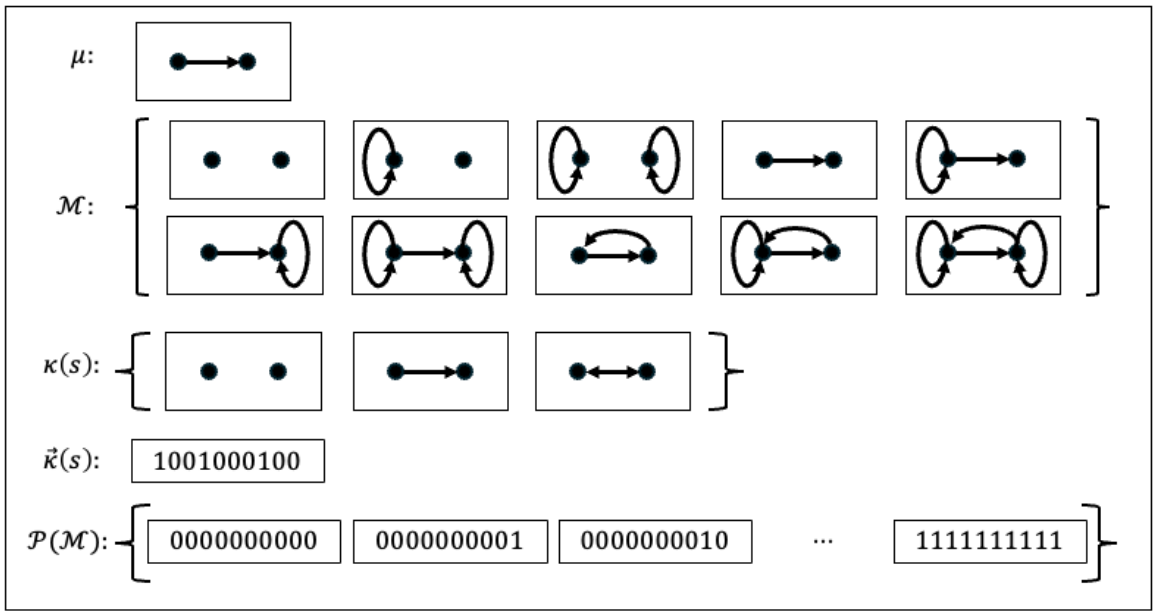


Figure 23: Pictorial representation of the objects $\mu, \mathcal{M}, \kappa(\mathtt{s}), \vec{\kappa}(\mathtt{s})$ and $\mathcal{P}(\mathcal{M})$ for the case of the First-Order Logic of Directed Graphs on 2 nodes. The displayed kernels, $\kappa(\mathtt{s})$ and $\vec{\kappa}(\mathtt{s})$ are for the sentence $\mathtt{s} = \forall x(\neg E(x, x))$.

7 A linkage with polynomial algebra

Thus far we have explored the connections between logic and information theory. In this section we explore a connection to a third area of mathematics, which is the algebra of multivariate polynomials with variables and coefficients belonging to a finite field. The fundamental insight in doing so is that Propositional Logic sentences can be represented using polynomials, which in turn provides us access to powerful mathematical tools such as Gröbner bases. We will then use these mathematical tools to establish a fundamental result on the "optimality of increments"; to paraphrase, synchronizing the knowledge between a receiver and a sender in a communication optimal way can be accomplished by having the receiver add new non-trivial sentences in its knowledge base of logic sentences (without changing its old sentences). Hence, our general approach

consists of first converting the original logic sentences into polynomials, then exploiting the foregoing mathematical tools to perform reduction and decomposition transformations in the polynomial domain, and finally converting the resulting polynomials back to logic expressions. The technique of our general approach presented in this portion of the article is largely independent of the rest, and can be read in isolation. It is also a very general technique that has applicability beyond this article.

Although we speak of a "logic sentence" in possession of the sender Alice or the receiver Bob throughout other parts of the article, we will slightly change this perspective throughout the present section. Instead of talking about a single logic sentence, we will talk about collections of logic sentences that are implicitly combined using conjunction; having explicit smaller logic sentences will significantly help in our exposition herein. In addition, we overload our previous notation and allow $\hat{\mathtt{s}}, \mathtt{r}$ to represent finite sets of logical expressions. Thus, we introduce the notation $\mathcal{P}_F(\cdot)$ to be the collection of all finite subsets of $\mathcal{P}(\cdot)$, and thus $\mathcal{P}_F(\mathcal{L}_m)$ represents the set of finite subsets of Propositional Logic sentences and we assume that $\hat{\mathtt{s}}, \mathtt{r} \in \mathcal{P}_F(\mathcal{L}_m)$. The main setup is that we have a collection of logic sentences $\hat{\mathtt{s}}$ that are being presented to a receiver as a new "truth" that should replace whatever the receiver previously had in his mind, which is the collection of logic sentences $\mathtt{r}$. It is assumed that $\hat{\mathtt{s}} \vdash \mathtt{r}$. The receiver Bob is not keen to completely replace the contents of his mind with the new $\hat{\mathtt{s}}$, and would rather extract from $\hat{\mathtt{s}}$ a $\Delta \in \mathcal{P}_F(\mathcal{L}_m)$ that can be more surgically added to his existing knowledge base $\mathtt{r}$ and that will have the effect that $\Delta \cup \mathtt{r}$ is logically equivalent to $\hat{\mathtt{s}}$ in the sense that the same logic sentences can be proved from either. The receiver would want Δ to have some kind of "minimality" or "orthogonality" with respect to $\mathtt{r}$, in particular he does not want any sentence in Δ to be provable using what he already knew ($\mathtt{r}$). Moreover, we would like to use the receiver's knowledge to reduce the size of the logic expressions in Δ when possible. For measuring size we can assume that logic sentences are expressed as an exclusive-or ($\oplus$) of a collection of conjunctions of logical variables. This representation allows us to preserve size when we convert between logic sentences and polynomials, since the logic operation $\oplus$ corresponds to the polynomial operation $+$ and logical conjunction ($\wedge$) corresponds to polynomial multiplication ($\cdot$).

The ultimate theoretical result for this section in the context of the above setup is our post-processing result with respect to incremental communications at the level of logic expressions, which we present next where the size of a logic expression is its length in characters using the ($\oplus$, $\wedge$) representation.

Theorem 11 (Post-processing at logic level to obtain incremental communications) *There exists a function*

$$\Delta : \mathcal{P}_F(\mathcal{L}_m)^2 \to \mathcal{P}_F(\mathcal{L}_m)$$

such that, for any $\hat{\mathtt{s}}, \mathtt{r}, \mathtt{w} \in \mathcal{P}_F(\mathcal{L}_m)$*, the function satisfies the property that, if* $\mathtt{r} \vdash \mathtt{w}$*, then* $\mathtt{w} \notin \Delta(\hat{\mathtt{s}}, \mathtt{r})$ *and* $\Delta(\hat{\mathtt{s}}, \mathtt{r}) \cup \mathtt{r} \vdash \hat{\mathtt{s}}$ *and* $\hat{\mathtt{s}} \vdash \Delta(\hat{\mathtt{s}}, \mathtt{r}) \cup \mathtt{r}$*. Furthermore, the size of every logic sentence in* $\Delta(\hat{\mathtt{s}}, \mathtt{r})$ *is bounded by* $O(m|\kappa(\mathtt{r})|)$*.*

Example 2 (Illustration of Theorem 11) *Consider a situation where we have three logical variables* $\mathtt{p}$*,* $\mathtt{q}$ *and* $\mathtt{r}$*. Assume the sender wants to send* $\{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$*, but the receiver already knows that* $\mathtt{p} \Rightarrow \mathtt{q}$*. So, taking advantage of the receiver's information, the sender's information can be reduced to* $\{\mathtt{p}, \mathtt{r}\}$*, i.e., the sender's original sentence is a logical consequence of* $\{\mathtt{p}, \mathtt{r}\}$ *using the receiver's knowledge.*

For the simple illustration of Theorem 11 in the above example, the reduction and decomposition steps are obvious. In this section, however, we present our more general approach that can always be used to make these transformations through a sequence of converting logic sentences to polynomials, performing reduction and decomposition transformations in the polynomial domain, and converting the resulting polynomials back to logic expressions. We first present in this context multivariate polynomials with variables and coefficients belonging to a general finite field, which includes defining key mathematical properties and tools, and establishing mathematical results toward the goal of this section with respect to such multivariate polynomials and general finite fields. We then make connections between these multivariate polynomials over general finite fields to corresponding logic sentences over a specific finite field comprising binary truth values in order to formally prove our main post-processing result in Theorem 11. Lastly, we discuss some applications of the incremental communications provided by this theorem, which can be used in settings throughout this article and beyond.

7.1 Multivariate polynomials over finite fields

Throughout our article, we have assumed for simplicity that propositions have binary truth value assignments (truth or false), and we will continue to adopt this perspective here. However, the results and proofs of this subsection will be provided for general finite fields, or Galois fields, denoted by K. The entire section is devoted to setting up our proof of Theorem 11 based on related results within the context of multivariate polynomials with coefficients belonging to a general finite field K. We focus in this subsection on these related results, which require us to establish the machinery of algebras of multivariate polynomials over finite fields, and then return to the proof of Theorem 11 in the next subsection. The set of all polynomials over variables $x_1, \ldots, x_m$ with coefficients in a finite field K is denoted by $K[x_1, \ldots, x_m]$.

Before proceeding to our proofs of the corresponding mathematical results, we provide definitions of key properties employed in some of the proofs. For more information on these mathematical properties, we refer the reader to [21].

