Lower Bounds 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—This paper (see also [1]) extends the *alternating Turing machine* (A-TM) of Chandra, Kozen and Stockmeyer [2], the private and the blind alternating machines of Reif [3,4] to model multiplayer games of incomplete information. We use these machines to provide matching lower bounds for our decision algorithms described in our companion paper [5]. We also apply multiple person alternation to other machine types.

We show that multiplayer games of incomplete information can be undecidable in general. However, one form of incomplete information games that is decidable we term as hierarchical games (defined later in this paper). In hierarchical multiplayer games, each additional *clique* (subset of players with same information) increases the complexity of the outcome problem by a further exponential. Consequently, if a multiplayer game of incomplete information with k cliques has a space bound of $S(n)$, then its outcome can be k repeated exponentials harder than games of complete information with the same space bound $S(n)$. This paper proves that this exponential blow-up must occur in the worst case.

We define TEAM-PRIVATE-PEEK and TEAM-BLIND-PEEK, extending the previous models of PEEK. These new games can be shown to be complete for their respective classes. We use these games to establish lower bounds on complexity of multiplayer games of incomplete information and blindfold multiplayer games.

We analyze the time bounded alternating machines, and conclude that time is not a very critical resource for multiplayer alternation. We also show DQBF (a variant of QBF) to be complete in *NEXPTIME*.

Keywords—Alternation, Algorithms, 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/bounds/bounds.pdf`.

Preprint of paper appearing in *Computers and Mathematics* with Applications, Volume 41, April 2001, pp 957-992.

1. INTRODUCTION

1.1. Motivation

The Turing machine was presented as a simple mathematical model of computation, which can model a general purpose (deterministic) computer in terms of elementary operations. The definition of the Turing machine accentuated the development of computability theory by formalization of algorithmic procedures. Subsequently, several other paradigms of computations (nondeterminism, parallel, etc.) have been introduced, and corresponding models of computations (the nondeterministic Turing machine, the parallel random access machine, etc.) have been developed accordingly.

The need for a formal computational model to address the computational aspects of games was fulfilled by Chandra, Kozen and Stockmeyer [2] with the *alternating Turing machine* (A-TM). Now, the A-TM has become the most widely accepted fundamental game theoretic model of computation. Subsequently, this A-TM model has been extended and enhanced to model more intricate games. Reif [3,4] extended the A-TM model to incorporate private and blindfold two-player games by introducing the private alternating Turing machine (PA-TM) and the blindfold alternating Turing machine (BA-TM), respectively.

In this paper, we introduce the multiplayer private alternating Turing machine (MPA_k-TM) and the multiplayer blindfold alternating Turing machine (MBA_k-TM) to model private and blindfold multiplayer games, respectively. We also define the PA_k-TM and the BA_k-TM to remove the over-generality of the MPA_k-TM and the MBA_k-TM, respectively.

Computer scientists are interested in developing and analyzing new game theoretic algorithms, as well as modeling computational paradigms with game theory. This paper is particularly concerned with the lower bound analysis of the game theoretic problems. We provide lower bounds for solving the outcome problem of multiplayer games of incomplete information. These lower bounds demonstrate that the corresponding decision algorithms (to decide the outcome) presented in the companion paper by Peterson, Reif and Azhar [5] (see also [1]) are asymptotically optimal.

All these new types of machines provide a deeper insight into the relationships between time and space bounded computation. Different types of games correspond to different modes of computation as shown in the Table 1. In particular, it is fascinating that the simplest type of game (solitaire, perfect information, unique next move) corresponds to our most natural notion of computation (deterministic). On the other hand, some of the most intricate game corresponds to novel and abstract models of computation.

Table 1. Comparing computation and games.

Computation Mode	Game
deterministic	solitaire, perfect information, unique next move
nondeterministic	solitaire, perfect information, open next move
alternation	two-player, perfect information
private alternation	two-player, incomplete information
multiplayer alternation	multiplayer, incomplete information

1.2. Games Theory in Computer Science

Reif [3,4] defines a two-player game as a 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 which are *private* to one of the players, whereas the rest are *common* portions accessible to both players.

The generalization of 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 finite alphabet. Every position p may contain certain information which is *private* to some player. The remaining information is *common*, and may be viewed by more than one player. The set of legal next-moves for a given player must be independent of the information which is inaccessible to him. These rights remain unmodified throughout the course of the game.

In any game, every player plays according to a strategy which dictates a single next move to the player for each and every possible sequence of previous moves that can legally occur. (Such strategies that dictate a single next move to the player for every possible sequence of previous moves that can legally occur 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 [6] for more detailed treatment of the complexity of finding such mixed strategies.) The 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 we analyze the complexity of these algorithms as a function of Team T_1's size. On the other hand, we model Team T_0 as a single player.

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, respectively) under all circumstances?"

In addition to the outcome problem, this paper also considers the following *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?" The Markov(1) outcome problem is considered by Peterson and Reif [1].

A game has perfect information if no position has any private component, and a game has incomplete information if there are certain private components to the game. Furthermore, if Team T_0 never modifies any portion of the position visible to players of Team T_1 then the game is categorized as a blindfold game.

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. In general, multiplayer games of incomplete information can be undecidable. However, they are decidable in at least one case, which is that of *hierarchical* multiplayer games of incomplete information. Hierarchical multiplayer games are multiplayer games in which the information is hierarchically arranged, i.e., players on Team T_1 (of k members) can be arranged such that all information visible to Player i is also visible to Player $i-1$ for all existential players $(i \in \{2,3,\ldots,k\})$. Our formulation of hierarchical games as decidable does not preclude other formulations of multiplayer games that are decidable.

In hierarchical multiplayer games, each additional *clique* (subset of players with same information) increases the complexity of the outcome problem by a further exponential. Consequently, if a multiplayer game of incomplete information with k cliques has a space bound of $S(n)$, then its outcome can be k repeated exponentials harder than games of complete information with the same space bound $S(n)$. This paper proves that this exponential blow-up must occur in the worst case.

[1] We use the two terms, multiplayer games and team games, interchangeably.

The results obtained from studying space bounded hierarchical multiplayer alternating machines give a more clear understanding of hyper-exponential complexity classes. In particular, the elementary recursive languages are characterized by a class of linear space bounded multiplayer alternating machines. This enables us to exhibit natural problems with super-complexity, but that is another topic, and should be addressed in another paper.

1.3. Main Results

In this paper, we define computation machines for private and blind multiplayer games. We introduce two new machines: the multiplayer private alternating machine (MPA-TM) and the multiplayer blind alternating machine (MBA-TM). To remove the evident over-generality of the MPA-TM and the MBA-TM, we introduce the PA-TM and the BA-TM, which are restricted versions of the respective machines. These machines are used to demonstrate the relations between time and space hierarchies.

The MPA_k-TM is a machine model which corresponds to a $(k+1)$-player multiplayer (team) game of incomplete information with k existential players and one universal player. The MBA_k-TM is derived from the MPA_k-TM by disallowing the $\forall$-player from writing on any resource which is readable by any $\exists$-player. In particular, for $k = 1$ the MPA_k-TM (i.e., the MPA_1-TM) bears resemblance to the PA-TM. For $k = 1$ the MBA_k-TM (i.e., the MBA_1-TM) bears resemblance to the BA-TM. We also introduce the PA_k-TM and the BA_k-TM to remove the over-generality of the MPA_k-TM and the MBA_k-TM, respectively.

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

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

let α-*SPACE* $(\mathcal{F})$ be the class of languages accepted by the corresponding α-*TMs* within some space bound in $\mathcal{F}$, and let α-*TIME* $(\mathcal{F})$ be the class of languages accepted by the corresponding α-*TMs* within some time bound in $\mathcal{F}$. Since the complexity of these games involve multiple exponentials, we need to develop notation to represent multiple exponentials.

DEFINITION 1.3.1. *$EXP_m(\mathcal{F})$. For set of functions $\mathcal{F}$, $EXP_m(\mathcal{F})$ is the tower of m repeated exponential of $f(n) \in \mathcal{F}$. Recursively, $EXP_m(\mathcal{F})$ is defined as follows:*

$$EXP_1(\mathcal{F}) = \left\{ c^{f(n)} \mid c > 0 \text{ and } f(n) \in \mathcal{F} \right\},$$
$$EXP_m(\mathcal{F}) = \left\{ c^{EXP_{m-1}(\mathcal{F})} \mid c > 0 \right\}, \qquad \text{for } m > 1.$$

If $\mathcal{F}$ consists of just one function $f(n)$, we drop the set notation. For example $EXP_m(f(n))$ is defined as $EXP_m(\{f(n)\})$ where $\{f(n)\}$ stands for the set with singleton element $f(n)$. Also, note that we consider $EXP(\mathcal{F}) = EXP_1(\mathcal{F})$ by default (and perhaps a slight abuse of notation).

Chandra, Kozen, and Stockmeyer [2] relate the time and space complexity of the A-TM and the D-TM as follows.

For $S(n) \geq \log n$

$$ASPACE(S(n)) = DTIME(EXP(S(n))),$$
$$ATIME(EXP(S(n))) = DSPACE(EXP(S(n))).$$

Reif [3,4] relates the time and space complexity of the D-TM with that the PA-TM and the BA-TM as follows:

$$BASPACE(S(n)) = ATIME(EXP(S(n)))$$
$$= DSPACE(EXP(S(n))),$$

$$\begin{aligned} PASPACE(S(n)) &= ASPACE(EXP(S(n))) \\ &= DTIME(EXP(EXP(S(n)))), \\ BATIME(EXP(S(n))) &= ATIME(EXP(S(n))) \\ &= DSPACE(EXP(S(n))), \\ PATIME(EXP(S(n))) &= ATIME(EXP(S(n))) \\ &= DSPACE(EXP(S(n))). \end{aligned}$$

It also follows that:

$$BATIME(EXP(S(n))) = PATIME(EXP(S(n))).$$

Hierarchical multiplayer games are used to illustrate the relationship between time and space complexity for k players with several restrictions which are defined later. The results are summarized below.

PROPOSITION 1.3.1. PRIVATE ALTERNATION. *For* $S(n) \geq \log n$

$$PA_k\text{-}SPACE(S(n)) = DTIME(EXP_{k+1}(S(n))).$$

For $T(n) \geq n$, *if* $k = 1$ *then*

$$NSPACE(T(n)) \subseteq PA_1\text{-}TIME\left(T(n)^2\right) \subseteq DSPACE\left(T(n)^2\right).$$

For $T(n) \geq n$, *if* $k \geq 2$ *then*

$$PA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))).$$

PROPOSITION 1.3.2. BLIND ALTERNATION. *For* $S(n) \geq \log n$

$$BA_k\text{-}SPACE(S(n)) = NSPACE(EXP_k(S(n))) = DSPACE(EXP_k(S(n))).$$

For $T(n) \geq n$, *if* $k = 1$ *then*

$$BA_1\text{-}TIME(T(n)) = \Sigma_2^{T(n)}.$$

For $T(n) \geq n$, *if* $k = 2$ *then*

$$NSPACE(T(n)) \subseteq BA_2\text{-}TIME\left(T(n)^2\right) \subseteq DSPACE\left(T(n)^2\right).$$

For $T(n) \geq n$, *if* $k \geq 3$ *then*

$$BA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))).$$

Figure 1 illustrates the relations of the space complexity of the multiplayer alternation machines with deterministic time and space. The deterministic complexity hierarchy shifts by exactly one level with alternation [2]. It shifts one level further when private alternation is introduced. And for each additional player added to private alternation, it shifts one more level.

1.4. Overview

Section 2 reviews computational models for two-player games, and then enhances them to incorporate multiplayer games. It defines the MPA$_k$-TM, the MBA$_k$-TM, the PA$_k$-TM, and the BA$_k$-TM. It formally defines complexity measures, and proves the speed-up and space compression. Section 2 also describes the concept of universal games.

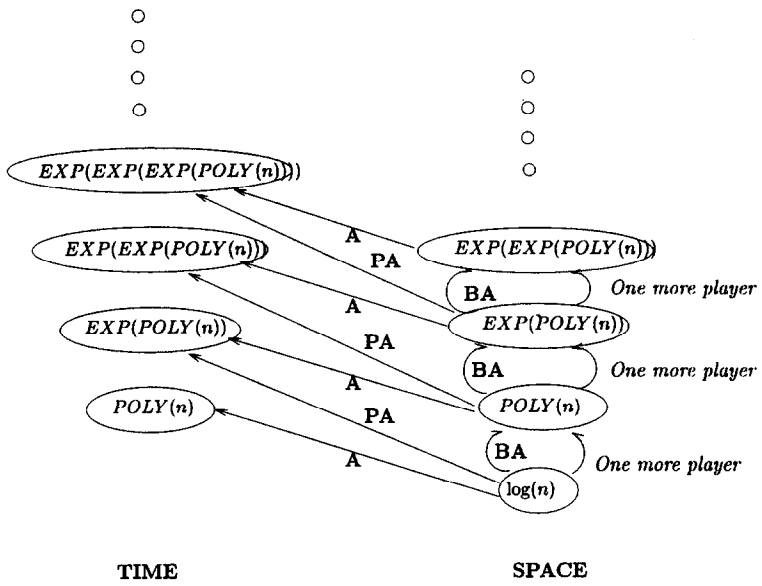


Figure 1. Complexity shifts for an α-TM's from α-SPACE to deterministic time and space starting from the two-player game of incomplete information. α = "A" for alternation; α = "PA" for private alternation; α = "BA" for blind alternation; α = "$k-1$" for private/blind alternation with *one less player*.

Section 3 focuses on providing matching lower bounds for the multiplayer game algorithms. It employs proof techniques using strategy counter game (SCG), a game tailored for lower bound proofs. It illustrates SCG based proofs for previous results of Reif [3,4]. Subsequently, it applies SCG techniques to multiplayer games.

Section 4 reviews PEEK games and introduces two new variants: TEAM-PRIVATE-PEEK and TEAM-BLIND-PEEK. These games are shown to be universal for their respective classes by using propositional formulae.

Section 5 is concerned with time bounded machines. It derives lower bounds for time bounded machines, and shows that DQBF to be *NEXPTIME*-complete. It concludes by highlighting the relationship between blindfold and incomplete information games.

Section 6 extends the idea of multiplayer alternation to finite state automata, pushdown store automata, and Markov machines.

2. GAMES AND COMPUTATIONAL MODELS

2.1. Computational Models for Two-Player Games

We shall describe how our machines models are derived from the deterministic Turing machine (D-TM) by successive refinement.

