

Decision Algorithms for Multiplayer Noncooperative Games of Incomplete Information

G. PETERSON

Computer Science Department
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

J. REIF AND S. AZHAR

Computer Science Department
Duke University
Durham, NC 27706, U.S.A.

Abstract—Extending the complexity results of Reif [1,2] for two player games of incomplete information, this paper (see also [3]) presents algorithms for deciding the outcome for various classes of multiplayer games of incomplete information, i.e., deciding whether or not a team has a winning strategy for a particular game. Our companion paper, [4] shows that these algorithms are indeed asymptotically optimal by providing matching lower bounds. The classes of games to which our algorithms are applicable include games which were not previously known to be decidable. We apply our algorithms to provide alternative upper bounds, and new time-space trade-offs on the complexity of multiperson alternating Turing machines [3]. We analyze the algorithms to characterize the space complexity of multiplayer games in terms of the complexity of deterministic computation on Turing machines.

In hierarchical multiplayer games, each additional *clique* (subset of players with the same information) increases the complexity of the outcome problem by a further exponential. We show that an $S(n)$ space bounded k -player game of incomplete information has a deterministic time upper bound of $k + 1$ repeated exponentials of $S(n)$. Furthermore, $S(n)$ space bounded k -player blindfold games have a deterministic space upper bound of k repeated exponentials of $S(n)$. This paper proves that this exponential blow-up can occur.

We also show that time bounded games do not exhibit such hierarchy. A $T(n)$ time bounded blindfold multiplayer game, as well as a $T(n)$ time bounded multiplayer game of incomplete information, has a deterministic space bound of $T(n)$.

Keywords—Algorithms, Alternation, Complexity theory, Game theory, Incomplete information, Blindfold.

Supported by Grants NSF/DARPA CCR-9725021, CCR-96-33567, NSF IRI-9619647, ARO contract DAAH-04-96-1-0448, and ONR contract N00014-99-1-0406.

A PDF version of this paper is at <http://www.cs.duke.edu/~reif/paper/games/alg/alg.pdf>.

Preprint of paper appearing in *Computers and Mathematics with Applications*, Vol. 43, Jan. 2002, pp 179–206.

1. INTRODUCTION

1.1. Motivation

Game theory paradigms arise quite naturally in computer science. In recent years, there has been a proliferation of applications of noncooperative game theory to computer science, and other disciplines. Several problems in computer science can be formulated in terms of games. These problems present numerous incentives for developing game theoretic algorithms. On the other hand, the notion of computation is essential to the fundamental questions of game theory like the *outcome problem*. The outcome problem can be stated as follows: "Do the players of Team T_1 have a winning (or nonlosing) strategy, which together would defeat Team T_0 (or save Team T_1 from defeat) under all circumstances?"

The development of the theory of games was motivated by economic decision making problems, just like the development of calculus was motivated by physics. Game theory was originally formulated by von Neumann and Morgenstern [5]. Subsequently, game theory, like calculus, has been applied to a wide range of fields [6].

Games are intimately related to models of computation. The fundamental question of concrete games (the *outcome problem*) is closely related to the membership question of languages and machines. A game with a computable next-move relation can be treated as a computation machine: machine M accepts input ω depending on the outcome of the corresponding game G^M from an initial position encoded by ω .

Computer scientists usually study games from one of two perspectives. First, developing and analyzing decision algorithms for particular games. For example, one can consider the algorithms for determining the optimal move in a position of Chess, and the associated complexity issues. Second, employing game theoretic models for particular computational paradigms and problems. For example, parallel computation models. Our paper bears special relevance to both of these concerns, and briefly reviews closely related literature.

Game theoretic ideas have been applied to design algorithms for distributed systems. In a distributed system, the processors are treated as players, and the information content of positions is based on the current states of the processes. Ben-Or and Linial [7,8] apply voting schemes of game theory to design algorithms that reconcile processors of a distributed system. Our multiplayer games can be used to model distributed computation.

Our work straddles classical game theory and computer science, and we have an unusual expository burden insofar as we wish our work to be accessible to researchers in both fields. For computer scientists, we explain the fundamentals of game theory, including the terminology and formalization of finite games. We also motivate the reasons for regarding these objects as central in noncooperative game theory. For game theorists, we explain several computer science concepts, such as computational procedures, computational models, and algorithmic analysis. Subsequently, we show how these computer science concepts are applied to the game algorithms under scrutiny.

1.2. Overview

The rest of Section 1 is devoted to reviews of the fundamentals of games as well as associated computational models, and directs the reader toward some related work in the field.

Section 2 formally defines the term "game" as a set of players faced with a choice of alternatives, at every stage of the game, until the game terminates according to a prespecified rule. A player may have to choose his/her alternate (move) at a stage with limited knowledge. Every terminating position is associated with a payoff, and the object of every player is to maximize their payoff. Section 2 emphasizes that we are interested in the most general characterization of games, and affords a detailed and mathematically precise description of two-player games and multiplayer games. Section 3 defines *win outcome problem*. This is the problem of ascertaining whether

players of Team T_1 have a strategy ensuring them of a win by yielding only winning plays regardless of opponent strategy. Winning plays must be finite because they must terminate in win for T_1 . A slight variation of the win outcome problem is *nonloss outcome problem*. This is the problem of ascertaining whether Team T_1 has a strategy yielding only plays with no losses to any player in Team T_1 . A nonloss play may be infinite, because we can satisfy the nonloss criterion as long as the play does not terminate in a loss for T_1 . We would also consider the *Markov ($m(n)$) problem*: given an initial assessment of length n , does Team T_1 have a winning (nonlosing) strategy dependent on the previous $m(n)$ positions of any play? Section 3 also introduces complexity theoretic notation to facilitate complexity analysis of the algorithms and machines related to games. This section also discusses several varieties of interesting games.

In Section 4, we provide an algorithm for the Markov ($m(n)$) outcome problem of any $S(n)$ space bounded games. This extends the result of Peterson and Reif [3], which addressed the special case of $m = 1$. We propose a decision algorithm for the outcome problem of any hierarchical game with both a space bound and an alternation bound. Finally, in Section 5, we describe a decision algorithm for time and branch bounded multiplayer games of incomplete information. The algorithmic results of Sections 4 and 5 lead to several corollaries, which give bounds for players induced by winning strategies in any hierarchical game as well as for Markov ($m(n)$) winning strategy for any game.

Section 6 concludes the discussion by summarizing the results of this paper, previewing the results of next paper, and highlighting areas of future research.

The main contribution of this paper is that it establishes the following facts.

- The time required to simulate a space bounded game of incomplete information increases by an exponential for every additional player.
- The space required to simulate a space bounded blindfold game increases by an exponential for every additional player.
- The time required to simulate time bounded games does not change with increase in number of players.

1.3. Fundamentals

This section provides an introduction to game theory for the computer scientists' benefit, and surveys the basic computational models for the game theorists' convenience. It describes basic terminology, and affords a concise survey of the fundamental principles of computational aspects of games.

Game theory is the theory of rational decisions involving computations of strategies to be used against "rational" opponents. By "rational" opponents, we mean actors also involved in formulation of optimal strategies in the pursuit of maximizing their payoff. The focus of the theory of games is on the fundamental issues, rather than on the development of specific strategies. Consequently, the perspective adopted in the theory of games is very different from the approach taken in developing algorithms to win in Chess, where the central problem is to formulate winning strategies for Chess. Game theorists hoist their study to increasing levels of abstraction by attempting to solve the general and basic problems.

From a game theorist's perspective, the theory of Chess is essentially trivial, since it can be reduced to an exhaustive search problem (provided the required computational resources are at one's disposal). However, this assumption is far from being realistic or practical. In reality, developing efficient algorithms for games like Chess have challenged researchers in computer science to develop more efficient models of thinking, reasoning, and searching. In computer science, we are not just interested in the underlying game theoretic principles of games, but also concerned about the computability issues involved in games. In this paper, we address both aspects.

In computer science literature, a two-player game is defined by disjoint sets of positions for two players (named 0 and 1), and relations specifying legal next moves for players. A position p may contain portions that are *private* to one of the players, whereas the rest are *common* portions accessible to both players. Reif [1,2] provides a detailed treatment of two-player games.

The generalization to a two-player game is a multiplayer game (also called team game)¹. In multiplayer games, there are at least three players partitioned into two teams, T_0 and T_1 . A multiplayer game is specified by a set of positions, a relation defining the possible next moves, division of teams, and access rights of players to view or modify certain components of a game position.

We assume that positions are strings over a finite alphabet. Every position p may contain certain information that is *private* to some subset of players. The remaining information is *common*, and may be viewed by all players. The set of legal next-moves for a given player must be independent of the information that is inaccessible to him. These rights remain unmodified throughout the game.

In any game, every player plays according to a strategy. This paper focuses on investigating strategies that dictate a single next move to the player for every possible sequence of previous moves that can legally occur. Such strategies are called *pure* strategies. Conversely, *mixed* strategies assign probabilities to all possible next moves that can be made from a nonterminating position. The reader is referred to the papers on mixed strategies by Azhar, McLennan and Reif [9] for more detailed treatment of the complexity of finding such mixed strategies.

A pure strategy of each player can only be dependent on components of the position visible to the player. Team T_1 is always the team of "preference" in the sense that we are interested in algorithms to formulate strategies for Team T_1 , and analyze the complexity of these algorithms as a function of Team T_1 's size. We model Team T_0 as a single player.

A winning strategy specifies a set of legal moves from current position to positions guaranteeing T_1 a win, regardless of Team T_0 's response. A nonloss strategy specifies a set of legal moves from current position to positions guaranteeing T_1 a nonloss, regardless of Team T_0 's response. The *win (or nonloss) outcome problem* is a fundamental problem in game theory. For a team game, it can be described as follows.

"Do the players of Team T_1 have a winning (or nonlosing) strategy, which together would defeat Team T_0 (or save Team T_1 from defeat) under all circumstances?"

Besides outcome problems, this paper also considers *Markov ($m(n)$) outcome problem*: "Given initial position of length n , does Team T_1 have a winning strategy dependent only on previous $m(n)$ positions?" Markov (1) outcome problem is considered by Peterson and Reif [9].

A game has perfect information if no position has any private component, and a game has incomplete information if there are certain restricted access components to the game. A player may have incomplete information about a position because it does not have rights to view some portion of the position. Games in which at least one player has incomplete information are naturally known as games of incomplete information. A game is categorized as a blindfold game if the Team T_0 never modifies any portion of the positions that is visible to players of Team T_1 .

In this paper, we shall show that (from a complexity theoretic point of view) multiplayer games of incomplete information are more difficult than two-player games of incomplete information. Strategies in games of perfect information depend only on the current position. On the other hand, strategies in games of incomplete information (as well as blindfold games) depend on the history of the visible portion of positions. Intuitively, the history of the visible portions is used to deduce limited information about inaccessible portions of the positions.

¹We will use the two terms, multiplayer games and team games, interchangeably

1.4. Computational Models

The definition of Turing machine facilitated the development of computability theory by formalization of algorithmic procedures (see [10]). Similarly, several other paradigms of computations (nondeterminism, parallel, etc.) were associated with corresponding models of computations (nondeterministic Turing machine, parallel random access machine, etc.). The need for a formal computational model to address the computational aspects of games was fulfilled by Chandra, Kozen and Stockmeyer [11] with the *alternating Turing machine* (A-TM). Subsequently, this model has been extended and enhanced to model more intricate games. Reif [1,2] extended A-TM model to incorporate private and blindfold two-player games by introducing private alternating Turing machine (PA-TM) and blind alternating Turing machine (BA-TM), respectively. In this paper, we introduce multiperson private alternating Turing machine ($\text{MPA}_k\text{-TM}$) and multiperson blind alternating Turing machine ($\text{MBA}_k\text{-TM}$) to model private and blindfold multiplayer games, respectively. We also define $\text{PA}_k\text{-TM}$ and $\text{BA}_k\text{-TM}$ to remove the over-generality of $\text{MPA}_k\text{-TM}$ and $\text{MBA}_k\text{-TM}$, respectively.

We assume that the reader is familiar with the usual definitions of Turing machines, and in particular, the definitions of tape storage, tape read/write heads, Turing machine configurations (which are the positions of these computational games), and legal next moves for Turing machines.

