Probabilistic Algorithmic Knowledge[*]

Joseph Y. Halpern
Cornell University
Ithaca, NY 14853 USA
halpern@cs.cornell.edu

Riccardo Pucella
Cornell University
Ithaca, NY 14853 USA
riccardo@cs.cornell.edu

Abstract

The framework of algorithmic knowledge assumes that agents use deterministic knowledge algorithms to compute the facts they explicitly know. We extend the framework to allow for randomized knowledge algorithms. We then characterize the information provided by a randomized knowledge algorithm when its answers have some probability of being incorrect. We formalize this information in terms of *evidence*; a randomized knowledge algorithm returning "Yes" to a query about a fact φ provides evidence for φ being true. Finally, we discuss the extent to which this evidence can be used as a basis for decisions.

1 Introduction

Under the standard possible-worlds interpretation of knowledge, which goes back to Hintikka [1962], an agent knows φ if φ is true at all the worlds the agent considers possible. This interpretation of knowledge has been found useful in capturing some important intuitions in the analysis of distributed protocols [Fagin, Halpern, Moses, and Vardi 1995]. However, its usefulness is somewhat limited by what Hintikka [1962] called the *logical omniscience problem*: agents know all tautologies and know all logical consequences of their knowledge. Many approaches have been developed to deal with the logical omniscience problem (see [Fagin, Halpern, Moses, and Vardi 1995, Chapter 10 and 11] for a discussion and survey). We focus on one approach here that has been called *algorithmic knowledge* [Halpern, Moses, and Vardi 1994]. The idea is simply to assume that agents are equipped with "knowledge algorithms" that they use to compute what they know. An agent *algorithmically knows* φ if its knowledge algorithm says "Yes" when asked φ.[1]

Algorithmic knowledge is a very general approach. For example, Berman, Garay, and Perry [1989] implicitly use a particular form of algorithmic knowledge in their analysis of Byzantine agreement. Roughly speaking they allow agents to perform limited tests based on the information they have; agents know only what follows from these limited tests. Ramanujam [1999] investigates a particular form of algorithmic knowledge, where the knowledge algorithm is essentially a model-checking procedure for a standard logic of knowledge. More specifically, Ramanujam considers, at every state, the part of the model that a particular agent sees (for instance, an agent in a distributed

[*]A preliminary version of this paper appeared in the *Proceedings of the Ninth Conference on Theoretical Aspects of Rationality and Knowledge*, pp. 118–130, 2003.

[1]We remark that what we are calling "knowledge algorithms" here are called "local algorithms" in [Fagin, Halpern, Moses, and Vardi 1995; Halpern, Moses, and Vardi 1994].

system may be aware only of its immediate neighbors, the ones with whom he can communicate) and takes as knowledge algorithm the model-checking procedure for epistemic logic, applied to the submodel generated by the visible states. Halpern and Pucella [2002] have applied algorithmic knowledge to security to capture intruders who are resource bounded (and thus, for example, cannot factor the products of large primes that arise in the RSA cryptosystem [Rivest, Shamir, and Adelman 1978]).

All these examples use *sound* knowledge algorithms: although the algorithm may not give an answer under all circumstances, when it says "Yes" on input φ, the agent really does know φ in the standard possible-worlds sense. Although soundness is not required in the basic definition, it does seem to be useful in many applications.

Our interest in this paper is knowledge algorithms that may use some randomization. As we shall see, there are numerous examples of natural randomized knowledge algorithms. With randomization, whether or not the knowledge algorithm says "Yes" may depend on the outcome of coin tosses. This poses a slight difficulty in even giving semantics to algorithmic knowledge, since the standard semantics makes sense only for deterministic algorithms. To deal with this problem, we make the algorithms deterministic by supplying them an extra argument (intuitively, the outcome of a sequence of coin tosses) to "derandomize" them. We show that this approach provides a natural extension of the deterministic case.

To motivate the use of randomized knowledge algorithms, we consider a security example from Halpern and Pucella [2002]. The framework in that paper lets us reason about principals communicating in the presence of adversaries, using cryptographic protocols. The typical assumption made when analyzing security protocols is that adversaries can intercept all the messages exchanged by the principals, but cannot necessarily decrypt encrypted messages unless they have the appropriate decryption key. To capture precisely the capabilities of adversaries, we use knowledge algorithms. Roughly speaking, a knowledge algorithm for an adversary will specify what information the adversary can extract from intercepted messages. In this paper, we consider an adversary that further attempts to guess the cryptographic keys used by the principals in the protocol. We show how to capture the knowledge of such an adversary using a randomized knowledge algorithm.

Having defined the framework, we try to characterize the information obtained by getting a "Yes" answer to a query for φ. If the knowledge algorithm is sound, then a "Yes" answer guarantees that φ is true. However, the randomized algorithms of most interest to us give wrong answers with positive probability, so are not sound. Nevertheless, it certainly seems that if the probability that the algorithm gives the wrong answer is low, it provides very useful information when it says "Yes" to a query φ. This intuition appears in the randomized algorithms literature, where a "Yes" answer from a highly reliable randomized algorithm that is, one with a low probability of being wrong, is deemed "good enough". In what sense is this true? One contribution of our work is to provide a formal answer to that question. It may seem that a "Yes" answer to a query φ from a highly reliable randomized knowledge algorithm should make the probability that φ is true be high but, as we show, this is not necessarily true. Rather, the information should be viewed as *evidence* that φ is true; the probability that φ is true also depends in part on the prior probability of φ.

Evidence has been widely studied in the literature on inductive logic [Kyburg 1983]. Evidence for a particular hypothesis can be accumulated from different sources, and it is possible to combine the evidence provided by knowledge algorithms with other kind of evidence already accumulated for a formula, obtained for example by running other kinds of tests, and look at the combined evidence. We do not consider evidence in such a general setting in this paper; rather, we focus on the evidence contributed specifically by a randomized knowledge algorithm. Our companion paper [Halpern and Pucella 2003] provides a treatment of evidence in a more general setting.

2 Reasoning about Knowledge and Algorithmic Knowledge

The aim is to be able to reason about properties of systems involving the knowledge of agents in the system. To formalize this type of reasoning, we first need a language. The syntax for a multiagent logic of knowledge is straightforward. Starting with a set Φ of primitive propositions, which we can think of as describing basic facts about the system, such as "the door is closed" or "agent A sent the message m to B", more complicated formulas are formed by closing off under negation, conjunction, and the modal operators K_1, ..., K_n and $X_1, \ldots, X_n$. Thus, if φ and ψ are formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, $K_i\varphi$ (read "agent i knows φ"), and $X_i\varphi$ (read "agent i can compute φ").

The standard possible-worlds semantics for knowledge uses *Kripke structures* [Kripke 1963]. Formally, a Kripke structure is composed of a set S of states or possible worlds, an interpretation π which associates with each state in S a truth assignment to the primitive propositions (i.e., $\pi(s)(p) \in \{\textbf{true}, \textbf{false}\}$ for each state $s \in S$ and each primitive proposition p), and equivalence relations $\sim_i$ on S (recall that an equivalence relation is a binary relation which is reflexive, symmetric, and transitive). The relation $\sim_i$ is agent i's possibility relation. Intuitively, $s \sim_i t$ if agent i cannot distinguish state s from state t (so that if s is the actual state of the world, agent i would consider t a possible state of the world). For our purposes, the equivalence relations are obtained by taking a set $\mathcal{L}$ of *local states*, and giving each agent a *view* of the state, that is, a function $L_i : S \rightarrow \mathcal{L}$. We define $s \sim_i t$ if and only if $L_i(s) = L_i(t)$. In other words, agent i considers the states s and t indistinguishable if he has the same local state at both states.

To interpret explicit knowledge of the form $X_i\varphi$, we assign to each agent a *knowledge algorithm* that the agent can use to determine whether he knows a particular formula. A knowledge algorithm $\mathtt{A}$ takes as inputs a formula of the logic, a local state ℓ in $\mathcal{L}$, as well as the state as a whole.[2] It is required to be deterministic and terminate on all inputs, with result "Yes", "No", or "?". An *algorithmic knowledge structure* M is a tuple $(S, \pi, L_1, \ldots, L_n, \mathtt{A}_1, \ldots, \mathtt{A}_n)$, where $L_1, \ldots, L_n$ are the view functions on the states, and $\mathtt{A}_1, \ldots, \mathtt{A}_n$ are knowledge algorithms.[3]

We define what it means for a formula φ to be true (or satisfied) at a state s in an algorithmic knowledge structure M, written $(M, s) \models \varphi$, inductively as follows:

$(M, s) \models p$ if $\pi(s)(p) = \textbf{true}$

$(M, s) \models \neg\varphi$ if $(M, s) \not\models \varphi$

$(M, s) \models \varphi \wedge \psi$ if $(M, s) \models \varphi$ and $(M, s) \models \psi$

[2]The original presentation of algorithmic knowledge by Halpern *et al.* [1994] did not have knowledge algorithms taking the state as input. Clearly, our knowledge algorithms are a generalization, since they can simply ignore the state argument. We provide justification for the added generality in Section 5.

[3]Halpern *et al.* [1994] introduced algorithmic knowledge in the context of dynamic systems, that is, systems evolving in time. The knowledge algorithm is allowed to change at every state of the system. Since the issues that interest us do not involve time, we do not consider dynamic systems in this paper. We remark that what we are calling "algorithmic knowledge structures" here are called "algorithmic structures" in [Fagin, Halpern, Moses, and Vardi 1995; Halpern, Moses, and Vardi 1994]. The term "algorithmic knowledge structures" is used in the paperback edition of [Fagin, Halpern, Moses, and Vardi 1995].

$(M, s) \models K_i\varphi$ if $(M, t) \models \varphi$ for all t with $s \sim_i t$

$(M, s) \models X_i\varphi$ if $\mathtt{A}_i(\varphi, L_i(s), s) =$ "Yes".

The first clause shows how we use the π to define the semantics of the primitive propositions. The next two clauses, which define the semantics of $\neg$ and $\wedge$, are the standard clauses from propositional logic. The fourth clause is designed to capture the intuition that agent i knows φ exactly if φ is true in all the states that i considers possible. The final clause interprets $X_i\varphi$ via agent i's knowledge algorithm. Thus, agent i has algorithmic knowledge of φ at a given state if the agent's algorithm outputs "Yes" when presented with φ, the agent's local state, and the state. (Both the outputs "No" and "?" result in lack of algorithmic knowledge.) As usual, we say that a formula φ is *valid in structure* M and write $M \models \varphi$ if $(M, s) \models \varphi$ for all states $s \in S$; φ is *valid* if it is valid in all structures.

We can think of K_i as representing *implicit knowledge*, facts that the agent implicitly knows, given its information. One can check that implicit knowledge is closed under implication, that is, $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$ is valid, and that an agent implicitly knows all valid formulas, so that if φ is valid, then $K_i\varphi$ is valid. These properties say that agents are very powerful reasoners. What is worse, while it is possible to change some properties of knowledge by changing the properties of the relation $\sim_i$, no matter how we change it, we still get closure under implication and knowledge of valid formulas as properties. They seem to be inescapable features of the possible-worlds approach. This suggests that the possible-worlds approach is appropriate only for "ideal knowers", ones that know all valid formulas as well as all logical consequences of their knowledge, and thus inappropriate for reasoning about agents that are computationally limited. In contrast, X_i represents *explicit knowledge*, facts whose truth the agent can compute explicitly. Since we put no a priori restrictions on the knowledge algorithms, an agent can explicitly know both φ and $\varphi \Rightarrow \psi$ without explicitly knowing ψ, for example.

As defined, there is no necessary connection between $X_i\varphi$ and $K_i\varphi$. An algorithm could very well claim that agent i knows φ (i.e., output "Yes") whenever it chooses to, including at states where $K_i\varphi$ does not hold. Although algorithms that make mistakes are common, we are often interested in knowledge algorithms that are correct. We say that a knowledge algorithm is *sound* for agent i in the structure M if for all states s of M and formulas φ, $\mathtt{A}_i(\varphi, L_i(s), s) =$ "Yes" implies $(M, s) \models K_i\varphi$, and $\mathtt{A}_i(\varphi, L_i(s), s) =$ "No" implies $(M, s) \models \neg K_i\varphi$. Thus, a knowledge algorithm is sound if its definite answers are correct. If we restrict attention to sound algorithms, then algorithmic knowledge can be viewed as an instance of awareness, as defined by Fagin and Halpern [1988].

There is a subtlety here, due to the asymmetry in the handling of the answers returned by knowledge algorithms. The logic does not let us distinguish between a knowledge algorithm returning "No" and a knowledge algorithm returning "?"; they both result in lack of algorithmic knowledge.[4] In later sections, where we define the notion of a reliable knowledge algorithm, reliability will be characterized in terms of algorithmic knowledge, and thus the definition will not distinguish between a knowledge algorithm returning "No" or "?". Thus, for simplicity, we consider algorithms that are *complete*, in the sense that they always return either "Yes" or "No", and not "?". More precisely, for a formula φ, define a knowledge algorithm $\mathtt{A}_i$ to be *φ-complete* for agent i in the structure

[4] There may be reasons to distinguish "No" from "?", and it is certainly possible to extend the logic to distinguish them.

