A Combinatorial Problem Which Is Complete in Polynomial Space

S. EVEN

Technion, Haifa, Israel

AND

R. E. TARJAN

Stanford University, Stanford, California

ABSTRACT This paper considers a generalization, called the Shannon switching game on vertices, of a familiar board game called Hex It is shown that determining who wins such a game if each player plays perfectly is very hard, in fact, if this game problem is solvable in polynomial time, then any problem solvable in polynomial space is solvable in polynomial time This result suggests that the theory of combinatorial games is difficult

KEY WORDS AND PHRASES. completeness in polynomial space, computational complexity, Hex, Shannon switching game

CR CATEGORIES 3 69, 5 25, 5 32

1. *Introduction*

Let G be an arbitrary graph with two distinguished vertices s and t. Suppose that two players, *short* and *cut*, alternately select vertices of G (except s and t). No player is allowed to select a vertex previously selected by another player The game concludes with *short* winning if some of the vertices selected by *short* form a path between s and t. The game concludes with *cut* winning if every path between s and t contains a vertex selected by *cut* We call this game a *Shannon switching game on the vertices* of G In a common version of this game called *Hex*, G is a diamond-shaped section of a planar triangular grid (Figure 1) The question is who wins if both players play perfectly, and what is the winning strategy? (Hex is peculiar in the following sense: It is easy to prove that the first player can win, but no efficient way to decide on the moves is known.)

If we allow the players to select edges of G instead of vertices, the game is a *Shannon switching game on the edges* of G. Efficient algorithms exist for finding winning strategies for such a game [2, 3]. These strategies are based on finding a pair of minimum overlapping spanning trees of G, and the fastest algorithm known requires $O(n^2)$ time if G has n vertices [10]. However, the Shannon switching game on vertices seems to be much harder to solve Here we give theoretical evidence to support this view. Specifically, we show that any algorithm for efficiently determining who wins the game on an arbitrary graph G can be used to efficiently carry out *any* polynomial-space bounded

The research of R E Tarjan was sponsored by the National Science Foundation under Grant GJ 35604X and by a Miller Research Fellowship at the University of California, Berkeley, and by the National Science Foundation under Grant DCR 72-03752 A02 at Stanford University, Stanford, California

Authors' addresses S Even, Department of Computer Science, Technion, Haifa, Israel, R E Tarjan, Department of Computer Science, Stanford University, Stanford, CA 94305

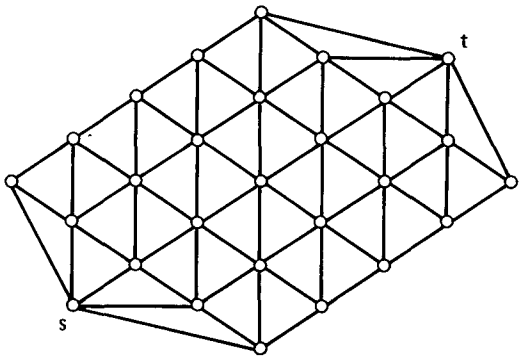

FIG 1 $k \times k$ Hex for $k = 5$

computation (the game problem is *complete in polynomial space*). As a corollary we show that the Shannon switching game played on the edges of a *directed* graph is polynomial-space complete.

2. *A Polynomial-Space Algorithm*

First we shall show that there is a polynomial-space bounded algorithm for determining who wins the game. By convention we shall assume that *short* moves first. (This is no loss of generality.) If G has n vertices, only $n - 2$ moves can be made in any game. Suppose we construct a *game tree* T for G [7]. T is a directed, rooted tree. Each vertex of T denotes a position of the game (indicating all moves made so far). The root of T denotes the initial position (no moves made). The sons of any vertex v of T denote the positions reachable from the position v by one move of the player whose turn it is. We use $v \rightarrow w$ to mean that w is a son of v in T A leaf of T corresponds to a final game position. We call a vertex of T a *short vertex* if it is *short*'s turn to move, a *cut vertex* if it is *cut*'s turn to move.

