SIAM J. COMPUT.
Vol. 8, No. 2, May 1979

0097-5397/79/0802-0005 $01.00/0

PROVABLY DIFFICULT COMBINATORIAL GAMES*

LARRY J. STOCKMEYER† AND ASHOK K. CHANDRA†

Abstract. For a number of two-person combinatorial games, the problem of determining the outcome of optimal play from a given starting position (that is, of determining which player, if either, has a forced win) is shown to be complete in exponential time with respect to logspace-reducibility. As consequences of this property, it is shown that (1) any algorithm which determines the outcome of optimal play for one of these games must infinitely often use a number of steps which grows exponentially as a function of the size of the starting position given as input; and (2) these games are "universal games" in the sense that, if G denotes one of these games and R denotes any member of a large class of combinatorial games (including Chess, Go, and many other games of popular or mathematical interest), then the problem of determining the outcome of R is reducible in polynomial time to the problem of determining the outcome of G.

Key words. computational complexity, combinatorial game, completeness in exponential time

1. Introduction. For many combinatorial games of perfect information (for example, Chess, Go, Kayles, and Nim) it is known that there are algorithms which determine whether or not the player moving first has a "forced win" from a given starting position. We say that such an algorithm *decides* the game. For a game such as Go (generalized to boards of arbitrary size) a position is essentially specified by a placement of stones on a board together with an indication of whose turn it is; a position in Nim is a sequence of nonnegative integers represented in, say, binary notation which specifies the number of sticks in each heap. We are interested primarily in the running times of decision algorithms where the time is measured as a function of the size of the starting position given as input. For example, it would be reasonable to define the size of a position in Go to be the number of squares on the board, and the size of a position in Nim to be the sum of the lengths of the binary representations comprising the sequence of heap sizes.

For both Go and Nim, the number of positions which could conceivably be reached from a given position π by one or more moves of the game grows roughly as an exponential function of the size of π. Therefore, Go and Nim can be decided in exponential time by algorithms which list all positions reachable from the input π and then determine the value of each listed position by the methods of classical game theory [24, § 15]. An exponential running time, while prohibitive in practice for all but very small initial positions, does provide a rough upper bound on the time that is sufficient to decide many examples of games.

For certain games this exponential running time can be substantially improved. The known analysis of Nim [3], [5], [11] yields a decision algorithm for Nim whose running time is a polynomial of low degree. The applications of Grundy-Sprague theory [11] and other clever analyses (see, for example, [5]) have produced nonobvious and efficient decision algorithms for a number of games.

However, other games have resisted analysis. It is not known, for example, if there is a decision algorithm for Go whose running time is bounded above by a polynomial in the board size, and it is possible that no such algorithm exists. Recently it has been proved that the decision problems for Go and Checkers are polynomial-space-hard [10], [18]; this provides evidence (but, as yet, not proof) that these games cannot be decided in polynomial time. The main purpose of this paper is to prove that the decision

* Received by the editors February 6, 1978.
† IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.

problems for certain simply-defined combinatorial games are complete in exponential time with respect to efficient reducibility (cf. [1], [22]) and, therefore, that these games require exponential time to decide, at least on some infinite sequence of starting positions. Since we have not been able to prove this for existing games such as Go, we have defined several games for the purpose of illustrating the proof methods.

In § 3 we consider games played on propositional formulas. In these games, a starting position is a propositional formula (or formulas) together with an assignment of truth values to the propositional variables in the formula(s). Two players alternate moves. A player moves by changing the truth values of certain variables subject to the rules of the particular game, and the winner is, for example, the player who first makes the formula *true*. Some of these games have more appealing representations. The following game of Peek is equivalent to one of our formula games and illustrates the kinds of results that are contained herein. A starting position in Peek is a box containing a finite number of horizontal plates which can be pushed in or pulled partially out; each movable plate has exactly two positions, "in" or "out", and we may assume that all plates are initially "in". There is also one immobile plate. Some of the movable plates "belong" to player I and the rest "belong" to player II. The plates have holes cut into them at various places and the locations of all holes are known to both players; see Fig. 1. The two players alternate moves with I moving first. A player moves by either passing, pulling one of his plates out, or by pushing one of his plates in. The game ends at the point when a hole appears through the entire stack of plates, and the winner is the player who made the last move which caused the hole to appear. Define the *size* of a position to be the number of plates. We place no a priori bound on the number of plates, but we assume that there is a fixed constant d such that the number of holes in each plate is at most the multiple d of the number of plates. By encoding each starting position as a string of symbols suitable as input to some formal machine model such as a Turing machine, the set of encodings of starting positions from which player I has a forced win is a set of strings which we denote W(Peek). For example, the encoding could contain, for each plate, a list of pairs of positive integers represented in radix notation which specifies the coordinates of all holes in that plate. The key result is that W(Peek) is complete in exponential time with respect to logspace-reducibility. Briefly, this means

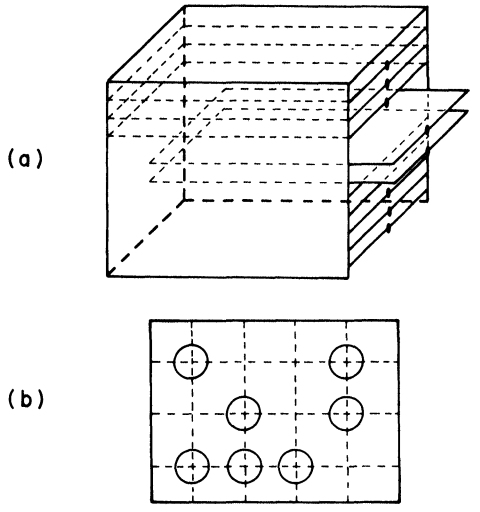

FIG. 1. (a) *A box with ten movable plates, eight "in" and two "out". The top plate is immobile.* (b) *A plate with holes.*

that (i) W(Peek) can be recognized in exponential time, and (ii) if A is any set of strings which can be recognized in exponential time, and (ii) if A is any set of strings which can be recognized in exponential time then A is logspace-reducible to W(Peek)—that is, there is a function f such that $w \in A$ iff $f(w) \in W$(Peek) for all strings w, and f can be computed by a Turing machine within logarithmic space (and, therefore, within polynomial time).

There are two interesting consequences of the fact that W(Peek) is complete in exponential time. First, we are able to derive an exponential lower bound on the time required (infinitely often) to decide Peek. Precisely, there is a constant $c > 1$, such that if M is a deterministic Turing machine which recognizes W(Peek), then there are infinitely many Peek positions π such that M runs for at least $c^{\text{size}(\pi)}$ steps when started on the encoding of π. It should be pointed out that we use Turing machines as our model of algorithm purely for technical convenience in proofs, and that this exponential lower bound (possibly with a different constant $c > 1$) holds for more realistic models such as random access register machines [1], [7]. This is true because there are sufficiently efficient simulations of random access machines by Turing machines.

The second consequence is that Peek is a "universal game" in the sense that the problem of deciding any reasonable game is logspace-reducible to the problem of deciding Peek. Informally, a game is "reasonable" if (i) the number of positions reachable from a given position π within an arbitrary number of moves is bounded above by an exponential function of the size of π, and (ii) it is not onerously difficult to recognize whether a move is allowed by the rules of the game. Many common games such as Chess and Go generalized in any number of ways to arbitrarily large (possibly multidimensional) boards are reasonable in this sense. The formal definition of "reasonable" precedes the statement of Corollary 3.2.

Besides serving as examples of exponential-time-complete games, the formula games of § 3 might be useful in showing that other games are complete in exponential time, in much the same way that the Boolean satisfiability problem [6] was used to show that certain problems are *NP*-complete [16], and the quantified Boolean formula problem [21], [22] was used to show that certain games are complete in polynomial space [8], [20]. To illustrate this, in § 4 we define a type of blocking game played by moving markers on a graph, prove that one of the formula games is logspace-reducible to the blocking game, and conclude that the blocking game is complete in exponential time.

It is instructive to view the results of this paper in the context of previous research [8], [15], [20] concerning the computational complexity of deciding games. One can identify three different types of games corresponding to three levels of complexity. Given a game G and a position π of the game, let *reach*(π) be the number of positions which can possibly be reached from π within an arbitrary number of moves, and let *reach of* G be that function which maps each positive integer s to the maximum of reach(π) taken over all positions π of size s. Games of the first type are those whose reach is bounded above by a polynomial. For example, if a game is played on a graph by moving a single marker from node to node, then the number of reachable positions is at most the number of nodes in the graph. Assuming that the legal moves of the game can be recognized in polynomial time, the straightforward decision algorithm described in the second paragraph of this section shows that any game of the first type can be decided in polynomial time. Jones and Laaser [15] describe a particular game of this type which is *complete* in deterministic polynomial time. Games of the second type are those like Hex and Dots-and-Boxes where players make permanent marks on a board. The distinguishing feature of games of the second type is that if the game is played from a

starting position π, then the game is assured to end after a number of moves which is bounded above by some polynomial in the size of π. However, there is a game of the second type which is not of the first type because its reach grows exponentially (even though the number of positions visited during a *particular* line of play is at most polynomial). By exhaustively examining all possible lines of play from a given starting position, it can be seen that any game of the second type can be decided by an algorithm which uses space (i.e., memory) bounded above by a polynomial in the size of the starting position (assuming again that the legal moves can be recognized in polynomial time). Even and Tarjan [8] and Schaefer [20] exhibit games of the second type which are *complete* in polynomial space. It follows that these polynomial-space-complete games can be decided in polynomial time if and only if any set recognizable in polynomial space is recognizable in polynomial time; this is viewed as providing evidence that these games cannot be decided in polynomial time. Games of the third type are those like Chess and Go where players can move, place, and remove pieces on a board. Games of the third type are those with exponentially bounded reach. There is a game of the third type which is not of the second type because its play lasts an exponential number of moves. Any game of the third type can be decided in exponential time by the straightforward position-listing algorithm. The purpose of this paper is to exhibit games of the third type which are *complete* in exponential time.

The relationship between the three types of games and their decision algorithms is summarized in Fig. 2, where we define the *space* of a game to be the logarithm of its reach (which measures, as a function of the size of π, the number of bits which is sufficient to assign a distinct binary string to each position reachable from π). These are but three instances of a general relationship between time (space) bounded games and space (time) bounded algorithms which is formalized in [4], [17] and outlined in the next section.

TYPE	GAME	ALGORITHM
1	LOGARITHMIC SPACE	POLYNOMIAL TIME
2	POLYNOMIAL TIME	POLYNOMIAL SPACE
3	LINEAR SPACE	EXPONENTIAL TIME

FIG. 2. *The relationship between resource bounds for games and algorithms.*

2. Games, algorithms, and completeness. For a finite alphabet Σ, Σ^* denotes the set of words (i.e., finite strings of symbols) over Σ including the empty word ε; $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. A *language* is a subset of Σ^+ for some finite Σ. For a word $w \in \Sigma^*$, $|w|$ denotes the length of w. **N** denotes the nonnegative integers, and **R** denotes the real numbers. For a finite set S, card(S) denotes the cardinality of S.

It is convenient to adopt the following definition of game.

DEFINITION. A (*two-person perfect-information*) *game* is a triple (P_1, P_2, R) where P_1 and P_2 are sets, $P_1 \cap P_2 = \varnothing$, and $R \subseteq P_1 \times P_2 \cup P_2 \times P_1$.

