Beyond Minimax: Nonzero-Sum Game Tree Search with Knowledge Oriented Players

Tian Sang and Jiun-Hung Chen
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{sang,jhchen}@cs.washington.edu

The classic minimax search for two-player zero-sum games such as chess has been thoroughly studied since the early years of AI; however for more general nonzero-sum games, minimax is non-optimal, given a player's knowledge about the opponent. Previously, a few opponent models and algorithms such as M^* were introduced to improve minimax by simulating the opponent's search, given the opponent's strategy. Unfortunately, they all assume that one player knows the other's strategy but not vice versa, which is a very special relationship between knowledge and strategies. In this paper, we characterize minimax and M^* and show examples where they can be non-optimal. We propose a new theoretical framework of Knowledge Oriented Players (KOP), using the powerful S5 axiom system to model and reason about players' knowledge. With KOP, the general relationship between knowledge and strategies can be analyzed, which is not handled by the traditional game theory. We show how strategies are constrained by knowledge and present new strategies for various problem settings. We also show how to achieve desired knowledge update via communication so that players can achieve the best possible outcome. With respect to players' knowledge, our strategies always dominate minimax and work well for the general knowledge cases where previous algorithms do not apply.

Introduction

Minimax search on two-player game trees (Shannon 1950) has been a classic topic and thoroughly studied since the early years of AI. Assuming players have perfect information of game trees and play perfectly, minimax is the optimal strategy in zero-sum games including chess, where both players use the same evaluation function for all leaf nodes. It has been one of the bases for the hugely successful computer chess programs, and eventually IBM Deep Blue defeated the human world chess champion Kasparov in 1997.

However, minimax is no longer optimal in more general nonzero-sum games, where players can have different evaluation functions that are not necessarily correlated. For example, a leaf good for one player is not necessarily good or bad for the other. Intuitively if a player knows the opponent's evaluation function, he could potentially do better by taking into account the opponent's possible responses. This is done in the Opponent Models (OM) presented in (Carmel & Markovitch 1993; Iida *et al.* 1993). It is assumed that player A uses the evaluation function E_A while B uses E_B, and A knows that B uses E_B but not vice versa. So B plays minimax using E_B, but A plays OM search, which takes advantage of knowing B's exact response to every possible move by A. Given that assumption, OM search dominates minimax for A.

M^* search (Carmel & Markovitch 1993; 1996) is a recursive extension of OM search, in which each player has his own OM of the other but A's OM strictly contains B's OM. For example, both A and B know the initial strategy of each other but A also knows that B knows A's initial strategy, so A knows that B will adjust his strategy to take advantage of A's initial strategy, and thus A will adjust his initial strategy as well to take advantage of B's behavior. Their opponent models can be nested to any level, but A always knows B's OM so that A can use M^* search, which accurately predicts B's responses and thus is at least as good as minimax.

Unfortunately, both OM search and M^* search assume knowledge asymmetry: A's knowledge strictly subsumes B's knowledge; in particular, they assume that A has correct knowledge about B's strategy while B has no knowledge or wrong knowledge about A's strategy, which is a very strong and "unfair" assumption not necessarily true in general. These issues were partly addressed in (Donkers 2004), in which the author suggested considering the opponent models with knowledge symmetry and showed an example in chess where with knowledge symmetry two players could reach a state of common interests, though the notion of knowledge symmetry was fairly vague.

In this paper, we propose a new theoretical framework of Knowledge Oriented Players (KOP), capable of modeling both knowledge asymmetry and knowledge symmetry by the powerful $S5_n$ axiom system. With KOP, we are able to define and analyze the general relationship between two key characteristics of players– knowledge and strategies, in finite-horizon two-player nonzero-sum games. As an important step forward, we show how strategies are constrained by knowledge. We present new strategies–M^h and HM^0M^d, for various problem settings. They dominate minimax with respect to players' knowledge and are much more general

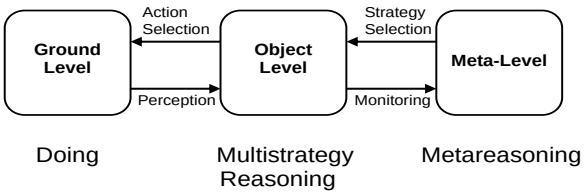

Figure 1: The high level playing processes of a player.

than M^* search. We also propose communication schemes that can achieve desired knowledge update, and show how they can help players select better strategies.

Figure 1 (Cox & Raja 2007) can be used to illustrate the high level playing processes of a player in nonzero-sum games. At the meta-level, the player needs to do metareasoning to select his best strategies, based on his knowledge about the possible outcomes, both players' preferences of the outcomes, and both players' knowledge about each other's knowledge, etc. At the object level, actions (moves) are selected for the current game state, based on the selected strategies. At the ground level, actions (moves) are executed; the corresponding effects/outcomes are observed and passed to the object level, which are further interpreted/evaluated and passed to the meta-level for knowledge/strategy update.

In the rest of the paper, first we describe minimax and M^* and illustrate their differences by example, then we introduce our KOP framework, new algorithms, communication schemes and analytical results, and finally we conclude.

Minimax and M^* Algorithms

For ease of understanding, we will describe the minimax (Shannon 1950) and the M^* (Carmel & Markovitch 1996) algorithms in a bottom-up fashion, which is equivalent to the top-down description and can be implemented as depth-first search. We assume that both players know the game tree perfectly but have different evaluation functions for the leaf nodes, which are the possible outcomes of the game. Throughout this paper we will use the following notations.

Objects:

t: root of a finite game tree, also used to represent the tree.
n: a node of t. At a non-leaf node, we know who moves.
l: a leaf node of t, as a possible outcome of the game.
A: max-player who maximizes the evaluation function E_A.
B: min-player who minimizes the evaluation function E_B.

