Reflective Oracles: A Foundation for Classical Game Theory (Draft)

Jessica Taylor
jessica@intelligence.org

Benja Fallenstein
benja@intelligence.org

ABSTRACT

We introduce a formalism in which programs can predict the output of other programs and show that it can be used to define rational agents using causal decision theory. We also show that agents using causal decision theory in this framework will play a Nash equilibrium.

1. INTRODUCTION

Classical decision theory and game theory are founded on the notion of a perfect Bayesian reasoner. Such an agent may be uncertain which of several possible worlds describes the state of its environment, but given any particular possible world, it is able to deduce exactly what outcome each of its available actions will produce.

This assumption is, of course, unrealistic: Agents in the real world must necessarily be boundedly rational reasoners, which make decisions with finite computational resources. Nevertheless, the notion of a perfect Bayesian reasoner provides an analytically tractable first approximation to the behavior of real-world agents, and underlies a huge body of work in economics, computer science, and other fields.

On closer examination, however, the assumption that agents can compute what outcome each of their actions leads to in every possible world is troublesome even if we assume that agents have unbounded computing power. For example, consider the game of *Matching Pennies*, in which two players each choose between two actions ("heads" and "tails"); if the players choose the same action, the first player wins a dollar, if they choose differently, the second player wins. Suppose further that both players' decision-making processes are Turing machines with unlimited computing power. Finally, suppose that both players know the exact state of the universe at the time they begin deliberating about what action to choose, including the source code of their opponent's decision-making algorithm.[1]

In this set-up, by assumption, both agents know exactly which possible world they are in. Suppose that they are able to use this information to accurately predict their opponent's behavior. Since both players' decision-making processes are deterministic Turing machines, their behavior is deterministic given the initial state of the world; each player either definitely plays "heads" or definitely plays "tails". But neither of these possibilities is consistent: For example, if the first player chooses heads and the second player can predict this, the second player will choose tails, but if the first player can predict this in turn, it will choose tails, contradicting the assumption that it chooses heads.

The problem is caused by the assumption that given its opponent's source code, a player can figure out what action the opponent will choose. One might think that it could simply run its opponent's source code, but if the opponent does the same, both programs will go into an infinite loop. Even giving the players access to a halting oracle does not help, because even though a machine with access to a halting oracle can predict the behavior of an ordinary Turing machine, it cannot in general predict the behavior of another oracle machine.

Classical game theory resolves this problem by allowing players to choose *mixed* strategies (probability distributions over actions); for example, the unique Nash equilibrium of Matching Pennies is for each player to assign "heads" and "tails" probability 0.5 each. However, instead of treating players' decision-making algorithms as computable processes which are an ordinary part of a world with computable laws of physics, classical game theory treats players as special objects. For example, to describe a problem in game-theoretic terms, we must provide an explicit list of all relevant players, even though in the real world, "players" are ordinary physical objects, not fundamentally distinct from objects such as rocks or clouds.

In this paper, we show that it is possible to define a certain kind of *probabilistic* oracle—that is, an oracle which answers some queries non-deterministically—such that a Turing machine with access to this oracle can perform perfect Bayesian reasoning about environments that can themselves be described as oracle machines with access to the same oracle. This makes it possible for players to treat opponents simply as an ordinary part of this environment.

When an environment contains multiple agents playing a game against each other, the probabilistic behavior of the oracle may cause the players' behavior to be probabilistic as well. We show that in this case, the players will always

[1]The technique of quining (Kleene's second recursion theorem) shows that it is possible to write two programs that have access to each other's source code.

play a Nash equilibrium, and for every particular Nash equilibrium there is an oracle that causes the players to behave according to this equilibrium. In this sense, our work can be seen as providing a foundation for classical game theory, demonstrating that the special treatment of players in the classical theory is not fundamental.

The oracles we consider are not halting oracles; instead, roughly speaking, they allow oracle machines with access to such an oracle to determine the probability distribution of outputs of other machines with access to the same oracle. Because of their ability to deal with self-reference, we refer to these oracles as *reflective oracles*.

2. REFLECTIVE ORACLES

In many situations, programs would like to predict the output of other programs. They could simulate the other program in order to do this. However, this method fails when there are cycles (e.g. program A is concerned with the output of program B which is concerned with the output of program A). Furthermore, if a procedure to determine the output of another program existed, then it would be possible to construct a liar's paradox of the form "if I return 1, then return 0, otherwise return 1".