A deterministic Turing machine (D-TM) is defined as a nine-tuple

$$(t, Q, \Sigma, \Gamma, \delta, q_0, \#, b, F),$$

where $t \in Z^+$ is the number of work tapes; Q is a finite set of *states*; Σ is a finite set of *input symbols*; Γ is a finite set of *tape symbols*; δ maps $(Q \times \Gamma^t)$ to $(Q \times \Gamma^t \times \{\text{'left','right','static'}\}^t)$ is the *transition relation* which defines a unique move for every element of $(Q \times \Gamma^t)$; $q_0 \in Q$ is a fixed *initial state*; $\# \in \Gamma - \Sigma$ is a distinguished *end marker*; $b \in \Gamma - \Sigma$ is a distinguished *blank symbol*; $F \subseteq Q$ is the set of accepting states.

A nondeterministic Turing machine (N-TM) is a nine-tuple

$$(t, Q, \Sigma, \Gamma, \delta, q_0, \#, b, F)$$

which is identical to the D-TM except that the next move function δ specifies some subset of

$$2^{(Q\times\Gamma^t\times\{\text{'left'},\text{'right'},\text{'static'}\}^t)}$$

for every $(Q \times \Gamma^t)$. Consequently, there may be any one of several possible next moves from a given state (unlike the D-TM, which is constrained to a deterministic fixed move). Furthermore, the N-TM accepts an input if there is any sequence of transitions leads to an accepting state.

An alternating Turing machine (A-TM) is defined as a ten-tuple

$$(t, Q, U, \Sigma, \Gamma, \delta, q_0, \#, b, g),$$

where t, Q, Σ, Γ, q_0, $\#$, and b are as defined above, and the other symbols are defined as follows.

- U is a set of universal states ($U \subseteq Q$).
- $Q - U$ is a set of existential states.
- g maps Q onto $\{\wedge, \vee$, accept, reject$\}$, such that for $q \in U : g(q) = \wedge$ to map universal states to conjunction, and for $q \in Q-U : g(q) = \vee$ to map existential states to disjunction.
- δ defines some subset of $2^{(Q\times\Gamma^t\times\{\text{'left'},\text{'right'},\text{'static'}\}^t)}$ for every $(Q \times \Gamma^t)$.

The A-TM models a two-player game in which the existential player has to pick the move that will lead to acceptance for all possible moves of the universal player. The alternation takes place as the existential states (identified with Player 1) alternate with the universal states (identified with Player 0) during the computation. Two-player games of perfect information are related in this way to the alternating Turing machines (A-TM) of Chandra *et al.* [2] in which existential and universal states alternate.

In two-player games of perfect information, the main interest is in solving the outcome problem for the existential ($\exists$) player, i.e., finding winning strategies for $\exists$-player. The A-TM accepts an input, corresponding to an initial position, if the existential player has a winning (or a nonloss) strategy which would guarantee to a win (or not lead to a loss, respectively) 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 [7], Even and Tarjan [8], Fraenkel *et al.* [9], Robson [10], Lichtenstein and Sipser [11,12], Fraenkel and Lichtenstein [13], Peterson [14]. Stockmeyer and Chandra [15] introduce a game called PEEK, and prove that PEEK 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.

Games are intimately related to models of computation in general. 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. Game G accepts input ω depending on the outcome of the game from an initial position encoded by ω. The correspondence between games and languages is illustrated in Table 2.

Table 2. Natural analogies between games and languages, with examples.

LANGUAGE	GAME	EXAMPLE
string	concrete game	a blindfold chess endgame
language	game	blindfold chess
class of languages	game type	two-player, incomplete information

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 nondeterministic Turing machine (N-TM) is mapped to games of perfect information with Player 0 absent because there are no universal states. The 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.

A PA-TM is derived from the A-TM by not allowing the existential ($\exists$) player to access some of the work tapes that are writable by the universal ($\forall$) player. These tapes are considered private to the universal ($\forall$) player. This is another way of restricting the existential ($\exists$) player from viewing some of the universal ($\forall$) player's moves.

A BA-TM is derived from the PA-TM by not allowing the universal ($\forall$) player write access to work tapes which can be read by the existential ($\exists$) players. This is another way of restricting the existential ($\exists$) player from viewing any of the universal ($\forall$) player's moves.

The PA-TM and the BA-TM model two-player games of incomplete information (e.g., rummy) and two-player blindfold games (e.g., BLIND-PEEK [3,4]), respectively.

2.2. Computational Models for Multiplayer Games

We define multiplayer alternation by building upon the notion of the nondeterministic Turing machine. These steps can be viewed as an algorithm for defining a game-like machine.

DEFINITION 2.2.1. MULTIPLAYER ALTERNATION. *Multiplayer alternation is modeled by an enhanced N-TM N, where:*

1. *we partition the states into distinct subsets such that each player is assigned exactly one subset of states;*
2. *we assign players various rights to view and modify tapes and portions of states;*
3. *then, we introduce a multiplayer game, called a computation game, which is used to define acceptance for the resulting multiplayer alternating Turing machine;*
4. *we divide the players into two teams; Team T_0 consists of universal ($\forall$) players and Team T_1 consists of existential ($\exists$) players.*

We could have built upon any kind of nondeterministic machine, not necessarily a nondeterministic Turing machine, to lead to the corresponding alternating machine. For example, if we begin with a nondeterministic random access machine (N-RAM), then we must assign players rights to view and modify various memory registers rather than tapes. Acceptance of resulting multiplayer alternating machine would be defined again by a computation game.

Now we can define an MPA$_k$-TM based on an enhanced N-TM as follows.

DEFINITION 2.2.2. MPA$_k$-TM: k+1-PERSON ALTERNATING TURING MACHINE. *A k+1-player alternating Turing machine MPA$_k$-TM is defined as a mapping VIS from $\{0,\ldots,k\}$, corresponding to the k players, to (not necessarily proper) subsets of $\{1,\ldots,r\}$ (corresponding to the position information) and a ten-tuple*

$$(t, Q, U, \Sigma, \Gamma, \delta, q_0, \#, b, g),$$

where

$t \in Z^+$ *is the number of work tapes;*
Q *is a finite set of states;*
U *is a set of universal states* ($U \subseteq Q$);
Σ *is a finite set of input symbols;*

Γ *is a finite set of tape symbols;*
δ *maps* $(Q \times \Gamma^t)$ *to* $(Q \times \Gamma^t \times \{\text{'left', 'right','static'}\}^t)$ *is the transition relation which defines a unique move for every element of* $(Q \times \Gamma^t)$;
$q_0 \in Q$ *is a fixed initial state;*
$\# \in \Gamma - \Sigma$ *is a distinguished end marker;*
$b \in \Gamma - \Sigma$ *is a distinguished blank symbol;*
g *maps* Q *onto* $\{\wedge, \vee, \text{accept, reject}\}$, *such that for* $q \in U : g(q) = \wedge$, *for* $q \in Q - U : g(q) = \vee$, *for final accepting states:* $g(q) =$ accept, *and for final rejecting states:* $g(q) =$ reject.

$(t, Q, U, \Sigma, \Gamma, \delta, q_0, \#, b, g)$ *is a nondeterministic Turing machine with the following restrictions.*

1. *We require the set of states* $Q \subseteq \{0, \ldots, k\} \times Q_1 \times \cdots \times Q_s$ *where* $Q_1, \ldots, Q_s$ *are finite sets denoting state components.*
2. *A configuration is a sequence* $(i, q_1, \ldots, q_s, (X_1, Y_1), \ldots, (X_t, Y_t))$ *where* $q = (i, q_1, \ldots, q_s)$ *is the current state, and* $(X_1, Y_1), \ldots, (X_t, Y_t)$ *are the current contents of tape* $1, \ldots, t$, *respectively. There are a total of* $1 + s + t$ *components in a configuration. Integer* i $(0 \leq i \leq k)$ *identifies the player who has to move next. The next* s *items keep track of state components* $q_i \in Q_i$ *for* $1 \leq i \leq s$. *The number of state components* (s) *may be different from the number of players. The remaining* t *components keep track of the tape configurations.* X_m *precedes the head of tape* m. *The head of tape* m *scans the first symbol of* Y_m.
3. *The initial configuration has the initial state* $q_0 \in Q$, *and the tape contents initialized as for the nondeterministic Turing machine* N.
4. *Let POS be the set of all configurations. The next-move relation* ($\vdash \subseteq POS \times POS$) *is a binary relation on configurations defined in the usual manner from the transition function* δ.
5. *Finally, we require the computation game* $G^M = (POS, \vdash, VIS, T_1)$ *to be a* $k + 1$ *player game satisfying Axioms 1 and 2 of Section 2.4 (with Teams* $T_0 = \{0\}$ *and* $T_1 = \{1, \ldots, k\}$). *These axioms simply prevent a player from modifying the information which is invisible to him, and inhibit him from using the invisible information in formulating his strategy.*
6. *We say that* M *accepts* $\omega \in \Sigma^*$ *if Team* $T_1 = \{1, \ldots, k\}$ *has a winning strategy in* G^M *from the initial configuration. Let the language of* M *be* $L(M) = \{\omega \in \Sigma^* \mid \omega$ *is accepted by* $M\}$.

Let $r = s + t$ *(number of state components plus number of tapes). VIS is a mapping from* $\{0, \ldots, k\}$, *corresponding to the* k *players, to (not necessarily proper) subsets of* $\{1, \ldots, r\}$ *(corresponding to the position information), i.e.,* $VIS : \{0, \ldots, k\} \mapsto 2^{\{1, \ldots, r\}}$. *Inclusion of* s_i *in* $VIS(j)$ *denotes right of Player* j *to view state* i, *and inclusion of an integer* t *in* $VIS(j)$ *denotes right of Player* j *to view tape* t.

A four player private alternating Turing machine is shown in Figure 2. The visibility rights of the players are specified by *VIS*, where

$$VIS(0) = \{s_0, s_1, s_2, s_3, 1, 2, 3\}, \qquad VIS(1) = \{s_1, s_2, s_3, 2, 3\},$$
$$VIS(2) = \{s_2, s_3, 3\}, \qquad VIS(0) = \{s_3, 3\},$$

with an additional constraint that Player 3 cannot write to Tape 3 (even though he can read from that tape).

A multiplayer blind alternating Turing machine (MBA$_k$-TM) with $k+1$ players is a restricted MPA$_k$-TM in which Player T_0 is not allowed to modify any information visible to players of Team T_1.

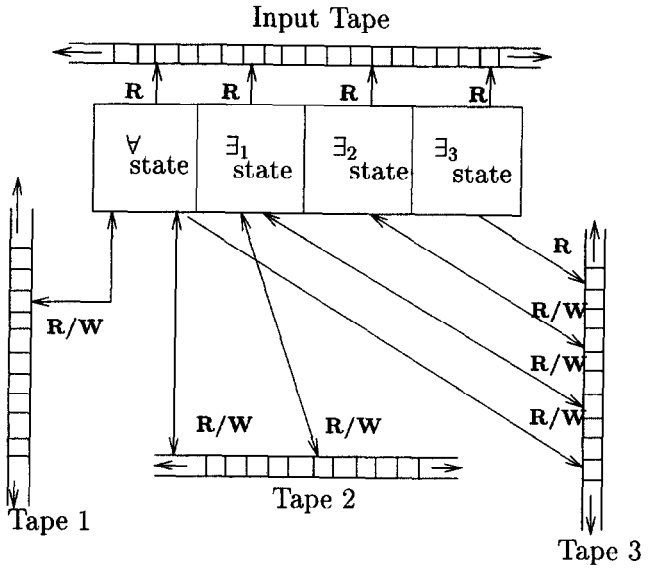


Figure 2. Multiplayer private alternating Turing machine.

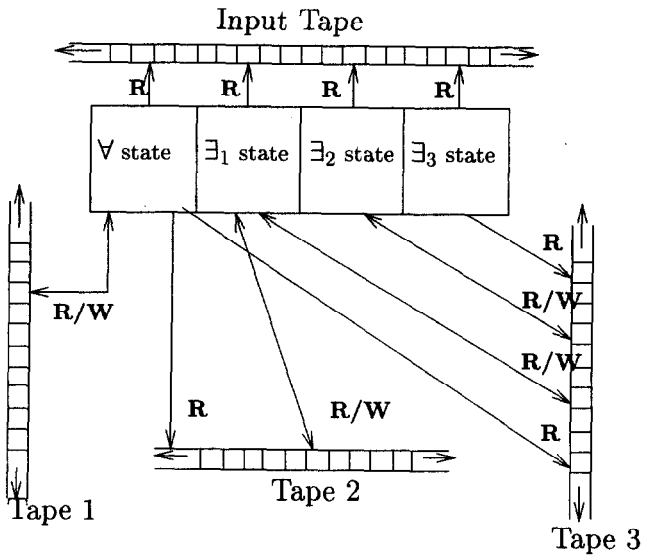


Figure 3. The multiplayer blind alternating Turing machine.

A four player blind alternating Turing machine is shown in Figure 3. The visibility rights of the players are specified by *VIS*, where

$$VIS(0) = \{s_0, s_1, s_2, s_3, 1, 2, 3\}, \qquad VIS(1) = \{s_1, s_2, s_3, 2, 3\},$$
$$VIS(2) = \{s_2, s_3, 3\}, \qquad VIS(3) = \{s_3, 3\},$$

with the additional constrains that Player 0 cannot write to Tapes 2 and 3, and Player 3 cannot write to Tape 3.

A MPA$_k$-TM is a machine model which 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: every element of the state contains a turn indicator, and denotes information that can be read and written by each player. Every player has a associated list of tapes to indicate read/write rights of various tapes for the player itself. Thus, the tapes are partitioned according to access rights.

In particular, for $k = 1$ the MPA_k-TM (i.e., the MPA_1-TM) bears resemblance to the PA-TM. Similarly, the MPA_1-TM with both players sharing resources corresponds to the A-TM (without logical "not" operation). Furthermore, an MPA_1-TM with unique next moves for the universal ($\forall$) player is an N-TM, whereas, a MPA_1-TM with unique next moves for all players is simply a D-TM. Hence, it is evident that the MPA_k-TM accepts the recursively enumerable (r.e.) languages since they are at least as powerful as an ordinary D-TM. Similarly, we can show that the MPA_k-TM accepts only r.e. languages by enumerating all possible accepting subtrees, and subsequently checking recursively whether each tree is a true accepting subtree.

An MBA_k-TM is derived from the MPA_k-TM by disallowing the $\forall$-player from writing on any resource which is readable by any $\exists$-player. Consequently, moves of the $\forall$-player are invisible to the $\exists$-player). Hence, the MBA_k-TM correspond to blindfold multiplayer games. Observe that for $k = 1$ the MBA_k-TM (i.e., the MBA_1-TM) bears resemblance to the BA-TM. We show that the MBA_k-TM and the BA_k-TM accept r.e. languages and only r.e. languages.