Function values (references that can be updated):

E_A, E_B: real-valued evaluations for leaves. Without loss of generality, we assume that there is a fixed deterministic tie-break scheme reflected in leaf values, so we have distinct leaf values: $\forall l \neq l', E_i(l) \neq E_i(l'), i \in \{A, B\}$.
$C(n)$: set of pointers to the children of n.
$height(n)$: height of node n. $height(l) = 0$ for leaf l; otherwise, $height(n) = 1 + \max_c height(c), c \in C(n)$.
$next(n)$: pointer to a child of n, or null for a leaf.
$value(n)$: value of node n, which is a pair (a, b) where a and b are the evaluations to A and B respectively.

Definition 1 *A **strategy** S is an algorithm that given a game tree t, $\forall n \in t$, S computes $next(n)$ and $value(n)$ and maintains an invariant for all non-leaf n:*
$(\exists c \in C(n), next(n) = c) \ \wedge \ value(n) = value(next(n))$

According to the estimate by a given a strategy S, at node n the move is $next(n)$ and the outcome is $value(n)$, so the first move is $next(t)$ and the final outcome is $value(t)$.

The Minimax Algorithm

Algorithm 1 Minimax

```
minimax(t, E_A)
for every leaf l
   next(l) := null; value(l) := (E_A(l), E_A(l))
for h = 1 to height(t)
   for every node n with height h
      if A moves at n     // to maximize
         next(n) := argmax_c value(c).a, c ∈ C(n)
      else    // B moves, to minimize
         next(n) := argmin_c value(c).a, c ∈ C(n)
      value(n) := value(next(n))
```

Algorithm 1 is minimax, where A uses E_A to maximize and assumes that B uses E_A to minimize. We assume that all nodes with the same height can be accessed sequentially.

Definition 2 *A **play** for players A and B on a game tree t where A moves first is a function that maps the input (t, E_A, E_B, S_A, S_B) to a leaf value, where S_A and S_B are strategies for A and B. That is, **play**$(t, E_A, E_B, S_A, S_B) = (a, b)$, where $a = E_A(l)$, $b = E_B(l), l \in t$.*

How *play* works is trivial: starting from the root, at node n it follows $next(n)$ by S_A if A moves otherwise $next(n)$ by S_B and finally it reaches a leaf and returns the value of it.

In a zero-sum game with $E_A = E_B$, it is well-known that if it is a perfect information game and both A and B play perfectly, minimax is optimal in terms of Subgame Perfect Equilibrium (SPE) in game theory (Kuhn 1953). That means both players will stay with minimax to get the best outcome.

$$\forall t, E_A, S_A, play(t, E_A, E_B, minimax, minimax).a \geq play(t, E_A, E_B, S_A, minimax).a \quad (1)$$

$$\forall t, E_A, S_B, play(t, E_A, E_B, minimax, minimax).b \leq play(t, E_A, E_B, minimax, S_B).b \quad (2)$$

In a nonzero-sum game with $E_A \neq E_B$, minimax is no longer optimal, because it wrongly assumes that both players use the same evaluation function. Nonetheless, A's minimax does guarantee the worst case outcome for A, because it proceeds as if B would always choose the worst possible moves against A. Therefore, minimax is used as the baseline for comparisons in our examples. More generally, we consider imperfect information nonzero-sum games, in which players can have incomplete mutual knowledge and thus SPE does not apply. A player's best strategy often depends on his knowledge about the opponent, so we describe the M^* algorithm for nonzero-sum games next.

The M^* Algorithm

Algorithm 2 is equivalent to the recursive M^* algorithm described in (Carmel & Markovitch 1996). We often refer

Algorithm 2 M^*

$M^*(t, E_A, E_B, caller, d)$ // requires $d \geq 0$
if ($caller = A$) XOR (d is even) // base case: A's minimax
  $minimax(t, E_A)$; $baseA$:= true
else // base case: B's minimax
  $minimax(t, E_B)$; $baseA$:= false
if $d \geq 1$
  for every leaf l // update leaf values for simulation
    $value(l) := (E_A(l), E_B(l))$
for $i := 1$ to d // run alternated simulations for A and B
  for $h := i$ to $height(t)$
    for every node n with height h
      if (A moves at n) and (i is odd XOR $baseA$)
      // now at max-node in simulation for A
        $next(n) := argmax_c\ value(c).a,\ \ c \in C(n)$
      else if (B moves at n) and (i is even XOR $baseA$)
      // now at min-node in simulation for B
        $next(n) := argmin_c\ value(c).b,\ \ c \in C(n)$
      $value(n) := value(next(n))$

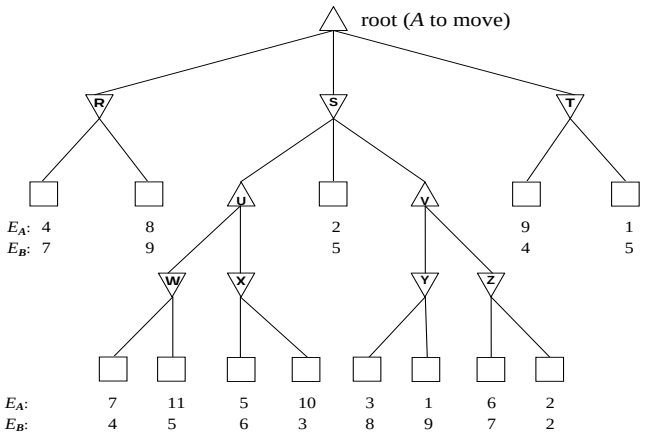

Figure 2: A nonzero-sum game tree. Max-player A tries to maximize E_A and min-player B tries to minimize E_B.