T clearly has depth at most $n - 2$ and contains at most $\sum_{i=0}^{n-2}(n-2)!/i!$ vertices (many of which correspond to the same game position but represent different sequences of moves). For a vertex v in T, let $W(v) = 1$ if *short* has a forced win from the position v denotes, $W(v) = 0$ otherwise. $W(v)$ is easy to calculate if v is a leaf of T. The following recursive formula defines $W(v)$ for all vertices.

If v is a *short* vertex, $W(v) = 1$ if there exists $v \leftarrow w$ such that $W(w) = 1$; $W(v) = 0$ otherwise.

If v is a *cut* vertex, $W(v) = 1$ if for all $v \rightarrow w$, $W(w) = 1$; $W(v) = 0$ otherwise.

We desire the value of $W(r)$ where r is the root of T. It is easy to calculate $W(r)$ by exploring T in a depth-first fashion [7, 9]. We need a stack to store the moves made in reaching the current position; the total amount of storage required is $O(n \log n)$ bits for the stack plus no more than $O(n^2 \log n)$ bits to store the graph and any work area required. Thus the search requires polynomial space to determine who wins the game. A simple modification gives an algorithm for playing the game in a winning fashion. A similar algorithm solves any Shannon switching game played on the edges of a directed graph.

There is no obvious way to determine the winner in polynomial time, even if we allow a nondeterministic algorithm It is thus reasonable to suspect that the game problem is even harder than the NP-complete problems [1, 4, 5], which include such problems as graph coloring, finding a maximum clique, and finding a Hamiltonian path.

3. *The Reduction*

We shall show that the Shannon switching game on vertices is log-space complete in

polynomial space. We need a few definitions. A *problem* is a set of words over a finite alphabet. Let Σ, Δ be finite alphabets and let $A \subseteq \Sigma^*$, $B \subseteq \Delta^*$ be problems. We say $A \leq_{\log} B$ (A is *log-space reducible* to B) if there is a log-space computable function f such that $x \in A$ iff $f(x) \in B$, for all $x \in \Sigma^*$. Log-space reducibility is reflexive and transitive. Problem A is *polynomial-space solvable* if there exists a polynomial-space bounded Turing machine which accepts A. Problem A is (*log-space*) *complete in polynomial space* if A is polynomial-space solvable and every polynomial-space solvable problem is log-space reducible to A. [1, Ch. 10] provides a good discussion of the meaning and consequences of the notion of completeness in polynomial space. Meyer and Stockmeyer [6] have exhibited several problems complete in polynomial space, including the

Quantified Boolean formula problem: QBF = $\{Q_1x_1Q_2x_2 \ldots Q_mx_mF \mid$ the Q_i are quantifiers, the x_i are Boolean variables, F is a well-formed formula in conjunctive normal form with variables $x_1, \ldots, x_m$, and the quantified formula is true$\}$.

To show that the game problem is complete in polynomial space, we exhibit a construction which, given any quantified Boolean formula $(Q_ix_i)F$, will produce a game graph G such that $(Q_ix_i)F$ is true if and only if *short* wins on G. We describe the construction informally; it will be obvious that the construction is log-space computable. The construction produces a graph with two parts: a tree (actually a combination of two smaller kinds of trees) and a ladder. Consider a graph of the form shown in Figure 2. It consists of a binary tree B with its root joined to s and some or all of its leaves joined to t. *Short* wins in this graph if and only if t is joined to *all* the leaves of the tree. For if t is joined to all the leaves of the tree, *short* wins by the following strategy:

First play the root.

On succeeding moves play one of the sons (in the tree) of the vertex you last played. Pick a son which is the root of a subtree containing no moves by *cut*.

If t is not joined to all the leaves of the tree, *cut* wins by the following strategy.

If some unplayed vertex will cut all remaining paths from s to t, play it.

Otherwise play one of the sons of the vertex last played by *short*. Find a son which is the root of a subtree having a leaf not connected to t, and play the *other* son.

The idea here is that *short* can force completion of a path from the root to some leaf, but *cut* can choose the leaf and in addition block paths from the root to other leaves. It is easy to prove by induction that these strategies work. We shall use such a tree B to represent the conjunction in the formula F.

