Electronic Colloquium on Computational Complexity, Report No. 46 (2008)

A Computational Theory of Awareness and Decision Making

Nikhil R Devanur
Toyota Technological Institute–Chicago

Lance Fortnow
Northwestern University

Abstract

We exhibit a new computational-based definition of awareness, informally that our level of unawareness of an object is the amount of time needed to generate that object within a certain environment. We give several examples to show this notion matches our intuition in scenarios where one organizes, accesses and transfers information. We also give a formal process-independent definition of awareness based on Levin's universal enumeration.

We show the usefulness of computational awareness by showing how it relates to decision making, and how others can manipulate our decision making with appropriate advertising, in particular, connections to sponsored search and brand awareness. Understanding awareness can also help rate the effectiveness of various user interfaces designed to access information.

1 Introduction

A beer company advertises to raise their brand awareness. Search engine companies do not only want to give the world access to information, but to let people be highly aware of the information when they need it. One typically chooses a restaurant to eat at among those with which one has a high awareness of. We keep an address book so we can be aware of a phone number when we need to call that person. A judge tries to determine what circumstances that a legislature was aware of when they passed a certain law. We used to be highly aware of John Edwards when he was an active candidate but our awareness of him has since faded.

We shine the "computational lens" on awareness to develop a new definition that captures our intuition in a number of scenarios such as the ones above. Awareness does not occur in a vacuum so we consider two types of inputs. The first is the environment which encodes all information sources such as a person's memory, possible interactions with other people and nature, books, the entire Internet and everything else one might have access to. The other is a context such as "Restaurants in Chicago." Informally we define the unawareness level of an object as *the amount of time needed to enumerate that object* where the enumerator gets the context as input and has random access to the environment. The intuition is that objects that you are more aware of you will generate first. We give a formal definition in Section 4. Using a universal enumeration procedure due to Levin [Lev73], our formal definition is independent of the particular enumeration procedure of the agent. There have been some previous notions of awareness in the literature (see Section 3 for a more detailed discussion), particularly a knowledge-based concept due to Fagin and Halpern [FH87]. Our model has the advantage of talking about abstract objects instead of formulas with truth values and allowing for a gradation of awareness that can possible decrease over time.

ISSN 1433-8092

We then show how one can use our definition to understand decision making processes. How do you decide on what restaurant to eat at? A large city typically has thousands of restaurants. You would consider restaurants where you have eaten before, recommendations from friends, reviews from newspapers, restaurant guides and search engines, some restaurant you might have walked by or seen an advertisement for. What you don't do is examine every restaurant in the city and making the choice that optimizes the various criteria, such as type and quality of food, cost, atmosphere, location, etc. You simply lack awareness of all the restaurants. That's not quite precisely true, since you can find nearly every restaurant with appropriate web surfing. In this paper we measure a certain cost of awareness, precisely a *computational* cost of awareness. In the restaurant example, you would likely choose restaurants with high awareness, balanced against your particular desires in food type, etc. at the time.

A typical decision making process goes like this:

The centerpiece is an *agent* who is the one making the decision. A process starts when the agent gets a *criteria*, who then interacts with the environment, and outputs a *decision* that satisfies the given criteria. For instance, a criteria could be, "find a used car that has less than 50,000 miles, costs less than $8,000 and within 50 miles of Chicago". The interaction with the environment could involve going through classifieds to look at used cars for sale, placing an advertisement asking for used cars which meet the criteria, and so on. At some point, the agent decides to stop the interaction and outputs a decision, which could also be "there are no such cars". Note that cars with low awareness will be enumerated earlier and thus be more likely bought by the agent.

We can use awareness to capture how the decision making process can be manipulated by someone who benefits from the decision made. For example, a used car dealer might want to make sure that the decision turns out to be one of his cars. Awareness also help us understand the cost to computation which factors into the decision making process.