to $M^*(..., d)$ by M^d, OM search by M^1 and minimax by M^0. M^* assumes knowledge asymmetry: A's OM contains B's OM, so A playing M^d means B playing M^{d-1} and the outcome is $play(t, E_A, E_B, M^d, M^{d-1})$. To compute it, M^* needs a series of simulations: A's M^d, B's M^{d-1}, A's M^{d-2}, ..., A's M^0 or B's M^0 (depending on the caller XOR d's parity). Therefore, algorithm 2 runs alternated bottom-up simulations from M^0 to M^{d-1}. M^* tries to take advantage of the opponent's strategy, and it dominates minimax for A under certain conditions, as stated in (Carmel & Markovitch 1996), which can be rephrased as:

$$\forall t, E_A, E_B, d \geq 1, play(t, E_A, E_B, M^d, M^{d-1}).a \geq play(t, E_A, E_B, minimax, M^{d-1}).a \quad (3)$$

Limitations of M^* (I) In essence, M^* is the same as OM search, because they all assume that A knows B's strategy but not vice versa, no matter how complicated the strategy can be. Therefore in M^*, B's strategy does not have to be the recursive form of minimax at all. (II) The assumption of M^* is too strong. Proposition 3 does not guarantee any bound for B, so why would B even use such a strategy? Instead, it is quite possible for B to use an alternative strategy that we will show later, which is guaranteed to be at least as good as minimax according to B's knowledge.

An Illustrative Example of Nonzero-sum Games

The following example is used to illustrate how minimax (M^0), OM search (M^1) and M^* search work. Both players know the game tree and their own evaluation function E_A or E_B. We assume that E_A and E_B are atomic: either fully known or unknown, and players use $M^d, d \geq 0$.

In Figure 2, triangles represent max-nodes where max-player A moves, inverted triangles represent min-nodes where min-player B moves, and squares represent leaves. A and B have different evaluation functions E_A and E_B for leaves. A leaf is referenced by a pair of its E_A and E_B values, while an internal node is referenced by its label. A's goal is to maximize E_A of the outcome and B's goal is to minimize E_B of it. We consider the following scenarios, in which players have different imperfect mutual knowledge.

(1) *A does not know E_B and B does not know E_A.* Since A has no idea of E_B, A uses minimax (M^0) against the worst case, assuming B plays minimax using E_A too (which is wrong). By A's minimax from the root, the R branch yields $E_A = 4$, the S branch yields $E_A \leq 2$, and the T branch yields $E_A = 1$. So A moves to R. At R, B moves to (4,7), which B prefers to (8,9). The final outcome is (4,7).

(2) *A knows E_B and B does not know E_A.* Without idea of E_A, B uses M^0 with E_B. Knowing E_B, A uses M^1 that takes into account B's M^0. At the root, A can choose to:

(2.1) move to R. A knows that B will choose (4,7) at R.

(2.2) move to T. A knows that B will choose (9,4) at T.

(2.3) move to S. To determine B's response at S, A simulates B's M^0: for B, the values are W–(7,4), X–(10,3), Y–(8,3), Z–(2,2), U–(7,4), V–(3,8) and S–(7,4). So A knows that at S B will move to U, and at U A will move to X because A knows that if he moves to W then B will move to (7,4), not preferred by A. At X, B will move to (10,3).

Considering all three choices, A moves from the root to S, which is the best for A. The final outcome is (10,3).

(3) *A knows E_B and B knows E_A.* Since A's knowledge is the same as in (2), A does the same reasoning and still moves to S. Knowing E_A, B now uses M^1 (instead of M^0 as in (2)) that takes into account A's M^0. From B's point of view, according to A's M^0, some node values are Y–(1,9), Z–(2,2) and V–(2,2). So B moves from S to V (out of expectation by A in (2)), expecting to reach Z and then (2,2) that has the best E_B value. At V, however, A moves to Y instead of Z (out of expectation by B), expecting to reach (3,8), because A knows if he moves to Z then B will move to (2,2), bad for A. At Y, B moves to (3,8), and the final outcome is (3,8), which is neither A's nor B's initial expectation.

(4) *A knows E_B, B knows E_A and A knows that B knows E_A.* Again A's knowledge subsumes B's knowledge, so as in (2), A can correctly simulate and predict B's moves. Us-

ing M^2 that takes into account B's M^1, A knows that if he moves to S, he will finally reach (3,8) as in (3), instead of (10,3) in (2); so A moves to T for (9,4), which is the best achievable outcome given B's knowledge. Since at T B prefers (9,4) to (1,5), the final outcome is (9,4).

Summary of Results and Motivation We use a shorthand of *play* (See Definition 2) with only strategy parameters. The outcomes of the scenarios are: (1) $play(M^0, M^0) = (4, 7)$; (2) $play(M^1, M^0) = (10, 3)$; (3) $play(M^1, M^1) = (3, 8)$; (4) $play(M^0, M^0) = (9, 4)$. The outcome value of (1) by minimax is the baseline. Scenario (2) and (4) are better than (1) for A—consistent with proposition 3.

However, scenario (3) is worse than (1) for both A and B, even though they have more knowledge in (3) than in (1). That violates our intuition "the more knowledge the better outcome", because M^* does not guarantee any bound of the outcomes when there is no strict knowledge asymmetry; yet we must have working strategies for all the cases and we must model the complicated knowledge scenarios (including knowledge about evaluation functions, strategies and knowledge about knowledge, etc.) in a rigorous way that can be reasoned about. This is our motivation for a new general framework that models players with knowledge.

The Framework of Knowledge Oriented Players (KOP)

We propose a new framework of Knowledge Oriented Players (KOP), which includes all the elements that determine the outcomes of finite-horizon nonzero-sum game trees and allows us to reason about knowledge and strategies. We should emphasize that since the players' mutual knowledge can be imperfect in general, the notion of SPE for perfect information games in game theory (Kuhn 1953; Fudenberg & Tirole 1991) does not apply here in general.