The A-TM models a two-player game in which the existential states (identified with Player 1) alternate with the universal states (identified with Player 0) during the computation. The A-TM accepts an input, corresponding to an initial position, if the existential player has a winning (or a nonloss) strategy. A winning (or a nonloss) strategy is one that would lead to a win (or never lead to a loss) for the existential player under all circumstances, regardless of the strategy adopted by the universal player. The complexity of various generalized games of perfect information is considered by Schaefer [12], Even and Tarjan [13], Fraenkel *et al.* [14], Robson [15], Lichtenstein and Sipser [16,17], Fraenkel and Lichtenstein [18], and Peterson [19]. Stockmeyer and Chandra [20] define a game PEEK, and prove that it is universal for two-player games of complete information.

A string ω encoding some position is accepted by an alternating Turing machine if

- the machine is in a universal (\forall) state, and all transitions from that state (based upon the current scanned symbols) are to accepting states;

or

- the machine is in an existential (\exists) state, and there is at least one transition from that state (based upon scanned symbols) to an accepting state.

The nondeterministic Turing machine (N-TM) is mapped to games of perfect information with Player 0 absent because there are no universal states. Deterministic Turing machines (D-TM) represent games of perfect information with at most single next-move from any position because there is only one possible transition from any given state. The A-TMs are essentially extensions of nondeterministic machines to include both existential and universal choices. These choices then correspond to moves by the two opposing player in a two-player game of perfect information.

Reif [1,2] extended the notion of alternation to two-player games of incomplete information by restricting the players to limited information for making their strategic decisions. He introduced private alternating Turing machine (PA-TM) and blind alternating Turing machine (BA-TM) to model two player games of incomplete information and two player blindfold games, respectively. PA-TM is derived from a A-TM by not allowing the existential (\exists) player to access all work tapes that are private to the universal (\forall) player. BA-TM is derived from a PA-TM by not allowing the universal (\forall) player write access to work tapes that can be read by the existential (\exists) players. PA-TM and BA-TM model two-player games of incomplete information (for example, Rummy) and two player blindfold games (for example, BLIND-PEEK [1,2]), respectively. Note that Blindfold Chess is considered a game of incomplete information blind because a player can deduce certain

characteristics of the board when the player attempts to make a move that is termed illegal due to information that previously was not known to them.

In this paper, we present extensions of the alternating machines of Reif [1,2] and Chandra, Kozen and Stockmeyer [11]. Our machines model multiplayer games of incomplete information.

$\text{MPA}_k\text{-TM}$ is a machine model that corresponds to a $(k + 1)$ -player multiplayer (team) game of incomplete information with k existential players and one universal player. The states of the machine are labeled with tuples: each element of the state contains a turn indicator, and denotes information that can be read and written by each player. Every player has an associated list of tapes to indicate read and write rights of various tapes for the player itself. Thus, the tapes are partitioned according to access rights.

In particular, for $k = 1$, the $\text{MPA}_k\text{-TM}$ (that is, $\text{MPA}_1\text{-TM}$) bears resemblance to PA-TM . Similarly, the $\text{MPA}_1\text{-TM}$ with both players sharing resources corresponds to A-TM (without logical “NOT” operation). Furthermore, an $\text{MPA}_1\text{-TM}$ with unique next moves for the universal (\forall) player is an N-TM , whereas a $\text{MPA}_1\text{-TM}$ with unique next move for all players is simply a D-TM . Hence, evidently $\text{MPA}_k\text{-TM}$ accepts the recursively enumerable (r.e.) languages since they are at least as powerful as ordinary D-TM . Similarly, we can show that $\text{MPA}_k\text{-TM}$ accepts only r.e. languages by enumerating all possible accepting subtrees, and subsequently, checking recursively whether each tree is a true accepting subtree.

$\text{MBA}_k\text{-TM}$ is derived from $\text{MPA}_k\text{-TM}$ by disallowing the \forall -player to write on any resource that is readable by any \exists -player. Consequently, moves of the \forall -player are invisible to the \exists -player. Hence, $\text{MBA}_k\text{-TM}$ correspond to blindfold multiplayer games. Observe that for $k = 1$, the $\text{MBA}_k\text{-TM}$ (that is, $\text{MBA}_1\text{-TM}$) bears resemblance to BA-TM . We show that $\text{MBA}_k\text{-TM}$ and $\text{BA}_k\text{-TM}$ accept r.e. languages and only r.e. languages.

We need to realize that analyzing both $\text{MPA}_k\text{-TM}$ and $\text{MBA}_k\text{-TM}$ are too general and powerful after we analyze them. We combat their over-generality by introducing restricted versions of each machine. A $k + 1$ -player private alternating Turing machine ($\text{PA}_k\text{-TM}$) is an $\text{MPA}_k\text{-TM}$ if resources visible to Player i are also visible to player $i - 1$ for all existential players ($i \in \{2, 3, \dots, k\}$). Hence, there is a hierarchical ordering of \exists -players. A $k + 1$ -player blind alternating Turing machine ($\text{BA}_k\text{-TM}$) is a $\text{PA}_k\text{-TM}$ where the \forall -player cannot change a resource visible to any other \exists -player. Examples of $\text{PA}_k\text{-TM}$ and $\text{BA}_k\text{-TM}$ for $k = 3$ are depicted in Figures 1 and 2, respectively.

1.5. Related Work

Games can be classified into two types: probabilistic games and nonprobabilistic games. Strategies for nonprobabilistic games involve specifying exactly one alternative at each position: such strategies are known as “pure” strategies. Nonprobabilistic games follow a set course of play once the participating players have formulated their strategies. On the other hand, the outcome of probability-related games can be influenced by random events (such as coin tosses or die rolls) which are not in any player’s control. Consequently, strategies for probabilistic games involve assigning probabilities to various alternatives available at each position: such strategies are known as “mixed” strategies. In this paper, we are primarily concerned with games involving pure strategies.

Papadimitriou [21,22] describes *games against nature*. In these games, one player plays randomly simulating the randomness we associate with nature, and the other player existentially selects a strategy that maximizes the probability of success against this random player. In this framework, the existential player is considered to have won the game, if it can win with a probability greater than $1/2$. *Games against nature* paradigms assist in formulation of decision problems under uncertainty. These games are similar to *Arthur-Merlin games* of Babai [23], in which Arthur plays randomly, and Merlin plays existentially. Interactive proof systems of Goldwasser *et al.* [24] are also among examples of games in which one player plays randomly whereas the

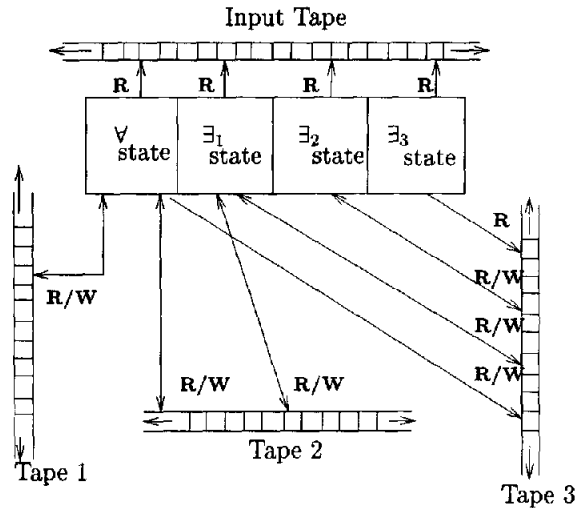


Figure 1. Private alternating Turing machine.

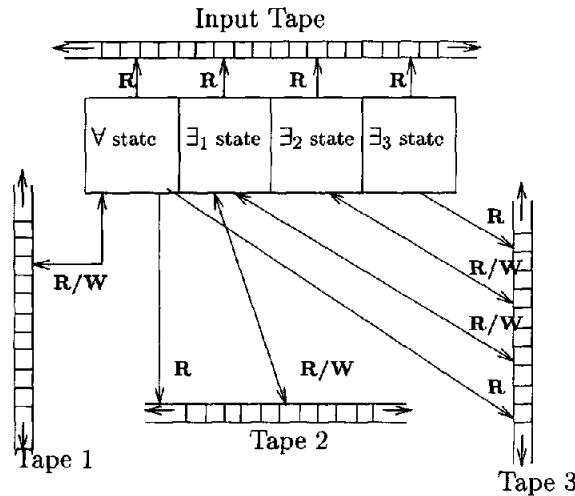


Figure 2. Blind alternating Turing machine.

other existentially picks a strategy. Goldwasser and Sipser [25] have proved the equivalence of interactive proof systems and Arthur-Merlin games. Shamir [26] proves both problems are in the same complexity class (PSPACE-complete).

Another special class of games is solitaire games. Solitaire games restrict the play of one of the players to be completely deterministic after the player's first move. These games are investigated by Ladner and Norman [27].

In games of incomplete information, the existential player does not have complete knowledge of some portions of the positions (which are private to the universal player). In blindfold games, the existential player does not have any rights whatsoever to view the moves of the universal player. The complexity of such games was first considered by Jones [28]. Reif [1,2] also discusses this issue.

The basic perfect information alternation problem of quantifier Boolean formula (QBF) [11,29] has been very useful in demonstrating that natural games are PSPACE-complete (or hard) [12,16–19,30]. Similarly, we expect the natural problems in complexity of *logical theories* may be resolved by way of multiple person alternation games of incomplete information and the characteristics we formulate. Note that upper bounds for logical theories frequently involve games. A modal logic with incomplete information is studied in more detail by Reif and Peterson [31]. In our companion paper [4], we prove that dependency QBF (DQBF) is NEXPTIME-complete.

2. MULTIPLAYER GAMES

2.1. Preliminaries

A game is defined as a set of rules, which specifies the following items.

1. A set of positions.
2. A set of players.
3. A specification of degree of knowledge a player has on its turn, that is an assignment of the rights of the players to view certain components of the game position,
4. A rule specifying the conditions under which the game starts.
5. A set of alternatives, depending on the situation (legal position), available to every player on its turn to modify certain components of the game position.
6. Rule specifying when the game terminates.
7. A set of payoffs (awarded to each player) associated with each possible outcome at the termination of the game. In this paper, we restrict ourselves to payoffs of $+1, 0$, and -1 denoting them as *win*, *draw*, and *loss*, respectively.

We assume that positions are strings over finite alphabet. A position p may contain certain information that is *private* to some players (and invisible to others). The remaining information is *common*, and may be viewed by all players. The set of legal next-moves for a given player must be independent of the information that is inaccessible to it. The rights to view components of different positions remain unchanged throughout the game.

REMARK 2.1.1. We take a very general view of games to include as many variants as possible. Accordingly, some important properties to expand the classes of games are listed below.

1. Players need not take turns in round-robin fashion. The rules of the game will dictate whose turn is next.
2. Players on the same team may communicate only as allowed by the rules of the game.
3. A player can recall its history of positions, and changes to the visible portions of the position which occur between its turns.
4. A player may not know who can change (or has changed) its visible portion of position.
5. A player may not know how many turns were taken by other players between its turns.

2.2. Two-Player Games with Incomplete Information

Before we delve deeper into multiplayer games, we shall first refresh the definitions of two-player games. Subsequently, we extend these definitions to multiplayer games.

DEFINITION 2.2.1. TWO-PLAYER GAME: (See [1,2].) A two-player game is a tuple (POS, \vdash) , where

- (i) players are named 0 and 1;
- (ii) POS is a set of positions, with $POS = \{0, 1\} \times PP_0 \times PP_1 \times CP$, such that PP_0, PP_1, CP are strings over finite alphabets;
- (iii) $\vdash \subseteq POS \times POS$ is a next move relation, which describes all alternatives available at any position. \vdash satisfies Axioms 1 and 2 below.

Consider position $p = (a, pp_0, pp_1, cp) \in POS$. Position p is composed of the following.

1. A number $a \in \{0, 1\}$ indicating which player's turn is next.
2. A portion pp_0 that is private to Player 0.
3. A portion pp_1 that is private to Player 1.
4. A common portion cp which consists of information accessible to both players (0 and 1).

Consider a position $p = (a, pp_0, pp_1, cp)$. For $i = 0$ or 1 , we let

- $vis_i(p) = (a, pp_i, cp)$ denote the portion of position p visible to Player i ;
- $priv_i(p) = (pp_i)$ denote the portion private to Player i .

We define POS_i to be the set of positions in which it is Player i 's turn to move, and W to be the set of positions without any legal next move. The object of the game is to force the opponent into a position where it cannot move.

The effect of incomplete information is captured by the following axioms imposing restriction on what a player may modify and access.

AXIOM 1. *A player cannot modify its opponent's private information.*

This axiom effectively states that if $p \in \text{POS}_1 - W$, and $p \vdash p'$ then $\text{priv}_0(p) = \text{priv}_0(p')$.