M if for all states s of M, $\mathtt{A}_i(\varphi, L_i(s), s) \in \{$"Yes", "No"$\}$. We consider such complete algorithms in Section 5.

3 Randomized Knowledge Algorithms

Randomized knowledge algorithms arise frequently in the literature (although they have typically not been viewed as knowledge algorithms). In order to deal with randomized algorithms in our framework, we need to address a technical question. Randomized algorithms are possibly nondeterministic; they may not yield the same result on every invocation with the same arguments. Since $X_i\varphi$ holds at a state s if the knowledge algorithm answers "Yes" at that state, this means that, with the semantics of the previous section, $X_i\varphi$ would not be well defined. Whether it holds at a given state depends on the outcome of random choices made by the algorithm. However, we expect the semantics to unambiguously declare a formula either true or false.

Before we describe our solution to the problem, we discuss another potential solution, which is to define the satisfaction relation probabilistically. That is, rather than associating a truth value with each formula at each state, we associate a probability $\Pr_s(\varphi)$ with each formula φ at each state s. The standard semantics can be viewed as a special case of this semantics, where the probabilities are always either 0 or 1. Under this approach, it seems reasonable to take $\Pr_s(p)$ to be either 0 or 1, depending on whether primitive proposition p is true at state s, and to take $\Pr_s(X_i\varphi)$ to be the probability that i's knowledge algorithm returns "Yes" given inputs φ, $L_i(s)$, and s. However, it is not then clear how to define $\Pr_s(\varphi \wedge \psi)$. Taking it to be $\Pr_s(\varphi)\Pr_s(\psi)$ implicitly treats φ and ψ as independent, which is clearly inappropriate if ψ is $\neg\varphi$. Even ignoring this problem, it is not clear how to define $\Pr_s(X_i\varphi \wedge X_i\psi)$, since again there might be correlations between the output of the knowledge algorithm on input $(\varphi, L_i(s), s)$ and input $(\psi, L_i(s), s)$.

We do not use probabilistic truth values in this paper. Instead, we deal with the problem by adding information to the semantic model to resolve the uncertainty about the truth value of formulas of the form $X_i\varphi$. Observe that if the knowledge algorithm $\mathtt{A}$ is randomized, then the answer that $\mathtt{A}$ gives on input (φ, ℓ, s) will depend on the outcome of coin tosses (or whatever other randomizing device is used by $\mathtt{A}$). We thus turn the randomized algorithm into a deterministic algorithm by supplying it with an appropriate argument. For example, we supply an algorithm that makes random choices by tossing coins a sequence of outcomes of coin tosses. We can now interpret a knowledge algorithm answering "Yes" with probability α at a state by considering the probability of those sequences of coin tosses at the state that make the algorithm answer "Yes".

Formally, we start with (possibly randomized) knowledge algorithms $\mathtt{A}_1, \ldots, \mathtt{A}_n$. For simplicity, assume that the randomness in the knowledge algorithms comes from tossing coins. A *derandomizer* is a tuple $v = (v_1, \ldots, v_n)$ such that for every agent i, v_i is a sequence of outcomes of coin tosses (heads and tails). There is a separate sequence of coin tosses for each agent rather than just a single sequence of coin tosses, since we do not want to assume that all agents use the same coin. Let V be the set of all such derandomizers. To every randomized algorithm $\mathtt{A}$ we associate a derandomized algorithm $\mathtt{A}^d$ which takes as input not just the query φ, local state ℓ, and state s, but also the sequence v_i of i's coin tosses, taken from a derandomizer $(v_1, \ldots, v_n)$. A *probabilistic algorithmic knowledge structure* is a tuple $N = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$, where ν is a probability distribution on V and $\mathtt{A}_i^d$ is the derandomized version of $\mathtt{A}_i$. (Note that in a probabilistic algorithmic knowledge structure the knowledge algorithms are in fact deterministic.)

The only assumption we make about the distribution ν is that it does not assign zero probability to the nonempty sets of sequences of coin tosses that determine the result of the knowledge algorithm. More precisely, we assume that for all agents i, formulas φ, and states s, $\{v \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v_i) = \text{“Yes”}\} \neq \varnothing$ if and only if $\nu(\{v \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v_i) = \text{“Yes”}\}) > 0$, and similarly for “No” and “?” answers. Note that this property is met, for instance, if ν assigns nonzero probability to every sequence of coin tosses. We do not impose any other restrictions on ν. In particular, we do not require that the coin be fair or that the tosses be independent. Of course, we can capture correlation between the agents' coins by using an appropriate distribution ν.

The truth of a formula is now determined relative to a pair (s, v) consisting of a state s and a derandomizer v. We abuse notation and continue to call these pairs states. The semantics of formulas in a probabilistic algorithmic knowledge structure is a straightforward extension of their semantics in algorithmic knowledge structures. The semantics of primitive propositions is given by π; conjunctions and negations are interpreted as usual; for knowledge and algorithmic knowledge, we have

$$(N, s, v) \models K_i\varphi \text{ if } (N, t, v') \models \varphi \text{ for all } v' \in V \text{ and all } t \in S \text{ such that } s \sim_i t$$

$$(N, s, v) \models X_i\varphi \text{ if } \mathtt{A}_i^d(\varphi, L_i(s), s, v_i) = \text{“Yes”, where } v = (v_1, \ldots, v_n).$$

Here, $\mathtt{A}_i^d$ gets v_i as part of its input. $\mathtt{A}_i^d(\varphi, L_i(s), s, v_i)$ is interpreted as the output of $\mathtt{A}_i^d$ given that v_i describes the outcomes of the coin tosses. Having the sequence of coin tosses as part of the input allows us to talk about the probability that i's algorithm answers yes to the query φ at a state s. It is simply $\nu(\{v \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v_i) = \text{“Yes”}\})$. To capture this in the language, we extend the language to allow formulas of the form $\Pr(\varphi) \geq \alpha$, read “the probability of φ is at least α”. The semantics of such formulas is straightforward:

$$(N, s, v) \models \Pr(\varphi) \geq \alpha \text{ if } \nu(\{v' \mid (N, s, v') \models \varphi\}) \geq \alpha.$$

Note that the truth of $\Pr(\varphi) \geq \alpha$ at a state (s, v) is independent of v. Thus, we can abuse notation and write $(N, s) \models \Pr(\varphi) \geq \alpha$. In particular, $(N, s) \models \Pr(X_i\varphi) < \alpha$ (or, equivalently, $(N, s) \models \Pr(\neg X_i\varphi) \geq 1 - \alpha$) if the probability of the knowledge algorithm returning “Yes” on a query φ is less than α, given state s.

If all the knowledge algorithms used are deterministic, then this semantics agrees with the semantics given in Section 2. To make this precise, note that if $\mathtt{A}$ is deterministic, then $\mathtt{A}^d(\varphi, \ell, v_i) = \mathtt{A}^d(\varphi, \ell, v_i')$ for all $v, v' \in V$. In this case, we abuse notation and write $\mathtt{A}(\varphi, \ell)$.

Proposition 3.1: *Let $N = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ be a probabilistic algorithmic knowledge, with $\mathtt{A}_1, \ldots, \mathtt{A}_n$ deterministic. Let $M = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1, \ldots, \mathtt{A}_n)$. If there are no occurrences of $\Pr$ in φ then, for all $s \in S$ and all $v \in V$, $(N, s, v) \models \varphi$ if and only if $(M, s) \models \varphi$.*

Thus, derandomizers are not needed to interpret the X_i operators if the knowledge algorithms are all deterministic. Moreover, in general, derandomizers are necessary only to interpret the $\Pr$ and X_i operators.

Proposition 3.2: *Let $N = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ be a probabilistic algorithmic knowledge structure and let $M = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1', \ldots, \mathtt{A}_n')$ be an algorithmic knowledge structure, where $\mathtt{A}_1', \ldots, \mathtt{A}_n'$ are arbitrary deterministic knowledge algorithms. If there are no occurrences of X_i and $\Pr$ in φ then, for all $s \in S$ and all $v \in V$, $(N, s, v) \models \varphi$ if and only if $(M, s) \models \varphi$.*

Propositions 3.1 and 3.2 justify the decision to “factor out” the randomization of the knowledge algorithms into semantic objects that are distinct from the states; the semantics of formulas that do not depend on the randomized choices do not in fact depend on those additional semantic objects.

4 An Example from Security

As we mentioned in the introduction, an important area of application for algorithmic knowledge is the analysis of cryptographic protocols. In previous work [Halpern and Pucella 2002], we showed how algorithmic knowledge can be used to model the resource limitations of an adversary. We briefly review the framework of that paper here.

Participants in a security protocol are viewed as exchanging messages in the free algebra generated by a set $\mathcal{P}$ of plaintexts and a set $\mathcal{K}$ of keys, over abstract operations $\cdot$ (concatenation) and $\{\!| \ |\!\}$ (encryption). The set $\mathcal{M}$ of messages is the smallest set that contains $\mathcal{K}$ and $\mathcal{P}$ and is closed under encryption and concatenation, so that if m_1 and m_2 are in $\mathcal{M}$ and $\mathsf{k} \in \mathcal{K}$, then $\mathsf{m}_1 \cdot \mathsf{m}_2$ and $\{\!|\mathsf{m}_1|\!\}_{\mathsf{k}}$ are in $\mathcal{M}$. We identify elements of $\mathcal{M}$ under the equivalence $\{\!|\{\!|\mathsf{m}|\!\}_{\mathsf{k}}|\!\}_{\mathsf{k}^{-1}} = \mathsf{m}$. We make the assumption, standard in the security literature, that concatenation and encryption have enough redundancy to recognize that a term is in fact a concatenation $\mathsf{m}_1 \cdot \mathsf{m}_2$ or an encryption $\{\!|\mathsf{m}|\!\}_{\mathsf{k}}$.

In an *algorithmic knowledge security structure*, some of the agents are participants in the security protocol being modeled, while other agents are *adversaries* that do not participate in the protocol, but attempt to subvert it. The adversary is viewed as just another agent, whose local state contains all the messages it has intercepted, as well as the keys initially known to the adversary, such as the public keys of all the agents. We use $initkey(\ell)$ to denote the set of initial keys known by an agent with local state ℓ and write $\mathsf{recv}(\mathsf{m}) \in \ell$ if m is one of the messages received (or intercepted in the case of the adversary) by an agent with local state ℓ. We assume that the language includes a primitive proposition $\mathsf{has}_i(\mathsf{m})$ for every message m, essentially saying that message m is contained within a message that agent i has received. Define the containment relation $\sqsubseteq$ on $\mathcal{M}$ as the smallest relation satisfying the following constraints:

(1) $\mathsf{m} \sqsubseteq \mathsf{m}$;

(2) if $\mathsf{m} \sqsubseteq \mathsf{m}_1$, then $\mathsf{m} \sqsubseteq \mathsf{m}_1 \cdot \mathsf{m}_2$;

(3) if $\mathsf{m} \sqsubseteq \mathsf{m}_2$, then $\mathsf{m} \sqsubseteq \mathsf{m}_1 \cdot \mathsf{m}_2$;

(4) if $\mathsf{m} \sqsubseteq \mathsf{m}_1$, then $\mathsf{m} \sqsubseteq \{\!|\mathsf{m}_1|\!\}_{\mathsf{k}}$.

Formally, $\mathsf{has}_i(\mathsf{m})$ is true at a local state ℓ if $\mathsf{m} \sqsubseteq \mathsf{m}'$ for some message m' such that $\mathsf{recv}(\mathsf{m}') \in \ell$.

Clearly, the adversary may not explicitly know that he has a given message if that message is encrypted using a key that the adversary does not know. To capture these restrictions, Dolev and Yao [1983] gave a now-standard description of capabilities of adversaries. Succinctly, a Dolev-Yao adversary can compose messages, replay them, or decipher them if he knows the right keys, but cannot otherwise “crack” encrypted messages. The Dolev-Yao model can be formalized by a relation $H \vdash_{DY} \mathsf{m}$ between a set H of messages and a message m. (Our formalization is equivalent to many other formalizations of Dolev-Yao in the literature, and is similar in spirit to that of Paulson [1998].) Intuitively, $H \vdash_{DY} \mathsf{m}$ means that an adversary can “extract” message m from a set of received messages and keys H, using the allowable operations. The derivation is defined using the following inference rules:

$$\frac{\mathsf{m} \in H}{H \vdash_{DY} \mathsf{m}} \quad \frac{H \vdash_{DY} \{\!|\mathsf{m}|\!\}_{\mathsf{k}} \quad H \vdash_{DY} \mathsf{k}^{-1}}{H \vdash_{DY} \mathsf{m}} \quad \frac{H \vdash_{DY} \mathsf{m}_1 \cdot \mathsf{m}_2}{H \vdash_{DY} \mathsf{m}_1} \quad \frac{H \vdash_{DY} \mathsf{m}_1 \cdot \mathsf{m}_2}{H \vdash_{DY} \mathsf{m}_2,}$$

where k^{-1} represents the key used to decrypt messages encrypted with k.