After analyzing the MPA_k-TM and the MBA_k-TM, we are forced to realize that both these machines are too general and powerful. We combat their over-generality by introducing restricted versions of each machine. The $k+1$-player private alternating Turing machine (PA_k-TM) is an MPA_k-TM if resources visible to Player i are also visible to Player $i-1$ for all existential players ($i \in \{2, 3, \ldots, k\}$). Hence, there is a hierarchical ordering of $\exists$-players. A $k+1$-player blind alternating Turing machine (BA_k-TM) is a PA_k-TM where the $\forall$-player cannot change a resource visible to any other $\exists$-player. Examples of the PA_k-TM and the BA_k-TM for $k = 3$ are depicted in Figures 2 and 3, respectively.

We have defined for each *game type* g in $\mathcal{G}$ = {two-player incomplete information, two-player blindfold, multiplayer incomplete information, multiplayer blindfold, perfect information, nondeterministic, deterministic} a corresponding machine type $m(g)$ in $\mathcal{M}$ = {two-person private alternating, two-person blind alternating, multiplayer private alternating, multiplayer blind alternating, alternating, nondeterministic, deterministic}.

2.3. Complexity of Games

This section defines alternation, time, space, and branch bounds.

For any input string $\omega \in \Sigma^*$, let the *initial position* $p_0(\omega)$ have the initial state q_0, and let tape contents be described in accordance with Definition 2.2.2. The *accepting* states are all universal states with no successors. The *rejecting* states are all existential states with no successors. Each play of the computational game G^M is called a *computation sequence*, and the corresponding game tree T is called *computation tree*. The input string $\omega \in \Sigma^*$ is *accepted* by M if the existential player has a winning strategy. The computation sequences induced by winning strategy form an *accepting subtree* of T.

A move $p \vdash p'$ is an alternation if the player to move next is on a different team from the player which just moved. That is to say that $p \in POS_i$ and $p' \in POS_j$, where i and j belong to different teams. Thus, either

(1) $i \in T_0$ and $j \in T_1$, or
(2) $i \in T_1$ and $j \in T_0$.

Now we can define alternation bound, time bound, space bound, and branch bounds.

A game G has a alternation bound $A(n)$ if for each initial position $p_0 \in POS$ of length n, for which Team T_1 has a winning strategy, there is some such winning strategy σ which induces a play π containing at most $A(n)$ alternations.

DEFINITION 2.3.1. ALTERNATION BOUND $A(n)$. *For every input string $w \in \Sigma^n$ (corresponding to some initial position p_0) accepted by the associated machine M, there exists an accepting subtree GT' such that a computation sequence corresponding to any play π, induced by strategy σ, does not have more than $A(n)$ alternations for every computation sequence $\pi \in GT'$.*

A game G has a time bound $T(n)$ if for each initial position $p_0 \in POS$ of length n, for which Team T_1 has a winning strategy, there is some such winning strategy σ which induces a winning play π containing at most $T(n)$ moves.

DEFINITION 2.3.2. TIME BOUND $T(n)$. *For every input string $w \in \Sigma^n$ (corresponding to some initial position p_0) accepted by the associated machine M, there exists an accepting subtree GT' such that a computation sequence corresponding to any play π, induced by strategy σ, does not have more than $T(n)$ moves for every computation sequence $\pi \in GT'$.*

A game G has a space bound $S(n)$ if for each initial position $p_0 \in POS$ of length n, for which Team T_1 has a winning strategy, there is some such winning strategy σ which induces a winning play π, where any position reachable from the initial position can be represented in space at most $S(n)$. We classify a game as reasonable if it has a space bound of $O(n)$.

DEFINITION 2.3.3. SPACE BOUND $S(n)$. *For every input string $w \in \Sigma^n$ (corresponding to some initial position p_0) accepted by the associated machine M, there exists an accepting subtree GT' such that a computation sequence corresponding to any play π, induced by strategy σ, does not use more than $S(n)$ nonblank cells for every computation sequence $\pi \in GT'$.*

A game G has a branch bound b if for all positions in $POS(p_0)$ there are at most b choices for the next transition.

DEFINITION 2.3.4. BRANCH BOUND b. *A game G has a branch bound b if for every starting position p, $|\{p' \mid p \vdash p'\}| \leq b$.*

It is imperative to note that the complexity bound definitions given above are relevant only to the win outcome problem of the game. When considering the Markov ($m(n)$) (nonlosing) outcome problem for a game, we bound space, time, and alternations only for plays associated to winning strategies. Recall that a nonlosing play can be infinite.

Our primary objective in introducing the MPA_k-TMs is to derive a characterization of full elementary recursive hierarchy. However, we discovered that even the MPA_2-TMs with constant space accept all r.e. languages. This resulted in variant definitions which give a more natural extension of the space and time results of Peterson and Reif [1].

2.4. Languages Accepted by Multiplayer Alternating Machines

We define the *computation game* $G^M = (POS, \vdash, VIS, T_1)$ where we have the following.

1. POS is a set of *positions* with

$$POS = \{0, \ldots, k\} \times P_1 \times \cdots \times P_r,$$

   where $0, \ldots, k$ represent Players $0, \ldots, k$, and $P_1, \ldots, P_r$ are sets of strings over a finite alphabet used to describe various components of the positions.
2. $\vdash \subseteq POS \times POS$ is the next move relation satisfying Axioms 1 and 2 stated below. These axioms simply prevent a player from modifying the information which is invisible to him, and inhibit him from using the invisible information in formulating his strategy.
3. The mapping $VIS : \{0, \ldots, k\} \mapsto 2^{\{1,\ldots,r\}}$ is a mapping from the set of players to all possible subsets of the set of positions. Let $p = (a, p[1], \ldots, p[r])$ be a position in POS. We say that Player i has right to view $p[j]$ if $j \in VIS(i)$.
4. Team $T_1 \in 2^{\{0,\ldots,k\}}$ is a subset of the set of Players $\{0, \ldots, k\}$. The opposing team is $T_0 = \{0, \ldots, k\} - T_1$ 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.

The mapping VIS describes the access rights of the players. For each Player $i = 0, \ldots, k$ we let $vis_i(p) = (v, b)$, where v is the list (in order of occurrence) of components of position p for

which Player i has access rights (i.e., the ordered list $\langle p[j] \mid j \in VIS(i)\rangle$),[2] and b is a Boolean variable which is 1 if it is Player i's turn to move (and otherwise it is 0). $priv_i(p)$ is the list of all components of the position which are known only to the Player i. More explicitly $priv_i(p)$ is $\langle p[j] \mid j \in VIS(i)$, and $j \notin VIS(k)$, for all $k \neq i\rangle$.

When we are dealing with multiplayer games, we need the following axioms to incorporate the notion of incomplete information.

AXIOM 1. *No player is permitted to modify any other player's private information.*

Therefore, if $p \in POS_i$ and $p \vdash p'$, then $priv_j(p) = priv_j(p')$ for all players $j \neq i$.

AXIOM 2. *If a pair of nonterminating positions p, q are indistinguishable to a player, then the sets of next moves he can make from this pair p, q must also be indistinguishable to the player.*

As a consequence of Axiom 2, if $p, q \in POS_i - W$ and $vis_i(p) = vis_i(q)$ then $\{vis_i(p') \mid p \vdash p'\} = \{vis_i(q') \mid q \vdash q'\}$.

We define the set of *winning positions* (W) to be the set of all positions from which there is no next move for the opponent on its turn to move

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

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

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

Player i loses if any position in the set $POS_i \cap W$ is encountered. The result of a 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.

Let the language of M be $L(M) = \{\omega \in \Sigma^* \mid \omega \text{ is accepted by } M\}$.

THEOREM 2.4.1. *The PA_k-TM accepts precisely the recursively enumerable sets.*

PROOF. The D-TM can be considered a special case of the PA_k-TM. Since the D-TM accepts r.e. sets, the PA_k-TM also accepts r.e. sets.

On the other hand, the winning strategies of a computation game (of incomplete information) can be recursively enumerated by all possible accepting subtrees and check recursively whether each tree is a true accepting subtree. Thus, the language of the PA_k-TM is r.e. The theorem follows. ∎

THEOREM 2.4.2. *The BA_k-TM accepts precisely the recursively enumerable set.*

PROOF. The D-TM is a special case of the BA_k-TM. Hence, the BA_k-TM accepts r.e. sets. On the other hand, the winning strategies of a blindfold computation game can be recursively enumerated. Consequently, the language of the BA_k-TM is r.e. ∎

THEOREM 2.4.3. SPACE COMPRESSION. *For any constant $\epsilon > 0$ and a machine M of some type $\mathcal{M}$, with space bound $S(n)$, there is a machine with space bound $\epsilon S(n)$ that accepts the same language as M, and with the same machine type $\mathcal{M}$ with no additional alternations or tapes.*

PROOF. We encode each $2/\epsilon$ consecutive work tape cells as a $2/\epsilon$-tuple in a new tape alphabet. This technique can be used to achieve desired space compression by any constant factor. ∎

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

THEOREM 2.4.4. SPEED-UP (TIME DILATION). *For any constant $\epsilon > 0$ and a machine M of some type $\mathcal{M}$, with time bound $T(n)$ such that $\liminf_{n\to\infty}(T(n)/n) = \infty$ and at least one tape, there is a machine of the same type $\mathcal{M}$ with time bound $\epsilon T(n)$ that accepts the same language as M, with no additional tapes.*

PROOF. We construct M' of same type $\mathcal{M}$ which accepts the same language as M, and is identical except as described below. We encode t consecutive tape cells of each tape of M as a t-tuple in the tape alphabet of M' (cf. Theorem 2.4.3 above). The number of possible next moves from any position must be finite, so we must have some constant upper bound d on the branching factor. Since the maximum branching factor is d, there are at most d^t positions reachable of M after t moves. Thus, there are at most 2^{d^t} possible strategies of the existential team within next t moves. M' will have a distinguished state associated with each of these possible strategies.

Given an input string $\omega \in \Sigma^n$, the simulation proceeds in $T(n)/t$ steps, where in each phase M' simulates t *nonprivate* steps of M.[3] M' would also have an additional tape which is private to the universal player. This contains a counter Δ ($0 \le \Delta \le t$), which keeps track of the number of nonprivate steps which remain to be simulated by M' during this phase. Δ is initialized to t at the start of a phase. Then, M' moves one cell left, two cells right, and one cell left (chronologically) on each of its tape to determine the currently relevant tape contents. Then, the existential team of M' is allowed to choose its strategy for the next t existential steps of M, by a single state transition. If no such strategy exists then M' rejects.

Subsequently, the universal player of M' executes a series of rounds, each of which requires only a single step of M'. These steps are not detectable by the existential team of M'. Inductively, $t - \Delta$ nonprivate steps of M have been simulated by M' during this phase. In the current round, M' simulates $t' = \min(t, \Delta)$ steps of M. Some of these steps may be existential, and for these steps the strategy already chosen by existential team is applied. At the end of the round, Δ is privately subtracted by the number of nonprivate (with respect to universal player) steps of the round. Thus, new Δ is set to old Δ minus $t - t_p$, where t_p is the number of steps of the round which were private to the universal player. We are guaranteed that t_p will not always be 0, since we are dealing with a game of incomplete information.

If $\Delta > 0$ then we proceed with the next round, otherwise we terminate the round. After the last round, the universal player makes visible (to the existential team) all modifications to common portion which were made on simulated nonprivate moves during this phase. M' makes four additional moves of the tape heads (once left, twice right, once left) to update the tapes, and then the simulation proceeds to next phase. M' takes at most ten steps for every t steps of M, and the total time bound of M' is

$$n + \left\lceil \frac{n}{t} \right\rceil + 10 \left\lceil \frac{T(n)}{t} \right\rceil,$$

which is less than $\epsilon T(n)$ if we let $t = 20/\epsilon$, and $n \le (9/20)\,\epsilon T(n)$. Since there are only a constant number of inputs of length $n > (9/20)\,\epsilon T(n)$, and for these inputs we can use the finite state control to decide acceptance within time n. The speed-up theorem follows. ∎

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.*

Now, we show that the computation games of various types of machine are universal for the corresponding classes of games. Fix some functions $A(n)$ and $S(n) \ge \log(n)$, and let g be a game type in $\mathcal{G}$. Let $\mathcal{C}$ be the class of games of the predetermined type g, with space bound $S(n)$

[3] A move by the universal team is *private* if it does not modify any portion of a position that is visible by existential team. Otherwise the move is *nonprivate*.

and alternation bound $A(n)$. For every $G = (POS, \vdash, VIS, T_1)$ of $\mathcal{C}$, let B_G be deterministic log space mapping from positions in POS to their binary representation. Let N_G be a binary string encoding the deterministic space $MS(n)$ next move transducer for $\vdash$.

THEOREM 2.4.5. *If $MS(S(n)) = O(S(n))$ then the computation game G^{M_C} is a universal game for the game class $\mathcal{C}$.*

PROOF. By definition, there is a machine M_C such that for each game $G \in \mathcal{C}$ and position p of G, M accepts $(N_G, B_G(p))$ iff Team T_1 has a winning strategy in G from initial position p. In other words, M_C decides the outcomes of all games of $\mathcal{C}$. Furthermore, M_C has a corresponding machine type $m(g)$ (i.e., its computation game is of type g) has a tape alphabet $\{0, 1, b, \#\}$, space bound $S(n) + MS(S(n))$, and alternation bound $A(n)$. If $MS(S(n)) = O(S(n))$ then by Theorem 2.4.3, M_C needs to have only space bound $S(n)$. The theorem follows. ∎

By applying space compression Theorem 2.4.3, we have the following.

COROLLARY. *For each game type $G \in \mathcal{G}$, if $\mathcal{R}$ is the class of reasonable games (i.e., with space bound $S(n) = n$ of type g, then there is a linear space bounded machine $M_{\mathcal{R}}$ of corresponding type $m(g)$ such that $G^{M_{\mathcal{R}}}$ is a universal game for $\mathcal{R}$.*

3. LOWER BOUNDS

3.1. Bounds for Two-Player Games

Reif [3,4] uses the A-TM complexity to derive the complexity of space bounded machines representing various two-player games. We shall derive our results by direct comparison to the D-TMs. The corresponding results for multiplayer games will follow as trivial extensions. We shall introduce a game to facilitate the statement and the proofs of our results.