Definition 31 (Ideal) *Given polynomials $f_i \in K[x_1, \ldots, x_m]$ for $i = 1, \ldots, n$, the set of all polynomials of the form $\sum_{i=1}^{n} a_i f_i$ with $a_i \in K[x_1, \ldots, x_m]$ is called the* ideal *generated by $f_1, \ldots, f_n$ and denoted by $(f_1, \ldots, f_n)$.*

Since we want to restrict our solutions to lie within the general finite field K of cardinality $|K|$, it is assumed throughout this section that these variables satisfy the conditions

$$x_i^{|K|} - x_i = 0. \tag{91}$$

These conditions are called the *field polynomials*, and they imply that $x_i \in K$.

We had earlier defined the kernel $\kappa(\mathtt{a})$ in propositional logic to be the function that maps each sentence $\mathtt{a} \in \mathcal{L}$ to the set of truth-value assignments that make $\mathtt{a}$ true. With abuse of notation, we define a parallel notion for a polynomial as follows.

Definition 32 *For a given $a \in K[x_1, \ldots, x_m]$, the kernel $\kappa(a)$ is defined as*

$$\kappa(a) \triangleq \{x_1 x_2 \ldots x_m \in K^m : a(x_1, \ldots, x_m) = 0\}.$$

Given an arbitrary subset of K^m, or equivalently, an element of $\mathcal{P}(K^m)$, it is possible to construct a polynomial whose kernel coincides exactly with that subset as is shown in the following result, after presenting a preliminary lemma used in its proof. For consistency with the rest of the paper, when operating on finite sets such as K^m, we will use the notation $\mathcal{P}$ instead of $\mathcal{P}_F$. With abuse of notation, we use ℓ defined earlier for logical expressions to apply herein for polynomials.

Lemma 12 (Product of finite field elements) *The product of all nonzero elements in any finite field is -1.*

Proof. First note that the only elements such that $xx = 1$ are 1 and -1. Thus each element different from 1 and -1 can be paired with its inverse in the product and cancels. In finite fields with an odd number of elements 1 and -1 are different, and so the reduced product is $1(-1) = -1$. In finite fields with an even number of elements $1 = -1$ and the result follows. □

Lemma 13 (Reconstruction of a polynomial from a proposed kernel) *There exists a function $\ell : \mathcal{P}(K^m) \to K[x_1, \ldots, x_m]$ such that, for every set $\psi \subseteq K^m$, $\kappa(\ell(\psi)) = \psi$.*

Proof. Since K is a finite field, $\psi \subseteq K^m$ is a finite set. We first observe that, given a point $p \in K$, there exists a polynomial $I_p \in K[x]$ such that $I_p(p) = 1$ and $I_p(q) = 0$ for all $q \in K$ where $q \neq p$. Recall from (91) that $x^{|K|} - x$ is a field polynomial whose roots are all the elements of K. Then $(x^{|K|} - x)/(x - p)$ is a polynomial that vanishes at all points other than p, but takes the value -1 at p, since its value at p is just the product of all nonzero elements of the finite field and Lemma 12 applies. We define $I_p \triangleq -(x^{|K|} - x)/(x - p)$. Note that when $|K| = 2$, in which case $-1 = 1$, this construction simplifies to $I_p = x + p + 1$. Now let $c = (c_1, \ldots, c_m) \in \psi$; then the polynomial $P_c = \prod_{i=1}^{m} I_{c_i}(x_i)$ takes the value 1 at the point c, and takes the

value 0 at all other points of K^m. Thus the polynomial $\ell(\psi) = -1 + \sum_{c\in\psi} P_c$ takes the value zero at each point of ψ and is nonzero at all other points of K^m. □

Next, we want to be able to reduce a polynomial modulo an ideal representing known information, for which we need a notion of division with remainder for multivariate polynomials. To define the notion of a leading term of a polynomial, we require a total ordering on the monomials in $K[x_1, \ldots, x_m]$ that respects multiplication; i.e., for all monomials m_1, m_2 and m_3 with $1 \leq m_3$, if $m_1 < m_2$, then $m_3 m_1 < m_3 m_2$. Since any monomial can be expressed as $\prod_{i=1}^{m} {x_i}^{e_i}$, one typical ordering is a lexicographic ordering on the exponents e_i. Any such ordering allows us to define the notion of reduction modulo a set of polynomials, together with the related notion of Gröbner bases.

Definition 33 (Polynomial Reduction) *Given polynomials p and q, if some monomial of q is divisible by the leading term of p, we can remove that monomial by subtracting a multiple of p from q. We can continue this process until no monomials of q are divisible by the leading term of p, at which point we say that q is* reduced with respect to p. *Similarly, given a finite collection of polynomials $\hat{p} \triangleq \{p_1, \ldots, p_k\}$, we call a polynomial q* reduced with respect to $\hat{p}$ *if no monomial of q is divisible by a leading monomial of some $p_i \in \hat{p}$. Note that this reduced form with respect to a collection of polynomials can depend on the order reductions are done and thus is not necessarily unique.*

Definition 34 (Gröbner Bases) *Among all sets of polynomials G_I that generate an ideal I, those which have the property that $p \in I$ if and only if p can be reduced to zero by G_I are called a* Gröbner basis *for the ideal I. The notion of Gröbner Basis depends on the notion of the leading term of a polynomial which depends on the choice of monomial ordering. Note that the reduced form modulo a Gröbner basis is uniquely determined.*

Next, we present some key results of ideals which contain all the field polynomials (91).

Lemma 14 (Decomposition of field polynomial ideal) *Let I be the ideal generated by the field polynomials $(x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)$, and let M_i be the collection of ideals of the form $M_i = (x_1 - c_{i,1}, \ldots, x_m - c_{i,m})$ as the coordinates $c_{i,j}$ range over all m-tuples of elements of K. Then, we have $I = \cap M_i$.*

Proof. Recall that $x^{|K|} - x = \prod_{c_i \in K}(x - c_i)$. Since the product of ideals generated by relatively prime polynomials in one variable is the same as their intersection, we have the desired result when $m = 1$. The general case can then be shown by induction on the number of variables. □

Lemma 15 (Kernel property of field polynomial ideal) *Given $v, w \in K[x_1, \ldots, x_m]$, let I be the ideal generated by v and the field polynomials* (91). *Then $w \in I$ if and only if $\kappa(v) \subseteq \kappa(w)$. In particular, if the zeros of v in K have coordinates $c_{i,1}, \ldots, c_{i,m}$, then $I = \cap(x_1 - c_{i,1}, \ldots, x_m - c_{i,m})$.*

Proof. The second claim is shown by adding v to the ideal decomposition in Lemma 14. The first claim is an application of the second, since $w \in I$ if and only if w is contained in each component ideal of I if and only if $\kappa(v) \subseteq \kappa(w)$. □

Exploiting these key properties, we now prove our post-processing result with respect to incremental communications at the level of multivariate polynomials over general finite fields.

Lemma 16 (Post-processing at polynomial level to obtain incremental communications) *There exists a function*

$$\Delta : \mathcal{P}_F(K[x_1, \ldots, x_m])^2 \to \mathcal{P}_F(K[x_1, \ldots, x_m])$$

such that, for any $u, v, w \in \mathcal{P}_F(K[x_1, \ldots, x_m])$, the function satisfies the property that, if $\kappa(v) \subseteq \kappa(w)$, then $w \notin \Delta(u, v)$ and $\kappa(\Delta(u, v) \cup v) = \kappa(u \cup v)$. Furthermore, the polynomials in $\Delta(u, v)$ are reduced with respect to a Gröbner basis for the ideal generated by v and the field polynomials (91).

Proof. We seek to remove, from u, any polynomials w that satisfy the supposition $\kappa(v) \subseteq \kappa(w)$. First, we observe that $\kappa(v) \subseteq \kappa(w)$ if and only if w is contained in the ideal $(v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)$ by Lemma 15.

Let G_v be a Gröbner basis for the ideal $(v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)$, and define $\Delta(u, v)$ to be the reduction of the elements of u with respect to G_v. The reduction process guarantees that all polynomials w such that $\kappa(v) \subseteq \kappa(w)$ will reduce to zero, and thus they are no longer contained in $\Delta(u, v)$, i.e., $w \notin \Delta(u, v)$. Since all solutions of v lie in K^m, a vector space of dimension m over K, we obtain $\kappa(v) = \kappa(v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)$, and therefore $\kappa(u \cup (v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)) = \kappa(u \cup v)$. But $\Delta(u, v)$ and u are the same modulo $(v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)$, and hence we have

$$\begin{aligned} \kappa(\Delta(u, v) \cup v) &= \kappa(\Delta(u, v) \cup (v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)) \\ &= \kappa(u \cup (v, x_1^{|K|} - x_1, \ldots, x_m^{|K|} - x_m)) \\ &= \kappa(u \cup v). \end{aligned}$$

By construction, the polynomials in $\Delta(u, v)$ are reduced with respect to the ideal generated by v and the field polynomials. □

Now, we provide an explicit bound on the number of monomials in a reduced polynomial.