These definitions of awareness and decision making turn out to be quite useful: we show how they can be applied in a number of scenarios including sponsored search, competition between brands (focusing on awareness of the brands themselves as well as the awareness of particular attributes that would make one brand look better than the other) and in analyzing user interfaces.

We end with a discussion of future applications and questions about awareness and decision making.

2 Examples of Computational Awareness

In this section we motivate our definition by giving examples where our definition coincides with the intuitive notion of awareness.

2.1 Decreasing Awareness

Consider a stack of research papers that you want to read. You download and print a new paper by Karp and put it on top of the stack. The Karp paper has high awareness as it is very easy to enumerate that paper, picking it off the top of the stack. But over time you will put more papers on the stack and the Karp paper now falls towards the middle of the stack. Your awareness of that

paper goes down as enumerating papers would take longer before the Karp paper is enumerated. As the environment changes over time, you can become less aware of a given object.

Awareness can be recovered. You might see a talk mentioning the Karp paper. In the context of the talk, your awareness of the paper becomes high again. This may cause you to put the Karp paper once again at the top of the stack, in effect changing your environment to increase the awareness of the paper.

2.2 Awareness of Unawareness

In the context of US Presidents, you are likely aware of recent presidents like George W. Bush and Bill Clinton, early presidents like George Washington and Thomas Jefferson and famous presidents like Abraham Lincoln and Franklin D. Roosevelt. But you likely have far less awareness of say James Garfield. Until you have read the last sentence. As a basic principle you cannot be aware of what you are unaware of.

2.3 Awareness and the Legal System

Consider a civil court case where the judge orders the defendant to hand over a specific document to the plaintiffs. A common tactic is to hand over a large stack of documents with the requested document placed in a random location. The defendants have followed the letter of the law by physically giving the document to the plaintiffs. But even though the plaintiffs have the document in this stack, they are not aware of the document because it would take a very long time to find the document given the environment of the large stack produced by the defense.

With a proper definition of awareness, the judge could order the defense to not only make the document available but produce an environment where the plaintiffs have high awareness of the document in the context of the court case. If the defense tried the above tactic they would be found in contempt of court unless they could exhibit a simple enumeration procedure that quickly produces the document.

In a different legal context, Chung and Fortnow [CF08], examined why loopholes occur in laws and contracts. A legislature creates a law and at some point in the future the judge interprets the law as it applies to a particular circumstance. Think of the judge as a function mapping a law and circumstance into a penalty. The goal of the judge is to apply the legislature's intent of the law.

This may not always be possible. A legislature, when they create a law, may not be aware of all future circumstances. Furthermore, a judge may not be aware of what the legislature is or is not aware of. Chung and Fortnow created a simple model and showed that legislatures, sometimes knowingly, create loopholes to keep laws interpreted as best as possible in the future. The model of awareness used by Chung and Fortnow was a very simple cost function. A better notion of awareness was needed to create better models and find other applications of awareness in the legal community. This goal of a new awareness model is one of the drivers for this current paper.

2.4 Future Awareness

Why do we keep calendars? After all we only add entries to a calendar that we already know. The key is awareness. When we add the FOCS conference to our calendar we do so that in the context of October 26, 27 or 28, we will quickly enumerate the event. This is useful not only during the event themselves but if we later schedule other events on those days, our context switches to those days and we have a high awareness of FOCS, allowing us to avoid conflicts. In short, keeping a calendar creates an environment to help us to have high awareness in the contexts where we need high awareness.

We can tell similar stories about address books, email programs, to do lists, file directories, desktop search, file cabinets, etc. Most organizational tools try to organize our information to maximize the awareness of information in the future contexts where we need it. One must apply these organizational tools carefully. Putting too many events on one day (say including sports teams schedules, theater events, colloquium talks all over campus, etc.) causes unawareness of some of the possibly important events schedule that day as it could take a long time to enumerate those events even in the context of that day. Organizational tools need the right balance to maximize awareness in the right context and using a notion of computational awareness can help us use those tools to maximize our efficiency.