AXIOM 2. *A player's next moves are independent of the opponent's private information.*

As a consequence of Axiom 2, for any Player i , if $p, q \in \text{POS}_i - W$ and $\text{vis}_i(p) = \text{vis}_i(q)$, then $\{\text{vis}_i(p') \mid p \vdash p'\} = \{\text{vis}_i(q') \mid q \vdash q'\}$.

2.3. Multiplayer Game Definition

We can extend the definition of two-player games to multiplayer games. Formally, a $k+1$ -player game can be defined as follows.

DEFINITION 2.3.1. *$k+1$ -PLAYER GAME: A $k+1$ player game is a quadruple $[G = (\text{POS}, \vdash, \text{vis}, T_1)]$, where we have the following.*

1. POS is a set of positions with $[\text{POS} = \{0, \dots, k\} \times P_1 \times \dots \times P_r]$, where $0, \dots, k$ represent players $0, \dots, k$, and P_1, \dots, P_r are sets of strings over a finite alphabet used to describe various components of the positions. Let $p = (a, p[1], \dots, p[r])$ be a position in POS , where for each $j = 1, \dots, r$: $p[j]$ is an element of P_j , and is called the j^{th} component of p .
2. $\vdash \subseteq \text{POS} \times \text{POS}$ is the next move relation satisfying Axioms 3 and 4 stated below. These axioms simply prevent a player from modifying the information that is invisible to him/her, and inhibit him/her from using the invisible information in formulating his/her strategy.
3. Let $p = (a, p[1], \dots, p[r])$ be a position in POS . $\text{next}(p) = a$ is the turn indicator specifying the player whose turn it is to move next. The turn indicator cannot be used in formulating a strategy according to the requirements listed in Remark 2.1.1.
4. The mapping $\text{vis} : \{0, \dots, k\} \mapsto 2^{\{1, \dots, r\}}$ is a mapping from the set of players to the set of all possible subsets of the set of positions. Let $p = (a, p[1], \dots, p[r])$ be a position in POS . We say that Player i has right to view $p[j]$ if $j \in \text{vis}(i)$.
5. Team $T_1 \in 2^{\{0, \dots, k\}}$ is a subset of the set of players $\{0, \dots, k\}$. The opposing team is $T_0 = \{0, \dots, k\} - T_1$ where “ $-$ ” represents the set difference. Players of Team T_1 move from existential (\exists) states, and players from the other Team T_0 move from universal (\forall) states and are called universal (\forall) players.

In two-player games, a certain component of position can either be accessible to one of the two players, or to both players. Any information inaccessible to both players is irrelevant to formulate strategy. Hence, we can express the incomplete information content with just three components of any position: pp_0 , pp_1 , and cp . On the other hand, in multiplayer games a certain component of the game might be simultaneously accessible by some *subset* of the players. Consequently, we need a more expressive notation, such as the mapping vis , to describe the access rights of the players.

DEFINITION 2.3.2. vis_i : For each player $i = 0, \dots, k$ in a $k+1$ -player game, we let $\text{vis}_i(p) = (v, b)$, where v is the list (in order of occurrence) of components of position p for which Player i has access rights (that is the ordered list $\langle p[j] \mid j \in \text{vis}(i) \rangle$)², and b is a Boolean variable that is 1 if it is Player i 's turn to move (and 0 otherwise).

²We use angled brackets to enclose items in an ordered list.

DEFINITION 2.3.3. priv_i : For each player $i = 0, \dots, k$ in a $k + 1$ -player game, $\text{priv}_i(p)$ is the list of all components of the position that are known only to Player i . Explicitly $\text{priv}_i(p)$ is $\langle p[j] \mid j \in \text{vis}(i) \rangle$, and $j \notin \text{vis}(k)$ for all $k \neq i$.

The object of the multiplayer games is to force the opponents into a position from where they cannot move. This corresponds to the objective of the two-player games, which is to force the opponent into a position from where (s)he cannot move. Consequently, we define the set of *winning positions*.

DEFINITION 2.3.4. WINNING POSITION (W): The set of winning position (W) is the set of all positions from which there is no next move for the opponent on its turn to move:

$$W = \{p \in \text{POS} \mid \text{there is no } p' \text{ such that } p \vdash p'\}.$$

DEFINITION 2.3.5. POS_i : For any Player i , we use POS_i to denote the set of positions such that it is Player i 's turn to move.

$$\text{POS}_i = \{p \in \text{POS} \mid \text{next}(p) = i\}.$$

Player i loses if any position in the set $\text{POS}_i \cap W$ is encountered. The result of any finite play is a loss for exactly one player. Any team's wins and losses are determined by the performance of the players on that team: a play π is a *win for team T_1* if it is a loss for some player on the other team. A play π is a *nonloss for team T_1* if it is not a loss for any player on the Team T_1 .

If players are allowed to compete individually to maximize their winnings (as in mathematical game theory), the outcome problem can no longer be transformed so succinctly into a "yes or no question". Consequently, there is no simple computational machine model for such a paradigm. One approach would be to treat every bit of the vector specifying the winnings as a "yes or no question" (i.e., 1 for yes, 0 for no). However, such an approach leads to nontrivial computational complexity.

When we are dealing with multiplayer games, we can adapt Axioms 1 and 2 to incorporate the notion of incomplete information.

AXIOM 3. No player is permitted to modify any other player's private information.

Therefore, if $p \in \text{POS}_i - W$ and $p \vdash p'$, then $\text{priv}_j(p) = \text{priv}_j(p')$ for all players $j \neq i$.

AXIOM 4. If a pair of nonterminating positions p, q are indistinguishable to a player, then the set of next moves is independent from this pair p, q .

As a consequence of Axiom 4, if $p, q \in \text{POS}_i - W$ and $\text{vis}_i(p) = \text{vis}_i(q)$ then $\{\text{vis}_i(p') \mid p \vdash p'\} = \{\text{vis}_i(q') \mid q \vdash q'\}$.

3. GAME PLAYING

3.1. Plays

We fix an initial position p_0 before a play of the game commences. Our definitions accommodate games with *arbitrary* initial positions, even though some popular strategic games have a fixed initial position. For example, Chess and Go have only one initial position. Risk is a popular strategy game in which there can be numerous possible initial positions. The initial position may be selected randomly, or the turns of players may be randomly permuted at the commencement of the game.

DEFINITION 3.1.1. PLAY: A *play* is a possibly infinite string $\pi = p_0 p_1 \dots$ of positions, such that for all relevant nonnegative integers i : $p_i \vdash p_{i+1}$, where $p_i \in \text{POS}$ and $p_{i+1} \in \text{POS}$ are positions.

For example, consider the play $\pi = p_0 p_1 p_2 p_3 p_4$. Here p_0 is the initial position, and $p_0 \vdash p_1$, $p_1 \vdash p_2$, $p_2 \vdash p_3$, and $p_3 \vdash p_4$ are the moves that constitute the play π . A *play prefix* π is a finite nonnull initial substring of a play that represents a sequence of legal moves starting from initial position. For example, $p_0 p_1 p_2$ is a *play prefix* of $\pi = p_0 p_1 p_2 p_3 p_4$.

A function $\text{last}(\pi)$ returns the last position in the play π .

DEFINITION 3.1.2. $\text{last}(\pi)$: If play π is finite, then $\text{last}(\pi)$ is defined to be the last element of the string π .

When a finite play terminates, by definition $\text{last}(\pi)$ must be in W ($\text{last}(\pi) \in W$). A play is a loss for Player i if the player is placed in a position where it is forced to move yet there is no legal move available to it. Formally, π is finite and $\text{last}(\pi) \in \text{POS}_i \cap W$.

$\text{vis}_i(\pi)$ represents the extent of Player i 's knowledge about the play in the game up to date. We say that the move $p \vdash p'$ is invisible to Player i if

1. it does not alter any portion visible to Player i (i.e., $\text{vis}_i(p) = \text{vis}_i(p')$), and
2. it is not Player i 's move (i.e., $p \notin \text{POS}_i$).

The extent to which Player i 's knowledge is modified by some other player's move is determined by how much the common knowledge is modified. We inductively define $\text{vis}_i(\pi)$ on a play π as follows.

DEFINITION 3.1.3. $\text{vis}_i(\pi)$: For a play π : If the length of π is 1 then $\text{vis}_i(\pi) = \{\text{vis}_i(\text{last}(\pi))\}$. Otherwise, suppose that there is a position $p' \in \text{POS}$ such that $\text{last}(\pi) \vdash p'$. Now, if the move $\text{last}(\pi) \vdash p'$ is invisible to Player i then $\text{vis}_i(\pi p') = \text{vis}_i(\pi)$, else $\text{vis}_i(\pi p') = \text{vis}_i(\pi) \cup \{\text{vis}_i(p')\}$.

3.2. Game Tree

A game can also be represented in the *extensive form* by a *game tree*. A game tree consists of a set of play prefixes, with the root node representing the starting position of the game. Each node represents a position, and its children are the positions after the next move. Every node is connected to its children with branches labeled with each of the alternative moves that can be chosen by the player whose turn it is.

It is important to note here that two equivalent situations in a game that occur at different stages of the game are considered distinct, and they correspond to different nodes in the game tree. Similar position may occur at different stages of the game due to several reasons, such as transposition of moves or repetition. It is also possible that the identity of the player who is to move next is determined by the situation of the game. A game represented by its game tree is said to be represented in its extensive form.

The *root* of a *game tree* is the initial position p_0 . The nodes of the *game tree* consist of all possible plays that can be enumerated by any sequence of legal moves starting from the initial position p_0 . The *children* of π are those play prefixes π' of length one more than π , such that π is a play prefix of π' , and there is a next-move relation between $\text{last}(\pi)$ and $\text{last}(\pi')$.

DEFINITION 3.2.1. **GAME TREE:** $GT = (\text{POS}(p_0), \vdash')$ is a game tree where

1. $\text{POS}(p_0) \subseteq \text{POS}$ is the set of all positions reachable from initial position p_0 ;
2. the root of GT is the initial position, p_0 ;
3. $\vdash' \subseteq \text{POS}(p_0) \times \text{POS}(p_0)$ is the maximal subset of \vdash with the domain restricted to positions which occur in $\{\text{last}(\pi) \mid \pi \in \text{POS}(p_0)\}$;
4. for any $\pi \in \text{POS}(p_0)$: $\text{last}(\pi) \vdash' p$ if and only if πp is a child of π and $(\text{last}(\pi), p) \in \vdash'$.

3.3. Outcome Problem

We categorize the nodes in a game tree by the index of the player whose turn it is to move. Let $\Pi(p_0)$ denote the set of play prefixes reachable from initial position p_0 . Let $\Pi_i(p_0) \subseteq \Pi(p_0)$ denote the set of play prefixes reachable from initial position p_0 with

1. Player i 's turn to move at last position of the play;
2. Player i has at least one legal move available.

These are the set of play prefixes π such that $\text{last}(\pi) \in \text{POS}_i - W$. *Strategy* is the approach used by the players to decide which move to select from their alternatives. For a game with a starting position p_0 , we shall abbreviate $\Pi(p_0)$ and $\Pi_i(p_0)$ as Π and Π_i , respectively. Any strategy σ must satisfy the conditions in the definitions below.

DEFINITION 3.3.1. STRATEGY: For a Player i , a strategy is a function $\sigma : \Pi_i \mapsto \Pi$ such that

1. for any $\pi \in \Pi_i$, $\sigma(\pi)$ is a child of π (i.e., there is a $\pi \vdash \sigma(\pi)$ relationship in GT);
2. if $\pi, \pi' \in \Pi_i$ and $\text{vis}_i(\pi) = \text{vis}_i(\pi')$, then $\text{vis}_i(\sigma(\pi)) = \text{vis}_i(\sigma(\pi'))$.

Condition 1 above restricts the player to legal moves. Condition 2 ensures that the strategic decisions must be made using only the knowledge visible to the player.

We say that a play π is *induced* by strategy σ if whenever π' is a play prefix of π , and π' is in the domain of σ , then $\sigma(\pi')$ is a (not necessarily proper) prefix of π .

REMARK 3.3.1. For our purposes, it is sufficient to model all the universal players by one single universal player (a “super-player”) who has all the information. Consequently, a $k+1$ -player game consists of team of k \exists -players and one \forall -player. Since our machine models are formulated with the outcome problem in mind, our definition of $\text{MPA}_k\text{-TM}$, $\text{MBA}_k\text{-TM}$, $\text{PA}_k\text{-TM}$, $\text{BA}_k\text{-TM}$ accommodates only one \forall -player.