Consider, now, a graph of the form shown in Figure 3. It is a tree with three levels (root, sons, grandsons). The root is connected to s One son of the root is a leaf and is connected to t. The other sons each are connected to two grandsons of the root, an a-grandson and a b-grandson. All the a-grandsons, and possibly some of the b-grandsons, are connected to t. It is easy to see that *short* wins in this graph if and only if t is connected to *at least one* of the b-grandsons. We shall use such a tree to represent each clause (disjunction of literals) in the formula F.

The last part of the construction is a ladder. Consider a graph of the form shown in Figure 4. This graph consists of a set of foursomes of the form $\{x_i(1), x_i(2), \bar{x}_i(1), \bar{x}_i(2)\}$. If *short* moves first, he can play to occupy either $\{x_i(1), x_i(2)\}$ or $\{\bar{x}_i(1), \bar{x}_i(2)\}$ for each foursome. For instance, if he wants to occupy $\{x_1(1), x_2(2), \bar{x}_2(1), \bar{x}_2(2), \bar{x}_3(1), \bar{x}_3(2), \cdots\}$ the play is:

$$\textit{short}\ x_1(1)\ x_1(2)\ \bar{x}_2(1)\ \bar{x}_2(2)\ \bar{x}_3(1)\ \bar{x}_3(2)\ \ldots$$

$$\textit{cut}\quad \bar{x}_1(2)\ \bar{x}_1(1)\ x_2(2)\ x_2(1)\ x_3(2)\ x_3(1)\ \ldots$$

All *cut*'s moves are forced.

On the other hand, if *cut* moves first, *he* can play to occupy either $\{x_i(1), x_i(2)\}$ or $\{\bar{x}_i(1), \bar{x}_i(2)\}$ for each foursome. For instance, if he wants to occupy $\{x_1(1), x_1(2), \bar{x}_2(1), \bar{x}_2(2), x_3(1), x_3(2), \ldots\}$, the play is:

$$cut \quad x_1(1)\ x_1(2)\ \bar{x}_2(1)\ \bar{x}_2(2)\ x_3(1)\ x_3(2)\ \ldots$$

$$short\ \bar{x}_1(1)\ \bar{x}_1(2)\ x_2(1)\ x_2(2)\ \bar{x}_3(1)\ \bar{x}_3(2)\ \ldots$$

All *short*'s moves are forced.

In the complete construction we use a ladder to represent the variables of the Boolean formula, with a few extra vertices to represent quantifiers. We use a binary tree to represent the conjunction in the formula, with a leaf for each clause. Each leaf of this binary tree is the root of a three-level tree which represents the corresponding clause. The binary tree contains $3m$ extra leaves, each representing a dummy clause $(x_i \vee \bar{x}_i)$. Each possible dummy clause $(x_i \vee \bar{x}_i)$ is represented three times.

Suppose $(Q_i x_i)F$ is a quantified Boolean formula with F in conjunctive normal form, having m variables and k clauses. We can assume without loss of generality that $Q_1 = Q_m = \exists$. Let G be a graph containing the following vertices:

