JOURNAL OF COMPUTER AND SYSTEM SCIENCES 16, 185–225 (1978)

On the Complexity of Some Two-Person Perfect-Information Games

THOMAS J. SCHAEFER*

Department of Mathematics, University of California, Berkeley, California 94720

Received February 14, 1977; revised October 26, 1977

We present a number of two-person games, based on simple combinatorial ideas, for which the problem of deciding whether the first player can win is complete in polynomial space. This provides strong evidence, although not absolute proof, that efficient general algorithms for deciding the winner of these games do not exist. The existence of a polynomial-time algorithm for deciding any one of these games would imply the unexpected result that polynomial-time algorithms exist for (a) all the rest of these games, (b) all NP-complete problems and (c) in general, any problem decidable by a polynomial tape bounded Turing machine.

1. INTRODUCTION

Games provide many examples of problems which, although theoretically solvable, appear to be beyond the limits of practical computation. For example, the difficulty of deciding whether a given position in the game of chess can be won by the player who is about to move is widely recognized. Although this problem is in principle solvable by an exact computation, no way is known of carrying out that computation on a real computer in less than an astronomical amount of time.

In this respect chess is typical of many two-person perfect-information games. Many examples can be given of games which are "unsolved," in the sense that there is no known method of exact analysis which is not based on the straightforward approach of "searching the game tree"—a process which typically requires an amount of time exponential in the depth of search, and which is therefore impractical except for problems of very small size.

In terms of computational complexity, one suspects that these games are intrinsically difficult and would like to obtain theoretical results which confirm this suspicion. Compared to the abundance of games which seem to be hard, the results in this direction have so far been rather meager. For interesting finite games like chess, no meaningful characterization of complexity is known. But by considering games of a much more general character—for example, games which can be played on an arbitrary graph—some interesting results can be obtained.

As an example, we describe the game of Generalized Hex, which Even and Tarjan have studied [4, 5]. This game may be played on any (finite) graph G having two distin-

* Author's present address: CALMA, 527 Lakeside Drive, Sunnyvale, Calif. 94086.

guished nodes n_1 and n_2. A move for the first (second) player consists of putting a white (black) marker on some node of G: the player is free to choose any unoccupied node except n_1 and n_2. After all nodes other than n_1 and n_2 are occupied, the first player wins if and only if there is an arc-connected chain of white-occupied nodes joining n_1 to n_2.

Even and Tarjan showed that the problem of deciding of an arbitrary given graph whether the first player can win this game is complete with respect to log-space reducibility in polynomial space ("log-complete in PSPACE"). This condition (which will be defined in Section 2) gives very strong grounds for believing that Generalized Hex is difficult, in the following sense: it is unlikely that there exists any algorithm for deciding the winner on an arbitrary graph, whose worst-case running time is bounded by a polynomial function of the size of the input graph. The significance of this condition is discussed further below.

Generalized Hex was the first example of a natural game to be shown log-complete in PSPACE. The proof relied on an earlier result of Stockmeyer and Meyer [11, 12], which states that a decision problem on quantified propositional formulas, called B_ω, is log-complete in PSPACE. Actually, B_ω itself can be regarded as the problem of deciding the winner of a certain game. The game in this case is somewhat less natural than Generalized Hex. We have named this game $G_\omega(\mathrm{CNF})$; it is described in the list of games below.

The purpose of this paper is to present a number of new games which are also log-complete in PSPACE. The list of these games appears below. Even and Tarjan [4] observed that "any game with a sufficiently rich structure" would probably be complete in PSPACE. Our results serve to confirm that observation and provide some indication of what constitutes "sufficiently rich structure."

Game Descriptions

For each of the following games, the problem of deciding of any given input whether the first player can win is log-complete in polynomial space. All games are between two players, I and II, with I making moves 1, 3, 5,... and II making moves 2, 4,... . There are no draws. Basic definitions involving formulas and graphs, and details of encoding, are given in the Appendix. The meaning of "via length $L(n)$" is explained under "Space Bounds" below.

Games on Graphs and Directed Graphs

1. NODE KAYLES. Input is a graph. A move consists of putting a marker on an unoccupied node which is not adjacent to any occupied node. The first player unable to move loses. (Theorem 3.2; via length $n^8 \log n$.)

Remark. By thinking of this game as being played on the complement of the graph, we can regard it as a clique-forming game, in which a move consists of choosing a node which is adjacent to every previously chosen node.

2. BIGRAPH NODE KAYLES. Input is a triple $(G, V_{\mathrm{I}}, V_{\mathrm{II}})$, where $G = (V, E)$ is a graph, V_{I} and V_{II} are disjoint sets whose union is V, and every arc of G joins a node

in V_{I} to a node in V_{II}. (Thus G is a bipartite graph.) The game is played like game 1, except that player I must choose nodes in V_{I} and II must choose nodes in V_{II}. (Theorem 3.4; via length $n^{10} \log n$.)

Remark. Conway [14] describes a game called SNORT, which is played on planar maps. The above result implies that when SNORT is generalized to arbitrary graphs, the problem of recognizing winning positions is complete in PSPACE. For details, see the remarks which follow Theorem 3.4.

3. GEOGRAPHY. Input is a pair (G, s), where G is a directed graph and s is a node of G. A marker is initially placed on node s. A move consists of moving the marker along a directed arc to an adjacent node. Each directed arc can be used once only. The first player unable to move loses. (Theorem 3.1; via length $n^2 \log n$.)

Games on Propositional Formulas

4. G_ω(CNF). Input is a pair (A, ξ), where A is a CNF formula and $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$ is a list of distinct variables, including all those occurring in A. Move i consists of assigning to ξ_i a value of 0 (false) or 1 (true). After n moves, I wins iff the assignment which has been produced makes A true. (Stockmeyer and Meyer; see Lemma 2.3; via length $n^2 \log n$.)

Note. For mnemonic convenience we usually write the input to this game as $(\exists \xi_1)$ $(\forall \xi_2)$ $(\exists \xi_3) \cdots (Q\xi_n)A$, where the quantifiers are alternately $\exists$ and $\forall$. An example of an input to this game is $(\exists x_1)$ $(\forall x_2)$ $(\exists x_3)$ $((x_1) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \wedge x_3))$. It is not hard to see that player I can win this example: he has a winning strategy which consists of setting x_1 to 1 on move 1, and setting x_3 to the same value as x_2 on move 3.

5. G_{pos}(POS CNF). Input is a positive CNF formula (that is, a CNF formula in which $\neg$ does not occur). A move consists of choosing some variable of A which has not yet been chosen. The game ends after all variables of A have been chosen. Player I wins iff A is true when all variables chosen by I are set to 1 and all variables chosen by II are set to 0. (In other words, I wins iff he succeeds in playing some variable in each conjunct.) For example, on input $x \wedge (y \vee z) \wedge (y \vee w)$ player II can win. The following variations of this game are also log-complete in PSPACE: (a) G_{pos}(POS CNF 11). This is the same game as just described, but with inputs restricted to CNF formulas having at most 11 variables in each conjunct. (b) G_{pos}(POS DNF 11). Same as (a), but the inputs are DNF formulas instead of CNF. (Player I wins iff he plays every variable of some disjunct.) (Theorem 3.6 and Corollary 3.7; via length $n^2 \log n$.)

6. $G_{\%\mathrm{free}}$(CNF). Input is a triple $(A, V_{\mathrm{I}}, V_{\mathrm{II}})$, where A is a CNF formula, and V_{I} and V_{II} are disjoint sets whose union is the set of variables of A, $| V_{\mathrm{I}} | = | V_{\mathrm{II}} |$. A move by player I (II) consists of choosing some variable in V_{I} (V_{II}) which has not yet been chosen and setting it to 0 or 1. After all variables have been played, I wins iff the assignment makes A true. (Theorem 3.8; via length $n^2 (\log n)^2$.)

7. G_{avoid}(POS DNF 2). Input is a positive DNF formula A with at most two variables in any disjunct. A move consists of choosing any variable of A that has not yet

been chosen. The loser is the first player after whose move A is true, when the variables which have been played so far are set to 1 and all other variables are set to 0. (Corollary 3.3; via length $n^8 \log n$.)

8. G_{seek}(POS DNF 3). Same as 7, except that "winner" replaces "loser" and the input formula may have as many as three variables per disjunct. (Corollary 3.3; via length $n^8 \log n$.)

9. G_{avoid}(POS CNF). Same as 7, except that the input is a positive CNF formula, and there is no bound on the number of variables per conjunct. (Theorem 3.10; via length $n^2 (\log n)^2$.)

10. G_{seek}(POS CNF). Same as 7, except "winner" replaces "loser" and the input is as in 9. (Corollary 3.13; via length $n^2 (\log n)^2$.)

11. $G_{\%\text{avoid}}$(POS DNF 2). Input is a triple $(A, V_{\text{I}}, V_{\text{II}})$, where A is a positive DNF formula with at most two variables in any disjunct, and V_{I} and V_{II} are disjoint sets of equal cardinality whose union is the set of variables of A. A move for player I (II) consists of choosing any variable in V_{I} (V_{II}) that has not yet been chosen. The loser is the first player after whose move A is true, when the variables which have been played so far are set to 1 and all other variables are set to 0. (Corollary 3.5; via length $n^{10} \log n$.)

12. $G_{\%\text{seek}}$(POS DNF 3). Same as 11, except that "winner" replaces "loser" and the input formula may have as many as three variables per disjunct. (Corollary 3.5; via length $n^{10} \log n$.)

13. $G_{\%\text{avoid}}$(POS CNF). Same as 11, except that the input formula is a positive CNF formula, and there is no bound on the number of variables per conjunct. (Corollary 3.12; via length $n^2 (\log n)^2$.)

14. $G_{\%\text{seek}}$(POS CNF). Same as 11, except that "winner" replaces "loser" and the input is as in 13. (Corollary 3.13; via length $n^2 (\log n)^2$.)

Games on Collections of Sets

15. SIFT. Input is two collections of finite sets, $\mathcal{A} = \{A_1, \ldots, A_m\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$ with $A_i \neq B_j$ for all i, j. A move consists of choosing some element of $A_1 \cup \cdots \cup A_m \cup B_1 \cup \cdots \cup B_n$ that has not yet been chosen. Call a set A_i or B_i "occupied" when some element of it has been chosen. Player I wins if all sets in $\mathcal{A}$ are occupied before all sets in $\mathcal{B}$ are occupied. Player II wins if all sets in $\mathcal{B}$ are occupied before all sets in $\mathcal{A}$ are occupied. Any player who simultaneously occupies the last unoccupied sets of both $\mathcal{A}$ and $\mathcal{B}$ loses. (Corollary 3.11; via length $n^2 (\log n)^2$.)

Remark. If the restriction $A_i \neq B_j$ is dropped, then this game with $\mathcal{A} = \mathcal{B}$ is equivalent to game 9, G_{avoid}(POS CNF).

Each of the above that is played on positive CNF or DNF formulas can be equivalently stated as a game on collections of sets. For example, the following games are equivalent to games 5 and 9, respectively:

5′. Input is a collection $\{A_1, ..., A_n\}$ of finite sets. A move consists of choosing some element of $A_1 \cup \cdots \cup A_n$ that has not yet been chosen. After all elements have been chosen, player I wins iff he has chosen at least one element of each input set.

9′. The input and definition of move are as in 5′. The game ends and the last player loses as soon as at least one element of each input set has been chosen.

Similar paraphrases can be given for games 7, 8, 10–14.

Significance of completeness in PSPACE. We here discuss informally the significance of completeness in PSPACE; formal definitions will be given in Section 2. The completeness in PSPACE of the above games rests ultimately on the fact that these games have enough complexity to simulate Turing machines. Each of these games provides, in effect, through its various inputs, a language which is rich enough to describe Turing machine computations. This language is succinct enough that any problem decidable by a polynomial-tape bounded Turing machine is reducible efficiently (i.e., by a polynomial-time transformation) to instances of the given game. (A polynomial-tape bounded Turing machine is one which, to decide any given input, uses only an amount of tape bounded by some polynomial function of the length of the input. The set of all problems decidable by such machines is called "polynomial space," or PSPACE.) As a consequence, if a polynomial-time algorithm were found to decide one of these games, then this would yield polynomial-time algorithms for all problems in PSPACE.

PSPACE is a very large class and it would be very surprising if every problem in PSPACE were polynomial-time decidable. Forone thing, PSPACE includes the class NP, which contains many well-known problems (for example, the problem of deciding whether a given formula in the propositional calculus is satisfiable) for which no polynomial-time algorithm is known [3, 9]. Also, each one of these games is itself in PSPACE. (This is essentially a consequence of the fact that for each of these games the maximum number of moves in a played game is bounded by some polynomial function of the input size; see Lemma 2.2.) Thus, if any of these games is polynomial-time decidable, all of them are. These facts make it seem very unlikely that any of these games is polynomial-time decidable. Thus, our results provide rather strong evidence that efficient general algorithms for deciding these games do not exist.

Space bounds. The significance of the phrase "via length $L(n)$" is explained in detail at the end of Section 2. Roughly, it means that any Turing machine M that runs in space $S(n)$ can be simulated by inputs to the given game of size at most $cL(S(n))$, for some constant c depending on M. This is of interest because it implies lower bounds on the space needed to decide the game: if $L(n) = n^k (\log n)^m$, then the game is not decidable (even nondeterministically) in space n^p for any $p < 1/k$. (See Lemma 2.4.)

Origin of the games. Since we were not aware of any previously studied games of sufficient generality to be likely candidates for completeness in PSPACE, we had to more or less make up our own. In a few cases, it was possible to generalize existing games. For example, GEOGRAPHY was suggested to us by R. M. Karp as a generalization of the well-known game in which players take turns naming place-names, subject to the condition that the first letter of each place-name must be the same as the last letter of the preceding place-name. Likewise, the game we call NODE KAYLES can be regarded as a generalization of the game of Kayles, and games similar to it, which are described in [1, 7]. Aside from such cases, we have tried to systematically explore various combinatorial ideas. Propositional formulas have proved to be a particularly convenient vehicle for this exploration.

2. Preliminaries

Terminology for Games

We begin by establishing some terminology for games. Our definitions here are somewhat informal, in keeping with our approach of relying on natural-language descriptions to talk about games. We call the games listed in Section 1 *general games*, because each of them may be played on any of a large class of inputs. When a particular input is provided, we have what we call a *specific game*. Thus, NODE KAYLES is a general game, but NODE KAYLES on G, where G is some particular graph, is a specific game.

Let H be a specific game. The definition of H includes the specification of some initial configuration, together with the definition of what constitutes a legal move and what conditions determine a won game. By a *played game* on H, or a *finished game* on H, we mean a legal sequence of moves which, starting from the initial configuration of H, results in a win for one of the players. Since there are no draws in the games we consider, any legal sequence of moves is an initial subsequence of some finished game. A *strategy* for player I (II) for H is a function which assigns to any legal sequence of moves of even (odd) length, which is not a finished game, some move which is a legal continuation of that sequence. A strategy is a *winning strategy* for a player if that player is the winner of every played game in which every one of that player's moves is the move assigned by the strategy to the sequence of preceding moves. We often say that a player "can win" H to mean that that player has a winning strategy for H.

Let G be a general game. We denote by $\mathrm{Inp}(G)$ the set of inputs to G. We define $L(G) = \{A \in \mathrm{Inp}(G)$: player I has a winning strategy for G on input $A\}$, $\bar{L}(G) = \mathrm{Inp}(G) - L(G)$. Since our games have no draws, $\bar{L}(G)$ is the set of inputs to G on which player II has a winning strategy. (This follows from standard game-theoretic arguments.)

We use some additional notation for the games played on propositional formulas. These general games have names of the form $G_{\#}(X)$, where X is a class of formulas and $\#$ is some subscripted descriptor. For such games we write $L_{\#}(X)$ instead of $L(G_{\#}(X))$, and $G_{\#}(A)$ denotes the specific game consisting of $G_{\#}(X)$ played on the input formula A. In games where one player tries to make the input formula true and the other tries to make it false, we refer to these players by the names T and F, respectively, instead of the usual names I and II. Thus, for example, if A is a CNF formula, $A \in L_{\%\mathrm{free}}(\mathrm{CNF})$ iff player T can win $G_{\%\mathrm{free}}(A)$.

Complexity Measures

Let Γ, Δ be finite alphabets. A function f: $\Gamma^* \rightarrow \Delta^*$ is *computable in space $S(n)$* (*time $T(n)$*) if there is a deterministic Turing machine having a two-way read-only input tape, a one-way write-only output tape, and a single work tape, which, on any input $w \in \Gamma^*$, outputs $f(w)$ and halts, having scanned at most $S(|w|)$ distinct squares of the work tape (having run for at most $T(|w|)$ steps). A function is *computable in log space* (*polynomial space*, *polynomial time*) if it is computable in space $c \log(n)$ (space cn^k, time $cn^k + d$) for some constants c, d, k. (In this context we adopt the convention that $\log(0) = 0$.) "Log-space computable" means the same as "computable in log space."