Let a *team strategy* for the Team T_1 be a mapping

$$\sigma : \bigcup_{i \in T_1} \text{POS}_i(p_0) \mapsto \text{POS}(p_0),$$

where for each Player $i \in T_1$, the restriction of σ to the domain $\text{POS}_i(p_0)$ is the strategy of Player i . We say that a play π is a *play by team strategy* σ if whenever π is a prefix of π' , and π is in the domain of σ , then $\sigma(\pi)$ is a (not necessarily proper) prefix of π' . In other words, the strategy σ explicitly dictates every move made by all players.

Strategy σ is a *winning strategy* (or *nonlosing strategy*) for Team T_1 if every play by strategy σ results in a loss for some player on Team T_0 (or does not result in a loss for any player on Team T_1).³ We can deduce that all plays by a winning strategy σ are finite because the play must terminate to result in a win for T_1 . However, a play by a nonlosing strategy may be potentially infinite.

It is self-evident that the outcome of a game is not affected if the players of Team T_0 are allowed to view the private portions of the players in Team T_1 . This is a direct consequence of the fact that a winning (or nonlosing) strategy of Team T_1 should work for all possible responses by Team T_0 . For each position $p \in \text{POS}$, let p^c be the position derived from p by making the private portions of Team T_1 's players accessible to Team T_0 . Suppose game G^c is so derived from game G . The following proposition holds.

PROPOSITION 3.3.1. Team T_1 has a winning strategy in G from initial position p_0 if and only if Team T_1 has a winning strategy from initial position p_0^c .

In order to discuss Markov strategies, we must introduce relevant notation.

DEFINITION 3.3.2. $\text{last}_m(\pi)$: Given a play π , and an integer $m \geq 0$, let $\text{last}_m(\pi)$ be the last m positions of π if m is less than the total number of positions in π (symbolically $m < |\pi|$), otherwise, let $\text{last}_m(\pi) = \pi$.

A strategy is Markov (m) for Player i if Player i 's moves only depend on the last m positions of any play. Formally, we say that strategy σ for Player i is Markov (m) if $\sigma(\pi) = \sigma(\pi')$ for all play prefixes $\pi, \pi' \in GT_i$ such that $\text{vis}_i(\text{last}_m(\pi)) = \text{vis}_i(\text{last}_m(\pi'))$. A strategy σ for a Team T_1 is Markov (m) if for every player $i \in T_1$ the strategy σ is Markov (m).

Let $G = (\text{POS}, \vdash, \text{vis}, T_1)$ be a game. We assume that the next move relation \vdash is represented by a *next-move transducer*, which is a D-TM transducer with input tape, and a one-way write-only output tape, and possibly some work tapes. The symbol alphabet Σ for this transducer contains the symbols appearing in the positions of POS . Given $p \in \Sigma^*$, the next-move transducer

³In the case of two-player games, the definitions of winning and nonlosing strategies are slightly different: strategy σ is a *winning strategy* for Player 1 if and only if every play by strategy σ is a win for Player 1, and σ is a *nonloss strategy* for Player 1 if and only if every play by strategy σ does not result in a loss for Player 1.

outputs, the set of moves $\{(p, p') \mid p \vdash p'\}$ if $p \in \text{POS}$, otherwise, if $p \notin \text{POS}$, it outputs a distinguished symbol $\$ \notin \Sigma$ to indicate illegal input. Any game G thus can be represented by a pair containing the next-move transducer and vis . Traditionally, a next-move relationship is required to be log-space computable.

Now, we can state the outcome problem.

DEFINITION 3.3.3. OUTCOME PROBLEM: *The win outcome problem for a game G is determining the existence of a winning (or winning Markov ($m(n)$)) strategy for Team T_1 from an initial position $p_0 \in \text{POS}$. The nonloss outcome problem for a game G is determining the existence of a nonlosing (or nonlosing Markov ($m(n)$)) strategy for Team T_1 from an initial position $p_0 \in \text{POS}$.*

DEFINITION OF SPACE AND TIME BOUNDED GAMES. We say a game has *space bound* $S(n)$ if the set of positions reached by a single move from any given position of length n can be computed in deterministic space $S(n)$. We say a game has *time bound* $T(n)$ if there are at most $T(n)$ positions reachable by any path of a game tree from any given position of length n .

3.4. Deterministic and Nondeterministic Games

This section defines special categories of games, for the sake of completeness. These games are important from a computational point of view, but are not the main focus of this paper. They are mentioned here for the sake of completeness.

DEFINITION 3.4.1. DETERMINISTIC PLAYER: *A Player i is a deterministic player if for each position it has only one move. Symbolically, for all $p \in \text{POS}_i$, there is at most one $p' \in \text{POS}$ such that $p \vdash p'$.*

There are two classes of games depending on the types of players involved: deterministic games and nondeterministic games.

DEFINITION 3.4.2. DETERMINISTIC GAME: *If all the players (in both teams are deterministic) then the game is also deterministic.*

An example of a deterministic game is “*Inni Minnie Miney Moe ...*”. This is a game often used by children to make deterministic choices that appear fair, but are in reality the initial position determines the outcome of the game.

DEFINITION 3.4.3. NONDETERMINISTIC GAME: *A game is nondeterministic if all the players in the (\forall) Team T_0 are deterministic, whereas the (\exists) players in the Team T_1 can nondeterministically select their move from a set of legal moves available on their turn (such that there is at least one position that allows more than one legal move).*

Examples of nondeterministic games are adversary games in which an adversary is required to respond accurately to the \exists -player’s guesses. For instance, the \exists -players have to deduce the integer that the adversary has been privately chosen. When any \exists -player guesses, and the adversary has to truthfully inform whether the guess was too high, too low, or correct. The process continues until the \exists -players deduce the correct integer or the maximum number of guesses have been exhausted.

Consequently, the only players that influence the result of the game are the existential members of T_1 , and this is why such games are classified as nondeterministic games. Nondeterministic games can be equivalently defined as games in which all *universal* (\forall) players are restricted to just one possible reply in every position. Consequently, all multiple choices occur at *existential* (\exists) nodes; and that can be modeled as nondeterministic choices.

Deterministic games involve exactly one choice for every nonterminating position. This is analogous to the deterministic transition function of Turing machine with exactly one transition for every nonterminating state. On the other hand, nondeterministic games involve an alternation between deterministic moves of universal player and nondeterministic moves of the existential player. This is analogous to the nondeterministic transition function of Turing machine with

exactly one transition for every universal state and nondeterministic transitions for existential moves. This observation is stated formally in the following proposition.

PROPOSITION 3.4.1. *Deterministic games can be mapped onto D-TM, whereas nondeterministic games can be mapped onto N-TM.*

DEFINITION 3.4.4. *SOLITAIRE GAME: A game is solitaire if on any play prefix on which Player 1 has made at least one move, the remaining moves of Player 0 are deterministic.*

Examples of solitaire games are Freecell, Mastermind, and Battleship.

3.5. Knowledge-Based Classifications of Games

In this section, we classify games according to the restrictions on the information visible to the players when they formulate their response.

DEFINITION 3.5.1. *PERFECT INFORMATION PLAYER: A Player $i \in T_1$ has perfect information of Team T_0 if Player i has at least as much visible information as any individual player of T_0 , i.e., $\text{vis}(i) \supseteq \text{vis}(j)$ for each $j \in T_0$.*

Since Team T_0 contains only universal player, we can model Team T_0 as a single universal player 0. Consequently, we can restate the above definition more succinctly in the following proposition.

PROPOSITION 3.5.1. *PERFECT INFORMATION PLAYER: A Player $i \in T_1$ has perfect information of Team T_0 if Player i has at least as much visible information as the universal player 0, i.e., $\text{vis}(i) \supseteq \text{vis}(0)$.*

DEFINITION 3.5.2. *GAME OF PERFECT INFORMATION: A game G is a perfect information game if all the players of T_1 have perfect information of every member of Team T_0 .*

Since there is no nontrivial information in private portions of a game of perfect information, we can reference all information as being in a common portion, and represent it by $\text{POS} \equiv \{0, \dots, k\} \times \text{CP}$. Go and Chess are examples of games of perfect information.

DEFINITION 3.5.3. *GAMES OF INCOMPLETE INFORMATION: Games in which at least one player of Team T_1 does not have perfect information are called games of incomplete information.*

A special category of games called *hierarchical games* are particularly interesting to us.

DEFINITION 3.5.4. *HIERARCHICAL GAME: A game G is hierarchical if the players of T_1 can be ordered as $\{i_1, \dots, i_h\}$ so that every player has at least all the visible information that the player indexed one greater than it has, i.e., for $1 \leq m < h$:*

$$\text{vis}(i_m) \supseteq \text{vis}(i_{m+1}).$$

Hierarchical games occur in a variety of situations such as multiplayer games of incomplete information of Reif and Peterson [31], where hierarchy of processes are generated by a sequence of fork operations.

DEFINITION 3.5.5. *BLINDFOLD: We say a (possibly hierarchical) game G is a blindfold game, if no player of T_1 can ever view any portion of a position which is modified by a player of T_0 in G .*

Note that if G is blindfold, we can disallow the players of T_0 rights to view any portions of the game which are viewed by any player of T_1 without modifying the outcome problems for the game G (by Proposition 3.3.1). The reason is simply that players in Team T_0 are universal (\forall) players. Consequently, Team T_1 has to win for all the strategies of the players of Team T_0 .

A classic example of a blindfold game is BLIND-PEEK [1,2]. However, Kriegspiel (Blindfold Chess) is not really considered blind by our definition, because a player can deduce certain characteristic of the board when the player attempts to make a move that is termed illegal due to information not previously known to the player. Consequently, Kriegspiel is categorized as a game of incomplete information.

3.6. Complexity Measures

Let \mathcal{F} be a set of functions on variable n . For each

$$\alpha \in \{D, N, A, PA, BA, MPA_k, MBA_k, PA_k, BA_k, MA_k\}$$

let $\alpha\text{SPACE}(\mathcal{F})$ be the class of languages accepted by α -TMs within some space bound in \mathcal{F} , and let $\alpha\text{TIME}(\mathcal{F})$ be the class of languages accepted by α -TMs within some time bound in \mathcal{F} .

4. DECISION ALGORITHMS FOR GAMES OF PERFECT INFORMATION

4.1. General Results

This section provides algorithms for deciding the outcome of games of perfect information. We assume that the perfect information game $G = (\text{POS}, \vdash, \text{vis}, T_1)$ has a space bound $S(n) \geq \log(n)$. We will let $\text{POS}(p_0)$ be the set of positions in POS reachable from p_0 by some sequence of moves, as defined by the relation \vdash , with space bound $S(n)$.

For noncomputer scientists, we review some results from two-player games.

PROPOSITION 4.1.1. *For any space bounded game with $S(n)$ that is greater or equal to $\log(n)$, there exists a constant $\alpha > 0$ such that $|\text{POS}(p_0)| \leq \alpha^{S(n)}$.*

PROOF. Since the game has a space bound of $S(n)$, the length of each position in $\text{POS}(p_0)$ is at most $S(n)$. Since we can treat the positions of POS as strings over finite alphabet, suppose the number of distinct symbols in the alphabet is α . Then there are at most $\alpha^{O(S(n))}$ distinct elements in $\text{POS}(p_0)$. ■

PROPOSITION 4.1.2. *The win, nonloss, and Markov ($m(n)$) outcome of any deterministic game with space bound $S(n)$ greater or equal to $\log(n)$, can be decided in deterministic space $S(n)$.*

PROOF. The proof follows from Definition 3.4.2 of deterministic games. ■

PROPOSITION 4.1.3. *The win and nonloss of any nondeterministic game with space bound $S(n)$ greater or equal to $\log(n)$, can be decided in nondeterministic space $S(n)$.*

PROOF. The proof for win outcome problem follows from Definition 3.4.3 of nondeterministic game. The proof for nonloss outcome follows from definitions and Immerman's result [32,33] nondeterministic space $O(S(n)) = \text{conondeterministic space } O(S(n))$. ■

Now we can apply Savitch's result [34] that $\text{NSPACE}(S(n)) = \text{DSPACE}(S(n)^2)$ and conclude the following corollary.

COROLLARY 4.1.4. *Any win, nonloss, and Markov($m(n)$) outcome of nondeterministic game with space bound $S(n)$ that is greater, or equal to $\log(n)$, can be decided in deterministic space $O(S(n)^2)$.*

4.2. Deciding Markov ($m(n)$) Outcome of a Space Bounded Game

We observe the following relationship between the space bound and the time bound with respect to the Markov ($m(n)$) outcome problem.