3 Relationship to Other Notions of Awareness

Awareness as a concept has had extensive study in many disciplines including psychology, philosophy and economics. We cannot, in the limited space of this paper, even begin to cover these approaches. Instead we focus on a computer science approach based on knowledge representation.

Fagin and Halpern [FH87] build awareness on top of the Kripke model of knowledge (see Halpern and Moses [HM84]. In the Kripke model, the world is in one of many states, for each person i there is a partition of the states into information sets. Person i knows what set his states lies in but not the state itself. Person i knows a formula ϕ ($K_i\phi$) if the formula ϕ is true in every state in the partition. Fagin and Halpern add to this two new modal operators $A_i\phi$ and $X_i\phi$. $A_i\phi$ represents some intuitive notion of awareness and define explicit knowledge, $X_i\phi$, as having implicit knowledge of ϕ, ($K_i\phi$) and being aware of ϕ. Fagin and Halpern do not put any restrictions on $A_i\phi$.

Two economists Mudica and Rusticini [MR94] suggest defining knowledge in terms of awareness: You are aware of ϕ if you know ϕ or you know you don't know ϕ. In the Kripke model, this definition is too broad, as for every formula ϕ you either know ϕ or you know you don't know ϕ. In a follow-up paper Mudica and Rusticini [MR99] give a more nuanced definition of awareness which Halpern [Hal01] later showed, under reasonable assumptions, is equivalent to saying you are aware of ϕ if you explicitly know ϕ or you explicitly know you don't explicitly know ϕ.

In that last paper [Hal01], Halpern suggests looking at a computational version of awareness:

> It is possible to consider more computationally oriented notions of awareness. The problem is then to come up with interesting notions of awareness that have enough structure to allow for interesting mathematical analysis. I believe it should also be

possible to use awareness structures to allow for natural reasoning about awareness and lack of it (so that an agent can reason, for example, about the possibility that she is unaware of certain features that another may be aware of). I am currently working on modeling such reasoning.

Halpern [Hal] has not yet fully pursued this direction and in any case has focused on the complexity of determining whether a formula is true or false.

While the model of awareness developed above has some appeal, we don't believe it properly models the intuitive ideas of awareness we express in this paper. Our model has many features that differ, including

- The Halpern-Fagin model focuses on formulas that have some truth value. We focus on awareness of strings, like "McDonalds" without any underlying truth value.
- Instead of a binary notion of awareness, we give a gradation of awareness, so one can say that they are more aware of a restaurant they ate at yesterday than one they ate at a few years ago. We also allow awareness to possibly decrease over time.
- We give a full formal definition of awareness based on a computational basis, as opposed to just an axiomatic definition that allows for many different awareness operators.

4 Formal Model

In this section we develop a formal model of awareness.

We fix an alphabet $\Sigma = \{0, 1\}$ though the theory easily generalizes to larger alphabets. We define the environment formally as a function $E : \Sigma^* \to \Sigma^*$. This is general enough to encode any of the intuitive notions of the environment we describe above. The context y is just a string in Σ^*. We define awareness (actually unawareness) on strings as well.

An enumeration process is an oracle Turing machine M such that given an oracle A and input w, $M^A(w)$ will output a potentially infinite series of strings $z_1, z_2, \ldots$. The oracle A can be a function whose output will appear on a special oracle output tape. We will count each oracle query as a single time step, though it will take more time to write each bit of the query and read each bit of the response.

We define the unawareness of a string x in environment E and context y using enumeration process M, $U_M^E(x|y)$ as the amount of time until $M^E(y)$ enumerates the string x. Note we are counting time, not the index of the string being enumerated. Also we do not require that $M^E(y)$ halt after producing string x on its list of outputs. If $M^E(y)$ never enumerates x we say the unawareness is infinite.

