A COMBINATORIAL PROBLEM WHICH IS COMPLETE IN POLYNOMIAL SPACE

Shimon Even
Computer Science Department
Technion, Haifa, Israel

R. Endre Tarjan†
Computer Science Division
Department of Electrical Engineering
and Computer Sciences
University of California, Berkeley, California

ABSTRACT

We consider a generalization, which we call the Shannon switching game on vertices, of a familiar board game called HEX. We show that determining who wins such a game if each player plays perfectly is very hard; in fact, it is as hard as carrying out any polynomial-space-bounded computation. This result suggests that the theory of combinatorial games is difficult.

Keywords: computational complexity, HEX, Shannon switching game, completeness in polynomial space

---

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 a good winning strategy?

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 [1]. These strategies are based on finding a pair of minimally overlapping spanning trees of G, and the fastest algorithm known requires $O(n^2)$ time if G has n vertices [6]. 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 determining who wins the game on an arbitrary graph G is as hard as carrying out any polynomial-space-bounded computation (the game problem is complete in polynomial space).

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 [5]. 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 possible positions reachable by making one move from the position v denotes. We use $v \to w$ 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 $\sum_{i=0}^{n-2} \frac{(n-2)!}{i!}$ 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 $v \to w$ such that W(w) = 1; W(v) = 0 otherwise;

If v is a cut vertex, W(v) = 1 if for all $v \to w$, 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 [5,7]. 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.

---

† Research sponsored by National Science Foundation Grant GJ-35604X and by a Miller Research Fellowship.

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 [2,3].

We shall show that the game problem 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. Stockmeyer and Meyer [4] have exhibited several problems complete in polynomial space, including the

Quantified Boolean Formula Problem:
$QBF = \{Q_1x_1\ Q_2x_2 \ldots Q_mx_mF \mid$ 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 will 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 the 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 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.

Suppose the tree B is such that its root has four grandsons. Consider the four subtrees with these grandsons as roots. Suppose that at least one leaf in each of these four subtrees is not connected to t. Then by an extension of the above idea cut wins even if short makes three moves before the first move by cut. We shall use tree B to represent the conjunction in the formula F.

Consider 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 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 $\{x_1(1),x_1(2),\bar{x}_2(1),\bar{x}_2(2),\bar{x}_3(1),\bar{x}_3(2),\ldots\}$ the play is:

short	$x_1(1)$	$x_1(2)$	$\bar{x}_2(1)$	$\bar{x}_2(2)$	$\bar{x}_3(1)$	$\bar{x}_3(2)\ldots$
cut	$\bar{x}_1(2)$	$\bar{x}_1(1)$	$x_2(2)$	$x_2(1)$	$x_3(2)$	$x_3(1)\ldots$

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	$x_1(1)$	$x_1(2)$	$\bar{x}_2(1)$	$\bar{x}_2(2)$	$x_3(1)$	$x_3(2)\ldots$
short	$\bar{x}_1(1)$	$\bar{x}_1(2)$	$x_2(1)$	$x_2(2)$	$\bar{x}_3(1)$	$x_3(2)\ldots$

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 four extra leaves, which represent dummy clause $(x_m \vee \bar{x}_m)$.

Suppose $(Q_ix_i)F$ 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:

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+4)-1 vertices forming a binary tree B having four grandsons of the root and k+4 leaves, designated by $\ell_1, \ldots \ell_{k+4}$. Let the four subtrees of B determined by the four grandsons of the root each contain exactly one of $\ell_{k+1}, \ldots \ell_{k+4}$;

$\ell_j(0)$, $1 \leq i \leq k$;

$y(0,j), y(a,j), y(b,j)$ for each literal y occuring in clause j, $1 \leq j \leq k$.

Let G have the following edges:

the edges of B;

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

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

$(\ell_j,x_m(2)), (\ell_j,\bar{x}_m(2))$ for $k+1 \leq j \leq k+4$;

$(\ell_j,\ell_j(0))$ for $1 \leq j \leq k$;

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

Figure 4 illustrates this construction for $\exists x\ \forall y\ \exists z((x \vee y) \wedge (y \vee z) \wedge (\bar{x} \vee \bar{z}))$.