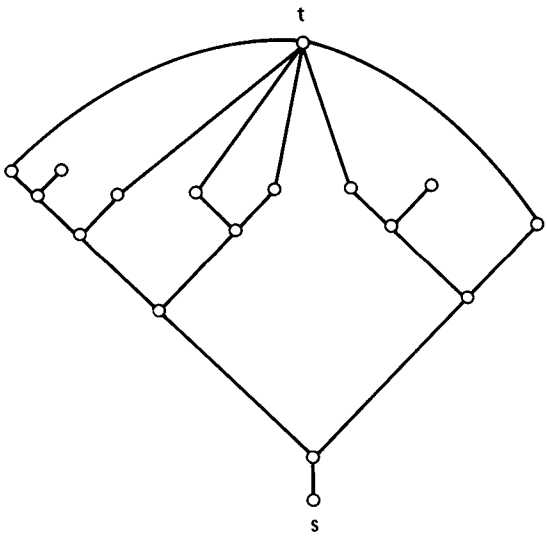

FIG 2 A binary tree representing a conjunction

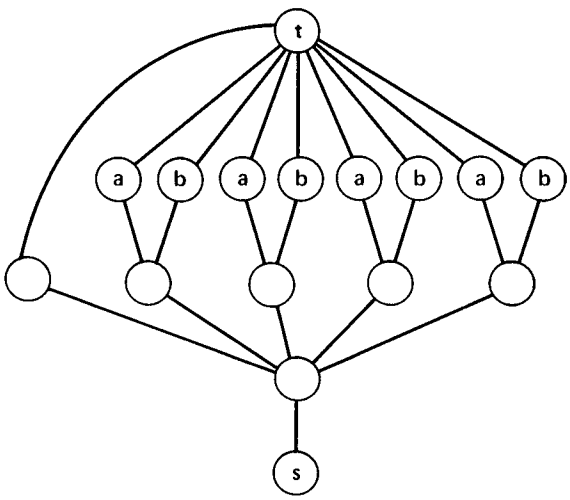

FIG 3. A three-level tree representing a disjunction

s, t;

$x_i(1), \bar{x}_i(1)\ x_i(2), \bar{x}_i(2)$ for $1 \leq i \leq m$, called *variable vertices*;

q_i for each $Q_i \neq Q_{i+1}$, called *quantifier vertices*;

$2(k + 3m) - 1$ vertices forming a binary tree B with $k + 3m$ leaves, designated by $l_1, l_2, \ldots, l_{k+3m}$;

$l_j(0)$, $1 \leq j \leq k + 3m$;

$y(0, j), y(a, j), y(b, j)$ for each literal y occurring in clause j, $1 \leq j \leq k$ (these vertices, together with $l_j(0)$ and l_j, form a three-level tree which represents the jth clause);

$$\left.\begin{array}{l} x_i(0, k + i), x_i(a, k + i), x_i(b, k + i) \\ x_i(0, k + m + i), x_i(a, k + m + i), x_i(b, k + m + i) \\ x_i(0, k + 2m + i), x_i(a, k + 2m + i), x_i(b, k + 2m + i) \end{array}\right\} \text{for } 1 \leq i \leq m.$$

(For example, $l_{k+i}, l_{k+i}(0), x_i(0, k + i), x_i(a, k + i)$, and $x_i(b, k + i)$ form a three-level tree which represents the first, out of three, dummy clauses $(x_i \vee \bar{x}_i)$.).

In addition to the edges of B, G has the following edges:

$(s, x_1(1)), (s, \bar{x}_1(1)), (t, x_1(2)), (t, \bar{x}_1(2))$;

$(x_i(2), x_{i+1}(2)), (x_i(2), \bar{x}_{i+1}(2)), (\bar{x}_i(2), x_{i+1}(2)), (\bar{x}_i(2), \bar{x}_{i+1}(2))$ for $1 \leq i < m$;

$(x_i(2), \bar{x}_i(1)), (\bar{x}_i(2), x_i(1))$ for $1 \leq i \leq m$;

$(x_m(1), r), (\bar{x}_m(1), r)$ where r is the root of B;

$(x_i(1), x_{i+1}(1)), (x_i(1), \bar{x}_{i+1}(1)), (\bar{x}_i(1), x_{i+1}(1)), (\bar{x}_i(1), \bar{x}_{i+1}(1))$ if $Q_i \neq \exists$ or $Q_{i+1} \neq \forall$;

$(x_i(1), q_i), (\bar{x}_i(1), q_i)$ if $Q_i \neq Q_{i+1}$;

$(q_i, x_{i+1}(1)), (q_i, \bar{x}_{i+1}(1))$ if $Q_i = \exists$, $Q_{i+1} = \forall$;

$(q_i, x_i(2)), (q_i, \bar{x}_i(2))$ if $Q_i = \forall$, $Q_{i+1} = \exists$;

$(l_j, l_j(0))$ for $1 \leq j \leq k + 3m$;

$(l_j(0), x_m(2)), (l_j(0), \bar{x}_m(2))$ for $1 \leq j \leq k + 3m$;