We can eliminate the dependence on the enumeration process by applying a universal enumeration procedure due to Levin [Lev73]. Levin shows that there exists a single enumeration procedure N such that for every enumeration procedure M, constant c_M such that for all x, context y and environment E,

$$U_N^E(x|y) \leq c_M U_M^E(x|y).$$

The value c_M does not depend on x, y or E. As a shortcut we use $U^E(x|y)$ to represent $U^E_N(x|y)$, the enumeration-independent unawareness measure of object x in context y and environment E.

Levin [Lev73] also gives a Kolmogorov-complexity definition equivalent up to constant factors: Fix a universal oracle Turing machine M_U. $U^E(x|y)$ is the minimum over all programs p of the quantity $t2^{|p|}$ where $M^E_U(p, y)$ outputs x within t steps.

The enumeration-independent definition has nice mathematical properties but can be difficult to properly analyze. Often in this paper we will often focus on limited though natural enumeration processes in a specifically structured environment.

This definition does not involve any actual actions or decisions made by an agent. One approach involves combining awareness with a fitness function. Some candidate functions include

- The input is a function $f : \Sigma^* \to \{0, 1\}$, measuring the "fitness" of a string. This models binary decision processes, where a string is either fit or not, and the goal is to find some string that is.
- The input is a function $F : \Sigma^* \times \Omega \to \{0, 1\}$, where Ω is a probability space. In this case, F is interpreted as a random variable that measures fitness. A further special case of this is that the fitnesses of the strings are independent of each other. In this case, we could simply have $f : \Sigma^* \to [0, 1]$, which measures the probability that a string satisfies the criteria. Such random functions allow us to model multiple agents, with a single process.
- The input is a function $f : \Sigma^* \to \mathbb{R}$. In this case fitness is a continuous measure, and the agent seeks to find a string that maximizes the fitness.

Each of the above cases can be extended to include auxiliary information, so that the domain of f is $\Sigma^* \times \Sigma^*$. The auxiliary information could involve such things as product reviews.

One can then get a decision making process by combining the fitness function with an enumeration procedure, for example, in the first case above, choosing an object x if it is the first object enumerated by $N^E(y)$ such that $f(x) = 1$. We can also explicitly incorporate the cost of computation into the decision making process, such an approach is shown in Appendix A.

One can manipulate the decision making process by manipulating the environment E, by changing the output of the function E on some inputs (for example manipulating search engine results). Let A be some set of actions (a subset of all possible actions), and $c(A)$ be the cost of changing all the entries of E corresponding to A. In other words, the cost of advertising on the actions in A is $c(A)$. A special case is when $c(A) = \sum_{a \in A} c(a)$. Also an advertiser w makes a profit $p_w(x)$ if x is the decision made. Hence it is profitable for an advertiser to advertise on actions in A if the resulting change in the decision made gets him an increase in profit more than $c(A)$.

5 Applications

In this section, we show the usefulness of our model by illustrating how it lets us formally analyze various scenarios. We make several simplifying assumptions to keep the analysis tractable. Analyzing more general models might lead to deeper insights.

5.1 Sponsored Search Auctions

We now use our model to analyze sponsored search: a user queries a search engine for a keyword, and is served a list of advertisements (ads) along with the "organic" results for the query. The list of ads is referred to as sponsored search listings. The goal of an ad is to increase awareness of say, some product and influence the decision of users who might be interested in that product. The advertiser pays the search engine a certain amount if the user clicks on his ad.

The following is a formalization of the above in our model. Define the context by what ads the user finds[1] relevant, and clicks on, if presented by itself. For each advertiser i, let $F_i \in \{0, 1\}$ be such that $F_i = 1$ if the ad is relevant, and 0 otherwise. Since we are more interested in a user drawn at random from a large population of users, rather than any particular user, we treat the F_i's as random variables.

Choosing the enumeration process is a non-trivial decision because it amounts to modeling the user behavior. One way to measure the accuracy of a process is to see how well it fits available data. This approach is similar to statistical estimation, but the difference here is that the focus is on an *algorithm* that generates the clicks. Moreover, the algorithmic reasonableness of a process can be used as another measure, although this is a subjective quantity.