These paradoxes can be resolved by using probabilities. Let $\mathcal{M}$ be the set of *probabilistic oracle machines*, defined here as Turing machines which can execute special instructions to (i) flip a coin that has an arbitrary rational probability of coming up heads, and to (ii) call an oracle O, whose behavior might itself be probabilistic.

Roughly speaking, the oracle answers questions of the form: "Is the probability that machine M returns 1 greater than p?" Thus, O takes two inputs, a machine $M \in \mathcal{M}$ and a rational probability $p \in [0,1] \cap \mathbb{Q}$, and returns either 0 or 1. If M is guaranteed to halt and to output either 0 or 1 itself, we want $O(M,p) = 1$ to mean that the probability that M returns 1 (when run with O) is at least p, and $O(M,p) = 0$ to mean that it is at most p; if it is equal to p, both conditions are true, and the oracle may answer randomly. In summary,

$$\begin{aligned} \mathbb{P}(M^O() = 1) \; > \; p \implies \mathbb{P}(O(M,p) = 1) \; = \; 1 \\ \mathbb{P}(M^O() = 1) \; < \; p \implies \mathbb{P}(O(M,p) = 0) \; = \; 1 \end{aligned}$$

where we write $\mathbb{P}(M^O() = 1)$ for the probability that M returns 1 when run with oracle O, and $\mathbb{P}(O(M,p) = 1)$ for the probability that the oracle returns 1 on input (M,p). We assume that different calls to the oracle are stochastically independent events (even if they are about the same pair (M,p)); hence, the behavior of an oracle O is fully specified by the probabilities $\mathbb{P}(O(M,p) = 1)$.

Definition A *query* (with respect to a particular oracle O) is a pair (M,p), where $p \in [0,1] \cap \mathbb{Q}$ and $M^O()$ is a probabilistic oracle machine which almost surely halts and returns an element of $\{0,1\}$.

Definition An oracle is called *reflective on R*, where R is a set of queries, if it satisfies the two conditions displayed above for every $(M,p) \in R$. It is called *reflective* if it is reflective on the set of all queries.

THEOREM 2.1. *(i) There is a reflective oracle.*

(ii) For any oracle O and every set of queries R, there is an oracle O' which is reflective on R and satisfies $\mathbb{P}(O'(M,p) = 1) = \mathbb{P}(O(M,p) = 1)$ for all $(M,p) \notin R$.

PROOF. See Appendix XXX. □

As an example, consider the machine given by $M^O() = 1 - O(M, 0.5)$, which implements a version of the liar paradox by asking the oracle what it will return and then returning the opposite. By the existence theorem, $R = \{(M, 0.5)\}$ admits a reflective oracle. This is no contradiction: We can set $\mathbb{P}(O(M,0.5) = 1) = \mathbb{P}(O(M,0.5) = 0) = 0.5$, leading the program to output 1 half the time and 0 the other half of the time.

3. FROM REFLECTIVE ORACLES TO CAUSAL DECISION THEORY

We now show how reflective oracles can be used to implement a perfect Bayesian reasoner. We assume that each possible environment that this agent might find itself in can likewise be modeled as an oracle machine; that is, we assume that the laws of physics are computable by a probabilistic Turing machine with access to the same reflective oracle as the agent. For example, we might imagine our agent as being embedded in a Turing-complete probabilistic cellular automaton, whose laws are specified in terms of the oracle.

We assume that each of the agent's hypotheses about which environments it finds itself in can be modeled by a (possibly probabilistic) "world program" $H^O()$, which simulates this environment and returns a description of what happened. We can then define a machine $W^O()$ which samples a hypothesis H according to the agent's probability distribution and runs $H^O()$. In the sequel, we will talk about $W^O()$ as if it refers to a particular environment, but this machine is assumed to incorporate subjective uncertainty about the laws of physics and the initial state of the world.

We further assume that the agent's decision-making process, $A^O()$, can be modeled as a probabilistic oracle machine embedded in this environment. As a simple example, consider the world program

$$W^O() = \begin{cases} \$20 & \text{if } A^O() = 0 \\ \$15 & \text{otherwise} \end{cases}$$

In this world, the outcome is \$20 (which in this case means the agent receives \$20) if the agent chooses action 0 and \$15 if the agent chooses action 1.