DEFINITION 3.1.1. STRATEGY COUNTER GAME (SCG). *The game involves two players, Player 0 and 1. Player 1 has to convince Player 0 that it can count from 0 up to $2^{2^n} - 1$ (inclusive). It does so by producing a sequence of numbers from 0 to $2^{2^n} - 1$ in binary as a sequence of groups in the format $\#0^{2^n}\#0^{2^n-1}1\#0^{2^n-2}10\#0^{2^n-2}11\#0^{2^n-3}101\#\cdots\#1^{2^n-1}\#$, and then halting. Player 1 sends this sequence (via a common state element) of groups to Player 0. Meanwhile, Player 0 attempts to find a flaw in the output sequence of Player 1. Player 1 wins if Player 0 fails to find a flaw in its output.*

PROPOSITION 3.1.1. *SCG requires at least 2^n bits when played deterministically because each group has 2^n bits.*

However, the ∀-player can do better than that by modeling SCG as a game of incomplete information in which Player 0 is the ∀-player, and Player 1 is the ∃-player. ∃-player communicates a sequence of groups $\#0^{2^n}\#0^{2^n-1}1\#\cdots\#1^{2^n-1}\#$ as described above. The ∀-player's strategy is to nondeterministically choose one of the following verifications.

1. The first group is in 0^*, and the last is in 1^*.
2. Choose any group and verify (secretly) that it has length 2^n.
3. Choose any particular *bit* of a group, remember it and its location. Check it against the same position in the next group, with respect to carry, etc.

LEMMA 3.1.1. *SCG requires only n tape cells for the ∀-player using the above strategy.*

PROOF. ∀-player models SCG as a game of incomplete information as illustrated above. The first and second check obviously require less than n bits if it counts using an n-bit binary counter. The third check involves three steps.

1. Store the selected bit and its location as n-bit index.
2. Sequentially decrement the index as we transverse the next group until the index becomes zero to indicate that we have reached the corresponding bit.
3. Verify the correctness of the chosen bit with respect to its corresponding bit and carry.

Since all three verifications can be performed using n tape cells, the theorem follows. ∎

It is critical that Player 1 does not have information of Player 0's verification process, otherwise Player 1 could cheat. SCG as described is a blindfold game, but if Player 0 is allowed to communicate with Player 1 then it generalizes to counting (i.e., the ∀-player can ask the ∃-player to count both up and down). We can apply the SCG game technique after some modifications to our lower bound proofs.

LEMMA 3.1.2. *(See [3,4].) For constructible* $S(n) \geq \log(n)$

$$BA_1\text{-}SPACE(S(n)) \supseteq DSPACE(EXP(S(n))).$$

PROOF. We modify the counting technique in SCG game above so that the ∃-player sends (instead of a sequence of numbers) a sequence of configurations, $\langle C_0, C_1, \ldots, C_m \rangle$, of a given N-TM which uses at most $2^{S(n)}$ tape cells. The BA_1-TM will accept the input if and only if there is an accepting sequence of configurations for the N-TM on that input (i.e., the input is also accepted by the N-TM).

The ∀ strategy is similar except

1. it checks that the first configuration (C_0) against the input;
2. it ascertains that the last configuration (C_m) is accepting;
3. it compares corresponding tape cells, and confirms that they are the same barring head motion, state change, tape cell changes, etc., in correspondence with transition rules.

The universal player chooses which condition to check privately from the existential player. Otherwise, the existential player could cheat.

To verify if (1) is violated, the universal player may utilize the $\log n$ cells of a private tape for a pointer to symbols of input string ω. Thus, the universal player can check the first (C_0) against the input. Ascertaining if (2) is violated is trivial. To verify (3), the universal player compares the corresponding tape cells of two configurations, tracking the location index would dominate the space cost as shown by Reif [4]. Since the length of any any configuration C_i is at most $EXP(S(n))$, a $\log(EXP(S(n)))$-bit counter track the index. An $S(n)$-bit counter is sufficient because the constants are absorbed in the $EXP(S(n))$ notation.

Consequently, the ∀-player only needs to write down the length of a configuration and a few other bits of information. Consequently, it needs only nondeterministic $S(n)$ space. The conversion to the D-TM affects only the constant in the exponent because

$$NSPACE(EXP(S(n))) = DSPACE\left(EXP(S(n))^2\right) = DSPACE(EXP(S(n))).$$

The lemma follows. ∎

LEMMA 3.1.2. *(See [3,4].) For constructible* $S(n) \geq \log(n)$

$$PA_1\text{-}SPACE(S(n)) \supseteq DTIME(EXP_2(S(n))).$$

PROOF. Recall the proof for $ASPACE(S(n)) \supseteq DTIME(EXP(S(n)))$ of Chandra *et al.* [2] relies on the fact that when a D-TM with space bound $EXP(S(n))$ is simulated by an A-TM, the space requirements are dominated by keeping track of two counters, of $S(n)$ bits each. These counters keep up with the configurations number and the tape cell index. Also, observe that if the universal player is allowed to communicate with Player 1 then it generalizes to counting, and the ∀-player can ask the ∃-player to count both up and down. We can apply the SCG game technique after some modifications to our lower bound proofs.

We replace these counters by private game counters, and simulate a $DTIME(EXP_2(S(n)))$ D-TM with an $S(n)$ space bounded PA_1-TM. The ∃-player is counting and making existential moves of the $ASPACE(S(n)) \supseteq DTIME(EXP(S(n)))$ proof. ∎

The following theorem provides the matching upper bounds, and is obtained as a special case (substituting $k = 1$) in Corollaries 5.2.1 and 5.2.3 of [5].

THEOREM 3.1.1. *(See [1,3–5].) For constructible* $S(n) \geq \log(n)$

$$BA_1\text{-}SPACE(S(n)) \subseteq DSPACE(EXP(S(n))),$$
$$PA_1\text{-}SPACE(S(n)) \subseteq DTIME(EXP_2(S(n))).$$

The proof of the above theorem is based upon a modified game tree argument.

COROLLARY 3.1.1. *For constructible* $S(n) \geq \log(n)$

$$BA_1\text{-}SPACE(S(n)) = DSPACE(EXP(S(n))),$$
$$PA_1\text{-}SPACE(S(n)) = DTIME(EXP_2(O(S(n)))) = ASPACE(EXP(S(n))).$$

PROOF. Follows from the above Lemmas 3.1.2 and 3.1.3 in conjunction with Theorem 3.1.1 and Chandra *et al.*'s [2] proof of $ASPACE(S(n)) = DTIME(EXP(S(n)))$. ∎

3.2. Space Bounds for Multiple Person Alternating Machines

We can readily generalize these results to multiplayer games.

THEOREM 3.2.1. *For constructible* $S(n) \geq \log(n)$

$$MPA_1\text{-}SPACE(S(n)) = PA_1SPACE(S(n)) = DTIME(EXP_2(S(n))).$$

PROOF. MPA_1-TM is by definition also a PA_1-TM. Consequently,

$$PA_1SPACE(S(n)) = DTIME(EXP_2(S(n)))$$

implies

$$MPA_1\text{-}SPACE(S(n)) = DTIME(EXP_2(S(n))).$$ ∎

For more than one player of team T_1, the MPA_k-TM is so powerful that their power is not restricted by space bounds.

THEOREM 3.2.2. *For constructible* $S(n) \geq \log(n)$, *if input tape is private to the* $\forall$*-player then*

$$MPA_2\text{-}SPACE(constant) = \text{r.e.}$$

PROOF. Given a D-TM M we shall construct an MPA_2-TM M' which accepts the same set of strings in constant time. The game will be based on having each of the $\exists$-players find a sequence of configurations of the D-TM M which on an input ω lead to acceptance. Accordingly, upon request each $\exists$-player will give the $\forall$-player the next character of its sequence of configuration. Each $\exists$-player does this secretly from the other $\exists$-player. The configuration will be of the form $\#C_0\#C_1\#\cdots\#C_m\#$, where C_0 is the initial configuration of M on the input, and C_m is an accepting configuration of M.

The $\forall$-player will choose to verify the sequences in one of the following ways.

1. Check one of the first configurations against the input, and ensure that it is in the correct form. This implies that the input tape is private to the $\forall$-player.
2. Check the last configuration for accepting state.
3. Check that the $\exists$-players are giving the same sequence by alternating turns between them.
4. Run one of the $\exists$-players ahead of the next #, and check its C_i against the other $\exists$-player's C_{i-1} for proper head motion, state change, and tape cell changes, in correspondence with the transition rules, etc.

Observe that the $\forall$-player need only track the incoming information. Therefore, it does not have to store anything on the tape, and can remember all information in its private state. Thus, we have shown $MPA_2\text{-}SPACE(constant) \supset$ r.e.

It is easy to see that the MPA_k-TMs accept the r.e. languages since they include ordinary the D-TMs. Similarly, acceptance by the MPA_k-TM is r.e. since one can enumerate all possible accepting subtrees and check recursively whether each tree is a true accepting subtree. The proof is now complete. ∎

COROLLARY 3.2.1. *For constructible* $S(n) \geq \log(n)$, *if input tape is public to the all players then*

$$MPA_2\text{-}SPACE(\log(n)) = \text{r.e.}$$

PROOF. Follows from the definitions for $MPA_k\text{-}SPACE(\log(n))$ with $k = 2$. ∎

COROLLARY 3.2.2. *Membership for* MPA_2*-SPACE(constant) is still undecidable.*

PROOF. Membership for MPA_2-*SPACE(constant)* is undecidable since it is a case of the halting problem on blank tape. ∎

3.3. The Hierarchical Private and Blind Alternating Turing Machines

In order to remove the over-generality of the MPA_k-TM and the MBA_k-TM, we consider variants of these machines. More specifically, we introduce the PA_k-TM and the BA_k-TM to avoid the nonrecursiveness of space bounded computation. In terms of space characterization, they are evidently natural extensions of the A-TMs, the PA-TMs, and the BA-TMs.

The lower bound techniques of Section 3.1 can be extended to count very high values with n bits by using k players. We introduce new notation to facilitate the description of the counting techniques used in our lower bounds.

DEFINITION 3.3.1. $TWOEXP_m(f(n))$. *For a function* $f(n)$, $TWOEXP_m(f(n))$ *is the tower of* m *repeated exponential of* $f(n)$ *to the base* 2. *Recursively* $TWOEXP_m(f(n))$ *is defined as follows:*

$$TWOEXP_1(f(n)) = TWOEXP(f(n)) = 2^{f(n)},$$
$$TWOEXP_m(f(n)) = 2^{TWOEXP_{m-1}(f(n))}, \qquad \text{for } m > 1.$$

Using techniques of Section 3.1, k players and n bits can be used to count up to $TWOEXP_{k+1}(n)$. The earlier method used n bits for one player counting alone on $TWOEXP_1(n)$ bits (up to $TWOEXP_2(n) = 2^{2^n}$). These bits were used to store lengths of numbers, positions within numbers, etc. Since we only need to start from 0 and count up to $TWOEXP_{k+1}(n)$, we can recursively apply the technique. The i^{th} $\exists$-player ($\exists_i$-player) will be responsible for supplying the correct count on $TWOEXP_i(n)$ bits, i.e., up to $TWOEXP_{i+1}(n)$, using the count sequence on $TWOEXP_{i-1}(n)$ bits supplied by the $\exists_{i-1}$-player. Hence, the k^{th} $\exists$-player ($\exists_k$-player) will supply the count on $TWOEXP_k(n)$ bits, i.e., count up to $TWOEXP_{k+1}(n)$. This will be checked by the $\forall$-player using the count sequence on $TWOEXP_{k-1}(n)$ bits which is supplied by the $\exists_{k-1}$-player. The $\forall$-player can verify the correctness of $\exists_{k-1}$-player's count against the sequence supplied by $\exists_{k-2}$-player's counting sequence. Inductively, any $\exists_i$-player's counting sequence can be checked against $\exists_{i-1}$-player's counting sequence.

Observe that, since this is a hierarchical machine, the $\exists_i$-player knows that its counting sequence is being used to check $\exists_{i+1}$-player but is not informed that it is being checked by $\exists_{i-1}$-player. If the order of hierarchy was not maintained a player might know where it is being checked, and use that information to cheat.

THEOREM 3.3.1. *For constructible* $S(n) \geq \log(n)$

$$PA_k\text{-}SPACE(S(n)) \supseteq DTIME(EXP_{k+1}(S(n))).$$

PROOF. Note that these are blindfold games because the players are blindfolded in the sense that they only receive turn information. If the $\exists$-players were not blind then the $\forall$-player can instruct them to count up and down. Consequently, the Chandra *et al.* [2] technique can be used using the new very large counters to simulate a $DTIME(TWOEXP_{k+1}(S(n))$ bounded D-TM with a $S(n)$-space bounded PA_k-TM. The theorem follows. ∎

THEOREM 3.3.2. *For constructible* $S(n) \geq \log(n)$

$$BA_k\text{-}SPACE(S(n)) \supseteq DSPACE(EXP_k(S(n))).$$

PROOF. For the blindfold case, the $\exists_k$-player sends out the configuration of sequences of size $DSPACE(TWOEXP_k(S(n)))$ with $\exists_{k-1}$ through $\exists_1$ players sending out their respective counts. The $\forall$-player uses the counts to check other counts, and uses the largest track information for checking the configurations, etc. The theorem follows. ∎

The matching upper bounds from Theorems 5.2.2 and 5.2.3, along with Corollaries 5.2.1 and 5.2.3 of [1,5] are as follows.

THEOREM 3.3.3. *For constructible* $S(n) \geq \log(n)$

$$PA_k\text{-}SPACE(S(n)) \subseteq DTIME(EXP_{k+1}(S(n))),$$
$$BA_k\text{-}SPACE(S(n)) \subseteq DSPACE(EXP_k(S(n))).$$

Now, we can combine the upper and lower bounds to deduce the main result of this paper.

THEOREM 3.3.4. *For constructible* $S(n) \geq \log(n)$

$$PA_k\text{-}SPACE(S(n)) = DTIME(EXP_{k+1}(S(n))),$$
$$BA_k\text{-}SPACE(S(n)) = DSPACE(EXP_k(S(n))).$$

PROOF. Follows from the upper and lower bounds of above Theorems 3.3.1–3.3.3. ∎

In light of the above result, we can see that the PA_k-TM and the BA_k-TM are quite natural generalizations to the PA-TM and the BA-TM. The well-ordering of the generated hierarchies indicates that space and time must complement each other extremely well.

As a consequence of Corollary 2.4.1 in conjunction with the results of this section, we have the following.

1. A space bounded PA_k-TM M whose computation game G^M is universal for all reasonable multiplayer games of incomplete information.
2. A space bounded BA_k-TM M' whose computation game $G^{M'}$ is universal for all reasonable multiplayer blindfold games.

By the hierarchy theorem for deterministic time complexity of Hartmanis and Stearns [16], we have the following corollary.

COROLLARY 3.3.1. *For M defined above, if any D-TM decides the outcome of G^M in time $T(n)$, then $T(n) > EXP_{k+1}(n/\log n)$.*

By space hierarchy results.