LEMMA 4.2.1. *If a team game G has space bound $S(n)$, that is greater or equal to $\log(n)$, with respect to the Markov ($m(n)$) outcome problem, then G has a time bound $2^{O(m(n)S(n))}$ with respect to the Markov ($m(n)$) outcome problem.*

PROOF. If a game G has space bound $S(n)$, that is greater or equal to $\log(n)$, then for some constant c , there can be at most $c^{m(n)S(n)}$ distinct plays with $m(n)$ positions or less (assuming finite set of alphabets). This is due to the fact that each position has an upper space bound of $S(n)$, and there are at most $m(n)$ positions in the play.

Therefore, we observe that there is a constant $c > 0$ such that if a Markov $(m(n))$ strategy σ induces a play π of length greater than $c^{m(n)S(n)}$, then π contains a repeated play sequence π_1 containing $m(n)$ moves. Consequently, π is of the form $\pi = \pi_0\pi_1\pi_2\pi_1\pi_3$, where π_0, π_2 are either play subsequences or empty strings, and π_3 is a play subsequence.

1. If the turn to move belongs to a player of Team T_1 at $\text{last}(\pi_1)$, then (in the play induced by strategy σ), the player would make the same choice that it made when it is confronted with the position $\text{last}(\pi_1)$ again. This would repeatedly force the same plays $\pi_1\pi_2$ in succession. As a result, π_3 has an infinite number of occurrences of π_1 . Consequently, π is infinite.
2. Otherwise, if the turn to move belongs to a player of T_0 at $\text{last}(\pi_1)$, then the universal team T_0 tries all of its moves again at the second occurrence of π_1 . Thus, there must be some play π' in which the same choice is repeated over and over. In other words, there is an infinite play π' such that $\pi' = \pi_0\pi_1\pi_2\pi_1\pi_2 \dots$ induced by strategy σ .

Since σ leads to infinite play in both cases, it cannot be a winning strategy in either case. Hence, a winning Markov $(m(n))$ strategy induces only plays of length $\leq c^{m(n)S(n)} = 2^{O(m(n)S(n))}$. It follows that G has a time bound of $2^{O(m(n)S(n))}$ with respect to Markov $(m(n))$ outcome. ■

Using Lemma 4.2.1 above, we can show the following.

THEOREM 4.2.1. *The Markov $(m(n))$ outcome of a game G with space bound $S(n) \geq \log(n)$ can be decided in deterministic space $2^{O(m(n)S(n))}$.*

PROOF. Using Lemma 4.2.1, we can deduce a winning Markov $(m(n))$ strategy σ need only be defined for plays of length at most $c^{m(n)S(n)}$ (for some constant $c > 0$). Therefore, we can verify a Markov $(m(n))$ strategy is a winning strategy within deterministic space at most $c^{m(n)S(n)} = 2^{O(m(n)S(n))}$. Moreover, σ can be represented by a function $\lambda : \text{POS}(p_0)^{m(n)} \mapsto \text{POS}(p_0)$ such that $\sigma(\pi) = \pi p$ if and only if $\lambda(\text{last}_{m(n)}(\pi)) = p$ for all π in the domain of σ .

Finally, there are at most $c^{S(n)c^{m(n)S(n)}}$ such functions λ because (by Proposition 4.1.1). The theorem follows. ■

We use $\text{MA}(m(n))_k - \text{SPACE}$ to denote space bound of a Markov Alternating machine which is abreast of last $m(n)$ moves. The subsequent corollary follows from Lemma 4.2.1 and Theorem 4.2.1

COROLLARY 4.2.1.

$$\begin{aligned} \text{MA}(m(n))_k - \text{SPACE}(S(n)) &\subseteq \text{ATIME}(\text{EXP}(m(n)S(n))) \\ \text{MA}(m(n))_k - \text{SPACE}(S(n)) &\subseteq \text{DSpace}(\text{EXP}(m(n)S(n))) \end{aligned}$$

PROOF. We know from [11] that $\text{ATIME}(T(n)) = \text{DSpace}(T(n))$. Lemma 4.2.1 shows that $\text{MA}(m(n))_k - \text{SPACE}(S(n)) \subseteq \text{ATIME}(\text{EXP}(O(m(n)S(n))))$. The corollary follows. ■

4.3. Deciding a Space Bounded Game of Perfect Information

Consider a game G of perfect information. We fix an initial position p_0 of length n .

LEMMA 4.3.1. *If Team T_1 has a winning strategy σ in a game G of perfect information, then Team T_1 has a winning Markov(1) strategy.*

PROOF. Let σ be any winning strategy. we define a strategy σ' to be a Markov(1) winning strategy by construction. For any play prefix π , $\sigma'(\pi) = \sigma(\pi')$, where π' is the lexically minimal play prefix such that $\text{last}(\pi) = \text{last}(\pi')$. It follows that σ' is a winning strategy if σ is a winning strategy. ■

Applying Lemma 4.3.1 above to Lemma 4.2.1, we derive the following lemma that shall prove to be useful in eliminating incomplete information in Sections 5.4 and 5.5.

LEMMA 4.3.2. *If a game G of perfect information has space bound $S(n) \geq \log n$, then G has time bound $2^{O(S(n))}$. Symbolically,*

$$\text{ASPACE}(S(n)) \subseteq \text{ATIME}(\text{EXP}(S(n))).$$

PROOF. Lemma 4.2.1 tells us that if a game G has space bound $S(n) \geq \log(n)$, with respect to the Markov $(m(n))$ outcome problem, then G has a time bound $2^{O(m(n)S(n))}$ with respect to the to the Markov $(m(n))$ outcome problem. Lemma 4.3.1 assures us that we can reduce $m(n)$ to 1. Consequently, G has a time bound $2^{O(S(n))}$. ■

Now, we will show that the following theorem can be derived from the above lemmas and proposition.

THEOREM 4.3.1. *The win and nonloss outcome of any game G of perfect information with space bound $S(n) \geq \log(n)$ can be decided in deterministic time $2^{O(S(n))}$. Symbolically,*

$$\text{ASPACE}(S(n)) \subseteq \text{DTIME}(\text{EXP}(S(n))).$$

PROOF. Consider a game $G = (\text{POS}, \vdash)$ of perfect information with space bound $S(n) \geq \log(n)$. We will assume that $S(n)$ is constructable, otherwise we try the following method with $S(n) = \{0, 1, \dots\}$.

Given an initial position p_0 of length n , we construct a set $\text{POS}(p_0)$ of all the positions reachable by moves of G from p_0 , with space $\leq S(n)$. Since G has position size bound $S(n)$, there must be a constant c (independent of n) such that $|\text{POS}(p_0)| \leq c^{S(n)}$. The theorem follows from the standard tree labeling algorithm (see Appendix A), which proves $\text{ASPACE}(O(S(n))) \subseteq \text{DTIME}(2^{O(S(n))})$ (i.e., every game of perfect information with space bound $O(S(n))$ can be decided in deterministic time $2^{O(S(n))}$).

By a similar procedure, we can develop a labeling corresponding to the nonloss outcome, and show the theorem to be true for nonloss outcome as well. Moreover, both labeling (for win and nonloss outcome) can be computed in deterministic time $2^{O(S(n))}$. ■

The following theorem is due to Chandra and Stockmeyer [35].

COROLLARY 4.3.1. (CS76). *For any $S(n) \geq \log(n)$:*

$$\begin{aligned} \text{ASPACE}(S(n)) &= \text{ATIME}(\text{EXP}(S(n))), \\ \text{ASPACE}(S(n)) &= \text{DTIME}(\text{EXP}(S(n))). \end{aligned}$$

PROOF. The proof follows from Lemma 4.3.2 and Theorem 4.3.1 in conjunction with the upper bound results of Chandra *et al.* [35]. ■

5. ELIMINATING INCOMPLETE INFORMATION

This section provides methods for converting games of incomplete information to perfect information games by making the incomplete information content explicit. Subsequently, we can use algorithms for deciding games of perfect information for deciding games of incomplete information. We assume that the game G of incomplete information $(\text{POS}, \vdash, \text{vis}, T_1)$ has a space bound $S(n) \geq \log n$. We show that the corresponding perfect information game will have a much higher space bound. We shall commence by reviewing two-player games, and then extend the method to multiplayer games.

5.1. Unraveling Information in a Two-Player Game

Reif [1,2] presents a powerset construction for transforming a two-player game $G = (\text{POS}, \vdash)$ of incomplete information into a corresponding game $G^+ = (\text{POS}^+, \vdash^+)$ of perfect information,

whose positions are sets of positions of G . The construction is reminiscent of the subset construction in finite automata (FA) to prove the relation between DFA (deterministic FA) and NFA (nondeterministic FA). The resulting decision algorithms are characterized by exponential blow up in space complexity. We prove in the companion paper [4] that this exponential blow up in space complexity must occur in the worst case.

Since the procedure described of unraveling incomplete information from two-player games will assist us in understanding the corresponding procedure for multiplayer games, we will summarize the powerset construction of Reif [1,2] for sake of completeness.

ALGORITHM 1.

1. Fix some initial position $p_0 \in \text{POS}$.
2. Assuming that the set of positions reachable from p_0 is finite, for each play prefix π of G we construct a position $P(\pi)$ of G^+ with common portion the set $\{\text{last}(\pi') \mid \pi' \text{ is a prefix with } \text{vis}_1(\pi) = \text{vis}_1(\pi')\}$. This is the set of all possible positions after π from the perspective of Player 1 (viewing only the information visible to Player 1). Let private portions of $P(\pi)$ be NULL to make G^+ a game of perfect information.
3. In $P(\pi)$, let the next player to move be the same as the next player to move in $\text{last}(\pi)$. Observe that $P(\pi) = P(\pi')$ if and only if $\text{vis}_1(\pi) = \text{vis}_1(\pi')$.
4. If Player 1 is supposed to move next, and π is some play prefix of G with $\text{last}(\pi) \in W$, then we do not allow any next-move from $P(\pi) \in \text{POS}^+$. Otherwise, let $P(\pi) \vdash^+ P(\pi')$ be the move of G^+ if π, π' are play prefixes of G and π' is a child of π in G . Thus, moves from a position $P(\pi)$ of G^+ simulate all possible moves of G from $\text{last}(\pi)$.
5. Finally, let $P(p_0)$ be the initial position of G^+ .

The following theorem is also due to [1,2].

THEOREM 5.1.1. (See [1,2].) *Player 1 has a winning strategy in G from position p_0 if and only if Player 1 has a winning strategy in G^+ from $P(p_0)$.*

PROOF. We can prove this by establishing a one-to-one correspondence between winning strategies of G and winning Markov strategies of G^+ . The proof is a special case of Lemma 5.2.1, which is proven later in this paper. The details can also be found in Reif's paper [1,2]. ■

Since there are no more than $2^{O(S(n))}$ positions reachable from any starting position p_0 , the size of any position P of G^+ is at most $2^{O(S(n))}$. So, Reif's algorithm [1,2] for eliminating incomplete information can be executed by an A-TM with space bound $2^{O(S(n))}$.

REMARK 5.1.1. Reif's discussion [1,2] continues to show that the outcome of any two-player game of incomplete information with space bound $S(n)$ can be decided by alternating Turing machine with space bound $2^{O(S(n))}$, and it presents an algorithm to that effect. Moreover, Theorem 4.3.1 implies that the outcome of any two-player game of incomplete information with space bound $S(n)$ can be decided in deterministic time $2^{2^{O(S(n))}}$.

5.2. Unraveling Information in a Hierarchical Multiplayer Game

If a multiplayer game with three or more players is not hierarchical, then it can be undecidable because the universal player can deceive the existential players to play a game which can be mapped to the halting problem as shown by Peterson and Reif [1,2].

Consider a hierarchical game $G = (\text{POS}, \vdash, \text{vis}, T_1)$. Fix some initial position $p_0 \in \text{POS}$ for Teams T_0, T_1 . We present a method for transforming G to a game of perfect information assuming the set of positions reachable from p_0 is finite. First, we need to state a couple of definitions.

DEFINITION 5.2.1. CLIQUE: *We define a clique to be maximal set of players of a game with exactly the same rights to view components of positions.*

DEFINITION 5.2.2. k : Let $\phi_0 \subseteq T_1$ be the maximal set of players with perfect information of T_0 (hence, no player in $T_1 - \phi_0$ has perfect information). We define the parameter k to be the number of cliques of $T_1 - \phi_0$.