In other words, P_1 (P_2) is the set of positions in which player I (II) has the initiative, and R is the set of allowable moves—if $(\pi, \pi') \in R$ and $\pi \in P_1$ ($\pi \in P_2$), then player I (II)

can move from position π to position π' in one move. By convention, a player who is unable to move is declared the loser.

DEFINITION. Let $G = (P_1, P_2, R)$ be a game. Let $W_{-1}(G) = \varnothing$ and for integer $i \geqq 0$ let

$$W_i(G) = W_{i-1}(G) \cup \{\pi \in P_1 | (\exists \pi' \in P_2)[(\pi, \pi') \in R \text{ and } \pi' \in W_{i-1}(G)]\}$$
$$\cup \{\pi \in P_2 | (\forall \pi' \in P_1)[(\pi, \pi') \in R \text{ implies } \pi' \in W_{i-1}(G)]\}.$$

Define

$$W(G) = \bigcup_{i \geqq 0} W_i(G).$$

$W(G)$ is the set of positions from which player I has a "forced win". In particular, $W_0(G)$ is the set of $\pi \in P_2$ from which player II cannot move, and $W_i(G)$ is the set of positions from which I has a forced win in no more than i moves. For certain games G, our objective is to establish bounds on the computational complexity of recognizing the set $W(G)$.

In measuring the computational complexity of sets, our model of computation is the deterministic one-tape Turing machine [1], [13]. We first define a more general device, the alternating Turing machine, which is a useful technical tool in the proofs of our results. The definition of an alternating Turing machine is very similar to that of a nondeterministic Turing machine (cf. [1], [13]) except that some subset of its states are referred to as *universal* states and the rest as *existential* states. Alternating Turing machines are discussed in more detail in [4], [17]. We state here a somewhat simplified version of the definition which is sufficient for the purposes of this paper.

DEFINITION. A one-tape *alternating Turing machine* (ATM) is a seven-tuple $M = (Q, \Gamma, \Sigma, \#, \delta, q_0, U)$ where:

Q	is the set of states;
Γ	is the tape alphabet;
Σ	is the input alphabet, $\Sigma \subseteq \Gamma$;
$\#$	is the blank tape symbol, $\# \in \Gamma - \Sigma$;
δ	$\subseteq (Q \times \Gamma) \times (Q \times (\Gamma - \{\#\}) \times \{\text{L, R, S}\})$ is the next-move relation;
q_0	is the initial state;
U	is the set of universal states, $U \subseteq Q$;
$Q - U$	is the set of existential states.

The tape is assumed to be one-way infinite to the right, and we assume that the head never moves off the left end of the tape.

A *configuration* of M is a triple of the form (q, γ, j) where $q \in Q$ is the current state, $\gamma \in (\Gamma - \{\#\})^*$ denotes the nonblank portion of the tape, and $j \geqq 1$ is an integer which indicates that the jth tape cell from the left end is currently being scanned; let $\mathscr{C}_M$ denote the set of all such configurations. For an input $w \in \Sigma^+$, the *initial configuration on* w is $(q_0, w, 1)$. If M is in state q scanning the symbol $u \in \Gamma$, and if $((q, u), (q', u', d)) \in \delta$, then M can in one step enter state q', print u' on the tape, and shift the head in direction d (Left, Right, or Stationary). For configurations C and C' we write $C \vdash_M C'$ iff C can reach C' in one step as just described; $\vdash_M^*$ denotes the reflexive transitive closure of $\vdash_M$. The configuration (q, γ, j) is a *universal* (*existential*) *configuration* if q is a universal (existential) state.

Several equivalent definitions of acceptance for ATM's are discussed in [4], [9], [17]. The following definition was suggested by M. Fischer and R. Ladner [9].

DEFINITION. Let M be an ATM. A *trace of* M is a set $\mathrm{Tr} \subseteq \mathscr{C}_M \times \mathbf{N}$ such that:

1) if $(C, k) \in \mathrm{Tr}$ and C is a universal configuration, then $(C', k-1) \in \mathrm{Tr}$ for all C' such that $C \vdash_M C'$; and
2) if $(C, k) \in \mathrm{Tr}$ and C is an existential configuration, then there exists a C' such that $C \vdash_M C'$ and $(C', k-1) \in \mathrm{Tr}$.

M accepts $w \in \Sigma^+$ iff there is a trace Tr of M and a $k \in \mathbf{N}$ such that $((q_0, w, 1), k) \in \mathrm{Tr}$; in this case, Tr is said to be an *accepting trace for w*.

Let $L(M)$ denote that subset of Σ^+ which M accepts.

Note. A configuration C is said to be *halting* if there is no C' such that $C \vdash_M C'$. Universal halting configurations serve as "accepting configurations" since such configurations can belong to any trace. Existential halting configurations serve as "rejecting configurations" since such configurations belong to no trace. Thus, the definition of ATM given above need not mention accepting and rejecting states explicitly.

Let $t, s \in \mathbf{N}$. The trace Tr *uses time at most t* iff $k \leqq t$ for all $(C, k) \in \mathrm{Tr}$. The trace Tr uses *space at most s* iff $j \leqq s$ for all $((q, \gamma, j), k) \in \mathrm{Tr}$.

DEFINITION. Let $F: \mathbf{N} \to \mathbf{R}$ and let M be an ATM. *M operates within time (space) $F(n)$* iff for each $w \in L(M)$ there is a trace Tr of M such that Tr is an accepting trace for w and Tr uses time (space) at most $F(|w|)$. Define

$$\mathrm{ATIME}(F(n))(\mathrm{ASPACE}(F(n))) = \{L(M) \mid M \text{ is an ATM which operates within time (space) } F(n)\}.$$

A *deterministic Turing machine* (DTM) is an ATM M such that for any configuration C of M there is at most one C' such that $C \vdash_M C'$. When restricted to DTM's, the above definitions of time and space bounded acceptance of languages are identical to the usual definitions [1], [13]. (Although nondeterministic Turing machines play no role in this paper, it might aid the reader's intuition to note that one could define a nondeterministic Turing machine to be an ATM with the restriction that every universal configuration is halting.) Define

$$\mathrm{DTIME}(F(n))\,(\mathrm{DSPACE}(F(n))) = \{L(M) \mid M \text{ is a DTM which operates within time (space) } F(n)\}.$$

Also let

$$\mathscr{P}\text{-TIME} = \bigcup_{c,k \geqq 1} \mathrm{DTIME}(cn^k), \qquad \mathscr{P}\text{-SPACE} = \bigcup_{c,k \geqq 1} \mathrm{DSPACE}(cn^k),$$

$$\mathscr{E}\text{-TIME} = \bigcup_{c \geqq 1} \mathrm{DTIME}(c^n).$$

The connection between space-bounded ATM's and time-bounded DTM's is embodied in the following result.

THEOREM 2.1 (Chandra, Kozen, Stockmeyer [4], [17]). *Let $F(n) \geqq n+1$.*

$$\mathrm{ASPACE}(F(n)) = \bigcup_{c \geqq 1} \mathrm{DTIME}(c^{F(n)}).$$

To be completely precise, an ATM is defined in [4], [17] to have a separate input tape, and Theorem 2.1 is proved for all $F(n) \geqq \log n$; the case $F(n) = \log n$ is implicit in Jones and Laaser [15, Thm. 13]. In the case that $F(n) \geqq n+1$, the presence or absence of an input tape is immaterial, so Theorem 2.1 follows trivially from [4], [17]. We are interested primarily in the following corollary of Theorem 2.1.

COROLLARY 2.1. ASPACE$(n+1) = \mathscr{E}$-TIME.

In § 3 we prove that the sets $W(G)$ for certain games G are complete in $\mathscr{E}$-TIME by exploiting the natural connection between ATM's and games. Briefly, the existential (universal) configurations of the ATM correspond to positions from which player I (player II) has the initiative to move, and the universal halting configurations correspond to immediate losing positions for II.

Remark. The papers [4], [17] also characterize time-bounded ATM's in terms of space-bounded DTM's. In particular,

$$\bigcup_{c,k\geqq 1} \text{ATIME}(cn^k) = \mathscr{P}\text{-SPACE}.$$

This equality embodies the connection between $\mathscr{P}$-SPACE and "polynomial-time bounded games" (games of the second type in § 1) which is exploited by Even and Tarjan [8] and Schaefer [20] in proving that certain games are complete in $\mathscr{P}$-SPACE.

Finally we define the notion of a language being *complete* in a class of languages. Let $\log n$ denote the base two logarithm for $n \geqq 2$, and $\log 0 = \log 1 = 1$. Let Σ and Δ be finite alphabets. The function $f\colon \Sigma^+ \to \Delta^+$ is *logspace-computable* (cf. [14], [15], [22]) if there is a deterministic Turing machine with a separate two-way read-only input tape, a read/write work tape, and a one-way output tape such that, when started with any word $w \in \Sigma^+$ on the input tape, the machine eventually halts with $f(w)$ on the output tape while having visited at most $\log |w|$ squares on the work tape. Let $l\colon \mathbf{N} \to \mathbf{R}$. The function f is *length $l(n)$ bounded* iff $|f(w)| \leqq l(|w|)$ for all $w \in \Sigma^+$.

Let $A \subseteq \Sigma^+$ and $B \subseteq \Delta^+$. *A transforms to B within logspace via f* ($A \leqq_{\log} B$ *via* f) iff f is a logspace-computable function, $f\colon \Sigma^+ \to \Delta^+$, such that $w \in A \leftrightarrow f(w) \in B$ for all $w \in \Sigma^+$. We remark that the class of logspace-computable functions is closed under composition [14], [22], so that $\leqq_{\log}$ is a transitive relation on languages.

Let B be a language and let $\mathscr{L}$ be a class of languages. Then

$$\mathscr{L} \leqq_{\log} B \quad \text{iff} \quad A \leqq_{\log} B \quad \text{for all } A \in \mathscr{L}.$$

Furthermore, $\mathscr{L} \leqq_{\log} B$ *via length order* $l(n)$ ($l\colon \mathbf{N} \to \mathbf{R}$) provided that for each $A \in \mathscr{L}$ there is a function f and a constant $b \in \mathbf{N}$ such that $A \leqq_{\log} B$ via f and f is length $b \cdot l(n)$ bounded.

The language B is *log-complete in* $\mathscr{L}$ iff both $B \in \mathscr{L}$ and $\mathscr{L} \leqq_{\log} B$.

3. Games on propositional formulas. In this section we describe six games which are played on propositional formulas and prove that their sets of winning positions are log-complete in $\mathscr{E}$-TIME. By *formula* we mean a well-formed parenthesized expression involving variable symbols (which are denoted in the text by (subscripted) letters t, u, v, x, y, z), the binary connectives $\wedge$ (conjunction), $\vee$ (disjunction) and $\oplus$ (exclusive- or), the unary connective $\sim$ (negation), and parentheses. We define the class of formulas and simultaneously define $V(F)$, the set of variable symbols in F, and size(F), the number of occurrences of variable symbols in F.