Definition 3 *A **Knowledge Oriented Player (KOP)** P is a 3-tuple $(E_P, S_P, \mathcal{K}(P))$, where E_P is the evaluation function, S_P is the strategy and $\mathcal{K}(P)$ is the knowledge of P represented by a set of formulas, that is, $\mathcal{K}(P) = \{\varphi | K_P\varphi\}$ ($K_P\varphi$ is read as "P knows φ"). $\mathcal{K}(P)$ is required to be a maximal consistent set in the $S5_n$ axiom system.*

The $S5_n$ axiom system is sound and complete, whose main axioms are listed below (with $n = 2$ and $P = A, B$). For more details, please refer to (Halpern & Moses 1992).

A1. All tautologies of the propositional calculus.
A2. $(K_P\varphi \wedge K_P(\varphi \Rightarrow \psi)) \Rightarrow K_P\psi$ (deduction).
A3. $K_P\varphi \Rightarrow \varphi$ (the knowledge axiom).
A4. $K_P\varphi \Rightarrow K_PK_P\varphi$ (positive introspection).
A5. $\neg K_P\varphi \Rightarrow K_P\neg K_P\varphi$ (negative introspection).

We use $K_P\varphi$ as a proposition for "$\varphi \in \mathcal{K}(P)$". We have $K_Ptrue \wedge \neg K_Pfalse$ by axioms. We assume (a) each player P is a KOP; (b) E_P and S_P are atoms that can be fully known or unknown: $K_PE_P \leftrightarrow \forall t \forall l \in t \forall r \in Real, (K_P(E_P(l) = r) \vee K_P(E_P(l) \neq r)))$ and $K_PS_P \leftrightarrow \forall t \forall n \in t, (C(n) = \emptyset) \vee (\forall c \in C(n), K_P(next(n)_{S_P} = c) \vee K_P(next(n)_{S_P} \neq c)))$; (c) $K_PE_P \wedge K_PS_P$.

Common knowledge is incorporated in KOP. A formula φ is common knowledge iff everybody knows φ, everybody knows that everybody knows φ, and so on (van Benthem, van Eijck, & Kooi 2005). Any common knowledge φ is just viewed as part of tautologies included in axiom A1.

Definition 4 *A formula set F is a **core** of $\mathcal{K}(P)$ iff F can be extended to the maximal consistent set $\mathcal{K}(P)$ by using the axioms of $S5_n$; **cores**-$\mathcal{K}(P)$ is the set of all cores of $\mathcal{K}(P)$.*

By definition 4, a potentially infinite $\mathcal{K}(P)$ can be represented by its *core* F, which can be small but not necessarily minimal, though $\mathcal{K}(P)$ may have many minimal cores.

Definition 5 *A strategy S is **t-selfish** for player P on game tree t, iff $\forall n \in t$ where P moves at n, if P is a max-player $value(n).a = max_c value(c).a$; otherwise $value(n).b = min_c value(c).b$, $c \in C(n)$. S is **selfish** for P iff $\forall t$, S is t-selfish for P; P is **selfish** iff P always uses selfish strategies.*

Naturally, players are always assumed to be *selfish* and they have common knowledge that everybody is *selfish*. Although we only discuss two-player games in this paper, KOP works for multi-player games and most of our results can be extended to multi-player games .

Definition 6 *Given selfish players A, B, a game tree t and $n \in t$, (A, B, t) $\mathcal{K}$-**decides** n, iff n is a leaf, or A and B agree on $next(n)$ and $value(n)$, by $\mathcal{K}(A)$ and $\mathcal{K}(B)$.*

We use "n is decided" as a shorthand for (A, B, t) $\mathcal{K}$-*decides* n. Intuitively, there can be some decided nodes (i.e. game states) in the game tree such that both players agree on what will happen afterwards once the game is in one of these states . Players can reason about decided nodes by their knowledge. Suppose A moves at n, with leaves l_1 and l_2 as n's children. Since A is selfish, A chooses l_1 if $E_A(l_1) \geq E_A(l_2)$; otherwise l_2. So given t and E_A, n is decided for A, but whether n is decided for B depends on B's knowledge: B will be certain of A's choice if B knows E_A, i.e., K_BE_A. Similarly, consider n's parent p where B moves and has another choice n' with the same height as n: p is decided if n and n' are decided plus (1) B knows where to move at p (trivial for B because B just chooses the one with smaller E_B value) and (2) A knows where B moves at p, so the necessary knowledge for A is that A knows that B knows that n and n' are decided, that is equivalent to $K_AK_BE_A$.

The same reasoning applies to the nodes all the way up the tree. Therefore, we can derive the following algorithm.

The Decide Algorithm

Algorithm 3 computes whether n is decided, and $next(n)$ and $value(n)$ for decided n, by checking a class of knowledge: if $K_AE_B \wedge K_BE_A$ then all the nodes with height one are decided; if $K_AE_B \wedge K_BE_A \wedge K_BK_AE_B \wedge K_AK_BE_A$ then all the nodes with height two are decided, and so on.