By a *recognition problem* we mean any set of (finite) strings over some finite alphabet. A recognition problem $A \subseteq \Gamma^*$ is *decidable* in some space or time bound iff the function $f: \Gamma^* \to \{0, 1\}$ defined by $f(w) = 1 \Leftrightarrow w \in A$ is computable in that space or time bound. *PSPACE*, *LOGSPACE*, and *P* denote, respectively, the sets of recognition problems that are decidable in polynomial space, log space, and polynomial time.

It is not hard to show that $P \subseteq$ PSPACE. There is considerable reason to believe that this inclusion is proper, but so far this has not been demonstrated, in spite of much effort. Among the reasons for thinking that $P \neq$ PSPACE is the fact that PSPACE includes the class *NP*, which contains many problems of great practical interest for which no polynomial-time algorithm is known [3, 6, 9, 11].

Let $A \subseteq \Gamma^*$, $B \subseteq \Delta^*$. Then A is *log-space reducible to B*, or $A \leqslant_{\text{lg}} B$, iff there is a log-space computable function $f: \Gamma^* \to \Delta^*$ such that for all $w \in \Gamma^*$, $w \in A \Leftrightarrow f(w) \in B$. If f is such a function, we say $A \leqslant_{\text{lg}} B$ *via* f. The relation $\leqslant_{\text{lg}}$ is transitive [8, 12].

The recognition problem A is $\leqslant_{\text{lg}}$*-complete*, or *log-complete*, in PSPACE iff $A \in$ PSPACE and, for all $B \in$ PSPACE, $B \leqslant_{\text{lg}} A$. Stockmeyer and Meyer [11, 12] and others have given examples of problems which are log-complete in PSPACE.

Log-space reducibility is a finer reducibility than the polynomial-time reducibility $\propto$ of Karp [9]; that is, $A \leqslant_{\text{lg}} B$ implies $A \propto B$ [11]. From this it follows that if $A \in P$ and $B \leqslant_{\text{lg}} A$, then $B \in P$. Hence, if A is log-complete in PSPACE and PSPACE contains any problem not decidable in polynomial time, then A is such a problem. Thus, log-completeness in PSPACE provides rather strong reason to believe that a problem is not decidable in polynomial time.

To show that A is log-complete in PSPACE, it suffices to show that $A \in$ PSPACE and that $B \leqslant_{\text{lg}} A$ for some B which is log-complete in PSPACE. (This is an immediate consequence of the transitivity of $\leqslant_{\text{lg}}$.) The proofs of log-completeness in this paper all use this fact.

Preliminary Lemmas

We now give some preliminary lemmas about the complexity of our games. The first lemma is very straightforward and we state it without proof.

LEMMA 2.1. *For each general game G listed in Section* 1, (*a*) $\text{Inp}(G) \in$ LOGSPACE, *and* (*b*) $L(G)$ *is log-complete in* PSPACE *if and only if* $\bar{L}(G)$ *is log-complete in* PSPACE. ▮

In our completeness proofs we typically show $L(G) \leqslant_{\text{lg}} L(G')$ by exhibiting some log-space computable function $f: \text{Inp}(G) \to \text{Inp}(G')$ such that $f^{-1}(L(G')) = L(G)$. This is imprecise in that the domain of f should be Σ^*, the set of all strings over the input alphabet. This imprecision is justified by part (a) of the above lemma: it is easy to see that f can be extended to the domain Σ^* by mapping elements of Σ^*-Inp(G) into some fixed string not in $L(G')$, and that the extended function is also log-space computable.

LEMMA 2.2. *For each general game G listed in Section* 1, $L(G) \in$ PSPACE; *in fact, $L(G)$ is decidable in space cn for some constant c.*

Proof. (sketch) Let G be one of the general games and let H be the specific game consisting of G played on input A. If α is any legal sequence of moves in H, let Succ(α) be the set of moves m such that αm (that is, α followed by m) is a legal sequence. Let Winner(α) denote the player (I or II) for whom α is a winning position. It follows from standard game-theoretic arguments that Winner(α) is computed by the recursive algorithm described as follows: If α is a finished game, let Winner(α) be the winner, as defined by the rules of the game. Otherwise, let $p(\alpha)$ be the player who is to move in position α, and let $\bar{p}(\alpha)$ be the other player. Set Winner$(\alpha) = p(\alpha)$ if there is some $m \in$ Succ(α) such that Winner$(\alpha m) = p(\alpha)$, and Winner$(\alpha) = \bar{p}(\alpha)$ otherwise.

It is easy to see that the total space required to compute Winner$(\varnothing)$, where $\varnothing$ is the empty sequence, using this algorithm is just whatever is needed to record any legal sequence of moves, plus whatever is required for deciding whether a sequence is a finished game and if so who wins it, and for enumerating Succ(α) for any legal sequence α. For any of our games, it is not hard to see that all these things can be done using an amount of space at most linear in the length of the input. Thus, $L(G)$ is decidable in space cn for some c. ▌

The basis of all our completeness results is a theorem of Stockmeyer and Meyer [11, 12]. Let B_ω be the set of all propositional formulas A whose variables are in $\{x_{i,j} : i, j \geqslant 1\}$ such that $(\exists X_1)\ (\forall X_2)\ (\exists X_3) \cdots A$. This quantified formula means: "There exists an assignment of truth-values (i.e., values in $\{0, 1\}$) to $x_{1,1}, x_{1,2}, \ldots$, such that for any assignment of truth-values to $x_{2,1}, x_{2,2}, \ldots$, there exists... (and so on until the variables of A are all assigned)..., A is true." Stockmeyer and Meyer showed that $B_\omega \cap$ CNF 3 is log-complete in PSPACE, where CNF 3 is the set of CNF formulas having at most three literals in any conjunct.

There is a well-known correspondence between quantified formulas and games. Consider a game played on formulas A with variables in $\{x_{i,j} : i, j \geqslant 1\}$ in which the ith move consists of assigning truth-values to the variables $x_{i,1}, x_{i,2}, \ldots$ and player I wins iff A is true after all variables are assigned. It can be seen that B_ω is precisely the set of formulas A on which player I can win this game.

The only essential difference between this game, restricted to CNF formulas, and the game we call G_ω(CNF) is that in the latter game exactly one variable is played on each move. Any given CNF formula with variables $x_{i,j}$ can be turned into an input for G_ω(CNF) by a simple transformation which essentially inserts a dummy variable between any two similarly quantified variables. This transformation can be done in log space, and so the following is immediate. (G_ω(CNF 3) is the restriction of G_ω(CNF) to input formulas with at most three literals in each conjunct.)

LEMMA 2.3. *L_ω(CNF 3) is log-complete in* PSPACE. ▌

When proving that $L_\omega(\text{CNF}) \leqslant_{\lg} L(G)$ for some G, we often make restrictive assumptions about the inputs to G_ω(CNF), and rely on the fact that L_ω(CNF) is complete even in this restricted form. For example, we might assume that the quantified variables are $x_1, x_2, ..., x_n$ in order and that n is even. In all such cases, the justification is that there is an obvious log-space computable transformation which maps any given $A \in \text{Inp}(G_\omega(\text{CNF}))$ into some A' of the restricted form such that $A \in L_\omega(\text{CNF})$ iff $A' \in L_\omega(\text{CNF})$.

As is customary in this kind of work, we rarely take the trouble to justify the statement that the reductions we give are computable in log space. Although our constructions are sometimes complicated, there is usually no difficulty in showing that they can be carried out by algorithms which scan the input and generate the output with the help of a fixed number of counters (which hold numbers that never exceed the length of the input) and no other working storage. In a few cases where the procedure might not be clear, further explanation is given.

Justification of Space Bounds

We have asserted in Section 1 certain lower bounds on the space required to decide each game. We here sketch the justification of these assertions, referring the reader to [11] for details. Let us say that $A \subseteq \Gamma^*$ is log-complete in PSPACE *via length $L(n)$* if A is log-complete in PSPACE and for any $B \subseteq \Delta^*$ such that B is decidable nondeterministically in space $S(n) = n$, $B \leqslant_{\lg} A$ via some function f such that for all $w \in \Delta^*$, $|f(w)| \leqslant cL(|w|)$, where c is a constant depending on B.

LEMMA 2.4. *Let A be log-complete in* PSPACE *via length $L(n)$, with $L(n) \leqslant n^k(\log n)^m$, $k \geqslant 1$, $m \geqslant 0$. Then for any nondeterministic Turing machine M that decides A, there is a $c > 0$ such that M uses space greater than $c(n/(\log n)^m)^{1/k}$ on infinitely many inputs. Hence A is not decidable, even nondeterministically, in space n^p for any $p < 1/k$.*

The proof of this lemma is almost identical to that of Corollary 6.6, part (2), of [11], so we omit it. To determine $L(n)$ for each of our games, we have relied on the following fact: L_ω(CNF 3) is log-complete in PSPACE via length $n^2 \log n$; moreover, this reduction is realized by functions f such that $f(w)$ is a CNF formula with at most $c\,|w|^2$ occurrences of variables, for any argument w. (In other words, the factor of $\log n$ arises only from the need to write out the names of the variables.) This fact can be verified from the proof of completeness of $B_\omega \cap$ CNF 3 in [11]. Since all our completeness proofs are based on reductions, direct or indirect, of L_ω(CNF) or L_ω(CNF 3) to the given problems, it was straightforward to determine a bound $L(n)$ for each problem: we simply examined the reduction of L_ω(CNF 3) to the given problem and let $L(n)$ be some easily estimated upper bound (ignoring constant factors) on the size of the inputs to which CNF formulas having cn^2 occurrences of variables are reduced. The bounds $L(n)$ which we found in this way are given in the list of games in Section 1. We have in most cases not taken the trouble to comment on these bounds in the proofs themselves.

3. Completeness Proofs

In this section we prove the completeness of all the games listed in Section 1. We briefly describe the common method of proof. For each game G, $L(G)$ is shown to be log-complete in PSPACE by showing that $L(G) \in \text{PSPACE}$ (cf. Lemma 2.2) and that $L(G_0) \leqslant_{\lg} L(G)$ for some game G_0 such that $L(G_0)$ has already been shown log-complete in PSPACE. The proof of the latter statement goes as follows. Let some $A \in \text{Inp}(G_0)$ be given (cf. remarks after Lemma 2.1). We define some $A' \in \text{Inp}(G)$ and then verify the following: (a) A' is log-space computable as a function of A, and (b) If player I (resp. II) has a winning strategy for G_0 on A, then player I (resp. II) has a winning strategy for G on A'. It should be clear from the last section that these steps suffice for a proof. The following proof provides a very simple illustration of this method.

Theorem 3.1. *L(GEOGRAPHY) is log-complete in* PSPACE.

Proof. By Lemma 2.2, $L(\text{GEOGRAPHY}) \in \text{PSPACE}$. Thus, by Lemma 2.3, it suffices to show $L_\omega(\text{CNF}) \leqslant_{\lg} L(\text{GEOGRAPHY})$. Let $A \in \text{Inp}(G_\omega(\text{CNF}))$ be given. Assume without loss of generality that $A = (\exists x_1)(\forall x_2) \cdots (\exists x_n)(A_1 \wedge A_2 \wedge \cdots \wedge A_m)$, where n is odd and each A_i is a disjunction of literals. Define the directed graph $G = (V, E)$ by

$$V = \{x_i, \bar{x}_i, u_i, v_i : 1 \leqslant i \leqslant n\} \cup \{u_{n+1}\} \cup \{y_k : 1 \leqslant k \leqslant m\},$$

$$\begin{aligned} E = {} & \{(u_i, x_i), (u_i, \bar{x}_i), (x_i, v_i), (\bar{x}_i, v_i), (v_i, u_{i+1}) : 1 \leqslant i \leqslant n\} \\ & \cup \{(u_{n+1}, y_k) : 1 \leqslant k \leqslant m\} \cup \{(y_k, x_i) : x_i \text{ occurs unnegated in } A_k\} \\ & \cup \{(y_k, \bar{x}_i) : x_i \text{ occurs negated in } A_k\}. \end{aligned}$$

This construction is illustrated by a simple example in Fig. 1.

When GEOGRAPHY is played on the directed graph G with the marker initially on u_1, it mimics the game $G_\omega(A)$ in a fairly straightforward way. On move 1, player I must move the marker to either x_1 or $\bar{x}_1$; this corresponds to player T's assigning a value of 1 or 0, respectively, to x_1 in $G_\omega(A)$. Then moves 2 and 3 in the GEOGRAPHY game are forced: there is no choice but for II to move to v_1 and then for I to move to u_2. On move 4, player II must choose to move to x_2 or $\bar{x}_2$; this corresponds to player F's setting x_2 to 1 or 0 in $G_\omega(A)$. The play continues in this way, with one of x_i, $\bar{x}_i$ being chosen on move $3i - 2$, for $i = 1, 2, \ldots, n$; the choice is made by I if i is odd and by II if i is even. Thus, in terms of the choices made by the two players, the first $3n$ moves of GEOGRAPHY on input (G, u_1) are essentially identical to the n moves of $G_\omega(A)$. We next show that the outcomes of the two games are also identically determined.

Suppose that $3n$ moves have been made in some partially played game of GEOGRAPHY on (G, u_1). When interpreted as moves in a game of $G_\omega(A)$ via the above correspondence, these moves yield some played game K on $G_\omega(A)$. We claim that if T (resp. F) is the winner of K, then I (resp. II) can win the GEOGRAPHY game. First suppose that F is the winner of K. Let A_k be some conjunct which has not been satisfied. Then on move $3n + 1$ of the GEOGRAPHY game, II can win by moving the marker from u_{n+1} to y_k.

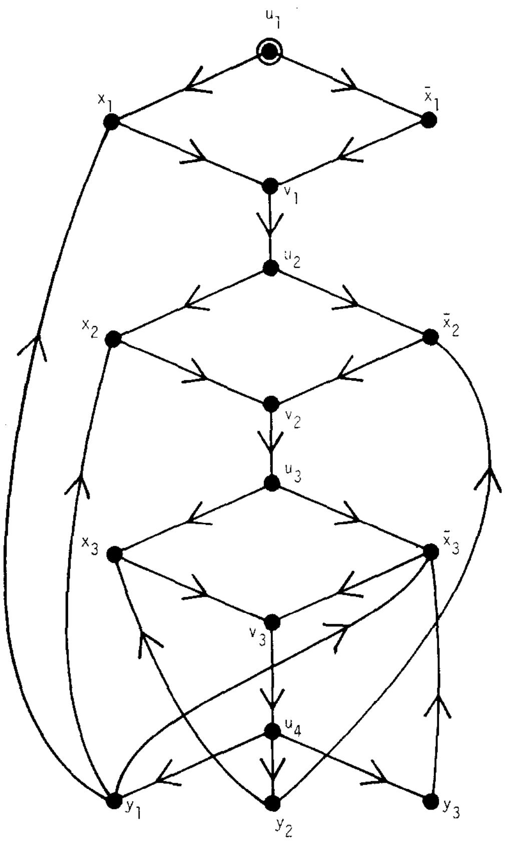

FIG. 1. Example of the construction of Theorem 3.1. This is the directed graph G which is constructed from the input formula $A = (\exists x_1)(\forall x_2)(\exists x_3)((x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_2) \wedge (\neg x_3))$. The marker is initially on node u_1.

Since A_k was not satisfied, every arc out of y_k goes to some x_i or $\bar{x}_i$ which has not been visited. After I moves to such an unvisited node, II will move the marker from that node to v_i. After that I has no move, since the only arc out of v_i has already been used, so II wins.

Next suppose that T is the winner of K. Then after II moves from u_{n+1} to some y_k on move $3n + 1$, I can win by moving to some visited node x_i or $\bar{x}_i$, where x_i is a variable of A_k which has been played so as to satisfy A_k. Since this node has already been visited, the only arc out of it has been used; thus II has no move and I wins.

In view of the above, it is easy to see that if player T (resp. F) has a winning strategy for $G_\omega(A)$, then I (resp. II) can use that strategy to win GEOGRAPHY on input (G, u_1).

To complete the proof, we observe that (G, u_1) is log-space computable as a function of A. ▌

It was relatively easy to use GEOGRAPHY to simulate G_ω(CNF), because the rules of GEOGRAPHY constrain the possible moves at any point to a small set. By contrast, the player in NODE KAYLES has his choice of any playable node on the graph. In order to use NODE KAYLES to simulate $G_\omega(\text{CNF})$, a way must be found to constrain the moves in spite of this apparent freedom. We do this by constructing the graph in such a way that at any point all but a few of the possible moves lead to sudden defeat for the player who makes them. The few moves that are not fatal are the "legitimate" moves. The idea of restricting freedom by punishing illegitimate moves was used by Even and Tarjan [4].

THEOREM 3.2 *L(NODE KAYLES) is log-complete in* PSPACE.