COROLLARY 3.3.2. *For M' defined above, if any D-TM decides the outcome of $G^{M'}$ in space $S(n)$, then $S(n) > EXP_k(n/\log n)$.*

4. TEAM-PEEK GAMES

4.1. Extensions of PEEK games

Chandra *et al.* [2] and Reif [3,4] have used PEEK games as concrete examples to illustrate and support the computation game theory results. We review these definitions, and in their tradition, introduce three enhanced versions of PEEK with more than two players (that are partitioned into two teams).

The game PEEK consists of a box containing a number of vertical plates each of which can be positioned either *in* or *out*. The original definition of PEEK described the plates to be

horizontal [2]. However, in order to extend it to multiplayer (team) PEEK, we need to improvise the original definition for aesthetic reasons that will be evident later. A subset of the plates is controlled by Player 0, and the remaining plates are controlled by Player 1. On its turn, each player can adjust the state of any (not necessarily nonempty) subset of its plates by pushing *in* or pulling *out* the plates not in the desired positions.

Every plate has uniform sized holes cut in it. During the course of the game, the players know where these holes are. The winner is the first player to adjust the plates so that the holes are placed in a horizontal alignment to allow it to peek from one end to the other. PEEK is a game of perfect information in the sense that the players know the pattern of holes as well as the positioning of all the plates. Figure 4 exhibits a typical plate and Figure 5a illustrates the set up of the game.

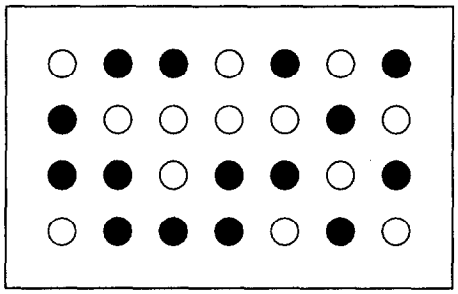
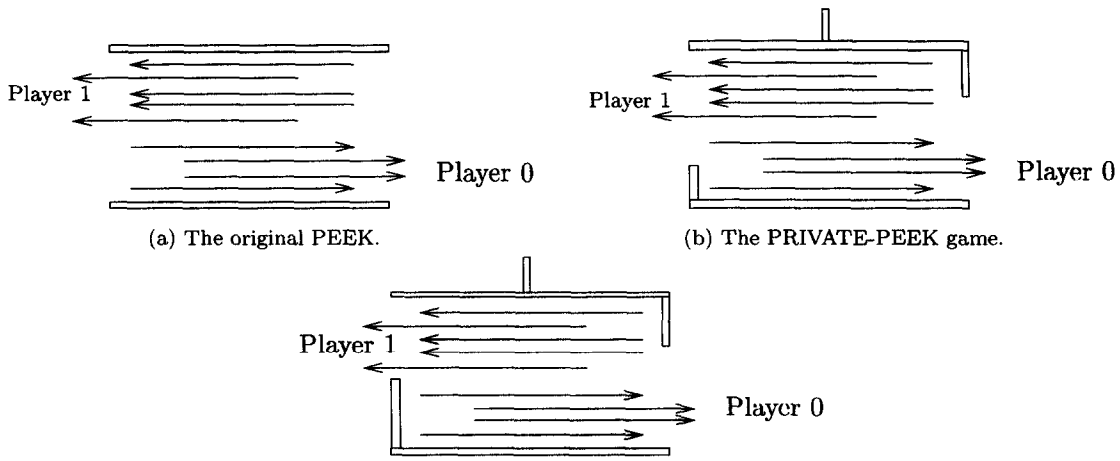


Figure 4. A side view of an isolated plate of PEEK.

Reif [3,4] described PRIVATE-PEEK, an extension of PEEK which introduces private portions of positions by using barriers. As shown in Figure 5b, partial barriers are used to conceal the location of some of the opponent's plates. Another barrier is placed on the "peeping end" of the box restricting the players to be able to peep only on their side of the box. Reif [3,4] also defined BLIND-PEEK by requiring the barriers on Player 1's side to obscure the locations of all opponent's plates (as shown in Figure 5c).



(a) The original PEEK.

(b) The PRIVATE-PEEK game.

(c) The BLIND-PEEK game.

Figure 5. (a) A position of PEEK; (b) PRIVATE-PEEK set-up; (c) BLIND-PEEK game.

In this paper, we describe three versions of PEEK involving at least three players (that are partitioned into two teams). The simplest one of these is TEAM-PEEK, which extends the original PEEK to accommodate several players. The players are divided into two teams that stand on each side of the box. The players move in turn as they do in original PEEK. A subset

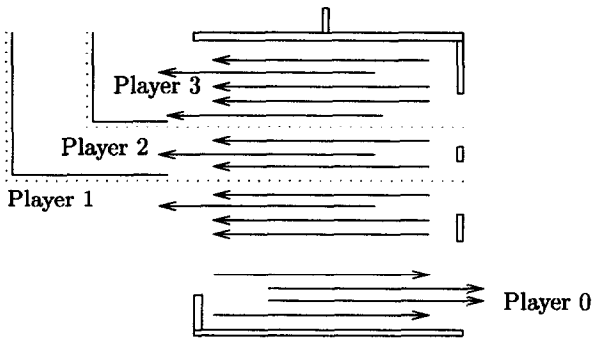


Figure 6. A position of TEAM-PEEK.

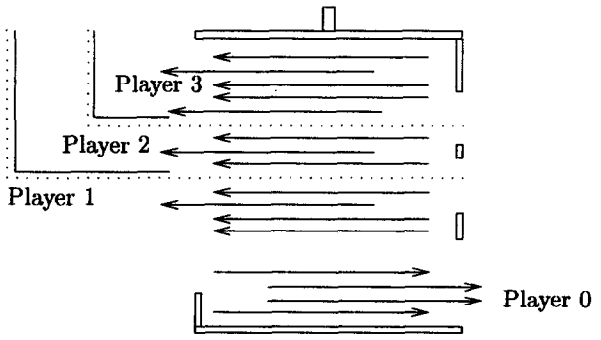


Figure 7. A position of TEAM-PRIVATE-PEEK.

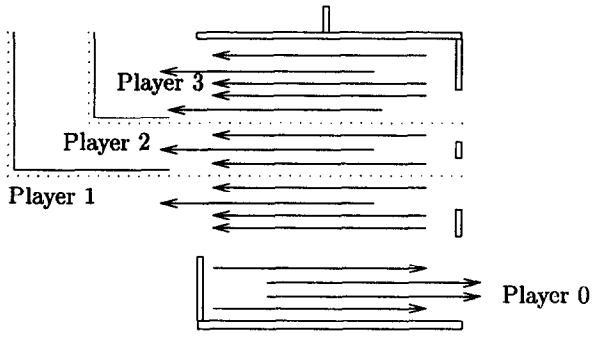


Figure 8. A position of TEAM-BLIND-PEEK.

of plates is controlled by each player, such that the elements of the set of all these subsets are mutually exclusive and collectively exhaustive with respect to the plates in the box. On its turn, each player can adjust the state of any (not necessarily nonempty) subset of its plates by pushing *in* or pulling *out* the plates that are not in the desired positions already. The winner is the first player to adjust the plates so that the uniform sized holes are placed in a horizontal alignment to allow it to peek from one side to the other.

TEAM-PEEK is a multiplayer game of perfect information in the sense that players know the pattern of holes as well as the positioning of all the plates. Figure 6 describes the set up of the game TEAM-PEEK.

TEAM-PRIVATE-PEEK is devised by placing one-way mirrors so that certain members of the team cannot see the positions of other plates. A barrier is also placed on the "peeping" side of the box to restrict the players to peeping only on their side of the box. A game of TEAM-PRIVATE-PEEK is considered hierarchical if the players of the Team T_1 can ordered in way such that each player has at most as much information as the player appearing just before him in the ordering. A *hierarchical* TEAM-PRIVATE-PEEK is sketched in Figure 7.

TEAM-BLIND-PEEK is an extension of TEAM-PRIVATE-PEEK which does not allow the players of Team T_1 to see any of the plates controlled by Team T_0. TEAM-BLIND-PEEK is sketched in Figure 8.

4.2. Games on Propositional Formulae

In this section, we define a sequence of games on propositional formulae which can be used to encode states of a computational machine. These games are used to prove completeness results for various flavors of PEEK.

Formula refers to well-formed parenthesized Boolean expression involving Boolean variable symbols, binary connectives, and a unary connective. The Boolean variable symbols are denoted by lower case alphabets like t, u, v, w, x, y, z. The binary connectives consist of *conjunction* ($\wedge$), *disjunction* ($\vee$), and *exclusive-or* ($\oplus$). The unary connective is *negation* ($\neg$). A *literal* is a Boolean variable or its negation. A formula F is categorized to be in conjunctive normal form (CNF) if it is a conjunction of sub-formulae $F_1 \wedge \cdots \wedge F_p$ such that each F_i is a disjunction of literals. F is further classified to be in k-CNF if each F_i is a disjunction of at most k literals. Disjunctive normal form (DNF) and k-DNF are defined analogously.

$V(F)$ is the set of variable symbols in F, and $size(F)$ is the number of occurrences of variable symbols in F. $F(X)$ denotes the formula on variables in set X, i.e., $V(F) \subseteq X$. Similarly, $F(X_1, \ldots, X_n)$ denotes a formula on variables in set $X_1 \cup \ldots X_n$, i.e., $V(F) \subseteq X_1 \cup \ldots X_n$, where every $X_1, X_2, \ldots, X_n$ are disjoint sets of variable symbols. For the set S, an S-assignment is a truth assignment to the variables in the set S. Consequently, a formula F maps $V(F)$-assignment to true or false (i.e., 1 and 0, respectively).

We will extend this notation to allow the X_i to denote sets of variables allow universal and existential quantification over these sets of variables, and also allow $\cup$ to be the union of all the elements of a list of sets.

We will define a few games G_1, G_2, and G_{2B}. We shall show that G_2 is universal for games of incomplete information, and G_{2B} is universal for blindfold games. Subsequently, we shall exhibit a mapping between these games and their corresponding PEEK games.

DEFINITION 4.2.1. GAME G_1. *We define G_1 to be a $k+1$-player game defined by a triple*

$$(\tau, F(X_0, X_1, \ldots, X_k, V_0, V_1, \ldots, V_k, a, s), \alpha),$$

where

- *$\tau \in 0, \ldots, k$ is a turn indicator which serves to identify the player who is going to take the initiative to move next;*
- *positions are encoded in the propositional CNF formula $F(X_0, X_1, \ldots, X_k, V_0, V_1, \ldots, V_k, a, s)$ where $X_0, X_1, \ldots, X_k$ and $V_0, V_1, \ldots, V_k$ are sets of Boolean variables with each $X_i \subseteq V_i$, and a and s are individual Boolean variables;*
- *α is a truth assignment on $S = X_0 \cup X_1 \cup \cdots \cup X_k$, mapping elements of S to 0 or 1 (false or true, respectively);*
- *for any Player $i \in \{0, \ldots, k\}$, i controls the truth assignment to all the variables in the set X_i;*
- *for any Player $i \in \{0, \ldots, k\}$, i knows the truth assignment to all the variables in the set V_i; to maintain consistency, $X_i \subseteq V_i$; furthermore, to maintain hierarchical structure of the game, we require $V_i \supseteq V_{i+1}$ for all valid i.*

When $\tau = 0$, Player 0 moves by

(i) *setting a to 0;*
(ii) *setting s to its complement (that is $\bar{s}$);*
(iii) *choosing a truth assignment to the variables of X_0.*

All other Players i ($i \neq 0$) move by setting a to 1, and choosing a new truth assignment for variables in set X_i.

The formula F is not modified by any of these moves, except for the changes in the truth assignment of its variables. The loser is the first player whose move yields truth assignment for which the formula F is false.

We shall prove that G_1 is as a stepping stone in our main goal to prove that TEAM-PEEK games universal for reasonable games of their respective classes.

LEMMA 4.2.1. *G_1 is universal for reasonable hierarchical games of incomplete information.*

PROOF. Let M be a PA_k-TM with space bound n. Suppose $\omega \in \Sigma^n$ is an input string to M. We encode each position of G^M as a bit vector of length $n' = c_M n$, where c_M depends only on the size of M's tape alphabet. Our encoding arranges bits $1, 2, \ldots, h_i$ to be those in $vis_i(p)$ (i.e., the portions of position p which are visible to Player i). $h_i \geq h_{i+1}$ to maintain hierarchical structure. In particular, $h_0 = n'$, and the bits $h_1 + 1, \ldots, n'$ are private to Player 0 ($\forall$-player).

Using the techniques of Stockmeyer [17], we may construct a linear size propositional formulae $NEXT(Z_1, Z_2, T)$, where Z_1, Z_2, T are sequences of n' variables each. Furthermore, if Z_1 encodes (by some fixed encoding which is computable in $O(\log n)$ space by a D-TM) a position p_1, then there exists an assignment to variables of T, such that $NEXT(Z_1, Z_2, T)$ is true if and only if Z_2 encodes a position p_2 derived from p_1 by a move of M.

We introduce a sequence of variables Y^C, Y^{P0}, Y^{P1}, X, (where X is the concatenation of $X_1, X_2, \ldots, X_k$) such that

- Y^C is of length $m_0 = h_1 + n'$ (and denotes portions of X which are common to all players);
- Y^{P_0} and Y^{P_1} are of length $l = n' - h_1$ (and denotes portions which are private to Player 0);
- X_i is of length $m_i = h_i + n'$. X_i for $i > 0$ denotes portions visible to Player i; X is the concatenation of $X_1, X_2, \ldots, X_k$.

Let $X[a \ldots b]$ denote $X(a), \ldots, X(b)$ for $a \leq b$ and a, b within range of X's indices. $Y^C[a \ldots b]$, $Y^{P0}[a \ldots b]$, and $Y^{P1}[a \ldots b]$ are defined analogously.

Players of the game take turns every round. Let the permutation which defines the order of moves in a around be $\Pi = (\pi_0, \pi_1, \pi_2, \ldots, \pi_k)$, where players follows one another in the order their indices occur in Π. We insist that $\pi_0 = 0$, so that Player 0's turn makes Π unique for every order of turns of players. However, regardless of what order the permutation is, we can emulate the moves of all players of Team T_1 by a single move by a "super-player".

For $s, \bar{s} \in \{0, 1\}$, let $NEXT_{a,s}(X, Y)$ be the formula derived from $NEXT(Z_1, Z_2, T)$ by substituting for Z_1, Z_2, T depending on whose turn it is. The substitution is shown in Tables 3 and 4.