Levels of Knowledge–$f(A, d)$ Next, we directly construct such a set f that includes the desired class of formulas in regular expressions, given a level parameter $d \geq 0$:

$$f(A, 0) = \{K_AE_A, K_AS_A\};$$
$$f(A, d) = \{K_A(K_BK_A)^{\lfloor \frac{(d'-1)}{2} \rfloor}E_B | \ 1 \leq d' \leq d\}$$
$$\cup \{(K_AK_B)^{\lfloor \frac{d'}{2} \rfloor}E_A | \ 2 \leq d' \leq d\} \cup f(A, 0) \quad (4)$$

Algorithm 3 Decide

$decide(A, B, t, n)$
// returns true or false, and computes $next(n)$ and $value(n)$
if height(n) = 0 return true // leaves are don't cares
if height(n) = 1
  if A moves at $n \wedge K_B E_A$
    $next(n) := argmax_c\ value(c).a, c \in C(n)$
  else if B moves at $n \wedge K_A E_B$
    $next(n) := argmin_c\ value(c).b, c \in C(n)$
  else return false // negative base case
  $value(n) := value(next(n))$
  return true // positive base case
if $\exists c \in C(n), \neg decide(A, B, t, c)$
  return false // not all children decided
if n has only one child c
  $next(n) := c$; $value(n) := value(c)$
  return true // the only child is decided
if $\forall c \in C(n)$, A moves at $n \wedge K_B(decide(A, B, t, c))$
  $next(n) := argmax_c\ value(c).a, c \in C(n)$
else if $\forall c \in C(n)$, B moves at $n \wedge K_A(decide(A, B, t, c))$
  $next(n) := argin_c\ value(c).b, c \in C(n)$
else return false // negative recursive case
$value(n) := value(next(n))$
return true // positive recursive case

Similarly $f(B, d)$ can be defined by switching A and B. For the different scenarios of the example in Figure 2, the *cores* of knowledge can be expressed as follows:

(1) A: $f(A, 0) = \{K_A E_A, K_A S_A\}$
  B: $f(B, 0) = \{K_B E_B, K_B S_B\}$
(2) A: $f(A, 1) = f(A, 0) \cup \{K_A E_B\}$, B: $f(B, 0)$
(3) A: $f(A, 1)$, B: $f(B, 1) = f(B, 0) \cup \{K_B E_A\}$
(4) A: $f(A, 2) = f(A, 1) \cup \{K_A K_B E_A\}$, B: $f(B, 1)$

By our construction, $f(A, d)$ has some nice properties, for examples, the following properties can be verified: $\forall d \geq 1$,

$$f(A, d-1) \subseteq f(A, d) \wedge f(B, d-1) \subseteq f(B, d) \quad (5)$$
$$f(A, d) \subseteq \mathcal{K}(A) \rightarrow f(B, d-1) \subseteq \mathcal{K}(B) \quad (6)$$
$$f(A, d) \in \textit{cores-}\mathcal{K}(A) \rightarrow f(A, d-1) \notin \textit{cores-}\mathcal{K}(A) \quad (7)$$

Property 5 is trivial. Property 6 holds because the first K_A of the formulas in $f(A, d)$ can be removed by axiom A3, then we get formulas in $f(B, d-1)$. Axiom A1-A5 cannot increase alternations of K and the largest number alternations of K in $f(A, d)$ is d: the corresponding formula cannot be extended from $f(A, d-1)$, so property 7 holds.

Using "levels of knowledge", we are able to give some formal results of algorithm 3, stated as theorem 1.

Theorem 1 *For algorithm 3, $\forall A, B, t, n \in t, d \geq 1$,*
1.1 If $decide(A, B, t, n)$ returns true, then (A, B, t) $\mathcal{K}$-decides n and $decide(A, B, t, n)$ computes the $next(n)$ and $value(n)$ that A and B agree on.
1.2 If $f(A, d) \subseteq \mathcal{K}(A) \wedge f(B, d) \subseteq \mathcal{K}(B) \wedge height(n) \leq d$, then $decide(A, B, t, n)$ returns true.
1.3 If $(\exists d \geq 1, f(A, d) \in$ cores-$\mathcal{K}(A))$, then $decide(A, B, t, n)$ returns true iff (A, B, t) $\mathcal{K}$-decides n.

Proof sketch: By 1.1, *decide* is sound; this holds because it exactly follows the reasoning process for decided nodes, as described before. 1.2 can be proved by mathematical induction: $n = 1$ is trivial; assume it holds for $n = m$, for $n = m + 1$, $f(P, m + 1)$ has one formula with $m + 1$ alternations of K, which makes the knowledge check in *decide* true, so it returns true. By 1.3, *decide* is sound and complete if $f(P, d)$ is a *core* of $\mathcal{K}(A)$; this holds because when A has no extra knowledge for reasoning, the only nodes that A and B can agree on are those decided by the alternations of K and those with single decided child, as done in *decide*.

Definition 7 *A strategy S is **t-informed** for player P on game tree t, iff $\forall n \in t$, if (A, B, t) $\mathcal{K}$-decides n then S computes the $next(n)$ and $value(n)$ that A and B agree on. S is **informed** for P iff $\forall t$, S is t-informed for P; P is **informed** iff P always uses informed strategies.*

Players are always assumed to be *informed* and have common knowledge about that fact. Therefore a player's strategy is always constrained by his knowledge, that is, his strategy has to be *selfish* and *informed* for him if he is rational.

Definition 8 *Given selfish and informed players A and B, a game tree t where A moves first, and strategies S and S', S $\mathcal{K}$-**dominates** S' on t for A, iff by $\mathcal{K}(A)$ A knows that $play(t, E_A, E_B, S, S_B).a \geq play(t, E_A, E_B, S', S_B).a$. S $\mathcal{K}$-**dominates** S' for A, iff by $\mathcal{K}(A)$ A knows that S $\mathcal{K}$-**dominates** S' on t for all t where A moves first.*

In definition 8, according to $\mathcal{K}(A)$ A knows that S_B is constrained by $\mathcal{K}(B)$ in a way that using S is better than S' for A, in contrast to unconstrained S_B.