DEFINITION. 1) If x denotes a variable symbol, then x is a *formula*, $V(x) = \{x\}$, and size$(x) = 1$;

2) if $F = (G \mathbin{@} H)$ where G and H are formulas and @ denotes a binary connective, then F is a *formula*, $V(F) = V(G) \cup V(H)$, and size(F) = size(G) + size(H);

3) if $F = \sim H$ where H is a formula, then F is a *formula*, $V(F) = V(H)$ and size(F) = size(H).

When writing formulas in the text, parentheses are deleted when not needed to determine the precedence of operations. If $X_1, \cdots, X_m$ denote disjoint sets of variable

symbols, we let $F(X_1, \cdots, X_m)$, $H(X_1, \cdots, X_m)$, etc., denote formulas containing only variables in $X_1 \cup \cdots \cup X_m$. For a set S of variable symbols, an *S-assignment* is a function from S to $\{0, 1\}$, where 0 and 1 in this context denote Boolean values *false* and *true*, respectively. A formula F defines, in the obvious way, a function mapping $V(F)$-assignments to $\{0, 1\}$. A *literal* is either x or $\sim x$ where x denotes a variable symbol. A formula is in *conjunctive normal form* (CNF) iff it is a conjunction of disjunctions of literals. A formula is in *disjunctive normal form* (DNF) iff it is a disjunction of conjunctions of literals. For positive integer k, let kDNF denote the set of formulas in DNF which are of the form $C_1 \vee C_2 \vee \cdots \vee C_m$ where each C_i $(1 \leqq i \leqq m)$ is a conjunction of at most k literals; kCNF is defined dually.

We now describe games $G_k = (P_{k1}, P_{k2}, R_k)$ for $1 \leqq k \leqq 6$. We prefer to describe the move-relations R_k informally, and in several cases we indicate the formal definition as well; it should be obvious how to translate these descriptions into complete formal definitions of the R_k as subsets of pairs of positions. In these games, each position contains a symbol $\tau \in \{1, 2\}$ which serves only to differentiate the positions in P_{k1} from those in P_{k2}.

G_1: A position is a triple $(\tau, F(X, Y, \{t\}), \alpha)$ where $\tau \in \{1, 2\}$, F is a formula in 4CNF whose variables have been partitioned into disjoint sets X, Y, and $\{t\}$, and α is a $V(F)$-assignment. Player I moves by setting t to 1 (*true*) and setting the variables in X to any values; player II moves by setting t to 0 (*false*) and setting the variables in Y to any values. A player loses if the formula F is *false* after his move.

G_2: A position is a 4-tuple $(\tau, \text{I-WIN}(X, Y), \text{II-WIN}(X, Y), \alpha)$ where $\tau \in \{1, 2\}$, I-WIN and II-WIN are formulas in 12DNF, and α is an $(X \cup Y)$-assignment. Player I (II) moves by changing the value assigned to at most one variable in X (Y); either player may pass since changing no variable amounts to a "pass". Player I (II) wins if the formula I-WIN (II-WIN) is *true* after some move of player I (II). More precisely, player I can move from (1, I-WIN, II-WIN, α) to (2, I-WIN, II-WIN, α') in one move iff α' differs from α in the assignment given to at most one variable in X and II-WIN is *false* under the assignment α; the moves of player II are defined symmetrically.

G_3: A position is a 4-tuple $(\tau, \text{I-LOSE}(X, Y), \text{II-LOSE}(X, Y), \alpha)$ where $\tau \in \{1, 2\}$, I-LOSE and II-LOSE are formulas in 12DNF, and α is an $(X \cup Y)$-assignment. Player I (II) moves by changing the value assigned to *exactly* one variable in X (Y) (i.e., passing is not allowed). Player I (II) loses if the formula I-LOSE (II-LOSE) is *true* after some move of player I (II). More precisely, player I can move from (1, I-LOSE, II-LOSE, α) to (2, I-LOSE, II-LOSE, α') iff α and α' differ in the assignment to exactly one variable in X and I-LOSE is *false* under the assignment α'.

G_4: A position is a triple $(\tau, F(X, Y), \alpha)$ where F is a formula in 13DNF and τ and α are as in game G_2. Player I (II) moves by changing at most one variable in X (Y); passing is allowed. The game ends at the point when F first becomes *true* and the winner is the player who made the last move which caused F to become *true*. In other words, a player has no legal move from a position in which F is *true*.

G_5: A position is a triple $(\tau, F(X, Y), \alpha)$ where F is a formula and τ and α are as in G_2. Player I (II) moves by changing at most one variable in X (Y); passing is allowed. Player I wins if the formula F ever becomes *true*. In other words, player II cannot move from $(2, F, \alpha)$ to $(1, F, \alpha')$ if F is *true* under α, but I can always move from $(1, F, \alpha)$ to $(2, F, \alpha')$ provided only that α and α' differ in the assignment to at most one variable in X.

G_6: Game G_6 is identical to G_5 except that F is restricted to be in CNF.

Note that the game of Peek described in the Introduction is merely a restatement of that game which is identical to G_4 except that F can be any formula in DNF. Among

these six games G_4 and G_6 stand out because they are played on one formula which is in restricted form (i.e., DNF or CNF) and the artificial "turn variable" t of G_1 is not involved. The games G_1, G_2, and G_5 are included since they arise as useful intermediate steps toward the proofs that G_4 and G_6 are $\mathscr{E}$-TIME-complete. The game G_3, a minor variant of G_2, is used in § 4.

In order to discuss the complexity of the sets $W(G_k)$, we must first encode the positions of these games as words over some fixed finite alphabet. The details of this encoding are for the most part immaterial, and we do not define the encoding formally. We do assume, however, that variable symbols are encoded as words over a finite alphabet by writing subscripts in binary notation; for example, $x_{5,6}$ would be encoded as $x101¢110$. This encoding can be extended in a natural way to give encodings of formulas and assignments (cf. [1], [6], [21]) and ultimately of positions. Since subscripts are written in binary, we assume that there is a constant $e > 0$ such that, if $|F|$ denotes the length of the encoding of the formula F, then

$$|F| \leqq e \cdot \text{size}(F) \cdot \log(\text{size}(F)); \tag{3.1}$$

and

$$\text{card}(V(F)) \leqq e \cdot |F|/\log(|F|); \tag{3.2}$$

and the length of the encoding of an S-assignment is at most $e \cdot \text{card}(S) \cdot \log(\text{card}(S))$. Fix some encoding with these properties, and let $EW(G_k)$ denote the set of encodings of positions in $W(G_k)$.

THEOREM 3.1. *For $1 \leqq k \leqq 6$, $EW(G_k)$ is log-complete in $\mathscr{E}$-TIME.*

Theorem 3.1 is immediate from Lemmas 3.1 and 3.2 below. The first lemma shows that each $EW(G_k)$ belongs to $\mathscr{E}$-TIME.

LEMMA 3.1. *There is a constant $d > 1$ such that*

$$EW(G_k) \in \text{DTIME}(d^{n/\log n}) \quad \textit{for } 1 \leqq k \leqq 6.$$

Proof. We consider the case $k = 1$; the proof in the other cases is virtually identical. Let w be a given input which encodes the position $\pi = (\tau, F, \alpha)$ and let $n = |w|$. Let

$$P = \{(\tau, F, \beta) \mid \tau \in \{1, 2\} \text{ and } \beta \text{ is a } V(F)\text{-assignment}\}.$$

By the convention (3.2) concerning encodings,

$$\text{card}(P) \leqq p = \lceil 2 \cdot 2^{en/\log n} \rceil.$$

The DTM which accepts $EW(G_1)$ first constructs the sets $W_i(G_1) \cap P$ for $0 \leqq i \leqq p$ using repeated application of the inductive definition of W_i, and then checks whether or not $\pi \in W_p(G_1) \cap P$. The time to carry out this procedure is clearly bounded above by some polynomial in np, and the conclusion follows. □

LEMMA 3.2.

$$\mathscr{E}\text{-TIME} \leqq_{\log} EW(G_k) \quad \textit{via length order } n \log n, \quad \textit{for } 1 \leqq k \leqq 5;$$

$$\mathscr{E}\text{-TIME} \leqq_{\log} EW(G_6) \quad \textit{via length order } n^3 \log n.$$

Lemma 3.2 is proved below. Several remarks and corollaries precede the exposition of this proof.

Lemma 3.2 combines with the hierarchy theorem for deterministic time complexity, proved by Hartmanis and Stearns [12], to yield an exponential lower bound on the time required (infinitely often) to accept $EW(G_k)$.

COROLLARY 3.1. *Let* $1 \leqq k \leqq 5$. *There is a rational constant* $c > 1$, *such that if a deterministic Turing machine accepts* $EW(G_k)$ *and operates within time* $T(n)$, *then* $T(n) > c^{n/\log n}$ *for infinitely many* n.

Proof. From [12] there is a language A such that $A \in \mathscr{E}$-TIME, and if a DTM accepts A and operates within time $T(n)$ then $T(n) \geqq 2^n$ for infinitely many n. By Lemma 3.2, $A \leqq_{\log} EW(G_k)$ via f where f is length $bn \log n$ bounded for some constant b; say that $b \geqq 1$. Choose $c > 1$ so that $c^b < 2$. Assume for contradiction that there is a DTM which accepts $EW(G_k)$ within time $T(n)$ where $T(n) \leqq c^{n/\log n}$ for almost all n. Since f is computable in polynomial time, it follows that there is a DTM which accepts A within time $T'(n)$ where

$$T'(n) \leqq c^{bn(\log n)/\log(bn \log n)} + p(n)$$

for almost all n, where $p(n)$ is a polynomial. By our choice of c, $T'(n) < 2^n$ for almost all n, contradicting one condition that A was chosen to satisfy. □

By a virtually identical proof one shows that $EW(G_6)$ requires time $c^{(n/\log n)^{1/3}}$ infinitely often.

The following definition and corollary formalize the assertion made in the Introduction that the formula games G_k are "universal games".

DEFINITION. The game $G = (P_1, P_2, R)$ is *reasonable* if:

1) there is a finite alphabet Σ such that $P_1, P_2 \subseteq \Sigma^+$; and

2) for all $\pi, \pi' \in P_1 \cup P_2$, if $(\pi, \pi') \in R$ then $|\pi| = |\pi'|$; and

3) for some symbol $\$ \notin \Sigma$, the language $\{\pi \$ \pi' \mid (\pi, \pi') \in R\}$ belongs to $\mathscr{P}$-TIME.

The condition 1) is a convenience to dispose of the issue of encoding positions as words. The intent of 2) is that each instance of the game be played on a "board" of fixed (but possibly arbitrary) size; for example, in generalized Go (cf. § 1), once a board size has been chosen the players are not permitted to enlarge the board during the course of play. The condition 3) ensures that the legal moves can be recognized in polynomial time.