Without loss of generality, we can assume that the teams are in strictly alternating order, and the first team to move is Team T_1. For each $a \in \{0, 1\}$, $NEXT_{a,s}$ defines a legal moves by Team T_a on switch variable $s \in \{0, 1\}$. Now, we consider the formula

$$\begin{aligned} F\left(X, Y^C, Y^{P0}, Y^{P1}, a, s\right) = &(a \wedge s \rightarrow NEXT_{1,1}(X, Y)) \\ &\wedge (a \wedge \neg s \rightarrow NEXT_{1,0}(X, Y)) \\ &\wedge (\neg a \wedge s \rightarrow NEXT_{0,1}(X, Y)) \\ &\wedge (\neg a \wedge \neg s \rightarrow NEXT_{0,0}(X, Y)). \end{aligned}$$

F can easily transformed into 5-CNF of size $O(n)$, and is constructible in $O(\log(n))$ space by a D-TM.

Let $p_o(\omega)$ be the initial configuration of M on input ω. Initially set $s = 1$ and $a = 1$. Also let variables $Y^C[1 \ldots k], Y^{P1}[1 \ldots l]$ be assigned to encode $p_0(\omega)$, and let all other variables be assigned arbitrarily. Let F and this initial truth assignment be the initial position of game G_1. "Team T_1 wins G_1" iff "Team T_1 wins computation game G^M" iff "M accepts input ω". Thus,

Table 3. Team T_0's move.

Z_1	$X[1..h_1]$	$Y^{P\bar{s}}$
Z_2	$Y^C[1..h_1]$	Y^{Ps}
T	$Y^C[h_1+1..m]$	

Table 4. T_1's move.

Z_1	$Y^C[1..h_1]$	Y^{Ps}
Z_2	$X[1..h_1]$	Y^{Ps}
T	$X[h_1+1..m]$	

we have a log-space reduction from acceptance problem for M to the outcome problem for G_1. By Corollary 2.4.1, G^M is universal for universal games. We conclude that G_1 is universal for reasonable games. ∎

Now, we define another formula game G_2, which is equivalent to TEAM-PRIVATE-PEEK game. G_2 uses G_1 as a sub game.

DEFINITION 4.2.2. FORMULA GAME G_2. *Let G_2 be a game in which each position contains the formulae $WIN_1(U_1, U_2, \ldots, U_k, V^C, V^P)$ and $WIN_0(U_1, U_2, \ldots, U_k, V^C, V^P)$ in disjunctive normal form and the truth assignments to the sequence of variables of $U_1, U_2, \ldots, U_k, V^C, V^P$. Let U be the concatenation of all U_i ($U = U_1 \cdot U_2 \cdots U_k$). We can emulate the moves of the individual players of the existential Team T_1 by a single move of a "super-player" as in game G_1.*

The formulae WIN_0 and WIN_1 and the truth assignments to $U \cup V^C$ are views by the universal player and the "super-player" emulating the existential team. However, V^P can only be viewed by the universal Player 0. Player 0 moves by changing at most one variable in $V^C \cup V^P$. For $i > 0$, Player i moves by changing at most one variable in U_i. The existential Team T_1 moves can be emulated by a "super-player" changing at most one variable in every U_i.

Team T_a wins if WIN_a is true after a move by Team T_a.

The following theorem is crucial to the main result of this section.

THEOREM 4.2.1. *Game G_2 is universal for reasonable games of incomplete information.*

PROOF. We introduce a sequence of variables U^A, U^B, V^A, V^B of length $m' = 4m + 2l + 4$. Let the concatenation of U^A and U^B be U, and let the concatenation of V^A and V^B be V. Recall, that U is also $U_1 \cdot U_2 \cdots U_k$. The variables of the sequences X, Y defined in previous construction will, in legal plays of our game G_2 be contained in U, V as in Table 5. The private portion V^P of V has a value of Y^{P0}, Y^{P1}, whereas V^C contains the values of the other elements of V.

For each $s \in \{0,1\}$ and $a \in \{0,1\}$, let $NEXT'_{a,s}(U,V)$ be the formula derived from $NEXT_{a,s}$ (X,Y) by substituting variables as in Table $5_{a,s}$.

We define *legal play* such that if both teams play legally then Team T_1 wins iff M accepts ω. Let legal cycle be a play which satisfies the restriction L for $i = 1, \ldots, m'$:

universal player changes the truth assignment of either $V^A(i)$ or $V^B(i)$,
existential team changes the truth assignment of either $U^A(i)$ or $U^B(i)$.

We also require restriction L' to hold within a cycle.

For distinct $s, \bar{s} \in \{0,1\}$ and each i, $t_{1,\bar{s}} \pmod{m'} < i \leq t_{0,s}$, universal Team T_0 assigns variables so that $NEXT_{0,s}$ = true when $i = t_{0,s}$, and for $t_{1,s} \pmod{m'} < i \leq t_{1,s}$, Team T_1 assigns variables so that $NEXT_{1,s}$ = true when $i = t_{1,s}$.

Thus, M accepts input ω iff Team T_1 has a winning strategy within legal plays satisfying restrictions L and L'. The following construction forces legal plays for both teams.

Table 5. The structure of game G_2.

	$\leftarrow m \rightarrow$		$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$
	X						
1,0							
				Y^{P0}		Y^C	Y^{P1}
		$t_{1,0}$					

	$\leftarrow m \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$		$\leftarrow m \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$
	X						
1,0							
		Y^C	Y^{P0}				Y^{P1}
				$t_{0,1}$			

	$\leftarrow m \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$	$\leftarrow m \rightarrow$		$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$
				X			
1,1							
		Y^C	Y^{P0}				Y^{P1}
					$t_{1,1}$		

	$\leftarrow m \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow m \rightarrow$	$\leftarrow l \rightarrow$	
				X			
0,0							
			Y^{P0}		Y^C	Y^{P1}	
							$t_{0,0}$

We introduce operation $\oplus'$ and Δ on any sequence Y, Z of Boolean variables of length m'. $\oplus'$ is defined as follows:

$$Y \oplus' Z = (Y(1) \oplus Z(1), \ldots, Y(n) \oplus Z(n)),$$

where $\oplus$ is the conventional Boolean exclusive or operative.

Δ is defined as follows:

$$\Delta Z = (\neg(Z(n) \oplus Z(1)),\ Z(1) \oplus Z(2), \ldots, Z(n-1) \oplus Z(n)).$$

We also introduce a threshold-two function to prevent any player from modifying two variables in the same turn. For a sequence B of Boolean variables, *THRESHTWO* is defined as follows:

$$THRESHTWO(B) = \bigvee_{1 \le i < j \le m} (B(i) \wedge B(j)).$$

Observe that $THRESHTWO(B)$ becomes true if two or more variables in the sequence B are true.

To detect illegal play we need to ensure that players move in turn. We introduce U' and V' as follows:

$$U' = \Delta\left(U^A \oplus U^B\right),$$
$$V' = \Delta\left(V^A \oplus V^B\right).$$

Now, we have developed all the notation required to define ILL_0 and ILL_1.

$$ILL_0 = THRESHTWO(V') \vee \bigvee_{1 \le i \le m'} (V'(i) \wedge U'(i+2) \wedge \neg U'(i)),$$

$$ILL_1 = \bigvee_{i \in T_1} THRESHTWO(U'_i) \vee \bigvee_{1 \le i \le m'} (U'(i) \wedge V'(i+1) \wedge \neg V'(i-1)).$$

ILL_0 is true if the universal player changes more than one variable on its turn, or moves out of turn. ILL_1 is true if the any player in the existential team changes more than one variable on its turn, or any player in the existential team moves out of turn. Consequently, if either ILL_0 or ILL_1 is true then the corresponding team has violated the aforementioned restriction L.

Also for Team T_a, we define ILL'_a to be true if restriction L' is violated:

$$ILL'_a = \bigvee_{s\in\{0,1\}} \left(U'(t_{a,s}) \wedge V'(t_{a,s}) \wedge \neg NEXT'_{a,s}(U,V)\right).$$

Now we can define the winning formulae in terms of ILL_a and ILL'_a. Team T_0 wins if any player in Team T_1 violates condition L or condition L'. Hence, WIN_0 and WIN_1 are defined as follows:

$$WIN_0 = ILL_1 \vee ILL'_1,$$
$$WIN_1 = ILL_0 \vee ILL'_0.$$

Given an input $\omega \in \Sigma^n$, let p_0 be the initial position of the formula game G_1 defined previously. Let p'_0 be the initial position of formula game G_2 contain formulae WIN_1 and WIN_0 as defined above, with initial truth assignments of p'_0 as in Table $5_{1,1}$ and $U' = V' = (1,0,0,\ldots,0)$ initially. It can be shown that Team T_1 wins game G_2 from initial position p'_0 if and only if M accepts ω. Thus, by Corollary 2.4.4, G_2 is also a formula game universal for all reasonable games. ∎

DEFINITION 4.2.3. FORMULA GAME G_{2B}. *Let G_{2B} be the blindfold game derived from G_2 by requiring that universal player does not modify any variables visible to Team T_1.*

THEOREM 4.2.2. *G_{2B} is universal for all reasonable blindfold games.*

PROOF. We just need to observe if M would be restricted to a BA_k-TM, and mimic the proof of Theorem 4.2.1. ∎

4.3. Completeness Proofs for TEAM-PEEK Games

We first recall two known results.

THEOREM 4.3.1. *(See [18].) PEEK is DTIME(EXP(n))-complete.*

THEOREM 4.3.2. *(See [3,4].) PRIVATE-PEEK is DTIME($EXP_2(n)$)-complete.*

Peterson and Reif [1] use a TEAM-PRIVATE-PEEK game which is not hierarchical to prove the following undecidability result.

THEOREM 4.3.3. *(See [1].) In general, a TEAM-PRIVATE-PEEK can be undecidable with two or more players on Team T_1.*

We get a hierarchical TEAM-PRIVATE-PEEK if Team T_1 side is restructured so that Player 1 looks over the shoulder of all other players, Player 2 looks over the shoulder of Player 3 through k (but not 1), etc. Hierarchical TEAM-PRIVATE-PEEK is not undecidable, and we can derive its complexity using the formula game G_2 of Section 4.2.

THEOREM 4.3.4. *Hierarchical TEAM-PRIVATE-PEEK is a universal reasonable multiplayer game of incomplete information.*

PROOF. TEAM-PRIVATE-PEEK is practically identical to the game G_2, which has been shown to be universal for hierarchical reasonable games of incomplete information (by Theorem 4.2.1). The variables of G_2 can be put in 1-1 correspondence with the plates in TEAM-PRIVATE-PEEK game box. The variables private to the universal team correspond to the plates not visible by the existential team. The clauses of WIN_1 and WIN_0 can be put in 1-1 correspondence with locations of holes which perforate the plates so that the players can peek from one side to another of the box iff a clause of WIN_a is satisfied. Since G_2 is universal for reasonable games in its class (cf. Theorem 4.2.1), hierarchical TEAM-PRIVATE-PEEK is a universal reasonable multiplayer game of incomplete information. ∎

THEOREM 4.3.5. *Hierarchical TEAM-BLIND-PEEK is a universal reasonable multiplayer blindfold game.*

PROOF. TEAM-BLIND-PEEK is practically identical to the game G_{2B}. In fact, we can establish a 1-1 correspondence between G_{2B} and TEAM-BLIND-PEEK by a mapping analogous to the one used for proof of Theorem 4.3.4. Since G_2 is universal for reasonable games in its class (cf. Theorem 4.2.2), hierarchical TEAM-BLIND-PEEK is a universal reasonable multiplayer blindfold game. ∎

Our log-space reduction from G^M to G^2 has an $O(n \log n)$ length bound. Thus, by Corollary 3.3.1 and Theorem 4.2.1, we can conclude the following.

THEOREM 4.3.6. *There is a D-TM which decides the outcome of G^2 or TEAM-PRIVATE-PEEK in time $T(n)$ then*

$$T(n) > EXP_{k+1}\left(\sqrt{\frac{n}{\log n}}\right).$$

Furthermore, by Corollary 3.3.2 and Theorem 4.2.2, we can conclude the following.

THEOREM 4.3.7. *There is a D-TM which decides the outcome of G^{2B} or TEAM-BLIND-PEEK in space $S(n)$ then*

$$S(n) > EXP_k\left(\sqrt{\frac{n}{\log n}}\right).$$

The following corollaries follows from above theorems.

COROLLARY 4.3.1. *Hierarchical TEAM-PRIVATE-PEEK is DTIME($EXP_{k+1}(O(n))$)-complete.*

COROLLARY 4.3.2. *Hierarchical TEAM-BLIND-PEEK is DSPACE($EXP_k(O(n))$)-complete.*

As a general rule, we can generate games which for unbounded number of players are nonelementary to decide. Hence, we have natural problems with both elementary and nonelementary complexity (in addition to undecidability).

5. TIME BOUNDED MACHINES

This section extends the previous results of Borodin [19,20] and Reif [3,4] to multiplayer games of incomplete information. We shall use dependency quantifier Boolean formula (DQBF) to illustrate the time bounded MPA_k-TMs. We use the results to classify the time complexity of the new machines introduced in this paper.

5.1. Previous Results

DEFINITION 5.1.1. $\Sigma^{T(n)}_{A(n)}$. $\Sigma^{T(n)}_{A(n)}$ *is defined to be the class of languages accepted with time bound $T(n)$ and alternation bound $A(n)$, and existential initial state.*

The following result is due to Borodin [19,20].

THEOREM 5.1.1. *For each $T(n) \geq n$*

$$DSPACE(T(n)) \subseteq \Sigma^{T(n)}_{A(n)} \subseteq NTIME(T(n)A(n)).$$

The proofs of the following two theorems are due to Reif [3,4].

THEOREM 5.1.2. *(See [3,4].) For each $T(n) \geq n$:*

$$BATIME(T(n)) = \Sigma_2^{T(n)}.$$

THEOREM 5.1.3. *(See [3,4].) For each* $T(n) \geq n$

$$PATIME(T(n)) = ATIME(T(n)).$$

Chandra *et al.* [18,2] show that for $T(n) \geq n$

$$NSPACE(T(n)) \subseteq ATIME\left(T(n)^2\right),$$
$$ATIME(T(n)) \subseteq DSPACE(T(n)).$$

Using these results in conjunction with Theorem 5.1.3, we can conclude the following.

COROLLARY 5.1.1. *For* $T(n) \geq n$

$$NSPACE(T(n)) \subseteq PA_1\text{-}TIME\left(T(n)^2\right)$$

and

$$PA_1\text{-}TIME(T(n)) \subseteq DSPACE(T(n)).$$

COROLLARY 5.1.2. *For* $T(n) \geq n$

$$NSPACE(T(n)) \subseteq BA_2\text{-}TIME\left(T(n)^2\right)$$

and

$$BA_2\text{-}TIME(T(n)) \subseteq DSPACE(T(n)).$$

The reader is referred to [21,22] for the case of lower bounds for the PA_1-TMs and the A-TMs with less than n^2 time.