Lemma 17 (Bound on size of reduced polynomials) *Given $v, w \in K[x_1, \ldots, x_m]$, the number of monomials in the reduced representation of w with respect to the ideal generated by v and the field polynomials* (91) *is bounded by $|\kappa(v)|$.*

Proof. Let I be the ideal generated by v and the field polynomials. The reduction mapping sends w to $K[x_1, \ldots, x_m]/I$, a finite dimensional vector space over K, which we can assume is generated by monomials. Let $z_1, \ldots, z_n$ represent the set of zeros of v in K. For each zero $z_i = c_{i,1}, \ldots, c_{i,m}$, let M_i be the ideal generated by $(x_1 - c_{i,1}, \ldots, x_m - c_{i,m})$, and thus by Lemma 15 we obtain $I = \cap_{i=1}^{n} M_i$. Since $\dim(K[x_1, \ldots, x_m]/M_i) = 1$, we have that $\dim(K[x_1, \ldots, x_m]/I) = n$, which is the number of zeros of I, i.e., $|\kappa(v)|$. □

Finally, we show that polynomials over an arbitrary finite field can be considered to be a proper Logic System, i.e., they satisfy Definition 1 and Definition 2.

Lemma 18 (Proper logic of polynomials) *The set of all polynomials over variables $x_1, \ldots, x_m$ with coefficients in the general finite field K, i.e., $K[x_1, \ldots, x_m]$, forms a Proper Logic system as specified by Definitions 1 and 2.*

Proof. Given polynomials s and t, we say that $s \vdash t$ if and only if t is contained in the ideal generated by s and the field polynomials (91). Lemma 13 shows the existence of ℓ in Definition 1 (c.f. Definition 28), and Lemma 15 shows that condition (6) of Definition 1 holds.

We define the meaning of the operators $\vee, \wedge, \neg$ for polynomials using the following formulas with respect to ℓ and κ:

1. $s \vee t = \ell(\kappa(s) \cup \kappa(t))$;
2. $s \wedge t = \ell(\kappa(s) \cap \kappa(t))$;
3. $\neg s = \ell(\kappa(s)^c)$.

Then, applying κ to both sides of these three equations and using condition (5) of Definition 1, we see that Definition 2 holds and the desired result follows. Note that a simpler definition of the operator $\vee$ is $s \vee t = st$, since $\kappa(st) = \kappa(s) \cup \kappa(t)$. □

7.2 Propositional Logic sentences

We now return to Propositional Logic sentences and the special case $K = GF(2)$, the finite field of size 2, which comprises the binary alphabet $\{0, 1\}$ together with multiplication $(\cdot)$ and addition $(+)$ corresponding to the logic operators $\wedge$ and $\oplus$, respectively. In this case, we associate 0 with the logic value false and 1 with the logic value true, and thus the variables $x_1, \ldots, x_m$ denote whether the corresponding properties are false or true. Throughout this subsection, all references to logic are intended to mean Propositional Logic.

Any logic sentence in $\mathcal{L}_m$ can be written as an equation involving a polynomial in $GF(2)[x_1, \ldots, x_m]$ and vice versa. To establish this, we will rely on Table 3. We assume that a logic sentence in $\mathcal{L}_m$ makes use of parentheses to ensure that at most two operands are clearly associated with any operation; then the logic sentence may be parsed so as to obtain a tree representation, where every node denotes an operation from the list $\{\neg, \vee, \wedge, \oplus, \Rightarrow\}$ and where the branches flowing downwards from the node (one or two, depending on the operator) represent the operands being passed to the operator. The tree is unique as per our earlier assumption that parentheses have been used to eliminate any possible ambiguity. Obtaining a polynomial equation representation for this logic sentence can be done through the following four steps:

1. Replacing every symbol $\mathtt{X}_i$ with its corresponding variable x_i.
2. Replacing every operator with operands fully described in terms of variables from $\{x_1, \ldots, x_m\}$ with the corresponding mathematical expression as described in Table 3, repeating until all operators have been replaced.
3. Adding 1 to the resulting polynomial expression. The reason for this is that we want the kernel of the polynomial form to correspond to its set of zeros, not its set of ones.
4. Equating to zero the resulting expression.

logic form	polynomial form
$\neg\mathtt{a}$	$a+1$
$\mathtt{a} \vee \mathtt{b}$	$a+b+a\cdot b$
$\mathtt{a} \wedge \mathtt{b}$	$a\cdot b$
$\mathtt{a} \oplus \mathtt{b}$	$a+b$
$\mathtt{a} \Rightarrow \mathtt{b}$	$a\cdot(1+b)+1$

Table 3: Conversion between logic sentences and polynomial expressions. Here $\mathtt{a}, \mathtt{b}$ denote logic sentences and a, b their corresponding polynomial expressions.

As a simple example to illustrate this, the translation of the truth of the logic sentence that "$(\mathtt{X}_2 \Rightarrow \mathtt{X}_1) \wedge \neg \mathtt{X}_3$" is given by $(x_2 \cdot (1 + x_1) + 1) \cdot (1 + x_3) + 1 = 0$. Obtaining an expression in $\mathcal{L}_m$ from a polynomial in $GF(2)[x_1, \ldots, x_m]$ (which is assumed to have been equated to zero) can be done similarly. In this case, we assume only two arithmetic operators are present $\{\cdot, +\}$, which are simply swapped with the logic operators $\{\wedge, \oplus\}$.

For notational purposes, we adopt the following convention.

Convention 1 *For a given logic sentence $\mathtt{a} \in \mathcal{L}_m$, the corresponding polynomial is $a \in GF(2)[x_1, \ldots, x_m]$, and vice versa. We will use the notation $Ł(a) = \mathtt{a}$. In contrast to the use of the bold Courier font for logic sentences throughout the article, we use the Times New Roman font in this section for the corresponding polynomials.*

The following basic result helps us transition between logic expressions and polynomials, thus providing a parallel with Lemma 8.

Lemma 19 (Duality of kernels) *For a given $\mathtt{a} \in \mathcal{L}_m$ and its corresponding polynomial $a \in GF(2)[x_1, \ldots, x_m]$, we have $\kappa(\mathtt{a}) = \kappa(a)$. In particular, $\mathtt{a} \vdash \mathtt{b}$ if and only if $\kappa(a) \subseteq \kappa(b)$.*

Proof. The first statement follows from the property that the mapping between logic sentences and polynomials (given by the four steps above) sends points where the logic sentence is true to points where the associated polynomial is zero (false). The second result then follows from Lemma 8. □

It is important to note that we can use ideal membership to decide whether $\mathtt{a} \vdash \mathtt{b}$. From Lemma 15, given $a, b \in GF(2)[x_1, \ldots, x_m]$, we have that $Ł(a) \vdash Ł(b)$ if and only if $b \in (a, x_1^2 - x_1, \ldots, x_m^2 - x_m)$. More generally, when $f_1, \ldots, f_n$ are the polynomial representations of the logic expressions $\mathtt{f}_1, \ldots, \mathtt{f}_1$, then $g \in (f_1, \ldots, f_n)$ shows $Ł(g)$ will be a logical consequence of $\{\mathtt{f}_1, \ldots, \mathtt{f}_n\}$.

Using Lemma 13 and converting the constructed polynomial back to a logic formula, we immediately obtain the following result.

Lemma 20 (Reconstruction of a logic sentence from a proposed kernel) *There exists a function* $\ell : \mathcal{P}(K^m) \to \mathcal{L}_m$ *such that, for every set* $\psi \subseteq K^m$, $\kappa(\ell(\psi)) = \psi$.

Finally, building on the above results at the polynomial level, we now prove our main post-processing result with respect to incremental communications at the logic level, which we restate for convenience.

Theorem 11 (Post-processing at logic level to obtain incremental communications) *There exists a function*

$$\Delta : \mathcal{P}_F(\mathcal{L}_m)^2 \to \mathcal{P}_F(\mathcal{L}_m)$$

such that, for any $\mathtt{s}, \mathtt{r}, \mathtt{w} \in \mathcal{P}_F(\mathcal{L}_m)$, *the function satisfies the property that, if* $\mathtt{r} \vdash \mathtt{w}$, *then* $\mathtt{w} \notin \Delta(\hat{\mathtt{s}}, \mathtt{r})$ *and* $\Delta(\hat{\mathtt{s}}, \mathtt{r}) \cup \mathtt{r} \vdash \hat{\mathtt{s}}$ *and* $\hat{\mathtt{s}} \vdash \Delta(\hat{\mathtt{s}}, \mathtt{r}) \cup \mathtt{r}$. *Furthermore, the size of every logic sentence in* $\Delta(\hat{\mathtt{s}}, \mathtt{r})$ *is bounded by* $O(m|\kappa(\mathtt{r})|)$.

Proof. Using the transformations provided in Table 3, we can convert all our logic expressions to polynomials over $GF(2)$. We can then use the construction in Lemma 16 to compute $\Delta(\mathtt{s}, \mathtt{r})$ in polynomial form. Since the multiplication $(\cdot)$ and addition $(+)$ operations on polynomials over $GF(2)$ correspond to the logic operations $\oplus$ and $\wedge$, respectively, each monomial corresponds to a conjunction of variables and the number of monomials in our polynomials is the same as the number of conjuncts in our logic expressions. Hence, the reduction of $\Delta(u, v)$ with respect to the ideal generated by v and the field polynomials established in Lemma 16 implies a corresponding reduction of logic formulas in $\Delta(\mathtt{s}, \mathtt{r})$ with respect to the logic information contained in $\mathtt{r}$. For each monomial a, $Ł(a)$ is a logic conjunction involving at most m variables. Hence, the bound $|\kappa(v)|$ on the number of monomials in a reduced polynomial yields the bound $O(m|\kappa(v)|)$ on the size of a reduced logic expression.