We can encode these capabilities via a knowledge algorithm $\mathtt{A}^{\mathrm{DY}}$ for the adversary as agent i. Intuitively, the knowledge algorithm $\mathtt{A}^{\mathrm{DY}}$ simply implements a search for the derivation of a message m from the messages that the agent has received and the initial set of keys, using the rules given above. The most interesting case in the definition of $\mathtt{A}^{\mathrm{DY}}$ is when the formula is $\mathsf{has}_i(\mathsf{m})$. To compute $\mathtt{A}_i^{\mathrm{DY}}(\mathsf{has}_i(\mathsf{m}), \ell, s)$, the algorithm simply checks, for every message m' received by the adversary, whether m is a submessage of m', according to the keys that are known to the adversary (given by the function keysof). Checking whether m is a submessage of m' is performed by a function submsg, which can take apart messages created by concatenation, or decrypt messages as long as the adversary knows the decryption key. (The function submsg basically implements the inference rules for $\vdash_{DY}$.) $\mathtt{A}_i^{\mathrm{DY}}(\mathsf{has}_i(\mathsf{m}), \ell, s)$ is defined by the following algorithm:

```
if m ∈ initkeys(ℓ) then return "Yes"
K = keysof(ℓ)
for each recv(m′) ∈ ℓ do
    if submsg(m, m′, K) then
        return "Yes"
return "?".
```

Note that the algorithm does not use the input s. Further details can be found in [Halpern and Pucella 2002], where it is also shown that this algorithm does indeed capture the Dolev-Yao adversary in the following sense:

Proposition 4.1: [Halpern and Pucella 2002] *If* $M = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ *is an algorithmic knowledge security structure with an adversary as agent* i *and* $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY}}$*, then* $(M, s) \models X_i(\mathsf{has}_i(\mathsf{m}))$ *if and only if* $\{\mathsf{m} \mid \mathsf{recv}(\mathsf{m}) \in L_i(s)\} \cup \mathit{initkeys}(\ell) \vdash_{DY} \mathsf{m}$*. Moreover, if* $(M, s) \models X_i(\mathsf{has}_i(\mathsf{m}))$ *then* $(M, s) \models \mathsf{has}_i(\mathsf{m})$*.*

Proposition 4.1 shows that $\mathtt{A}_i^{\mathrm{DY}}$ is a sound knowledge algorithm and that it captures the Dolev-Yao adversary.

The Dolev-Yao algorithm is deterministic. It does not capture, for example, an adversary who guesses keys in an effort to crack an encryption. Assume that the key space consists of N keys, and let $\mathit{guesskeys}(n)$ return n of these, chosen uniformly at random. Let $\mathtt{A}_i^{\mathrm{DY}+\mathrm{rg}(n)}$ be the result of modifying $\mathtt{A}_i^{\mathrm{DY}}$ to take random guessing into account (the rg stands for *random guess*), so that

$\mathtt{A}_i^{\mathrm{DY+rg}(n)}(\mathsf{has}_i(\mathsf{m}), s, \ell)$ is defined by the following algorithm:

if $\mathsf{m} \in \mathit{initkeys}(\ell)$ then return "Yes"
$K = \mathit{keysof}(\ell) \cup \mathit{guesskeys}(n)$
for each $\mathsf{recv}(\mathsf{m}') \in \ell$ do
  if $\mathit{submsg}(\mathsf{m}, \mathsf{m}', K)$ then
    return "Yes"
return "?".

(As before, the algorithm does not use the input s.) Using $\mathtt{A}_i^{\mathrm{DY+rg}(n)}$, the adversary gets to work with whatever keys it already had available, all the keys it can obtain using the standard Dolev-Yao algorithm, and the additional n randomly chosen keys returned by $\mathit{guesskeys}(n)$.

Of course, if the total number of keys is large relative to n, making n random guesses should not help much. Our framework lets us make this precise.

Proposition 4.2: *Suppose that $N = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ is a probabilistic algorithmic knowledge security structure with an adversary as agent i and that $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY+rg}(n)}$. Let K be the number of distinct keys used in the messages in the adversary's local state ℓ (that is, the number of keys used in the messages that the adversary has intercepted at a state s with $L_i(s) = \ell$). Suppose that $K/|\mathcal{K}| < 1/2$ and that ν is the uniform distribution on sequences of coin tosses. If $(N, s, v) \models \neg K_i X_i(\mathsf{has}_i(\mathsf{m}))$, then $(N, s, v) \models \Pr(X_i(\mathsf{has}_i(\mathsf{m}))) < 1 - e^{-2nK/|\mathcal{K}|}$. Moreover, if $(N, s, v) \models X_i(\mathsf{has}_i(\mathsf{m}))$ then $(N, s, v) \models \mathsf{has}_i(\mathsf{m})$.*

Proposition 4.2 says that what we expect to be true is in fact true: random guessing of keys is sound, but it does not help much (at least, if the number of keys guessed is a small fraction of the total numbers of keys). If it is possible that the adversary does not have algorithmic knowledge of m, then the probability that it has algorithmic knowledge is low. While this result just formalizes our intuitions, it does show that the probabilistic algorithmic knowledge framework has the resources to formalize these intuitions naturally.

5 Probabilistic Algorithmic Knowledge

While the "guessing" extension of the Dolev-Yao algorithm considered in the previous section is sound, we are often interested in randomized knowledge algorithms that may sometimes make mistakes. We consider a number of examples in this section, to motivate our approach.

First, suppose that Bob knows (or believes) that a coin is either fair or double-headed, and wants to determine which. He cannot examine the coin, but he can "test" it by having it tossed and observing the outcome. Let $\mathtt{dhead}$ be a proposition that is true if and only if the coin is double-headed. Bob uses the following $\mathtt{dhead}$-complete randomized knowledge algorithm $\mathtt{A}_{\mathrm{Bob}}$: when queried about $\mathtt{dhead}$, the algorithm "tosses" the coin, returning "Yes" if the coin lands heads and "No" if the coin lands tails. It is not hard to check that if the coin is double-headed, then $\mathtt{A}_{\mathrm{Bob}}$ answers "Yes" with probability 1 (and hence "No" with probability 0); if the coin is fair, then $\mathtt{A}_{\mathrm{Bob}}$ answers "Yes" with probability 0.5 (and hence "No" with probability 0.5 as well). Thus, if the coin fair, there is a chance that $\mathtt{A}_{\mathrm{Bob}}$ will make a mistake, although we can make the probability of error arbitrarily small by applying the algorithm repeatedly (alternatively, by increasing the number of coin tosses performed by the algorithm).

Second, consider a robot navigating, using a probabilistic sensor. This sensor returns the distance to the wall in front of the robot, within some tolerance. For simplicity, suppose that if the wall is at distance m, then the sensor will return a reading of $m-1$ with probability $1/4$, a reading of m with probability $1/2$, and a reading of $m+1$ with probability $1/4$. Let $\mathsf{wall}(m)$ be a proposition true at a state if and only if the wall is at distance at most m in front of the robot. Suppose that the robot uses the following $\mathsf{wall}(m)$-complete (for all m) knowledge algorithm $\mathtt{A}_{\mathrm{Robot}}$ to answer queries. Given query $\mathsf{wall}(m)$, $\mathtt{A}_{\mathrm{Robot}}$ takes a sensor reading r; if $r \leq m$, the algorithm returns "Yes", otherwise, it reading r; if $r \leq m$, the algorithm returns "Yes", otherwise, it returns "No". It is not hard to check that if the wall is actually at distance less than or equal to m, then $\mathtt{A}_{\mathrm{Robot}}$ answers "Yes" to a query $\mathsf{wall}(m)$ with probability $\leq 3/4$ (and hence "No" with probability $\geq 1/4$). If the wall is actually at distance greater than m, then $\mathtt{A}_{\mathrm{Robot}}$ answers "Yes" with probability $\leq 1/4$ (and hence "No" with a probability $\geq 1/4$). This example also illustrates the usefulness of allowing knowledge algorithms to take the state as input. Intuitively, the knowledge algorithm taking a sensor reading corresponds, in the model, to the knowledge algorithm querying the state for the distance to the wall.

As a last example, Finally, suppose that Alice has in her local state a number $n > 2$. Let prime be a proposition true at state s if and only if the number n in Alice's local state is prime. Clearly, Alice either (implicitly) knows prime or knows $\neg\mathsf{prime}$. However, this is implicit knowledge. Suppose that Alice uses Rabin's [1980] primality-testing algorithm to test if n is prime. That algorithm uses a (polynomial-time computable) predicate $P(n,a)$ with the following properties, for a natural number n and $1 \leq a \leq n-1$:

(1) $P(n,a) \in \{0,1\}$;

(2) if n is composite, $P(n,a) = 1$ for at least $n/2$ choices of a;

(3) if n is prime, $P(n,a) = 0$ for all a.

Thus, Alice uses the following prime-complete randomized knowledge algorithm $\mathtt{A}_{\mathrm{Alice}}$: when queried about prime, the algorithm picks a number a at random between 0 and the number n in Alice's local state; if $P(n,a) = 1$, it says "No" and if $P(n,a) = 0$, it says "Yes". (It is irrelevant for our purposes what the algorithm does on other queries.) It is not hard to check that $\mathtt{A}_{\mathrm{Alice}}$ has the following properties. If the number n in Alice's local state is prime, then $\mathtt{A}_{\mathrm{Alice}}$ answers "Yes" to a query prime with probability 1 (and hence "No" to the same query with probability 0). If n is composite, $\mathtt{A}_{\mathrm{Alice}}$ answers "Yes" to a query prime with probability $\leq 1/2$ and "No" with probability $\geq 1/2$. Thus, if n is composite, there is a chance that $\mathtt{A}_{\mathrm{Alice}}$ will make a mistake, although we can make the probability of error arbitrarily small by applying the algorithm repeatedly. While this problem seems similar to the double-headed coin example above, note that we have only *bounds* on the probabilities here. The actual probabilities corresponding to a particular number n depend on various number theoretic properties of that number. We return to this issue in Section 5.2.

Randomized knowledge algorithms like those in the examples above are quite common in the literature. They are not sound, but are "almost sound". The question is what we can learn from such an "almost sound" algorithm. Returning to the first example, we know the probability that $\mathtt{A}_{\mathrm{Bob}}$ says "Yes" (to the query dhead) given that the coin is double-headed; what we are interested in is the probability that the coin is double-headed given that $\mathtt{A}_{\mathrm{Bob}}$ says "Yes". (Of course, the coin is either double-headed or not. However, if Bob has to make decisions based on whether the coin is double-headed, it seems reasonable for him to ascribe a subjective probability to the coin being double-headed. It is this subjective probability that we are referring to here.)

Taking "dh" to represent "the coin is double-headed", by Baye's rule,

$$\Pr(\text{dh} \mid \mathtt{A}_{\text{Bob}} \text{ says "Yes"}) = \frac{\Pr(\mathtt{A}_{\text{Bob}} \text{ says "Yes"} \mid \text{dh})\Pr(\text{dh})}{\Pr(\mathtt{A}_{\text{Bob}} \text{ says "Yes"})}.$$

The only piece of information in this equation that we have is $\Pr(\mathtt{A}_{\text{Bob}}$ says "Yes" $\mid$ dh$)$. If we had $\Pr(\text{dh})$, we could derive $\Pr(\mathtt{A}_{\text{Bob}}$ says "Yes"$)$. However, we do not have that information, since we did not assume a probability distribution on the choice of coin. Although we do not have the information needed to compute $\Pr(\text{dh} \mid \mathtt{A}_{\text{Bob}}$ says "Yes"$)$, there is still a strong intuition that if $X_i\mathsf{dhead}$ holds, this tells us something about whether the coin is double-headed. How can this be formalized?

5.1 Evidence

Intuitively, the fact that $X_i\varphi$ holds provides "evidence" that φ holds. But what is evidence? There are a number of definitions in the literature. They all essentially give a way to assign a "weight" to different hypotheses based on an observation; they differ in exactly how they assign the weight (see [Kyburg 1983] for a survey). Some of these approaches make sense only if there is a probability distribution on the hypotheses. Since this is typically not the case in the applications of interest to us (for example, in the primality example, we do not want to assume a probability on the input n), we use a definition of evidence given by Shafer [1982] and Walley [1987], which does not presume a probability on hypotheses.

We assume a set $\mathcal{H}$ of mutually exclusive and exhaustive hypotheses; thus, exactly one hypothesis holds at any given time. For the examples of this paper, the hypotheses of interest have the form $\mathcal{H} = \{h_0, \neg h_0\}$, where the hypothesis $\neg h_0$ is the negation of hypothesis h_0. Intuitively, this is because we want to reason about the evidence associated with a formula or its negation (see Section 5.3). For example, if h_0 is "the coin is double-headed", then $\neg h_0$ is "the coin is not double-headed" (and thus, if there are only two kinds of coins, double-headed and fair, then $\neg h_0$ is "the coin is fair"). We are given a set $\mathcal{O}$ of *observations*, which can be understood as outcomes of experiments that we can make. Assume that for each hypothesis $h \in \mathcal{H}$ there is a probability space $(\mathcal{O}, 2^{\mathcal{O}}, \mu_h)$. Intuitively, $\mu_h(ob)$ is the probability of ob given that hypothesis h holds. While this looks like a conditional probability, notice that it does not require a probability on $\mathcal{H}$. Define an *evidence space* to be a tuple $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mu_h \mid h \in \mathcal{H}\})$, where $\mathcal{H}$, $\mathcal{O}$, and μ_h (for $h \in \mathcal{H}$) are as above.