COROLLARY 3.2. *If the game* G *is reasonable, then*

$$W(G) \leqq_{\log} EW(G_k) \quad \textit{for } 1 \leqq k \leqq 6.$$

Proof. By the definition of reasonable it is easy to see that $W(G) \in \mathscr{E}$-TIME using the method described in the proof of Lemma 3.1. Now the conclusion is immediate from Lemma 3.2. □

Our aim in this section is simply to illustrate the kinds of games on formulas which are complete in $\mathscr{E}$-TIME rather than to give a full analysis of the various combinations of rules, CNF versus DNF formulas, etc. Two points are worth mention, however. First, the games G_k remain complete in $\mathscr{E}$-TIME if formulas are not restricted to CNF or DNF, or if passing is disallowed, or both. If arbitrary formulas appear in positions of these games, the winning positions can still be accepted within time $d^{n/\log n}$ as the proof of Lemma 3.1 demonstrates. If passing is disallowed, we can give each player an additional variable upon which the formulas do not depend. Secondly, the "turn variable" t appears to be essential to the $\mathscr{E}$-TIME-completeness of games like G_1 where a player can change the assignment to all of his variables in one move. For example, if we define the game G_4' like G_4 except that player I (II) can change the assignment of the entire set $X(Y)$ in one move, then a position $(1, F(X, Y), \alpha)$ belongs to $W(G_4')$ iff F is *false* under α and $(\exists X)[F'(X)]$ where $F'(X)$ is obtained from $F(X, Y)$ by setting the variables in Y according to the assignment α. This problem is trivial for DNF formulas and log-complete in NP [6], [1] for CNF or arbitrary formulas.

If player II has the first move, a position $(2, F(X, Y), \alpha)$ belongs to $W(G_4')$ iff either F is *true* under α or

$$(\forall Y)[\sim F''(Y) \text{ and } (\exists X)[F(X, Y)]]$$

where $F''(Y)$ is obtained from F by setting X according to α. This problem is log-complete in co-*NP* for DNF formulas and log-complete in Π_2^p [21] for CNF or arbitrary formulas.

The remainder of § 3 is devoted to the proof of Lemma 3.2. In this proof it is technically convenient to deal with ATM's of a special type described next. A *standard linear ATM* is an ATM $M = (Q, \Gamma, \Sigma, \not\equiv, \delta, q_0, U)$ with the properties that: (i) for all $w \in \Sigma^+$, when started on input w, M can reach no configuration in which tape cell $|w|+2$ is being scanned (formally, there do not exist $q \in Q$ and $\gamma \in \Gamma^*$ such that $(q_0, w, 1) \vdash_M^* (q, \gamma, |w|+2)$); (ii) the initial state q_0 is existential; and (iii) if C and C' are configurations of M with $C \vdash_M C'$, then C is existential if and only if C' is universal. Because of the following lemma, we can restrict attention to standard linear ATM's in the sequel.

LEMMA 3.3. $\mathscr{E}$-TIME $= \{L(M) \mid M \text{ is a standard linear ATM}\}$.

Proof. In light of Corollary 2.1 it suffices to observe that if A is accepted by an ATM M which operates within space $n+1$, then M can be modified to satisfy the necessary constraints (i), (ii) and (iii), and still accept A. The constraint (i) can be met by modifying M so that it enters a halting existential (i.e., rejecting) configuration whenever the original M would attempt to shift the head to cell $|w|+2$. Now (ii) and (iii) can be met by introducing new states. Say, if $((p, u), (q, u', d)) \in \delta$ where both p and q are existential states, then remove this element from δ and introduce a new universal state p' such that both $((p, u), (p', u', \text{Stationary}))$ and $((p', u'), (q, u', d))$ belong to δ. □

With each standard linear ATM M we associate a game $G_M = (P_{M1}, P_{M2}, R_M)$ as follows: P_{M1} (P_{M2}) is the set of existential (universal) configurations of M, and R_M is the next-move relation $\vdash_M$.

LEMMA 3.4. *Let M be a standard linear ATM and let G_M be the game associated with M.*

$$L(M) = \{w \in \Sigma^+ \mid (q_0, w, 1) \in W(G_M)\}.$$

Proof. The proof follows easily from the definitions of $W(G_M)$ and of acceptance for ATM's, together with the conventions concerning standard linear ATM's. On the one hand, if Tr is a trace of M then one proves by induction on i that

$$\{C \mid (C, i) \in \text{Tr}\} \subseteq W_i(G_M) \quad \text{for all } i \in \mathbf{N}.$$

On the other hand, it also follows from definitions that

$$\{(C, i) \mid i \in \mathbf{N} \text{ and } C \in W_i(G_M)\}$$

is a trace of M. □

Lemmas 3.3 and 3.4 provide the necessary link between $\mathscr{E}$-TIME and games. Note, in particular, the similarity between G_M and G_1. The proof that $\mathscr{E}$-TIME $\leqq_{\log} EW(G_1)$ follows easily from known methods of expressing a Turing machine's next-move relation $\vdash_M$ as a propositional formula [1], [6], [21]. We next formalize this method.

First, we need a convention for representing a configuration by an assignment to a set of Boolean variables. Let M be a standard linear ATM with states Q and tape

alphabet Γ. With each configuration $C = (q, \gamma_1\gamma_2 \cdots \gamma_l, j)$ where $1 \leqq j \leqq l+1$ and $\gamma_i \in \Gamma - \{\#\}$ for all i, we associate the word

$$\omega(C) = \gamma_1\gamma_2 \cdots \gamma_{j-1} q \gamma_j \gamma_{j+1} \cdots \gamma_l.$$

Let $s = s(M) \overset{\text{def}}{=} \lceil \log(\text{card}(Q \cup \Gamma)) \rceil$. Fix a one-to-one map $h_M: (Q \cup \Gamma) \to \{0, 1\}^s$ such that $h_M(\#) = 0^s$ and extend h_M to a map from $(Q \cup \Gamma)^*$ to $\{0, 1\}^*$ in the obvious way. Let $U = \{u_1, \cdots, u_p\}$ be a set of variable symbols where $p \geqq s \cdot (l+1)$. The U-assignment α *represents* C iff

$$\alpha(u_1)\alpha(u_2) \cdots \alpha(u_{s(l+1)}) = h_M(\omega(C)),$$

and $\alpha(u_i) = 0$ for $s(l+1) < i \leqq p$. It is important to note that, once M and h_M have been fixed, each U-assignment represents at most one configuration.

LEMMA 3.5. *Let M be a standard linear ATM. For each $w \in \Sigma^+$ there is a formula $\text{NEXT}_{M,w}(U, V)$ involving the variables $U = \{u_1, \cdots, u_p\}$ and $V = \{v_1, \cdots, v_p\}$ where $p = s(M) \cdot (|w|+2)$ with the following properties. If α_U is a U-assignment and α_V is a V-assignment such that α_U represents a configuration C such that $(q_0, w, 1) \vdash_M^* C$, then $\text{NEXT}_{M,w}$ is* true *under α_U and α_V iff α_V represents a configuration C' such that $C \vdash_M C'$. Moreover,*

$$\text{size}(\text{NEXT}_{M,w}) \leqq c_M |w|$$

for some constant c_M depending only on M, and the function which maps w to the encoding of $\text{NEXT}_{M,w}$ is logspace-computable.

The proof of Lemma 3.5 is not difficult, and the details (in a slightly different context) can be found in [21, Lemma 6.3]. Briefly, the formula $\text{NEXT}_{M,w}$ is a conjunction of $|w|$ subformulas; the ith subformula checks that the ith, $(i+1)$th, $(i+2)$th symbols of $h^{-1}(\alpha_U)$ and $h^{-1}(\alpha_V)$ are consistent with a legal move of M. This can be done in such a way that the size of each subformula is a constant depending only on M.

The next lemma permits us in certain cases to replace arbitrary formulas by formulas in 3CNF while incurring only a constant factor dilation in formula size. The proof, which is not repeated here, is based on a method of Tseitin [23], (cf. [2], [21]) for converting an arbitrary formula to a formula in 3CNF while preserving satisfiability.

LEMMA 3.6. *There is a constant a, such that for any formula $F(S)$, there is a formula $H(S, Z)$ in 3CNF where Z is a set of variables disjoint from S, such that*

1) $\text{size}(H) \leqq a \cdot \text{size}(F)$, *and*
2) *for any S-assignment α, $F(S)$ is* true *under α iff there exists a Z-assignment β such that $H(S, Z)$ is* true *under the combined assignments α and β.*

Moreover, there is a logspace-computable function which maps each F to an H satisfying 1) *and* 2).

We have now collected the technical machinery to be used in the proof of Lemma 3.2. In each case $1 \leqq k \leqq 6$, given a standard linear ATM M and an input w, we describe a position π_w such that M accepts w iff $\pi_w \in W(G_k)$. We also note how the length of the encoding of π_w depends on $|w|$ (with M fixed). If $g_1, g_2: \mathbf{N} \to \mathbf{R}$, then we write $g_1 = O_M(g_2)$ to assert that there is a constant b_M depending only on M such that $g_1(n) \leqq b_M \cdot g_2(n)$ for all $n \geqq 1$. In each case it is not difficult to see that the function mapping w to the encoding of π_w is logspace-computable (given that the functions described in Lemmas 3.5 and 3.6 are logspace-computable) and we let the reader convince himself that these functions are indeed logspace-computable.

Proof of Lemma 3.2.

1. Let M be a standard linear ATM, let w be an input, and let $n = |w|$. Let p, U, V, and $\mathrm{NEXT}_{M,w}(U, V)$ be as in Lemma 3.5 for this M and w. Let $\mathrm{NEXT}(U, V, Z)$ be the formula obtained by applying Lemma 3.6 with $S = U \cup V$ and $F = \mathrm{NEXT}_{M,w}$. Recall that $\mathrm{NEXT}(U, V, Z)$ is in 3CNF and its size is $O_M(n)$. Say that $Z = \{z_1, \cdots, z_k\}$.

Now we describe a formula $F_1(X_1, Y_1, \{t\})$ where $X_1 = \{x_1, \cdots, x_m\}$, $Y_1 = \{y_1, \cdots, y_m\}$, and $m = p + k$. Let $\mathrm{MOVE}_1(Y_1, X_1)$ denote the formula $\mathrm{NEXT}(U, V, Z)$ after substituting y_i for u_i, x_i for v_i $(1 \leqq i \leqq p)$, and x_{p+j} for z_j $(1 \leqq j \leqq k)$. Let $\mathrm{MOVE}'_1(X_1, Y_1)$ denote $\mathrm{NEXT}(U, V, Z)$ after substituting x_i for u_i, y_i for v_i $(1 \leqq i \leqq p)$ and y_{p+j} for z_j $(1 \leqq j \leqq k)$. Let F'_1 denote the formula

$$(\sim t \vee \mathrm{MOVE}_1(Y_1, X_1)) \wedge (t \vee \mathrm{MOVE}'_1(X_1, Y_1)).$$

F'_1 is easily transformed via the distributive laws to an equivalent formula F_1 in 4CNF with $\mathrm{size}(F_1) = O_M(n)$. Let α_1 be an $(X_1 \cup Y_1)$-assignment such that the restriction of α_1 to $\{y_1, \cdots, y_p\}$ represents the initial configuration of M on input w; the assignment to the other variables can be arbitrary. The position π_w is $(1, F_1, \alpha_1)$. Recalling Lemma 3.4, and noting that the legal moves of G_1 mimic the legal moves of G_M, it should be obvious that $\pi_w \in W(G_1)$ iff M accepts w. For example, say that player I is about to move, so that t must be set to 1. To avoid losing, player I must set the variables in X_1 so that $\mathrm{MOVE}_1(Y_1, X_1)$ assumes the value 1. If C is the configuration of M currently represented by the assignment to $\{y_1, \cdots, y_p\}$ then, by Lemmas 3.5 and 3.6, I must choose the X_1-assignment so that the assignment to $\{x_1, \cdots, x_p\}$ represents a configuration C' with $C \vdash_M C'$. The reasoning is similar in the case that II is about to move, except that $\sim t$ is 1 so $\mathrm{MOVE}'_1(\mathrm{X}_1, \mathrm{Y}_1)$ is enabled.