If $\mathtt{r} \vdash \mathtt{w}$, then the polynomial corresponding to $\mathtt{w}$ is contained in the ideal generated by the polynomials corresponding to $\mathtt{r}$ and the field polynomials (91), and thus it can be reduced to zero; moreover, as in Lemma 16, we have that $\mathtt{w} \notin \Delta(\mathtt{s}, \mathtt{r})$. Finally, the same lemma shows that $\kappa(\Delta(\mathtt{s}, \mathtt{r}) \cup \mathtt{r}) = \kappa(\mathtt{s} \cup \mathtt{r})$. □

Example 3 (Illustration of Lemma 16 and Theorem 11) *Within the context of Example 2 above, the polynomial version of the sender's sentence,* $\mathtt{p} \wedge \mathtt{q} \wedge \mathtt{r}$, *is* $pqr + 1$ *using Table 3 and adding 1 as specified in step 3 above. The polynomial version of the receiver's knowledge,* $\mathtt{p} \Rightarrow \mathtt{q}$, *is* $p(1 + q) + 1 + 1$ *or simply* $p(1 + q)$ *since* $1 + 1 = 0$ *in* $Z/2$, *again using Table 3 and complementing. Since we are operating over* $Z/2$, *the field equations associated with the polynomial variables* p, q, r *are* $\{p^2 + p, q^2 + q, r^2 + r\}$. *The Gröbner basis for the ideal of the receiver's knowledge and field equations is* $G_v = (pq + p, p^2 + p, q^2 + q, r^2 + r)$. *If we reduce* $pqr + 1$ *by* G_v *we obtain* $pr + 1$, *which is simply the remainder after dividing by* $pq + p$. *Hence, in this case, the function* Δ *returns* $(pr + 1)$. *One can also take an additional step to replace* Δ *with a collection of smaller polynomials. If we next compute a Gröbner basis for* $pr + 1$ *along with the field equations, we obtain* $(p + 1, r + 1, q^2 + q)$. *The polynomial* $q^2 + q$ *vanishes on both elements of* $Z/2$, *and thus corresponds to a logical tautology and can be omitted. We therefore have a reduced* Δ *of the form* $(p+1, r+1)$, *which converts to the set of logic expressions* $\{\mathtt{p}, \mathtt{r}\}$ *by complementing and reducing* $1 + 1$ *to* 0.

7.3 Applications of Incremental Communications

The construction presented in Theorem 11 can then be used in any number of settings; in this article, it is relevant to Theorems 3, 7 and 8. As a representative example, we illustrate in Figures 24 and 25 how the device implied by Theorem 11 is used in the case of Theorem 3. In Figure 24 we repeat the communication diagram associated with Theorem 3 and below it we demonstrate an alternative, in principle more restrictive, setup where the goal of the decoder g is to produce an increment that needs to be incorporated into the existing knowledge base of logic sentences $\mathtt{r}$. As suggested by the achievable Shannon limit column on the right, the fundamental limits in both setups are exactly the same; that is, the restriction introduced by the incremental communication requirement does NOT make the Shannon limit worse. The fact that this is the case is a simple corollary of the upper bound in Theorem 3 coupled with Theorem 11, and thus no further formal statement or proof is given; instead we refer the reader to Figure 25 which is an updated version of Figure 14 where the algebraic reduction step is implemented by Theorem 3.

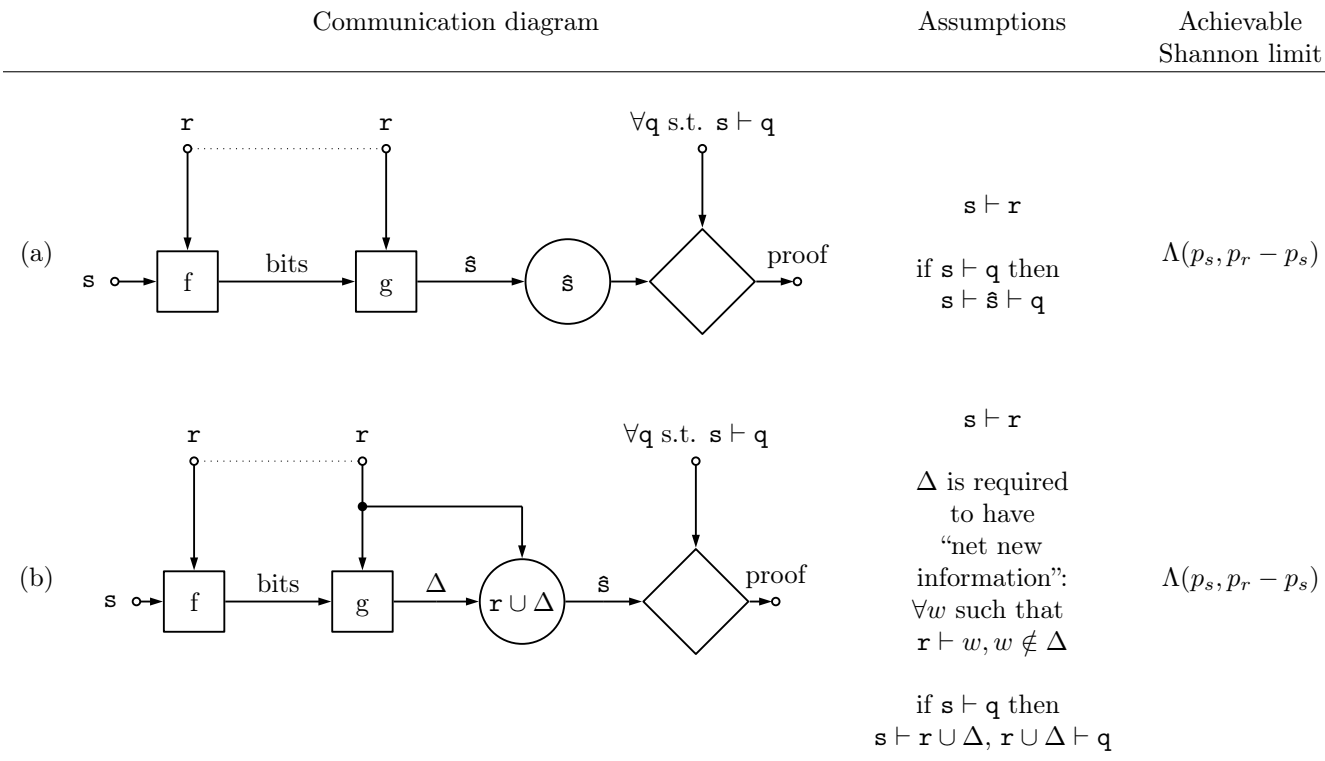


Figure 24: Motivation for incremental communications.

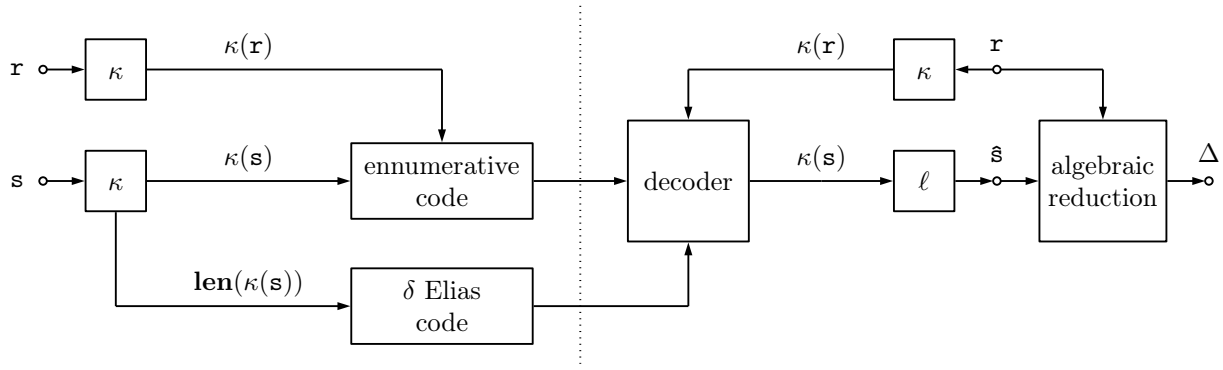


Figure 25: Proof strategy for adding incremental communications to the setup of Theorem 3, with ℓ from Lemma 20 and Δ from Theorem 11.

8 Speculative Future Directions

The fact that logic is so foundational and deeply intertwined in computer science and mathematics leads one to be able to imagine many possibilities. In this section, we will allow ourselves to speculate on some ways that this work might be elaborated upon to address various important topics, in the hopes of inspiring future authors. We discuss these in four subsections: practical methods and applications; extensions of the core setup, including the logic foundations; additional social scenarios; and, lastly, other perspectives on semantic information not treated in our article.

8.1 Practical methods and applications

Improvements on practical coding techniques. In our article we introduced practical methods, based on both linear and nonlinear codes, for both the "less is more" and "no need to know" scenarios. These codes nonetheless are optimal in the sense of approaching the Shannon bounds only for a very limited set of parameters for these scenarios. It is an open problem to design optimal codes for even the scenarios introduced in our article, let alone other variants as discussed in this section.

Artificial intelligence. Some practical usage possibilities lie in AI, where logic has long provided one of its most fundamental representations, for expressing human knowledge and allowing reasoning upon it [62]. A direct practical realization of the situation described above is possible due to the steady rise over the years in the efficacy of the technologies of *semantic parsing* [43], or translating natural language sentences into logic sentences which represent their underlying meaning, *e.g.*, [1]. The canonical situation above is motivated by the long-standing idea of a *theory of mind* [23], a model of the mind of an agent in terms of his/her knowledge (and possibly goals) in terms of logic sentences. In this context, the ability to determine the most informative logic sentence to send would provide a solution to the question *what is the next best thing to say?* to a formal and quantifiable degree for which we know no equivalent in AI to date. This would lie in sharp contrast to the lack of intentionality in current approaches to text generation based on large language models, which represent a kind of stream of consciousness-like random walk that generates statistically likely combinations of words, though progress in prompting technologies can make the output appear more intentional [31].

