# User-Agent Value Alignment

**Daniel Shapiro**
Institute for the Study of Learning and Expertise
2164 Staunton Court
Palo Alto, CA 94306

**Ross Shachter**
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305

**Pat Langley**
Institute for the Study of Learning and Expertise
2164 Staunton Court
Palo Alto, CA 94306

## Abstract

The principal-agent problem concerns delegation in the absence of trust. Given a principal and an agent with different value structures, the principal wants to motivate the agent to address the principal's aims by providing appropriate incentives. We address this problem in the context of a real-world complication, where the principal and agent lack a common problem frame. This context is especially relevant when the principal is a user, and the agent is a technological artifact with a limited repertoire of percepts and actions. We identify necessary conditions for establishing trust between such disparate actors, and we show, via a constructive proof, that it is always possible to create these conditions. Next, we show when and how the principal can rank the expected quality of agent behavior. We conclude by discussing the practical construction of aligned agents. We present an implemented tool for building them, and the results of several experiments that compare the behavior of aligned and unaligned agents in their user's eyes.

## 1 INTRODUCTION

The principal-agent problem can arise in any situation that calls for the delegation of responsibility. If the principal and agent hold different values, the task is to develop incentive

structures (e.g., to build contractual relations) that ensure the agent serves the principal's interests while acting outside the principal's supervision. For example, when a homeowner (as principal) employs a contractor (as agent) to put on a new roof, a reasonable contract would penalize schedule delays and thus mediate the homeowner's concern with time against the contractor's desire to take on more work. The straightforward, but implicit assumption is that the contractor and homeowner perceive time and dollars in comparable ways, as a backdrop for building the necessary incentives.

This assumption breaks down on deeper inspection. As an expert in roof work, the contractor possesses skills and distinctions that the homeowner lacks, and carries a value structure over those distinctions that will guide a variety of choices while conducting the work. The contractor's functional and aesthetic decisions will affect the principal's value, outside any contractual relationship relating time to completion and cost. To dramatize this point, imagine a color-blind interior decorator. This agent cannot even perceive things that matter to the principal. How can we establish trust in a situation of this kind? How well can any agent perform in the face of such a barrier? Current principal-agent theory lacks tools for bridging this gap.

The same issues commonly arise in the interaction between people and machines. For example, while a cruise control (as agent) maintains a desired vehicle speed, the driver (as user) cares about the distance to the car in front. Since this distinction lies outside the cruise control's ken, the driver has to monitor it very carefully. However, we would like to build more autonomous tools. Consider an autopilot for a car on an automated freeway; when you climb into this vehicle for a ride to the airport, you ask an artifact to make decisions in your stead. It will choose routes, select maneuvers, and react to traffic, while you would like to arrive safely, relatively unruffled, and on time for your flight. You care about the agent's methods and its end result, but you

might have very little insight into its sensor observations and action strategies. The gap between agent and user reference frames acts as a barrier to trust that decreases our willingness to deploy such systems.

This paper provides a framework for establishing trust between such disparate actors. We identify necessary conditions for aligning the value structures of a user and an agent despite a gap in reference frames, and we show that it is always possible (but potentially costly) to create these conditions. This produces harmony; an aligned and cooperative agent will provably maximize the user's utility as a byproduct of maximizing its own reward. As a result, the user will be as happy as possible with the agent's behavior.

## 2 THE PROBLEM FRAME

We address these questions in the context of a decision theoretic problem frame. We represent the principal's concerns by a utility function, and the agent's by an analogous function called its reward. We treat the principal as a passive evaluator of the agent's behavior, and cast the agent as the sole actor. Thus, we give the agent a set of mutually exclusive and collectively exhaustive observations about state, which it obtains before choosing from a set of actions that impact the principal's preferences.

We illustrate this framework in Figure 1. Here, **U** is utility, **R** is agent reward, and **x** and **y** are attribute vectors sensed by the agent and principal respectively. **R** and **U** are deterministic functions of **x**, and **y**. **D** represents agent decisions, and **o** stands for the agent observations used to select **D**. In this influence diagram (Howard & Matheson, 1984) arcs between uncertainties represent possible conditional dependence, while the absence of such arcs denote conditional

independence. Arcs to decision nodes represent information available at the time of decision, and arcs downstream from decision nodes represent the effects of those actions.

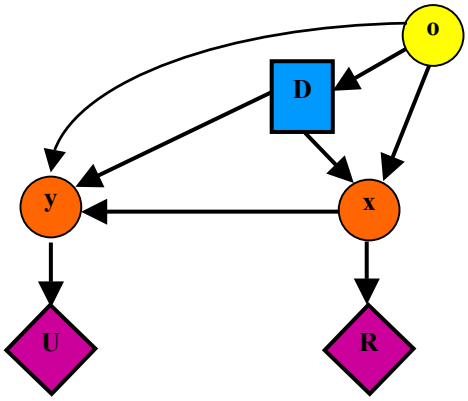


Figure 1. The joint principal-agent problem frame.

In order to establish trust between the principal and the agent, we want to align **R** with **U** such that the agent's decisions, **D**, maximize the principal's utility. The task is difficult for two reasons. First, **R** and **U** can be based on different attribute sets, meaning that the agent cannot sense **y**, and it cannot necessarily represent **U**. Its decisions can therefore diverge from the ones the principal would prefer. Second, since the agent's behavior can inadvertently impact utility, the agent can adversely affect the principal's utility in the process of pursuing its reward.

In general, **U** is predetermined, while **R** is fashioned to balance the agent's interests in its decisions with incentives that encourage cooperation with the principal. If, however, the agent is an artifact and the principal is a human user, we can engineer the agent's reward to capture the user's intent. We can also construct the agent's action suite (its decisions) such that it has the capacity to perform well in the user's eyes. The notion of incentives is irrelevant in this case, since the agent will do the user's bidding by design. Instead, the problem is communication; we need to provide a reward function and a decision frame that let the agent maximize **U**.

The remainder of this paper will focus on the case where the agent is an artifact serving the interests of a human user. We will employ the principal-agent vocabulary whenever the agent can be either human or a machine.

## 3 VALUE ALIGNMENT

In order to establish trust between a user and an agent, we need to ensure that the agent is motivated by the user's concerns, and that the agent's actions will not impact the user in unforseen ways. We address these concerns in two parts. First, we define a conditional independence relationship between the agent and user problem frames called *structural value alignment*. When it holds, the agent can recognize all ways its actions affect user utility. Given structural value alignment, we identify a numerical manipulation of agent reward that produces *functional value alignment*, a situation in which the agent chooses an action that would have been preferred by the user. We will show that it is always possible to establish structural alignment, and thus that we can achieve functional alignment between any user and any agent. However, there may be a cost.