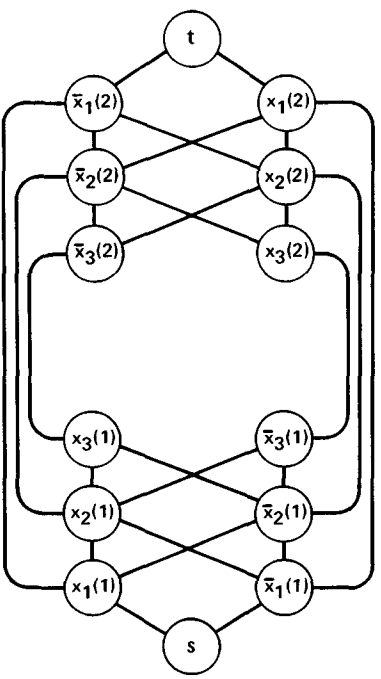

FIG 4 A ladder

$(l_j, y(0,j)), (y(0,j), y(a,j)), (y(0,j), y(b,j)), (y(a,j), x_m(2)), (y(a,j), \bar{x}_m(2)), (y(b,j), y(2))$ for each literal y occurring in clause j, $1 \leq j \leq k$;

$(l_{k+hm+i}, x_i(0, k + hm + i)), (x_i(0, k + hm + i), x_i(a, k + hm + i)),$
$(x_i(0, k + hm + i), x_i(b, k + hm + i)), (x_i(a, k + hm + i), x_m(2)),$
$(x_i(a, k + hm + i), \bar{x}_m(2)), (x_i(b, k + hm + i), x_i(2)), (x_i(b, k + hm + i),$
$\bar{x}_i(2))$ for $1 \leq i \leq m$, $0 \leq h \leq 2$.

Figure 5 illustrates this construction for $\exists x\ \forall y\ \exists z((x \vee z) \wedge (\bar{x} \vee \bar{y}))$.

If we use some simple rule to choose one of the several possibilities for B, this construction defines a function from formulas to graphs. The function is clearly log-space computable. We must show that $(Q_i x_i)F$ is true if and only if *short* wins on G.

First we describe a *normal* way of playing the game and show that *short* wins if and