Machine learning has a rich landscape of logic-based models, including decision trees [13] and the emerging area of neuro-symbolic AI [18]. [16] calls for a new brand of "semantic machine learning" that has a world model, which is one of the emphases within neuro-symbolic AI. Information theory provides key tools for learning theory [35], suggesting that these results could be used to deeply characterize the effect of logically-expressed knowledge on sample complexity. [11] shows a use of semantic information to characterize logical expressivity in a neural network.

Data transmission and compression. As mentioned earlier, a recent batch of vision papers such as [60] have argued for the great potential of "much lower data transmission requirements" [54] held by semantic approaches to communication, typically sketching nominal communication architectures with additional semantic components [16]. This has been fueled by some theory showing that focusing on preserving meaning has the potential for savings over focusing on preserving bits, notably [6], which also showed some empirical gains in a limited setting.

As part of this recent surge of interest in semantic communication ideas, a number of approaches have rushed to show empirical gains, as described in many surveys such as [76]. A prominent thread showing some gains using deep learning is exemplified by [75]. Some approaches such as [51], leveraging the intuition that one does not need to get the bits right to get the meaning right, have shown some gains particularly in the low-SNR regime.

With our approach, one can imagine potential direct impact for at least compression scenarios involving logic or programs, both in software and hardware. The extent to which the presumed gains can transfer to other forms of data depends on their ease of translation to equivalent forms in terms of some form of logic or program (see additional discussion below).

Automated theorem proving and mathematics. The process of reasoning, or automated theorem proving [58], itself contains a difficult problem, premise selection (or which logic sentence should be operated upon next, by an inference rule, in order to prove the desired sentence) or proof guidance in general, which could perhaps be aided by a time-efficient or approximate version of our model scenario.

8.2 Core setup

More expressive logics. As one can see, our conditions for treatable logic systems are fairly general. This includes First-Order Logic of structures of fixed finite sizes, or of structures of sizes up to a given maximum size. It also includes the All Positive First-Order Logic for structures of fixed size / up to a given fixed size.

Future researchers may consider extensions to First-Order Logic with Counting, Multi-Valued and Probabilistic Logics such as [53, 22, 28], upon which many of the techniques of the fast-emerging area of neuro-symbolic AI are founded, and Second and Higher-order logics. Extension to Higher-order logics has a particular significance with respect to programs, discussed next.

Programs. Algorithmic information theory (AIT), whose roots were laid in the 1960s [48], generalizes information theory and relates it to computation. The *algorithmic complexity* or *Kolmogorov complexity* of a string x is defined as the length of the shortest program that computes or outputs x, where the program is run on some fixed reference universal computer. AIT's powerful ideas have long sparked the imagination around many possibilities for practical implementations of the theory. However, AIT's core concept, the length of the shortest program, is incomputable, making AIT largely impractical. Though various schemes exist for approximating the quantity, such as Levin search [47] and its later improvements, they are generally still not efficient enough for practical usage, and may yield programs far from the theory's bound.

We suggest that generalizing semantic information theory to a semantic AIT could open new doors. Noting the Curry-Howard isomorphism [41], which states an equivalence between programs and (higher-order) logic, this might bear some resemblance to the current notion of sending the minimum amount of logic (in the form of bits) to the receiver. One can similarly imagine sending program statements to allow the receiver's existing program to be extended to compute results of interest. A notion that could correspond to that of the information value of a logical sentence that can already be deduced given existing logic statements being zero is one of the information value of a computation that can already be performed given existing program elements being zero. In this case, we would invoke formal notions of *program semantics*, which are fully grounded in logic, going back to the work of Hoare [39]. Some of the ideas of [42] may be relevant here.

In the same setting as standard information theory, for a given distribution, AIT achieves exactly the same bounds. We have shown that a semantic information theory can provide increased compression over such bounds. This is a consequence of AIT being ultimately semantics-free, as is standard information theory. We have also shown practically computable codes which achieve our stated bounds. It is thus possible that semantic information theory applied to programs could, for some purposes, serve as a practically computable version of Kolmogorov complexity. Further, it can be thought of as extending the idea of program shortness to account for the amount of program capability (e.g., libraries of functions) that the receiver already has.

Reasoning power and computation. The assumption that the dialog counterpart has a reasoning engine at hand accounts for intelligence on the part of the recipient – preventing the need for the speaker or teacher to communicate things that will be obvious since they can be deduced from the recipient's existing knowledge. However, the computational cost needed to make those deductions is not accounted for in the current model, while in more complex instances, it may be more realistic not to assume the receiver can always perform any reasoning needed. For example, one could account for limited reasoning time/resources, as in bounded rationality [32] or teaching a child.

Sharper results for more general distributions. Early in our paper (see the expression (4) and Theorem 2) we remarked that the optimal code length to fully convey what a sentence represents is given by $-\log_2 P(\kappa(\mathtt{s}))$. Our results are mostly focused nonetheless on kernel sizes, which give us a simpler, but coarser theory. What if there are some structures in $\mathcal{M}$ that are more likely than others, or what if there are correlations in their occurrences? What if the number of structures that model a logic sentence is so small that effectively we are in the setting where the parameters p_s, p_r, p_q are zero? In all these scenarios, a much more refined theory is necessary.

Uncertainty in logic sentences. Throughout the article, we have assumed that Alice's logic sentence is believed to be true in all cases. Similarly Bob's sentence is believed to be true in most cases except the ones where we explicitly state that there could be conflict or misinformation. But what if this belief was only partial? How would this change the results and algorithms?

8.3 Additional scenarios

Throughout our paper, we mostly centered around the concept that Alice holds some version of truth that Bob wants to take advantage of. In what follows we postulate other scenarios that are interesting in their own right.

Unawareness of possible conflict. In the situation modeled by Theorem 3, Alice's sentence does not necessarily entail Bob's (there's a possible conflict), however Alice is fully aware of what sentence Bob has. The proposed extension lifts this last assumption – what is the total minimum expected communication cost?

Collaboration. Here Alice and Bob each have a sentence that is presumed true but unknown to each other and they wish to pool their logic sentences with a minimal amount of total communication. This type of extension to our problem appears to be directly in line with problems studied under communication complexity [82] and interactive communication [57].

Adversarialness and disinformation. Bob, who holds a sentence he believes to be true, does not necessarily trust Alice, who may or may not hold a true sentence and is unaware of what Bob knows. Assuming Bob's goal is to sharpen his sentence without accepting something that is not true, and Alice's goal is to have Bob accept something that is not true, what are good strategies for either?

Consultancy. In our current model, at communication time Alice is provided the query. What if Bob is the one given the query, instead of Alice? How does that change the problem?

Teaching. What would be the solution if there are multiple Bobs and one Alice, and a single message will be transmitted to all Bobs? Building on logical theories of teaching/learning such as [38], this work may enable the practical implementation of a sense of optimal (or at least principled) teaching in the long-standing area of intelligent tutoring systems [3]. One could imagine its use at the stage of curricular design or even at the granular level during live interaction, particularly in combination with a logical model of natural language (described next).

Scientific inquiry. Scientific inquiry can be thought of asking questions of nature [37], suggesting analogous usage as a basis for optimal/principled automated scientific experimentation [2]. Extensions of our model to account for query design could help with some of these problems.

8.4 Other perspectives on semantic information

Foundations of probability and information theory. Ellerman [27] observes a duality that can be stated using the deep relationships between logic and probability. He examines the mathematical foundations of information and constructs a theory around the concept of "information as distinctions"; Ellerman goes on to define "logic entropy" and shows how Shannon's entropy can be seen as a special case of logic entropy. At a high level, our conclusions are natural implications of the ability of logic to compactly describe sets. It can be shown that standard information theory can be derived from this logical standpoint. In fact, our framework can be related to Shannon's own work on "lattice information theory" [63] which similarly describes a more general abstract formalism for information theory from the starting point of discrete sets, though it did not connect to logic.

Alternative treatments of semantic information. Because our line of thinking is unique in utilizing the power of deductive reasoning, it is likely that parallel efforts to treat semantics in information theory are complementary and that multiple such ideas can be leveraged in combination for even greater efficiencies.

A category of alternative treatments are those that are motivated by philosophical or qualitative considerations. These approaches generally seek definitions of semantic information which seem as psychologically natural as possible. Broadly speaking, they tend not to arrive at a well-defined metric upon which a semantic communication theory can be built, which is our sole focus in this paper, and we do not address philosophical concerns in this paper. The most cited critique of the Carnap and Bar-Hillel conceptualization is that of Floridi [30] who addresses the putative paradox in Carnap and Bar-Hillel's theory, contradictory sentences have a very high amount of semantic information, and proposes an alternate theory. However, it requires a reference statement to which the statement of interest can be compared in similarity, and cannot quantify an amount of information without one, making it difficult to formalize as quantitative machinery. Perhaps the second most prominent alternate proposal surrounds the idea of truthlikeness [25], which also does not lend itself easily to quantitative formalization. Their approaches are rooted in the framework using information flow and situation theory of Barwise and Seligman [9] and that of Devlin based on the idea

that information can be studied through empirical observation methods as one does in physical sciences, leading to an extension to First-Order logic based on his own observations about how information appears to behave in a variety of circumstances [24]. Through our non-philosophical lens, one could argue that it is in fact practically very informative to observe a contradiction. With the full admission of naivete in matters of philosophy, we leave as an open question whether our work might inspire further work in philosophical directions.