### 3.1 Structural value alignment

One way to ensure that an agent will address the user's concerns is to give it an exact copy of the user's utility function. If $\mathbf{R} \equiv \mathbf{U}$ (and therefore $\mathbf{x} \equiv \mathbf{y}$), the agent will obviously perform as well as it possibly can for the user. However, we cannot achieve this unity because users and artificial agents perceive the world in decidedly different terms. To make this concrete, assume a human driver cares about safety (an element of $\mathbf{y}$). When we try to construct an agent with the ability to perceive safety we discover that the transformation from accessible measures like distance and velocity is apparently easy for people but difficult for machines. Other percepts (like a precise

measure of time to impact) will be easier for machines to acquire. Some such asymmetry will apply for any conceivable artificial entity because it is a consequence of available technology. Thus, the mechanism for aligning artifacts with humans must bridge reference frames.

Our solution is to define agent-held surrogates for user concerns. In particular, we look for a set of distinctions that can function as a sufficient (versus a complete) surrogate for the attributes underlying user utility, such that the agent can only effect user utility through attributes the agent cares about as well.

Figure 2 illustrates this condition. It states that agent action can only affect user utility via a change in **x**. More formally, it says that **y** is conditionally independent of **D** and **o** given **x**, and that user utility is *caused* by **x** with respect to the agent's decisions (Heckerman & Shachter, 1995). We call this relation *structural value alignment*, and assume it holds across time periods. That is, $\mathbf{y}_t$ is conditionally independent of past history, $\vec{\mathbf{D}}, \vec{\mathbf{o}}$, given $\mathbf{x}_t$.

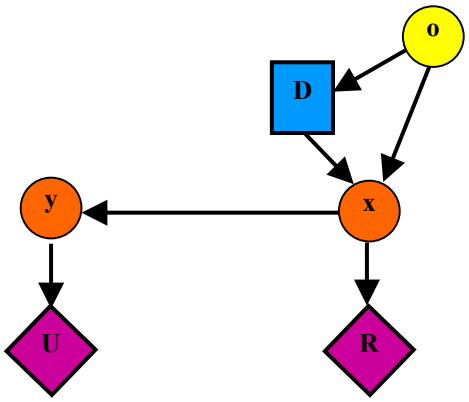

Figure 2. Structural value alignment.

### 3.2 Functional value alignment

If all interaction between the agent and the user passes through **x**, we can motivate the agent to address user concerns. In particular, we can choose the agent's reward function so that the

policy that produces the highest expected reward stream for the agent also produces the highest possible expected utility stream for the user. We call this condition *functional value alignment*. That is, if $\vec{D}$ is a time series of agent decisions $D_0 \ldots D_t$, $\vec{o}$ is a similar time series of agent observations, and $\gamma$ is the user's discount rate,

$$\begin{aligned} &\underset{\vec{D}}{\operatorname{argmax}} \sum_t \gamma^t \underset{y_t}{E}\left[ U(y_t) | \vec{o}, \vec{D} \right] = \\ &\underset{\vec{D}}{\operatorname{argmax}} \sum_t \gamma^t \underset{x_t}{E}\left[ R(x_t) | \vec{o}, \vec{D} \right] \end{aligned}$$

for all possible observations, $\vec{o}$. The following theorem expresses this condition.

**Theorem (Functional value alignment):** If structural value alignment holds, we can choose the agent's reward function $R(x_t)$ such that functional value alignment holds.

**Proof** (by construction): We simply define **R** as the expected utility for the agent's observations. That is,

$$R(x_t) \equiv E_{y_t}\left[ U(y_t) | x_t \right] = E_{y_t}\left[ U(y_t) | , x_t, \vec{o}, \vec{D} \right]$$

The second clause follows from the definition of structural value alignment. Given this construction,

$$\begin{aligned} \underset{y_t}{E}\left[ U(y_t) | \vec{o}, \vec{D} \right] &= \underset{x_t}{E}\left[ \underset{y_t}{E}\left[ U(y_t) | x_t \right] | \vec{o}, \vec{D} \right] \\ &= \underset{x_t}{E}\left[ R(x_t) | \vec{o}, \vec{D} \right] \end{aligned}$$

Thus, the agent's optimal policy also maximizes user expected reward. An analogous theorem applies if the object is to maximize the average reward stream as opposed to a discounted sum. ♦

Functional value alignment guarantees the optimality of the agent's policy in human eyes no matter how **x**, **D**, and **o** are related within or across time periods. Moreover, the relationship between agent and the user can be quite broad. The user can care about attributes outside the agent's ken (elements of **y** can be independent of **x, o,** and **D**), and the agent can care about attributes that are irrelevant to the user (elements of **x** can be independent of **y**). However, anything the user cares about that the agent can observe or effect must be visible in **x**.

### 3.3 Positive and negative examples

The concept of alignment may become clearer if we examine positive and negative examples. Figure 3 illustrates the positive case. Here, we assume that the agent decides whether to slow down or cruise after observing the car in front of it, and that its action in the given situation may alter the time to impact. Structural alignment holds if either: (1) the agent can only affect safety by changing the time to impact; or (2) the user knows the time to impact, and knowledge of the agent's action (or observations) cannot alter his/her assessment of safety. That is, time to impact is a sufficient surrogate for the user's concern with safety.

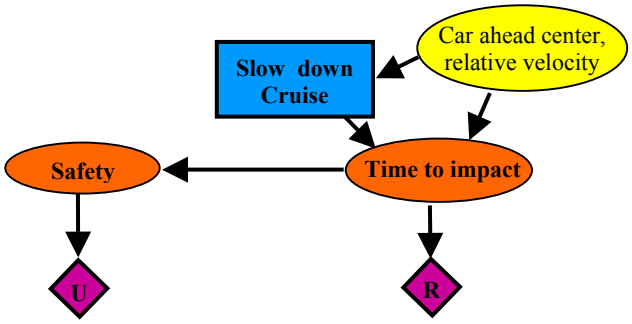


Figure 3. A positive example of value alignment.