Since $\mathrm{size}(F_1) = O_M(n)$, it follows from the convention (3.1) concerning encodings that the function mapping w to the encoding of π_w is length $b_M n \log n$ bounded for some constant b_M. Since M was arbitrary, we conclude that $\mathscr{E}\text{-TIME} \leqq_{\log} EW(G_1)$ via length order $n \log n$.

2. We introduce some new terminology which will be useful in this part of the proof. Let $G_2 = (P_1, P_2, R)$, and let $\perp \notin P_1 \cup P_2$. A I-strategy (II-strategy) is a total function $\sigma\colon P_1 \cup \{\perp\} \to P_2 \cup \{\perp\}$ $(\sigma\colon P_2 \cup \{\perp\} \to P_1 \cup \{\perp\})$ such that $\sigma(\perp) = \perp$, and, for all $\pi \in P_1$ $(\pi \in P_2)$, if there exists a π' such that $R(\pi, \pi')$ then $R(\pi, \sigma(\pi))$, and if there does not exist a π' such that $R(\pi, \pi')$ then $\sigma(\pi) = \perp$. For a position $\pi_1 \in P_1$, a I-strategy σ_1 and a II-strategy σ_2, define play $(\pi_1, \sigma_1, \sigma_2)$ to be the infinite sequence $\pi_1, \pi_2, \pi_3, \cdots$ where $\pi_{i+1} = \sigma_1(\pi_i)$ for i odd and $\pi_{i+1} = \sigma_2(\pi_i)$ for i even. We say that play$(\pi_1, \sigma_1, \sigma_2)$ *ends* if $\pi_i = \perp$ for some i, and in this case we let last$(\pi_1, \sigma_1, \sigma_2)$ denote that position $\pi_j \neq \perp$ with largest subscript j. For games of perfect information, there is no loss of generality in assuming that players choose their moves by a "strategy" as just defined; for example, it is not difficult to see that, for any game G, the inductive definition of $W(G)$ yields an optimal I-strategy and an optimal II-strategy for G.

Let M be a standard linear ATM and let w be an input. We construct a pair of formulas $\text{I-WIN}(X_2, Y_2)$ and $\text{II-WIN}(X_2, Y_2)$ and an $(X_2 \cup Y_2)$-assignment α_2 such that the position $\pi_w = (1, \text{I-WIN}, \text{II-WIN}, \alpha_2)$ has the following properties (3.3) and (3.4). If $H(X_2, Y_2)$ is a formula, $\pi = (\tau, \text{I-WIN}, \text{II-WIN}, \alpha)$ is a position, and $b \in \{0, 1\}$, we say that π *satisfies* $H = b$ iff H assumes the value b under the assignment α.

(3.3) If M accepts w then there is a I-strategy σ_1, such that for all II-strategies σ_2:
(a) play$(\pi_w, \sigma_1, \sigma_2)$ ends with last$(\pi_w, \sigma_1, \sigma_2) \in P_2$ (i.e., I wins), and last$(\pi_w, \sigma_1, \sigma_2)$ satisfies I-WIN = 1 and II-WIN = 0; and

(b) if player II passes on some move, then I next moves to a position which satisfies I-WIN = 1 and II-WIN = 0.

(3.4) If M does not accept w then there is a II-strategy σ_2, such that for all I-strategies σ_1:

(a) if play$(\pi_w, \sigma_1, \sigma_2)$ ends then last$(\pi_w, \sigma_1, \sigma_2) \in P_1$ (i.e., II wins), and last$(\pi_w, \sigma_1, \sigma_2)$ satisfies I-WIN = 0 and II-WIN = 1; and

(b) if player I passes on some move, then II next moves to a position which satisfies I-WIN = 0 and II-WIN = 1.

In particular, (3.3)(a) and (3.4)(a) imply that M accepts w iff $\pi_w \in W(G_2)$. The properties (3.3) and (3.4) will be useful in proving cases $k = 4, 5$ of Lemma 3.2 where we construct formulas using I-WIN and II-WIN as subformulas. We there use the fact that the game never ends with both I-WIN = 1 and II-WIN = 1, and that if one player passes then the other player wins immediately on the next move.

Let m be as in part 1 for this M and w. I-WIN and II-WIN contain variables $X_2 = \{x_{i,j}\}$ and $Y_2 = \{y_{i,j}\}$ where $1 \leqq i \leqq 2m+2$ and $j = 1, 2$. The sets of variables $X_c = \{x_{i,1} \mid 1 \leqq i \leqq m\}$ and $Y_c = \{y_{i,1} \mid m+2 \leqq i \leqq 2m+1\}$ play the roles of X_1 and Y_1, respectively, in the previous part. However, since the rules of G_2 allow only one variable to be changed in one move, we must constrain the play so that, for example, while I is changing variables in X_c, player II can only change variables not belonging to Y_c. Similar to the proofs in [8], [20], we describe a *legitimate play* such that if both players play legitimately then it is obvious that (3.3) and (3.4) hold, and a player who departs from legitimate play loses after the next move of the other player. When we say that a player *plays a variable* x we mean that the player changes the truth value of x. *Legitimate play* is described as follows:

$i \leftarrow 1$;
loop: I plays exactly one of $x_{i,1}$ or $x_{i,2}$;
II plays exactly one of $y_{i,1}$ or $y_{i,2}$;
$i \leftarrow$ (if $i < 2m+2$ then $i+1$ else 1);
go to loop.

If we assume legitimate play, then as play progresses from $i = 1$ to $i = m$, player I can assign any values to variables in X_c; then as play progresses from $i = m+2$ to $i = 2m+1$, player II can assign any values to variables in Y_c; and so on.

We next describe formulas I-ILL and II-ILL which punish players I and II, respectively, for illegitimate play. We use the symbols a_i, b_i, a_i', and b_i' to denote certain subformulas of I-ILL and II-ILL. For $1 \leqq i \leqq 2m+2$ define

$$a_i = x_{i,1} \oplus x_{i,2},$$

$$b_i = y_{i,1} \oplus y_{i,2}.$$

If we choose the initial assignment so that the a_i and b_i are all 0 initially, then during legitimate play we always have $a_1 a_2 \cdots a_{2m+2}$ and $b_1 b_2 \cdots b_{2m+2}$ in $0^*1^* \cup 1^*0^*$. The formulas a_i' and b_i' detect the boundaries between the blocks of 0's and 1's.

$$a_i' = \begin{cases} a_{i-1} \oplus a_i & \text{if } 2 \leqq i \leqq 2m+2, \\ \sim(a_{2m+2} \oplus a_1) & \text{if } i = 1. \end{cases}$$

The b_i' are defined similarly in terms of the b_i. If we assume legitimate play then: just before a move of I there is a j such that $a_j' = b_j' = 1$ and $a_i' = b_i' = 0$ for all $i \neq j$; and just before a move of II there is a j such that $a_{j+1}' = b_j' = 1$ and $a_{i+1}' = b_i' = 0$ for all $i \neq j$.

(Here and subsequently, subscripts are evaluated modulo $2m+2$ to lie in the range from 1 to $2m+2$.) Define

$$\text{I-ILL} = \bigvee_{1 \leqq i \leqq 2m+2} ((a'_i \wedge a'_{i+1}) \vee (a'_i \wedge b'_{i+1} \wedge \sim b'_{i-1})),$$

$$\text{II-ILL} = \bigvee_{1 \leqq i \leqq 2m+2} ((b'_i \wedge b'_{i+1}) \vee (b'_i \wedge a'_{i+2} \wedge \sim a'_i)).$$

Note that during legitimate play, both I-ILL and II-ILL remain 0. In addition, these formulas satisfy the following properties.

(3.5) Suppose that both players have played legitimately to a position where II is about to move. Then:
- (a) player II cannot in one move reach a position which satisfies I-ILL = 1; and
- (b) any illegitimate nonpassing move of player II reaches a position which satisfies I-ILL = 0 and II-ILL = 1; and
- (c) if II passes, then I can in one move reach a position which satisfies I-ILL = 0 and II-ILL = 1.

To verify (3.5), consult Fig. 3 which depicts a typical situation where II is about to move; in this example, $a'_{j+1} = b'_j = 1$. First note that in order to reach a position which satisfies I-ILL = 1, the move must change both b'_j to 0 and b'_{j+2} to 1; this is impossible in one move, so (a) is true. If the next move of player II changes b_j, then this move is legitimate. If II changes b_{j-1} then the term $(b'_{j-1} \wedge a'_{j+1} \wedge \sim a'_{j-1})$ of II-ILL becomes 1. If II changes b_l with $l \neq j$ and $l \neq j-1$, then the term $(b'_l \wedge b'_{l+1})$ of II-ILL becomes 1. If II passes, then I changes a_{j+1} from 1 to 0 on the next move, so that the term $(b'_j \wedge a'_{j+2} \wedge \sim a'_j)$ of II-ILL becomes 1. In a completely analogous fashion, one verifies the symmetric version of (3.5) where the players I and II are interchanged and the formulas I-ILL and II-ILL are interchanged.

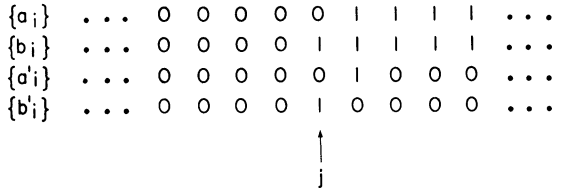

FIG. 3. *A typical situation in G_2 when player* II *is about to move.*

Now let $\text{MOVE}_2(Y_c, X_c)$ and $\text{MOVE}'_2(X_c, Y_c)$ denote the formulas $\text{MOVE}_1(Y_1, X_1)$ and $\text{MOVE}'_1(X_1, Y_1)$ of part 1, after substituting $x_{i,1}$ for x_i and $y_{m+i+1,1}$ for y_i $(1 \leqq i \leqq m)$ in both. The formulas MOVE_2 and MOVE'_2 check that configurations are chosen correctly; these formulas are enabled only at the proper times during play. Let

$$\text{I-WIN}' = (\text{II-ILL} \vee (a'_{2m+2} \wedge b'_{2m+1} \wedge \sim \text{MOVE}'_2(X_c, Y_c))),$$

$$\text{II-WIN}' = (\text{I-ILL} \vee (a'_{m+1} \wedge b'_{m+1} \wedge \sim \text{MOVE}_2(Y_c, X_c))).$$

These two formulas can be transformed to equivalent formulas I-WIN and II-WIN in 12DNF with size(I-WIN) = size(II-WIN) = $O_M(n)$. For example, the terms $(a'_i \wedge a'_{i+1})$ and $(a'_i \wedge b'_{i+1} \wedge \sim b'_{i-1})$ of I-ILL are formulas which involve at most twelve variables (after the abbreviations a'_i and b'_i have been replaced by their definitions). Each term is, therefore, equivalent to a formula in 12DNF, so I-ILL is in 12DNF. Also, $\sim\text{MOVE}_2$

and $\sim\text{MOVE}'_2$ are equivalent to formulas in 3DNF by DeMorgan's laws. Let α_2 be an $(X_2 \cup Y_2)$-assignment such that the assignment to Y_c represents the initial configuration on input w, and $a_i = b_i = 0$ for all i.