5.2. Dependency Quantifier Boolean Formula

Time bounded complexity for all our machines with three or more players is identical. We also introduce an analog to QBF [18,23] extended to multiplayer games. QBF was originally defined by Henkin, and later discussed by Barwise. Later, QBF was used by Chandra, Kozen and Stockmeyer [18,2] to demonstrate the complexity of the time bounded A-TMs. We shall use dependency quantifier Boolean formula (DQBF) to illustrate the time bounded MPA_k-TMs.

Consider the following QBF formula:

$$\forall X_1 \exists Y_1 \forall X_2 \exists Y_2, \qquad F(X_1, X_2, Y_1, Y_2),$$

where X_1, X_2, Y_1, Y_2 denote the tuples of Boolean variables, and $F(X_1, X_2, Y_1, Y_2)$ denotes some function over the Boolean variables X_1, X_2, Y_1, Y_2. We assume F and the sets of variables has size $O(n)$. Note that selecting Y_1 depends on the value of X_1. Selecting Y_2 depends on both X_1 and X_2. We can denote this in a notation akin to Skolemization by $Y_1(X_1)$ and $Y_2(X_1, X_2)$. This allows us to modify the order of the variables

$$\forall X_1 \forall X_2 \exists Y_1(X_1) \exists Y_2(X_1, X_2), \qquad F(X_1, X_2, Y_1, Y_2).$$

We call a formula in the above form a dependency quantifier Boolean formula (DQBF). DQBF consist of universal variables and existential variables with dependencies, followed by a Boolean function. Whereas all QBF formulae have an associated succinct DQBF formula, the reverse is not true. For example, the following DQBF is not likely to have a succinct QBF:

$$\forall X_1 \forall X_2 \exists Y_1(X_1) \exists Y_2(X_2), \qquad F(X_1, X_2, Y_1, Y_2).$$

All DQBF can be transformed into a functional form. For example, the above formula can be written as follows:

$$\exists G_1 \exists G_2 \forall X_1 \forall X_2, \qquad F(X_1, X_2, G_1(X_1), G_2(X_2).$$

However, the size of G_1 and G_2 can be exponential in the size of their inputs. Hence, this cannot be classified as a succinct representation.

QBF has been shown to be *PSPACE*-complete by Stockmeyer and Meyer [23]. In this paper, we show DQBF to be *NEXPTIME*-complete. Consequently, DQBF is more succinct representation than QBF if we assume that there is a language in *NEXPTIME* which is not in *PSPACE*. We now present our main result for DQBF.

THEOREM 5.2.1. *DQBF is NEXPTIME-complete.*

PROOF. The proof is based on Lemmas 5.2.1 and 5.2.2 presented below. Lemma 5.2.1 shows that DQBF validity $\subseteq$ *NEXPTIME*, and Lemma 5.2.2 shows that DQBF validity is *NEXPTIME*-hard. Note that the reduction is logarithmic space, since we can construct in logarithmic space a linear size DQBF formula that is valid iff the input is accepted by the nondeterministic TM. The theorem follows. ∎

LEMMA 5.2.1. *DQBF validity* $\subseteq$ *NEXPTIME.*

PROOF. Our proof is based on the aforementioned functional interpretation of DQBF formula. We follow a two-phase procedure.

- In the first phase, the functional dependencies (i.e., truth tables) for all existential ($\exists$) variables are guessed. This can be done in no more than nondeterministic exponential time.
- In the second phase, the function is evaluated for all possible assignments to the universal ($\forall$) variables, looking up in the truth tables the values of existential ($\exists$) variables to use in the evaluation. Each evaluation takes at most exponential time, and there are at most an exponential number of these evaluations. Consequently, the time consumed by the second phase is within the nondeterministic exponential bound.

Since each phase take at most nondeterministic exponential time, the problem is within nondeterministic exponential time. ∎

LEMMA 5.2.2. *DQBF validity is NEXPTIME-hard.*

PROOF. We shall prove DQBF is *NEXPTIME*-hard by showing that: given a single tape, nondeterministic exponential time bounded N-TM and an input string, we can construct (in log-space) an $O(n)$ length DQBF formula which is valid if and only if the input is accepted by the N-TM.

The DQBF formula will have the general form

$$\forall T_1, T_2 \exists Y_1(T_1) Y_2(T_2), \qquad F(T_1, T_2, Y_1, Y_2).$$

This formula is in direct correspondence with a three player game where the universal player requests each of the existential players for a complete description of the activities of the N-TM at two times, T_1 and T_2, on an accepting sequence of moves on the input of their choosing. In particular, T_1 and T_2 are each sets of $O(n)$ Boolean variables, that can therefore represent exponential time. The Y_i are tuples of variables representing the information

$$\mathit{OldState}_i, \mathit{NewState}_i, \mathit{OldSym}_i, \mathit{NewSym}_i, \mathit{Head}_i, \mathit{Last}_i, \mathit{Motion}_i,$$

where

$\mathit{OldState}_i$ denotes previous state of the machine,
$\mathit{NewState}_i$ denotes current state of the machine,
OldSym_i denotes previous symbol of the machine,
NewSym_i denotes current symbol of the machine,
Head_i is the head position, and
Last_i is the last time the head was at that position (Last_i is assigned as 0 for the first visit by default), and
Motion_i indicates the direction the head takes at that time (-1 for left, 0 for stationery, $+1$ for right).

$\mathit{OldState}_i$, $\mathit{NewState}_i$, OldSym_i, NewSym_i, and Motion_i need a constant number of variables to represent. Head_i and Last_i need $O(n)$ variables to represent. Consequently, the entire information $\mathit{OldState}_i$, $\mathit{NewState}_i$, $\mathit{OldSym}_i, \mathit{NewSym}_i$, Head_i, Last_i, Motion_i can be represented in $O(n)$ space.

The universal checking of the $\forall$-player is represented by the set of conjunctive clauses forming the function F. We employ the following.

1. Both players choose the same nondeterministic set of moves. Symbolically, if $T_1 = T_2$ then $Y_1 = Y_2$ and the state, symbol, motion transition is allowed. That is to say that both players choose the same nondeterministic set of moves when $T_1 = T_2$.
2. The players initiate correctly, i.e., if $T_1 = 1$ then the $OldState_1$ is the initial state and $Head_1 = 1$ is the initial position, $Last_i = 0$, etc.
3. Whenever two players give the same head location, the time given by the first player for the previous time it was at that position ($Last_1$) cannot be earlier than the time for the second player at that position ($Last_2$). Furthermore, if the times are the same then the symbol written by the second player is the symbol read by the first player.

   In other words, if $Head_1 = Head_2$ and $T_1 > T_2$ then $Last_1 \leq T_2$, and if $T_1 = T_2$ then $OldSym_1 = NewSym_2$.
4. The first visit to a cell gives the old symbol according to the initial contents of the tape, i.e., if $Last_1 = 0$ then $OldSym_1$ is the $Head_1^{\text{th}}$ input symbol if $Head_1 < n$ and blank otherwise.
5. The motion of machine is legal and the state is preserved, i.e., if $T_1 + 1 = T_2$ then $Head_2 = Head_1 + Motion_1$ and $OldState2 = NewState_1$.
6. If $T_1 = T(n)$ (the exponential time limit) then $NewState_1$ is accepting.

Hence, we are existentially choosing the complete operation of the N-TM without writing it all down, and it is being tested by universal quantification.

The lemma follows since we have shown that

$$\forall T_1, T_2 \exists Y_1(T_1) Y_2(T_2), \qquad F(T_1, T_2, Y_1, Y_2),$$

is *NEXPTIME*-hard. ∎

5.3. Time Bounded Machines

We are now ready to classify the time complexity of the new machines introduced in this paper. Unlike space bounded computations, time bounded multiple person alternation games do not form a hierarchy. The MPA_k-TM and the PA_k-TM when $k \geq 2$ as well as the BA_k-TM when $k \geq 3$, are all of same complexity for same time bound. We can prove this by using DQBF verification. Consequently, space is a much more powerful and critical resource than time for the aforementioned types of multiple person alternating Turing machines. There is not a clear division between two person games and three (or more) person games based on time complexity.

LEMMA 5.3.1. *For countable* $T(n) \geq n$ *and* $k \geq 2$

$$NTIME(EXP(T(n))) \subseteq PA_k - TIME(T(n)).$$

PROOF. In a game with two existential players and a universal player, we can simulate the evaluation of the lower bound formula in the DQBF = *NEXPTIME* proof as follows:

1. The universal player sends a time to the $\exists_1$-player.
2. The $\exists_1$-player replies with Y_1 tuple as a response.
3. Similarly, the universal player sends a time to the $\exists_2$-player.
4. The $\exists_2$-player replies Y_2 tuple as a response.
5. The universal player then checks the responses as in the lower bound proof.

Consequently, in time $T(n)$ we can verify the acceptance of an N-TM with time bound $NTIME(EXP(T(n)))$. The lemma follows. ∎

Observe that this game is also hierarchical since the $\exists_1$-player is asked first without it knowing what $\exists_2$ player would be asked. Since the PA_k-TM is a special case of the MPA_k-TM, we can extend Lemma 5.3.1 to the MPA_k-TM.

COROLLARY 5.3.1. *For countable* $T(n) \geq n$ *and* $k \geq 2$

$$NTIME(EXP(T(n))) \subseteq MPA_k\text{-}TIME(T(n)).$$

Similarly, we can construct an analogous result for the BA_k-TM.

COROLLARY 5.3.2. *For countable* $T(n) \geq n$ *and* $k \geq 3$

$$NTIME(EXP(T(n))) \subseteq BA_k\text{-}TIME(T(n)).$$

PROOF. The lower bound proof for BA_k-*TIME* is based on a trick which allows the universal and existential players to get around to the blindfold rule and communicate with each other. The trick enables the universal player to send information to the $\exists_1$-player as well as get its response. Subsequently, the universal player can send information to $\exists_2$-player, and get its response as well.

The $\exists_2$-player is initially shuttling between two designated states on first $2T(n)$ moves. The universal player can advance the $\exists_2$-player to one of those states, advance the $\exists_1$-player, which will see the $\exists_2$-player's state and a information bit has been sent. This process is repeated until all $T(n)$ bits have been sent. The $\exists_1$-player now sends the Y_1-tuple to the universal player via shared tape cells.

The universal player again sends $T(n)$ more pieces of information to the $\exists_1$-player who relays it back to the $\exists_2$-player.

The $\exists_2$-player does not know when $\exists_1$-player makes its move, so it does not know what time was sent to the $\exists_1$-player. Similarly, the $\exists_1$-player has already sent its response when $\exists_2$-player's time is received. Consequently, the hierarchical structure is preserved.

The proof follows from DQBF simulation. ∎

Now, we turn our attention to upper bounds proofs. Since the MPA_k-TM is the most powerful of our machines we need only to show that the MPA_k-TM is contained in $NTIME(EXP(T(n)))$ to complete our results.

LEMMA 5.3.2. *For countable* $T(n) \geq n$, *and* $k \geq 2$

$$MPA_k\text{-}TIME \subseteq NTIME(EXP(T(n))).$$

PROOF. In general, this proof follows the argument used for upper bound proof for the DQBF.

We first modify the machines so that the existential players' strategies only depend on the current visible position. This is accomplished by requiring each player record on extra tapes the complete history of visible information up to that point. This shall consume $O(T(n))$ time per player. The resulting machine is Markov machine. Since each position implicitly contains the history of visible position, Markov machines have the property that the existential players' strategies only depend on the current visible position (not the history of visible positions).

We simulate the running of this machine by first guessing the correct strategy for each player. This is like a "truth table" method where we record what each existential player is supposed to do in each situation. This takes nondeterministic time $2^{O(T(n))}$.

Each player's strategy is written on separate tape in time order. The time is equal to length of history. The machine is now simulated deterministically. The universal player is represented by a set of states as in the previous section. A set of configuration for $\forall$-player takes no longer than $2^{O(T(n))}$ space to write down. The existential players' moves are simulated by looking up the appropriate move for each situation from the tapes.

Each simulated move will be made in time exponential in time $(T(n))$ for either the universal player or the existential player. There are $T(n)$ simulated moves, hence, the total time is nondeterministic exponential in $T(n)$. ∎

THEOREM 5.3.1. *For countable $T(n) \geq n$ and $k \geq 2$*

$$MPA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))),$$
$$PA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))),$$
$$BA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))).$$

PROOF. $MPA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n)))$ follows Corollary 5.3.2 and Lemma 5.3.1. Since the PA_k-TM and the BA_k-TM are special cases of the MPA_k-TM, we can deduce that $PA_k\text{-}TIME(T(n)) \subseteq MPA_k\text{-}TIME(T(n))$, and that $BA_k\text{-}TIME(T(n)) \subseteq MPA_k\text{-}TIME(T(n))$. Consequently, $PA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n)))$ follows from Corollary 5.3.1 and Lemma 5.3.1, $BA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n)))$ follows from Corollary 5.3.2 and Lemma 5.3.1. The proof is now complete. ∎

5.4. Relationship Between Private and Blind Machines

It follows from definition that any BA_k-TM is a PA_k-TM with additional constraints. However, the converse relationship is more involved.

THEOREM 5.4.1. *For any PA_{k-1}-TM, there is an equivalent BA_k-TM.*

PROOF. From Theorem 3.3.4,

$$PA_{k-1}\text{-}SPACE(S(n)) = DTIME(EXP_k(S(n)))$$

and

$$BA_k\text{-}SPACE(S(n)) = DSPACE(EXP_k(S(n))).$$

We also know from fundamental complexity theory that

$$DTIME(EXP_k(S(n))) \supseteq DSPACE(EXP_k(S(n))).$$

Consequently, $PA_{k-1}\text{-}SPACE(S(n)) \supseteq BA_k\text{-}SPACE(S(n)$.

From Theorem 5.3.2, $PA_{k-1}\text{-}TIME(T(n)) = BA_k\text{-}TIME(T(n))$. Combining the time and space bounds, the theorem follows. ∎

6. OTHER ALTERNATING MACHINES

6.1. Alternating Finite State Machines

Intuitively, we would have defined the PA_k-FA and the BA_k-FA to be a constant space PA_k-TM and BA_k-TM, respectively. However, if we do so, these machines would not even be able to accept regular languages. On the other hand, if the input tape were private to the universal player it could be used as a size n upward only counter. Since all players know the input initially, the privacy of input tape to the universal team really amounts to the privacy of input head position for the universal player. Hence, the technique of Section 3 can be applied all over again, since we can count to large numbers. Observe that the universal player can count up to n only, so a game with k existential players can count up to $TWOEXP_k(n)$ with size n counter. Consequently, it is trivial to generalize the results of Section 3.