Given structural alignment, we can produce functional alignment by giving the agent a reward function that mirrors human utility. We can do this empirically via an assessment process. We ask the agent to report the time to impact, **x**, and ask the user to simultaneously assess the utility

of the current situation. This results in an attribute vector, **x**, and a utility **U(y)** given **x**. Since the user may observe many attribute vectors, **y**, when the agent sees a specific **x**, we repeat the experiment to extract an expected utility. We set

$$R(\text{time to impact}) = E[U(\text{safety} \mid \text{time to impact})]$$

and repeat this process (in concept) for all values of time to impact. In practice, we would rely on approximation functions instead of this exact, tabular form. When we equate the agent's reward function to the user's expected utility, we motivate the agent to achieve against the human-held standard on the basis of available measures. This lets the agent focus on increasing time to impact with the side effect (known to us) that its actions will increase user safety.

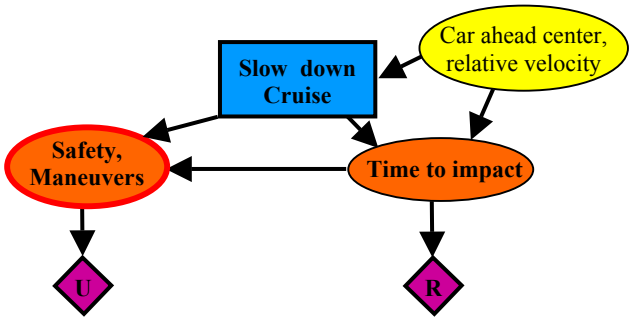


Figure 4. Side-effects can undo structural alignment.

Alignment fails when the agent can impact user utility without altering features the agent cares about as well. Figure 4 provides an example of this situation. Here, we assume the user has preferences over the motions of the car (perhaps he is a queasy passenger), implying that the agent can make the user arbitrarily ill in the process of increasing time to impact. So, actions that do not affect time to impact can effect the user nonetheless. To capture this effect, the influence diagram includes an arc between the agent's decision and the feature set that determines user utility. Here, the agent is unaware its actions have an undesirable side effect and

it is not motivated to correct the problem. This conflict cannot be removed by any change to **R** within this problem structure.

Figure 5 illustrates how structural value alignment can fail if the agent makes additional observations. Here, we begin with the problem structure of Figure 3, but assume that the agent can also sense the maneuver frequency of the car in front. This observation is relevant to user safety because it provides insight into the likely future behavior of that vehicle. (High values suggest an erratic driver.) It is a direct effect because a change to maneuver frequency plausibly alters the perception of safety even if time to impact is known.

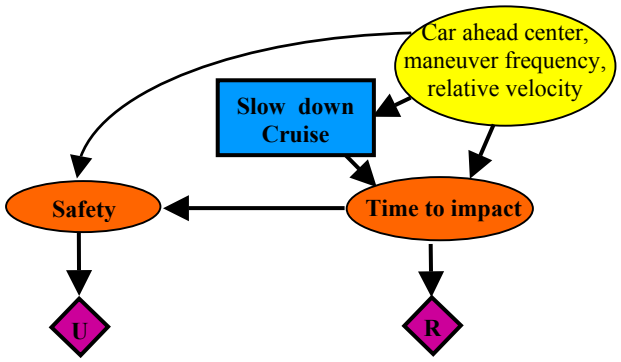


Figure 5. Observations can destroy structural alignment.

The agent's optimal policy may not maximize user utility in this situation. In particular, the user might prefer the agent to slow down in the presence of an erratic driver because of the safety implications, while the agent's focus on time to impact could lead it to maintain speed (i.e., to "cruise"). Note that the agent's optimal policy in Figure 5 would be identical to its optimal policy in Figure 3. (The agent would observe and simply ignore maneuver frequency.) The difference is that in Figure 5 the user is aware the agent has access to a better mapping from situation to action (better for the user), and would like the agent to incorporate those observations into its policy.

### 3.4 The value alignment theorem

We have shown that we can align agent reward with user utility whenever the conditional independence relation called structural value alignment holds. This permits functional value alignment: the happy situation where the agent's best policy simultaneously maximizes user reward. We have also shown, by example, that we cannot (in general) produce functional value alignment in the absence of structural value alignment. We summarize these results in the following theorem.

**Theorem (Value alignment):** Structural value alignment holds if and only if functional value alignment can be satisfied for all problem frames consistent with the diagram.

**Proof:** Let $p$ be any probability distribution over **x**, **y**, **o**, **U** and **D**, that is consistent with the model. We need to show that structural value alignment is a necessary and sufficient condition for functional value alignment. Assume structural alignment holds. As shown earlier, setting $R(x) \equiv E_y[U(y)|x]$ establishes functional value alignment. If structural alignment does not hold, then **D** or **o** is relevant to **y** given **x**. Suppose it is **D**. In this case, we choose $p$ so that **x** is irrelevant to **D**, and no reward function **R** can recognize the effect of **D** on **U**. If **o** is relevant to **y** given **x** instead, choose $p$ so that **x** is irrelevant to **o** given **D**. Again, no reward **R** can represent the expected value of **U** given **o** and **D**. This establishes the necessity of structural alignment. ♦

## 4 ESTABLISHING ALIGNMENT

Surprisingly, we can always establish structural, and therefore functional value alignment where none would appear to exist. We discuss two methods. The first (and less general technique) revolves around the invention of clever surrogates for user concerns. The second applies in general, but requires a potentially extensive procedure for defining **R**.

### 4.1 The explicit method

We can create alignment by inventing additional surrogates for user-held concerns. Consider the example in Figure 6. If acceleration is a sufficient surrogate for the user's concern with maneuvers, we can address the user's queasiness by giving the agent an accelerometer and including acceleration in the agent's reward function. This reestablishes structural value alignment, since the agent can only affect the utility-laden attributes of safety and maneuver by changing conditions that also matter to the agent. We create functional value alignment by tuning the numeric values of the agent's reward to reflect the user's expected utility, as before.

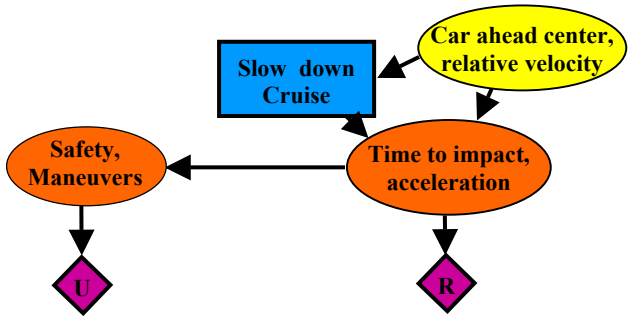


Figure 6. Explicit alignment requires surrogate attributes.