Proof. By Lemma 2.2, $L(\text{NODE KAYLES}) \in \text{PSPACE}$. Thus it suffices to show $L_\omega(\text{CNF}) \leqslant_{\lg} L(\text{NODE KAYLES})$. Let $A \in \text{Inp}(G_\omega(\text{CNF}))$ be given. Assume without loss of generality that $A = (\exists x_n)\,(\forall x_{n-1}) \cdots (\exists x_1)\,(B_1 \wedge \cdots \wedge B_m)$, where each B_i is a disjunction of literals, n is odd, and $B_1 = (x_1 \vee \neg x_1)$. This last assumption is permissible because such a conjunct can be added to any formula A without affecting the outcome of $G_\omega(A)$. Define the graph $G = (V, E)$ by

$$V = \bigcup_{0 \leqslant i \leqslant n} X_i,$$

$$X_0 = \{x_{0,k} \colon 1 \leqslant k \leqslant m\},$$

$$X_i = \{x_i, \bar{x}_i\} \cup \{y_{i,j} \colon 0 \leqslant j \leqslant i-1\} \qquad \text{for} \quad i = 1, 2, \ldots, n,$$

$$E = \bigcup_{0 \leqslant i \leqslant n} [X_i]^2 \cup D \cup \bigcup_{\substack{1 \leqslant i \leqslant n \\ 0 \leqslant j \leqslant i-1}} C_{i,j}, \qquad \text{where} \quad [X]^2 = \{\{x, y\} \colon x, y \in X, x \neq y\},$$

$$D = \{\{x_i, x_{0,k}\} \colon x_i \text{ occurs unnegated in } B_k\} \cup \{\{\bar{x}_i, x_{0,k}\} \colon x_i \text{ occurs negated in } B_k\},$$

$$C_{i,j} = \left\{\{y_{i,j}, w\} \colon w \in \bigcup_{\substack{0 \leqslant k < i \\ k \neq j}} X_k\right\}, \qquad i = 1, \ldots, n; \quad j = 0, \ldots, i-1.$$

A simple example of this construction is shown in Fig. 2. To avoid confusion resulting from the large numbers of arcs, this figure uses certain conventions to represent arcs. Groups of nodes are indicated by cigar-shaped enclosures. An arc which is shown terminating at the edge of an enclosure represents multiple arcs leading to each node within the enclosure. Also, all nodes within each enclosure are joined pairwise by arcs, which are not shown.

It is straightforward to show that there is a log-space algorithm which computes G from input A.

We say that a game of NODE KAYLES on G is played *legitimately* if for $i = 1, 2, \ldots, n$, the node played on move i is either x_{n-i+1} or $\bar{x}_{n-i+1}$. (The moves of a legitimate game mimic those of $G_\omega(A)$ in an obvious way: on move 1 player I plays x_n or $\bar{x}_n$, on move 2 player II plays x_{n-1} or $\bar{x}_{n-1}$, and so on.) The graph G has been constructed so as to force players to play legitimately: If on any of the first n moves a player fails to play legitimately, when all previous moves were legitimate, then that player is subject to immediate defeat. To show this, fix some integer i, $1 \leqslant i \leqslant n$, and suppose that the first

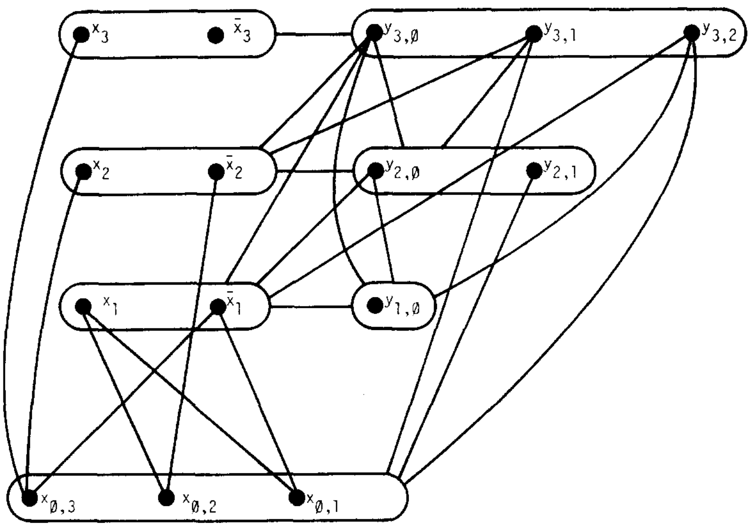

FIG. 2. Example of the construction of Theorem 3.2. This is the graph G which is constructed from the input formula $A = (\exists x_3)(\forall x_2)(\exists x_1)((x_1 \vee \neg x_1) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3))$. An arc leading to the edge of an enclosure represents multiple arcs leading to each node in the enclosure. Also, within each enclosure, all nodes are mutually joined by arcs, which are not shown.

$n - i$ moves have been made legitimately. Thus the nodes played so far are one of each pair $\{x_n, \bar{x}_n\},\ldots,\{x_{i+1}, \bar{x}_{i+1}\}$. Within each set X_k, any two nodes are adjacent; hence, at most one node can be played within each X_k. Thus all nodes of $X_n \cup X_{n-1} \cup \cdots \cup X_{i+1}$ are now unplayable. Suppose that the player on move $n - i + 1$ plays illegitimately, that is, he plays some node of $X_i \cup X_{i-1} \cup \cdots \cup X_0$ other than x_i or $\bar{x}_i$. If he plays a node in X_k for some $k < i$, then his opponent can win by playing $y_{i,k}$; this leaves no playable nodes, since every node of X_k is adjacent to the illegitimately played node, and every node of $(X_0 \cup X_1 \cup \cdots \cup X_i) - X_k$ is adjacent to $y_{i,k}$. On the other hand, if the illegitimate move is in X_i, it must be $y_{i,k}$ for some $k < i$. In this case, the opponent can win by playing x_k if $k > 0$ or $x_{0,1}$ if $k = 0$. (The latter is playable because of the assumption about B_1.) Again, the response wins because no playable nodes are left. Thus, any illegitimate move by the player on move i leads to immediate defeat.

We complete the proof by showing that player T can win $G_\omega(A)$ iff player I can win NODE KAYLES on G.

Suppose that T has a winning strategy for $G_\omega(A)$. Then player I can win NODE KAYLES on G as follows. On the first n moves of the game, I plays legitimately as long as II does, and uses T's winning strategy for $G_\omega(A)$, via the obvious correspondence whereby setting x_i to 0 or 1 in $G_\omega(A)$ corresponds to playing $\bar{x}_i$ or x_i, respectively, in NODE KAYLES on G. If II plays illegitimately on any of these moves, then of course I wins at once as described above. If II plays legitimately all the way, then after move n there are no playable nodes left: No node of $X_1 \cup \cdots \cup X_n$ is playable, since one node of X_i has been played for each $i > 0$; and since I has played by T's winning strategy, it is clear that every node $x_{0,k}$ of X_0 is adjacent to some played node (or else T's strategy would not have satisfied the conjunct A_k). Thus player I can win NODE KAYLES on G.

In a similar manner, one shows that if F has a winning strategy for $G_\omega(A)$, then II can win NODE KAYLES on G. As long as I plays legitimately, II does so too and uses F's winning strategy. If I plays legitimately all the way, then on move $n+1$ II can play $x_{0,k}$, where A_k is a conjunct of A which is not satisfied. After that there are no playable nodes and so II wins. ∎

COROLLARY 3.3. *L_{avoid}(POS DNF 2) and L_{seek}(POS DNF 3) are log-complete in* PSPACE.

Proof. To show L(NODE KAYLES) $\leqslant_{\text{lg}} L_{\text{avoid}}$(POS DNF 2), let a graph G be given and assume that its nodes are named $x_1, x_2, \ldots, x_n$. Let A be the disjunction of $\{(x_i \wedge x_j) : \{x_i, x_j\}$ is an arc of $G\} \cup \{(x_i \wedge x_i') : x_i$ is an isolated node of $G\}$. Then it is easy to see that I can win NODE KAYLES on G iff I can win $G_{\text{avoid}}(A)$.

Next we show L_{avoid}(POS DNF 2) $\leqslant_{\text{lg}} L_{\text{seek}}$(POS DNF 3). Let a positive DNF formula $A = A_1 \vee A_2 \vee \cdots \vee A_n$ be given. Let $A' = (u_1 \wedge u_2) \vee (u_1 \wedge A_1) \vee (u_1 \wedge A_2) \vee \cdots \vee (u_1 \wedge A_n)$, where u_1, u_2 are new variables not occurring in A. It is easy to see that I can win $G_{\text{seek}}(A')$ iff I can win $G_{\text{avoid}}(A)$. (Until some A_i is satisfied, it is fatal to play one of the new variables, because one's opponent will play the other and win. After some A_i is satisfied, playing u_1 wins.) If A has at most two variables in any disjunct, A' has at most three. ∎

NODE KAYLES seems to be very difficult even on very simple graphs. Even on graphs of degree 2, it is not obvious how to tell the winner; an elegant polynomial-time algorithm for this case is provided by [7]. We do not know of a polynomial-time algorithm for any of the following classes of graphs: (a) acyclic graphs of degree 3; (b) connected acyclic graphs having only one node of degree greater than 2; (c) graphs consisting of an m by n rectangular array of nodes, each joined by arcs to its nearest neighbors horizontally and vertically (even if $m = 3$ this is open).

The complexity of ARC KAYLES, the analog of NODE KAYLES in which players choose arcs instead of nodes, is not known. If ARC KAYLES is varied by letting the input graph have a distinguished subset A' of arcs, with the rule that no arc outside A' may be played as long as any arc in A' is playable, the result is TWO-PHASE ARC KAYLES. We have shown that L(TWO-PHASE ARC KAYLES) is log-complete in PSPACE; the proof, which is not given here, consists in reducing L_{pos}(POS CNF) to this problem.

THEOREM 3.4. *L(BIGRAPH NODE KAYLES) is log-complete in* PSPACE.

Proof. This proof uses basically the same construction that was used for NODE KAYLES, with a number of modifications. These modifications arise from the fact that the graph constructed must be bipartite; this limits our freedom to join pairs of nodes by arcs.

By Lemma 2.2, L(BIGRAPH NODE KAYLES) $\in$ PSPACE. Thus it suffices to show L_ω(CNF) $\leqslant_{\text{lg}} L$(BIGRAPH NODE KAYLES). Let $A \in \text{Inp}(G_\omega(\text{CNF}))$ be given.

Assume without loss of generality that $A = (\exists x_n)(\forall x_{n-1}) \cdots (\exists x_1)(B_1 \wedge \cdots \wedge B_m)$, where each B_i is a disjunction of literals and n is odd. Let the graph $G = (V, E)$ be defined by

$$\begin{aligned}
V &= \bigcup_{0 \leqslant i \leqslant 3n} (X_i \cup Y_i \cup Z_i),\\
X_0 &= \{x_{0,k} \colon 1 \leqslant k \leqslant m\},\\
Y_0 &= Z_0 = \varnothing,\\
X_{3i} &= \{x_i, \bar{x}_i\}, && 1 \leqslant i \leqslant n,\\
X_{3i-1} &= \{x_i', \bar{x}_i'\}, && 1 \leqslant i \leqslant n,\\
X_{3i-2} &= \{u_i\}, && 1 \leqslant i \leqslant n,\\
Y_i &= \{y_{i,k} \colon 0 \leqslant k \leqslant i-1\}, && 1 \leqslant i \leqslant 3n,\\
Z_i &= \{z_{i,k} \colon 1 \leqslant k \leqslant m + i^2\}, && 1 \leqslant i \leqslant 3n,\\
V_{\mathrm{I}} &= \bigcup_{\substack{1 \leqslant i \leqslant 3n\\ i\,\text{odd}}} X_i \cup \bigcup_{\substack{0 \leqslant i \leqslant 3n-1\\ i\,\text{even}}} (Y_i \cup Z_i),\\
V_{\mathrm{II}} &= V - V_{\mathrm{I}},\\
E &= E_1 \cup E_2 \cup E_3 \cup E_4,\\
E_1 &= \{\{\xi, \eta\} \colon \xi \in X_i, \eta \in Y_i \cup Z_i;\ 1 \leqslant i \leqslant 3n\},\\
E_2 &= \Bigg\{\{y_{i,k}, \xi\} \colon \xi \in V_{\mathrm{opp}} \cap \Bigg(\bigcup_{\substack{0 \leqslant j < i\\ j \neq k}} (X_j \cup Y_j \cup Z_j)\Bigg), \text{ where } V_{\mathrm{opp}} = V_{\mathrm{I}}\\
&\qquad \text{if } i \text{ is odd and } V_{\mathrm{opp}} = V_{\mathrm{II}} \text{ if } i \text{ is even; } 1 \leqslant i \leqslant 3n, 0 \leqslant k \leqslant i-1\Bigg\},\\
E_3 &= \{\{x_i, \bar{x}_i'\}, \{\bar{x}_i, x_i'\} \colon 1 \leqslant i \leqslant n\},\\
E_4 &= \{\{x_i, x_{0,k}\} \colon i \text{ is odd and } x_i \text{ occurs unnegated in } B_k\}\\
&\quad \cup \{\{x_i', x_{0,k}\} \colon i \text{ is even and } x_i \text{ occurs unnegated in } B_k\}\\
&\quad \cup \{\{\bar{x}_i, x_{0,k}\} \colon i \text{ is odd and } x_i \text{ occurs negated in } B_k\}\\
&\quad \cup \{\{\bar{x}_i', x_{0,k}\} \colon i \text{ is even and } x_i \text{ occurs negated in } B_k\}.
\end{aligned}$$

Observe that all arcs of G join a node of V_{I} to a node of V_{II}; hence the graph is bipartite as required. It is straightforward to show that $(G, V_{\mathrm{I}}, V_{\mathrm{II}})$ is log-space computable as a function of A.

We will say that a game of BIGRAPH NODE KAYLES on G is played *legitimately* if for $i = 1, 2, \ldots, 3n$, the node played on move i is contained in X_{3n-i+1}. Legitimacy has the same significance in this proof as in the proof for NODE KAYLES: if a game is played legitimately, then it mimics the game $G_\omega(A)$; and if a player moves illegitimately, he is subject to sudden defeat.

To see how a legitimately played game mimics $G_\omega(A)$, observe that each group of three moves in a legitimate game on G corresponds to a single move in $G_\omega(A)$. On the first move of each triple (that is, on moves 1, 4, etc.), the player to move plays x_i or $\bar{x}_i$ for some i; this corresponds to setting x_i to 1 or 0, respectively, in $G_\omega(A)$. On the second move of the triple (that is, moves 2, 5, etc.), there is no choice: because of the arcs E_3, the player must play x_i' if x_i was played on the preceding move, and $\bar{x}_i'$ if $\bar{x}_i$ was played. Thus the second move of the triple echoes the first move. The third move of the triple is also forced: by the definition of legitimacy, u_i must be played. Thus, the only real choice is on the first move of each triple, and there is a clear correspondence between the first $3n$ moves of a legitimately played game on G and the n moves of $G_\omega(A)$.

The echoing in the second move of the triple serves two purposes. First, it renders unplayable (via the arcs E_3) the node of $\{x_i, \bar{x}_i\}$ that was not played on the first move of the triple. Second, it solves a problem that arises in connecting the nodes $x_{0,k}$ representing the conjuncts of A to the nodes x_i, $\bar{x}_i$ representing the played variables. The problem is that we cannot, as in the proof for NODE KAYLES, always connect x_i or $\bar{x}_i$ to $x_{0,k}$ whenever x_i occurs in B_k, because sometimes x_i, $\bar{x}_i$ and $x_{0,k}$ all belong to the same player (that is, are all in $V_{\rm I}$ or all in $V_{\rm II}$), and then the bipartiteness condition forbids an arc between them. We get around this problem by forcing the opponent to echo the move with x_i' or $\bar{x}_i'$ and connecting the latter nodes to $x_{0,k}$. The third move of each triple simply serves to mark time, so that each new triple is started by a different player from the last.

We now show that if on any of the first $3n$ moves in a game of BIGRAPH NODE KAYLES on G, a player fails to play legitimately, when all previous moves were legitimate, then his opponent can win. To show this, fix some i, $1 \leqslant i \leqslant 3n$, and suppose that the first $3n - i$ moves were legitimate and that the player on move $3n - i + 1$ fails to play legitimately, that is, fails to play a node of X_i. (We call this player the "bad" player and his opponent the "good" player.) Because the previous moves were legitimate, it is easy to see that all nodes of $X_k \cup Y_k \cup Z_k$ for $k > i$ are unplayable, except possibly one node of X_{i+1}. The bad player cannot play any node of $X_{i+1} \cup Y_i \cup Z_i$, since these nodes belong to the good player. Thus the illegitimate move is in $X_k \cup Y_k \cup Z_k$ for some $k < i$. The good player can now win by playing $y_{i,k}$. To see that this wins, note that after this move the only playable nodes left for the bad player are contained in X_k or in $Y_k \cup Z_k$, depending on whether $i + k$ is even or odd. Thus the number of such nodes is at most $| Y_{i-1} \cup Z_{i-1} | = (i - 1) + m + (i - 1)^2 = m + i^2 - i$. (This equality holds for $i > 1$; if $i = 1$ it is not possible to make an illegitimate move.) But the good player will be able to play every node in Z_i, since these are all playable and none of them is adjacent to any of the bad player's playable nodes. Since $| Z_i | = m + i^2$, the first player unable to move will be the bad player, so the bad player will lose.