The M^h and Hybrid-minimax-M^h Algorithms

In this section we will present two algorithms that $\mathcal{K}$-*dominate* minimax, given players' complete or incomplete knowledge. Furthermore, we show how communication can help update knowledge from incomplete to complete for better outcomes. The new algorithms work well for the general knowledge scenarios (e.g., case (3) in Figure 1) where M^* does not apply. First we study the case where every node is decided and thus the *decide* algorithm can be simplified to the M^h algorithm.

The M^h Algorithm

Algorithm 4 M^h

$M^h(t, E_A, E_B)$ // computes all $next(n)$ and $value(n)$
for every leaf l
  $next(l) := null$; $value(l) := (E_A(l), E_B(l))$
for h = 1 to $height(t)$
  for every node n with height h
    if A moves at n // to maximize
      $next(n) := argmax_c\ value(c).a, c \in C(n)$
    else // B moves, to minimize
      $next(n) := argmin_c\ value(c).b, c \in C(n)$
    $value(n) := value(next(n))$

Compared to minimax, M^h has another parameter E_B, which enables it to determine the true leaf values for B, rather than assuming $E_A = E_B$. M^h correctly determines

both player's optimal moves if they have enough knowledge about their evaluation functions. We can view M^h as the nonzero-sum version of minimax or $\mathcal{K}$-minimax: M^h determines the best outcome for *selfish* and *informed* players with respect to their knowledge.

Theorem 2 *Given selfish and informed players A, B, and a game tree t, $\forall n \in t$, if $decide(A, B, t, n)$ returns true, then M^h computes the same $next(n)$ and $value(n)$ as $decide(A, B, t, n)$.*

Proof: The M^h algorithm simplifies the *decide* algorithm in the way that all the conditions involving knowledge check are set to true, so they compute the same results. □

By theorem 2, the M^h algorithm can be viewed as a special case of algorithm 3. However, we must emphasize that algorithm 3 itself is not a *strategy* (see definition 1), because it may not compute $next(n)$ and $value(n)$ for every node n; while M^h is a *strategy* because it computes $next(n)$ and $value(n)$ for every n.

Theorem 3 *M^h and M^* computes the same results for all game trees iff M^* has the parameter $d = \infty$.*

Proof: M^* with parameter $d = \infty$ corresponds to the case that both players simulate each other for an infinite number of levels, therefore, for any given depth tree t, the simulation process converges to M^h, which can be verified by comparing algorithm 2 and algorithm 4. □

Even though M^* can converge to M^h under certain conditions, they are still quite different algorithms in general.

Differences of M^* and M^h (I) M^* requires expensive recursive simulations while M^h does not; consequently, in (Carmel & Markovitch 1996) the multi-pass M^* algorithm needs to evaluate a node multiple times (compared to one evaluation per node in M^h), and the one-pass M^* algorithm needs to compute vector values (compared to single value computation in M^h). (II) M^* represents a family of algorithms by parameter d, which relies on special assumptions about the opponent's strategies; while M^h does not need such assumptions. The conditions under which M^h works will be elaborated in the following theorems.

Definition 9 *A leaf l of t is the **global optimum** of t, iff* $\forall l' \in t, (l' \neq l) \rightarrow (E_A(l) > E_A(l') \wedge E_B(l) < E_B(l'))$

Any game tree has at most one *global optimum* because of our assumption of distinct leaf values that reflect a fixed deterministic tie-break scheme, while some game trees may not have any *global optimum* at all. It is worth knowing that when there exists a *global optimum*, whether a strategy always finds it. Surprisingly, although M^h can find the *global optimum*, if there is any, minimax and M^* are not guaranteed to do so, as stated in theorem 4.

Theorem 4 *If $l \in t$ is the global optimum of t, M^h finds it: $M^h(t, E_A, E_B)$ yields $value(t) = (E_A(l), E_B(l))$; however, minimax and M^* do not guarantee $value(t) = (E_A(l), E_B(l))$.*

Proof: M^h always has $next(p) = l$ for l's parent p no matter who moves at p because l is the best choice for both. Similarly p's parent has its next pointer to p, and so on. Therefore, $value(l)$ is passed all the way up to the root, that is, $value(t) = value(l)$. In minimax, one player is not even aware of the other's evaluation function, so there is no way for him to recognize the *global optimum*, let alone to find it. Similarly, since M^* is a recursive simulation process of minimax, it is also possible to miss the *global optimum*. □

Corollary 1 *If t has a global optimum leaf l and both players use M^h, $play(t, E_A, E_B, M^h, M^h) = (E_A(l), E_B(l))$.*

Corollary 1 holds because when both players use M^h, they agree on the final outcome $value(l)$, which is the same global optimum computed by M^h.

Corollary 2 *If E_A and E_B are uncorrelated and unbounded, in general there does not exist a sound and complete pruning algorithm that does not check every leaf.*

Corollary 2 holds because any leaf not checked by a pruning algorithm could be a global optimum, given unconstrained E_A and E_B. This is a bad news for any general pruning attempt—to be able to do pruning, one has to assume correlations between E_A and E_B, as done in (Carmel & Markovitch 1996).

The following theorem reveals the key property of M^h.

Theorem 5 *M^h dominates any strategy if the opponent uses M^h: $\forall t, E_A, E_B, S_A, play(t, E_A, E_B, M^h, M^h).a \geq play(t, E_A, E_B, S_A, M^h).a$ (8)*
$\forall t, E_A, E_B, S_B, play(t, E_A, E_B, M^h, M^h).b \leq play(t, E_A, E_B, M^h, S_B).b$ (9)

Proof: For (8), let $S_A \neq M^h$, $play(t, E_A, E_B, S_A, M^h)$ follows a path p of next pointers from the root to a leaf, and $play(t, E_A, E_B, M^h, M^h)$ follows another path $q \neq p$. Consider the node n with smallest height where p deviates from q, B does not move at n because B uses the same $next(n)$ by M^h for two paths, so A moves at n. Since $next(n)$ of M^h points to n's child with best value to A, it must be that S_A's $next(n)$ points to a child with worse value; thus S_A can be improved by changing its $next(n)$ to that of M^h for a better outcome. This improving process can be repeated until p matches q, which means q leads to a better outcome than p. Because of symmetry, (9) holds. □

Since M^h can be also viewed as backward induction (Kuhn 1953), it is not surprising that propositions (8) and (9) about M^h are identical to propositions (1) and (2) about minimax. By theorem 5, using M^h is mutually-enforced by both players in nonzero-sum games, just like what happens to using minimax in zero-sum games.