Our task is to find an appropriate implementation of $A^O()$. Here, we consider agents implementing causal decision theory (CDT), which evaluate actions according to the consequences they cause: for example, if the agent is a robot embedded in a cellular automaton, it might evaluate the expected utility of taking action 0 or 1 by simulating what would happen in the environment if the output signal of its decision-making component were replaced by either 0 or 1.

We will assume that the agent's model of the counterfactual consequences of taking different actions a is described by a machine $W^O_A(a)$, satisfying $W^O() = W^O_A(A^O())$ since in the real world, the agent takes action $a = A^O()$. We assume that both $W^O_A(0)$ and $W^O_A(1)$ halt almost surely and return a value in the domain of $u(\cdot)$. In our example,

$$W^O_A(a) = \begin{cases} \$20 & \text{if } a = 0 \\ \$15 & \text{otherwise} \end{cases}$$

Furthermore, we assume that the agent has a utility function over outcomes, $u(\cdot)$, implemented as a lookup table, which takes rational values in $[0,1]$.[2] Causal decision theory then prescribes choosing the action that maximizes expected utility; in other words, we want to find an $A^O()$ such that

$$A^O() \;=\; \operatorname*{argmax}_a \mathbb{E}\left[u\left(W_A^O(a)\right)\right]$$

In the case of ties, any action maximizing utility is allowed, and it is acceptable for $A^O()$ to randomize.

We cannot compute this expectation by simply running $u(W_A^O(a))$ many times to obtain samples, since the environment might contain other agents of the same type, potentially leading to infinite loops. However, we can find an optimal action by making use of a reflective oracle. This is easiest when the agent has only two actions (0 and 1), but similar analysis extends to any number of actions. Define a machine

$$E^O() := \text{flip}\left(\frac{u(W_A^O(1)) - u(W_A^O(0)) + 1}{2}\right)$$

where flip(p) is a probabilistic function that returns 1 with probability p and 0 with probability $1-p$.

THEOREM 3.1. *O is reflective on $\{(E,1/2)\}$ if and only if $A^O() := O(E,1/2)$ returns a utility-maximizing action.*

PROOF. The demand that $A^O()$ return a utility-maxmizing action is equivalent to

$$\mathbb{E}[u(W_A^O(1))] > \mathbb{E}[u(W_A^O(0))] \implies A^O() = 1$$
$$\mathbb{E}[u(W_A^O(1))] < \mathbb{E}[u(W_A^O(0))] \implies A^O() = 0$$

We have

$$\mathbb{P}(E^O() = 1) = \mathbb{E}\left[\frac{u(W_A^O(1)) - u(W_A^O(0)) + 1}{2}\right]$$

It is not difficult to check that $\mathbb{E}[u(W_A^O(1))] \gtrless \mathbb{E}[u(W_a^O(0))]$ iff $\mathbb{P}(E^O() = 1) \gtrless 1/2$. Together with the definition of $A^O()$, we can use this to rewrite the above conditions as

$$\mathbb{P}(E^O() = 1) > 1/2 \implies O(E,1/2) = 1$$
$$\mathbb{P}(E^O() = 1) < 1/2 \implies O(E,1/2) = 0$$

But this is precisely the definition of "O is reflective on $\{(E,1/2)\}$". $\square$

In order to handle agents which can choose between more than two actions, we can compare action 0 to action 1, then compare action 2 to the best of actions 0 and 1, then compare action 3 to the best of the first three actions, and so on. Adding more actions in this fashion does not substantially change the analysis.

4. FROM CAUSAL DECISION THEORY TO NASH EQUILIBRIA

Since we have taken care to define our agents' world models $W_A^O(a)$ in such a way that they can embed other agents,[3] we need not do anything special to pass from single-agent to multi-agent settings. As in the single-agent case, we model the environment by a program $W^O()$ that contains embedded agent programs $A_1^O, \dots, A_n^O$ and returns an outcome. We can make the dependency on the agent program explicit by writing $W^O() = F^O(A_1^O(), \dots, A_n^O())$ for some oracle machine $F^O(\cdots)$. This allows us to define machines $W_i^O(a_i) := F^O(a_i, A_{-i}^O()) := F(A_1^O(), \dots, A_{i-1}^O(), a_i, A_{i+1}^O(), \dots, A_n^O())$, representing the causal effects of player i taking action a_i.