For an evidence space $\mathcal{E}$, the weight that the observation ob lends to hypothesis $h \in \mathcal{H}$, written $w_{\mathcal{E}}(ob, h)$, is

$$w_{\mathcal{E}}(ob, h) = \frac{\mu_h(ob)}{\sum_{h' \in \mathcal{H}} \mu_{h'}(ob).} \tag{1}$$

The weight of evidence $w_{\mathcal{E}}$ is not defined by Equation (1) for an observation ob such that $\sum_{h \in \mathcal{H}} \mu_h(ob) = 0$. Intuitively, this means that the observation ob is impossible. In the literature on confirmation theory it is typically assumed that this case never arises. More precisely, it is assumed that all observations are possible, so that for every observation ob, there is an hypothesis h such that $\mu_h(ob) > 0$. In our case, making this assumption is unnatural. We want to view the answers given by knowledge algorithms as observations, and it seems perfectly reasonable to have a knowledge algorithm that never returns "No", for instance. As we shall see below, the fact that the weight of evidence is undefined in the case that $\sum_{h\in\mathcal{H}} \mu_h(ob) > 0$ is not a problem in our intended application, thanks to our assumption that ν does not assign zero probability to the nonempty sets of sequences of coin tosses that determine the result of the knowledge algorithm.

We extend $w_{\mathcal{E}}$ to sets of hypotheses by taking $w_{\mathcal{E}}(ob, H) = \sum_{h\in\mathcal{H}} w_{\mathcal{E}}(ob, h)$. Intuitively, $w_{\mathcal{E}}(ob, H)$ is the weight of evidence that the observation ob lends to the fact that the hypothesis is one of those in H. Observe that the measure $w_{\mathcal{E}}$ always lies between 0 and 1, with 1 indicating that the full weight of the evidence of observation ob is provided to the hypotheses. While the weight of evidence $w_{\mathcal{E}}$ looks like a probability measure (and, in fact, for each fixed observation ob for which $\sum_{h\in\mathcal{H}} \mu_h(ob) > 0$, $w_{\mathcal{E}}(ob, \cdot)$ is a probability measure on $\mathcal{H}$), it should not be interpreted as a probability measure. It is simply a way to assign a weight to hypotheses given observations. It is possible to interpret the weight function w as a prescription for how to update a prior probability on the hypotheses into a posterior probability on those hypotheses, after having considered the observations made. We do not focus on these aspects here; see [Halpern and Pucella 2003] for more details.

For the double-headed coin example, the set $\mathcal{H}$ of hypotheses is $\{\mathsf{dhead}, \neg\mathsf{dhead}\}$. The observations $\mathcal{O}$ are simply the possible outputs of the knowledge algorithm $\mathtt{A}_{\mathrm{Bob}}$ on the formula dhead, namely, $\{\text{"Yes"}, \text{"No"}\}$. From the discussion following the description of the example, it follows that $\mu_{\mathsf{dhead}}(\text{"Yes"}) = 1$ and $\mu_{\mathsf{dhead}}(\text{"No"}) = 0$, since the algorithm always says "Yes" when the coin is double-headed. Similarly, $\mu_{\neg\mathsf{dhead}}(\text{"Yes"})$ is the probability that the algorithm says "Yes" if the coin is not double-headed. By assumption, the coin is fair if it is not double-headed, so $\mu_{\neg\mathsf{dhead}}(\text{"Yes"}) = 1/2$ and $\mu_{\neg\mathsf{dhead}}(\text{"No"}) = 1/2$. Let

$$\mathcal{E} = (\{\mathsf{dhead}, \neg\mathsf{dhead}\}, \{\text{"Yes"}, \text{"No"}\}, \{\mu_{\mathsf{dhead}}, \mu_{\neg\mathsf{dhead}}\}).$$

We can compute that $w_{\mathcal{E}}(\text{"Yes"}, \mathsf{dhead}) = 2/3$ and $w_{\mathcal{E}}(\text{"Yes"}, \neg\mathsf{dhead}) = 1/3$. Intuitively, a "Yes" answer to the query dhead provides more evidence for the hypothesis dhead than the hypothesis $\neg$dhead. Similarly, $w(\text{"No"}, \mathsf{dhead}) = 0$ and $w(\text{"No"}, \neg\mathsf{dhead}) = 1$. Thus, an output of "No" to the query dhead indicates that the hypothesis $\neg$dhead must hold.

This approach, however, is not quite sufficient to deal with the sensor example because, in that example, probability of an observation does not depend solely on whether the hypothesis is true or false. The probability of the algorithm answering "Yes" to a query $\mathsf{wall}(10)$ when $\mathsf{wall}(10)$ is true depends on the actual distance m to the wall:

- if $m \leq 9$, then $\mu_{\mathsf{wall}(10)}(\text{"Yes"}) = 1$ (and thus $\mu_{\mathsf{wall}(10)}(\text{"No"}) = 0$);
- if $m = 10$, then $\mu_{\mathsf{wall}(10)}(\text{"Yes"}) = 3/4$ (and thus $\mu_{\mathsf{wall}(10)}(\text{"No"}) = 1/4$).

Similarly, the probability of the algorithm answering "Yes" to a query $\mathsf{wall}(10)$ in a state where $\neg\mathsf{wall}(10)$ holds depends on m in the following way:

- if $m = 11$, then $\mu_{\neg\mathsf{wall}(10)}(\text{"Yes"}) = 1/4$;
- if $m \geq 12$, then $\mu_{\neg\mathsf{wall}(10)}(\text{"Yes"}) = 0$.

It does not seem possible to capture this information using the type of evidence space defined above. In particular, we do not have a single probability measure over the observations given a particular hypothesis. One reasonable way of capturing the information is to associate a *set* of probability measures on observations with each hypothesis; intuitively, these represent the possible probabilities on the observations, depending on the actual state.

To make this precise, define a *generalized evidence space* to be a tuple $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mathcal{P}_h \mid h \in \mathcal{H}\})$. What is the most appropriate way to define weight of evidence given sets of probability measures? As a first step, consider the set of all possible weights of evidence that are obtained by taking any combination of probability measures, one from each set $\mathcal{P}_h$. This gives us a range of possible weights of evidence. We can then define *upper* and *lower* weights of evidence, determined by the maximum and minimum values in the range, somewhat analogous to the notions of upper and lower probability [Halpern 2003]. (Given a set $\mathcal{P}$ of probability measures, the *lower probability* of a set U is $\inf_{\mu \in \mathcal{P}} \mu(U)$; its *upper probability* is $\sup_{\mu \in \mathcal{P}} \mu(U)$.) Let

$$\mathcal{W}_{\mathcal{E}}(ob, H) = \left\{ \frac{\sum_{h \in H} \mu_h(ob)}{\sum_{h \in \mathcal{H}} \mu_h(ob)} \;\middle|\; \mu_h \in \mathcal{P}_h \right\}.$$

$\mathcal{W}_{\mathcal{E}}(ob, H)$ is the set of possible weights of evidence for the set of hypotheses H given by ob. Note that we define $\mathcal{W}_{\mathcal{E}}$ directly in terms of sets of hypotheses, rather than for single hypotheses. Define the *lower weight of evidence function* $\underline{w}_{\mathcal{E}}$ by taking $\underline{w}_{\mathcal{E}}(ob, H) = \inf \mathcal{W}_{\mathcal{E}}(ob, H)$; similarly, define the *upper weight of evidence function* $\overline{w}_{\mathcal{E}}$ by taking $\overline{w}_{\mathcal{E}}(ob, H) = \sup \mathcal{W}_{\mathcal{E}}(ob, H)$. Perhaps not surprisingly, $\underline{w}_{\mathcal{E}}(ob, \cdot)$ behaves like a lower probability on $\mathcal{H}$ for a fixed observation ob, while $\overline{w}_{\mathcal{E}}(ob, \cdot)$ behaves like an upper probability. Because $\underline{w}_{\mathcal{E}}(ob, \cdot)$ and $\overline{w}_{\mathcal{E}}(ob, \cdot)$ behave like lower and upper probabilities, they inherit properties such as $\underline{w}_{\mathcal{E}}(ob, H) = 1 - \overline{w}_{\mathcal{E}}(ob, \overline{H})$, where $\overline{H} = \mathcal{H} - H$ is the complement of H in $\mathcal{H}$ [Halpern 2003]. Of course, if $\mathcal{P}_h = \{\mu_h\}$ for all hypotheses $h \in \mathcal{H}$, then $\underline{w}_{\mathcal{E}} = \overline{w}_{\mathcal{E}} = w_{\mathcal{E}}$.

Lower and upper evidence can be used to model the examples at the beginning of this section. In the sensor example, with $\mathcal{H} = \{\mathsf{wall}(10), \neg\mathsf{wall}(10)\}$, there are two probability measures associated with the hypothesis $\mathsf{wall}(10)$, namely, $\mu_{\mathsf{wall}(10),\leq 9}(\text{“Yes”}) = 1$ and $\mu_{\mathsf{wall}(10),=10}(\text{“Yes”}) = 3/4$; similarly, there are two probability measures associated with the hypothesis $\neg\mathsf{wall}(10)$, namely $\mu_{\neg\mathsf{wall}(10),=11}(\text{“Yes”}) = 1/4$ and $\mu_{\neg\mathsf{wall}(10),\geq 12}(\text{“Yes”}) = 0$. Let $\mathcal{E}$ be the corresponding generalized evidence space. It is easy to check that $\mathcal{W}_{\mathcal{E}}(\text{“Yes”}, \mathsf{wall}(10)) = \{4/5, 1, 3/4\}$, and thus $\underline{w}_{\mathcal{E}}(\text{“Yes”}, \mathsf{wall}(10)) = 3/4$ and $\overline{w}_{\mathcal{E}}(\text{“Yes”}, \mathsf{wall}(10)) = 1$. Similarly, $\mathcal{W}_{\mathcal{E}}(\text{“Yes”}, \neg\mathsf{wall}(10)) = \{1/5, 1/4, 0\}$, and thus $\underline{w}_{\mathcal{E}}(\text{“Yes”}, \neg\mathsf{wall}(10)) = 0$ and $\overline{w}_{\mathcal{E}}(\text{“Yes”}, \neg\mathsf{wall}(10)) = 1/4$. In particular, if the algorithm answers “Yes” to a query $\mathsf{wall}(10)$, the evidence supports the hypothesis that the wall is indeed at a distance less than 10 from the robot.

The primality example can be dealt with in the same way. Take $\mathcal{H} = \{\mathsf{prime}, \neg\mathsf{prime}\}$. There is a single probability measure μ_{prime} associated with the hypothesis prime, namely $\mu_{\mathsf{prime}}(\text{“Yes”}) = 1$; intuitively, if the number is prime, the knowledge algorithm always returns the right answer. In contrast, there are a number of different probability measures $\mu_{\neg\mathsf{prime},n}$ associated with the hypothesis $\neg\mathsf{prime}$, one per composite number n, where we take $\mu_{\neg\mathsf{prime},n}(\text{“Yes”})$ to be the probability that the algorithm says “Yes” when the composite number n is in Alice's local state. Note that this probability is different for every composite n, and depends on number-theoretic properties of n. Moreover, Alice is unlikely to know the actual probability $\mu_{\neg\mathsf{prime},n}$. For now, we view the set $\{\mu_{\neg\mathsf{prime},n} \mid n \text{ is composite}\}$ as the set of probabilities used by an “ideal” agent, who knows these probabilities. We consider how to model the set of probabilities of a “less-than-ideal” agent in Section 5.2. Let $\mathcal{E}$ be the corresponding generalized evidence space. Then $\mathcal{W}_{\mathcal{E}}(\text{“Yes”}, \mathsf{prime}) = \{1/(1 + \mu_{\neg\mathsf{prime},n}(\text{“Yes”})) \mid n \text{ composite}\}$. It is a known fact of number theory that, for all composite n, $\mu_{\mathsf{prime},n}(\text{“Yes”}) \leq 1/2$. since $\mu_{\neg\mathsf{prime},n}(\text{“Yes”})\ 1/2$ for all composite n. Thus, $r \geq 2/3$

for all $r \in \mathcal{W}_{\mathcal{E}}(\text{"Yes"}, \mathsf{prime})$, so $\underline{w}_{\mathcal{E}}(\text{"Yes"}, \mathsf{prime}) \geq 2/3$. Similarly, $\mathcal{W}_{\mathcal{E}}(\text{"Yes"}, \neg\mathsf{prime}) = \{\mu_{\mathsf{prime},n}(\text{"Yes"})/(\mu_{\mathsf{prime},n}(\text{"Yes"})+1) \mid n \text{ composite}\}$; since $\mu_{\neg\mathsf{prime},n}(\text{"Yes"}) \leq 1/2$ for all composite n, we have that $r \leq 1/3$ for all $r \in \mathcal{W}_{\mathcal{E}}(\text{"Yes"}, \neg\mathsf{prime})$, and thus $\overline{w}_{\mathcal{E}}(\text{"Yes"}, \neg\mathsf{prime}) \leq 1/3$. Therefore, if the algorithm answers "Yes" to a query prime, the evidence supports the hypothesis that the number in Alice's local state is indeed prime. Note that this way of modeling the example does not assume that Alice can see the number in her local state. We consider another way of capturing this example in Section 5.2, where we assume the number is in Alice's state.