9 Concluding remarks

This article introduces what we believe are the first collection of sharp results on an information theory for the communication of logic sentences under with an assumption of a deductive mechanism at the receiver side, including a rigorous development of the type logics for which our results apply. A diverse set of communication situations are treated in this work, including settings where the goal of the communication is to efficiently allow a receiver to deduce all or a subset of the logic statements that a sender can deduce, as well as settings where a receiver may already be in possession of a related logic statement which the sender may or may not be aware of. Practical codes based on linear coding techniques are developed and experimental results are offered demonstrating potential significant gains compared to classical communication techniques.

Acknowledgements

The authors acknowledge helpful conversations with the following individuals: Ron Fagin, Phokion Kolaitis, Jason Rute, Kush Varshney and Mark Wegman. Work of W. Szpankowski was partially supported by the NSF Center for Science of Information (CSoI) Grant CCF-0939370, and also by NSF Grants CCF-2006440, and CCF-2211423.

References

[1] I. Abdelaziz, S. Ravishankar, P. Kapanipathi, S. Roukos, and A. Gray. A Semantic Parsing and Reasoning-Based Approach to Knowledge Base Question Answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 of *AAAI 2021*, pages 15985–15987, May 2021.

[2] D. Adam. The automated lab of tomorrow. *Proceedings of the National Academy of Sciences*, 121(17):e2406320121, 2024.

[3] J. K. Ali Alkhatlan. Intelligent tutoring systems: A comprehensive historical survey with recent developments. *International Journal of Computer Applications*, 181(43):1–20, Mar 2019.

[4] J. Bao, P. Basu, M. Dean, C. Partridge, A. Swami, W. Leland, and J. A. Hendler. Towards a theory of semantic communication. In *IEEE Network Science Workshop*, pages 110–117, Los Alamitos, CA, USA, jun 2011. IEEE Computer Society.

[5] J. Bao, P. Basu, M. Dean, C. Partridge, A. Swami, W. Leland, and J. A. Hendler. Towards a theory of semantic communication. In *Extended Technical Report*, 2011.

[6] J. Bao, P. Basu, M. Dean, C. Partridge, A. Swami, W. Leland, and J. A. Hendler. Towards a theory of semantic communication. In *IEEE Network Science Workshop*, pages 110–117, Los Alamitos, CA, USA, jun 2011. IEEE Computer Society.

[7] Y. Bar-Hillel and R. Carnap. Semantic information. *The British J. Philosophy of Science*, 4:147–157, 1953.

[8] J. Barwise. Model theoretic logics: Concepts and aims. In J. Barwise and S. Feferman, editors, *Model Theoretic Logics*, pages 1–24. Cambridge University Press, 1985.

[9] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.

[10] P. Basu, J. Bao, M. Dean, and J. Hendler. Preserving quality of information by using semantic relationships. *Pervasive and Mobile Computing*, 11:188–202, 2014.

[11] J.-C. Belfiore, D. Bennequin, and X. Giraud. Logical information cells i, 2021.

[12] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall electrical engineering series. Prentice-Hall, 1971.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[14] Y. Breitbart, H. Hunt, and D. Rosenkrantz. On the size of binary decision diagrams representing boolean functions. *Theoretical Computer Science*, 145(1):45–69, 1995.

[15] S. Buss. *Handbook of Proof Theory*, chapter An introduction to proof theory, pages 1–78. Elsevier, New York, 1998.

[16] E. Calvanese Strinati and S. Barbarossa. 6G networks: Beyond shannon towards semantic and goal-oriented communications. *Computer Networks*, 190:107930, 2021.

[17] R. Carnap. *Introduction to Semantics*. Harvard University Press, Cambridge, 1942.

[18] Centaur AI Institute. 3rd Neuro-Symbolic AI Summer School. `https://neurosymbolic.github.io/nsss2024`, 2024.

[19] S. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

[20] T. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1):73–77, 1973.

[21] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2015.

[22] F. Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

[23] F. Cuzzolin, A. Morelli, B.-I. Cîrstea, and B. Sahakian. Knowing me, knowing you: Theory of mind in ai. *Psychological Medicine*, 50:1057–1061, 04 2020.

[24] K. Devlin. *Logic and Information*. Cambridge University Press, 1991.

[25] S. D'Alfonso. On quantifying semantic information. *Information*, 2(1):61–101, 2011.

[26] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

[27] D. Ellerman. *New Foundations for Information Theory: Logical Entropy and Shannon Entropy*. Springer, 2021.

[28] R. Fagin, R. Riegel, and A. Gray. Foundations of reasoning with uncertainty via real-valued logics. *Proceedings of the National Academy of Sciences*, 121(21), 2024.

[29] R. Feynman, R. Leighton, M. Sands, and E. Hafner. *The Feynman Lectures on Physics; Vol. I*. Addison-Wesley, 1965.

[30] L. Floridi. Outline of a theory of strongly semantic information. *Minds and Machines*, 14, 05 2004.

[31] J. Grindrod. Large language models and linguistic intentionality. *Synthese*, 204(2):71, 2024.

[32] B. Grosof and T. Swift. Radial restraint: a semantically clean approach to bounded rationality for logic programs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, page 379–386. AAAI Press, 2013.

[33] T. Guo, Y. Wang, J. Han, H. Wu, B. Bai, and W. Han. Semantic compression with side information: A rate-distortion perspective, 2022.

[34] D. Gündüz, Z. Qin, I. E. Aguerri, H. S. Dhillon, Z. Yang, A. Yener, K. K. Wong, and C.-B. Chae. Beyond transmitting bits: Context, semantics, and task-oriented communications. *IEEE Journal on Selected Areas in Communications*, 41(1):5–41, 2023.

[35] F. Hellström, G. Durisi, B. Guedj, and M. Raginsky. Generalization bounds: Perspectives from information theory and pac-bayes. *ArXiv*, abs/2309.04381, 2023.

[36] A. Heyting. *Intuitionism: An Introduction*. North-Holland Pub. Co., Amsterdam,, 1956.

[37] J. Hintikka. On the logic of an interrogative model of scientific inquiry. *Synthese*, 47(1):69–83, 1981.

[38] J. Hintikka. A dialogical model of teaching. *Synthese*, 51(1):39–59, 1982.

[39] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, Oct. 1969.

[40] W. Hodges. *A Shorter Model Theory*. Cambridge University Press, 1997.

[41] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. original paper manuscript dated 1969.

[42] B. Juba and M. Sudan. Universal semantic communication I. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 123–132, New York, NY, USA, 2008. Association for Computing Machinery.

[43] A. Kamath and R. Das. A survey on semantic parsing, 2019.

[44] J. Konorski and W. Szpankowski. What is information? *2008 IEEE Information Theory Workshop*, pages 269–270, 2008.

[45] D. Kozen. Positive first-order logic is NP-complete. *IBM Journal of Research and Development*, 25(4):327–332, 1981.

[46] S. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[47] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):115–116, 1973.

[48] M. Li, P. Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

[49] J. Liu, S. Shao, W. Zhang, and H. V. Poor. An indirect rate-distortion characterization for semantic sources: General model and the case of gaussian observation. *IEEE Transactions on Communications*, 70(9):5946–5959, 2022.

[50] J. Liu, W. Zhang, and H. V. Poor. A rate-distortion framework for characterizing semantic information. In *IEEE International Symposium on Information Theory*, 05 2021.

[51] Y. Liu, Y. Zhang, P. Luo, S. Jiang, K. Cao, H. Zhao, and J. Wei. Enhancing communication reliability from the semantic level under low snr. *Electronics*, 2022.

[52] D. Mehta and V. Raghavan. Decision tree approximations of boolean functions. *Theoretical Computer Science*, 270(1):609–623, 2002.

[53] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[54] K. Niu, J. Dai, S. Yao, S. Wang, Z. Si, X. Qin, and P. Zhang. A paradigm shift toward semantic communications. *IEEE Communications Magazine*, 60(11):113–119, 2022.

[55] K. Niu and P. Zhang. A mathematical theory of semantic communication, 2024.

[56] R. O'Donnell, M. Saks, O. Schramm, and R. Servedio. Every decision tree has an influential variable. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 31–39, 2005.

[57] A. Orlitsky. Interactive communication: balanced distributions, correlated files, and average-case complexity. In *Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 228–238, 1991.

[58] M. Pantsar. Theorem proving in artificial neural networks: new frontiers in mathematical ai. *European Journal for Philosophy of Science*, 14(1):4, 2024.

[59] C. H. Papadimitriou and M. Sipser. Communication complexity. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 196–200, New York, NY, USA, 1982. Association for Computing Machinery.

[60] P. Popovski, O. Simeone, F. Boccardi, D. Gündüz, and O. Sahin. Semantic-effectiveness filtering and control for post-5g wireless connectivity. *Journal of the Indian Institute of Science*, 100:435–443, 2019.

[61] J. Riordan and C. E. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21(1-4):83–93, 1942.

[62] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[63] C. Shannon. The lattice theory of information. *Transactions of the IRE Professional Group on Information Theory*, 1(1):105–107, 1953.

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

[65] C. E. Shannon. Coding theorems for a discrete source with a fidelity criterion. In *Claude E. Shannon: Collected Papers*, pages 325–350. Wiley-IEEE Press, 1993.

[66] Y. Shao, Q. Cao, and D. Gündüz. A theory of semantic communication. *IEEE Transactions on Mobile Computing*, 23:12211–12228, 2022.

[67] J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, 1st edition, 2001.

[68] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, 1973.