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. This is not too hard to see. If both players play well, the game proceeds in the following way:

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 forced. 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.

The four leaves of B corresponding to the dummy clause $(x_m \vee \bar{x}_m)$ are included in the construction so that short cannot win (by playing in the tree before the ladder is completely occupied) if the formula is false. No additional safeguards are needed to prevent cut from winning if the formula is true.

Here is a strategy which wins for short if the formula is true:

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 cut plays outside of these variable vertices or on the next quantifier vertex before all these foursomes are played, play the rest of these foursomes, picking truth values of variables arbitrarily. Then win by playing the next quantifier vertex (if cut has not played it) or continue on the next block of foursomes.

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 dummy leaf (thus winning) or to a leaf ℓ_i which is the root of a three-level tree containing no moves by cut. Then win in this three-level tree.

Here is a strategy which wins for cut if the formula is false:

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 short does not reply to one of these moves (say

$x_i(j)$) by playing $\bar{x}_i(j)$, you play $\bar{x}_i(j)$. Either s is cut from t (if $j = 1$), or the leaves in B corresponding to $(x_m \vee \bar{x}_m)$ cannot now be connected to t (if cut plays correctly). In either case short loses (see below).

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 short plays outside of these variable vertices before all these foursomes are played, play the next quantifier vertex if unoccupied, or the root of B if it is unoccupied and only one block of foursomes is unfilled. In either case short cannot win (henceforth you play $y(3-j)$ each time short plays $y(j)$). If such a move is impossible, play the rest of the block of foursomes, picking truth values of variables arbitrarily. If short does not reply to one of these moves (say $x_i(j)$) by playing $\bar{x}_i(j)$, you play $\bar{x}_i(j)$. Short now loses (see below).

If the ladder is full and you have not played a pair $\{x_i(2), \bar{x}_i(2)\}$, 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 representing a false clause. Then beat short in the three-level tree rooted at this leaf.

If you have played a pair $\{x_i(2), \bar{x}_i(2)\}$, short has in the meantime (possibly) played the root of B plus at most two more vertices (in B, the three-level trees, and the ladder). You win as follows:

(i) If short plays in B (or if he has just made three moves in B) you respond using the previously described strategy, eventually blocking all paths in B except one from the root to some dummy leaf.

(ii) If short has just made at most one move in the ladder (say $y(j)$) respond by playing $y(3-j)$. Henceforth respond $y(3-j)$ each time short makes a move of the form $y(j)$.

(iii) If short has just made two moves in the ladder, henceforth respond $y(0,j)$ each time short makes a move of the form $y(b,j)$; respond $y(b,j)$ each time short makes a move of the form $y(0,j)$.

This strategy will guarantee that t is connected only to vertices above $x_i(2), \bar{x}_i(2)$ in the ladder and to leaves in the three-level trees. Thus short loses.

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

If the game problem has a polynomial-time algorithm, our result implies 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.

REFERENCES

[1] S. Chase, "An implemented graph algorithm for winning Shannon switching games," IBM Research Technical Report RC-3121, Yorktown Heights, New York (1970).

[2] S. Cook, "The complexity of theorem-proving procedures," Proceedings Third Annual ACM Symposium on Theory of Computing (1971), 151-158.

[3] R. Karp, "Reducibility among combinatorial problems," Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York (1972), 85-104.

[4] A.R. Meyer and L.J. Stockmeyer, "Words problems requiring exponential time," Proceedings Fifth Annual ACM Symposium on Theory of Computing (1973), 1-9.

[5] N. Nilsson, Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, New York (1971).

[6] R. Tarjan, unpublished notes (1974).

[7] R. Tarjan, "Depth-first search and linear graph algorithms," SIAM J. Comput., Vol. 1, No. 2 (1972), 146-160.