Now suppose that T has a winning strategy for $G_\omega(A)$. Player I can then win BIGRAPH NODE KAYLES on G as follows. As long as player II plays legitimately, player I does so also, and uses T's winning strategy in the obvious way, via the correspondence of moves sketched above. If II makes an illegitimate move, I wins as described in the preceding paragraph. If II plays all moves legitimately, then after $3n$ moves, II is to move and there are no playable nodes left: by legitimacy no node of $X_k \cup Y_k \cup Z_k$ is playable for any $k > 0$; and, because I used T's winning strategy, it is easy to see that no node of X_0 is playable. Thus I wins the game.

A similar argument shows that if F has a winning strategy for $G_\omega(A)$, then II can win BIGRAPH NODE KAYLES on G. If II follows F's winning strategy and I plays legitimately, then after $3n$ moves, all nodes outside X_0 are unplayable, and at least one node of X_0 is playable, corresponding to some conjunct of A which is not satisfied. Hence II will play a node of X_0 and I will be unable to move, and so II wins.

Thus, I can win BIGRAPH NODE KAYLES on G iff T can win $G_\omega(A)$. ▌

We estimate the size of the graph constructed in the above proof. Let $N = |A|$, $W_1 = \bigcup_i (X_i \cup Y_i)$, $W_2 = \bigcup_i Z_i$. It is easy to see that $|W_1| \leqslant c_1N^2$, $|W_2| \leqslant c_2N^3$ for constants c_1, c_2. Since no arc joins a node in W_2 to another node in W_2, the number of arcs in G is $\leqslant c_1{}^2N^4 + c_1c_2N^5 \leqslant c_3N^5$. This, together with the remarks after Lemma 2.4, justifies the claim made in Section 1 that L(BIGRAPH NODE KAYLES) is log-complete in PSPACE via length $n^{10} \log n$.

The game of SNORT ON GRAPHS is defined as follows. The input is a graph. A move consists of putting a marker on an unoccupied node. Player I uses white markers, and player II uses black markers. A marker may not be placed adjacent to a marker of the opposite color, but a player may play adjacent to his own markers. The first player unable to move loses. (SNORT ON GRAPHS is a generalization of the game of SNORT, which is described by Conway [14]. SNORT is more restricted in that it is played on planar maps, or equivalently on planar graphs, but aside from that it has the same rules.)

The problem of recognizing a winning position in SNORT ON GRAPHS is log-complete in PSPACE. This follows from the completeness of BIGRAPH NODE KAYLES, which is reduced to SNORT ON GRAPHS as follows. Given an input $(G, V_{\rm I}, V_{\rm II})$ to BIGRAPH NODE KAYLES, add two new nodes, n_1 and n_2, and make n_1 adjacent to every node in $V_{\rm I}$, and n_2 adjacent to every node in $V_{\rm II}$. Place a white marker on n_1 and a black marker on n_2. It is now not hard to see that this position is a winning position for player I in SNORT ON GRAPHS if and only if player I can win BIGRAPH NODE KAYLES on $(G, V_{\rm I}, V_{\rm II})$. This completes the reduction.

The problem we have just shown complete is the problem of recognizing a winning position in a game where some markers have already been played. This leaves open the question of the complexity of SNORT ON GRAPHS when the input is just a graph with no markers yet played.

COROLLARY 3.5. *$L_{\%\rm avoid}$(POS DNF 2) and $L_{\%\rm seek}$(POS DNF 3) are log-complete in* PSPACE.

Proof. To show L(BIGRAPH NODE KAYLES) $\leqslant_{\rm lg} L_{\%\rm avoid}$(POS DNF 2), let an input (G, V_1, V_2) to BIGRAPH NODE KAYLES be given. Assume $V_1 = \{x_1, \ldots, x_n\}$, $V_2 = \{y_1, \ldots, y_m\}$. Let A be the disjunction of $\{(x_i \wedge y_j) : \{x_i, y_j\}$ is an arc of $G\} \cup \{(\xi \wedge \xi') : \xi$ is an isolated node of $G\} \cup Z_1 \cup Z_2$, $V_i' = V_i \cup \{\xi' : \xi \in V_i$ is isolated$\} \cup Z_i$ for $i = 1, 2$. Here one of Z_1, Z_2 is empty and the other is $\{z_1, \ldots, z_p\}$ where p is the number required to make $|V_1'| = |V_2'|$. It is easy to see that I can win BIGRAPH NODE KAYLES on (G, V_1, V_2) iff I can win $G_{\%\rm avoid}(A, V_1', V_2')$.

Next we show $L_{\%\text{avoid}}(\text{POS DNF } 2) \leqslant_{\lg} L_{\%\text{seek}}(\text{POS DNF } 3)$. Let an input $(A, V_{\mathrm{I}}, V_{\mathrm{II}})$ to $G_{\%\text{avoid}}(\text{POS DNF } 2)$ be given, with $A = A_1 \vee \cdots \vee A_n$. Let A' be the disjunction of $\{(u_1 \wedge u_2)\} \cup \{(u_k \wedge A_i) : 1 \leqslant i \leqslant n,\ k = 1, 2\}$ where u_1 and u_2 are new variables not occurring in A. Then reasoning as in Corollary 3.3, it is easy to see that I can win $G_{\%\text{seek}}(A', V_{\mathrm{I}} \cup \{u_1\}, V_{\mathrm{II}} \cup \{u_2\})$ iff I can win $G_{\%\text{avoid}}(A, V_{\mathrm{I}}, V_{\mathrm{II}})$. ▌

THEOREM 3.6. *$L_{\text{pos}}(\text{POS CNF } 11)$ is log-complete in* PSPACE.

Proof. By Lemma 2.2, $L_{\text{pos}}(\text{POS CNF } 11) \in \text{PSPACE}$. We show that $L_\omega(\text{CNF } 3) \leqslant_{\lg} L_{\text{pos}}(\text{POS CNF } 11)$. Let $A \in \text{Inp}(G_\omega(\text{CNF } 3))$ be given. Assume without loss of generality that $A = (\exists x_{2n})(\forall x_{2n-1}) \cdots (\exists x_2)(\forall x_1)(F_1 \wedge \cdots \wedge F_m)$, where each F_i is a disjunction of at most three literals. We will define a positive CNF formula A' such that player T can win $G_{\text{pos}}(A')$ iff T can win $G_\omega(A)$. Let

$$A_1 = \bigwedge_{\substack{2 \leqslant j \leqslant 2n \\ j \text{ even}}} (C_{j-2} \wedge D_j \wedge E_j) \wedge \bigwedge_{1 \leqslant k \leqslant m} B_k,$$

where

$$\begin{aligned}
C_j &= x_{j+1} \vee \bar{x}_{j+1} \vee U(j+3, j+5) \vee X(j+2, j+5),\\
D_j &= x_j \vee \bar{x}_j \vee U(j+1, j+5) \vee X(j+1, j+5),\\
E_j &= u_j \vee U(j+1, j+5) \vee X(j, j+5),\\
B_k &= F_k' \vee U(1, 5) \vee X(1, 5).
\end{aligned}$$

Here $U(i, j)$ denotes the disjunction of $\{u_k : k \text{ is odd},\ i \leqslant k \leqslant j, \text{ and } k \leqslant 2n\}$, $X(i, j)$ denotes the disjunction of $\{(x_k \wedge \bar{x}_k) : i \leqslant k \leqslant j \text{ and } k \leqslant 2n\}$, and F_k' is the formula which results from replacing each negated variable $\neg x_i$ by the unnegated variable $\bar{x}_i$ throughout the formula F_k. For example, D_{2n} is simply the formula $x_{2n} \vee \bar{x}_{2n}$. Thus, A_1 is a positive formula with variables $\{x_i, \bar{x}_i, u_i : 1 \leqslant i \leqslant 2n\}$, but it is not a CNF formula because of the disjuncts $X(i, j)$.

Form A' from A_1 by expanding each conjunct of A_1 to conjunctive normal form. Since every conjunct of A_1 is of the form $H_1 \vee H_2$, where H_1 is a disjunction of variables and H_2 contains at most 12 distinct variables, this expansion increases the size of A_1 by a factor of at most 2^{12}. Slightly closer examination shows that A' has at most 11 variables in any conjunct. It is straightforward to show that A' is log-space computable as a function of A.

The semantic properties of the formula A' needed for this proof have been isolated in Lemmas 3.6.1, 3.6.2, and 3.6.3. In this proof, the only facts about A' which we use are that A' is a CNF formula whose variables are $\{x_i, \bar{x}_i, u_i : 1 \leqslant i \leqslant 2n\}$ and that these three lemmas hold.

The motivation of our definition of A' is explained following this proof.

Call a played game on $G_{\text{pos}}(A')$ *legitimate* iff for each $k = 0, 1, \ldots, n-1$:

(1) On move $6k + 1$, T plays one of x_{2n-2k}, $\bar{x}_{2n-2k}$.

(2) On move $6k + 2$, F plays the remaining variable of the pair $\{x_{2n-2k}, \bar{x}_{2n-2k}\}$.

(3) On move $6k + 3$, T plays u_{2n-2k}.

(4) On move $6k + 4$, F plays one of $x_{2n-2k-1}$, $\bar{x}_{2n-2k-1}$.

(5) On move $6k + 5$, T plays the remaining variable of the pair $\{x_{2n-2k-1}, \bar{x}_{2n-2k-1}\}$.

(6) On move $6k + 6$, F plays $u_{2n-2k-1}$.

A move which is not in accord with these rules is called *illegitimate.* A player *plays legitimately* if he does not make the first illegitimate move in a played game on $G_{\text{pos}}(A')$.

It is not hard to see how a legitimate played game of $G_{\text{pos}}(A')$ mimics the game $G_\omega(A)$. Every three moves in a legitimate game of $G_{\text{pos}}(A')$ correspond to a single move in $G_\omega(A)$. Player T's (F's) choice of x_{2n-2k} or $\bar{x}_{2n-2k}$ ($x_{2n-2k-1}$ or $\bar{x}_{2n-2k-1}$) on move $6k + 1$ ($6k + 4$) of the game on $G_{\text{pos}}(A')$ corresponds to player T's (F's) setting x_{2n-2k} ($x_{2n-2k-1}$) to 1 or 0, respectively, on move $2k + 1$ ($2k + 2$) in $G_\omega(A)$, for $k = 0,..., n - 1$.

We now show that T can win $G_\omega(A)$ iff T can win $G_{\text{pos}}(A')$.

($\Rightarrow$) Assume T has a winning strategy for $G_\omega(A)$. We first claim that T has a strategy for $G_{\text{pos}}(A')$ which wins any played game in which *F* plays legitimately. The strategy consists of playing legitimately and applying T's winning strategy for $G_\omega(A)$ according to the correspondence of moves described above. Then after $3n$ moves it is easy to check that the sufficient conditions for T to win stated in Lemma 3.6.1 hold, and so T wins.

It remains to show that T also can win if F does not play legitimately. Suppose F makes an illegitimate move at some point, when all previous moves were legitimate. We show that, whatever this move is, T has a response which leads to one of two results: (a) F has a lost position, or (b) F has on his next move the option of restoring the game to a "legitimate" position; if he accepts this option, the game continues as if it had been legitimate with no disadvantage to T; if he declines it he has a lost position.

We enumerate all possible illegitimate moves by F, assuming in each case that all previous moves were legitimate.

Case 1

Move $6k + 4$: Legitimate play requires F to play one of x_r, $\bar{x}_r$, where $r = 2n-2k-1$, but he does not do so.

Subcase 1*A*. F plays u_r. In this case T responds by playing x_r. If F next plays $\bar{x}_r$, the position is again legitimate—just as if the order of moves had been $\bar{x}_r$ by F on move $6k + 4$, then x_r by *T*, then u_r by F—and the game continues normally; the net effect is that F has let T make his choice for him with regard to the pair x_r, $\bar{x}_r$, and T is clearly none the worse off by it. On the other hand, if *F* plays anything other than $\bar{x}_r$ on move $6k + 6$, T plays $\bar{x}_r$, and then after F's next move the hypotheses of Lemma 3.6.2 will hold (with $j = r$ and hypothesis (3)(a) holding), and so by that lemma T can win.

Subcase 1B. F plays something other than u_r. In this case, T plays u_r. If F's next move is not x_r or $\bar{x}_r$, T wins after that move by Lemma 3.6.2 (with $j = r$ and hypothesis (3)(b) holding). If F's next move is x_r or $\bar{x}_r$, T plays the other of x_r, $\bar{x}_r$; then after F's next move T wins by Lemma 3.6.2 (with $j = r$ and hypothesis (3)(c) holding).

Case 2

Move $6k + 2$: Legitimate play requires F to play x_{2n-2k} or $\bar{x}_{2n-2k}$, whichever of the two was not played on move $6k + 1$. In this case, if F fails to play the required variable, T plays that variable. After F's next move, T wins by Lemma 3.6.2 (with $j = 2n - 2k$ and hypothesis (3)(a) holding).

Case 3

Move $6k + 6$: Legitimate play requires F to play $u_{2n-2k-1}$, but he does not do so. In this case, T plays that variable. Then after F's next move, T wins by Lemma 3.6.2 (with $j = 2n - 2k - 1$ and hypothsesis (3)(c) holding).

Thus, no matter what illegitimate move F makes, T can win. This completes the proof of ($\Rightarrow$).

($\Leftarrow$) Suppose that F has a winning strategy for $G_\omega(A)$. We show that F also can win $G_{\text{pos}}(A')$. We first claim that F has a strategy for $G_{\text{pos}}(A')$ which wins if T plays legitimately. The proof of this is similar to the proof of the corresponding claim for T in the last part; we leave the details to the reader.

We next show that F also can win if T does not play legitimately. We enumerate all possible illegitimate moves by T, assuming in each case that all previous moves were legitimate.

Case 1

Move $6k + 1$: Legitimate play requires T to play one of x_r, $\bar{x}_r$, where $r = 2n - 2k$, but he does not do so.

Subcase 1A. T plays u_r. In this case, F responds by playing x_r. If T next plays $\bar{x}_r$, the position is again legitimate—just as if the order of moves had been $\bar{x}_r$ by T on move $6k + 1$, then x_r by F, then u_r by T—and the game continues normally; F suffers no disadvantage by this transposition. On the other hand, if T fails to play $\bar{x}_r$ on move $6k + 3$, then F plays it and wins, by Lemma 3.6.3 with $j = r$.

Subcase 1B. T plays something other than u_r. In this case, F plays u_r and on his next move plays x_r or $\bar{x}_r$, thereby winning by Lemma 3.6.3.

Case 2

Move $6k + 5$: Legitimate play requires T to play whichever of $x_{2n-2k-1}$, $\bar{x}_{2n-2k-1}$ was not played on move $6k + 4$. In this case, if T fails to play the required variable, F plays it on his next move, and wins by Lemma 3.6.3.

Case 3

Move $6k + 3$: Legitimate play requires T to play u_{2n-2k}, but he does not. In this case, F plays that variable and wins by Lemma 3.6.3.

Thus, F can win in all cases. This completes the proof of ($\Leftarrow$) and of the theorem. ▌

We now briefly discuss the motivation for our definition of A' in the above proof. Suppose the conditions of the game $G_{\text{pos}}(A')$ were relaxed, so that A' was allowed to be any positive propositional formula, not necessarily a CNF formula. Then we could have taken A' to be the formula A_{2n} defined by $A_0 = A$ and

$$A_i = (x_i \vee \bar{x}_i) \wedge ((x_i \wedge \bar{x}_i) \vee u_i \vee A_{i-1}) \quad \text{for } i \text{ odd,}$$

$$A_i = (x_i \wedge \bar{x}_i) \vee ((x_i \vee \bar{x}_i) \wedge u_i \wedge A_{i-1}) \quad \text{for } i \text{ even} > 0.$$

It is rather easy to verify that the following three lemmas are true for A_{2n} as for A'. Thus the above proof can be used, with essentially no change, to show that T can win $G_\omega(A)$ iff T can win $G_{\text{pos}}(A_{2n})$.

The formula A' arose from the attempt to convert A_{2n} to conjunctive normal form. The full CNF expansion of A_{2n} would involve an exponential increase in size; but we were able, by trimming away unneeded conjuncts and disjuncts, to come up with the formula A'.

The formula A_{2n} was conceived as a kind of propositional analog of the graph constructed by Even and Tarjan in [4]. The above proof, applied to A_{2n}, is quite similar to the Even-Tarjan proof.

The main problem which arises in replacing A_{2n} by a CNF formula is finding a way for player T to punish F's illegitimate moves. With a CNF formula, it is much easier for F to threaten T with sudden defeat than for T to do so to F, since F can win by merely falsifying one conjunct. The solution of this problem is provided by Lemma 3.6.2.

The three lemmas which follow establish the semantic properties of the formula A' required by the above proof. In the statement of these lemmas, A' and n are as in that proof.

LEMMA 3.6.1. *Consider a played game on* $G_{\text{pos}}(A')$ *and let* V_T *be the set of variables which were played by* T *in this game. Assume, for* $i = 1, 2, \ldots, 2n$: (a) $|\{x_i, \bar{x}_i\} \cap V_T| = 1$, *and* (b) $u_i \in V_T$ *iff* i *is even. Then* T *has won the game if and only if for all* i, $1 \leqslant i \leqslant m$, F_i *is satisfied by the assignment* $s\colon \{x_1, \ldots, x_{2n}\} \to \{0, 1\}$ *defined by* $s(x_i) = 1$ *iff* $x_i \in V_T$.