5.2 Evidence for Randomized Knowledge Algorithms

We are now ready to discuss randomized knowledge algorithms. What does a "Yes" answer to a query φ given by an "almost sound" knowledge algorithm tell us about φ? As the discussion in Section 5.1 indicates, a "Yes" answer to a query φ provides evidence for the hypotheses φ and $\neg\varphi$. This can be made precise by associating an evidence space at every state of the model to capture the evidence provided by the knowledge algorithm. To simplify the presentation, we restrict our attention to knowledge algorithms that are φ-complete. (While it is possible to deal with general knowledge algorithms that also can return "?" using these techniques, we already saw that the logic does not let us distinguish between a knowledge algorithm returning "No" and a knowledge algorithm returning "?"; they both result in lack of algorithmic knowledge. In the next section, where we define the notion of a reliable knowledge algorithm, reliability will be characterized in terms of algorithmic knowledge, and thus the definition will not distinguish between a knowledge algorithm returning "No" or "?". In order to establish a link between the notion of reliability and evidence, it is convenient to either consider φ-complete algorithms, or somehow identify the answers "No" and "?". We choose the former.) Note that the knowledge algorithms described in the examples at the beginning of this section are all complete for their respective hypotheses. We further assume that the truth of φ depends only on the state, and not on coin tosses, that is, φ does not contain occurrences of the X_i operator.

Our goal is to associate, with every local state ℓ of agent i in M, an evidence space over the hypotheses $\{\varphi, \neg\varphi\}$ and the observations $\{\text{"Yes"}, \text{"No"}\}$. Let $S_\ell = \{s \mid L_i(s) = \ell\}$ be the set of states where agent i has local state ℓ. At every state s of S_ℓ, let $\mu_s(ob) = \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \ell, s, v'_i) = ob\})$. Intuitively, μ_s gives the probability of observing "Yes" and "No" in state s. If the agent considers only φ possible in local state ℓ, that is, if only φ holds at states in S_ℓ, then the agent implicitly knows φ. We can view this as the degenerate case where the evidence space has a single hypothesis, and the evidence for that single hypothesis given any observation is always 1. Assume that only φ is possible. Then we take the evidence space to be $\mathcal{E}_{A_i,\varphi,\ell} = (\{\varphi\}, \{\text{"Yes"}, \text{"No"}\}, \{\mu_s \mid s \in S_\ell\})$. We proceed in a similar way if $\neg\varphi$ holds at all states in S_ℓ.

If the agent considers both φ and $\neg\varphi$ possible, then we define the probability of answers of the knowledge algorithm to the query φ at every state of s; this probability will be used to derive the sets of probabilities corresponding to the hypotheses φ and $\neg\varphi$. Let $S_{\ell,\varphi} = \{s \in S_\ell \mid (N, s) \models \varphi\}$, and $S_{\ell,\neg\varphi} = \{s \in S_\ell \mid (N, s) \models \neg\varphi\}$. (Recall that φ depends only on the state, and not on coin tosses.) We then take

$$\mathcal{E}_{\mathtt{A}_i,\varphi,\ell} = (\{\varphi, \neg\varphi\}, \{\text{"Yes"}, \text{"No"}\}, \{\{\mu_s \mid s \in S_{\ell,\varphi}\}, \{\mu_s \mid s \in S_{\ell,\neg\varphi}\}\})$$

as the evidence space for local state ℓ. Since the agent does not know which state $s \in S_\ell$ is the true state, he must consider all these probabilities in his evidence space.

To be able to talk about evidence within the logic, we introduce operators to capture the lower and upper evidence provided by the knowledge algorithm of agent i, $\underline{\mathrm{Ev}}_i(\varphi)$ and $\overline{\mathrm{Ev}}_i(\varphi)$, read "$i$'s lower (resp., upper) weight of evidence for φ."

$$(N, s, v) \models \underline{\mathrm{Ev}}_i(\varphi) \geq \alpha \text{ if } \underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi) \geq \alpha$$

$$(N, s, v) \models \overline{\mathrm{Ev}}_i(\varphi) \geq \alpha \text{ if } \overline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi) \geq \alpha.$$

Of course, the above semantics are defined only if φ is one of the hypotheses in the evidence space $\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}$. We similarly define $(N, s, v) \models \underline{\mathrm{Ev}}_i(\varphi) \leq \alpha$ and $(N, s, v) \models \overline{\mathrm{Ev}}_i(\varphi) \leq \alpha$. Our restrictions on the probability ν on derandomizers ensures that $\underline{\mathrm{Ev}}_i(\varphi)$ and $\overline{\mathrm{Ev}}_i(\varphi)$ are always defined, despite the weight of evidence not being defined when the denominator is zero. Intuitively, since the knowledge algorithm $\mathtt{A}_i^d(\varphi, L_i(s), s, v_i)$ returns a result ob at state s, our restriction on ν ensures that we have either $\mu_s(ob) > 0$ for $s \in S_{L_i(s),\varphi}$ or $s \in S_{L_i(s),\neg\varphi}$. In other words, if we actually observe ob, then the probability of observing ob must be greater than 0.

Proposition 5.1: *For all probabilistic algorithmic knowledge structures N where $\mathtt{A}_i$ is φ-complete, states s of N, and derandomizers v, if $\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)} = (\mathcal{H}, \mathcal{O}, \{\mathcal{P}_h \mid h \in \mathcal{H}\})$ and $\varphi \in \mathcal{H}$ then both $\underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi)$ and $\overline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi)$ are defined.*

This definition of evidence has a number of interesting properties. For instance, full evidence in support of a formula φ essentially corresponds to establishing the truth of φ.

Proposition 5.2: *For all probabilistic algorithmic knowledge structures N where $\mathtt{A}_i$ is φ-complete, all states s where $\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)} = (\mathcal{H}, \mathcal{O}, \{\mathcal{P}_h \mid h \in \mathcal{H}\})$ and $\varphi \in \mathcal{H}$, and all derandomizers v,*

$$(N, s, v) \models \underline{\mathrm{Ev}}_i(\varphi) = 1 \Rightarrow \varphi.$$

Consider again the examples at the beginning of the section, this time deriving the evidence spaces following the recipe above. For the double-headed coin example, consider a structure N with two states s_1 and s_2, where the coin is double-headed at state s_1, and fair at state s_2. Hence, $(N, s_1, v) \models \mathsf{dhead}$, and $(N, s_2, v) \models \neg\mathsf{dhead}$. Assume that Bob has the same local state ℓ_0 at both states. Thus, $S_{\ell_0} = \{s_1, s_2\}$, $S_{\ell_0,\mathsf{dhead}} = \{s_1\}$, and $S_{\ell_0,\neg\mathsf{dhead}}\{s_2\}$. The evidence space is therefore $\mathcal{E} = (\{\mathsf{dhead}, \neg\mathsf{dhead}\}, \{\text{"Yes"}, \text{"No"}\}, \{\mu_{s_1}\}, \{\mu_{s_2}\})$, where $\mu_{s_1}(\text{"Yes"}) = 1$, and $\mu_{s_2}(\text{"Yes"}) = 1/2$. We can check that, for all states (s, v) where $\mathtt{A}_{\mathrm{Bob}}(\mathsf{dhead}, \ell_0, s, v_{\mathrm{Bob}}) = \text{"Yes"}$, $(N, s, v) \models \underline{\mathrm{Ev}}(\mathsf{dhead}) = 2/3$ and $(N, s, v) \models \overline{\mathrm{Ev}}(\mathsf{dhead}) = 2/3$, while at all states (s, v) where $\mathtt{A}_{\mathrm{Bob}}(\mathsf{dhead}, \ell_0, s, v_{\mathrm{Bob}}) = \text{"No"}$, then $(N, s, v) \models \underline{\mathrm{Ev}}(\mathsf{dhead}) = 0$ and $(N, s, v) \models \overline{\mathrm{Ev}}(\mathsf{dhead}) = 0$. In other words, the algorithm answering "Yes" provides evidence for the coin being double-headed, while the algorithm answering "No" essentially says that the coin is fair.

For the probabilistic sensor example, consider a structure N with states s_m ($m \geq 1$), where the wall at state s_m is at distance m from the robot. Suppose we are interested in the hypotheses $\mathsf{wall}(10)$ and $\neg\mathsf{wall}(10)$, so that $(N, s_m, v) \models \mathsf{wall}(10)$ if and only $m \leq 10$. The local state of the robot is the same at every state, say ℓ_0. Thus, $S_{\ell_0} = \{s_m \mid m \geq 1\}$, $S_{\ell_0,\mathsf{wall}(10)} = \{s_m \mid 1 \leq m \leq 10\}$, and $S_{\ell_0,\neg\mathsf{wall}(10)} = \{s_m \mid m \geq 11\}$. The evidence space is therefore $\mathcal{E} = (\{\mathsf{wall}(10), \neg\mathsf{wall}(10)\}, \{\text{"Yes"}, \text{"No"}\}, \{\mu_m \mid 1 \leq m \leq 10\}, \{\mu_m \mid m \geq 11\})$, where μ_m is given by $\mu_m(\text{"Yes"}) = 1$ if $m \leq 9$, $\mu_{10}(\text{"Yes"}) = 3/4$, $\mu_{11}(\text{"Yes"}) = 1/4$, and $\mu_m(\text{"Yes"}) = 0$ if $m \geq 12$. It is straightforward to compute that, for all states (s, v) where $\mathtt{A}_{\mathrm{Robot}}(\mathsf{wall}(10), \ell_0, s, v_{\mathrm{Robot}}) =$

"Yes", $(N,s,v) \models \underline{\mathrm{Ev}}(\mathsf{wall}(10)) \geq 3/4$ and $(N,s,v) \models \overline{\mathrm{Ev}}(\mathsf{wall}(10)) \leq 1$, while at all states (s,v) where $\mathtt{A}_{\mathrm{Robot}}(\mathsf{wall}(10), \ell_0, s, v_{\mathrm{Robot}}) =$ "No", $(N,s,v) \models \overline{\mathrm{Ev}}(\mathsf{wall}(10)) \leq 1/4$ and $(N,s,v) \models \underline{\mathrm{Ev}}(\mathsf{wall}(10)) \geq 0$. In other words, the algorithm answering "Yes" provides evidence for the wall being at distance at most 10, while the algorithm answering "No" provides evidence for the wall being further away.

Finally, we consider the primality example. Here, an interesting phenomenon arises. We cannot naively take as states s_n, with n a natural number representing the number in Alice's local state: for any local state ℓ_n (where n is the number in the local state), either the number n is prime, or it is not. Therefore, for every ℓ, the evidence space at S_ℓ is the degenerate evidence space with a single hypothesis (either prime, or $\neg\mathsf{prime}$, depending on whether n is prime or not), which yields trivial evidence (either 0 or 1). This way of modeling things fails to capture the fact that we care about what Alice *subjectively* thinks about the number. Initially, she does not know whether the number is prime or not. Intuitively, this means that for every n, she must consider possible a world where n is prime and a world where n is not prime. Moreover, if n is composite, she does not know how likely the knowledge algorithm is to return "Yes". This depends on the number of witnesses a there are such that $P(n,a) = 0$. While it is known that there are at least $n/2$ such witnesses, the exact number is not known. To model this, we need to use *impossible possible worlds*, worlds where the usual laws of arithmetic, for instance, do not hold [Hintikka 1975]. Thus, consider the structure N with states $s_{n,\mathsf{prime}}$ and $s_{n,\neg\mathsf{prime},\alpha}$ (for $n \geq 2$, $\alpha \in [1/2,1]$). Intuitively, $s_{n,\neg\mathsf{prime},\alpha}$ is the state where the fraction of witnesses is α. We are implicitly assuming that nothing is known about the fraction of witnesses except that it is at least $1/2$. Clearly, if there is more information about the number of witnesses, then the set of states should be modified appropriately. At states $s_{n,\mathsf{prime}}$ and $s_{n,\neg\mathsf{prime},\alpha}$, Alice has the same local state ℓ_n, with the number n store in her local state; however, $(N, s_{n,\mathsf{prime}}, v) \models \mathsf{prime}$, while $(N, s_{n,\neg\mathsf{prime},\alpha}, v) \models \neg\mathsf{prime}$. For a local state ℓ_n, define $S_{\ell_n} = \{s_{n,\mathsf{prime}}\} \cup \{s_{n,\neg\mathsf{prime},\alpha} \mid \alpha \in [1/2,1]\}$, $S_{\ell_n,\mathsf{prime}} = \{s_{n,\mathsf{prime}}\}$, and $S_{\ell_n,\neg\mathsf{prime}} = \{s_{n,\neg\mathsf{prime},\alpha} \mid \alpha \in [1/2,1]\}$. The evidence space for local state ℓ_n is therefore $\mathcal{E}_n = (\{\mathsf{prime}, \neg\mathsf{prime}\}, \{\text{"Yes"}, \text{"No"}\}, \{\{\mu_{n,\mathsf{prime}}\}, \{\mu_{n,\neg\mathsf{prime},\alpha} \mid \alpha \in [1/2,1]\}\})$, where $\mu_{n,\mathsf{prime}}(\text{"Yes"}) = 1$ and $\mu_{n,\neg\mathsf{prime},\alpha}(\text{"Yes"}) = 1-\alpha$ and With this natural evidence space, we can check that, for all states (s,v) where $\mathtt{A}_{\mathrm{Alice}}(\mathsf{prime}, s_{\mathrm{Alice}}, s, v) =$ "Yes", $(N,s,v) \models \underline{\mathrm{Ev}}(\mathsf{prime}) \geq 2/3$, while at all states (s,v) where $\mathtt{A}_{\mathrm{Alice}}(\mathsf{prime}, s_{\mathrm{Alice}}, s, v) =$ "No", $(N,s,v) \models \overline{\mathrm{Ev}}(\mathsf{prime}) = 0$. In other words, the algorithm returning "Yes" to the query whether the number in Alice's local state is prime provides evidence for the number being prime, while the algorithm returning "No" essentially says that the number is composite.