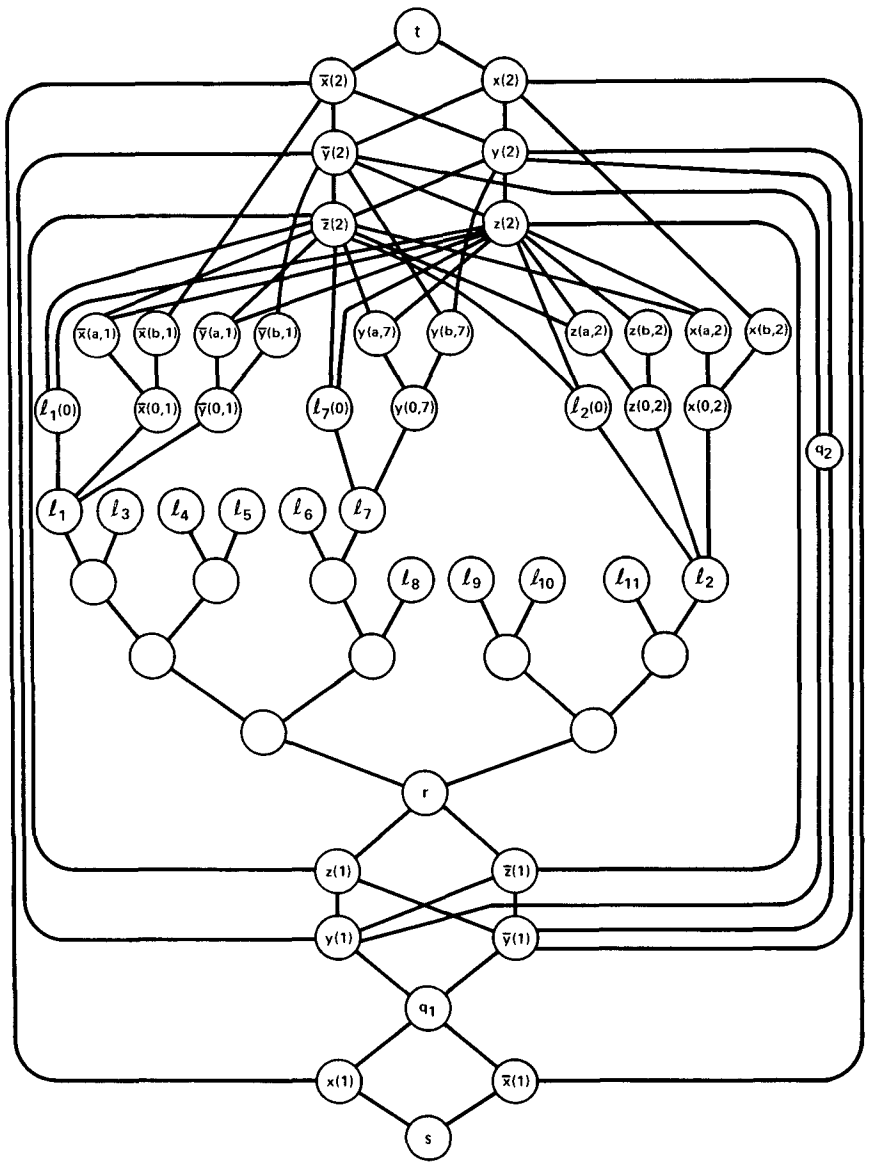

FIG 5 Game graph for $\exists x\ \forall y\ \exists z((\bar{x} \vee \bar{y}) \wedge (x \vee z))$ For clarity, most of the three-level trees, except those for $j = 1, 2$, and 7, are not shown

only if $(Q_i x_i)F$ is true. Second we show that *cut* cannot gain by deviating from the normal play. Third we show that *short* cannot gain by such a deviation.

4. *Normal Play*

Initially *short* has the initiative. He plays in the ladder, choosing vertices to represent the truth values of the first few (existentially quantified) variables. *Cut*'s replies are forced. *Short* then plays the first quantifier vertex (which marks the beginning of a block of universally quantified variables). This play gives *cut* the initiative. *Cut* plays vertices representing the truth values of these universally quantified variables; *short*'s replies are collaborative. *Cut* then plays the next quantifier vertex (which marks the beginning of a block of existentially quantified variables). This play returns the initiative to *short*. Play continues in this way, with *short* choosing vertices for existentially quantified variables and *cut* choosing vertices for universally quantified variables, until the ladder is exhausted. When this happens the root of B is connected to s and the leaves of B representing true clauses each have at least one b grandson connected to t. *Short* wins in the tree if and only if F is true, given the selected truth assignments.

Within this framework of normal play, *short* can win if the formula is true in the following way.

If the next unoccupied foursome in the ladder corresponds to $\exists x_i$, pick a truth value for x_i which makes the formula true and play the corresponding two vertices on successive moves. *Cut*'s replies are forced.

If the next unoccupied foursome corresponds to $\forall x_{i+1}$ and the last occupied foursome corresponds to $\exists x_i$, play q_i. Then reply to *cut*'s moves on foursomes corresponding to the next block of universally quantified variables.

If the entire ladder is occupied, formula F must be true, given the selected truth values. Use the previously described strategy to form a path from the root of B to a leaf l_j which is the root of a three-level tree containing no moves by *cut*. Then win in this three-level tree.

Similarly, *cut* can win if the formula is false, as follows.

If the next unoccupied foursome in the ladder corresponds to $\forall x_i$, pick a truth value for x_i which makes the formula false and play the corresponding two vertices on successive moves.

If the next unoccupied foursome corresponds to $\exists x_{i+1}$ and the last occupied foursome corresponds to $\forall x_i$, play q_i. Then reply to *short*'s moves on foursomes corresponding to the next block of existentially quantified variables.

If the ladder is full, then the chosen truth values must make the formula false. Using the previously described strategy, block all paths in B except one from the root to a leaf l_j representing a false clause. Then beat *short* in the three-level tree rooted at this leaf.

5. *Cut Cannot Win if Formula True*

Assume that the formula is true and *cut* tries to force a win by deviating from normal play. Our purpose is to show that this cannot be done.

If *cut* has the initiative while play is in a universal quantifier block and he does not proceed by playing the next foursome in the normal way, *short* plays the rest of the foursomes in the block, picking truth values for the variables arbitrarily. All *cut*'s responses are forced. Finally, *short* plays the next quantifier vertex and wins. If any one of *cut*'s responses in the block has been played before, the initiative returns to *cut* and the play returns to normal. If *cut*'s nonnormal move is in the next quantifier, then once all foursomes of the block are played the game returns to normal.