It may not always be possible to establish explicit alignment because the process can require new sensors and perceptual software. In addition, the environment has to support the desired conditional independence relation between the agent's actions and their effect on user utility, given the agent-held distinctions we are able to invent and instrument. The next section discusses a method of establishing alignment that overcomes these concerns.

### 4.2 The implicit method

The second method of creating alignment applies without the need to invent new surrogates for user concerns. We simply include the agent's actions and observations as attributes in its reward function, and implicitly incorporate their effects on user utility during the assessment process.

This approach transforms Figure 4 into Figure 7. However, the path to functional value alignment from this base now requires us to assess the utility of many combinations of agent perceptions and maneuver choice: What was the time to impact? Did the agent slow down, or cruise? Given these measures, how safe and/or queasy did the user feel?

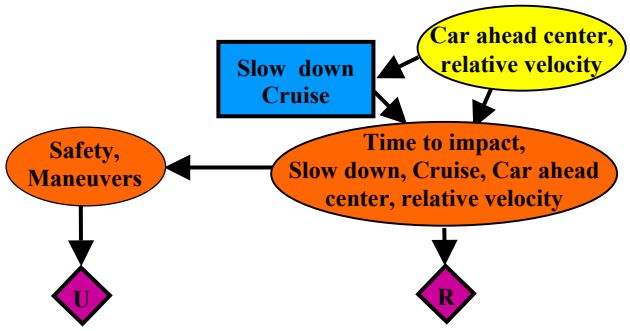


Figure 7. Implicit alignment requires no new attributes.

This assessment may require a large quantity of effort. If *m* agent-held attributes influence *n* user attributes, we need to assess *m* x *n* attribute pairs with all their associated degrees of distinction. Well-chosen surrogates will tend to decouple this problem, as in Figure 6, where time to impact influenced safety and acceleration influenced queasiness, with no cross-effects. That assessment required *m* one-to-one mappings. In practice, the actual size of the assessment problem is open to test. Although the combinatorics argue against large *m* and *n*, we can expect to define some useful surrogates, and we will still build agents for constrained domains, even while agent autonomy increases. Users may also bring a limited set of concerns to particular agent applications. In addition, since the agent's value structure is only relevant where its plans offer a choice, we can constrain the dimension of the agent's reward function by limiting its options. Finally, we can imagine agents that employ user feedback to learn which attributes affect utility, and which do not. Taken together, these factors act to reduce the size of the attribute sets *m* and *n*, and the complexity of the interactions between them.

If good surrogates are hard to find, it is important to notice that we can always include the agent's actions and observations as attributes in its reward function. This approach produces structural value alignment through a kind of default path, since every agent action and observation must affect agent reward en route to altering user utility. The consequent, suprisingly, is that we can *always* produce the conditions satisfying the value alignment theorem.

## 5 RANKING AGENTS

Although functional value alignment ensures maximally good agent behavior, it says nothing about how good that behavior will be in absolute terms. As a simple illustration, consider the null agent (a stone), that has no action options and makes no observations about its world. Although this agent does nothing, it meets the criteria of the value alignment theorem since it never takes actions different from those preferred by the user. At the opposite extreme, consider a perfect user clone. This agent is also value aligned, and it serves the user in a more or less ideal way (although some agents could do better). In general, we would like to rank agents in the user's eyes. We offer a few simple propositions towards this end. The first compares aligned agents with each other, while the second compares aligned and unaligned agents.

Given two agents, A and B, we say that A is *at least as capable* as B if A possesses a weak superset ($\supseteq$) of B's observations and decision options. That is, A's observations and decision options are at least as fine-grained as B's. We use the notation $A \succeq B$ to mean "A is weakly preferred to B". Under these definitions:

**Proposition (Aligned preference):** If A and B are functionally value aligned agents, $A \succeq B$ if A is at least as capable as B.

**Proof:** For any situation observed by B, the choice that A would make carries at least as much expected user utility as the choice that B would make. ♦

In particular, if A's extra knowledge and capabilities provide no added user utility, A can employ the same optimal policy found by B. If A can do better, as a result of making more observations, or applying different actions, it will.

**Proposition (Unaligned preference):** If A is a functionally value aligned agent and B is not, $A \succeq B$ if A is at least as capable as B.

**Proof:** By definition, A will employ a policy that maximizes expected user utility, yet A has access to any policy B chooses. ♦

A might appear superior to B if the attribute set for its reward function is more fine-grained. In this case, we might be able to establish functional value alignment for A and not for B. However, whenever B is functionally aligned, A will be as well, and the user will be indifferent between them if they are equally capable.

In summary, the user should always pick an aligned agent over any less capable agent, and by extension, we should always employ a more skilled individual that has our interests at heart. In contrast, the propositions offer no advice for a choice among unaligned agents, or on the interesting problem of comparing an aligned agent against an unaligned one that is more capable. This is a practical concern, since it represents the choice between an unskilled (but dutiful) worker and a skilled (but independent) practitioner. Our analysis makes it clear that dutiful workers will address our aims, while skilled but unaligned practitioners have the potential to impact our utility in many ways (intentionally and unintentionally) and they have the capacity to

select actions that diverge from the ones we would have them choose. Motivational differences may also play a role. In short, delegation without trust carries risk.

## 6 LEARNING ALIGNED POLICIES

Now that we have developed a theoretical structure for user-agent value alignment, we turn to the underlying goal of implementing real value-aligned agents. This requires solutions to two key problems. First, we need a practical method of establishing value alignment. Towards that end we have developed (in the previous section) one approach that is guaranteed to succeed, and another, more heuristic version whose efficiency is subject to empirical test. Next, given alignment, we need a practical technique for discovering the agent's optimal action policy. That is, we would like to find the **D*** that simultaneously maximizes **R** and **U**.

In general, it will be difficult (or impossible) to find the optimal policy if the relations between **o**, **D**, and **x** are unconstrained over time. However, the problem becomes theoretically tractable if the system obeys additional assumptions. For example, Figure 8 depicts one set of possible Markov assumptions. Here, **o(t)** is conditionally independent of $\mathbf{o_{t-k}}$, $\mathbf{D_{t-k}}$, for all **k** greater than 1, given $\mathbf{o_{t-1}}$ and $\mathbf{D_{t-1}}$. Thus, $\mathbf{x_t}$ and **R(t)** are independent of the past, given $\mathbf{o_t}$ and $\mathbf{D_t}$. We also assume the agent and the user carry a standard, additive model of reward.