5.3 Reliable Randomized Knowledge Algorithms

As we saw in the previous section, a "Yes" answer to a query φ given by an "almost sound" knowledge algorithm provides evidence for φ. We now examine the extent to which we can characterize the evidence provided by a randomized knowledge algorithm. To make this precise, we need to first characterize how reliable the knowledge algorithm is. A randomized knowledge algorithm $\mathtt{A}_i$ is (α,β)*-reliable for* φ *in* N *(for agent* i*)* if $\alpha, \beta \in [0,1]$ and for all states s and derandomizers v,

- $(N,s,v) \models \varphi$ implies $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{"Yes"}\}) \geq \alpha$;
- $(N,s,v) \models \neg\varphi$ implies $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{"Yes"}\}) \leq \beta$.

These conditions are equivalent to $(N, s, v) \models \varphi$ implying $(N, s, v) \models \Pr(X_i\varphi) \geq \alpha$ and $(N, s, v) \models \neg\varphi$ implying $(N, s, v) \models \Pr(X_i\varphi) \leq \beta$. In other words, if φ is true at state s, then an (α, β)-reliable algorithm says "Yes" to φ at s with probability at least α (and hence is right when it answers "Yes" to query φ with probability at least α); on the other hand, if φ is false, it says "Yes" with probability at most β (and hence is wrong when it answer "Yes" to query φ with probability at most β). The primality testing knowledge algorithm is $(1, 1/2)$-reliable for prime.

The intuition here is that (α, β)-reliability is a way to bound the probability that the knowledge algorithm is wrong. The knowledge algorithm can be wrong in two ways: it can answer "No" or "?" to a query φ when φ is true, and it can answer "Yes" to a query φ when φ is not true. If a knowledge algorithm is (α, β)-reliable, then the probability that it answers "No" or "?" when the answer should be "Yes" is at most $1 - \alpha$: the probability that it answers "Yes" when it should not is at most β.

We can now capture the relationship between reliable knowledge algorithms and evidence. As we saw in Section 5.2, both formulas φ and $\neg\varphi$ should be hypotheses in the evidence space at every state of interest if the evidence is to be truly meaningful; if there is a single hypothesis, every observation provides evidence for that hypothesis. There is a single hypothesis in the evidence space at a state s whenever one of the formulas is implicitly known to the agent at state s. The properties characterizing the relationship between algorithmic knowledge and evidence need to check for this; we use the formula $K_i^{\pm}\varphi$ as an abbreviation for $(K_i\varphi) \vee (K_i\neg\varphi)$, which holds at states where φ or $\neg\varphi$ is implicitly known to the agent.

Proposition 5.3: *If $\mathtt{A}_i$ is φ-complete and (α, β)-reliable for φ in N then*

(a) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta}$ *if* $(\alpha, \beta) \neq (0, 0)$*;*

(b) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) = 1$ *if* $(\alpha, \beta) = (0, 0)$*;*

(c) $N \models \neg X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}$ *if* $(\alpha, \beta) \neq (1, 1)$*;*

(d) $N \models \neg X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) = 0$ *if* $(\alpha, \beta) = (1, 1)$.

In other words, if a randomized knowledge algorithm says "Yes", then that provides evidence for φ being true. The weight of evidence depends on the reliability. Similarly, if the randomized knowledge algorithm says "No", there is less evidence in favor of φ being true.

Proposition 5.3 becomes interesting in the context of well-known classes of randomized algorithms [Motwani and Raghavan 1995]. An **RP** (random polynomial-time) algorithm is a polynomial-time randomized algorithm that is $(1/2, 0)$-reliable. It thus follows from Proposition 5.3 that if $\mathtt{A}_i$ is an **RP** algorithm, then $X_i\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) = 1$ and $\neg X_i\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) \leq 1/3$ are both true at all states of N where neither φ nor $\neg\varphi$ are known to i. By Proposition 5.2, $\underline{\mathrm{Ev}}_i(\varphi) = 1 \Rightarrow \varphi$ is valid, and thus we have $X_i\varphi \Rightarrow \varphi$, as expected. Similarly, a **BPP** (bounded-error probabilistic polynomial-time) algorithm is a polynomial-time randomized algorithm that is $(3/4, 1/4)$-reliable. Thus, by Proposition 5.3, if $\mathtt{A}_i$ is a **BPP** algorithm, then $X_i\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) \geq 3/4$ and $\neg X_i\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) \leq 1/4$ are both true at every state of N where neither φ nor $\neg\varphi$ is known to i.

Notice that Proposition 5.3 talks about the evidence that the knowledge algorithm provides for φ. Intuitively, we might expect some kind of relationship between the evidence for φ and the evidence for $\neg\varphi$. A plausible relationship would be that high evidence for φ implies low evidence for $\neg\varphi$, and low evidence for φ implies high evidence for $\neg\varphi$. Unfortunately, given the definitions in this section, this is not the case. Evidence for φ is completely unrelated to evidence for $\neg\varphi$. Roughly speaking, this is because evidence for φ is measured by looking at the results of the knowledge algorithm when queried for φ, and evidence for $\neg\varphi$ is measured by looking at the results of the knowledge algorithm when queried for $\neg\varphi$. There is nothing in the definition of a knowledge algorithm that says that the answers of the knowledge algorithm to queries φ and $\neg\varphi$ need to be related in any way.

A relationship between evidence for φ and evidence for $\neg\varphi$ can be established by considering knowledge algorithms that are "well-behaved" with respect to negation. There is a natural way to define the behavior of a knowledge algorithm on negated formulas. Intuitively, a strategy to evaluate $\mathtt{A}_i(\neg\varphi, \ell, s, v_i)$ is to evaluate $\mathtt{A}_i(\varphi, \ell, s, v_i)$, and returns the negation of the result. There is a choice to be made in the case when the $\mathtt{A}_i$ returns "?" to the query for φ. One possibility is to return "?" to the query for $\neg\varphi$ when the query for φ returns "?"; another possibility is to return "Yes" is the query for φ returns "?". Say that a randomized knowledge algorithm $\mathtt{A}$ *weakly respects negation* if, for all local states ℓ and derandomizers v,

$$\mathtt{A}^d(\neg\varphi, \ell, s, v_i) = \begin{cases} \text{“Yes”} & \text{if } \mathtt{A}^d(\varphi, \ell, s, v_i) = \text{“No”} \\ \text{“No”} & \text{if } \mathtt{A}^d(\varphi, \ell, s, v_i) = \text{“Yes”} \\ \text{“?”} & \text{if } \mathtt{A}^d(\varphi, \ell, s, v_i) = \text{“?”}. \end{cases}$$

Similarly, say that a randomized knowledge algorithm $\mathtt{A}$ *strongly respects negation*, if for all local states ℓ and derandomizers v,

$$\mathtt{A}^d(\neg\varphi, \ell, s, v_i) = \begin{cases} \text{“Yes”} & \text{if } \mathtt{A}^d(\varphi, \ell, s, v_i) \neq \text{“Yes”} \\ \text{“No”} & \text{if } \mathtt{A}^d(\varphi, \ell, s, v_i) = \text{“Yes”}. \end{cases}$$

Note that if $\mathtt{A}_i$ is φ-complete, then the output of $\mathtt{A}_i$ on input $\neg\varphi$ is the same whether $\mathtt{A}_i$ weakly or strongly respects negation. Say $\mathtt{A}_i$ *respects negation* if it weakly or strongly respects negation. If $\mathtt{A}_i$ is φ-complete and respects negation, then $\mathtt{A}_i$ is $\neg\varphi$-complete.

Our first result shows that for knowledge algorithms that respect negation, reliability for φ is related to reliability for $\neg\varphi$:

Proposition 5.4: *If $\mathtt{A}_i$ respects negation, is φ-complete, and is (α, β)-reliable for φ in N, then $\mathtt{A}_i$ is (α, β)-reliable for φ in N if and only if $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N.*

It is easy to check that if $\mathtt{A}_i$ is φ-complete and respects negation, then $X_i\varphi \Leftrightarrow \neg X_i\neg\varphi$ is a valid formula. Combined with Proposition 5.3, this yields the following results.

Proposition 5.5: *If $\mathtt{A}_i$ respects negation, is φ-complete, and is (α, β)-reliable for φ in N, then*

(a) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta} \wedge \overline{\mathrm{Ev}}_i(\neg\varphi) \leq \frac{\beta}{\alpha+\beta}\right)$ *if* $(\alpha, \beta) \neq (0, 0)$*;*

(b) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\varphi) = 1 \wedge \overline{\mathrm{Ev}}_i(\neg\varphi) = 0\right)$ *if* $(\alpha, \beta) = (0, 0)$*;*

(c) $N \models X_i\neg\varphi \wedge K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\neg\varphi) \geq \frac{1-\beta}{2-(\alpha+\beta)} \wedge \overline{\mathrm{Ev}}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}\right)$ *if* $(\alpha, \beta) \neq (1, 1)$*;*

(d) $N \models X_i\neg\varphi \wedge K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\neg\varphi) \geq \frac{1}{2} \wedge \overline{\mathrm{Ev}}_i(\varphi) \leq \frac{1}{2}\right)$ *if* $(\alpha, \beta) = (1, 1)$.

6 Conclusion

The goal of this paper is to understand what the evidence provided by a knowledge algorithm tells us. To take another example from security, consider an enforcement mechanism used to detect and react to intrusions in a system. Such an enforcement mechanism uses algorithms that analyze the behavior of users and attempt to recognize intruders. While the algorithms may sometimes be wrong, they are typically reliable, in our sense, with some associated probabilities. Clearly the mechanism wants to make sensible decisions based on this information, How should it do this? What actions should the system take based on a report that a user is an intruder?

If we have a probability on the hypotheses, evidence can be used to update this probability. More precisely, as shown in [Halpern and Fagin 1992], evidence can be viewed as a function from priors to posteriors. For example, if the (cumulative) evidence for n being a prime is α and the prior probability that n is prime is β, then a straightforward application of Bayes' rule tells us that the posterior probability of n being prime (that is, the probability is $(\alpha\beta)/(\alpha\beta + (1-\alpha)(1-\beta))$. Therefore, if we have a prior probability on the hypotheses, including the formula φ, then we can decide to perform an action when the posterior probability of φ is high enough. (We believe a similar interpretation holds for the evidence expressed by $\underline{w}_{\mathcal{E}}$ and $\overline{w}_{\mathcal{E}}$; we hope to report on this topic in future work.) However, what can we do when there is no probability distribution on the hypotheses, as in the primality example at the beginning of this section? The probabilistic interpretation of evidence still lets us guide our decisions. As before, we assume that if the posterior probability of φ is high enough, we will act as if φ holds. The problem, of course, is that we do not have a prior probability. However, the evidence tells us what prior probabilities we must to be willing to assume for the posterior probability to be high enough. For example, a "Yes" from a $(.999, .001)$-reliable algorithm for φ says that as long as the prior probability of φ is at least .01, then the posterior is at least .9. This may be sufficient assurance for an agent to act.

Of course, it is also possible to treat evidence as primitive, and simply decide to go for the hypothesis for which there is more evidence, or for the hypothesis for which evidence is above a certain threshold. It would in fact be of independent interest to study the properties of a theory of decisions based on a primitive notion of evidence. We leave this to future work.

Acknowledgments

Thanks to Tugkan Batu and Hubie Chen for discussions. Authors supported in part by NSF under grant CTC-0208535, by ONR under grants N00014-00-1-03-41 and N00014-01-10-511, by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grant N00014-01-1-0795, and by AFOSR under grant F49620-02-1-0101.