We assume that each agent has a utility function $u_i(\cdot)$ of the same type as in the previous subsection. Hence, we can define the agent programs $A_i^O()$ just as before:

$$A_i^O() = O(E_i, 1/2)$$
$$E_i^O() = \text{flip}\left(\frac{u_i(W_i^O(1)) - u_i(W_i^O(0)) + 1}{2}\right)$$

Here, each $E_i^O()$ calls $W_i^O()$, which calls $A_j^O()$ for each $j \neq i$, which refers to the source code of $E_j^O()$, but again, Kleene's second recursion theorem shows that this kind of self-reference poses no theoretical problem.

This setup very much resembles the setting of normal-form games. In fact:

THEOREM 4.1. *Given an oracle O, consider the n-player normal-form game in which the payoff of player i, given the pure strategy profile $(a_1,\dots,a_n)$, is $\mathbb{E}[u_i(F^O(a_1,\dots,a_n))]$. The mixed strategy profile given by $s_i := \mathbb{P}(A_i^O() = 1)$ is a Nash equilibrium of this game if and only if O is reflective on $\{(E_1,1/2),\dots,(E_n,1/2)\}$.*

PROOF. For $(s_1,\dots,s_n)$ to be a Nash equilibrium is equivalent to every player's mixed strategy being a best response; i.e., a pure strategy a_i can only be assigned positive probability if it maximizes

$$\mathbb{E}[u_i(F^O(a_i, A_{-i}^O()))] \;=\; \mathbb{E}[u_i(W_i^O(a_i))]$$

By an application of Theorem 3.1, this is equivalent to O being reflective on $\{(E_i,1/2)\}$. $\square$

Note that, in particular, any normal-form game with rational-valued payoffs can be represented in this way by simply choosing F^O to be the identity function. In this case, the theorem shows that every reflective oracle (which exists by Theorem 2.1) gives rise to a Nash equilibrium. In the other direction, Theorem 4.1 together with Theorem 2.1(ii) show that for any Nash equilibrium $(s_1,\dots,s_n)$ of the normal-form game, there is a reflective oracle such that $\mathbb{P}(A_i^O() = 1) = s_i$.

5. FROM NASH EQUILIBRIA TO REFLECTIVE ORACLES

In the previous section, we have shown that a reflective oracle can be used to find Nash equilibria in arbitrary normal-form games. It is interesting to note that we can also go in the other direction: For finite sets R satisfying certain conditions, we can construct normal-form games G_R such that the existence of oracles reflective on R follows from the existence of Nash equilibria in G_R. This existence theorem is a special case of Theorem 2.1, but it not only provides a more elementary proof, but also provides a constructive way of finding such oracles (by applying any algorithm for finding Nash equilibria to G_R).

[2]Since the meaning of utility functions is invariant under affine transformations, the choice of the particular interval $[0,1]$ is no restriction.

[3]More precisely, we have only required that $W_A^O(a)$ always halt and produce a value in the domain of the utility function $u(\cdot)$. Since all our agents do is to perform a single oracle call, they always halt, making them safe to call from $W_A^O(a)$.

Definition A set R of queries is *closed* if for every $(M,p) \in R$ and every oracle O, $M^O()$ is guaranteed to only invoke the oracle on pairs $(N,q) \in R$. It is *bounded* if there is some bound $B_R \in \mathbb{N}$ such that for every $(M,p) \in R$ and every oracle O, $M^O()$ is guaranteed to invoke the oracle at most B_R times.

Definition Given a finite set $R = \{(M_1,p_1),\ldots,(M_n,p_n)\}$ and a vector $\vec{x} \in [0,1]^n$, define $O_{\vec{x}}$ to be the oracle satisfying $\mathbb{P}(O_{\vec{x}}(M_i,p_i)=1) = x_i$ for $i=1,\ldots,n$, and $\mathbb{P}(O_{\vec{x}}(M,p)=1)=0$ for $(M,p) \notin R$.