Many reinforcement learning algorithms (Sutton & Barto, 1998) can find an optimal policy for such stochastic systems, where the object is to maximize (or improve) an expected reward stream, most often a future discounted sum of in-period rewards. While reinforcement learning typically treats the system dynamics as an unknown, 'model-free' methods never even attempt to determine an explicit probabilistic mapping between states. Instead, they learn a direct map from action to the expected reward stream by searching the space of possible control policies. This

strategy is appropriate to our context where the agent has a limited cognitive power, but the opportunity to interact with its environment over a long period of time.

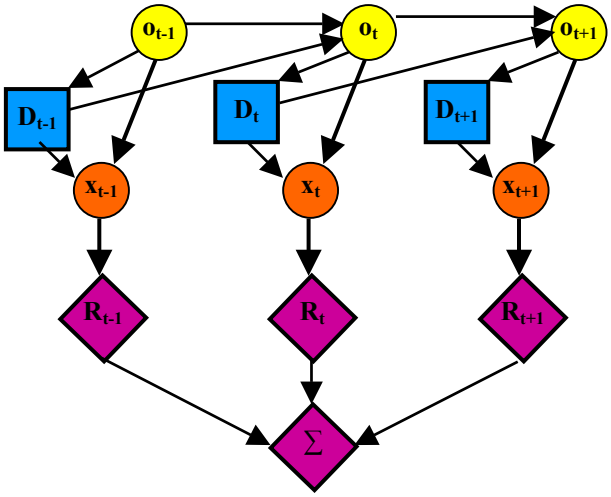


Figure 8. The agent's policy learning problem as a Markov decision process.

As an example, a well-known algorithm called SARSA (Sutton & Barto, 1998) operates on state-action pairs. It learns an estimate for the value of taking a given action in a given state by sampling its future trajectory. SARSA repeats the following steps: select and apply an action in the current state; measure the in-period reward; observe the subsequent state and commit to an action in that state; and update the estimate for the starting state-action pair, using its current value, the current reward, and the estimate associated with the destination pair.

In other words, SARSA bootstraps; it updates value estimates with other estimates, grounding the process in a real reward signal. Singh, Jaakola, Littman, and Szepesvari (2000) have recently shown that SARSA converges to the optimal policy and the correct values for the future discounted reward stream. (The proof imposed common Markov assumptions, required an exact (tabular) representation of the true reward function, and allowed a range of action selection policies that guaranteed sufficient exploration of apparently sub-optimal choices.)

While SARSA converges in theory, it behaves rather poorly in practice. To illustrate why, consider a simple, 'flat' formulation for the task of learning how to pilot a car on a freeway. Here, let **D** be a set of five decisions: change lane to the left or right, slow down, speed up, and cruise at the current speed in the current lane. Let **o** be a set of 15 observations, for the presence or absence of up to six adjacent cars, their relative velocity (faster/slower), the distance to cars in lane (near/far), and own speed relative to a desired set point. Finally, let **x** measure the time to impact ahead and behind, and the agent's difference from its target speed. Given this setup, the cross product of feasible observations and decisions contains ~20,000 state-action pairs, and SARSA has to search this space for the right action sequence (the one that maximizes the future discounted sum over **R(x)**). When we implemented this design, SARSA failed to find the optimal policy in a practical amount of time (i.e., within 250,000 iterations of the algorithm).

If a standard, convergent learning algorithm fails to behave practically on such a simple example, we clearly need more sophisticated tools to address real applications. Our solution is to infuse the learning process with domain knowledge. Instead of considering all feasible **D** and **o**, we construct a hand-coded program that controls behavior in some cases, but allows options in others. The space of possible options defines a much more constrained optimization problem.

We have embodied this idea in an architecture for creating 'value-driven' agents that learn (Shapiro, 2001). This architecture provides a language (called Icarus) for expressing agent behavior, whose interpreter embeds a reinforcement learning algorithm (itself an extension of SARSA). This marriage supports a novel development process. Here, the programmer writes an approximately correct plan that determines agent behavior in some cases but provides options in others, the agent isolates the best options by learning from experience, and the user supplies an appropriately aligned reward function to guide the agent's optimization.

Table 1 contains an example of an Icarus plan. It describes driving as an ordered set of objectives: to react to emergencies, to react to trouble in front (defined as a slower car ahead), to get to one's own target speed, to react to trouble behind, and then to simply cruise if none of the previous contexts apply. The Icarus interpreter considers these clauses in order, starting with the first one on each execution cycle. If a clause returns true, no action was necessary and the interpreter advances to the next step. If it returns false, the plan fails. Alternatively, a clause can return an action that furthers a subplan's objectives. Icarus collects and returns these actions, and then selects one for execution. Icarus agents are extremely adaptive because this style of interpretation lets the locus of execution shift from any statement to any other in one time step.

Table 1. The top level of an Icarus driving plan.

```
Drive ()
  :objectives
   [ Not( Emergency-brake() )
     Not( Avoid-trouble-ahead() )
     Get-to-target-speed();
     Not( Avoid-trouble-behind() )
      Cruise() ]
```

While the top-level plan for driving lacks options, they appear within the subplan in Table 2 for avoiding trouble in front. This subplan causes the agent to look for a slower car ahead, and lets it choose among the options to move left, move right, slow down, or cruise. This choice is the focus of learning. The learning algorithm updates estimates for the reward stream generated by each option as the agent gains experience. (It calculates those estimates using the observations passed into the options as parameters in Table 2). If several options apply, Icarus selects one on the basis of its expected value. We have shown that Icarus' learning algorithm converges to the optimal policy under a common set of Markov assumptions (Shapiro & Shachter, 2000), even when the plan expresses a hierarchy of choices over abstract strategies and primitive actions. Other efforts in hierarchical reinforcement learning provide similar guarantees, but pay less

attention to the concerns of programming (Dietterich, 2000; Sutton, Precup, & Singh, 1998; Parr & Russell, 1998; Andre & Russell, 2001).