A quantity of great interest is the click-through rate (CTR) of an ad, which is the ratio of the number of clicks on an ad to the number of times the ad is displayed. Clearly the CTR of an ad depends on a lot of contextual factors, like the relevance of the ad to the keyword queried, the text displayed by the ad, and position of all the ads. A common assumption is that the CTR is a product of an ad-dependent factor (which is equivalent to the fitness f_i) and a slot dependent factor, θ_j for slot j (see Varian [Var07], for example). One could reverse-engineer this assumption to see what processes generate such clicks. One such process is, the user samples a slot j with probability θ_j and clicks on the ad in that slot if relevant, which happens with probability f_i. Otherwise he does not click anything.

A simpler and more reasonable enumeration process is the following: The user goes down the list of ads starting from the top to bottom, and clicks on the first ad that is relevant. If ad i is shown in slot $\pi(i)$, then the CTR of the ad is

$$\Pr[F_j = 0 \ \ \forall \ \ j : \pi(j) < \pi(i), \text{ and } F_i = 1].$$

A simplifying assumption is that the F_i's are independent random variables. Experiment results by Crasswell et. al.[CZTR08] indicate that this process gives a better fit to the data than the standard one. Athey and Ellison [AE07] have developed and analyzed a different model based on a similar enumeration principle.

Given an enumeration process such as this one, we can analyze various aspects of sponsored search. Typically the slots are priced through an auction. Two of the standard auctions are the Vickery-Clarke-Groves (VCG) auction and the Generalized Second Price (GSP) auction. Various questions related to these auctions are:

1. The VCG auction allocates the slots to maximize social welfare. What is the order of the ads that achieves this maximum?

[1] In reality, the user *thinks* that the ad is relevant and clicks on it. If he discovers that it is not, then he may come back and click on more ads. To start with, we consider a simple model.

2. What are VCG prices?

3. What is the equilibrium of the GSP auction?

The answers to some of these questions provide surprising insight. For instance, for the first question, suppose advertiser i derives a value v_i from one click on his ad. The social welfare of a particular order of listings is the expected value of a click from one user. The answer is that the ads are to be ranked by their valus v_i regardless of their fitnesses f_i (Theorem 1). This is in contrast to the standard model in which ranking by $f_i v_i$ is optimal.

Remark: The conventional wisdom is that ranking by expected revenue ($f_i v_i$) is better. However, in the analysis, we have assumed that we can display all the ads. When the number of ads that can be displayed is restricted, the set of ads should be chosen in order to maximize welfare, and this set of ads should be ranked by v_i's. This makes sure that ads with very small f_i's are not displayed, which was the original motivation behind ranking by revenue.

Theorem 1. *Given a set of ads, their values and fitnesses, ordering by decreasing values of v_i maximizes*

$$W(\pi) := \sum_i v_i f_i \prod_{j:\pi(j)<\pi(i)} (1 - f_j)$$

among all permutations π.

Proof. Consider any permutation π. We will show that if there exist i and j such that $\pi(i) < \pi(j)$ and $v_i < v_j$, then switching the positions of i and j increases W. Without loss of generality, we may assume that π is the identity permutation and $j = i + 1$. With these assumptions, the only terms in W that change when i and $i + 1$ are switched are the ones with v_i and v_{i+1}. They both have as a common factor $\prod_{j:\pi(j)<\pi(i)}(1 - f_j)$ which we can ignore for the sake of comparison. Hence the terms of interest are initially

$$v_i f_i + v_{i+1} f_{i+1}(1 - f_i) = v_i f_i + v_{i+1} f_{i+1} - v_{i+1} f_i f_{i+1},$$

which is easily seen to be smaller than the terms after switching,

$$v_{i+1} f_{i+1} + v_i f_i(1 - f_{i+1}) = v_i f_i + v_{i+1} f_{i+1} - v_i f_i f_{i+1}.$$

□

5.2 Two competing brands

In this section, we analyze a slightly different situation, where the choice is between two competing entities that the agent is already aware of. The decision is based on the many features of each entity and in particular, his awareness of the features. The entities choose which features to highlight in order to get selected. For example, the entities could be presidential candidates, and the features are their stand on various issues. Or the entities could be two competing brands, say Microsoft and Apple, and the features are usability, security, available applications for the operating system and so on.