A Proofs

Proposition 3.1: *Let $N = (S, \pi, L_1, \dots, L_n, \mathtt{A}_1^d, \dots, \mathtt{A}_n^d, \nu)$ be a probabilistic algorithmic knowledge, with $\mathtt{A}_1, \dots, \mathtt{A}_n$ deterministic. Let $M = (S, \pi, L_1, \dots, L_n, \mathtt{A}_1, \dots, \mathtt{A}_n)$. If there are no occurrences of* $\Pr$ *in φ, then for all $s \in S$ and all $v \in V$, $(N, s, v) \models \varphi$ if and only if $(M, s) \models \varphi$.*

Proof: The key observation here is that if a knowledge algorithm $\mathtt{A}$ is deterministic, then for all $v \in V$, $\mathtt{A}^d(\varphi, \ell, s, v_i) = \mathtt{A}(\varphi, \ell, s)$. The result then follows easily by induction on the structure of

φ. If φ is p, then $(N,s,v) \models p$ if and only if $\pi(s)(p) =$ **true** if and only if $(M,s) \models p$. If φ is $\psi_1 \wedge \psi_2$, then $(N,s,v) \models \psi_1 \wedge \psi_2$ if and only if $(N,s,v) \models \psi_1$ and $(N,s,v) \models \psi_2$ if and only if $(M,s) \models \psi_1$ and $(M,s) \models \psi_2$ (by the induction hypothesis) if and only if $(M,s) \models \psi_1 \wedge \psi_2$. If φ is $\neg\psi$, then $(N,s,v) \models \neg\psi$ if and only if $(N,s,v) \not\models \psi$ if and only $(M,s) \not\models \psi$ (by the induction hypothesis) if and only if $(M,s) \models \neg\psi$. If φ is $K_i\psi$, consider first $(N,s,v) \models K_i\psi$, that is, for all $v' \in V$ and all $t \sim_i s$, $(N,t,v') \models \psi$; by the induction hypothesis, this means that for all $t \sim_i s$, $(M,t) \models \psi$, that is, $(M,s) \models K_i\psi$. Conversely, assume $(M,s) \models K_i\psi$, so that for all $t \sim_i s$, $(M,t) \models \psi$; by the induction hypothesis, for every $t \sim_i s$, we have $(N,t,v') \models \psi$ for all $v' \in V$, and thus $(N,s,v) \models K_i\psi$. If φ is $X_i\psi$, then $(N,s,v) \models X_i\psi$ if and only if $\mathtt{A}_i^d(\psi, L_i(s), s, v_i) =$ "Yes" if and only if $\mathtt{A}_i(\psi, L_i(s), s) =$ "Yes" (since $\mathtt{A}_i$ is deterministic) if and only if $(M,s) \models X_i\psi$. ∎

Proposition 3.2: *Let $N = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ be a probabilistic algorithmic knowledge structure, and let $M = (S, \pi, L_1, \ldots, L_n, \mathtt{A}'_1, \ldots, \mathtt{A}'_n)$ be an algorithmic knowledge structure where $\mathtt{A}'_1, \ldots, \mathtt{A}'_n$ are arbitrary deterministic knowledge algorithms. If there are no occurrences of X_i and $\Pr$ in φ, then for all $s \in S$ and all $v \in V$, $(N,s,v) \models \varphi$ if and only if $(M,s) \models \varphi$.*

Proof: This result in fact follows from the proof of Proposition 3.1, since the only use of the assumption that knowledge algorithms are deterministic is in the inductive step for subformulas of the form $X_i\psi$. ∎

Proposition 4.2: *Suppose that $N = (S, \pi, L_1, \ldots, L_n, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ is a probabilistic algorithmic knowledge security structure with an adversary as agent i and that $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY+rg}(n)}$. Let K be the number of distinct keys used in the messages in the adversary's local state ℓ (that is, the number of keys used in the messages that the adversary has intercepted at a state s with $L_i(s) = \ell$). Suppose that $K/|\mathcal{K}| < 1/2$ and that ν is the uniform distribution on sequences of coin tosses. If $(N,s,v) \models \neg K_i X_i(\mathsf{has}_i(\mathsf{m}))$, then $(N,s,v) \models \Pr(X_i(\mathsf{has}_i(\mathsf{m}))) < 1 - e^{-2nK/|\mathcal{K}|}$. Moreover, if $(N,s,v) \models X_i(\mathsf{has}_i(\mathsf{m}))$ then $(N,s,v) \models \mathsf{has}_i(\mathsf{m})$.*

Proof: It is not hard to show that the n keys that the adversary guesses do no good at all if none of them match a key used in a message intercepted by the adversary. By assumption, K keys are used in messages intercepted by the adversary. The probability that a key chosen at random is one of these K is $K/|\mathcal{K}|$, since there are $|\mathcal{K}|$ keys altogether. Thus, the probability that a key chosen at random is not one of these K is $1 - (K/|\mathcal{K}|)$. The probability that none of the n keys chosen at random is one of these K is therefore $(1 - (K/|\mathcal{K}|))^n$. We now use some standard approximations. Note that $(1 - (K/|\mathcal{K}|))^n = e^{n \ln(1-(K/|\mathcal{K}|)}$, and

$$\ln(1-x) = -x - x2/2 - x3/3 - \cdots > -x - x2 - x3 - \cdots = -x/(1-x).$$

Thus, if $0 < x < 1/2$, then $\ln(1-x) > -2x$. It follows that if $K/|\mathcal{K}| < 1/2$, then $e^{n \ln(1-(K/|\mathcal{K}|)} > e^{-2nK/|\mathcal{K}|}$. Since the probability that a key chosen at random does not help to compute algorithmic knowledge is greater than $e^{-2nK/|\mathcal{K}|}$, the probability that it helps is less than $1 - e^{-2nK/|\mathcal{K}|}$.

Soundness of $\mathtt{A}_i$ with respect to $\mathsf{has}_i(\mathsf{m})$ follows from Proposition 4.1 (since soundness follows for arbitrary $\mathit{initkeys}(\ell) \subseteq \mathcal{K}$). ∎

Proposition 5.1: *For all probabilistic algorithmic knowledge structures N where $\mathtt{A}_i$ is φ-complete, states s of N, and derandomizers v, if $\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)} = (\mathcal{H}, \mathcal{O}, \{\mathcal{P}_h \mid h \in \mathcal{H}\})$ and $\varphi \in \mathcal{H}$ then both $\underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi)$ and $\overline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi)$ are defined.*

Proof: Given N, a state s of N, and a derandomizer v, $\underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi)$ and $\overline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i), \varphi)$ are defined if and only if $\mathcal{W}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}$ is nonempty. We consider two cases.

First, suppose that $\mathcal{H} = \{\varphi\}$, when the evidence space at s is degenerate. By our assumption on ν, we have $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \mathtt{A}_i^d(\varphi, L_i(s), s, v_i)\}) > 0$ (since v itself is such a derandomizer), from which it is easy to check that $\mathcal{W}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}} = \{1\} \neq \varnothing$, as required.

Second, suppose that $\mathcal{H} = \{\varphi, \neg\varphi\}$, then consider $S_{L_i(s),\varphi}$ and $S_{L_i(s),\neg\varphi}$, both nonempty. $\mathcal{W}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}$ is nonempty if and only if there exists at least one state s' in either $S_{L_i(s),\varphi}$ or $S_{L_i(s'),\neg\varphi}$ such that $\mu_{s'}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i)) > 0$. If $\mathtt{A}_i^d(\varphi, L_i(s), s, v_i) =$ "Yes", then either $(N, s, v) \models \varphi$, in which case s is in $S_{L_i(s),\varphi}$, and by assumption on ν, $\mu_s(\text{"Yes"}) > 0$ (since v is itself a randomizer where the algorithm answers "Yes"), or $(N, s, v) \models \neg\varphi$, in which case s is in $S_{L_i(s),\neg\varphi}$, and by assumption on ν, $\mu_s(\text{"Yes"}) > 0$ (since v is itself a randomizer where the algorithm answer "Yes"). A similar argument can be used if $\mathtt{A}_i^d(\varphi, L_i(s), s, v_i) =$ "No", showing that $\mu_{s'}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i)) > 0$ for some s' in either $S_{L_i(s),\varphi}$ or $S_{L_i(s),\neg\varphi}$, as required. ∎

Proposition 5.2: *For all probabilistic algorithmic knowledge structures N where $\mathtt{A}_i$ is φ-complete, all states s where $\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)} = (\mathcal{H}, \mathcal{O}, \{\mathcal{P}_h \mid h \in \mathcal{H}\})$ and $\varphi \in \mathcal{H}$, and all derandomizers v,*

$$(N, s, v) \models \underline{\mathrm{Ev}}_i(\varphi) = 1 \Rightarrow \varphi.$$

Proof: We consider two cases. First, suppose that $\mathcal{H} = \{\varphi\}$, when the evidence space at s is degenerate. This means that $(N, s', v) \models \varphi$ for all states s' such that $L_i(s') = L_i(s)$, including s itself, so that $(N, s, v) \models \varphi$. (Recall that φ depends only on the state.)

Second, suppose that $\mathcal{H} = \{\varphi, \neg\varphi\}$. If $(N, s, v) \models \underline{\mathrm{Ev}}_i(\varphi) = 1$, then $\underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\mathtt{A}_i^d(\varphi, L_i(s), v_i), \varphi) = 1$. By definition of $\underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}$, this implies that for all $s' \in S_{L_i(s),\neg\varphi}$, $\mu_{s'}(\mathtt{A}_i^d(\varphi, L_i(s), s, v_i)) = 0$. By our assumption ν, this means that there is no state s' (and derandomizer v') such that $L_i(s') = L_i(s)$ and $(N, s', v') \models \neg\varphi$ where $\mathtt{A}_i^d(\varphi, L_i(s), s', v'_i) = \mathtt{A}_i^d(\varphi, L_i(s), s, v_i)$. Thus, we cannot have $(N, s, v) \models \neg\varphi$ (otherwise, s is just such a state shown not to exist). Hence, $(N, s, v) \models \varphi$, as required. ∎

To prove Propositions 5.3 and 5.5, the following lemma, alluded to in the text, is useful. It describes the relationship between the reliability of a knowledge algorithm and the properties of probability measures in $\mathcal{P}_\varphi$ and $\mathcal{P}_{\neg\varphi}$ that appear in the evidence space $\mathcal{E}_{\mathtt{A}_i,\varphi,\ell}$, as defined in Section 5.3.

Lemma A.1: *If $\mathtt{A}_i$ is φ-complete and (α, β)-reliable for φ in N, then for all states s, s' with $L_i(s) = L_i(s')$, $(N, s) \models \varphi$, and $(N, s') \models \neg\varphi$, we have $\mu_s(\text{"Yes"}) \geq \alpha$, $\mu_s(\text{"No"}) \leq 1 - \alpha$, $\mu_{s'}(\text{"Yes"}) \leq \beta$, and $\mu_{s'}(\text{"No"}) \geq 1 - \beta$.*

Proof: This is immediate from the definitions. By assumption, $s \in S_{L_i(s),\varphi}$ and $s' \in S_{L_i(s),\neg\varphi}$. By the definition of reliability, since for every v, $(N, s, v) \models \varphi$, then $\mu_s(\text{"Yes"}) = \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{"Yes"}\}) \geq \alpha$, and thus by φ-completeness, $\mu_s(\text{"No"}) = 1 - \mu_s(\text{"Yes"}) \leq 1 - \alpha$. Similarly, since for every v, $(N, s', v) \models \neg\varphi$, then $\mu_{s'}(\text{"Yes"}) = \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s', v'_i) = \text{"Yes"}\}) \leq \beta$, and thus by φ-completeness, $\mu_{s'}(\text{"No"}) = 1 - \mu_{s'}(\text{"Yes"}) \geq 1 - \beta$. ∎

The following lemma captures the algebraic relationships that are useful in assessing the evidence in Propositions 5.3 and 5.5.

Lemma A.2: *Suppose x, y, a, b are real numbers in $[0,1]$, such that x and y are not both 0, and a and b are not both 0. If $x \geq a$ and $y \leq b$, then $x/(x+y) \geq a/(a+b)$ and $y/(x+y) \leq b/(a+b)$.*

Proof: Note that $x(a+b) = xa + xb \geq xa + ay = a(x+y)$, so that $x/(x+y) \geq a/(a+b)$. Similarly, $y(a+b) = ya + yb \leq xb + yb = b(x+y)$, so that $y/(x+y) \leq b(a+b)$. ∎

Proposition 5.3: *If $\mathtt{A}_i$ is φ-complete and (α, β)-reliable for φ in N then*

(a) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta}$ *if* $(\alpha, \beta) \neq (0,0)$*;*

(b) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) = 1$ *if* $(\alpha, \beta) = (0,0)$*;*

(c) $N \models \neg X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}$ *if* $(\alpha, \beta) \neq (1,1)$*;*

(d) $N \models \neg X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) = 0$ *if* $(\alpha, \beta) = (1,1)$.