The use of background knowledge has a profound impact on learning. Relative to the 'flat' plan for driving, we have shown that the application of domain knowledge decreases plan size by three orders of magnitude, increases learning rate by two orders of magnitude, and provides better asymptotic performance as the effort dedicated to learning approaches the feasible limit (Shapiro, Langley, & Shachter, 2001). These improvements are substantial and suggest that our approach offers a qualitative increase in the scope and efficacy of policy learning systems. More to the point, it suggests that we can efficiently learn optimal, and near-optimal policies for non-trivial problem domains, even when the Markov assumptions that support theoretical convergence proofs do not hold.

Table 2. The plan for avoiding trouble ahead.

```
Avoid-trouble-ahead ()
   :requirements
     [?c = car-ahead-center()
       velocity() >  velocity(?c)
      ?tti = time-to-impact()
      ?rd = distance-ahead()
      ?rt = target-speed() – velocity()
      ?art = abs(?rt) ]

    :options
     [Safe-cruise (?tti ?rd ?art)
      Safe-slow-down (?tti ?rd ?rt)
      Move-left (?art)
      Move-right (?art) ]
```

## 7 EXPERIMENTS WITH ALIGNED AND UNALIGNED AGENTS

Now that we have shown how to align agents with users, and how agents can learn optimal action policies, this section empirically examines the ability of various aligned and unaligned agents to deliver utility in practice. In particular, we define four synthetic drivers whose value

functions and alignment relations are easy to establish, and we conduct an experiment to rank the behavior of each in each other's eyes. This yields a four by four matrix of "users" and agents.

We begin by defining four, qualitatively distinct reward functions that are linear in their attribute values. Table 3 associates the reward functions with mnemonic names. Here, the Airport driver is solely motivated by the desire to get to the airport on time, and it becomes less happy as its velocity deviates from target speed. The Safe driver wants to avoid collisions, and its reward function penalizes small times to impact with cars in front and cars behind; the shorter the time to impact, the larger the penalty, while times greater than 100 seconds have no reward. The Goldfish driver has an imaginary fishbowl as luggage, and does not want to upend the fish. Alternatively, we can think of the Goldfish driver as a bit queasy, since its reward function penalizes all forms of maneuver. Finally, the reckless teenager is out for thrills; it garners reward for near misses, but it also cares about maintaining its target speed.

Table 3. A comparison of agent-held reward functions.

| | **Time to impact ahead** | **Time to impact behind** | **Deviation from target speed** | **Slowing down** | **Speeding up** | **Changing lanes** |
|---|---|---|---|---|---|---|
| **Airport Agent** | | | − | | | |
| **Safe Agent** | + | + | | | | |
| **Reckless Agent** | − | − | − | | | |
| **Goldfish Agent** | | | | − | − | − |

We create a set of synthetic drivers by using these reward functions to train the driving program illustrated in Table 1 and Table 2. This results in four agents that possess the exact same set of observations and actions, but that exhibit qualitatively different behavior (Shapiro & Langley,

2002). Note that this behavior emerges without the need to write new code, which is an interesting result from an application perspective. In effect, we have implemented new agents via "programming by reward" (Shapiro, Langley, & Shachter, 2002).

Our experiment evaluated each driver's learned behavior against all four reward functions. This is equivalent to a study of backseat drivers, where we cast, say, the Airport driver as the passenger, and placed it in the back seat of cars piloted by the Safe driver, the Reckless driver, the Goldfish driver, and another Airport driver. We did this with each agent as the passenger, in turn, as shown in the four separate bar graphs of Figure 9.

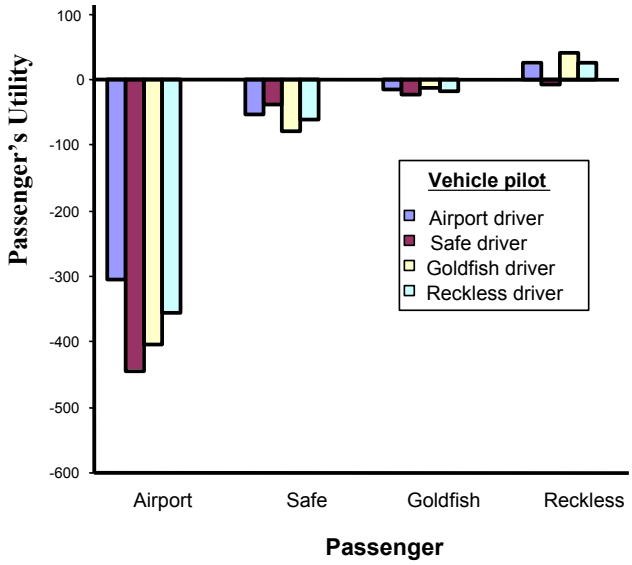


Figure 9 An experiment in backseat driving. Each bar charts identifies the utility experienced by a given driver as a passenger in vehicles piloted by each of the four drivers.

The most obvious feature of the first bar graph is that the Airport driver is happiest as a passenger when another Airport driver is the pilot, because it realizes the greatest (i.e., the least negative) score. The pattern repeats in the second and third bar graphs, when the Safe and Goldfish drivers are passengers. This result makes intuitive sense, since like-minded drivers should approve of one another's behavior. However, the fourth bar graph contains a counterexample, in which the Goldfish driver (as pilot) is apparently better at being Reckless

than the Reckless driver! It is not immediately clear how to account for this reversal. We can also observe a cluster of consistent behavior across the graphs. The Airport, Goldfish, and Reckless drivers appear to "like" each other, while the Safe driver ranks last on all of their lists. This must reflect some compatibility or incompatibility of desires.

We can use the vocabulary of alignment to shed light on these results. In particular, since the agents share the same set of observations and actions they are *at least as capable as* each other (according to the definition in Section 5), and the associated theorems apply. Many of the observed interactions follow from these theorems and an understanding of the driver's alignment relations, which are easy to determine.

Table 4 catalogues the alignment relations among the different drivers, with "users" (or passengers) listed in the rows, and "agents" (vehicle pilots) in the columns. The diagonal elements are clear: since the passenger and pilot have identical rewards, functional alignment holds. The off-diagonal elements require some simple comparisons among reward functions. For example, the Safe pilot will alter its speed to maximize time to impact, but it will be oblivious to the consequences of that action for deviation from target speed, which matters to the Airport passenger. As a result, the Airport passenger is neither structurally nor functionally aligned with the safe pilot (per the Value Alignment theorem). Furthermore, according to the Unaligned Preference proposition, the Airport passenger cannot prefer an unaligned Safe pilot over an aligned and equally capable Airport pilot, as shown within the first bar graph of Figure 9. A similar analysis rules out many other possibilities for alignment, as marked in Table 4.