We now formalize the above situation. The context is given by an initial bias plus weights for each feature. These are just vectors of real values. The enumeration process picks some k

features and the decision output is argmax of the sum of weights of the features picked plus the initial bias. We don't say how the k features are picked, since that could vary from situation to situation. For example, the process interacts with the environment at random for a certain time and picks k most repeated features. Or it interacts with the environment for k rounds and picks the ones returned. The initial bias can be used to include all the factors that are not susceptible to be changed.

Suppose that the enumeration process makes n queries to the oracle, which returns a feature for each query. The process picks k most repeated features. Let the advertising budgets of the two entities be such that they can pick the features returned on n_1 and n_2 queries each. ($n_1+n_2 = n$.) Rank the features by decreasing order of difference between the weights and let $i_1, i_2, \ldots, i_k$ be the top k features, and $j_1, j_2, \ldots, j_k$ be the bottom k. The i's are the best features of entity 1 and the j's are those of 2. Let's say the k most repeated features are $i_1, i_2, \ldots, i_{k_1}$ and $j_1, j_2, \ldots, j_{k_2}$. Let w_l be the difference in the weights for feature l (entity 1 minus entity 2) and $v_l = -w_l$. Then 1 is chosen if

$$\sum_{l=i_1}^{i_{k_1}} w_l > \sum_{l=j_1}^{j_{k_2}} v_l,$$

and 2 is chosen otherwise. This is a min-max game between the 2 entities where their strategies are to decide which features to return on the queries they control. The strategy set of entity 1 is the set of all partitions of n_1 into $\leq k$ parts (and respectively for 2).

Let us consider simple cases. If $k = 1$, then the entity who controls the larger number of queries wins. If $k = 2$, then 1 wins if $w_{i_1} \geq v_{j_1}$ and $n_1 \geq n_2/2$, or if $n_1 \geq 2n_2$. In general, let k_1 and k_2 be such that

$$\sum_{l=i_1}^{i_{k_1}} w_l > \sum_{l=j_1}^{j_{k_2-1}} v_l,$$

$$\sum_{l=i_1}^{i_{k_1-1}} w_l < \sum_{l=j_1}^{j_{k_2}} v_l$$

and $k_1 + k_2 = k + 1$. That is, entity 1 is guaranteed to win if he can control at least k_1 of the features picked (and respectively for entity 2). So the best strategy for 1 is to spread his top k_1 features among the n_1 queries that he controls. Thus entity 1 wins if $\lfloor \frac{n_1}{k_1} \rfloor > \lfloor \frac{n_2}{k_2} \rfloor$ and vice versa. If they are the same, then let's say there is a tie.

There are several ways to refine the model to make it more realistic, for instance, k could be a random variable with some known distribution, or the costs of advertising could be different for different queries. We do not pursue it further, as our goal is to simply show how interesting analysis emerges from our model.

5.3 User Interface Design

In this section, we give a principled way to evaluate user interfaces using our model. This is different from the preceding examples in that here, the task is to design the environment (the interface is part of the environment) to increase awareness of the right things at the right time. Examples are designing a calendar application, or file system design.

The process of designing and evaluation of interfaces can be broken down into the following steps.

1. Define all contexts the interface is intended for.
2. Define the enumeration process for each context, including interaction with the interface.
3. Evaluate the (average?) complexity of the enumeration process.