Proof: For part (a), suppose that $(\alpha, \beta) \neq (0,0)$. Let s be a state of N and let v be a derandomizer. If $(N, s, v) \models X_i\varphi \wedge \neg K_i^{\pm}\varphi$, then $\mathtt{A}_i^d(\varphi, L_i(s), v_i) =$ "Yes", and moreover the evidence space for local state $L_i(s)$ is not degenerate, that is, both φ and $\neg\varphi$ are considered possible by agent i at state s. Let s', s'' be two arbitrary states in $S_{L_i(s),\varphi}$ and $S_{L_i(s),\neg\varphi}$, respectively. By Lemma A.1, we have $\mu_{s'}(\text{"Yes"}) \geq \alpha$ and $\mu_{s''}(\text{"Yes"}) \leq \beta$. Therefore, by Lemma A.2, we have

$$\begin{aligned}&\mu_{s'}(\text{"Yes"})/(\mu_{s'}(\text{"Yes"}) + \mu_{s''}(\text{"Yes"}))\\ &\quad \geq \alpha/(\alpha+\beta).\end{aligned}$$

Since s' and s'' were arbitrary, this means that for all $w \in \mathcal{W}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}$, $w \geq \alpha/(\alpha+\beta)$, so that $\underline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\text{"Yes"}, \varphi) \geq \alpha/(\alpha+\beta)$. Thus, we have $(N, s, v) \models \underline{\mathrm{Ev}}_i(\varphi) \geq \alpha/(\alpha+\beta)$. Since s and v were arbitrary, we have $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \underline{\mathrm{Ev}}_i(\varphi) \geq \alpha/(\alpha+\beta)$.

For part (c), suppose that $(\alpha, \beta) \neq (1,1)$. If $(N, s, v) \models \neg X_i\varphi \wedge \neg K_i^{\pm}\varphi$, then $\mathtt{A}_i^d(\varphi, L_i(s), v_i) =$ "No" (since $\mathtt{A}_i$ is φ-complete), and moreover the evidence space for local state $L_i(s)$ is not degenerate, that is, both φ and $\neg\varphi$ are considered possible by agent i at state s. Let s', s'' be two arbitrary states in $S_{L_i(s),\varphi}$ and $S_{L_i(s),\neg\varphi}$, respectively. By Lemma A.1, we have $\mu_{s'}(\text{"No"}) \leq 1-\alpha$ and $\mu_{s''}(\text{"No"}) \geq 1-\beta$. Therefore, by Lemma A.2, we have

$$\begin{aligned}&\mu_{s'}(\text{"No"})/(\mu_{s'}(\text{"No"}) + \mu_{s''}(\text{"No"}))\\ &\quad \leq 1-\alpha/(2-(\alpha+\beta)).\end{aligned}$$

Since s' and s'' were arbitrary, this means that for all $w \in \mathcal{W}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}$, $w \leq 1-\alpha/(2-(\alpha+\beta))$, so that $\overline{w}_{\mathcal{E}_{\mathtt{A}_i,\varphi,L_i(s)}}(\text{"No"}, \varphi) \leq 1-\alpha/(2-(\alpha+\beta))$. Thus, we have $(N, s, v) \models \overline{\mathrm{Ev}}_i(\varphi) \leq 1-\alpha/(2-(\alpha+\beta))$. Since s and v were arbitrary, we have $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \overline{\mathrm{Ev}}_i(\varphi) \leq 1-\alpha/(2-(\alpha+\beta))$.

The proofs of parts (b) and (d) are similar and left to the reader. ∎

Proposition 5.4: *If $\mathtt{A}_i$ respects negation, is φ-complete, and is (α, β)-reliable for φ in N, then $\mathtt{A}_i$ is (α, β)-reliable for φ in N if and only if $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N.*

Proof: Suppose that $\mathtt{A}_i$ respects negation, is φ-complete, and is (α, β)-reliable for φ in N. Consider the reliability of $\mathtt{A}_i$ with respect to $\neg\varphi$. If $(N, s, v) \models \neg\varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) = \text{“No”}\}) \quad \text{by } \varphi\text{-completeness} \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad \geq 1 - \beta.
\end{aligned}$$

Similarly, if $(N, s, v) \models \varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) = \text{“No”}\}) \quad \text{by } \varphi\text{-completeness} \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{“Yes”}) \\
&\quad \leq 1 - \alpha.
\end{aligned}$$

Thus, $\mathtt{A}_i$ is $(1 - \beta, 1 - \alpha)$-reliable for $\neg\varphi$ in N.

Conversely, suppose that $\mathtt{A}_i$ is $(1 - \beta, 1 - \alpha)$-reliable for $\neg\varphi$ in N, and consider the reliability of $\mathtt{A}_i$ with respect to φ. If $(N, s, v) \models \varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{“No”}\}) \quad \text{by } \varphi\text{-completeness} \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad \geq 1 - (1 - \alpha) \\
&\quad = \alpha.
\end{aligned}$$

Similarly, if $(N, s, v) \models \neg\varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, L_i(s), s, v'_i) = \text{“No”}\}) \quad \text{by } \varphi\text{-completeness} \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, L_i(s), s, v'_i) = \text{“Yes”}\}) \\
&\quad \leq 1 - (1 - \beta) \\
&\quad = \beta.
\end{aligned}$$

Thus, $\mathtt{A}_i$ is (α, β)-reliable for φ in N. ∎

To prove Proposition 5.5, we need a preliminary lemma, alluded to in the text.

Lemma A.3: *If N is a probabilistic algorithmic knowledge structure where agent i uses a knowledge algorithm $\mathtt{A}_i$ that is φ-complete and that respects negation, then the formula $X_i\varphi \Leftrightarrow \neg X_i\neg\varphi$ is valid in N.*

Proof: Let s be a state of N and let v be a derandomizer. If $(N, s, v) \models X_i\varphi$, then $\mathtt{A}_i^d(\varphi, L_i(s), s, v_i) =$ "Yes". Since $\mathtt{A}_i$ respects negation and is φ-complete, this implies that $\mathtt{A}_i^d(\neg\varphi, L_i(s), s, v_i) =$ "No" (the algorithm cannot return "?" by φ-completeness) and hence that $(N, s, v) \not\models X_i\neg\varphi$, so $(N, s, v) \models \neg X_i\neg\varphi$. Thus, $(N, s, v) \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. Since s and v were arbitrary, we have that $N \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. Conversely, let s be a state of N and let v be a derandomizer. If $(N, s, v) \models \neg X_i\neg\varphi$, then $(N, s, v) \not\models X_i\neg\varphi$, that is, $\mathtt{A}_i^d(\neg\varphi, L_i(s), s, v_i) \neq$ "Yes". Since A_i is φ-complete and respects negation, A_i is $\neg\varphi$-complete, so it must be the case that $\mathtt{A}_i^d(\neg\varphi, L_i(s), s, v_i) =$ "No". Therefore, $\mathtt{A}_i^d(\varphi, L_i(s), s, v_i) =$ "Yes", and $(N, s, v) \models X_i\varphi$. Since s and v were arbitrary, we have that $N \models X_i\varphi$. ∎

Proposition 5.5: *If $\mathtt{A}_i$ respects negation, is φ-complete, and is (α, β)-reliable for φ in N, then*

(a) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta} \wedge \overline{\mathrm{Ev}}_i(\neg\varphi) \leq \frac{\beta}{\alpha+\beta}\right)$ *if* $(\alpha, \beta) \neq (0, 0)$*;*

(b) $N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\varphi) = 1 \wedge \overline{\mathrm{Ev}}_i(\neg\varphi) = 0\right)$ *if* $(\alpha, \beta) = (0, 0)$*;*

(c) $N \models X_i\neg\varphi \wedge K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\neg\varphi) \geq \frac{1-\beta}{2-(\alpha+\beta)} \wedge \overline{\mathrm{Ev}}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}\right)$ *if* $(\alpha, \beta) \neq (1, 1)$*;*

(d) $N \models X_i\neg\varphi \wedge K_i^{\pm}\varphi \Rightarrow \left(\underline{\mathrm{Ev}}_i(\neg\varphi) \geq \frac{1}{2} \wedge \overline{\mathrm{Ev}}_i(\varphi) \leq \frac{1}{2}\right)$ *if* $(\alpha, \beta) = (1, 1)$.

Proof: Suppose that $\mathtt{A}_i$ is (α, β)-reliable for φ in N. Since $\mathtt{A}_i$ is φ-complete and respects negation, by Proposition 5.4, $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N. For part (a), suppose that $(\alpha, \beta) \neq (0, 0)$. Let s be a state of N and let v be a derandomizer. By Proposition 5.3 applied to φ, $(N, s, v) \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta)$. By Lemma A.3, $(N, s, v) \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. By Proposition 5.3 applied to $\neg\varphi$, $(N, s, v) \models \neg X_i\neg\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \mathrm{Ev}_i(\neg\varphi) \leq (1-(1-\beta))/(1-(1-\alpha)+1-(1-\beta))$, that is, $(N, s, v) \models \neg X_i\neg\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \mathrm{Ev}_i(\neg\varphi) \leq \beta/(\alpha+\beta)$. Putting this together, we get

$$(N, s, v) \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta) \wedge \mathrm{Ev}_i(\neg\varphi) \leq \beta/(\alpha+\beta))\,.$$

Since s and v are arbitrary,

$$N \models X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta) \wedge \mathrm{Ev}_i(\neg\varphi) \leq \beta/(\alpha+\beta))\,.$$

For part (c), suppose that $(\alpha, \beta) \neq (1, 1)$. By Proposition 5.3 applied to $\neg\varphi$, $(N, s, v) \models X_i\neg\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq (1-\beta)/(2-(\alpha+\beta))$. By Lemma A.3, $(N, s, v) \models X_i\neg\varphi \Rightarrow \neg X_i\varphi$. By Proposition 5.3 applied to φ, $(N, s, v) \models \neg X_i\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq (1-\alpha)/(2-(\alpha+\beta))$. Putting this together, we get

$$(N, s, v) \models X_i\neg\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow (\mathrm{Ev}_i(\neg\varphi) \geq (1-\beta)/(2-(\alpha+\beta)) \wedge \mathrm{Ev}_i(\varphi) \leq (1-\alpha/(2-(\alpha+\beta)))\,.$$

Since s and v are arbitrary,

$$N \models X_i\neg\varphi \wedge \neg K_i^{\pm}\varphi \Rightarrow (\mathrm{Ev}_i(\neg\varphi) \geq (1-\beta)/(2-(\alpha+\beta)) \wedge \mathrm{Ev}_i(\varphi) \leq (1-\alpha/(2-(\alpha+\beta)))\,.$$

Again, we leave the proof of parts (b) and (d) to the reader. ∎

References

Berman, P., J. Garay, and K. J. Perry (1989). Towards optimal distributed consensus. In *Proc. 30th IEEE Symposium on the Foundations of Computer Science (FOCS'89)*, pp. 410–415.

Dolev, D. and A. C. Yao (1983). On the security of public key protocols. *IEEE Transactions on Information Theory 29*(2), 198–208.

Fagin, R. and J. Y. Halpern (1988). Belief, awareness, and limited reasoning. *Artificial Intelligence 34*, 39–76.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1995). *Reasoning about Knowledge*. MIT Press.

Halpern, J. Y. (2003). *Reasoning About Uncertainty*. MIT Press.

Halpern, J. Y. and R. Fagin (1992). Two views of belief: belief as generalized probability and belief as evidence. *Artificial Intelligence 54*, 275–317.

Halpern, J. Y., Y. Moses, and M. Y. Vardi (1994). Algorithmic knowledge. In *Proc. 5th Conference on Theoretical Aspects of Reasoning about Knowledge (TARK'94)*, pp. 255–266. Morgan Kaufmann.

Halpern, J. Y. and R. Pucella (2002). Modeling adversaries in a logic for reasoning about security protocols. In *Proc. Workshop on Formal Aspects of Security (FASec'02)*, Volume 2629 of *Lecture Notes in Computer Science*, pp. 115–132.

Halpern, J. Y. and R. Pucella (2003). A logic for reasoning about evidence. In *Proc. 19th Conference on Uncertainty in Artificial Intelligence (UAI'03)*, pp. 297–304.

Hintikka, J. (1962). *Knowledge and Belief.* Cornell University Press.

Hintikka, J. (1975). Impossible possible worlds vindicated. *Journal of Philosophical Logic 4*, 475–484.

Kripke, S. (1963). A semantical analysis of modal logic I: normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9*, 67–96.

Kyburg, Jr., H. E. (1983). Recent work in inductive logic. In T. Machan and K. Lucey (Eds.), *Recent Work in Philosophy*, pp. 87–150. Rowman & Allanheld.

Motwani, R. and P. Raghavan (1995). *Randomized Algorithms*. Cambridge University Press.

Paulson, L. C. (1998). The inductive approach to verifying cryptographic protocols. *Journal of Computer Security 6*(1/2), 85–128.

Rabin, M. O. (1980). Probabilistic algorithm for testing primality. *Journal of Number Theory 12*, 128–138.

Ramanujam, R. (1999). View-based explicit knowledge. *Annals of Pure and Applied Logic 96*(1–3), 343–368.

Rivest, R. L., A. Shamir, and L. Adelman (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*(2), 120–126.

Shafer, G. (1982). Belief functions and parametric models (with commentary). *Journal of the Royal Statistical Society, Series B 44*, 322–352.

Walley, P. (1987). Belief function representations of statistical evidence. *Annals of Statistics 18*(4), 1439–1465.