THEOREM 6.1.1.

$$PA_k\text{-}SPACE(constant) = DTIME(EXP_k(n))$$

and

$$BA_k\text{-}SPACE(constant) = DTIME(EXP_{k-1}(n)),$$

This is where the fact that the BA_k-TMs simulate, and are simulated by, the N-TMs becomes important. For $k = 1$, the nondeterminism cannot be eliminated without affecting the order of growth.

To get the PA_k-TM that accept only regular languages, we need to make the input tape a public resource.

DEFINITION 6.1.1. PA_k-FA. *A PA_k-FA is a constant space PA_k-TM with input tape one-way and public to all.*

DEFINITION 6.1.2. BA_k-FA. *A BA_k-FA is a constant space BA_k-TM with input tape one-way and public to all.*

Now we can characterize the languages accepted by these machines.

LEMMA 6.1.1. *PA_k-FA and BA_k-FA accept only the regular languages.*

PROOF. The extended subset construction of Sections 3.1 and 3.3 when applied to PA_k-FA results in configurations in the modified game tree of constant size. The only nonconstant size of information is the input head position which does not need to be put into configurations since it is public to all. Hence, the whole game tree is constant size, and can be explored for acceptance by an NFA with $EXP_k(n)$ states, where n is the number of states of PA_k-FA.

The proof for BA_k-FA is analogous. ∎

Chandra and Stockmeyer [18] prove that alternating finite automata (A-FA) is exponentially more succinct, in some cases, than NFAs. We show that our PA_k-FAs are even more succinct. Consider the variant of language used by Chandra and Stockmeyer [18]

$$L_n = \{\{0,1\}^* w \{0,1\}^* \$ w \mid w \in \{0,1\}^n\}.$$

It is easy to show (via counting arguments) that L_n is not accepted by any DFA with one-way input head which has fewer than 2^{2^n} states, or any NFA with fewer 2^n states. It can be accepted by an $O(n)$ state A-FA (but not by any A-FA with fewer states). Given PA_k-FA with $O(n)$ states, we can again apply our counting methods again to count up to $TWOEXP_k(n)$. It is a simple matter to use this to accept L_n with a $O(\log n)$ state PA_1-FA. Intuitively, one can expect that L_n can be accepted by $O(\log^k(n))$-state PA_k-FA, where $\log^k(n)$ refers to k repeated logarithms of n. However, not all L_ns can be represented by $O(\log\log n)$ state PA_2-FAs. Some can be accepted by $O(\log\log n)$ state PA_2-FA, for example when n is a power of two. Hence, we have to phrase our main succinctness theorem accordingly.

THEOREM 6.1.2. *$L_{TWOEXP_k(n)}$ is accepted by $O(n)$ state PA_k-FA (and also a BA_k-FA), but no DFA with fewer than $TWOEXP_{k+2}(n)$ states, nor any NFA with fewer than $TWOEXP_{k+1}(n)$ states, nor any A-FA with fewer than $TWOEXP_k(n)$ states. The constant factor for PA_k-FA includes a c^k term.*

As a direct application of PA_k-FA and BA_k-FA, we consider the "(not) emptiness of language" question. Given a $TWOEXP_k(O(n))$ space bounded N-TM and an input, it is quite easy to construct an $O(n)$ state BA_k-FA which accepts only the accepting sequence of configurations of the N-TM on the input, if it exists.

The technique uses the ability to count to large numbers to check length of configurations, corresponding positions in adjacent configurations, etc. Similarly, for PA_k-FA the upper bounds follow from the extended "subset construction" for the DFA, which results in an NFA with $TWOEXP_k(O(n))$ states. Hence, only strings up to that length have to be checked, with the space to write down the string being the dominant factor in the analysis.

THEOREM 6.1.3. *The (not) emptiness of language question is $NSPACE(TWOEXP_k(O(n)))$-complete for PA_k-FA as well as for BA_k-FA, where n is the number of states. In terms of representation of the machine, a $\log n$ factor is introduced using a standard form which reduces the top exponent of the lower bound by a factor of $\log n$.*

Stockmeyer [24] proves the "not empty complement" question for extended regular expressions with nested complements at most k deep is $NSPACE(TWOEXP_k(O(n)))$-complete (with poly-linear reduction). Hence, nesting of players in a hierarchical PA_k-FA is indirectly related to nesting of complements in extended regular expressions.

6.2. Private Pushdown Store Automata

A large number of results based upon alternating versions of pushdown store automata (A-PDA) are due to Ladner, Lipton and Stockmeyer [25,26,27]. Characteristics of several versions resulted in well-defined time and space complexity classes. We do not expect that studying multiplayer PDAs will provide any such insights as even very simple forms accept r.e. languages.

DEFINITION 6.2.1. MPA$_k$-PDA. *An MPA$_k$-PDA is an MPA$_k$-TM with only two tapes: a one way, read only tape public to all players, and pushdown store tape. The pushdown store tape simulates a pushdown store by printing blanks whenever it moves left.*

THEOREM 6.2.1. *All r.e. languages are accepted by MPA$_1$-PDA.*

PROOF. The pushdown store is allowed to be private to the universal player. For a given single tape D-TM, the existential player of the game for that D-TM has the task of sending to the universal player, a character at a time, the sequence of configurations leading to acceptance on the input (if it exists). Every other configuration will be in reverse order: $\langle C_0, C_1^R, C_2, \ldots, C_M^{(R?)} \rangle$. The universal player checks the first against the input to ascertain if it is the initial configuration. It chooses to check two adjacent configurations to verify that one legally follows from the other. The first one it chooses is secretly copied over to the pushdown store as it comes in. It is then popped in phase with the next configuration as it comes in. The player verifies that corresponding tape cells match up right, head motion, and symbol written are correct, etc. The fact that every other configuration is in reverse order enables the MPA$_1$-PDA to use the pushdown store like a stack to check adjacent configurations. Finally, the universal player checks the last tape configuration to ensure that it is associated with an accepting state. ∎

6.3. Markov Alternating Machines

We have referred to Markov alternating Turing machines (MA$_k$-TMs) in several places in this paper. These machines restrict the existential players' access to the history of positions in formulating their strategy. We are interested in obtaining a (nondeterministic) hierarchy of machines to go along with our deterministic time and nondeterministic space hierarchies. While we obtain a nondeterministic time class with a space bounded MA$_1$-TMs, they do not generalize for more players. Our main theorem for a time bounded MA$_k$-TMs is as follows.

THEOREM 6.3.1. *For countable $T(n) \geq n$ and any $k \geq 1$*

$$MA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))).$$

PROOF. The time bounded MA$_k$-TMs represent the PA$_k$-TMs which have written their histories down. Since the MA$_k$-TM is a special case of the MPA$_k$-TM, the upper bound then follows from Theorem 5.3.1., which states that $MPA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n)))$.

For the lower bound, we just need to show that the MA$_1$-TM can play DQBF lower bound game in $O(T(n))$ time. The universal player initiates by sending the existential player the first time, which it writes down. The existential player responds with appropriate Y_1-tuple, and erases its tape. Subsequently, the universal player sends the second time. Observe that the existential player has forgotten the first time, and must respond to as if it were another player (like the second player of the DQBF game). Consequently, it sends the Y_2-tuple. The universal player is responsible for checking both the Y_1 and Y_2 tuples. The theorem follows. ∎

Now, we derive the corresponding space result. Unlike other alternating machines, we find that "*SPACE = TIME*".

THEOREM 6.3.2. *For countable $S(n) \geq \log n$ and any $k \geq 1$*

$$MA_k\text{-}SPACE(T(n)) = NTIME(EXP(S(n))).$$

PROOF. The lower bound simply follows from the previous theorem since a $S(n)$ space bounded machine has at least $S(n)$ time available to it. (The case for $S(n) < n$ is a simple extension to the previous proof.)

For the upper bound, we will first show that any MA_k-TM with space bound $S(n)$ can be simulated by an MA_1-TM with the same space. We arrange for one player to make moves for all players. This technique as in the proof of Theorem 6.3.1. When it is time for $\exists_i$-player to make a move, the universal player sends the existential player of the MA_1-TM the information visible to that player along with that player's number. The existential player must respond as if it were that player by sending the new visible position back to the universal player. Subsequently, it clears its tapes and state to prepare for the next request. This results in the existential player to forget what move it made for that player. It involves no increases in space. To simulate a MA_1-TM with bounded space, we first note that any path of the game tree which is deeper than exponential in $S(n)$, contains an infinite loop. For the MA_k-TMs, unlike the PA_k-TMs, any repetition of configurations (an exponential in space number) causes an infinite loop. If the existential player has a winning strategy, then it must also have one that corresponds with the accepting subtree whose depth is at most exponential in $S(n)$. This, we have to explore the game tree to only a depth of $EXP(S(n))$.

The tree is explored as in proof of Corollary 5.3.3. First, the complete set of moves to be made in any situation by the existential player is written down, taking (nondeterministic) time $EXP(S(n))$. Then, the game tree is explored using the extended "subset construction" to represent the universal player's private configurations. There will be $EXP(S(n))$ steps, each taking time $EXP(S(n))$. The total time therefore, is $NTIME(EXP(S(n)))$. The proof is now complete. ∎

7. CONCLUSION

This paper has provided matching lower bounds for algorithms to decide the outcome of multiplayer game of incomplete information in our paper [1,5]. We define enhanced alternating Turing machines to capture the notions of these multiplayer games. We also extend the idea of multiplayer alternation to other machines, like FA, PDA, and Markov machines.

In general, multiplayer games of incomplete information can be undecidable, unless it 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) increases 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 $S(n)$. The space bound of blindfold multiplayer games is related to deterministic space bound. The main results are summarized in Theorem 3.3.4 as follows.

For $S(n) \geq \log(n)$

$$PA_k\text{-}SPACE(S(n)) = DTIME(EXP_{k+1}(S(n))),$$
$$BA_k\text{-}SPACE(S(n)) = DSPACE(EXP_k(S(n))).$$

Time bounded games are shown not to exhibit such complexity of towering exponentials. The main results are summarized in Theorem 5.3.1, Corollaries 5.1.1, and 5.1.2. For countable $T(n) \geq n$ and $k \geq 2$:

$$NSPACE(T(n)) \subseteq PA_1\text{-}TIME\left(T(n)^2\right) \subseteq DSPACE\left(T(n)^2\right),$$
$$NSPACE(T(n)) \subseteq BA_2\text{-}TIME\left(T(n)^2\right) \subseteq DSPACE\left(T(n)^2\right),$$
$$MPA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))),$$
$$PA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))),$$
$$BA_k\text{-}TIME(T(n)) = NTIME(EXP(T(n))).$$

Two variants TEAM-PRIVATE-PEEK and TEAM-BLIND-PEEK are defined and shown to be universal for their respective classes of games. DQBF is shown to be *NEXPTIME*-complete.

It would be interesting to research more general game theoretic models with arbitrary pay-offs. Our models can be enhanced to reflect a more economics-oriented approach to games.

REFERENCES

1. G.L. Peterson and J.H. Reif, Multiple-person alternation, In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pp. 348–363, (October 1979).
2. A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, *J. of ACM* **28** (1), 114–133, (1981).
3. J.H. Reif, Universal games of incomplete information, In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pp. 288–308, (May 1979).
4. J.H. Reif, The complexity of two-player games of incomplete information, *J. Comp. Sys. Sci.* **29** (2), 274-301, (1984).
5. G.L. Peterson, J.H. Reif and S. Azhar, Decision algorithms for multiplayer noncooperative games of incomplete information, *Computers Math. Applic.* (to appear); `http://www.cs.duke.edu/~reif/paper/games/alg/alg.pdf`.
6. S. Azhar, A. McLennan and J.H. Reif, Computation of equilibria in noncooperative games, TR CS-1991-36, Computer Science Department, Duke University, Durham, NC, (October 1991); `http://www.cs.duke.edu/~reif/paper/games/mixed/mixed.pdf`.
7. T.J. Schaefer, On the complexity of some two player perfect information games, *J. Comput. System Sci.* **16** (2), 185–225, (1978).
8. S. Even and R.E. Tarjan, A combinatorial problem which is complete in polynomial space, *J. of ACM* **23**, 710–719, (1976).
9. 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 the 19th Annual IEEE Symposium on Foundations of Computer Science*, pp. 55–64, (October 1978).
10. J.M. Robson, N by N checkers is Exptime complete, *SIAM J. of Comput.* **13**, 252–267, (1984).
11. D. Lichtenstein and M. Sipser, Go is PSPACE hard, In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pp. 48–54, (October 1978).
12. D. Lichtenstein and M. Sipser, Go is PSPACE hard, *J. of ACM* **27**, 393–401, (1980).
13. A.S. Fraenkel and D. Lichtenstein, Computing a perfect strategy $N \times N$ chess requires time exponential in N, In *Proceedings of the 8th Int'l Coll. Automata, Lang., & Programming*, pp. 278–293, (July 1981).
14. Gary L. Peterson, Succinct representation, random strings, and complexity classes, In *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science*, pp. 86–95, (October 1980).
15. L.J. Stockmeyer and A.K. Chandra, Provably difficult combinatorial games, *SIAM J. Comput.* **8** (2), 151–174, (1979).
16. J. Hartmanis and R.E. Stearns, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* **117**, 285–306, (1965).
17. L.J. Stockmeyer, The polynomial time hierarchy, *Theoretical Computer Science* **3**, 1–22, (1977).
18. A.K. Chandra and L.J. Stockmeyer, Alternation, In *Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science*, pp. 98–108, (October 1976).
19. A. Borodin, Complexity classes of recursive functions and the existence of complexity gaps, In *Proceedings of the 1st Annual ACM Symposium on Theory of Computing*, pp. 67–78, (May 1969).
20. A. Borodin, Complexity classes of recursive functions anf the existence of complexity gaps, *J. of ACM* **19**, 158–174, (1972).
21. W.J. Paul, E.J. Prauss and R. Reischuk, On alternation, In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pp. 113–122, (October 1978).
22. W.J. Paul, E.J. Prauss and R. Reischuk, On alternation, *Acta Inform.* **14**, 243-255, (1980).
23. L.J. Stockmeyer and A.R. Meyer, Word problems requiring exponential time: Preliminary report, In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*, pp. 1–9, (May 1973).
24. L.J. Stockmeyer, The complexity of decision problems in in automata theory and logic, Ph.D. Thesis, MIT Project MAC, TR-133, (July 1974).
25. R.E. Ladner, R.J. Lipton and L.J. Stockmeyer, Alternating pushdown automata, In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pp. 92–106, (October 1978).
26. R.E. Ladner, R.J. Lipton and L.J. Stockmeyer, Alternating Pushdown and Stack Automata, *SIAM J. of Comput.* **13**, 135–155, (1984).
27. R.E. Ladner, L.J. Stockmeyer and R.J. Lipton, Alternation bounded auxiliary Pushdown Automata, *Information and Control* **62**, 93–108, (1984).