Proof. The proof is straightforward from the definition of A'. Hypotheses (a) and (b) imply that all the formulas C_j, D_j, E_j are satisfied because their sets of disjuncts include $\{x_{j+1}, \bar{x}_{j+1}\}$, $\{x_j, \bar{x}_j\}$, $\{u_j\}$, respectively. Thus A' is satisfied iff all the formulas B_i are satisfied. For each i, $1 \leqslant i \leqslant m$, all disjuncts of B_i are false except F_i', which is satisfied iff F_i is satisfied by s. ▌

LEMMA 3.6.2. *Let* $1 \leqslant j \leqslant 2n$. *Consider a position in* $G_{\text{pos}}(A')$ *in which* T *is to move. Let* V_T (V_F) *be the set of variables which have been played so far by* T (F), *and assume*:

(1) *For* $i = j+1, j+2, \ldots, 2n$: (a) $x_i \in V_T$ *or* $\bar{x}_i \in V_T$, *and* (b) *if* i *is even,* $u_i \in V_T$.

(2) $|V_F \cap \{x_i, \bar{x}_i, u_i : i < j\}| \leqslant 2$.

(3) *One of the following holds*:

(a) $x_j \in V_T$ *and* $\bar{x}_j \in V_T$.

(b) j *is odd,* $u_j \in V_T$, *and* $\{x_j, \bar{x}_j\} \cap (V_T \cup V_F) = \varnothing$.

(c) j *is odd,* $u_j \in V_T$, *and* $\{x_j, \bar{x}_j\} \cap V_T \neq \varnothing$.

Then player T *can win from this position.*

Proof. Assume the hypotheses hold. Let r be the greatest even integer $\leqslant j$. Since A_1 is logically equivalent to A', it suffices to show how T can satisfy all conjuncts of A_1. Define the *rank* of a conjunct H of A_1 to be 0 if $H = B_k$ for some k and otherwise to be the even integer i such that H is C_i, D_i, or E_i. We omit the straightforward proof of the following claim:

CLAIM 1. All conjuncts of A_1 of rank $\geqslant r - 4$ are already satisfied, except, if hypothesis (3)(b) holds, the conjunct C_r.

Thus T, to win, must just play so as to satisfy all conjuncts of rank $0, 2, \ldots, r-6$, and also C_r in case (3)(b). The last case will present no problem: by (3)(b) both x_j and $\bar{x}_j$ are unplayed, and both these occur as disjuncts of C_r. T simply waits for F to play one of these two, and then plays the other. It will be seen that this does not interfere with T's strategy for the other conjuncts. In particular, if $r < 6$, there are no other conjuncts and so T wins easily. In what follows we therefore assume $r \geqslant 6$.

As for the conjuncts of rank $0, 2, \ldots, r-6$, we show that T can not only satisfy them, but can do so in a particular way. Call a disjunct of a conjunct of A_1 *principal* if it occurs other than as a part of one of the subformulas F_i' and is not of the form $(x_k \wedge \bar{x}_k)$. Since every such disjunct consists of a single variable, we also call the variable involved a *principal variable* of the conjunct. We show that T can succeed in playing a principal variable of every conjunct of rank $0, 2, \ldots, r-6$.

CLAIM 2. Disregard the hypotheses of the lemma. Let k be even, $0 \leqslant k \leqslant 2n-6$. Assume that F has played at most 2 of the principal variables occurring in conjuncts of A_1 of rank $\leqslant k$, and T is to move. Then T can succeed in playing some principal variable of each conjunct of rank $\leqslant k$.

We first verify that Claim 2 with $k = r - 6$ suffices to prove the lemma. All principal variables of conjuncts of rank $\leqslant k$ have subscripts at most $k + 5$. Thus hypothesis (2) of the lemma implies the hypothesis of Claim 2 that F has played at most two principal variables of these conjuncts. Furthermore, we see that the method described above for T to satisfy C_r in case (3)(b) will work, since the playing of x_j and $\bar{x}_j$ has no effect on the struggle for principal variables. Thus to complete the proof of the lemma, we must only prove Claim 2. It suffices to assume that F has made *exactly* 2 prior moves such as hypothesized in the claim.

Claim 2 is proved by induction on even integers k. First, let $k = 0$. Every conjunct of rank 0 has u_3 and u_5 as principal variables. If F's two prior moves are not u_3 and u_5, T will play one of them and achieve his goal. So assume that F's two moves are u_3 and u_5. Then T plays u_1. This is principal in all conjuncts of rank 0 except C_0. C_0 has two unplayed variables, x_1 and $\bar{x}_1$ as principal variables, and so T is assured of winning on his next move. Thus Claim 2 holds when $k = 0$.

Now assume Claim 2 holds when $k = h - 2$; we show it holds when $k = h$. Call a variable *new* if h is the least rank of any conjunct in which the variable is principal. Thus the new variables are u_{h+5}, u_h, x_{h+1}, $\bar{x}_{h+1}$, x_h, $\bar{x}_h$. We distinguish three cases:

(A) Neither of F's prior moves is a new variable. In this case, T follows his winning strategy for $k = h - 2$ until F plays a new variable. Whenever F plays a new variable, T plays the variable which is paired with it in the following list: (u_{h+5}, u_h), $(x_{h+1}, \bar{x}_{h+1})$, $(x_h, \bar{x}_h)$. Otherwise T plays his strategy for $k = h - 2$. (This is possible by the definition of new.) Clearly T in this way plays a principal variable in each conjunct of rank $\leqslant h - 2$, by the inductive hypothesis. Moreover, he also succeeds on conjuncts of rank h: By his pairing strategy he plays one of u_{h+5}, u_h, one of x_{h+1}, $\bar{x}_{h+1}$, and one of x_h, $\bar{x}_h$—which gives him a principal variable in E_h, C_h, and D_h, respectively. Thus in case (A), T achieves his goal.

(B) Exactly one of F's prior moves is a new variable. This is essentially the same as (A). T responds to the new variable played by F according to the above pairing and thereafter plays as in (A). As in (A) he achieves his goal.

(C) Both of F's prior moves are new variables. In this case T plays u_{h+3}, which is principal in every conjunct of rank h. It remains for him to play principal variables in all conjuncts of rank $\leqslant h - 2$, which he does easily using his strategy for $k = h - 2$, since F has not yet played any of those principal variables.

This completes the proof of Claim 2 and of the lemma. ▮

LEMMA 3.6.3. *Let* $1 \leqslant j \leqslant 2n$. *Consider a position in* $G_{\text{pos}}(A')$ *such that the following hold*: (1) *For* $i = j + 1, j + 2,..., 2n$: (a) *at least one of* x_i, $\bar{x}_i$ *has been played by* F, *and* (b) u_i *has been played by* F *if* i *is odd.* (2) *One of the following holds*: (a) x_j, $\bar{x}_j$ *have both been played by* F, *or* (b) j *is even and* F *has played* u_j *and one of* x_j, $\bar{x}_j$. *Then player* F *wins the game.*

Proof. Assume the hypotheses. We show there is a conjunct of A_1 all of whose disjuncts have been made false. Thus F will win regardless of subsequent play. If (2)(a) holds and j is odd, then every disjunct of C_{j-1} is false. If (2)(a) holds and j is even, then every disjunct of D_j is false. If (2)(b) holds, then every disjunct of E_j is false. Thus in every case F wins. ▮

COROLLARY 3.7. L_{pos}(POS DNF 11) *is log-complete in* PSPACE.

Proof. By Lemma 2.1, it suffices to show that $\bar{L}_{\text{pos}}$(POS DNF 11) is log-complete in PSPACE. Referring to the proof of the preceding theorem, we sketch a proof that L_ω(CNF 3) $\leqslant_{\lg} \bar{L}_{\text{pos}}$(POS DNF 11). Let some $A_0 \in \text{Inp}(G_\omega(\text{CNF } 3))$ be given. Assume that $A_0 = (\exists x_{2n-2})(\forall x_{2n-3}) \cdots (\exists x_2)(\forall x_1)B$. Let $A = (\exists x_{2n})(\forall x_{2n-1})A_0$. Clearly T can win

$G_\omega(A_0)$ iff T can win $G_\omega(A)$. Construct A' from A as in the preceding proof. As was seen in that proof, T will lose $G_{\text{pos}}(A')$ unless his first move is x_{2n}, $\bar{x}_{2n}$, or u_{2n}, and at least one of x_{2n}, $\bar{x}_{2n}$ is not a worse move than u_{2n}. Moreover, in this case, x_{2n} and $\bar{x}_{2n}$ occur symmetrically in A', so T has no better first move than x_{2n}; that is, T can win after this first move iff T can win $G_{\text{pos}}(A')$ in the first place. Form A_1 from A' by dropping every conjunct in which x_{2n} occurs. It is clear that T can win $G_{\text{neg}}(A_1)$ iff T can win $G_{\text{pos}}(A')$, where $G_{\text{neg}}(A_1)$ is the variation of $G_{\text{pos}}(A_1)$ in which T forfeits his first move and lets F move first. In the game $G_{\text{neg}}(A_1)$, T wins iff he plays some variable in every conjunct, that is, iff the first player (F) cannot play all variables of some conjunct; that is, iff the first player cannot win $G_{\text{pos}}(A_2)$, where A_2 is the positive DNF formula formed from A_1 by replacing $\wedge$ by $\vee$, and $\vee$ by $\wedge$, throughout. Thus T can win $G_\omega(A_0)$ iff F can win $G_{\text{pos}}(A_2)$. Thus, $L_\omega(\text{CNF } 3) \leqslant_{\lg} \bar{L}_{\text{pos}}(\text{POS DNF } 11)$ and so the latter is log-complete in PSPACE. ▮

We next consider the game $G_{\%\text{free}}(\text{CNF})$. In this game, the players are initially allocated distinct sets of variables, and each player plays only from his own set, assigning values of 0 or 1. If this partitioning of the variables were not done, and any player could play any variable, we would have a game which we call $G_{\text{free}}(\text{CNF})$. $G_{\text{free}}(\text{CNF})$ is complete in PSPACE even under the restriction that the input formula is positive; this follows at once from the completeness of $G_{\text{pos}}(\text{POS CNF})$, since when playing on a positive formula T would never want to set a variable to 0, nor F to 1.

Although similar in description, $G_{\text{free}}(\text{CNF})$ and $G_{\%\text{free}}(\text{CNF})$ differ vastly in terms of play. We can picture $G_{\text{free}}(\text{CNF})$ as an aggressive game, in which players race to claim territory. By contrast, $G_{\%\text{free}}(\text{CNF})$ is a passive game, in which one tries to make as little commitment as possible, to force one's opponent to reveal his intentions before one reveals one's own. If some variable occurs trivially in the input formula, so that the truth of the formula in no way depends on it, then it is always optimal to play that variable in $G_{\%\text{free}}(\text{CNF})$; in $G_{\text{free}}(\text{CNF})$ a player would not usually want to play such a variable.

THEOREM 3.8. *$L_{\%\text{free}}(\text{CNF})$ is log-complete in* PSPACE.

Proof. By Lemma 2.2, $L_{\%\text{free}}(\text{CNF}) \in \text{PSPACE}$. By $G_{\%\text{free}}(\text{PROP})$ we mean the game identical to $G_{\%\text{free}}(\text{CNF})$, but with arbitrary propositional formulas for input instead of CNF formulas. Lemma 3.9 states that $L_{\%\text{free}}(\text{PROP}) \leqslant_{\lg} L_{\%\text{free}}(\text{CNF})$. Thus it suffices here to show $L_\omega(\text{CNF}) \leqslant_{\lg} L_{\%\text{free}}(\text{PROP})$.

Let some $A \in \text{Inp}(G_\omega(\text{CNF}))$ be given. Assume without loss of generality that $A = (\exists x_1)(\forall x_2) \cdots (\forall x_n)A_0$, where n is even and A_0 is a CNF formula. We will define a formula A' with variables $V_1 \cup V_2$ such that (A', V_1, V_2) is log-space computable from A, and T can win $G_\omega(A)$ iff T can win $G_{\%\text{free}}(A', V_1, V_2)$. Let

$$A' = \bigvee_{\substack{1 \leqslant i \leqslant n-1 \\ i \text{ odd}}} \neg C_i \vee \left(\bigwedge_{\substack{2 \leqslant i \leqslant n \\ i \text{ even}}} C_i \wedge \left(\bigvee_{\substack{2 \leqslant i \leqslant n \\ i \text{ even}}} \neg B_i \vee \left(\bigwedge_{\substack{3 \leqslant i \leqslant n-1 \\ i \text{ odd}}} B_i \wedge A_0 \right)\right)\right),$$

where $B_i = (y_i \equiv x_{i-1})$ and $C_i = (z_i \equiv x_i) \vee (z_i \equiv y_i)$. Also let $V_1 = \{x_i, y_i, z_{i+1} : i$ is odd, $1 \leqslant i \leqslant n-1\}$ and $V_2 = \{x_i, y_i, z_{i-1} : i \text{ is even}, 2 \leqslant i \leqslant n\}$. Thus the variables of A' are $X \cup Y \cup Z$, where $X = \{x_i : 1 \leqslant i \leqslant n\}$, $Y = \{y_i : 1 \leqslant i \leqslant n\}$, and $Z = \{z_i : 1 \leqslant i \leqslant n\}$. We call members of X *X-variables*, and so on. It is straightforward to show that there is a log-space algorithm which computes (A', V_1, V_2) from input A. For brevity we write $G_{\%}(A')$ instead of $G_{\%\text{free}}(A', V_1, V_2)$.

We first describe informally the idea of the proof. The formulas C_i occur in A' in such a way as to force each z_i to be set to the same value as either x_i or y_i. Thus it is fatal for a player to play z_i before at least one of x_i, y_i has been played. But once x_i or y_i has been played, z_i can be played with complete safety. (Since z_i occurs only in C_i, the player of z_i, having made C_i true, will never have any reason to wish he had assigned a different value to z_i.) Because this move is completely safe, it is at least as good as any other move. (Recall the remarks preceding this theorem.) Thus, it can be assumed, without affecting the outcome of optimal play, that as soon as any player plays the first of $\{x_i, y_i\}$, his opponent will play z_i at once.

With this assumption, what does a player do after he plays x_i or y_i and his opponent responds with z_i ? There is no z_k that he can play safely, since by assumption he would have already played any such z_k. He could play a member of some unplayed pair $\{x_k, y_k\}$, but there is not much point in this move, since his opponent has z_k as a completely safe response. The only move that can possibly force his opponent to take any risk is to play the second member of some pair $\{x_k, y_k\}$ one of which has already been played. This being the case, a player might as well adopt the policy that he will never play the first member of a pair $\{x_k, y_k\}$ unless he intends to play the second member of the pair on his very next move. Thus, modulo this somewhat sketchy reasoning, we may assume, without affecting the outcome of optimal play, that each triple $\{x_k, y_k, z_k\}$ is played on three consecutive moves.

Given that the z_i's are always played so as to make C_i true, the formula A' can be paraphrased as follows: (i) If some y_i played by F fails to match x_{i-1}, then F loses. (ii) Otherwise, if some y_i played by T, $i > 1$, fails to match x_{i-1}, then T loses. (iii) Otherwise, T wins iff A_0 is true. The effect of clauses (i) and (ii) is that no player will play y_k with $k > 1$ unless he can correctly echo x_{k-1}, that is, unless x_{k-1} has been played. Thus, for $k > 1$, the triple $\{x_k, y_k, z_k\}$ will not be played until $\{x_{k-1}, y_{k-1}, z_{k-1}\}$ has been played. This forces the triples to be played in the strict order $\{x_1, y_1, z_1\}$, $\{x_2, y_2, z_2\}, \ldots, \{x_n, y_n, z_n\}$. The only essential choice left is the value assigned to $x_1, x_2, \ldots, x_n$. These values are assigned in order by players T and F alternately, and in the end player T wins iff these choices have been made in such a way that A_0 is true. Thus this game is equivalent to $G_\omega(A)$, and so whichever player can win $G_\omega(A)$ can also win $G_{\%}(A')$.

This completes our informal and admittedly imprecise description of how the formula A' works. We have gone to some length because the motivating ideas are not apparent from the rigorous proof which follows.

In a played game on $G_{\%}(A')$, we say that a player plays *unsoundly* if he plays some variable y_i with $i > 1$ before x_{i-1} has been played, or if he plays some z_i when both x_i and y_i are unplayed. The notation $s(\xi)$ means "the value assigned to the variable ξ." We now prove that T can win $G_\omega(A)$ iff T can win $G_{\%}(A')$.

($\Rightarrow$) Assume that T has a winning strategy for $G_\omega(A)$. Let T's strategy for $G_{\%}(A')$ be defined by the following rules:

T0. On his first move, T sets y_1 to an arbitrary value.

T1. Whenever F plays the first member of a pair $\{x_i, y_i\}$ (i.e., plays x_i when y_i is unplayed or y_i when x_i is unplayed), T responds by playing z_i and setting it to the same value as the variable played by F.

T2. Whenever F plays the second member of a pair $\{x_i, y_i\}$, T responds by playing y_k, where k is the least odd integer such that y_k is unplayed.

(a) If F has played z_k unsoundly, T sets $s(y_k) = 1 - s(z_k)$.