We will perform the transformation in stages. At every stage, we effectively eliminate from the game the incomplete information associated with a clique ϕ of players in Team T_1 . The players still remain in the game. However, the information content is unraveled. In the resulting game G^1 , the players of ϕ have perfect information. After a finite number of applications of this treatment (to all the cliques in the game G), we succeed in defining a game of perfect information which corresponds to the original game G . We extend the definition of vis to encompass a set as argument. For any set ϕ of players of G , let

$$\text{vis}(\phi) = \bigcup_{i \in \phi} \text{vis}(i).$$

PROPOSITION 5.2.1. If we are given a game G , we can derive a corresponding game G^0 by substituting for each clique of T_1 a single “super-player” with the same information access rights as the players of that clique, and allowing the “super-player” to move the same as the players of that clique.

Let ϕ_0 be the clique with perfect information of T_0 . Let ϕ_1 be the clique of $T_1 - \phi_0$ with “most knowledge”, i.e., $\text{vis}(\phi_1) \supseteq \text{vis}(i)$ for all players $i \in T_1 - \phi_0$. We shall unravel the incomplete information of clique ϕ_1 , thus deriving a new hierarchical game $G^1 = (\text{POS}^+, \vdash^+, \text{vis}^+, T_1)$, where G^1 has the same Teams T_0, T_1 , but $\phi_0, \phi_1 \subseteq T_1$ have perfect information. We define a new variable vis^+ to denote the rights to knowledge of various information of the individual players. $\text{vis}^+(i)$ denotes the rights of Player i in the new game G^1 , and is defined as follows: $\text{vis}^+(i) = \{1, \dots, r+1\}$ for each player $i \in \phi_1 \cup \phi_0$, and $\text{vis}^+(i) = \text{vis}(i)$ for all other players. Observe that vis^+ is different from vis , and that by this assignment of rights, the players of ϕ_1 and ϕ_0 have perfect information. For each position $p = (i, p[1], \dots, p[r]) \in \text{POS}$, and every play prefix π of G , with $p = \text{last}(\pi)$, we derive a new position $P(\pi) = (i, p[1], \dots, p[r], p[r+1]) \in \text{POS}^+$, where

1. the $r+1^{\text{st}}$ component of $P(\pi)$ (i.e., $p[r+1]$) is defined to be $\{\text{last}(\pi') \mid \pi' \text{ is a play prefix with } \text{vis}_{\phi_1}(\pi) = \text{vis}_{\phi_1}(\pi')\}$,
2. $\text{vis}_j(\pi) = \text{vis}_{\phi_1}(\pi)$ for any $j \in \phi_1$.

Intuitively, the $r+1^{\text{st}}$ component of $P(\pi)$ is the set of all possible positions after π from the perspective of the players of ϕ_1 , each one of who is allowed to view only $\text{vis}_{\phi_1}(\pi)$. Observe that $P(\pi) = P(\pi')$ if and only if $\text{vis}_{\phi_1}(\pi) = \text{vis}_{\phi_1}(\pi')$. We do not allow any legal next moves from $P(\pi) \in \text{POS}^+$ if it is Team T_1 's turn to move and π is some play prefix with $\text{last}(\pi) \in W$. Hence, Team T_0 wins at $P(\pi)$ for any π which is winning for Team T_0 . Otherwise, let $P \vdash^+ P'$ be a move of G^1 if $P = P(\pi)$ and $P' = P(\pi')$ for some child π' of π in the original game G . By this definition, a move of G^1 from P simulates all possible moves of G from any position $\text{last}(\pi)$, where $P = P(\pi)$. Finally, we need to fix $P_0 = P(p_0)$ to be the initial position of G^1 .

REMARK 5.2.1. Observe that G^1 is space bounded $2^{O(S(n))}$ where n is the length of the original position p_0 in G , and $S(n)$ is the space bound of game G . Consequently, we have exponential blow-up in space complexity.

Now, we turn our attention to the relationship between the game G and the derived game G^1 .

LEMMA 5.2.1. Team T_1 has a winning (nonloss) strategy in G from the initial position p_0 if and only if Team T_1 has a winning (nonloss) strategy in G^1 from the initial position P_0 .

PROOF. We will prove this by establishing a one-to-one correspondence between winning (non-loss) strategies in G , and winning (nonloss) strategies in G^1 . Let us first establish a correspondence between the winning strategies of the two games G and G^1 .

Suppose σ is a winning strategy for Team T_1 in G . Consider all play prefixes π^+ of G^1 in which the turn to move belongs to one of the players in Team T_1 . For each such play, we

define $\sigma^+(\pi^+) = \pi^+P(\sigma(\pi))$ where π is the play prefix (of G) from which π^+ is derived (i.e., $P(\pi) = \text{last}(\pi^+)$). This simply states that the method of selecting the next move at π^+ by σ^+ is looking up the move selected by σ for the corresponding play prefix π in G for that situation, and then translating that move to the corresponding move in G^1 . Now, in anticipation of a contradiction, suppose σ^+ induces a play π^+ of G^1 which is not a win for Team T_1 in G^1 . If so, then there is a corresponding play π of G induced by σ where $\text{last}(\pi)$ is contained in the $r + 1^{\text{st}}$ component of $P(\pi)$, and some such π is not a winning play for Team T_1 in G . However, this contradicts our supposition that σ is a winning strategy in G . Consequently, σ^+ is a winning strategy in G^1 .

Now to prove the implication in other direction, suppose σ^+ be a winning strategy for Team T_1 in G^1 . Consider all play prefixes π of G in which the turn belongs to one of the players in Team T_1 . For each such play, we define $\sigma(\pi)$ to be the child of π such that $\sigma^+(\pi^+) = \pi^+P(\sigma(\pi))$, where π^+ is the play prefix of G^1 which corresponds to the play prefix π of G . Now, in anticipation of a contradiction, suppose σ induces a play π of G which is not a win for Team T_1 in G . If so, then there is a corresponding play π^+ of G^1 induced by σ^+ where $P(\pi)$ is π^+ , and some such π^+ is not a winning play for Team T_1 in G^1 . However, this contradicts our supposition that σ^+ is a winning strategy in G^1 . Consequently, σ is a winning strategy in G . In a similar fashion, we can develop one-to-one correspondence between the nonloss strategies of the two games G and G^1 . The lemma follows. ■

DEFINITION 5.2.3. $\text{EXP}_m(\mathcal{F})$: For set of functions \mathcal{F} , $\text{EXP}_m(\mathcal{F})$ is the tower of m repeated exponentials of $f(n) \in \mathcal{F}$. Recursively, $\text{EXP}_m(\mathcal{F})$ is defined as follows:

$$\begin{aligned}\text{EXP}_1(\mathcal{F}) &= \left\{ c^{f(n)} \mid c > 0 \text{ and } f(n) \in \mathcal{F} \right\}, \\ \text{EXP}_m(\mathcal{F}) &= \left\{ c^{\text{EXP}_{m-1}(\mathcal{F})} \mid c > 0 \right\}, \quad \text{for } m > 1.\end{aligned}$$

If \mathcal{F} consists of just one function $f(n)$, we drop the set notation. For example, $\text{EXP}_m(f(n))$ is defined as $\text{EXP}_m(\{f(n)\})$ where $\{f(n)\}$ stands for the set with singleton element $f(n)$. Also, note that $\text{EXP}(\mathcal{F}) = \text{EXP}_1(\mathcal{F})$ by default.

The following theorems relate the space bound of a hierarchical game to its deterministic time bound.

THEOREM 5.2.1. Given a hierarchical k -player game G with space bound $S(n) \geq \log(n)$. G can be transformed into a game G^* such that Team T_1 has a winning (nonlosing) strategy from initial position p_0 in G if and only if it has a winning (nonlosing) strategy from the position corresponding to p_0 in G^* . The win outcome and the nonloss outcome of G^* can be decided in deterministic space $\text{EXP}_k(S(n))$. Symbolically,

$$\text{PA}_k\text{SPACE}(S(n)) \subseteq \text{ASPACE}(\text{EXP}_k(S(n))).$$

PROOF. G^* is obtained by applying the procedure of eliminating incomplete information for each clique in succession.

Fix an initial position p_0 of G with length n . If there are any cliques with two or more players, we replace these cliques by “super-players” (cf. Proposition 5.2.1). The i^{th} clique is represented by \exists_i -player. Now, the construction method of Lemma 5.2.1 can be applied k times to yield a game G^* of perfect information. The \exists_1 -player knows everything all other players know except the \forall -player. Therefore, we create intermediate configurations which combine the current state of the \exists_1 -player with all possible states of \forall -player. Each of these new configuration will have the space bound $2^{O(S(n))}$.

Subsequently, we create new configurations that represent the \exists_2 -player’s private states and the set of possible states of \exists_1 -player (which now include the set of possible states of \forall -player). Each of these new configuration will have the space bound $2^{2^{O(S(n))}}$.

The process is repeated until we have only \exists_k -player's private states and the set of possible states of the \exists_{k-1} -player, which includes the set of possible states of the \exists_{k-2} -player, which includes the set of possible states of the \exists_{k-3} -player, etc.

Removal of each clique contributes a further exponential (cf. Remark 5.2.1). The space bound of this game would be $\text{EXP}_k(O(S(n)))$. Team T_1 would have a winning strategy the initial position P_0^* (of G^*) if and only if Team T_1 has a winning strategy from the corresponding initial position p_0 of G (as a consequence of k repeated applications of Lemma 5.2.1.). ■

THEOREM 5.2.2. *The win outcome and the nonloss outcome of any k -player hierarchical game G of incomplete information with space bound $S(n) \geq \log(n)$ can be decided in deterministic time $\text{EXP}_{k+1}(S(n))$. Symbolically,*

$$\text{PA}_k\text{SPACE}(S(n)) \subseteq \text{DTIME}(\text{EXP}_{k+1}(S(n))).$$

PROOF. Suppose, we have a game G with space bound $S(n)$. From Theorem 5.2.1, the space bound of the corresponding perfect information game G^* would be $S^*(n) = \text{EXP}_k(O(S(n)))$. Theorem 4.3.1 states that a game with space bound $S^*(n)$ can be decided in time $2^{O(S^*(n))}$. Hence, we can conclude that G^* can be decided in deterministic time $2^{\text{EXP}_k(O(S(n)))}$, i.e., deterministic time $\text{EXP}_{k+1}(O(S(n)))$. ■

The following corollary is a consequence of Theorems 5.2.1 and 5.2.2.

COROLLARY 5.2.1. *For any $S(n) \geq \log(n)$ and $k \geq 0$,*

$$\begin{aligned} \text{PA}_k\text{SPACE}(S(n)) &\subseteq \text{ASPACE}(\text{EXP}_k(S(n))), \\ \text{PA}_k\text{SPACE}(S(n)) &\subseteq \text{DTIME}(\text{EXP}_{k+1}(S(n))). \end{aligned}$$

THEOREM 5.2.3. *The win (nonloss) outcome of any hierarchical blindfold game G with space bound $S(n) \geq \log(n)$ can be decided in nondeterministic (conondeterministic) space $\text{EXP}_k(O(S(n)))$.*

PROOF. Fix an initial position p_0 (in G) of length n . As in Theorem 5.2.1, the construction of Lemma 5.2.1 can be applied k times to yield a game G^* of perfect information with space bound $\text{EXP}_k(O(S(n)))$. However, since G is blindfold, any winning or nonloss strategy of Team T_1 can be made oblivious to moves of Team T_0 . Using this fact, G^* can be transformed into a nondeterministic game G_N^* by allowing Team T_1 to nondeterministically choose a strategy σ_1 (this can be done nondeterministically in space $\text{EXP}_k(O(S(n)))$), and then allowing Team T_0 to iteratively choose each possible strategy σ_0 of T_0 (this can be done deterministically in space $\text{EXP}_k(O(S(n)))$). During the simulated play of game G^* , Team T_0 must move by strategy σ_0 , and the players of T_1 must move by strategy σ_1 .

By Proposition 4.1.2, the win (nonloss) outcome of the game G_N^* can be decided nondeterministic (conondeterministic) space $\text{EXP}_k(O(S(n)))$. ■

COROLLARY 5.2.2. *By the result of Immerman [32,33], the nonloss outcome of the game G_N^* can be decided nondeterministic space $\text{EXP}_k(O(S(n)))$.*

PROOF. The corollary follows from Immerman's result [32,33] that:

$$\text{NSPACE}(O(S(n))) = \text{co-NSPACE}(O(S(n))).$$

The above Theorem 5.2.3 and Corollary 5.2.2 lead to the following result.