We can also identify several instances of alignment. In particular, the Reckless pilot cares about a superset of the attributes in the Safe and Airport passenger's reward functions, so it can only effect their utility via attributes it cares about as well. Thus, these drivers are structurally

aligned. However, they are not functionally aligned. The Reckless pilot will sacrifice target speed to experience a near-collision (harming the Airport passenger), and it will act exactly opposite to the Safe passenger's desires because their reward functions consider time to impact with opposite sign. This is an instance where structural alignment provides the power to do harm. Note that the Reckless and Airport drivers both seek to minimize deviation from target speed. Their agreement suggests a positive cross ranking, no matter which is pilot or passenger.

This analysis implies that functionally aligned pilots are preferred to structurally aligned pilots, while structurally aligned ones are preferable to unaligned ones, assuming the sign of effects agree. Thus, as shown in Figure 9, the Airport passenger should prefer the Airport, Reckless, and Safe pilots in that order, and the Safe passenger should prefer the Safe, Airport, and Reckless pilots in that order. Moreover, the Reckless passenger should prefer the Reckless, Airport, and Safe pilots, in that order, owing to sign of effect considerations where alignment fails.

Table 4. Structural alignment (s), functional alignment (f), and the absence of alignment (x), between drivers as passengers (rows) and those same drivers as vehicle pilots (columns.)

| | **Airport Pilot** | **Safe Pilot** | **Reckless Pilot** | **Goldfish Pilot** |
|---|---|---|---|---|
| **Airport Passenger** | f | × | s | s |
| **Safe Passenger** | × | f | s | s |
| **Reckless Passenger** | × | × | f | f |
| **Goldfish Passenger** | × | × | × | f |

We can extend this analysis to include the Goldfish driver by examining its alignment properties. In particular, since the Goldfish driver explicitly cares about all forms of maneuver (including the decision to cruise, by process of elimination), all methods of effecting time to impact and

deviation from target speed are visible in its reward function. This establishes structural alignment between the Goldfish pilot and *all* passengers via the implicit path mentioned in Section 4.2. Note also that the Goldfish driver's desire to avoid maneuver specifies very dangerous behavior, since it will ignore other cars until the Icarus program for driving *forces* it take some action. This commonly occurs in extreme situations where safety is at stake. Thus, the Goldfish pilot will behave against the wishes of the Safe passenger, while its actions will provide sufficient thrill for the Reckless passenger. They may even be functionally aligned, albeit by accident and not design. Given the uncertainties of the training process, the Goldfish pilot can easily be more Reckless than a Reckless driver.

In summary, we have shown that aligned agents deliver higher levels of utility to users than unaligned agents in this simple driving domain, and that our theoretical appreciation of alignment provides a basis for understanding such empirical results. Going further, since we can align agent reward with human utility, and agents can learn action policies that optimize reward, we have established a novel connection between agent behavior and human utility. An aligned agent will do everything it possibly can to maximize human utility as a byproduct of learning to maximize its own reward. Moreover, our experiments suggest that this benefit can be delivered in practice. This supports a novel evaluation metaphor focused on creating and verifying user-agent value alignment, instead of measuring and validating agent behavior.

## 8 RELATED WORK

The principal-agent problem concerns delegation in the absence of trust (Varian, 1992). The theory typically assumes that the principal and agent share a common perceptual base, and thus (in our terms) that their value structures can be functionally aligned by incorporating monetary incentives. More broadly, principal-agent theory is based on the expectation that the agent is

inherently motivated not to address the principal's aims, but rather to pursue its own separate agenda (e.g., to steal all the office furniture). In contrast, our model is deeply cooperative: an aligned agent will do even more for you if only you can communicate your values more exactly. Principal-agent theory provides very little advice on how to surmount the key barrier, which is the gulf between reference frames. Collard's work (1978) on the economics of altruism provides the closest match. He lets one person's utility depend upon another's consumption, or directly upon their utility. If we call the user the 'principal', and the agent the 'agent', we can capture this model by incorporating a user-supplied term into the agent's reward function. Collard's setting, however, is unconcerned with the design of utility functions (which he takes as given) and it assumes no gulf between agent and user perceptions.

We need to look in the literature of the computing sciences to find other formal bridges between agent and user reference frames. Here, the motivation is to place some form of guarantee on artificial agent behavior. Planning systems often promise that execution will produce the desired ends, assuming the action models are accurate, and (typically) that no changes take place to the world outside of agent action. Such results address the soundness and completeness of agent plans taken in isolation. In contrast, very little research strives to explicitly connect agent and user perceptual frames. Rosenschein and Kaelbling (1986) consider the issue in their paper on provable epistemic properties; they assume a relationship between agent knowledge and human concerns, and they show that the agent will never knowingly act in a way that destroys that relationship. (For example, if the agent holds a one in memory whenever a lamp is on the table in the physical world, the agent will never build a plan that it knows will cause that bit to be set to zero, even by inference from its physical actions.) Our work complements this line of reasoning as we engineer the desired relation between agent perceptions and user concerns.

Schoppers and Shapiro (1997) are among the few authors who attempt to build an explicit bridge between agent and human reference frames. They use simultaneous observations to resolve a probabilistic relation between agent and user perceptions of state. Given this relation, and a Markov model that represents agent behavior, they can compute and then ascend the gradient of user utility with respect to design decisions deep within the agent model. Our work preserves this concept of a probabilistic relation between agent and user perceptions, although we use it to cleave the agent design problem: we separate the task of constructing a well-aligned value function from the problems of composing agent skills and finding optimal policies.

Wolpert, New, and Bell (1999) share our interest in constructing agent-held utility functions. However, our goal is to construct a reward function that embodies user concerns, while their work treats reward as a coordination tool; they manipulate and factor the functions passed to multiple agents so that they can learn to achieve against a single, global utility function. This concept of coordination will become relevant as we address multiple agent domains. We note that Wolpert et al.'s work is unconcerned with practical guarantees. While they model an agent's ability to acquire reward with a single number (called its 'intelligence'), we have invented a programming language (Shapiro & Langley, 1999), implemented executable skills (Shapiro, Langley, & Shachter, 2001), and developed a learning algorithm that finds optimal agent policies, given the domain obeys certain Markov assumptions (Shapiro & Shachter, 2000). As a result, our development of user-agent value alignment includes both empirical methodology and theoretical tools.