(b) Otherwise, if x_{k-1} has been played, T sets $s(y_k) = s(x_{k-1})$.

(c) Otherwise, T sets $s(y_k)$ arbitrarily.

T3. Whenever F plays a Z-variable z_i, T responds by playing x_i.

(a) If F has played z_i unsoundly, T sets $s(x_i) = 1 - s(z_i)$.

(b) Otherwise, if F has played y_{i+1} unsoundly, T sets $s(x_i) = 1 - s(y_{i+1})$.

(c) Otherwise, if all of $x_1, x_2, \ldots, x_{i-1}$ have been played, T sets $s(x_i) = \sigma(s(x_1), s(x_2), \ldots, s(x_{i-1}))$, where σ is T's winning strategy for $G_\omega(A)$.

(d) Otherwise, T sets $s(x_i)$ arbitrarily.

Assume that T uses this strategy in some played game on $G_{\%}(A')$. We will show that T wins that played game; thus, this is a winning strategy for T.

First, consider the case where F does not make any unsound move in the played game. In this case, we make the following claim: At any point in this played game where F is to move, the set of variables which have been played so far is the union of (a) for some odd k, the first k variables in the list $y_1, z_1, x_1, y_2, z_2, x_2, \ldots, y_n, z_n, x_n$, and (b) possibly some pairs of the form $\{x_i, z_i\}$ for various even i.

First note that the condition asserted by the claim holds when F is first to move, since at that time only y_1 (T's first move) has been played. Thus, to prove the claim it suffices to show that if this condition holds at any time when F is to move, it will also hold the next time F is to move. Assume that F is to move and the condition holds. Let ξ be the last variable in the list of (a) above such that all variables up to and including ξ have been played. *Case* 1. $\xi = y_i$ for some odd i. In this case, F may now play z_i or any of his unplayed X-variables, and these are the only variables that F can play soundly. If F plays z_i, then T plays x_i by rule T3. If F plays x_k for some even k, then y_k is unplayed and so T plays z_k by rule T1. In either case the condition of the claim still holds when F is next to move. *Case* 2. $\xi = x_i$ for some odd i. In this case, F may soundly play y_{i+1} or any of his unplayed X-variables. If he plays y_{i+1} and the pair $\{x_{i+1}, z_{i+1}\}$ is unplayed, then T plays z_{i+1} by rule T1, and the condition of the claim again holds. If F plays y_{i+1} and the pair $\{x_{i+1}, z_{i+1}\}$ has been played, then T plays y_{i+2} by rule T2, and again the condition of the claim holds. If F plays x_k for some even k, then y_k is unplayed and so T plays z_k by rule T1, and the condition again holds. *Case* 3. $\xi = z_i$

for some even i. In this case, F may only play an X-variable soundly. If F plays x_i, T plays y_{i+1} by rule T2, and the condition still holds. If F plays x_k for some even $k > i$, then T plays z_i and again the condition holds. This completes the proof of the claim.

Now in the played game under consideration, T played all his Y-variables except y_1 by rule T2(b)—rule T2(a) was not used because F played soundly, and by the above claim rule T2(c) was never used—and hence B_i is true for each odd i, $3 \leqslant i \leqslant n-1$. Also, T played all his Z-variables by rule T1, and so C_i is true for each even i, $2 \leqslant i \leqslant n$. And T played all his X-variables by rule T3(c)—the above claim tells us that rule T3(d) was never used; thus T played all his X-variables according to T's winning strategy for $G_\omega(A)$ and as a result A_0 is true. It is easily seen that these things suffice to make A' true. Thus T wins the played game in the case where F plays soundly.

Next suppose that F played some variable unsoundly in the played game. Since T played all his Z-variables by rule T1, C_i is true for each even i, $2 \leqslant i \leqslant n$. If F played z_k unsoundly for some odd k, T would have played x_k by rule T3(a) and y_k by rule T2(a), so that C_k is false, and this makes A' true. If F played all his Z-variables soundly, but played y_k unsoundly for some even k, then T would have played x_{k-1} by rule T3(b), making B_k false and A' true. Thus in all cases T wins the played game.

This completes the proof of ($\Rightarrow$).

($\Leftarrow$) Assume that F has a winning strategy for $G_\omega(A)$. Let F's strategy for $G_{\%}(A')$ be defined as follows:

F1. Whenever T plays the first member of a pair $\{x_i, y_i\}$, F responds by playing z_i and setting it to the same value as the variable just played by T.

F2. Whenever T plays the second member of a pair $\{x_i, y_i\}$, F responds by playing y_{i+1}.

(a) If T has played z_{i+1} unsoundly, F sets $s(y_{i+1}) = 1 - s(z_{i+1})$.

(b) Otherwise, F sets $s(y_{i+1}) = s(x_i)$.

F3. Whenever T plays a Z-variable z_i, F responds by playing x_i.

(a) If T has played z_i unsoundly, F sets $s(x_i) = 1 - s(z_i)$.

(b) Otherwise, if T has played y_{i+1} unsoundly, F sets $s(x_i) = 1 - s(y_{i+1})$.

(c) Otherwise, if all of $x_1, x_2, \ldots, x_{i-1}$ have been played, F sets $s(x_i) = \sigma(s(x_1), s(x_2), \ldots, s(x_{i-1}))$, where σ is F's winning strategy for $G_\omega(A)$.

(d) Otherwise, T sets $s(x_i)$ arbitrarily.

Assume that F uses this strategy in some played game on $G_{\%}(A')$. We will show that F wins this played game; thus, this is a winning strategy for F.

First consider the case where T does not make any unsound move. In this case, we make the following claim: At any point in the played game where T is to move, the set of variables which have been played so far is the union of (a) for some even $k \geqslant 0$, the first k variables in the list $y_1, z_1, x_1, y_2, z_2, x_2, \ldots, y_n, z_n, x_n$, and (b) possibly some pairs of the form $\{x_i, z_i\}$ for various odd i.

The proof of this claim is essentially identical to that of the corresponding claim in the last part, and so we omit it.

In the played game under consideration, this claim tells us that F played all his X-variables by rule F3(c), that is, according to F's winning strategy for $G_\omega(A)$, thereby making A_0 false. Also, F played all his Y-variables by rule F2(b), making B_i true for each even i, $2 \leqslant i \leqslant n$; and F played all his Z-variables by rule F1, making C_i true for each odd i, $1 \leqslant i \leqslant n-1$. From these things it can be seen that A' is false. Thus F wins the played game in the case where T plays soundly.

Now suppose that T played some variable unsoundly. Since F played all his Z-variables by rule F1, C_i is true for each odd i, $1 \leqslant i \leqslant n-1$. If T played z_k unsoundly for some even k, F would have played x_k by rule F3(a) and y_k by rule F2(a), making C_k false and hence A' false, and so F wins. Suppose then that T played all his Z-variables soundly but played y_k unsoundly for some odd k. Then F played x_{k-1} by rule F3(b), making B_k false. Moreover, F played all his Y-variables by rule F2(b), making B_i true for each even i, $2 \leqslant i \leqslant n$. These things make A' false, so in this case too F wins.

Thus in all cases F wins the played game. Thus the strategy given is a winning one. This completes the proof of $(\Leftarrow)$ and of the theorem. ▌

The game $G_{\%\text{free}}(\text{PROP})$ was defined in the above proof: It is the same as $G_{\%\text{free}}(\text{CNF})$, but with arbitrary propositional formulas for input instead of CNF formulas.

LEMMA 3.9. $L_{\%\text{free}}(\text{PROP}) \leqslant_{\lg} L_{\%\text{free}}(\text{CNF})$.

Proof. Let an input (A, V_T, V_F) to $G_{\%\text{free}}(\text{PROP})$ be given. We will define a CNF formula A' with variables $V_T' \cup V_F'$ such that (A', V_T', V_F') is log-space computable from (A, V_T, V_F) and such that T can win $G_{\%\text{free}}(A, V_T, V_F)$ iff T can win $G_{\%\text{free}}(A', V_T', V_F')$.

Let the logical connectives (i.e., the occurrences of $\wedge$, $\vee$, and $\neg$) in A be numbered from left to right with numbers 1, 2,..., n and assume $n = 2^p$ for some integer $p \geqslant 1$. (This entails no loss of generality, since one can increase the number of connectives in any formula to a power of 2 by repeatedly choosing some occurrence of a variable x and replacing it by $(x) \vee (x)$.)

With each connective i we associate a new variable $u_{0,i}$. In this way a distinct variable is associated with each subformula of A: With each nonatomic subformula (including A itself) we associate $u_{0,i}$, where i is the number of its main connective, and with each atomic subformula, consisting of a single variable, we associate the variable itself. For $i = 1,..., n$, let $ⓐ_i$ be the ith connective and define the formula D_i as follows. If $ⓐ_i$ is $\wedge$ or $\vee$, let D_i be $u_{0,i} \equiv (\xi \, ⓐ_i \, \eta)$, where ξ and η are the variables associated with the two subformulas joined by $ⓐ_i$. If $ⓐ_i$ is $\neg$, let D_i be $u_{0,i} \equiv (\neg\eta)$, where η is the variable associated with the subformula negated by $ⓐ_i$.

Let $u_{0,m}$ be the variable associated with the entire formula A. Define

$$A^{\#} = C \wedge \bigwedge_{\substack{0 \leqslant i \leqslant p \\ 1 \leqslant j \leqslant 2^{p-i}}} B_{i,j},$$

where

$$
\begin{aligned}
C &= H_0 \vee u_{0,m}\,, \\
B_{0,j} &= H_0 \vee D_j\,, \\
B_{i,j} &= H_i \vee E_{i,j} \qquad \text{if } \ i > 0, \\
E_{i,1} &= u_{i,1} \equiv ((\hat{u}_{i-1,1} \equiv u_{i-1,1}) \wedge (\hat{u}_{i-1,2} \equiv u_{i-1,2})), \\
E_{i,j} &= u_{i,j} \equiv (u_{i,j-1} \wedge (\hat{u}_{i-1,2j-1} \equiv u_{i-1,2j-1}) \wedge (\hat{u}_{i-1,2j} \equiv u_{i-1,2j})) \qquad \text{if } \ j > 1, \\
H_i &= \Bigg(\bigvee_{i+1 \leqslant k \leqslant p} \neg\, u_{k,2^{p-k}} \Bigg) \vee (\hat{u}_{p,1} \not\equiv u_{p,1}).
\end{aligned}
$$

Form A' from $A^{\#}$ by expanding each conjunct of $A^{\#}$ to conjunctive normal form. Since each conjunct of $A^{\#}$ is of the form $F_1 \vee F_2$, where F_1 is a disjunction of literals and F_2 involves at most 8 distinct variables, this expansion increases the size of $A^{\#}$ by a factor of at most $2^8 = 256$. Let $V_T' = V_T \cup U_T$ and $V_F' = V_F \cup U_F$, where $U_T = \{u_{i,j} : 0 \leqslant i \leqslant p,\ 1 \leqslant j \leqslant 2^{p-i}\}$ and $U_F = \{\hat{u}_{i,j} : u_{i,j} \in U_T\}$. We call members of $V_T \cup V_F$ *V-variables* and members of $U_T \cup U_F$ *U-variables.*

It is straightforward to describe a log-space algorithm which on input (A, V_T, V_F) outputs (A', V_T', V_F'). We note just the following points: (i) To compute C, one must find the main connective of A. This is done by scanning A from left to right and counting parentheses until equal numbers of left and right parentheses have been passed and a connective is scanned. This connective is then the main connective of A. (ii) To compute D_j, one must find the main connectives of the subformulas joined by $ⓐ_j$. This is done by first moving the input head to scan $ⓐ_j$, and then moving it leftward or rightward and counting parentheses in a manner similar to the above. Further details are left to the reader.

For brevity we write $G_{\%}(A)$ and $G_{\%}(A')$ instead of $G_{\%\text{free}}(A, V_T, V_F)$ and $G_{\%\text{free}}(A', V_T', V_F')$.

We now informally describe the idea of the proof. We intend that the game $G_{\%}(A')$ will be played in a number of distinct phases. In the first phase, all the V-variables are played, in the same way they would be played in a game of $G_{\%}(A)$. In phase $2 + i$, for $i = 0, 1, \ldots, p$, all the variables $u_{i,j}$ and $\hat{u}_{i,j}$ are played. In phase 2, T has the chance to "prove" that he has succeeded in satisfying A; he does this by setting each variable $u_{0,j}$ to the truth-value of the corresponding subformula of A, as determined by the values assigned to the V-variables in phase 1. In order to keep F busy while T is doing this, we require F to echo each of T's moves, by setting $\hat{u}_{0,j}$ to the same value as $u_{0,j}$. (Without this echoing requirement, F could prematurely play $\hat{u}_{0,j}$ in phase 1.) The purpose of phase i, for $i > 2$, is to check up F's echoing in phase $i - 1$. For each $i > 2$, T in phase i reports on F's echoing by setting $u_{i,j}$ to 1 iff F correctly echoed $u_{i-1,k}$ for $k = 1, 2, \ldots, 2j$. Thus, the $u_{i,j}$ with highest j is a complete report on F's echoing in phase $i - 1$; it is set to 1 iff F's echoing in phase $i - 1$ was completely correct. This variable appears negated (through the formulas H_i) in those conjuncts of A' where it is needed to punish F for echoing incorrectly. While T is doing this reporting, we again keep F busy by requiring him to echo each of T's moves, by setting $\hat{u}_{i,j}$ equal to $u_{i,j}$.

Thus each phase of checking up on F's echoing gives rise to the need for another phase of checking up. Fortunately, the number of variables played in each phase is half that of the preceding phase, so that after $p = \log(n)$ phases of checking up, it remains only to check that F has correctly echoed the single variable $u_{p,1}$. This is done by including a disjunct $\hat{u}_{p,1} \not\equiv u_{p,1}$ in every conjunct of $A^{\#}$.

Getting T to play in the way described above is quite simple: There is a conjunct of $A^{\#}$ which states what value $u_{i,j}$ shall have; if T does not set $u_{i,j}$ to this value, the conjunct $B_{i,j}$ is false (unless F echoes badly) and so T loses.

This informal sketch has overlooked many difficulties. For example, we cannot strictly enforce the division of the game into phases: T may quite legitimately start reporting on F's echoes in phase i before phase i is over. Also, it is not obvious from the above that the punishing of bad echoes by F can be carried out without creating loopholes which the punisher could use to win even with correct echoing. We now give the formal proof that T can win $G_{\%}(A)$ iff T can win $G_{\%}(A')$. As before, $s(x)$ means "the value assigned to the variable x."

($\Rightarrow$) Assume that T has a winning strategy for $G_{\%}(A)$. Let T's strategy for $G_{\%}(A')$ be defined as follows:

T1. On his first move, and whenever F plays a V-variable which is not his last remaining V-variable, T plays a V-variable according to his winning strategy for $G_{\%}(A)$. That is, T makes the same move he would have made in $G_{\%}(A)$, given the play which has taken place so far on the V-variables.

T2. Whenever F plays a variable $\hat{u}_{i,j}$ such that $u_{i,j}$ is unplayed, T responds by playing $u_{i,j}$ and assigning it the value $1 - s(\hat{u}_{i,j})$.

T3. Otherwise, T plays $u_{i,j}$, where (i, j) is the lexicographically least pair such that $u_{i,j}$ is unplayed. (Lexicographic ordering is defined by: $(i, j) < (h, k)$ iff $i < h$ or ($i = h$ and $j < k$).) If all variables in the set $V_T \cup V_F \cup \{u_{h,k}, \hat{u}_{h,k} : (h, k) < (i, j)\}$ have been played, T assigns $u_{i,j}$ a value by (a) and (b) below; otherwise he assigns an arbitrary value.

(a) If $i = 0$, T assigns to $u_{0,j}$ the truth value of the subformula of A associated with the variable $u_{0,j}$, as determined by the values assigned to the V-variables.

(b) If $i > 0$, T assigns a value so as to make $E_{i,j}$ true. That is, T sets $u_{i,j}$ to 1 iff $s(\hat{u}_{i-1,2j-1}) = s(u_{i-1,2j-1})$ and $s(\hat{u}_{i-1,2j}) = s(u_{i-1,2j})$ and, if $j > 1$, $s(u_{i,j-1}) = 1$.

This completes the definition of T's strategy for $G_{\%}(A')$.

Assume that T uses this strategy in some played game on $G_{\%}(A')$. We will show that T wins this played game; thus, this is a winning strategy for T.

We make the following claim: At any point in the game when F is to move, the set of variables which have been played so far is the union of (a) possibly some pairs of the form $\{u_{i,j}, \hat{u}_{i,j}\}$ for various i, j and (b) either a subset of $V_T \cup V_F$ or, for some pair (i, j), the set $\{u_{i,j}\} \cup \{u_{h,k}, \hat{u}_{h,k} : (h, k) < (i, j)\} \cup V_T \cup V_F$.

The proof of this claim is straightforward. One shows that it holds when F is first to move, and that if it holds at any time when F is to move, it also holds the next time F is to move. The reader can supply the details.

From this claim it can be seen that whenever T plays $u_{i,j}$ by rule T3, he assigns it a value by T3(a) or T3(b); he never assigns an arbitrary value.