Then it is important to show under what knowledge scenarios M^h is the optimal strategy.

Theorem 6 *For any game tree t where A moves first, if $height(t) \leq 1 \vee f(A, height(t) - 1) \subseteq \mathcal{K}(A)$, then M^h $\mathcal{K}$-dominates any other strategy on t for A.*

Proof: The base case of $height(t) \leq 1$ is trivial, because with M^h A chooses the best move to a leaf according to E_A. Given $f(A, height(t) - 1) \subseteq \mathcal{K}(A)$, by property 6 we have $f(B, height(t) - 2) \subseteq \mathcal{K}(B)$, and then all n with $height(n) \leq height(t) - 2$ are decided by theorem 1.2. Given B is selfish and informed, B's strategy computes the same $next(n)$ and $value(n)$ as M^h does for every n with

$height(n) \leq height(t) - 2$; so by theorem 5, for these n, A's strategy has to match M^h or it can be improved. Because B moves at every n with $height(n) = height(t) - 1$, B's move is decided for B and A knows that exactly by $\mathcal{K}(A)$; therefore A knows that A's first move at the root has to match that of M^h too, or it can be improved. Overall A's strategy has to match M^h, or it can be improved. □

By theorem 6, if a player has "enough" knowledge then he knows that M^h is the optimal strategy. Common knowledge is more than enough, so the following corollary holds.

Corollary 3 *If both A and B have common knowledge about E_A and E_B, that is $\forall d \geq 1, f(A,d) \subseteq \mathcal{K}(A)$, then M^h $\mathcal{K}$-dominates any strategy for both A and B.*

Intuitively the more knowledge a player has the better and more accurate outcome he can estimate. When the conditions in theorem 6 or in corollary 3 are met, certainly A will estimate the outcome by M^h. We claim that this estimated outcome is at least as good as the one estimated by minimax using only E_A when A knows nothing about E_B.

Theorem 7 *For any game tree t where A moves first, let $v_1 = value(t)$ computed by $M^h(t, E_A, E_B)$ and $v_2 = value(t)$ computed by $minimax(t, E_A)$, then $v_1.a \geq v_2.a$.*

Proof: At each node where B moves, minimax (as in algorithm 1) always let B choose the child with smallest E_A value; whereas M^h let B choose the child with smallest E_B value. Unless the same child is chosen, M^h always chooses a child better for A, compared to what minimax chooses, so M^h yields a better estimated final outcome for A. □

When player A does not have enough knowledge about B, the outcome estimated by M^h can be inaccurate, because B may not be aware that B should play M^h according to B's knowledge. To solve this problem, we introduce the hybrid-minimax-M^h algorithm (algorithm 5), which works well with incomplete knowledge.

The Hybrid-minimax-M^h Algorithm

Algorithm 5 Hybrid-minimax-M^h

Hybrid-minimax-$M^h(t, E_A, E_B, d)$ // requires $d \geq 0$
for every leaf l
  $next(l) := null$; $value(l) := (E_A(l), E_B(l))$
for h = 1 to d
  for every node n with height h
    if A moves at n // to maximize
      $next(n) := argmax_c\ value(c).a, c \in C(n)$
    else // B moves, to minimize
      $next(n) := argmin_c\ value(c).b, c \in C(n)$
    $value(n) := value(next(n))$
for $h = d + 1$ to $height(t)$
  for every node n with height h
    if max-player moves at n // to maximize
      $next(n) := argmax_c\ value(c).a, c \in C(n)$
    else // min-player moves, to minimize
      $next(n) := argmin_c\ value(c).a, c \in C(n)$
    $value(n) := value(next(n))$

Next we abbreviate Hybrid-minimax-$M^h(t, E_A, E_B, d)$ in algorithm 5 as HM^0M^d. It can be viewed as a combination of minimax (M^0) and M^h with parameter d: it reduces to A's M^0 (ignoring unused E_B) if $d = 0$; for a given t, it reduces to M^h if $d = height(t)$. In other words, M^h is applied to the nodes with height at most d, while M^0 is applied to the other nodes above. The reason is that, the lower level nodes are decided by mutual knowledge so M^h yields the best result there; while the higher level nodes are not decided, so conservative minimax is used to guarantee the worst case outcome. The following theorem guarantees that the hybrid strategy is at least as good as minimax, given the player's knowledge.

Theorem 8 *For any game tree t where A moves first, $\forall d \geq 1$, if $f(A,d) \subseteq \mathcal{K}(A)$ then HM^0M^d $\mathcal{K}$-dominates minimax on t for A.*

Proof: For any n with height d, A moves at n or $parent(n)$, so by theorem 5, M^h $\mathcal{K}$-dominates any strategy for A in the subtrees rooted by all such n. Since HM^0M^d is exactly the same as minimax for the nodes with height more than d, HM^0M^d is at least as good as minimax overall. □

Extending theorem 7 to HM^0M^d, we have corollary 4, which can be proved similarly.