COROLLARY 5.2.3. *For any $S(n) \geq \log(n)$ and $k \geq 0$:*

$$\text{BA}_k\text{SPACE}(S(n)) \subseteq \text{NSPACE}(\text{EXP}_k(S(n)))$$

5.3. Decision Algorithms for Games with Space and Alternation Bounds

In this section, we study games with both space and alternation bounds. To facilitate our discussion, we define several predicates: $\text{PATH}(p, p')$, $\text{APATH}(p, p')$, $\text{DIVERGE}(p)$ are defined in Figures 3, 4, and 5, respectively. Function $\text{PATH}(p, p')$ returns TRUE if and only if there is a sequence of moves from p to p' with no alternations. Function $\text{APATH}(p, p')$ returns TRUE if and only if there is a sequence of moves from p to p' with no alternations, except the last move must be an alternation. Function $\text{DIVERGE}(p)$ returns TRUE if and only if there is an infinite nonalternating play from p , or if there is a move from p to a position which exceeds the space bound $S(n)$. We assume that all the transitory positions visited are in $\text{POS}(p_0)$ in all these function computations.

LEMMA 5.3.1. *We can see that the predicates $\text{PATH}(p, p')$, $\text{APATH}(p, p')$, $\text{DIVERGE}(p)$, and membership in $\text{POS}(p_0)$ can be decided in nondeterministic space $S(n)$, and deterministic space $S(n)^2$.*

PROOF. This is a consequence of the space bound of $S(n)$ in the computation of each move. From Savitch's theorem [34], we know that $\text{NSPACE}(S(n)) = \text{DSPACE}(S(n)^2)$. Hence, $\text{PATH}(p, p')$, $\text{APATH}(p, p')$, $\text{DIVERGE}(p)$, and membership in $\text{POS}(p_0)$ can be decided in deterministic space $S(n)^2$. ■

Now, we describe a recursive function $\text{DECIDE}(p, a)$ in Figure 8 that employs two functions $\text{DECIDE-universal}(p, a)$ and $\text{DECIDE-existential}(p, a)$ (illustrated in Figures 6 and 7). DECIDE returns TRUE if Team T_1 has a winning strategy from position p where all plays have less than a alternations, and returns FALSE otherwise. In other words, $\text{DECIDE}(p, A(n))$ decides the outcome of G (with alternation bound $A(n)$).

THEOREM 5.3.1. *(See [11].) If a perfect information game G has space bound $S(n) \geq \log(n)$ and alternation bound $A(n)$, then the win outcome and the nonloss outcome of G can be computed in deterministic space $(A(n) + S(n))S(n)$.*

PROOF. We can utilize the aforementioned algorithm for deciding acceptance of alternating Turing machines with space and alternation bounds. Let $p_0 \in \text{POS}$ be an initial position of length n . We assume $S(n)$ is constructable (otherwise we can try $S(n) = 0, 1, 2, \dots$). Let $\text{POS}(p_0) \subseteq \text{POS}$ be exactly the set of positions reachable from p_0 within space $S(n)$. Each invocation of the functions Decide-Existential and Decide-Universal and can be implemented in deterministic space $S(n)$, and the depth of recursive calls is at most $A(n)$ (because $A(n)$ is a nonnegative integer which is decreased by 1 every time the depth of recursion is increased by 1). Moreover, $S(n)^2$ global space is required to compute the predicates APATH and DIVERGE . Consequently, the total space requirement is $(A(n) + S(n))S(n)$. The procedure for deciding nonloss outcome of G is similar, except that we replace

```

if (DIVERGE( $p$ ) = TRUE)
  then
    return(FALSE)
with
if (DIVERGE( $p$ ) = TRUE)
  then
    return(TRUE)

```

This is done because an infinite play is not a “win”, but it is certainly “nonloss”. Of course, the rest of the procedure remains the same. ■

The following corollary is a consequence of Theorem 5.3.1.

COROLLARY 5.3.1. *(See [36,37].) For $S(n) \geq \log(n)$ and $A(n) \geq 0$,*

$$\text{ASPACE}, \text{ALT}(S(n), A(n)) \subseteq \text{DSPACE}((A(n) + S(n))S(n)).$$

```

function PATH( $p, p'$ ): returns Boolean
  Input: Positions  $p$  and  $p'$  in  $\text{POS}(p_0)$ 
  Output: TRUE if there is a sequence of moves from  $p$  to  $p'$  with no alternations,
           and where all the transitory positions visited are in  $\text{POS}(p_0)$ .
           and returns FALSE otherwise.

begin
  if  $p' \in \{p \vdash^* p' \mid \text{there are no alternations}\}$ 
    then return(TRUE)
    else return(FALSE);
end;

```

Figure 3. Function $\text{PATH}(p, p')$.

```

function APATH( $p, p'$ ): returns Boolean
  Input: Positions  $p$  and  $p'$  in  $\text{POS}(p_0)$ 
  Output: TRUE if and only if there is a sequence of moves from  $p$  to  $p'$  with no alternations,
           except the last move is an alternation, and where all the transitory positions visited
           are in  $\text{POS}(p_0)$ .

begin
  if ( $p'' \in \{p \vdash^* p' \mid \text{there are no alternations}\}$ )
    and ( $p'' \vdash p'$  with an alternation)
    then return (TRUE)
    else return(FALSE);
end;

```

Figure 4. Function $\text{APATH}(p, p')$.

```

function DIVERGE( $p$ ): returns Boolean
  Input: Position  $p \in \text{POS}(p_0)$ 
  Output: TRUE if and only if there an infinite nonalternating play from  $p$ ,
           or if there is a move from  $p$  to a position which exceeds the space bound  $S(n)$ .

begin
  if ( $(\exists p' \in \text{POS}(p_0))$  and ( $\text{PATH}(p, p')$ )
    and ( $(\text{PATH}(p', p'))$ 
    or ( $\exists p'' \in \text{POS}(p_0): p' \vdash p''$  and space bound  $> S(n)$ )))
    then return(TRUE)
    else return(FALSE);
end;

```

Figure 5. Function $\text{DIVERGE}(p)$.

THEOREM 5.3.2. *Consider a hierarchical game of incomplete information G with k distinct cliques has a space bound $S(n) \geq \log(n)$ and an alternation bound $A(n)$. The win and nonloss outcomes of G can be decided in deterministic space $(A(n) + 1)\text{EXP}_k(S(n))$.*

PROOF. Let G be a hierarchical game of incomplete information with k distinct cliques. Suppose G has a space bound of $S(n) \geq \log(n)$ and alternation bound of $A(n)$. Fix an initial position p_0 of length n . By Theorem 5.2.1, G can be transformed into a game G^* such that Team T_1 has a winning (nonlosing) strategy from initial position p_0 in G if and only if it has

```

function Decide-Universal( $p, a$ ): returns Boolean
  Input: Position  $p \in \text{POS}(p_0)$ , and Alternation bound  $a$ 
  Output: TRUE if and only if  $T_1$  has a winning strategy from  $p$ 
           with an alternation bound of  $a$  and a space bound  $S(n)$ .

  COMMENT: This block decides the moves of Team  $T_0$ 
  begin
  if (DIVERGE ( $p$ ) = TRUE)
    then
      return(FALSE)
    else
      if ( $a$  is 0)
        then
          return(TRUE)
        else
          COMMENT: Deterministically check each position  $p' \in \text{POS}(p_0)$ 
                   such that APATH( $p, p'$ ) holds. If Decide-Existential( $p, a - 1$ )
                   is TRUE for all such  $p'$  then return TRUE.
          begin
          set  $flag$  to TRUE;
          for all  $p' \in \{p' \in \text{POS} \mid \text{APATH}(p, p') = \text{TRUE}\}$ 
            begin
            if  $flag$  is TRUE
              then
                set  $flag$  to Decide-Existential( $p', a - 1$ );
            end;
          return( $flag$ );
          end
        end
      end
    end
  end

```

Figure 6. Function Decide-Universal.

a winning (nonlosing) strategy from the position corresponding to p_0 in G^* . The win outcome and the nonloss outcome of G^* can be decided in deterministic space bound $\text{EXP}_k(S(n))$, and the alternation bound remains $A(n)$. Consequently, by Theorem 5.3.1, G has deterministic space bound of $(A(n) + \text{EXP}_k(S(n)))\text{EXP}_k(S(n))$. This is the same as $(A(n) + 1)\text{EXP}_k(S(n))$. ■

The following corollary is a direct consequence of Theorem 5.3.2.

COROLLARY 5.3.2. *For $S(n) \geq \log(n)$ and $A(n) \geq 0$,*

$$\text{PA}_k \text{ SPACE}, \text{ALT}(S(n), A(n)) \subseteq \text{DSPACE}(A(n) + 1)\text{EXP}_k(S(n)).$$

5.4. Decision for Games with Time and Branch Bounds

In this section, we consider the games with time and branch bounds. Recall that a game G has a time bound $T(n)$ if the outcome problem can be solved in at most $T(n)$ moves for every play sequence in the game tree. Analogously, a game G has a branch bound b if for each position $p \in \text{POS} : |\{p' \mid p \vdash p'\}| \leq b$. In other words, at any position in POS , there are at most b choices for the next transition. Consider a game G of incomplete information with time bound $T(n)$. In order to decide the win outcome of G , we only need to choose each strategy σ for Team T_1 , and verify that Team T_1 wins for any play π induced by σ . Observe that each such play has at most $T(n)$ moves.

```

function Decide-Existential( $p, a$ ): returns Boolean
  Input: Position  $p \in \text{POS}(p_0)$ , and Alternation bound  $a$ 
  Output: TRUE if and only if  $T_1$  has a winning strategy from  $p$ 
           with an alternation bound of  $a$  and a space bound  $S(n)$ .

COMMENT: This block decides the moves of Team  $T_1$ 
begin
  if (DIVERGE ( $p$ ) = TRUE)
    then
      return(FALSE)
    else
      if ( $a$  is 0)
        then
          return(FALSE)
        else
          COMMENT: Deterministically check each position  $p' \in \text{POS}(p_0)$ 
                   such that APATH( $p, p'$ ) holds. If Decide-Universal( $p, a - 1$ )
                   is TRUE for any such  $p'$  then return TRUE.
          begin
            set flag to FALSE;
            for all  $\{p' \in \text{POS} \mid \text{APATH}(p, p') = \text{TRUE}\}$ 
              begin
                if flag is FALSE
                  then
                    set flag to Decide-Universal( $p', a - 1$ );
              end;
            return(flag);
          end
        end
      end
    end
  end
end

```

Figure 7. Function Decide-Existential.

```

function Decide( $p, a$ ): returns Boolean
  Input: Position  $p \in \text{POS}(p_0)$ , and Alternation bound  $a$ 
  Output: TRUE if and only if  $T_1$  has a winning strategy from  $p$ 
           with an alternation bound of  $a$  and a space bound  $S(n)$ .

COMMENT: This function calls Decide-Universal or Decide-Existential
         depending upon which team moves first.
begin
  if ( $p \in \text{POS}_0$ )
    then
      Decide-Universal( $p, a$ )
    else
      Decide-Existential( $p, a$ )
    end
  end
end

```

Figure 8. Function Decide.

THEOREM 5.4.1. *The win outcome and Markov ($m(n)$) outcome of any multiplayer game of incomplete information with time bound $T(n) \geq n$, and branch bound b can be decided in deterministic space $T(n) \log(b)$.*

PROOF. Consider the aforementioned game G of incomplete information with time bound $S(n) \geq n$. We assume that $T(n)$ is constructable, otherwise we try $T(n) = 0, 1, 2, \dots$

As described above, we only need to choose each possible strategy σ for Team T_1 , and verify that Team T_1 wins for any play π induced by some such strategy σ . Observe that each such play has at most $T(n)$ moves. Consequently, it can be stored in $T(n) \log(b)$ bits, because we need $\log(b)$ distinct bits to indicate one of b branches for each move in a play of at most $T(n)$ moves. Then, using this storage, the determination that a given strategy σ is winning can be made by computing the outcome labeling of the game tree within deterministic space $T(n)$. Moreover, this space also suffices to deterministically verify that σ is Markov ($m(n)$). The theorem follows. ■

The following corollary is an immediate consequence of Theorem 5.4.1.