We consider a simple example here to illustrate the method. Consider two contexts, one in which a user wants to start an application and the other in which he wants to switch between applications. We assume that he only uses a mouse (no keyboard) and that the applications in question are frequently used. We consider a cost for clicking on an icon and for searching the icon. We further distinguish between linear and random access search. Linear search refers to searching for an icon that could be anywhere in a list of icons. Random access search refers to searching an icon from a list that is always at the same place.

We consider two approaches followed by the two major operating systems[2]. In the Windows approach, the process for starting an application is, click on the start menu, search the list and click. The process for switching is search the task bar and click. In the Mac approach, the process is same for both starting and searching, search from the dock and click. The complexity for windows is 2 clicks and 1 linear search for start; 1 click and 1 linear search for switch. The complexity for Mac is, 1 click and 1 random access search for both start and switch. Further considering the complexity of the two types of search gives a better comparison between the two approaches, but is out of scope for this paper.

6 Conclusions

In this paper, we have given a new computational-based definition of awareness that appears to fit well with many intuitive uses of the concept. We apply our notion to give a new way to think about decision making and show applications to sponsored search, brand awareness and user interfaces.

We have only scratched the surface with our uses of awareness. Deeper analysis of the models described in Section 5 should lead to a greater understanding of those topics. We also hope to see several new and unexpected applications of awareness as well as other ways we can use computational thinking to understand the various ways we process information.

Acknowledgments

The second author would like to thank Kim-Sau Chung. Their paper [CF08] and subsequent discussions exhibited the need for a new computational-based definition of awareness that eventually led to the model described here. The second author also would like to acknowledge useful

[2]We make several simplifying assumptions to make the analysis easier, and the conclusions are not supposed to be a judgement on the actual operating systems.

discussions with Rakesh Vohra and Nabil Al-Najjar. Jason Hartline pointed us to the sponsored search work of Athey and Ellison [AE07] and Abie Flaxman pointed us to the experiments of Craswell, et. al. [CZTR08].

References

[AE07] S. Athey and G. Ellison. Position auctions with consumer search. Working paper, 2007.

[CF08] K. Chung and L. Fortnow. Loopholes. Submitted, 2008.

[CZTR08] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *WSDM*, 2008.

[FH87] R. Fagin and J. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, December 1987.

[Hal] J. Halpern. Personal Communication.

[Hal01] J. Halpern. Alternative semantics for unawareness. *Games and Economic Behavior*, 37(2):321–339, November 2001.

[HM84] J. Halpern and Y. Moses. *Knowledge and common knowledge in a distributed environment.* ACM Press, New York, 1984.

[Lev73] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems: in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973. Corrected English translation in [Tra84].

[MR94] S. Modica and A. Rustichini. Awareness and partitional information structures. *Theory and Decision*, 37(1):107–124, July 1994.

[MR99] S. Modica and A. Rustichini. Unawareness and partitional information structures. *Games and Economic Behavior*, 27(2):265–298, May 1999.

[Tra84] R. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Var07] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

A Cost of Awareness

In this section we show how to explicitly incorporate the computational cost of awareness. We consider a continuous measure of fitness, $f : \Sigma^* \to \mathbb{R}$. We introduce a cost function $g : \mathcal{N} \to \mathbb{R}$ that measures the cost incurred by t time steps. This captures the fact that the cost may scale with time. Finally, suppose that at every time step, the process has a *belief* about the outcome of the process if it was continued further. The belief is a probability distribution over all possible outcomes of the process. What we need about this belief is that at time t, the process can calculate

$$B(t) := \max_{t'>t} \mathbf{E}[f(x^*(t'))] - g(t'), \quad \text{where}$$

$$x^*(t) = \arg\max\{f(x) : U^E(x|y) \leq t\}.$$

Given this, we can give an explicit stopping criteria for the process: stop when

$$B(t) \leq f(x^*(t)) - g(t)$$

and output $x^*(t)$. This gives us a clean way of balancing the potential increased benefit from continuing the search with the expected cost.

ECCC ISSN 1433-8092

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