The verification of (3.3) and (3.4) is a combination of (3.5) (and its symmetric version) with the fact that legitimate play mimics G_M. Say that M accepts w. Player I's strategy is to play legitimately and play a "side game" of G_M to determine, each time play progresses from $i = 1$ to $i = m$, which configuration to represent by X_c. If II plays legitimately, then just as in part 1, eventually the game will reach a position which satisfies $a'_{2m+2} = b'_{2m+1} = 1$ and $\text{MOVE}'_2(X_c, Y_c) = 0$. Therefore, we need only consider the case that II makes an illegitimate move. If II's first illegitimate move is a pass, then by (3.5) (c), player I can next move to a position (2, I-WIN, II-WIN, α) which satisfies II-ILL = 1 and I-ILL = 0. Since I has been playing legitimately and II passed, we also have that a'_{m+1} and b'_{m+1} are not both 1 under α, so the position satisfies I-WIN = 1 and II-WIN = 0. Say then that II's first illegitimate move is a nonpassing move from position π to π'. By (3.5) (b), the position π' satisfies II-ILL = 1 (and, therefore, I-WIN = 1) and I-ILL = 0. We must check that π' satisfies

$$(a'_{m+1} \wedge b'_{m+1} \wedge \sim\text{MOVE}_2(Y_c, X_c)) = 0.$$

If π satisfies $a'_{m+1} = 0$ we are done. If π satisfies $a'_{m+1} = 1$ then, since both players have been playing legitimately up to π, π satisfies $b'_{m+1} = 0$. Since I has chosen the current assignment to X_c using a winning strategy for G_M, π satisfies $\sim\text{MOVE}_2 = 0$. Since the formula b'_{m+1} contains no variable in Y_c, we conclude that either π' satisfies $b'_{m+1} = 0$ or π' satisfies $\sim\text{MOVE}_2(Y_c, X_c) = 0$. This completes the verification of (3.3). The verification of (3.4) is completely analogous, using the symmetric version of (3.5) (i.e., interchanging I and II), and is left to the reader.

Having noted above that the sizes of I-WIN and II-WIN are $O_M(n)$, it follows that $\mathscr{E}\text{-TIME} \leqq_{\log} W(G_2)$ via length order $n \log n$.

3. Define

$$\text{I-LOSE}' = (\text{I-ILL} \vee (a'_{m+1} \wedge \sim\text{MOVE}_2(Y_c, X_c))),$$

$$\text{II-LOSE}' = (\text{II-ILL} \vee (b'_{2m+2} \wedge \sim\text{MOVE}'_2(X_c, Y_c))).$$

As in part 2, these formulas are equivalent to formulas I-LOSE and II-LOSE in 12DNF of size $O_M(n)$. It is easy to see that M accepts w iff $(1, \text{I-LOSE}, \text{II-LOSE}, \alpha_2) \in W(G_3)$. First recall that, by the rules of G_3, neither player can pass and player I (II) cannot move to a position which satisfies I-LOSE = 1 (II-LOSE = 1). This forces both players to play legitimately, so it should be obvious that this starting position has the property claimed.

4. Let $\text{I-WIN}(X_2, Y_2)$ and $\text{II-WIN}(X_2, Y_2)$ be the formulas described in part 2. We construct a formula $F_4(X_4, Y_4)$ where

$$X_4 = X_2 \cup \{x_1, x_2, x_3, x_4, x_5\} \quad \text{and} \quad Y_4 = Y_2 \cup \{y_1, y_2, y_3, y_4, y_5\}.$$

Let

$$F'_4 = ((y_1 \vee \text{I-WIN}) \wedge (x_2 \vee y_3)) \vee (x_4 \wedge x_5 \wedge \sim y_3) \vee ((x_1 \vee \text{II-WIN}) \wedge (y_2 \vee x_3)) \vee (y_4 \wedge y_5 \wedge \sim x_3).$$

Since I-WIN and II-WIN are formulas in 12DNF of size $O_M(n)$, F'_4 is equivalent to a formula F_4 in 13DNF of size $O_M(n)$. Let α_4 be an assignment that assigns X_2 and Y_2 as in part 2 and assigns x_i and y_i to 0 for $1 \leqq i \leqq 5$. We claim that M accepts w iff $(1, F_4, \alpha_4) \in W(G_4)$.

Say that M accepts w. We describe a winning strategy for player I. As long as II plays only variables in Y_2, I plays a side game of G_2 using the strategy of (3.3) to determine his plays in X_2. Before each of his moves, I switches strategy if one of the following two conditions are met:

1) if II has just played one of the y_i, then I switches to one of the strategies 1a)–1d);
2) if I has a play (possibly a pass) which moves the side game of G_2 to a position which satisfies I-WIN = 1 and II-WIN = 0, then I switches to one of 2a) or 2b).

Note that (3.3) ensures that either 1) or 2) will eventually occur. If 1) and 2) are met simultaneously then 1) takes precedence.

In describing the strategies 1a)–1d) we can assume that none of the x_i have been played and that exactly one of the y_i has been played. Moreover, the current position satisfies I-WIN = 0 and II-WIN = 0 since condition 2) was not met before I's most recent previous move.

1a) If II has just set y_1 to 1, then I sets x_2 to 1 and wins.

1b) If II has just set y_2 to 1, then I sets x_1 to 1 and wins.

1c) If II has just set y_3 to 1, then I views this as a pass by II in the side game; by (3.3) (b), I has a play in X_2 which sets I-WIN = 1, and I wins by making this play.

1d) If II has just set either y_4 or y_5 to 1, then I sets x_3 to 1. Since I has been playing the strategy (3.3) up to this point, II cannot reach a position which satisfies II-WIN = 1 on his next move. Since also y_1, x_2, y_3 and $\sim x_3$ are 0, II cannot set F_4 to 1 in one move. Therefore, player I can set x_1 to 1 on his next move and win.

In describing the strategies 2a) and 2b) we can assume that none of the x_i or y_i have yet been played.

2a) If the current position satisfies I-WIN = 1, then I sets x_2 to 1 and wins.

2b) If the current position satisfies I-WIN = II-WIN = 0, but I can reach a position which satisfies I-WIN = 1 on his next move, then I sets x_4 to 1. Since the variables x_i and y_i for $1 \leqq i \leqq 5$ other than x_4 are all 0, it is easy to check that II cannot reach a position which satisfies $F_4 = 1$ in one move. Now if II does not set y_3 to 1, then I sets x_5 to 1 and wins. If II does set y_3 to 1, then I makes the play in X_2 that sets I-WIN = 1.

This completes the proof that if M accepts w then $(1, F_4, \alpha_4) \in W(G_4)$. The proof of the converse is symmetric utilizing the symmetry between (3.3) and (3.4) and the symmetry in the definition of F_4'.

5. Let the formulas I-WIN and II-WIN be as in part 2. Let $Y_2' = \{y' \mid y \in Y_2\}$. We describe a formula $F_5(X_5, Y_5)$ where $X_5 = X_2 \cup \{x_0\}$ and $Y_5 = Y_2 \cup Y_2' \cup \{y_0, y_1\}$. Let I-WIN$_5$ and II-WIN$_5$ be the formulas I-WIN and II-WIN, respectively, after substituting $(y \oplus y')$ for all occurrences of y for each $y \in Y_2$. Let T denote a formula which is the *exclusive-or* of the variables in $Y_2' \cup \{y_1\}$. In what follows, it is useful to imagine that T is a "variable" and that I-WIN$_5$ and II-WIN$_5$ contain variables in Y_2 just as in part 2. The effect is that in one move player II can either play one $y \in Y_2$ while leaving T fixed, or play T while leaving all $y \in Y_2$ fixed, or *simultaneously* play T and one $y \in Y_2$. Let

$$F_5 = ((T \wedge \text{I-WIN}_5) \vee (\sim T \wedge x_0)) \wedge \sim \text{II-WIN}_5 \wedge (T \vee y_0).$$

Note that size$(F_5) = O_M(n)$. Let α_5 be an assignment which assigns X_2 and Y_2 as in part 2, assigns x_0 to 0, y_0 to 1 and y_1 to 1, and assigns all $y' \in Y_2'$ to 0; note that the "variable" T assumes the value 1 under α_5. We claim that M accepts w iff $(1, F_5, \alpha_5) \in W(G_5)$; furthermore, if M accepts w then I has a winning strategy such that if II passes then I wins immediately on his next move.

Say that M accepts w. As long as II doesn't play T or y_0, I plays a side game of G_2 using the strategy (3.3) to determine his plays in X_2. Eventually the side game will reach a position which satisfies I-WIN = 1 and II-WIN = 0, so this position satisfies $F_5 = 1$ and I wins. If II either passes or plays y_0, then I views this as a pass by II in the side game, and I next moves to a position which satisfies I-WIN = 1 and II-WIN = 0. If II sets T to 0 (possibly in parallel with a move in the side game), then I sets x_0 to 1 and wins.

Say now that M does not accept w. As long as I doesn't play x_0, II plays a side game of G_2 using (3.4) to determine his plays in Y_2. The "variable" T is left fixed at 1 unless II sees that his next play in the side game reaches a position which satisfies II-WIN = 1. In this case, II makes this play while simultaneously setting T to 0. Now I needs at least two moves to reach a position which satisfies $F_5 = 1$ since he must set x_0 to 1 and make some play in X_2 which sets II-WIN to 0. Therefore, before I can win, II can set y_0 to 0 and I cannot win thereafter. If I sets x_0 to 1 before the side game ends in II's favor, then II sets y_0 to 0. Since II has been playing the strategy (3.4) up to this point, I cannot win on his next move, and II sets T to 0 on his next move. If I passes then II passes.

6. Let $F_5(X_5, Y_5)$ be the formula just described in part 5. By invoking Lemma 3.6 with $S = X_5 \cup Y_5$ and $F = F_5$, there is a formula $H(X_5, Y_5, Z)$ in 3CNF with size$(H) = O_M(n)$ such that, for all assignments to X_5 and Y_5,

$$F_5(X_5, Y_5) = 1 \quad \text{iff} \quad (\exists Z)[H(X_5, Y_5, Z) = 1]. \tag{3.6}$$

Say that $Z = \{z_1, \cdots, z_k\}$. Let H' denote the formula H after substituting $(z_i \vee z_i')$ for z_i for $1 \leqq i \leqq k$. Let

$$X_6 = X_5 \cup \{z_i, z_i' \mid 1 \leqq i \leqq k+1\}, \qquad Y_6 = Y_5 \cup \{u_i, u_i' \mid 1 \leqq i \leqq k\},$$

and

$$F_6' = H' \vee (z_1 \wedge \cdots \wedge z_k \wedge z_{k+1} \wedge (u_1 \vee \cdots \vee u_k)) \vee (z_1' \wedge \cdots \wedge z_k' \wedge z_{k+1}' \wedge (u_1' \vee \cdots \vee u_k')).$$

By the distributive laws, F_6' is equivalent to a formula F_6 in CNF with size$(F_6) = O_M(n^3)$. Let α_6 be an assignment that assigns X_5 and Y_5 as in part 5, assigns z_i and z_i' to 0 for all i and assigns u_i and u_i' to 1 for all i. We show that M accepts w iff $(1, F_6, \alpha_6) \in W(G_6)$.