The previously described strategy for *short*'s play in the B tree and the three-level trees is sufficient to win even if *cut* deviates from normal play.

6. *Short Cannot Win If Formula False*

Several safeguards have been put in the construction to prevent *short* from forcing a win

by deviating from normal play, when the formula is false. These include vertices l_{k+1} through l_{k+3m} and the three-level trees.

If *short*'s first nonnormal move is made while play is on a universal block, then *cut* breaks the lower or upper part of the ladder (by playing $x_i(j)$ if his previous move was $\bar{x}_i(j)$, and $\bar{x}_i(j)$ if his previous move was $x_i(j)$). If the lower part of the ladder is cut, all paths from s to t are blocked and *cut* has won. If the upper part of the ladder is cut, there are still paths *cut* should block. They all go through some vertices of the tree, and we shall discuss *cut*'s strategy to block them shortly. However, before *cut* can make his first move to do this, *short* has already made two nonnormal moves.

If *short*'s nonnormal move is made while play is on an existential block, *cut*'s strategy is again to try and cut the graph at the lower part of the ladder on the lower two vertices of the next foursome or on the next quantifier vertex (or root of B), if the previously played foursome is the last one in the existential block. If this cannot be done, then *cut* tries to cut the upper part of the ladder on the upper two vertices of the next foursome. *Short* can block such a *cut* strategy by responding to *cut*'s moves in a normal fashion, except that *cut* has the initiative although the block is existential. This continues until either *short* makes another nonnormal move, in which case *cut* achieves one of the goals described above, or until the foursome or the quantifier vertex in which *short* has made his nonnormal move is reached and the play returns to normal, or until the end of the existential block is reached and *cut*, who has the initiative, takes the empty quantifier vertex (which may also be the root of B). In all cases, either play returns to normal, or *cut* wins immediately by breaking the lower part of the ladder, or *cut* breaks the upper part of the ladder. In the last case *short* will have made *three* nonnormal moves before *cut* has a chance to play again.

It remains to be shown that *cut* can block all paths (through tree vertices) in the case where he has cut the upper part of the ladder by playing some $x_i(2)$ and $\bar{x}_i(2)$, even if *short* has made as many as three nonnormal moves in the meantime.

Let us call vertices $y(0, j)$, $y(b, j)$ *special* and decide on *cut*'s strategy according to the following: Has *short* used at least two of his (three) nonnormal moves to play on special vertices? In case he has not, all paths from the upper part of the upper ladder to the rest of the graph are simply cut by complementary play on the $y(0, j)$, $y(b, j)$ pairs. (All paths through edges $(x_k(2), \bar{x}_k(1))$ or $(\bar{x}_k(2), x_k(1))$ for $k < i$ have been cut previously in the course of normal play.) In case *short* has used two of his nonnormal moves to play special vertices, he has made at most one move elsewhere. *Cut* can win by the following strategy:

(1) Respond to *short* moves in the tree to cut B in its root, or elsewhere, or allow a path in B to a dead vertex, i.e. l_{k+i}, or l_{k+m+i}, or l_{k+2m+i}. In the latter case, choose one whose three-level tree has no moves of *short* on it.

(2) Respond to *short* moves on variable vertices by complementary play as follows: If *short* plays $y(k)$, play $\bar{y}(3 - k)$.

(3) Respond to *short* moves on quantifier vertices by complementary play as follows: If *short* plays a universal quantifier, play the preceding existential vertex, and vice versa. Note that the quantifier vertex for a block of variables follows rather than precedes the block in the ladder. The first nonnormal move must have been made when playing an existential block; thus the pairing does not include previously played vertices. If the first nonnormal move has been made when playing a universal block, *short* has made no nonnormal moves other than the two on special vertices, and *cut* takes, on his first move, the next universal quantifier vertex.

7. *Remarks*

It is not hard to define these strategies rigorously and to prove that they work. Thus the Shannon switching game on vertices is log-space complete in polynomial space.