First suppose that, in the played game under consideration, rule T2 was never used. Then, since all the variables $u_{0,j}$ were assigned by rule T3(a), the formulas D_j are all true, for $1 \leqslant j \leqslant n$. Moreover, since T played the V-variables by his winning strategy for $G_{\%}(A)$, the variable $u_{0,m}$ corresponding to the entire formula A has a value of 1; hence the formula C is true. Finally, each variable $u_{i,j}$ with $i > 0$ was played by rule T3(b), making the formula $E_{i,j}$ true. Thus all the conjuncts of $A^{\#}$ are true, so A' is true, and so T wins.

Next suppose that rule T2 was used. Let (h, k) be the lexicographically greatest pair such that $u_{h,k}$ was played by rule T2. If $h = p$, then $k = 1$, since there is only one variable of the form $u_{p,k}$, and so by rule T2, $s(u_{p,1}) \neq s(\hat{u}_{p,1})$, and so T wins, since $\hat{u}_{p,1} \not\equiv u_{p,1}$ is a disjunct of every conjunct of $A^{\#}$. If $h < p$, then T played all variables $u_{i,j}$ with $i > h$ by rule T3(b), thereby making each formula $E_{i,j}$ with $i > h$ true. Further, let $q = \lceil k/2 \rceil$, $r = 2^{p-h-1}$ (where $\lceil x \rceil$ denotes the least integer not less than x); then by rule T3(b) each of $u_{h+1,q}, u_{h+1,q+1}, \ldots, u_{h+1,r}$ was assigned the value 0, which makes all the formulas C and $B_{i,j}$ with $i \leqslant h$ true, since these all have $\neg u_{h+1,r}$ as a disjunct. Thus all conjuncts of A' are true and T wins.

Thus, in all cases T wins $G_{\%}(A')$. Thus the strategy given is a winning one. This completes the proof of ($\Rightarrow$).

($\Leftarrow$) Assume that F has a winning strategy for $G_{\%}(A)$. Let F's strategy for $G_{\%}(A')$ be defined as follows:

F1. Whenever T plays a V-variable, F responds by playing a V-variable according to his winning strategy for $G_{\%}(A)$.

F2. Whenever T plays a U-variable $u_{i,j}$, F responds by playing $\hat{u}_{i,j}$ and setting $s(\hat{u}_{i,j}) = s(u_{i,j})$.

We claim that this is a winning strategy for F. To see this, consider a played game on $G_{\%}(A')$ in which F uses this strategy.

First suppose that T played all the variables $\{u_{i,r} : 1 \leqslant i \leqslant p, r = 2^{p-i}\}$ to the value 1. This makes all the formulas H_i false. Now if any of the formulas D_j were false, A' would be false and F would win. So assume that all the formulas D_j are true, $1 \leqslant j \leqslant n$. This means that the variables $u_{0,j}$ all correctly reflect the truth values of the corresponding subformulas of A, as determined by the values of the V-variables. In particular, the variable $u_{0,m}$ corresponding to the whole formula A has value 0, since F played the V-variables by his winning strategy for $G_{\%}(A)$. Thus the formula C is false, and hence A' is false and F wins.

Next suppose that T set $s(u_{i,r}) = 0$ for some $i \geqslant 1$, $r = 2^{p-i}$. Let i be the greatest such i; thus H_i is false. If $s(u_{i,1}) = 0$, then $E_{i,1}$ is false in view of rule F2, and so $B_{i,1}$ is false; hence A' is false and F wins. Otherwise, let j be the greatest j such that $s(u_{i,j}) = 1$. Then $E_{i,j+1}$ is false in view of rule F2, and so $B_{i,j+1}$ is false, hence A' is false and F wins.

Thus, in all cases F wins $G_{\%}(A')$. Thus the strategy given is a winning one. This completes the proof of ($\Leftarrow$) and of the lemma. ▌

We have not been able to produce any fixed integer bound k such that $L_{\%\text{free}}(\text{CNF})$ is complete when restricted to formulas with at most k disjuncts in each conjunct. Nevertheless, the above construction does provide some nontrivial bounds on the size of conjuncts. Observe that no conjunct of A' in the above proof has more than $\log_2(n) + 7$ variables, of which at most 3 are in V_F. It follows that $L_{\%\text{free}}(\text{CNF})$ is complete in PSPACE when restricted to inputs (A, V_T, V_F) such that each conjunct of A has at most $\log_2(|A|)$ variables, including at most three variables from V_F.

We next consider the game $G_{\text{avoid}}(\text{POS CNF})$. Recall that in $G_{\text{avoid}}(\text{POS CNF})$, the game ends, and the last player loses, as soon as all conjuncts of the input formula are satisfied (i.e., at least one variable in each conjunct has been played). This game has the following interesting property. If at some point in the playing of the game, the number of unplayed variables is even and every conjunct which is not yet satisfied has an odd number of distinct variables, then the player who is to move can win. This is true because the player to move can, on the current move or any future move, avoid immediate loss by choosing some unsatisfied conjuct and playing a variable that is not in it—such a variable exists by the hypothesis. If he plays in this way, he can never lose; hence his opponent must lose. Similarly, at any point when an odd number of variables remain and every unsatisfied conjunct has an even number of distinct variables, the player who is to move can win. In view of this, we can think of the game as a kind of race. If the input formula has an even number of variables, the first player tries to satisfy all the even conjuncts (i.e., conjuncts having an even number of variables) while leaving some odd conjunct unsatisfied. The second player tries to satisfy all the odd conjuncts while leaving some even conjunct unsatisfied. This insight motivates the following proof.

THEOREM 3.10. *$L_{\text{avoid}}(\text{POS CNF})$ is log-complete in* PSPACE.

Proof. We show $L_{\%\text{free}}(\text{CNF}) \leqslant_{\lg} L_{\text{avoid}}(\text{POS CNF})$. Let an input (A, V_T, V_F) to $G_{\%\text{free}}(\text{CNF})$ be given, with $A = A_1 \wedge \cdots \wedge A_m$, where each A_i is a disjunction of literals. Assume without loss of generality that $V_T = \{x_1, ..., x_n\}$, $V_F = \{y_1, ..., y_n\}$, that every variable of $V_T \cup V_F$ occurs both negated and unnegated in A, and that every conjunct of A contains some variable of V_T. (The first of these last two assumptions can be made to hold by adding conjuncts of the form $(\xi \vee \neg\xi)$ to A; then the second can be made to hold by replacing each deficient conjunct A_i by $(x_1 \vee A_i) \wedge (\neg x_1 \vee A_i)$.) Let A' be the conjunction of $\{C_i, D_i : 1 \leqslant i \leqslant n\} \cup \{B_k : 1 \leqslant k \leqslant m\}$, where $C_i = (x_i \vee \bar{x}_i)$, $D_i = (y_i \vee \bar{y}_i \vee y_i')$ and B_k is the disjunction of the variables $\{\xi, \xi' : \xi$ occurs unnegated in $A_k\} \cup \{\bar{\xi}, \bar{\xi}' : \xi$ occurs negated in $A_k\}$. Thus, the variables of A' are $\{x_i, y_i, \bar{x}_i, \bar{y}_i, x_i', y_i', \bar{x}_i', \bar{y}_i' : 1 \leqslant i \leqslant n\}$. We call the variables $x_i, \bar{x}_i, x_i', \bar{x}_i'$ *X-variables*, and $y_i, \bar{y}_i, y_i', \bar{y}_i'$ *Y-variables*. It is easy to see that there is a log-space algorithm which computes A' from (A, V_T, V_F).

In a played game on $G_{\text{avoid}}(A')$, we say that a player *initiates* a conjunct of A' if he is the first player to play a variable in that conjunct.

The motivating idea lies in the remarks preceding this theorem: Because each C_i has an even number of variables and each D_i has an odd number, we expect that I (resp. II) will want to try to satisfy all the C_i (resp. D_i) as quickly as possible by initiating some new C_i (resp. D_i) on each move. If we could be sure that the players played in this way, the game would exactly imitate $G_{\%\text{free}}(A, V_T, V_F)$ and so the proof would be easy. The essence of the detailed arguments which follow is to verify that neither player has anything to gain by not playing in this way.

We now show that T can win $G_{\%\text{free}}(A, V_T, V_F)$ iff I can win $G_{\text{avoid}}(A')$. As before, we abbreviate $G_{\%\text{free}}(A, V_T, V_F)$ by $G_{\%}(A)$.

($\Rightarrow$) Assume that T has a winning strategy for $G_{\%}(A)$. We describe player I's strategy for $G_{\text{avoid}}(A')$ in terms of a "side game" of $G_{\%}(A)$ which I plays in parallel with his playing of the main game $G_{\text{avoid}}(A')$. At any time when I is to move in $G_{\text{avoid}}(A')$, let him play by the following rules:

T1. On his first move, player I first makes the first move for player T in the side game, according to T's winning strategy for $G_{\%}(A)$. Let x_i be the variable played by T. In the main game, I plays x_i or $\bar{x}_i$, depending on whether T set x_i to 1 or 0.

T2. If II on his last move initiated D_i for some i, I turns to the side game and moves for F as follows: If II played y_i or y_i', F sets y_i to 1; otherwise, F sets y_i to 0. Then I makes a move for T in the side game, according to T's winning strategy. Let x_k be the variable played by T. In the main game, I then plays x_k or $\bar{x}_k$, depending on whether T set x_k to 1 or 0. If, however, this variable has already been played in the main game, I arbitrarily plays some variable subject to the conditions in rule T3.

T3. Otherwise, I does nothing in the side game, but arbitrarily plays some variable in the main game, subject to the following conditions: If possible, he chooses an X-variable; otherwise, he chooses, if possible, a variable which does not immediately lose, that is, a variable that does not occur in every unsatisfied conjunct of A'.

At first glance, rule T2 would appear to be incomplete, since it does not tell what to do if the move made for F in the side game is the last move of that game. This will happen only when rule T2 is invoked for the nth time, that is, after II has initiated D_i for every i. We claim that at this point I will already have won $G_{\text{avoid}}(A')$. To see this, go ahead and make F's final move in the side game as in rule T2. In view of rules T1 and T2, at least one of $\{x_i, \bar{x}_i\}$ and at least one of $\{y_i, \bar{y}_i, y_i'\}$ have been played in the main game for each i, $1 \leqslant i \leqslant n$. Hence all the conjuncts C_i and D_i are satisfied. Also, since T played by a winning strategy in the side game, every conjunct of A is satisfied in the side game; it is clear then from rule T2 that every conjunct B_k has been satisfied in the main game. Thus all conjuncts of A' became satisfied upon II's last move, and so I has won as claimed.

We now show that the above strategy wins for I. First, observe that each formula B_k contains some X-variable, since by assumption each conjunct of A contained a variable in V_T. Thus, if it is ever I's turn to move and all X-variables have been played, then all of the formulas B_k have been satisfied and so every unsatisfied conjunct of A' is D_k for some k. Thus every unsatisfied conjunct has an odd number of variables, so I will win by the remarks which precede this theorem.

So let us restrict attention to played games in which it is never the case that all X-variables have been played when I is to move. In such a game, by the above rules, I plays only X-variables. Since the game will not be over until each D_i is satisfied, and since the formulas D_i contain Y-variables only, II must initiate each D_i. But, as argued above, as soon as II has initiated the last D_i, he has lost the game. Thus the strategy we have described is a winning strategy for I.

($\Leftarrow$) Assume that F has a winning strategy for $G_{\%}(A)$. In $G_{\text{avoid}}(A')$, let II play his first n moves as follows:

F1. If player I has on every one of his moves so far initiated some C_i, then II responds according to F's winning strategy for $G_{\%}(A)$, using the correspondence whereby setting ξ to 1 or 0 in $G_{\%}(A)$ corresponds to playing ξ or $\bar{\xi}$, respectively, in $G_{\text{avoid}}(A')$. Note that whenever II plays in this way he initiates some D_i.

F2. Otherwise, he chooses some i such that D_i has not been initiated and plays y_i. (If all the D_i have been initiated, then he plays some arbitrary Y-variable.)

Consider the situation after each player has made n moves, with II playing as above. It is clear that all the conjuncts D_i are already satisfied. If I did not initiate some C_i on every move, then some C_i is still unsatisfied. If I did initiate some C_i on every move, then some B_k is unsatisfied, since II used F's winning strategy. Thus, in any case, I is to move and all unsatisfied conjuncts have an even number of variables, so by the remarks preceding this theorem II can easily win. Thus, II has a winning strategy for $G_{\text{avoid}}(A')$. ▮

In view of the above proof and the remarks following Lemma 3.9, L_{avoid}(POS CNF) is complete even when restricted to inputs A such that no conjunct of A has more than $\log_2(|A|)$ variables.

Corollary 3.11. *L(SIFT) is log-complete in* PSPACE.

Proof. We show $L_{\text{avoid}}(\text{POS CNF}) \leqslant_{\text{lg}} L(\text{SIFT})$. Let a positive CNF formula $A = A_1 \wedge \cdots \wedge A_m$ be given, and assume that the variables of A are $\{x_1, ..., x_n\}$. For $k = 1, ..., m$, let $S_k = \{j : x_j \text{ occurs in } A_k\}$. Let $\mathcal{A} = \{S_k : |S_k| \text{ is even}\}$, $\mathcal{B} = \{S_k : |S_k| \text{ is odd}\}$. By the remarks preceding Theorem 3.10, if n is even, player I can win SIFT on $(\mathcal{A}, \mathcal{B})$ iff player I can win $G_{\text{avoid}}(A)$. Similarly, if n is odd, I can win SIFT on $(\mathcal{B}, \mathcal{A})$ iff I can win $G_{\text{avoid}}(A)$. ▮

$G_{\%\text{avoid}}$(POS CNF) is defined like G_{avoid}(POS CNF), except that the variables of the input formula are partitioned into two disjoint sets, V_{I} and V_{II}, of equal cardinality, and player I (II) is required to play variables from V_{I} (V_{II}). This game has the following useful property. If at some point in the game every unsatisfied conjunct has fewer variables of V_{I} than of V_{II}, then player I can win. This is true because, once this condition holds, player I can on any subsequent move avoid immediate defeat simply by choosing some unsatisfied conjunct and playing some variable of V_{I} that is not in it; this is possible because whenever I is to move, equal numbers of variables in V_{I} and V_{II} remain unplayed. Similarly, if at any point in the game every unsatisfied conjunct has at least as many variables of V_{I} as of V_{II}, and at least one conjunct is unsatisfied, player II can win. The following proof of completeness of $L_{\%\text{avoid}}$(POS CNF) is very similar to that of L_{avoid} (POS CNF). It uses the property just stated in place of the property described before Theorem 3.10.

THEOREM 3.12. *$L_{\%\text{avoid}}$(POS CNF) is log-complete in* PSPACE.

Proof. Let $L_{\%\text{free}}(\text{CNF}/3\text{F}) = \{(A, V_{\text{T}}, V_{\text{F}}) \in L_{\%\text{free}}(\text{CNF})$: each conjunct of A contains at most 3 occurrences of variables in $V_{\text{F}}\}$. By the remarks following the proof of Lemma 3.9, $L_{\%\text{free}}$(CNF/3F) is log-complete in PSPACE. We show that $L_{\%\text{free}}$ (CNF/3F) $\leqslant_{\text{lg}} L_{\%\text{avoid}}$(POS CNF). Let some input $(A, V_{\text{T}}, V_{\text{F}})$ to $G_{\%\text{free}}$(CNF/3F) be given, with $A = A_1 \wedge \cdots \wedge A_m$. Assume that $V_{\text{T}} = \{x_1, \ldots, x_n\}$, $V_{\text{F}} = \{y_1, \ldots, y_n\}$. Also assume that every conjunct A_k of A contains at least as many variables of V_{T} as of V_{F} (counting a variable twice if it occurs both negated and unnegated). This entails no loss of generality, since any conjunct A_k which fails to meet this condition can be replaced by the 8 conjuncts comprising the CNF expansion of $(x_1 \wedge \neg x_1) \vee (x_2 \wedge \neg x_2) \vee (x_3 \wedge \neg x_3) \vee A_k$, and each new conjunct will have at least as many variables of V_{T} as of V_{F}. Let A' be the conjunction of $\{C_i, D_i : 1 \leqslant i \leqslant n\} \cup \{B_k : 1 \leqslant k \leqslant m\}$, where $C_i = (x_i \vee \bar{x}_i)$, $D_i = (y_i \vee \bar{y}_i)$, and B_k is formed from A_k by replacing each negated variable $\neg x_i$ or $\neg y_i$ by the unnegated variable $\bar{x}_i$ or $\bar{y}_i$. Let $V_{\text{T}}' = \{x_i, \bar{x}_i : 1 \leqslant i \leqslant n\}$, $V_{\text{F}}' = \{y_i, \bar{y}_i : 1 \leqslant i \leqslant n\}$. It is easy to see that $(A', V_{\text{F}}', V_{\text{T}}')$ is log-space computable from $(A, V_{\text{T}}, V_{\text{F}})$. We write $G_{\%}(A)$ and $G_{\%\text{avoid}}(A')$ instead of $G_{\%\text{free}}(A, V_{\text{T}}, V_{\text{F}})$ and $G_{\%\text{avoid}}(A', V_{\text{T}}', V_{\text{F}}')$.