Say that M accepts w. As long as II plays only variables in Y_5, I determines his plays in X_5 by playing a side game of G_5 starting on position $(1, F_5, \alpha_5)$ using the strategy described in part 5. At some point, the side game will reach a position $\pi = (1, F_5, \alpha)$ such that I can move to a position $\pi' = (2, F_5, \alpha')$ where π' satisfies $F_5 = 1$. At the point where such a π occurs, I begins a strategy we call the *end strategy*. By (3.6), there is a Z-assignment ζ such that H assumes the value 1 under the combined assignments α' and ζ. Let $\mathscr{Z} = \{i \mid \zeta(z_i) = 1\}$. To play the end strategy, I does not immediately make the play in X_5 which moves π to π', but rather I first sets z_i to 1 for all $i \in \mathscr{Z}$ on his next card($\mathscr{Z}$) moves. Each time I sets some z_i to 1, II must respond by setting some u_i to 0; for otherwise I can set all the z_i to 1 before II can set all the u_i to 0, and I wins G_6. (We are still assuming that II played only variables in Y_5 up to the point where the side game reached π.) After I has set z_i to 1 for all $i \in \mathscr{Z}$ and II has responded by setting some u_i to 0, I makes the play in X_5 which moves the side game from π to π'; the new position satisfies $H = 1$ and I wins. If II passes (before playing some u_i or u_i') then, as was noted in part 5, the side game is at a position π as above (i.e., I can win G_5 on his next move). Now I plays the end strategy.

If II plays some u_i' before I begins the end strategy, then I views this as a pass by II in the side game, and I plays the end strategy. If II plays some u_i, then again I views this as a pass and plays the end strategy except that the variables z_i' for $i \in \mathscr{Z}$ are set to 1.

Say now that M does not accept w. Player II plays a side game of G_5 to determine his responses to plays of I in X_5. If I plays some z_i (z_i'), then II changes some u_i (u_i') from 1 to 0 if possible, or II passes otherwise. If I passes then II passes.

This completes the proof of Lemma 3.2.

4. Games on graphs. In the previous section we have exhibited several games on propositional formulas which are log-complete in $\mathscr{E}$-TIME. It is possible that these games will be useful as starting points for reductions to other games, in the same way that the quantified Boolean formula "game" [21], [22] has been used to show that $\mathscr{P}$-SPACE is reducible to certain games [8], [10], [18], [20]. Since $\leqq_{\log}$ is a transitive relation, to show that $\mathscr{E}$-TIME $\leqq_{\log} W(G)$ for some game G, it suffices to show that $EW(G_k) \leqq_{\log} W(G)$ where G_k is one of the formula games. The main purpose of this section is to illustrate, for a particular game G on graphs, how a reduction $EW(G_3) \leqq_{\log} W(G)$ can be performed.

Before presenting this example in detail, we remark that by combining Theorem 3.1 with Schaefer's notion of a *pseudoformula* [19] it is easy to devise $\mathscr{E}$-TIME-complete games which are based on known NP-complete problems. For example, the following game HAM is obtained by combining G_5 with the NP-complete Hamiltonian circuit problem for undirected graphs [16].

HAM: A position in HAM is a tuple $(\tau, V, E, E_1, E_2, \alpha_1, \alpha_2)$ where $\tau \in \{1, 2\}$, V is a finite set (the vertices of the graph), $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ (the edges of the graph), $E_1, E_2 \subseteq E$, $E_1 \cap E_2 = \varnothing$, and $\alpha_i: E_i \to \{\text{in, out}\}$ for $i = 1, 2$. Player I (II) moves by either passing or changing the status of one edge in E_1 (E_2) from "in" to "out" or vice versa. Player I wins if, after some move of either player, there is a Hamiltonian circuit in the graph (V, E) (that is, a circuit which contains each vertex exactly once) such that all of the edges currently declared "in" belong to the circuit and none of the edges currently declared "out" belong to the circuit.

For any of the standard methods of encoding graphs as strings of symbols, EW(HAM) is log-complete in $\mathscr{E}$-TIME. By [19, Fact 3.1] it is immediate that $EW(G_5) \leqq_{\log} EW$(HAM), so Theorem 3.1 and the transitivity of $\leqq_{\log}$ imply that $\mathscr{E}$-TIME $\leqq_{\log} EW$(HAM). Also, EW(HAM) $\in \mathscr{E}$-TIME by the method of Lemma 3.1.

We now describe another game, BLOCK, for which the reduction from $W(G_k)$ to W(BLOCK) is more involved. The portion of a position of BLOCK which remains fixed throughout play of the game is referred to as a *board.*

A *board* is a tuple (V, E, ν, W_1, W_2) where:
V is a finite set;
$E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$;
$\nu: E \to \{1, 2, 3\}$; and
$W_1, W_2 \subseteq V$.

V and E should be thought of as the vertices and edges of an undirected graph.

A position of BLOCK is a tuple (τ, B, M_1, M_2) where:
$\tau \in \{1, 2\}$ signifies whose turn it is;
$B = (V, E, \nu, W_1, W_2)$ is a board; and
$M_1, M_2 \subseteq V$ and $M_1 \cap M_2 = \varnothing$.

M_1 and M_2 represent the movable part of a position; M_1 (M_2) should be thought of as a set of markers which belong to player I (II) and are placed on vertices of the board. A player moves by choosing one of his markers and moving it to a new unoccupied vertex by traversing edges of the graph subject to the restrictions that (i) all traversed edges are given the same value by ν, and (ii) no traversed vertex is occupied by a marker of either player. The players are not permitted to pass. The function ν gives a "direction" to each edge; the restriction (i) corresponds, for example, to the situation in Chess that a queen can move in any of four directions but cannot change direction during a single move. Player I (II) wins by placing one of his markers on a vertex in W_1 (W_2). Formally, $(1, B, M_1, M_2)$ can reach $(2, B, M_1', M_2')$ in one move iff $M_2 = M_2'$, $M_2 \cap W_2 = \varnothing$, and there exist $u_1, \cdots, u_k \in V$ and $c \in \{1, 2, 3\}$ such that $u_1 \neq u_k$, $u_1 \in M_1$, $M_1' = (M_1 - \{u_1\}) \cup \{u_k\}$, $u_i \notin M_1 \cup M_2$ for $2 \leqq i \leqq k$, and $\{u_i, u_{i+1}\} \in E$ and $\nu(\{u_i, u_{i+1}\}) = c$ for $1 \leqq i < k$. The legal moves of player II are defined symmetrically.

To encode positions as words over a finite alphabet, associate each element $u \in V$ with a distinct binary word $\omega(u)$ of length roughly $\log(\text{card}(V))$, and list the various elements of a position in some natural way; for example, letting $e = \text{card}(E)$, list $E = \{\{u_1, v_1\}, \cdots, \{u_e, v_e\}\}$ as $\omega(u_1)\$\omega(v_1)\$ \cdots \$\omega(u_e)\$\omega(v_e)$. Let $EW(\text{BLOCK})$ denote the set of encodings of positions in $W(\text{BLOCK})$.

THEOREM 4.1. *EW(BLOCK) is log-complete in $\mathscr{E}$-TIME.*

Proof. By the algorithm of Lemma 3.1 and the method of encoding positions, one finds that there is a constant d such that

$$EW(\text{BLOCK}) \in \text{DTIME}(d^{n/\log n}).$$

To show that $\mathscr{E}\text{-TIME} \leqq_{\log} EW(\text{BLOCK})$ it suffices to prove

$$EW(G_3) \leqq_{\log} EW(\text{BLOCK}). \tag{4.1}$$

Let

$$\pi = (\tau, \text{I-LOSE}(X, Y), \text{II-LOSE}(X, Y), \alpha)$$

be a given position of G_3. We describe a position

$$\pi' = (\tau, (V, E, \nu, W_1, W_2), M_1, M_2)$$

in BLOCK such that $\pi \in W(G_3)$ iff $\pi' \in W(\text{BLOCK})$.

Say that $X = \{x_i \mid 1 \leqq i \leqq m_1\}$ and $Y = \{y_i \mid 1 \leqq i \leqq m_2\}$. Let $\bar{x}_i$ denote the literal $\sim x_i$ and let $\bar{y}_i$ denote $\sim y_i$. For each variable x_i and y_i the graph (V, E) contains the subgraph depicted in Figs. 4(a) and 4(b), respectively. In these figures, vertices are labeled by subscripted lower case Roman letters. The value of each edge under ν is indicated by drawing the edge as either solid, dashed, or dotted. Vertices which belong to W_1 (W_2) are indicated by open (solid) stars. The markers of player I (II) are shown as open (solid) circles. The dashed edges in these figures connect these subgraphs to the remainder of the graph, and these are the only such connections. The rest of the graph is constructed in such a way that (i) neither y_i nor $\bar{y}_i$ (resp., x_i nor $\bar{x}_i$) is connected to a vertex in W_2 (resp., W_1) by a path of dashed edges, and (ii) any attempt by either player to move a marker from outside one of these subgraphs to a vertex x_i, $\bar{x}_i$, y_i, or $\bar{y}_i$ results in an immediate win on the next move for the other player.

Consider Fig. 4(a). The marker currently shown on vertex x_i is termed the *i*th *value marker* (*of player* I) and is free to move between x_i and $\bar{x}_i$. The position of this marker is associated with a truth value of the variable x_i as follows: if the marker is on x_i ($\bar{x}_i$), then x_i has value 0 (1). The markers on d_{1i} and c_{1i} are termed *guard markers*. If I moves his *i*th value marker to a vertex other than x_i or $\bar{x}_i$, then II wins on the next move by moving

his guard marker to either a_{1i} or b_{1i}, one of which must be unoccupied. However, if II moves his guard to either $\bar{x}_i$, e_{1i}, or f_{1i}, then I wins immediately by moving his guard marker from c_{1i} to either e_{1i} or f_{1i}. The terminology and behavior associated with Fig. 4(b) is analogous. Therefore, at each move player I (II) must either move his ith value marker from x_i to $\bar{x}_i$ or vice versa (resp., from y_i to $\bar{y}_i$ or vice versa) for some i, or move some marker other than a value marker or a guard marker.