THEOREM 5.1. *For any finite, closed, bounded set $R = \{(M_1,p_1),\ldots,(M_n,p_n)\}$, there is a normal form game G_R with $m := n\cdot(2B_R+1)$ players, each of which has two pure strategies, such that for any Nash equilibrium strategy profile $(s_1,\ldots,s_m)$, the oracle $O_{\vec{x}}$ with $\vec{x} := (s_1,\ldots,s_n)$ is reflective on R.*

PROOF. We divide the $n\cdot(2B_R+1)$ players in our game into three sets: the *main players* $i=1,\ldots,n$, the *copy players* $g(i,j) := j\cdot n + i$, and the *auxiliary players* $h(i,j) := (B_R+j)\cdot n + i$, for $i=1,\ldots,n$, $j=1,\ldots,B_R$.

The mixed strategy s_i of a main player i will determine the probability that $O_{\vec{x}}(M_i,p_i)=1$. We will force $s_{g(i,j)} = s_i$, i.e., we will force the mixed strategy of each copy player to equal that of the corresponding main player; thus, the copy players will provide us with independent samples from the Bernoulli(s_i) distribution, allowing us to simulate up to B_R independent calls to $O(M_i,p_i)$. Finally, the auxiliary players are used to enforce the constraint $s_{g(i,j)} = s_i$, by having the copy player $g(i,j)$ play a variant of Matching Pennies against the auxiliary player $h(i,j)$.

In order to define the game's payoff function, note first that by writing out each possible way that the at most B_R oracle calls of $M_i^{O_{\vec{x}}}()$ might come out, we can write the probability that this machine returns 1 as a polynomial,

$$\mathbb{P}(M_i^{O_{\vec{x}}}()=1) \;=\; \sum_{k=1}^{K} c_{i,k} \prod_{i'=1}^{n} x_{i'}^{d_{i,k,i'}}$$

where $d_{i,k,i'} \le B_R$. We want to force the main player i to choose pure strategy 1 if this probability is strictly greater than p_i, pure strategy 0 if it is strictly smaller.

To do so, we set player i's payoff function $u_i(\vec{a})$ to

$$u_i(\vec{a}) \;=\; \begin{cases} \sum_{k=1}^{K} f_{i,k}(\vec{a}), & \text{if } a_i = 1, \\ p_i, & \text{otherwise} \end{cases}$$

where

$$f_{i,k}(\vec{a}) \;=\; \begin{cases} c_{i,k} & \text{if } a_{g(i',j)} = 1 \;\forall 1 \le i' \le n, 1 \le j \le d_{i,k,i'} \\ 0 & \text{otherwise} \end{cases}$$

Then, assuming we can guarantee $s_{g(i,j)} = s_i$, the expected payoff of strategy 1 to player i is exactly $\mathbb{P}(M_i^{O_{\vec{x}}}()=1)$, while the payoff of strategy 0 is always p_i; hence, as desired, the Nash equilibrium conditions force i to choose 1 if the probability is greater than p_i, 0 if it is smaller.

It remains to choose the payoffs $(u_{g(i,j)}(\vec{a}), u_{h(i,j)}(\vec{a}))$ of the copy and auxiliary players. In order to force $s_{g(i,j)} = s_i$, we set these payoffs as follows:

$a_i = 0$		
	$a_{h(i,j)} = 0$	$a_{h(i,j)} = 1$
$a_{g(i,j)} = 0$	$(1,0)$	$(0,0)$
$a_{g(i,j)} = 1$	$(0,1)$	$(1,0)$

$a_i = 1$		
	$a_{h(i,j)} = 0$	$a_{h(i,j)} = 1$
$a_{g(i,j)} = 0$	$(1,0)$	$(0,1)$
$a_{g(i,j)} = 1$	$(0,0)$	$(1,0)$

We show in Appendix A that at Nash equilibrium, these payoffs force $s_{g(i,j)} = s_i$. $\square$

Theorem 5.1 is a special case of Theorem 2.1(i). The proof can be adapted to also show an analog of Theorem 2.1(ii), but we omit the details here.

6. RELATED WORK

Space-time embedded intelligence, technical agenda ← Benja can handle

Ken Binmore. Modeling rational players: Part i. Economics and Philosophy, 3(02):179âĂŞ, 1987.