The proof that T can win $G_{\%}(A)$ iff I can win $G_{\%\text{avoid}}(A')$ is identical to the proof in Theorem 3.10 that T can win $G_{\%}(A)$ iff I can win $G_{\text{avoid}}(A')$, except for the following points: (a) $G_{\%\text{avoid}}(A')$ replaces $G_{\text{avoid}}(A')$, (b) references to primed variables such as y_i' should be ignored, (c) references to I playing Y-variables or II playing X-variables should be ignored, since this is not possible in $G_{\%\text{avoid}}(A')$, and (d) the phrase "has an odd (even) number of variables" should be replaced by "has fewer (at least as many) X-variables than (as) Y-variables." ∎

COROLLARY 3.13. *L_{seek}(POS CNF) and $L_{\%\text{seek}}$(POS CNF) are log-complete in* PSPACE.

Proof. We show $L_{\text{avoid}}(\text{POS CNF}) \leqslant_{\text{lg}} L_{\text{seek}}$(POS CNF). Let a positive CNF formula $A = A_1 \wedge \cdots \wedge A_m$ be given. Let A' be the conjunction of $\{(u_1 \vee u_2)\} \cup \{(u_i \vee A_k) : 1 \leqslant k \leqslant m, i = 1, 2\}$, where u_1, u_2 are new variables not occurring in A. In $G_{\text{seek}}(A')$ it is fatal to play one of u_1, u_2 before all the A_k are satisfied, because one's opponent will play the other of these two variables and win. But after all the A_k are satisfied, playing u_1 or u_2 wins at once. In view of this, it is easy to see that I can win $G_{\text{seek}}(A')$ iff I can win $G_{\text{avoid}}(A)$.

Next we show $L_{\%\text{avoid}}(\text{POS CNF}) \leqslant_{\text{lg}} L_{\%\text{seek}}$(POS CNF). Let an input $(A, V_{\text{I}}, V_{\text{II}})$ to $G_{\%\text{avoid}}$(POS CNF) be given. Letting A' be the same as above, it is easy to see that I can win $G_{\%\text{seek}}(A', V_{\text{I}} \cup \{u_1\}, V_{\text{II}} \cup \{u_2\})$ iff I can win $G_{\%\text{avoid}}(A, V_{\text{I}}, V_{\text{II}})$. ∎

4. FURTHER OBSERVATIONS

We here offer some observations on the foregoing results and proofs. Some familiarity with the notion of *NP*-completeness is assumed (e.g., see [3, 9]).

Comparison with NP-Complete Problems

In comparing these problems with known *NP*-complete problems, some interesting contrasts can be seen. The first lies in the difficulty of the proofs. It seems fair to say that for *NP*-complete problems having a comparable simplicity of definition, the proofs of completeness would typically be much simpler than those given here. The relative simplicity of *NP*-completeness proofs probably reflects the fact that *NP*-complete problems simulate Turing machine computations in a very straightforward way (cf. the proof of *NP*-completeness of the satisfiability problem in [3]). Complete problems in PSPACE can describe computations more concisely, but by means which are less straightforward and hence less readily transferable between problems.

Another contrast lies in the large gap which seems to exist between many of the general games we have shown complete in PSPACE and any special cases of these games which are known to be polynomial-time decidable. For example, we do not know how to solve NODE KAYLES even on extremely simple graphs such as an n by m rectangular grid, or acyclic graphs of degree 3. Another example is G_{pos}(POS CNF 11); we have not been able to show this complete for any fewer than 11 variables per conjunct, and yet it appears to be very hard even when restricted to three variables per conjunct. Such large "gray zones" seem much less likely to be found in *NP*-complete problems of comparable simplicity. For example, the satisfiability problem is *NP*-complete with three literals per conjunct, but it is in P when restricted to two literals per conjunct [3]. Similarly, the problem of deciding whether a graph is 3-colorable is in P for graphs of degree three, but is *NP*-complete on planar graphs of degree four [6]. Again, this contrast probably reflects the more straightforward manner in which *NP*-complete problems simulate Turing machines.

Games and NP-Hardness

A problem is *NP-hard* if some *NP*-complete problem can be reduced to it in polynomial time. Any problem that is log-complete in PSPACE is also *NP*-hard, and *NP*-hardness alone constitutes evidence of intractability which is, for practical purposes, just about as strong as completeness in PSPACE. Thus, a researcher whose main interest is in finding evidence for intractability of a problem may well feel satisfied in proving *NP*-hardness, thereby sparing himself the extra work of trying to prove completeness in PSPACE.

But proving completeness in PSPACE is not always "extra work." It happens for quite a few games that a proof of *NP*-hardness generalizes almost immediately to a proof of completeness in PSPACE. This is particularly true of "impartial" games, that is, games where the same set of moves is permitted to both players. Many of the proofs in this paper lend support to this idea, in that it is hard to see how limiting the goal to an *NP*-hardness proof would result in any significant simplification. (On the other hand, there can be a striking difference for games which are not impartial. For example, it is almost trivial to prove that $G_{\%\text{free}}$(CNF) is *NP*-hard, whereas our proof of completeness in PSPACE of this game is very long and complicated.)

A number of researchers have exhibited games which are *NP*-hard. Berlekamp [2] has shown that the game of *L-R* Hackenbush on Redwood Furniture is *NP*-complete. This game is a restricted form of the game called Hackenbush Restrained in [14]; hence, the latter game is *NP*-hard. Fraenkel and Yesha [15] have shown the *NP*-hardness of a number of games on graphs. It would be interesting to investigate whether any of these *NP*-hard games are complete in PSPACE.

Other Complexity Categories

There are many games in which the length of a played game is not bounded by a polynomial in the size. One example of such a game is the familiar game of checkers, generalized to be played on an n by n square board. Such games do not, in general, lie in PSPACE.

Chandra and Stockmeyer [13] have exhibited a number of games on propositional formulas which are complete in exponential time, a condition which implies that these games cannot be decided in less than exponential time. The same paper introduces the interesting theoretical notion of alternating computers (a generalization of nondeterministic computers), which provides a link between various time and space complexity classes and also has an intimate connection with games.

Advantages of Using Propositional Formulas

As seen in Section 1, a number of our games on propositional formulas can be more naturally described as games on collections of sets. But there are a number of reasons why we prefer to work with games on formulas. One reason is that the propositional calculus offers a degree of expressive power that is not available from sets alone. This can be seen in the proofs of Theorem 3.6 and Lemma 3.9, where rather complicated formulas are defined, to be replaced by equivalent CNF formulas. The CNF formulas do not have to be explicitly written out and would be very clumsy if they were; it suffices to know that they exist and are not too large.

Another benefit of propositional formulas is their heuristic value. A process which we used repeatedly in discovering our games is the following. First think of some game that can be played on arbitrary propositional formulas, and show it complete in PSPACE. Then try to narrow down the class of formulas on which completeness holds, e.g., to CNF formulas or positive CNF formulas. We have already noted how such a heuristic process led to our proof of completeness of G_{pos}(POS CNF). Such a process is also evident in the use of Lemma 3.9 to prove the completeness of $G_{\%\,\text{free}}$(CNF).

A third use of propositional formulas has to do with games based on *NP*-complete problems. We have described in [10] a method whereby *NP*-complete problems can be used to simulate the propositional calculus, thereby allowing any game on propositional formulas to be translated into a game played on inputs to the *NP*-complete problem, such that the new game is log-complete in PSPACE if the original game was. This gives rise to a multitude of new games that are complete in PSPACE. These games generally have, however, a somewhat contrived character. One example of a game which is shown complete in PSPACE in this way is the following game played on graphs. The players take turns choosing arcs from a predefined subset of the arcs, subject to the condition that there exist a Hamiltonian cycle of the graph which contains all arcs chosen so far.

The first player to violate this condition loses. We refer to [10] for a discussion of these games, which is beyond the scope of this paper.

Open Questions

We briefly list a few games whose complexity is unresolved. These are mostly variations of games that we have shown complete in PSPACE.

1. ARC KAYLES. This is the analogue of NODE KAYLES played on arcs of a graph instead of nodes. (See the remarks after Corollary 3.3.)

2. ARC KAYLES and NODE KAYLES restricted to starlike graphs. We call a graph *starlike* if it is connected and acyclic and has at most one node of degree greater than 2. Can a polynomial-time algorithm be found?

3. NODE KAYLES restricted to bipartite graphs. This game looks like it could be complete in PSPACE.

4. IMPARTIAL HEX. This is played like Generalized Hex, described in the Introduction, except that all markers are white. The winner is the first player who completes a chain of white markers from s to t. (This can also be played on directed graphs.) If the preceding game, NODE KAYLES on bipartite graphs, is hard, then so is this one (both directed and undirected version), by virtue of a simple reduction.

5. BIGRAPH NODE KAYLES restricted to inputs $(G, V_{\text{I}}, V_{\text{II}})$ such that $|V_{\text{I}}| = |V_{\text{II}}|$. It would seem an elementary gesture of fairness to give both players the same number of nodes to play on. But we have not been able to show the game in this form hard.

6. Other restricted problems. For each game on graphs (e.g., GEOGRAPHY, NODE KAYLES, Generalized Hex), the complexity of the game restricted to planar graphs is of obvious interest. For games on formulas, one can try to limit the number of variables in each clause. For example, how hard is G_{pos}(POS CNF) when restricted to positive CNF formulas with at most three variables in each conjunct?

5. Conclusion

We have exhibited a number of two-person games, based on simple combinatorial ideas, for which the problem of deciding the outcome of optimal play is log-complete in polynomial space. This provides strong evidence, although not absolute proof, that efficient general algorithms for deciding these games do not exist. We have also pointed out asymptotic lower bounds, roughly of the form $n^{1/k}$, on the space required to decide each of these problems.

For many of these games there is a rather large gap between the general form of the game which is complete and any special cases for which polynomial-time algorithms are known. In this respect, these results are less helpful than comparable *NP*-completeness results, which often provide rather sharp delineation between "solvable" and "hard" cases.

On a more theoretical level, these games are of interest because they augment the still rather small list of known problems that are complete in PSPACE. Every such problem serves as a test case for the open question of whether $P =$ PSPACE, since such a problem is decidable in polynomial time if and only if $P =$ PSPACE. Thus the role of these problems in PSPACE is analogous to that of the NP-complete problems in relation to the well-known open question of whether $P = NP$.

These results are a natural sequel to the proof by Even and Tarjan [4] that the game of Generalized Hex is complete in PSPACE. Together with that result, they firmly establish the notion that games are natural candidates for complete problems in PSPACE.

It is hoped that these results will pave the way for further completeness results, both by providing specific problems for use in reductions and by sharpening our intuitive understanding of the kind of complexity that serves to make a game or other problem complete in polynomial space.

Appendix

Notation. If Γ is an alphabet (i.e., a finite set of symbols), then Γ^* denotes the set of all strings over Γ (that is, all finite sequences of elements of Γ). $|A|$ denotes the cardinality of the set A, or, if A is a sequence or string, the length of A.

Propositional formulas. By a *variable* we mean a string of symbols consisting of a letter or augmented letter such as x, y, u, $\hat{u}$, x', $\bar{x}$ followed by a subscripted string of decimal digits and commas. Examples of variables are x_0, $z_{1,2}$, $\bar{y}_8$, $\hat{u}_{10,4}$. VBL denotes the set of all variables. By a *(propositional) formula* we mean a string formed from variables and the symbols $\neg$, $\wedge$, $\vee$, (,) which is contained in the smallest set F such that VBL $\subseteq$ F and $\neg(A)$, $(A) \wedge (B)$ and $(A) \vee (B)$ are in F whenever A and B are in F. The *immediate subformulas* of C are the formulas A, B such that C is $\neg(A)$ or $(A) \wedge (B)$ or $(A) \vee (B)$. A *subformula* of C is a member of the smallest set which contains C and contains every immediate subformula of any of its members. A formula is *atomic* if it has no immediate subformulas, i.e., consists of a single variable. We generally omit parentheses where they are not needed for readability; e.g., we informally write $x_1 \vee \neg x_2 \vee x_3$ instead of $((x_1) \vee (\neg(x_2))) \vee (x_3)$. We sometimes write formulas using the additional connectives $\equiv$ and $\not\equiv$. $A \equiv B$ means "A if and only if B"; this can be regarded as an abbreviation of $(A \vee \neg B) \wedge (B \vee \neg A)$. $A \not\equiv B$ is equivalent to $\neg(A \equiv B)$.

A *literal* is a formula of the form ξ or $\neg\xi$, where $\xi \in$ VBL. Note that $\bar{x}_i$, $\bar{y}_i$, etc. are simple variables and not negated variables. A formula is in *conjunctive normal form*, or a CNF *formula*, if it is of the form $B_1 \wedge B_2 \wedge \cdots \wedge B_n$ with each B_i of the form $(\alpha_{i,1} \vee \cdots \vee \alpha_{i,m_i})$, where each $\alpha_{i,j}$ is a literal and m_i, $n \geqslant 1$. Similarly, a *disjunctive normal form* (*DNF*) formula is one which is a disjunction of conjunctions of literals. A formula is *positive* if $\neg$ does not occur in it (i.e., its only connectives are $\wedge$ and $\vee$).

If s is a function from the set of variables of the formula C into $\{0, 1\}$, then *s satisfies C*, or makes *C true*, iff C is true when the symbols 0, 1, $\neg$, $\vee$, $\wedge$ are interpreted in the usual way as "false," "true," "not," "or," "and."

Graphs and directed graphs. A *graph* is a pair $G = (V, E)$, where V is a nonempty finite set (the *nodes* of G) and E is a collection of unordered pairs of elements of V (the *arcs* of G). The nodes u and v are *adjacent* iff $\{u, v\} \in E$. The *degree* of a node is the number of arcs containing it; the *degree* of a graph is the maximum degree of any of its nodes. A node is *isolated* if its degree is 0. A *directed graph* is similarly defined, except that the elements of E are ordered pairs of elements of V; (u, v) is called the *directed arc* from u to v.

Details of encoding. For a game to be formally stated as a recognition problem, its inputs must be defined as strings over some alphabet. We assume that the inputs to all our games are presented as strings over some fixed alphabet Σ, which contains at least all the symbols used to form propositional formulas. (Thus propositional formulas used as inputs need no encoding.) Graphs and directed graphs are assumed to be encoded as a list of names of nodes, followed by a list of arcs, each arc being given as a pair of names of nodes. We omit further details, since just about any straightforward system of encoding is equivalent for our purposes (cf. [9]). When we refer to the size of an input, we always mean the length of the string in Σ^* that represents it.

Acknowledgment

I am grateful to R. M. Karp, R. E. Tarjan, and E. R. Berlekamp for discussions which stimulated various parts of this work.

References

1. W. W. Rouse Ball and H. S. M. Coxeter, "Mathematical Recreations and Essays," 11th ed., London, 1939.
2. E. R. Berlekamp, "*L-R* Hackenbush is *NP* Complete," rough preliminary draft, April 1975.
3. S. A. Cook, The complexity of theorem-proving procedures, *in* "Third ACM Symposium on Theory of Computing," pp. 151–158, Association for Computing Machinery, New York, 1971.
4. S. Even and R. E. Tarjan, A combinatorial problem which is complete in polynomial space, *in* "Seventh Annual ACM Symposium on Theory of Computing," pp. 66–71, Association for Computing Machinery, New York, 1975.
5. S. Even and R. E. Tarjan, A combinatorial problem which is complete in polynomial space, *J. Assoc. Comput. Mach.* 23 (1976), 710–719.
6. M. R. Garey, D. S. Johnson, and L. Stockmeyer, Some simplified *NP*-complete problems, *in* "Sixth Annual ACM Symposium on Theory of Computing," pp. 47–63, Association for Computing Machinery, New York, 1974.
7. R. K. Guy and C. A. B. Smith, The *G*-values of various games, *Proc. Cambridge Philos. Soc.* 52 (1956), 514–526.
8. N. D. Jones, Space-bounded reducibility among combinatorial problems, *J. Comput. System Sci.* 11 (1975), 68–85.
9. R. M. Karp, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–104, Plenum, New York, 1972.
10. T. J. Schaefer, Complexity of decision problems based on finite two-person perfect-information games, *in* "Eighth Annual ACM Symposium on Theory of Computing," pp. 41–49, Association for Computing Machinery, New York, 1976.

11. L. J. STOCKMEYER, The polynomial-time hierarchy, IBM Research Report RC5379, 1975; *Theoret. Comput. Sci.* 3 (1976), 1–22.
12. L. J. STOCKMEYER AND A. R. MEYER, Word problems requiring exponential time: Preliminary report, *in* "Fifth Annual ACM Symposium on Theory of Computing," pp. 1–9, Association for Computing Machinery, New York, 1973.
13. A. K. CHANDRA AND L. J. STOCKMEYER, Alternation, *in* "17th Annual Symposium on Foundations of Computer Science," pp. 98–108, IEEE Computer Society, 1976.
14. J. H. CONWAY, "On Numbers and Games," Academic Press, New York, 1976.
15. A. S. FRAENKEL AND Y. YESHA, "Complexity of Problems in Games, Graphs and Algebraic Equations," manuscript, Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel, 1977.