[69] P. A. Stavrou and M. Kountouris. The role of fidelity in goal-oriented semantic communication: A rate distortion approach. *IEEE Transactions on Communications*, 71(7):3918–3931, 2023.

[70] W. Szpankowski and A. Grama. Frontiers of science information: Shannon meets turing. *IEEE Computer*, (51):32–42, 2018.

[71] A. Tarski. The semantic conception of truth and the foundations of semantics. *Philosophy and Phenomenological Research*, 4(3):341–376, 1944.

[72] A. Tarski. The concept of truth in formalized languages. In *A. Tarski: Logic, Semantics, Metamathematics: papers from 1923-1938*, pages 152–278. Oxford: Clarendon Press, 1956.

[73] J. Väänänen. Second-order and Higher-order Logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021. `https://plato.stanford.edu/archives/fall2021/entries/logic-higher-order/`.

[74] W. Weaver. Recent contributions to the mathematical theory of communication. In *ETC A Review of General Semantics*, volume 10, pages 261–281, 1953.

[75] Z. Weng, Z. Qin, and G. Y. Li. Semantic communications for speech signals. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[76] D. Wheeler and B. Natarajan. Engineering semantic communication: A survey. *IEEE Access*, PP:1–1, 01 2023.

[77] Wikipedia contributors. Logic in computer science — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Logic_in_computer_science`, 2024.

[78] Wikipedia contributors. Mathematical logic — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Mathematical_logic`, 2024.

[79] Wikipedia contributors. Propositional calculus — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Propositional_calculus`, 2024.

[80] Wikipedia contributors. Semantic theory of truth — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Semantic_theory_of_truth`, 2024.

[81] A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, 1976.

[82] A. C.-C. Yao. Some complexity questions related to distributive computing(preliminary report). STOC '79, page 209–213, New York, NY, USA, 1979. Association for Computing Machinery.

[83] H. Yu, J. A. Evans, and L. R. Varshney. Information lattice learning. *J. Artif. Intell. Res.*, 77:971–1019, July 2023.

[84] H. Yu and L. R. Varshney. Semantic compression with information lattice learning. *2024 IEEE International Symposium on Information Theory Workshops (ISIT-W)*, pages 1–6, 2024.

A Proofs of miscellaneous results

Lemma 2 (Elementary properties of $\Lambda(a,b)$) *The function $\Lambda(a,b)$ is concave over the domain $[0,+\infty)\times[0,+\infty)$. If $\Delta_a,\Delta_b\geq 0$ with at least one of them being strictly positive, then $\Lambda(a+\Delta_a,b+\Delta_b)>\Lambda(a,b)$. If $a+b<1$, then for any mixture parameter $\lambda\in[0,1]$, $\Lambda(a,b)<H_{bin}(\lambda a+(1-\lambda)b)$. For any ξ, $\xi\Lambda(a,b)=\Lambda(\xi a,\xi b)$.*

Proof. By definition, it is clear that $\Lambda(a,b)\geq 0$ for $a,b\geq 0$. The gradient and Hessian of $\Lambda(a,b)$ are respectively given by

$$\nabla\Lambda(a,b)=\begin{bmatrix}\log_2\frac{a+b}{a}\\ \log_2\frac{a+b}{b}\end{bmatrix},\qquad \mathrm{Hess}(\Lambda(a,b))=\frac{1}{\log 2}\begin{bmatrix}-\frac{b}{a(1+b)} & \frac{1}{a+b}\\ \frac{1}{a+b} & -\frac{a}{b(a+b)}\end{bmatrix}.$$

Over $a,b\in(0,\infty)$, the gradient is strictly positive. As a consequence, the function is monotonically increasing when one of the arguments is fixed, and thus $\Lambda(a,b)\leq\Lambda(a,b+\Delta_b)\leq\Lambda(a+\Delta_a,b+\Delta_b)$ with at least one of those inequalities being strict due to the assumption. Moreover, the Hessian is positive semi-definite since its eigenvalues are 0 and $-\frac{a^2+b^2}{ab(a+b)}$, the latter always being strictly negative in the same domain. From this we conclude that the function is concave $\cap$. Because $\Lambda(a,0)=\Lambda(0,b)=0$, the Lemma assertions can be extended to the full domain $[0,+\infty)\times[0,+\infty)$. To prove the last statement, first note that $\Lambda(a,b)=\Lambda(b,a)$. For a given $\lambda\in[0,1]$, let $a_\lambda=\lambda a+(1-\lambda)b$ and $b_\lambda=\lambda b+(1-\lambda)a$. Due to the concavity $\cap$ of the Λ function, we deduce that

$$\Lambda(a,b)=\lambda\Lambda(a,b)+(1-\lambda)\Lambda(b,a)\leq\Lambda(a_\lambda,b_\lambda).$$

Next note that, since $a+b<1$, then $a_\lambda+b_\lambda<1$, and from the monotonicity property proved earlier we have

$$\Lambda(a_\lambda,b_\lambda)<\Lambda(a_\lambda,1-a_\lambda)=H_{\mathrm{bin}}(a_\lambda).$$

The final statement in the Lemma is an elementary consequence of the definition of Λ. □

B Formalization of the nonlinear code of subsection 4.2 within Propositional Logic

To make the nonlinear code of subsection 4.2 more memorable, we crafted a problem statement involving a space mission where the spacecraft has six possible actions, many of which can be applied potentially simultaneously (but not sequentially). One action saves the mission, another results in a catastrophe, and the other four have no effect. Houston knows which action saves the mission and which action results in a catastrophe. What is the minimum number of bits that Houston needs to send to the spacecraft? The answer is 2 bits and the associated algorithm is implied by small code.

It turns out that we can encode the 6-action scenario entirely with three Boolean variables, call the variables $\mathtt{b}_1, \mathtt{b}_2$ and $\mathtt{b}_3$. There are several ways to do this, but rather than encoding which action is the remedial action, let us instead encode which action is the *catastrophic* action. We choose the following encoding:

$$\begin{aligned}
A_1 \text{ is the catastrophic action } (000) &: \chi_1 = \neg\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \neg\mathtt{b}_3 \\
A_2 \text{ is the catastrophic action } (001) &: \chi_2 = \neg\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \mathtt{b}_3 \\
A_3 \text{ is the catastrophic action } (010) &: \chi_3 = \neg\mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \neg\mathtt{b}_3 \\
A_4 \text{ is the catastrophic action } (011) &: \chi_4 = \neg\mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \mathtt{b}_3 \\
A_5 \text{ is the catastrophic action } (100) &: \chi_5 = \mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \neg\mathtt{b}_3 \\
A_6 \text{ is the catastrophic action } (101) &: \chi_6 = \mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \mathtt{b}_3
\end{aligned}$$

The symbols $\chi_1, ..., \chi_6$ are shorthand names for the respective propositional sentences, and not additional variables. The three digit binary numbers in parentheses are mnemonics for how we are doing the encoding with the three Boolean variables.

With this encoding in hand, the objective of the Sender can be thought of as giving the Receiver just enough information so that for the actual remedial action, A_k, the Receiver can prove that A_k is *not* the catastrophic action. As long as the Sender can convey this information then they can be assured that the Receiver will eventually take the needed remedial action per the agreed upon protocol.

Consider the 4×6 partition matrix from subsection 4.2, which we have copied in below,

$$\begin{array}{cccccc}
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1
\end{array}$$

where we think of the ith action as being associated with the ith column. In any given scenario faced by the astronauts, exactly one of the rows will apply. If, say, the 2nd row applies and there is a 1 in cell $(2, i)$ the meaning is that action A_i either is the remedial action or harmless, and so *not* the catastrophic action. On the other hand, if cell $(2, i)$ contains a 0, the meaning is that action A_i is either the catastrophic action or harmless, and so not the remedial action. The important feature of this partition matrix is that for any distinct pair of indices i, j, where we can think of A_i as being the remedial action and A_j the catastrophic action, there is a row of this matrix such that the number in the ith entry is 1 and the number in the jth entry is 0. Hence the name "partition matrix".

Row 1 (the row consisting of 0 0 0 1 1 1) in the partition matrix is then given by the Boolean expression, which in English reads "One of the actions {1, 2, 3} is the catastrophic action and each the actions {4, 5, 6} is not the catastrophic action":

$$\begin{aligned}
\sigma_1 \quad = \quad & ((\neg\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \neg\mathtt{b}_3) \vee (\neg\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \mathtt{b}_3) \vee (\mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \neg\mathtt{b}_3)) \quad \wedge \\
& \neg(\neg\mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \mathtt{b}_3) \wedge \neg(\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \neg\mathtt{b}_3) \wedge \neg(\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \mathtt{b}_3).
\end{aligned}$$

The other rows of the matrix are then analogously given by the Boolean expressions:

$$\sigma_2 \quad = \quad ((\neg\mathtt{b}_1 \wedge \neg\mathtt{b}_2 \wedge \neg\mathtt{b}_3) \vee (\neg\mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \mathtt{b}_3) \vee (\wedge\neg\mathtt{b}_2 \wedge \mathtt{b}_3)) \quad \wedge$$

$$
\begin{aligned}
& \neg(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \mathtt{b}_3) \wedge \neg(\neg \mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \neg \mathtt{b}_3) \wedge \neg(\mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \neg \mathtt{b}_3), \\
\sigma_3 \quad = \quad & ((\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \mathtt{b}_3) \vee (\mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \neg \mathtt{b}_3) \vee (\mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \mathtt{b}_3)) \quad \wedge \\
& \neg(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \neg \mathtt{b}_3) \wedge \neg(\neg \mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \neg \mathtt{b}_3) \wedge \neg(\neg \mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \mathtt{b}_3), \\
\sigma_4 \quad = \quad & ((\neg \mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \neg \mathtt{b}_3) \vee (\neg \mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \mathtt{b}_3) \vee (\mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \neg \mathtt{b})) \quad \wedge \\
& \neg(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \neg \mathtt{b}_3) \wedge \neg(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \mathtt{b}_3) \wedge \neg(\mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \mathtt{b}_3).
\end{aligned}
$$

The sender and receiver can then agree to transmit 2 bits according to which expression σ_i applies, for example $00 \mapsto \sigma_1, 01 \mapsto \sigma_2, 10 \mapsto \sigma_3, 11 \mapsto \sigma_4$.