Corollary 4 *For any game tree t where A moves first, let $v_1 = value(t)$ by $M^h(t, E_A, E_B)$, $v_2 = value(t)$ by HM^0M^d, $v_3 = value(t)$ by $HM^0M^{d'}$ ($height(t) \geq d \geq d' \geq 0$), and $v_4 = value(t)$ by $minimax(t, E_A)$, then for A: $v_1.a \geq v_2.a \geq v_3.a \geq v_4.a$.*

Once again, our intuition is confirmed that the more knowledge a player has the better outcome he can expect.

Complexity Given t, M^h and HM^0M^d can be done by depth-first search (DFS) in O(height(t)) space and O(size(t)) time. These bounds are already optimal for sound and complete strategies that are guaranteed to find the global optimum if there is one, because first, DFS requires O(height(t)) space for backtracking; second, any sound and complete strategy needs O(size(t)) time, due to the lack of general pruning algorithms by corollary 2.

Knowledge Update via Communication

By theorem 8 and corollary 4, given $f(A,d) \subseteq \mathcal{K}(A)$, a strategy dominating minimax for A is HM^0M^d, which could be still worse than M^h though; but using M^h requires enough knowledge that the players may not have. So knowledge update should be used to increase players' knowledge to a level that allows them to use M^h. We suggest two communication schemes for desired knowledge update, assuming the communication channel is perfect and has no delay.

(I) If $d \geq 1 \wedge f(A,d) \subseteq \mathcal{K}(A)$, A can announce both E_A and E_B to B.

With (I), after A's announcement, E_A and E_B become common knowledge for A and B, so M^h becomes the dominant strategy for both A and B by corollary 3. This would require the logic with both relativized common knowledge and public announcements (PAL-RC), and more details can be found in (van Benthem, van Eijck, & Kooi 2005).

(II) If $d \geq 2 \wedge f(A,d) \subseteq \mathcal{K}(A)$, A can announce to B that A is using M^h.

With (II), given $d-1 \geq 1$ and $f(B,d-1) \subseteq \mathcal{K}(B)$, by property 6 A knows that $K_B E_A \in \mathcal{K}(B)$, and thus A knows that B can verify whether A really uses M^h during the play. By theorem 5 M^h is mutually enforced, in that, A cannot benefit from lying to B that he uses M^h, because once B is forced to use M^h A has to use M^h as well for his own best. Therefore, neither A nor B has the incentive to deviate from M^h if the opponent appears to be using it.

The Example Revisited with M^h and HM^0M^d For the example in Figure 2, the baseline value is (4,7), by minimax in scenarios (1). If communication is available, in scenarios (2)-(4), A will first use the above schemes for desired knowledge update. Then A will use M^h to force B to use M^h too, as explained before, and finally they will reach the outcome (10,3), which is better than M^* in (3) and (4) and the same in (2).

If communication is not available, A will use HM^0M^d (with $d=1$ for (2) and (3) and $d=2$ for (4)) based on A's knowledge in each scenario, which is described right before theorem 1. Finally, players will reach the outcome (9,4).

With communication or not, both our strategies dominate minimax in all the scenarios and outperform M^* in scenario (3) where players' knowledge does not subsume each other.

Conclusions and Future Work

This paper is focused on finite-horizon nonzero-sum game tree search with potentially imperfect mutual knowledge between the players, which is a topic addressed little by the traditional game theory. We have proposed a new theoretical framework of Knowledge Oriented Players, which uses the $S5_n$ axiom system to model and reason about players' knowledge. The good expressiveness of our framework has allowed us, for the first time, to analyze the general relationship between knowledge and strategies in such games, in particular, how strategies are constrained by and can benefit from knowledge. Leveraging this analysis, we have developed two new algorithms–M^h and HM^0M^d, which provably dominate minimax under a class of knowledge scenarios that is significantly more general than previous algorithms for such problems. Furthermore, our communication schemes allow players to achieve desired knowledge update, so that they can use the mutually-enforced M^h algorithm to reach the best possible outcome, given their knowledge.

There is plenty room for future work. Players may have partial knowledge of the opponents' evaluation functions; so far we only consider atomic knowledge of evaluation functions, either fully known or unknown. Another direction is to study other classes of knowledge that can impose constraints on strategies. Furthermore, knowledge may be not absolutely true but with some probability to hold and how can uncertainty be incorporated? Finally, can interesting topics in game theory, such as mixed strategies , be relevant to our discussions of knowledge under certain circumstances?

References

Carmel, D., and Markovitch, S. 1993. Learning models of opponents strategies in game playing. In *AAAI Fall Symposion on Games: Planning and Learning*, 140–147.

Carmel, D., and Markovitch, S. 1996. Incorporating opponent models into adversary search. In *Proceedings of 13th National Conference on Artificial Intelligence (AAAI)*.

Cox, M. T., and Raja, A. 2007. Metareasoning: A manifesto, technical report, bbn tm-2028, bbn technologies. www.mcox.org/metareasoning/manifesto.

Donkers, H. 2004. Opponent models and knowledge symmetry in game tree search. In *Proceedings of the First Knowledge and Games Workshop*, 82–88.

Fudenberg, D., and Tirole, J. 1991. *Game Theory*. MIT Press.

Halpern, J. Y., and Moses, Y. 1992. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence* 54(3):319–379.

Iida, H.; Uiterwijk, J.; Herik, H.; and Herschberg, I. 1993. Potential applications of opponent-model search. part 1: the domain of applicability. *ICCA* 4(16):201–208.

Kuhn, H. 1953. Extensive games and the problem of information. *Annals of Mathematics Studies* Volume II, Contributions to Game Theory:193–216.

Shannon, C. 1950. Programming a computer for playing chess. *Philosophical Magzine* 256–275.

van Benthem, J.; van Eijck, J.; and Kooi, B. 2005. Common knowledge in update logics. In *Proceedings of 10th Theoretical Aspects of Rationality and Knowledge (TARK)*.