Binmore discusses the philosophical justification for game theoretic concepts such as Nash equilibria in the context of players controlled by Turing machines that have access to each other's source code. He notes that paradoxes (of the type that reflective oracles partially resolve) can prevent these programs from both predicting each other's actions and participating in the game:

> In any case, if Turing machines are used to model the players, it is possible to suppose that the play of a game is prefixed by an exchange of the players' Godel numbers... Within this framework, a perfectly rational machine ought presumably to be able to predict the behavior of the opposing machines perfectly, since it will be familiar with every detail of their design. And a universal Turing machine *can* do this. What it *cannot* do is predict its opponents' behavior perfectly *and* simultaneously participate in the action of the game. It is in this sense that the claim that perfect rationality is an unattainable ideal is to be understood.

http://plato.stanford.edu/entries/decision-causal/ (the part about ratification)

Joyce, James and Allan Gibbard. 1998. âĂIJCausal Decision Theory.âĂİ In Salvador Barbera, Peter Hammond, and Christian Seidl, eds., Handbook of Utility Theory (Volume 1: Principles), Dordrecht: Kluwer Academic Publishers, pp. 627âĂŞ666.

Joyce and Gibbard (1998) describe one justification for mixed Nash equilibria in terms of causal decision theory. Specifically, they discuss a *self-ratification* condition that extends CDT to cases when one's action is evidence of different underlying conditions that might change which actions are rational. An action self-ratifies if and only if it causally maximizes expected utility in a world model that has been updated on the evidence that this action is taken.

For example, consider the setting of a matching pennies game where players can predict each other accurately. The fact that player A plays "heads" is evidence that player B will predict that player A will play "heads" and play "tails" in response, so player A would then have preferred to play "tails", and so the "heads" action would fail to self-ratify. However, the mixed strategy of flipping the coin *would* self-ratify. In our model, agents' actions are statistically independent, but the reflection principle encodes some global constraints on players' mixed strategies that are similar to self-ratification.

7. DISCUSSION

APPENDIX

A. NASH EQUILIBRIA IN A VARIANT OF MATCHING PENNIES

We have a 2 by 2 by 2 game. Player Row has strategies Up and Down, player Column has strategies Left and Right, and player Matrix has strategies Front and Back. The payoffs of (Row, Column) are as follows:

$(1,0)$	$(0,0)$
$(0,1)$	$(1,0)$

$(1,0)$	$(0,1)$
$(0,0)$	$(1,0)$

The first matrix indicates the payoffs when Matrix plays Front, and the second matrix indicates the payoffs when Matrix plays Back.

Write p for the probability that Row plays Down, and q for the probability that Matrix plays Back. At Nash equilibrium, we have $p = q$.

- Case 1: $0 < q < 1$.

  Suppose that there is a Nash equilibrium where Column plays Left. Then Row would play Up, but then Column would strictly prefer Right, which is a contradiction.

  Suppose that there is a Nash equilibrium where Column plays Right. Then Row would play Down, but then Column would strictly prefer Left, which is a contradiction.

  So at every Nash equilibrium, Column must mix between strategies. Thus, at equilibrium, Column must be indifferent between Left and Right. This is equivalent to $p(1-q) = (1-p)q$. This implies $p > 0$, since otherwise we'd have $0(1-q) = (1-0)q$, i.e. $0 = q$, but we assumed $0 < q < 1$. Thus, we can divide the equation by pq, yielding:

  $$\begin{aligned} &(1-q)/q = (1-p)/p \\ \Leftrightarrow &1/x - 1 = 1/p - 1 \\ \Leftrightarrow &1/q = 1/p \\ \Leftrightarrow &q = p \end{aligned}$$

- Case 2: $q = 0$.

  This gives us the following payoff matrix:

$(1,0)$	$(0,0)$
$(0,1)$	$(1,0)$

  Suppose that there is a Nash equilibrium with $p > 0$. Then at this equilibrium, Column must play Left; but if Column plays Left, then Row strictly prefers Up, which contradicts $p > 0$. Hence, we must have $p = 0 = q$.

- Case 3: $q = 1$.

  This gives us the following payoff matrix:

$(1,0)$	$(0,1)$
$(0,0)$	$(1,0)$

  Suppose that there is a Nash equilibrium with $p < 1$. Then at this equilibrium, Column must play Right; but if Column plays Right, then Row strictly prefers Down, which contradicts $p < 1$. Hence, we must have $p = 1 = q$.