To show that the Shannon switching game on the *edges* of a *directed graph* is complete in polynomial space, we modify the construction above slightly. First we convert each undirected edge in the game graph G into a directed edge leading from s toward t (from

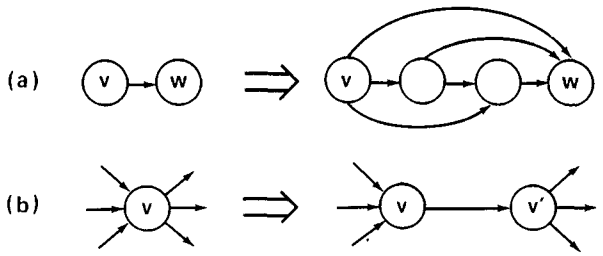

FIG. 6 Modifications for play on edges (a) edge transformation, (b) vertex transformation

bottom to top in Figure 5). Next we replace each edge by the configuration indicated in Figure 6(a). Clearly *short* can force a path from v to w in such a configuration even if *cut* moves first (if play is on the edges). Last we replace each vertex except s and t *in the original graph* G by two vertices joined by a single edge, as indicated by Figure 6(b).

It is not hard to see that *short* wins by playing on the edges in the modified graph if and only if *short* wins by playing on vertices in the original graph. Thus the Shannon switching game on the edges of a directed graph is also log-space complete in polynomial space.

A further question of interest is whether the game problems remain complete in polynomial space even for graphs of restricted degree. By modifying the main construction, we can show that the Shannon switching game on vertices is complete even for undirected graphs all of whose vertices are of degree five or less. We can also show that the Shannon switching game on edges is complete even for directed graphs all of whose vertices have in-degree less than or equal to 3 and out-degree less than or equal to 2, or out-degree less than or equal to 3 and in-degree less than or equal to 2. Since these bounds are probably improvable, we do not include the details of these constructions.

If either game problem has a polynomial-time algorithm, our results imply that *any* problem solvable in polynomial space is solvable in polynomial time. This seems unlikely. The construction we have given also suggests that what makes "games" harder than "puzzles" (e.g. NP-complete problems) is the fact that the initiative (the "move") can shift back and forth between the players. Such a shift corresponds to an alternation of quantifiers in a Boolean formula (the NP-complete problems correspond to Boolean formulas with no quantifier alternation). For any game with a sufficiently rich structure, a construction like that outlined here can probably be made Recently, Schaefer [8] has proved several other combinatorial games complete in polynomial space.

ACKNOWLEDGMENT. The authors would like to thank Prof. Albert R. Meyer of M.I.T. for his helpful comments concerning the use of the quantified formulas.

REFERENCES

1 AHO, A V , HOPCROFT, J E , AND ULLMAN, J D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass , 1974
2 BRUNO, J , AND WEINBERG, L A constructive graph-theoretic solution of the Shannon switching game *IEEE Trans on Circuit Theory CT-17,* 1 (Feb. 1970), 74–81
3. CHASE, S An implemented graph algorithm for winning Shannon switching games. Tech. Rep RC-3121, IBM Thomas T Watson Research Center, Yorktown Heights, N Y , 1970
4 COOK, S The complexity of theorem-proving procedures Proc Third Annual ACM Symp on Theory of Computing, 1971, pp 151–158
5 KARP, R Reducibility among combinatorial problems In *Complexity of Computer Computations,* R E Miller and J. W. Thatcher, Eds., Plenum Press, New York, 1972, pp. 85–104.
6. MEYER, A.R , AND STOCKMEYER, L.J Word problems requiring exponential time. Proc. Fifth Annual ACM Symp on Theory of Computing, 1973, pp. 1–9

7 NILSSON, N *Problem-Solving Methods in Artificial Intelligence* McGraw-Hill, New York, 1971
8 SCHAEFER, T J Complexity of decision problems based on finite two-person perfect-information games Proc 8th Ann ACM Symp on Theory of Computing, 1976, pp 41-49
9. TARJAN, R Depth-first search and linear graph algorithms. *SIAM J Computing 1*, 2 (1972), 146-160
10 TARJAN, R Unpublished notes (1974)

RECEIVED JANUARY 1976, REVISED MARCH 1976