C Algorithms Used for the Experiments

C.1 Generation of Kernels

Pseudocode for the heuristic to generate kernels consistent with the receiver's knowledge, the query, and the sender's knowledge is given in Algorithm 1. The values p_r, p_q, p_s refer respectively to the probability that

Algorithm 1 GENERATE_KERNELS(NUM_VARS, p_r, p_q, p_s)

```
NUM_VARS                                  ▷ Given as input. Set to 10 for all of our test cases.
p_r, p_q, p_s                             ▷ Given as input.

sender_kernel = [2^NUM_VARS]
query_kernel = [2^NUM_VARS]
receiver_kernel = [2^NUM_VARS]
for i = 1, ..., 2^NUM_VARS do
    r = random()
    if r ≤ p_r then
        receiver_kernel[i] = True
        if r ≤ p_q then
            query_kernel[i] = True
            if r ≤ p_s then
                sender_kernel[i] = True
            end if
        end if
    end if
end for

return sender_kernel, query_kernel, receiver_kernel
```

a randomly generated truth value assignment is consistent with the receiver's knowledge, the query, and the sender's knowledge.

C.2 Generation of Generalized Decision Tree Sentences

Pseudocode for implementing the generalized decision tree algorithm, given a kernel, is given in Algorithm 2. We say that the resulting sentence is a Generalized Decision Tree, or GDT for short. The algorithm runs by recursively calling the method `GENERATE_GDT_FOR_KERNEL()` with the kernel as an argument. The output is a Boolean expression in Generalized Decision Tree form. We adopt the convention that the empty Boolean expression, `""`, is true for all variable assignments. The first step in the recursive routine is to check if there any true values (equivalently, any `1`s) in the kernel. Since we maintain the kernel as an indicator array this is not quite the same thing as checking that the kernel is empty, though that is what is being done conceptually. If there are none then we have a contradiction so the simplest contradictory sentence is output. This condition only happens, as we shall see, at the highest level of the recursive stack. The next step is to

extract constants, in other words to extract any variables that appear only positively or only negatively in every satisfying truth value assignment in the kernel. Say two such variables are found and they are $\texttt{X}_\texttt{i}$ and $\texttt{X}_\texttt{j}$ with $\texttt{X}_\texttt{i}$ appearing only positively and $\texttt{X}_\texttt{j}$ appearing only negatively. Then the kernel is reduced to remove the variables $\texttt{X}_\texttt{i}$ and $\texttt{X}_\texttt{j}$ and the output string (`gdt` in the pseudocode) is initialized to $\texttt{X}_\texttt{i} \wedge -\texttt{X}_\texttt{j}$. Next, a check it made to see if either the number of remaining variables is zero (which would also mean that there are no true values in the kernel) or the number of true values is equal to 2 to the power of the number of remaining variables. If this is the case, the function exits, just outputting the string of conjuncted constants, if any. (Note that this immediate return prevents the routine from ever being invoked with no zeroes except on initial invocation.) Otherwise, the function finds a variable, let us call it $\texttt{X}_\texttt{b}$, that is most balanced in terms of its positive and negative occurrences among the true values associated with the kernel. With this variable identified, two further reduced kernels are computed, one consisting of the true values associated with all variables minus $\texttt{X}_\texttt{b}$, when $\texttt{X}_\texttt{b}$ is true, and one one consisting of the true values associated with all variables minus $\texttt{X}_\texttt{b}$, when $\texttt{X}_\texttt{b}$ is false. In the pseudocode, these reduced kernels are identified as `pos_kernel` and `neg_kernel` respectively. Finally, the psdeudocode distinguishes two case depending on whether `tnf` is non-empty (the truth set associated with the original kernel contained constants) or otherwise. If `tnf` is not empty then it is a conjunction of (one or more) variables or their negations. We take the conjunction of this with $((\texttt{X}_\texttt{b} \wedge \texttt{GENERATE\_GDT\_FOR\_KERNEL}(\texttt{pos\_kernel})) \wedge (-\texttt{X}_\texttt{b} \wedge \texttt{GENERATE\_GDT\_FOR\_KERNEL}(\texttt{neg\_kernel})))$. If `tnf` is empty than we do not need to conjunct in anything and we also don't need the outer parentheses.

We illustrate the recursive calling of `GENERATE_GDT_FOR_KERNEL()` through a small example. Suppose we have 5 Boolean variables that we call $\texttt{X}_1, ..., \texttt{X}_5$, and the truth set associated with the kernel consists of the values $\{00100, 00101, 10101, 10111, 11001\}$. Thus, in the first of the true values (equivalently, satisfying truth value assignments), 00100, $\texttt{X}_1 = \texttt{X}_2 = \texttt{X}_4 = \texttt{X}_5 = \texttt{False}$ and $\texttt{X}_3 = \texttt{True}$. In the outer call to `GENERATE_GDT_FOR_KERNEL()`, there are no constants, the number of true values is not 2^5 so we proceed to finding a most balanced variable, in this case there is exactly one most balanced variable and it is $\texttt{X}_1$. We then prepare the reduced positive kernel, which is `pos_kernel` $=\{0101, 0111, 1001\}$, and the reduced negative kernel which is `neg_kernel` $=\{0100, 0101\}$.

We thus set

$$\begin{aligned} \texttt{gdt} \quad = \quad & (\texttt{X}_1 \wedge \texttt{GENERATE\_GDT\_FOR\_KERNEL}(\texttt{pos\_kernel})) \ \wedge \\ & (-\texttt{X}_1 \wedge \texttt{GENERATE\_GDT\_FOR\_KERNEL}(\texttt{neg\_kernel})). \end{aligned} \tag{92}$$

Now, in the call to `GENERATE_GDT_FOR_KERNEL`(`pos_kernel`) we see that for the reduced kernel `pos_kernel` $= \{0101, 0111, 1001\}$, the last variable, which is named $\texttt{X}_5$, appears only positively and is therefore constant. We can therefore pull out this variable and reduce the kernel further to $\{010, 011, 100\}$. The number of true values that remain is not 2^3 so we proceed to finding a most balanced variable, one such is the first of the remaining variables, which is named $\texttt{X}_2$. We omit the remaining details, but the result is the following expression:

$$\texttt{gdt} = (\texttt{X}_1 \wedge \texttt{X}_5 \wedge ((\texttt{X}_2 \wedge -\texttt{X}_3 \wedge -\texttt{X}_4) \vee (-\texttt{X}_2 \wedge \texttt{X}_3))) \vee (-\texttt{X}_1 \wedge -\texttt{X}_2 \wedge \texttt{X}_3 \wedge -\texttt{X}_4).$$

It is worth noting that the subexpression for `GENERATE_GDT_FOR_KERNEL`(`neg_kernel`) in (92) is especially simple since for the truth set, `neg_kernel` $=\{0100, 0101\}$, the first three variables, which have original names $\texttt{X}_2, \texttt{X}_3$ and $\texttt{X}_4$, are all constant and can be pulled out. We are then left with the (much) reduced kernel $\{0, 1\}$, and here, since $\texttt{NUM\_VARS} = 1$ we do have that `num_zeroes_to_match` $== 2^{\texttt{NUM\_VARS}}$ and so can omit the last variable ($\texttt{X}_5$).

Algorithm 2 GENERATE_GDT_FOR_KERNEL(kernel)

```
kernel[2^NUM_VAR]                                          ▷ Given as input.
NUM_VARS          ▷ Inferred from the size of the input array. Set to 10 for all test cases.
num_trues = COUNT_TRUES(zeroes_to_match)                   ▷ The number of 1s in the input array.

if num_trues == 0 then                                     ▷ Can only happen at the highest level;
    return "X1 ∧ −X1"                                      ▷ Return simple contradiction
end if

constants = EXTRACT_CONSTANTS(kernel)
num_trues = REDUCE_KERNEL_MOD_CONSTANTS(kernel, constants)
NUM_VARS = NUM_VARS − len(constants)
gdt = GET_CONJUNCTION_FOR_CONSTANTS(constants)
if NUM_VARS == 0  or  num_trues == 2^NUM_VARS then
    return gdt
end if

var = FIND_MOST_BALANCED_VAR(kernel)
pos_kernel = REDUCE_KERNEL_MOD_POS_VARIABLE(kernel, var)
neg_kernel = REDUCE_KERNEL_MOD_NEG_VARIABLE(kernel, var)

if gdt ≠ "" then
    gdt = gdt + " ∧ ((" + var + " ∧ " + GENERATE_GDT_FOR_KERNEL(pos_kernel) + ")  ∨ "
                + "(−" + var + " ∧ " + GENERATE_GDT_FOR_KERNEL(neg_kernel) + "))"
else
    gdt = "(" + var + " ∧ " + GENERATE_GDT_FOR_KERNEL(pos_kernel) + ")  ∨ "
                + "(−" + var + " ∧ " + GENERATE_GDT_FOR_KERNEL(neg_kernel) + ")"
end if

return gdt
```