## 9 DISCUSSION

Given a principal-agent problem, we would like to know if the agent is willing, able, and competent to address the principal's desires. Here, a *willing* agent will choose to pursue the

principal's aims, an *able* agent can represent those aims and know what to do, and a *competent* agent has the skills to perform well in the principal's eyes.

Our framework sheds light on each of these issues. While a human agent generally has to be enticed with incentives to address the principal's desires, artifacts are willing by design. Given that an agent is willing, functional value alignment establishes that it is able. Thus, an aligned agent recognizes all the ways its actions can effect user utility, and it will not knowingly choose actions that harm the principal's interests. This formalizes a popular theme (Asimov, 1950). Finally, the preference propositions rank agents by their competence to deliver user utility. The concept of alignment underlies this capacity.

The value alignment theorem also clarifies several reasons for incentive failures. In particular, structural value alignment fails when agent actions carry unexpected consequences for the principal, or when the agent lacks a sufficient means of representing the principal's utility (either because of poor communication or incommensurate perceptions). When this happens, the optimal policies for the principal and agent can diverge. We can view moral hazard in the same light, as a case where attributes of agent reward function that are not functionally aligned come into play. This will lead the agent to act in its own interests and not the principal's. Finally, the preference propositions expose a new reason for being annoyed at agent behavior: while a skilled, non-aligned agent may perform quite well for us (as principals), we will know the agent had access to better options. In contrast, we are often more tolerant of poor results produced by a good-willed agent with lesser skills, since we know that an aligned agent is doing the best job for us it possibly can. Note that it might be harder to establish alignment with more competent agents because their skills afford more possible effects. This is a somewhat troubling thought.

The concept of alignment raises interesting questions about the design of practical systems. For example, Horvitz, Jacobs, and Hovel (1999) describe an artificial agent that identifies and reacts to email, by taking actions that include discarding a message, paging the user, and augmenting his/her calendar. This agent serves the user's interests given a rich basis of observations and action options. (In theory, it has access to all of the same observations and actions its user can make online). The question is, what does the email reader have to understand about its user in order to represent the user's interests? How perceptive does it have to be to serve the user's needs? With such a rich repertoire, it is indeed a challenge to construct an aligned agent.

In summary, value alignment is a very desirable property because of the power it provides. It supports harmony, it ensures the agent's best efforts, and it creates trust, which in turn enables autonomy. This motivates an empirical question: although we know we can always establish alignment in theory, what will it take to do so in practice? It may or may not be difficult to invent surrogates for user utility, and the complexity of the required assessment process is open to test. However, we know that it *is* plausible to build agents that discover their optimal policy. In particular, our architecture for value-driven agents employs learning to optimize agent-held reward functions, and alignment to relate the learned policies to user objectives. This combination yields a very novel property we call the "Be all you can be" guarantee. It ensures that an aligned agent will do everything it possibly can to maximize human utility, and it tells us this fact before the agent has even been deployed to learn from its environment. It is hard to ask much more.

Curiously, we obtain this level of performance through an act of trust that gives the agent the autonomy to act in our stead. Thus, we can increase our utility by offering our agents the freedom to make value-based choices. This is the art of delegation.

**Acknowledgements**

We thank Derek Ayers, William Bricken, Michael Fehling, Marvin Mandelbaum, and Marcel Schoppers for many discussions on the topic of constructivist agent models that preceded this work.

**References**

Andre, D., & Russell, S. (2001). Programmable reinforcement learning agents. *Advances in Neural Information Processing Systems, 13* (pp. 1019-1025). MIT Press.

Asimov, I. (1950). *I, Robot*. (1950). New York: Grosset & Dunlap.

Collard, D. A. (1978). *Altruism and economy: A study in non-selfish economics*. New York: Oxford University Press.

Dietterich, T.G. (2000). State abstraction in MAXQ hierarchical reinforcement learning. *Advances in Neural Information Processing Systems, 12*. MIT Press.

Heckerman, D., & Shachter, R. (1995). Decision-theoretic foundations for causal reasoning. *Journal of Artificial Intelligence Research, 3*, 405-430.

Horvitz, E., Jacobs, A., & Hovel, D. (1999). Attention-sensitive alerting. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 305-313). San Francisco: Morgan Kaufmann.

Howard, R., & Matheson, J. (1984). *Readings on the principles and applications of decision analysis.* Strategic Decisions Group, Menlo Park, CA.

Parr, R., & Russell, S. (1998). Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems, 10* (pp. 1043-1049). MIT Press.

Rosenschein, S. J., & Kaelbling, L. P. (1986). The synthesis of digital machines with provable epistemic properties. In Halpern, J. Y. (Ed.), *Theoretical aspects of reasoning about knowledge.* Los Altos: Morgan Kaufmann.

Schoppers, M., & Shapiro, D. (1997). Designing embedded agents to optimize end-user objectives. *Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages.* Providence, RI.

Shapiro, D., & Langley, P. (1999). Controlling physical agents through reactive logic programming. *Proceedings of the Third International Conference on Autonomous Agents* (pp. 386-387). Seattle: ACM.

Shapiro, D., & Shachter, R. (2000). *Convergent reinforcement learning algorithms for hierarchical reactive plans.* Unpublished manuscript. Department of Management Science and Engineering, Stanford University, Stanford, CA.

Shapiro, D., & Langley, P., & Shachter, R. (2001) Using background knowledge to speed reinforcement learning. *Fifth International Conference on Autonomous Agents*.

Shapiro, D. (2001). *Value-driven agents.* PhD thesis, Department of Management Science and Engineering, Stanford University, Stanford, CA.

Shapiro, D., & Langley, P., & Shachter, R. (2002) Programming by Reward. *Proceedings of the Nineteenth International Conference on Machine Learning.* (In press.)

Singh, S., Jaakola, T., Littman, M., & Szepesvari, C. (2000). Convergence results for single-step on-policy reinforcement learning algorithms. *Machine Learning*, *38:3* (pp. 287-308).

Sutton, R. S., & Barto, A.G. (1998). *Introduction to reinforcement learning.* Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1998). Intra-option learning about temporally abstract actions. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 556-564). Morgan Kaufmann.

Varian, H. R. (1992). *Microeconomic analysis.* Third edition. New York: W. W. Norton & Company, Inc.

Wolpert, D., New, M., & Bell, A. (1999). *Distorting reward functions to improve reinforcement learning.* Tech. Report IC-99-71, NASA Ames Research Center, Mountain View, CA.