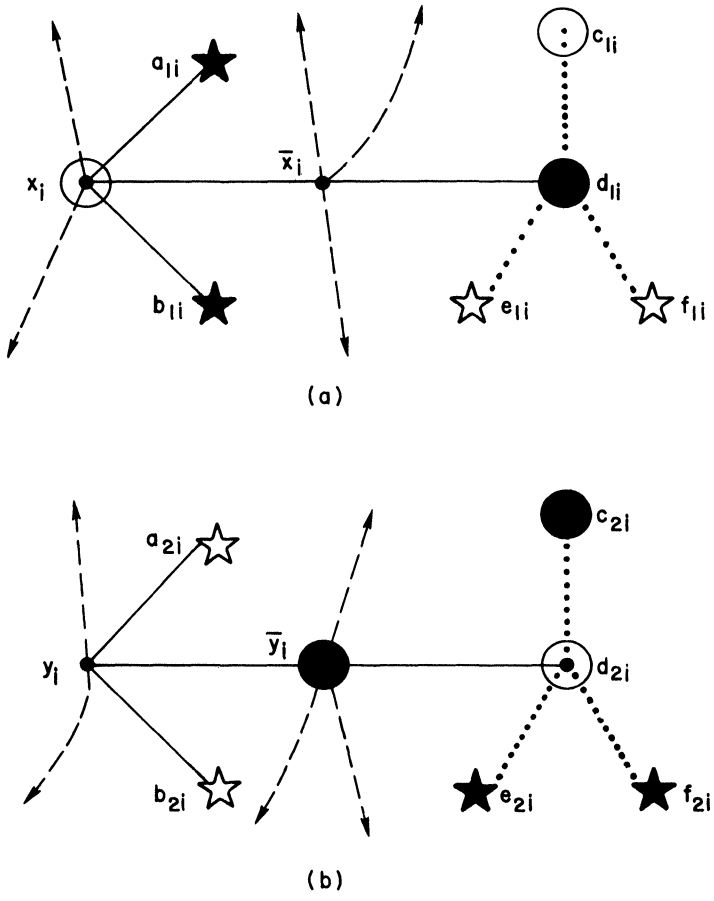

FIG. 4. *Subgraphs which represent the truth value of x_i and y_i.*

Write I-LOSE as $C_{11} \vee \cdots \vee C_{1k_1}$ and write II-LOSE as $C_{21} \vee \cdots \vee C_{2k_2}$ where each C_{ij} is a conjunction of literals. Another portion of the graph (V, E) is constructed for each C_{ij}. For example, say that C_{2j} is $(x_3 \wedge \bar{y}_5)$. Then the graph would contain the subgraph shown in Fig. 5. (With the exception of x_3 and $\bar{y}_5$, each vertex label in Fig. 5 actually has two additional subscripts, 2 and j, which have been repressed for readability.) This subgraph has two relevant properties. First, if it is player I's turn to move and both x_3 and $\bar{y}_5$ are unoccupied (corresponding to an assignment for which $x_3 \wedge \bar{y}_5$ is *true*) then I has a forced win. The strategy which achieves the win is termed the *end strategy*. To play the end strategy, I first moves the marker from w to s_1. Now II is forced to place a marker on r_1, and the marker shown currently on v is the only one that can reach r_1 in one move. Now I is forced to place a marker on u_1, and he does this by moving the marker from s_1 to u_1. Now II is forced to move his marker from r_1 to t_1, I is

forced to move from u_1 to s_2, and so on. Finally, I is able to move a marker from u_2 to z, and I wins. Note that as soon as I moves the marker from w to s_1, the moves of both players are forced. This yields the second relevant property of this subgraph: if I moves the marker from w to s_1 at a time when either x_3 or $\bar{y}_5$ is occupied, then II has a forced win. For example, suppose that x_3 is occupied and I moves from w to s_1. Then II moves from v to r_1. The only marker of player I which can reach u_1 in one move is the one currently on x_3. However, if I moves this marker from x_3 to u_1, then II moves his guard marker from d_{13} to a_{13} and wins (see Fig. 4(a) where $i = 3$).

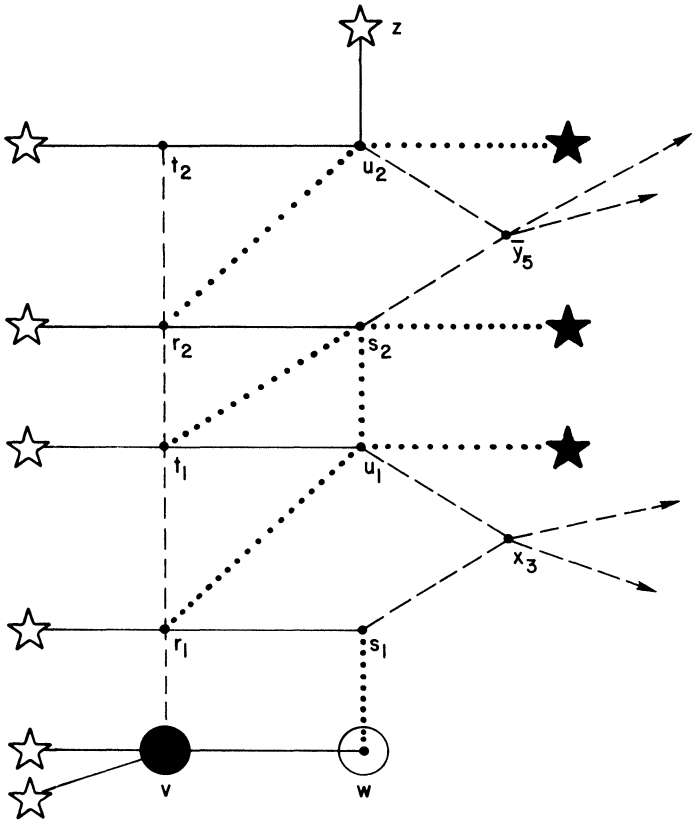

FIG. 5. *The subgraph which corresponds to the clause* $(x_3 \wedge \bar{y}_5)$ *in* II-LOSE.

It should be obvious how to generalize the graph of Fig. 5 for conjunctions with more than two literals. For conjunctions C_{1j} in I-LOSE, the construction is similar except that the markers on vertices v and w are interchanged and the vertices in W_1 (W_2) become vertices in W_2 (W_1). The position π' consists of the subgraphs of Fig. 4(a) and 4(b) for each variable together with the subgraph of Fig. 5 for each conjunction in I-LOSE and II-LOSE. The initial placement of value markers is specified by the assignment α. Observe that there is a constant b such that

$$\text{card}(V)+\text{card}(E) \leqq b \cdot (\text{size(I-LOSE)}+\text{size(II-LOSE)}). \tag{4.2}$$

It is not difficult to see that $\pi \in W(G_3)$ iff $\pi' \in W(\text{BLOCK})$. Suppose that $\pi \in W(G_3)$. Player I plays a side game of G_3 starting on π to determine his movement of value markers as long as II moves only value markers. At some point, player II must

move the side game to a position (1, I-LOSE, II-LOSE, β) where II-LOSE = 1 under β, so $C_{2j} = 1$ for some j. Now I can successfully play the end strategy on the subgraph corresponding to C_{2j}. The only other possibility is that II moves a marker from w_{1j} to s_{11j} (cf. Fig. 5), for some j, before II-LOSE = 1 in the side game. But since I has been playing a winning strategy in G_3, some literal in C_{1j} is 0 (i.e., the vertex corresponding to that literal is covered by a marker). Now II's moves are forced, and I wins as discussed above. The argument that $\pi \notin W(G_3)$ implies $\pi' \notin W$(BLOCK) is symmetric. This completes the proof of (4.1) and, therefore, that of Theorem 4.1. □

Moreover, from the proofs of Lemma 3.2 and (4.1), the inequality (4.2), and the method of encoding positions of BLOCK, it can be seen that the following is true.

COROLLARY 4.1. 1) $\mathscr{E}$-TIME $\leqq_{\log} EW$(BLOCK) *via length order* $n \log n$.

2) *There is a constant* $c > 0$, *such that if a* DTM *accepts* EW(BLOCK) *within time* $T(n)$, *then* $T(n) > c^{n/\log n}$ *for infinitely many* n.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] M. BAUER, D. BRAND, M. FISCHER, A. MEYER AND M. PATERSON, *A note on disjunctive form tautologies*, SIGACT News, 5 (April 1973), pp. 17–20.

[3] C. L. BOUTON, *Nim, a game with a complete mathematical theory*, Ann. Math. Princeton, 3 (1902), pp. 35–39.

[4] A. K. CHANDRA AND L. J. STOCKMEYER, *Alternation*, Proc. 17th IEEE Symp. on Foundations of Computer Science, 1976, IEEE, New York, 1976, pp. 98–108.

[5] J. H. CONWAY, *On Numbers and Games*, Academic Press, New York, 1976.

[6] S. A. COOK, *The complexity of theorem proving procedures*, Proc. Third ACM Symp. on Theory of Computing, 1971, Assoc. for Comput. Mach., New York, 1971, pp. 151–158.

[7] S. A. COOK AND R. A. RECKHOW, *Time bounded random access machines*, J. Comput. System Sci., 7 (1973), pp. 354–375.

[8] S. EVEN AND R. E. TARJAN, *A combinatorial problem which is complete in polynomial space*, J. Assoc. Comput. Mach., 23 (1976), pp. 710–719.

[9] M. J. FISCHER AND R. E. LADNER, *Propositional modal logic of programs*, Proc. Ninth ACM Symp. on Theory of Computing, 1977, Assoc. for Comput. Mach., New York, 1977, pp. 286–294.

[10] A. S. FRAENKEL, M. R. GAREY, D. S. JOHNSON, T. SCHAEFER AND Y. YESHA, *The complexity of Checkers on an* $N \times N$ *board–preliminary report*, Proc. 19th IEEE Symp. on Foundations of Computer Science, 1978, IEEE, New York, pp. 55–64.

[11] R. K. GUY AND C. A. B. SMITH, *The G-values of various games*, Proc. Cambridge Philos. Soc., 52 (1956), pp. 514–526.

[12] J. HARTMANIS AND R. E. STEARNS, *On the computational complexity of algorithms*, Trans. Amer. Math. Soc., 117 (1965), pp. 285–306.

[13] J. E. HOPCROFT AND J. D. ULLMAN, *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, MA, 1969.

[14] N. D. JONES, *Space-bounded reducibility among combinatorial problems*, J. Comput. System Sci., 11 (1975), pp. 68–85.

[15] N. D. JONES AND W. T. LAASER, *Complete problems for deterministic polynomial time*, Theoretical Computer Science, 3 (1977), pp. 105–117.

[16] R. M. KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–104.

[17] D. KOZEN, *On parallelism in Turing machines*, Proc. 17th IEEE Symp. on Foundations of Computer Science, 1976, IEEE, New York, 1976, pp. 89–97.

[18] D. LICHTENSTEIN AND M. SIPSER, *Go is Pspace hard*, Proc. 19th IEEE Symp. on Foundations of Computer Science. 1978, IEEE, New York, pp. 48–54.

[19] T. J. SCHAEFER, *Complexity of decision problems based on finite two-person perfect-information games*, Proc. Eighth ACM Symp. on Theory of Computing, 1976, Assoc. for Comput. Mach., New York, 1976, pp. 41–49.

[20] ———, *On the complexity of some two-person perfect information games*, J. Comput. System Sci., 16 (1978), pp. 185–225.

[21] L. J. STOCKMEYER, *The polynomial-time hierarchy*, Theoretical Computer Science, 3 (1977), pp. 1–22.
[22] L. J. STOCKMEYER AND A. R. MEYER, *Word problems requiring exponential time: preliminary report*, Proc. Fifth ACM Symp. on Theory of Computing, 1973, Assoc. for Comput. Mach., New York, 1973, pp. 1–9.
[23] G. S. TSEITIN, *On the complexity of derivation in propositional calculus*, Studies in Constructive Mathematics and Mathematical Logic, Part II, A. O. Slisenko, ed., Steklov Math. Institute, Leningrad, 1968.
[24] J. VON NEUMANN AND O. MORGENSTERN, *Theory of Games and Economic Behavior*, 3rd ed., Princeton University Press, Princeton, NJ, 1953.