COROLLARY 5.4.1. *For any $T(n) > n$ and a maximum branch factor $b = O(2^{nT(n)})$,*

$$\text{PA}_k \text{ TIME}(T(n)) \subseteq \text{DSpace}(O(T(n))).$$

6. CONCLUSION

This paper has provided algorithms to decide the outcome of any multiplayer game of incomplete information. The algorithms are shown to be optimal in our companion paper [4]. Multiplayer games of incomplete information can be undecidable in general, unless the information is hierarchically arranged (as defined earlier in this paper). Hierarchical multiplayer games of incomplete information are decidable, and each additional *clique* (subset of players with same information) compounds the complexity of the outcome problem by a further exponential. Consequently, if multiplayer games of incomplete information with k cliques have a space bound of $S(n)$, then their outcome is k repeated exponentials harder than games of complete information with space bound $S(n)$. Blindfold games are related to the nondeterministic space in a similar way. The main results are summarized in Corollaries 5.2.1 and 5.2.3, and are for any $S(n) \geq \log(n)$ and $k \geq 0$,

$$\text{PA}_k \text{ SPACE}(S(n)) \subseteq \text{DTIME}(\text{EXP}_{k+1}(S(n))),$$

$$\text{BA}_k \text{ SPACE}(S(n)) \subseteq \text{NSPACE}(\text{EXP}_k(S(n))).$$

If in addition to space bound $S(n) \geq \log(n)$, alternation bound $A(n) \geq 0$ is also present then Corollary 5.3.2 states:

$$\text{PA}_k \text{ SPACE, ALT}(S(n), A(n)) \subseteq \text{DSpace}(A(n) + 1) \text{EXP}_k(S(n)).$$

Our algorithms for deciding the outcome problem are based on the extension of Reif's method [1,2] of eliminating incomplete information (for two-player games) to multiplayer games. Our method can prove to be useful in other problems involving ambiguity and incomplete information; e.g., natural language understanding. Time bounded games are shown not to exhibit such complexity of towering exponentials. In this paper, we provided an algorithm to decide $T(n)$ -time bounded games in deterministic space $T(n)$ (independent of the number of players and cliques). The main result appears in Corollary 5.4.1, and is reiterated below: for any $T(n) > n$:

$$\text{PA}_k \text{ TIME}(T(n)) \subseteq \text{DSpace}(T(n)).$$

Our decision algorithms yield upper bounds that are shown to be asymptotically optimal by providing matching lower bounds in the second part [4] of this pair of papers. Our algorithm for space bounded games is utilized to decide certain formulae in multiprocessor logic of incomplete information. It would be interesting and useful to investigate techniques for (heuristically and otherwise) reducing the search to decide games of incomplete information.

APPENDIX

TREE LABELING ARGUMENT

We will construct a sequence of mappings from $\text{POS}(p_0) \mapsto \{\text{TRUE}, \text{FALSE}\}$. Initially, let $l(p) = \text{FALSE}$ for each $p \in \text{POS}(p_0)$. We also introduce a labeling $f(l)$ such that for each $p \in \text{POS}(p_0)$, we define:

$$\begin{aligned} f(l)(p) &= \text{False}, & \text{if } p \in W \wedge \text{POS}_1, \\ &\quad \vee_{p \vdash p'} l(p'), & \text{if } p \in \text{POS}_1 - W, \\ &= \text{True}, & \text{if } p \in W \wedge \text{POS}_0, \\ &\quad \wedge_{p \vdash p'} l(p'), & \text{if } p \in \text{POS}_0 - W. \end{aligned}$$

Let $a = |\text{POS}(p_0)|$. Now, by Proposition 4.1.1, $|\text{POS}(p_0)| = 2^{O(S(n))}$. $f(l)$ can be computed from l in deterministic time a^2 . Furthermore, we define l^* to be the labeling derived by repeatedly applying f to l^4 until there is no change induced by an application of f .⁵ Note that sequence of mappings $l_0, f(l_0), f(f(l_0)), \dots$ is monotone in the sense that $f^m(l_0)(p) = \text{TRUE}$ implies $f^{m+1}(l_0)(p) = \text{TRUE}$ for all $m \geq 0$ and $p \in \text{POS}(p_0)$. Hence, the computation converges to fixed point of f ⁶ after at most $a = |\text{POS}(p_0)|$ iterations, and each iteration can be completed in $2^{O(S(n))}$ deterministic time (since we have assumed that next-move relations in any game is computable in linear space). Consequently, a total time of $2^{O(S(n))}$ is required. We can show one-to-one correspondence between Markov strategies σ of player and the labelings l^* constructed by the above process. More specifically, the positions mapped by l^* to TRUE correspond to the positions appearing as winning plays induced by some strategy σ , and the positions mapped by l^* to FALSE correspond to the positions not appearing as winning plays induced by any strategy σ . Consequently, we can deduce that $l^*(p_0)$ is TRUE if and only if Team T_1 has a winning Markov strategy for p_0 . We show this by induction on the length of plays. The basis is trivial because it is obviously true for plays of length 0. By induction hypothesis, assume for any $m \geq 0$, and for any $p \in \text{POS}(p_0)$ that $f^m(l_0)(p) = \text{TRUE}$ if and only if Team T_1 has a winning strategy from p where all plays are of length smaller than m . If $p \in W$, then by definition $f^{m+1}(l_0)(p) = \text{TRUE}$ if and only if the turn to move at p belongs to a player on Team T_0 , and that is true if and only if Team T_1 has a trivial winning strategy from p . If the turn to move at $p \notin W$ belongs to a player on Team T_0 , then by definition $f^{m+1}(l_0)(p) = \text{TRUE}$ if and only if $f^m(l_0)(p') = \text{TRUE}$ for all $p' \in \text{POS}(p_0)$ such that $p \vdash p'$, and by induction this is true if and only if Team T_1 has a winning strategy of length less than m from each $p' \in \text{POS}(p_0)$ such that $p \vdash p'$. And, if a player of T_1 has to move at $p \notin W$, then by definition $f^{m+1}(l_0)(p) = \text{TRUE}$ if and only if for some $p' \in \text{POS}(p_0)$ such that $p \vdash p'$, and by induction hypothesis this holds if and only if Team T_1 has a winning strategy of length less than m from some $p' \in \text{POS}(p_0)$ such that $p \vdash p'$. In either case, $f^{m+1}(l_0)(p) = \text{TRUE}$ if and only if Team T_1 has a winning strategy of length less than $m+1$ from p .

REFERENCES

1. J.H. Reif, Universal games of incomplete information, In *Proceedings, 17th Annual ACM Symposium on Theory of Computing*, pp. 288–308, (May 1979).
2. J.H. Reif, The complexity of two-player games of incomplete information, *J. Comp. Sys. Sci.* **29** (2), 274–301, (1984).
3. G.L. Peterson and J.H. Reif, Multiple-person alternation, In *Proceedings of 20th IEEE Symposium on Foundations of Computer Science*, pp. 348–363, (October 1979).
4. G.L. Peterson, J.H. Reif and S. Azhar, Lower bounds for multiplayer non-cooperative games of incomplete information, *Computers Math. Applic.* **41** (7/8), 957–992, (2001); <http://www.cs.duke.edu/~reif/paper/games/bounds/bounds.pdf>.

⁴For all $q \in \text{POS}(p_0)$ l initialized as $l(q) = \text{FALSE}$

⁵This can be done by repeatedly applying F starting with the initial mapping $l(\text{POS}(p_0)) = \text{FALSE}$, until a fixed point is reached. This computation of l^* is similar to a procedure used by Chandra, Kozen and Stockmeyer [11] to decide acceptance of space bounded alternating Turing machine.

⁶ $l^* = f^a(l_0)$ where l_0 is what l was initialized to.

5. J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Third Edition, Princeton University Press, Princeton, NJ, (1953).
6. R.J. Aumann and S. Hart, *Handbook of Game Theory, Volume 1*, Elsevier Science, (1992).
7. M. Ben-Or and N. Linial, Collective coin flipping, robust voting schemes and minima of Banzhaf values, In *Proceedings of 26th Annual IEEE Symposium on Foundations of Computer Science*, pp. 408–416, (October 1985).
8. M. Ben-Or and N. Linial, *Advances in Computing Research 5: Randomness and Computation*, (Edited by D. Micali), JAI, Greenwich, CT, (1989).
9. S. Azhar, A. McLennan and J.H. Reif, *Computation of Equilibria in Noncooperative Games*, TR CS-1991-36, Computer Science Department, <http://www.cs.duke.edu/~reif/paper/games/mixed/mixed.pdf>, Duke University, Durham, NC, (October 1991).
10. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, (1979).
11. A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer, Alternation, *J. of ACM* **28** (1), 114–133, (1981).
12. T.J. Schaefer, On the complexity of some two player perfect information games, *J. Comput. System Sci.* **16** (2), 185–225, (1978).
13. S. Even and R.E. Tarjan, A combinatorial problem which is complete in polynomial space, *J. of ACM* **23**, 710–719, (1976).
14. A.S. Fraenkel, M.R. Garey, D.S. Johnson, T.J. Schaefer and Y. Yesha, The complexity of checkers on an $N \times N$ board—Preliminary report, In *Proceedings of 19th Annual IEEE Symposium on Foundations of Computer Science*, pp. 55–64, (October 1978).
15. J.M. Robson, N by N checkers is Exptime complete, *SIAM J. of Comput.* **13**, 252–267, (1984).
16. D. Lichtenstein and M. Sipser, October, In *Proceedings of 19th Annual IEEE Symposium on Foundations of Computer Science*, pp. 48–54, (1978).
17. D. Lichtenstein and M. Sipser, Go is PSPACE hard, *J. of ACM* **27**, 393–401, (1980).
18. A.S. Fraenkel and D. Lichtenstein, Computing a perfect strategy $N \times N$ chess requires time exponential in N , In *Proceedings, 8th Int'l Coll. Automata, Lang., & Programming*, pp. 278–293, (July 1981).
19. G.L. Peterson, Succinct representation, random strings, and complexity classes, In *Proceedings of 21st IEEE Symposium on Foundations of Computer Science*, pp. 86–95, (Oct 1980).
20. L.J. Stockmeyer and A.K. Chandra, Provably difficult combinatorial games, *SIAM J. Comput.* **8** (2), 151–174, (1979).
21. C.H. Papadimitriou, Games against nature, In *Proceedings of 24th IEEE Symposium on Foundations of Computer Science*, pp. 446–450, (October 1983).
22. C.H. Papadimitriou, Games against nature, *J. Comput. System Sci.* **31**, 288–301, (1985).
23. L. Babai, Trading group theory for randomness, In *Proceedings of 17th Annual ACM Symposium on Theory of Computing*, pp. 421–429, (May 1985).
24. S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive protocols, In *Proceedings, 17th Annual ACM Symposium on Theory of Computing*, pp. 291–304, (May 1985).
25. S. Goldwasser and M. Sipser, Public coins versus private coins in interactive proof systems, In *Advances in Computing Research 5: Randomness and Computation*, Edited by D. Micali, JAI, Greenwich, CT, (1989).
26. A. Shamir, $IP = PSPACE$, In *Proceedings, 31st Annual IEEE Symposium on Foundations of Computer Science*, pp. 11–15, (1990).
27. R.E. Ladner and J.K. Norman, Solitaire automata, *J. Comp. System Sci.* **30** (1), 116–129, (1985).
28. N.D. Jones, Blindfold game are harder than games of perfect information, *Bull. Euro. Assoc. for Theoret. Comp. Sci.* **6**, 4–7, (1978).
29. L.J. Stockmeyer and A.R. Meyer, Word problems requiring exponential time: Preliminary report, In *Proceedings, 5th Annual ACM Symposium on Theory of Computing*, pp. 1–9, (May 1973).
30. G.L. Peterson, Press-ups is PSPACE-complete, TR, Computer Science Department, University of Rochester, Rochester, NY, (1979).
31. J.H. Reif and G.L. Peterson, A dynamic logic of multiprocessing with incomplete information, In *Proceedings, 7th Annual ACM Symposium on Principles of Programming Languages*, pp. 193–202, Las Vegas, NV, (April 1980).
32. N. Immerman, Languages which capture complexity classes, In *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, pp. 347–354, (May 1983).
33. N. Immerman, Non-deterministic space is closed under complementation, *SIAM J. Comp* **17**, 935–938, (1988).
34. W.J. Savitch, Relationship between deterministic and nondeterministic tape complexities, *J. Comp. Sci.* **4**, 177–192, (1970).
35. A.K. Chandra and L.J. Stockmeyer, Alternation, In *Proceedings of 17th Annual IEEE Symposium on Foundations of Computer Science*, pp. 98–108, (October 1976).
36. A. Borodin, Complexity classes of recursive functions and the existence of complexity gaps, In *Proceedings, 1st Annual ACM Symposium on Theory of Computing*, pp. 67–78, (May 1969).
37. A. Borodin, Complexity classes of recursive functions and the existence of complexity gaps, *J. of ACM* **19**, 158–174, (1972).