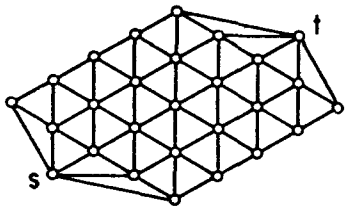

FIGURE 1: k x k Hex for k = 5.

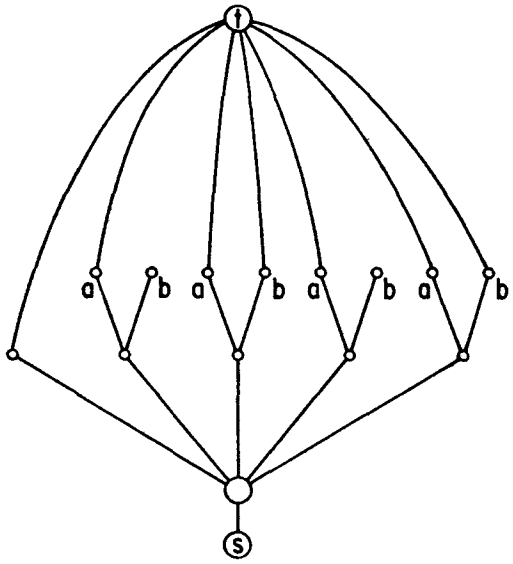

FIGURE 3: A three-level tree representing a conjunction.

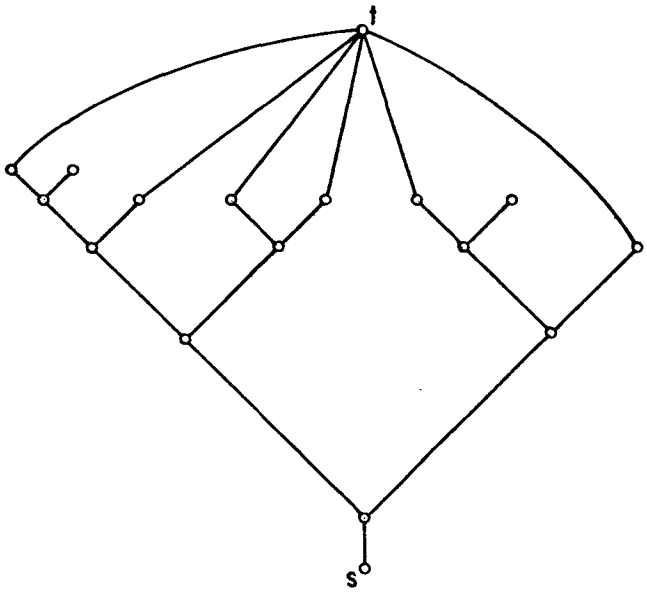

FIGURE 2: A binary tree representing a conjunction.

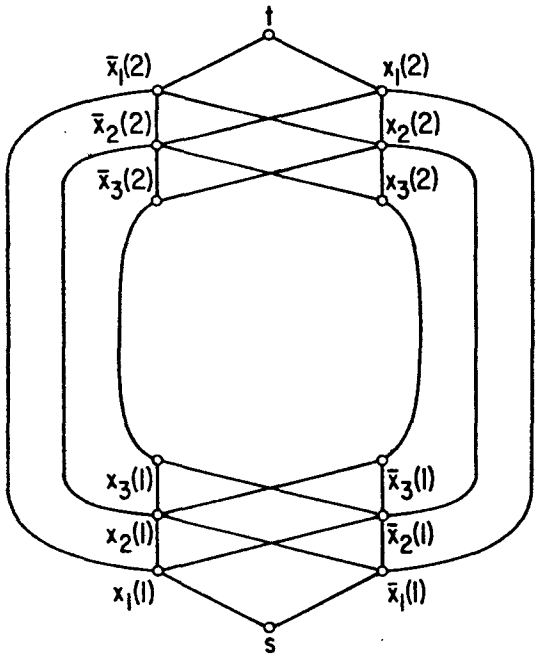

FIGURE 4: A ladder.

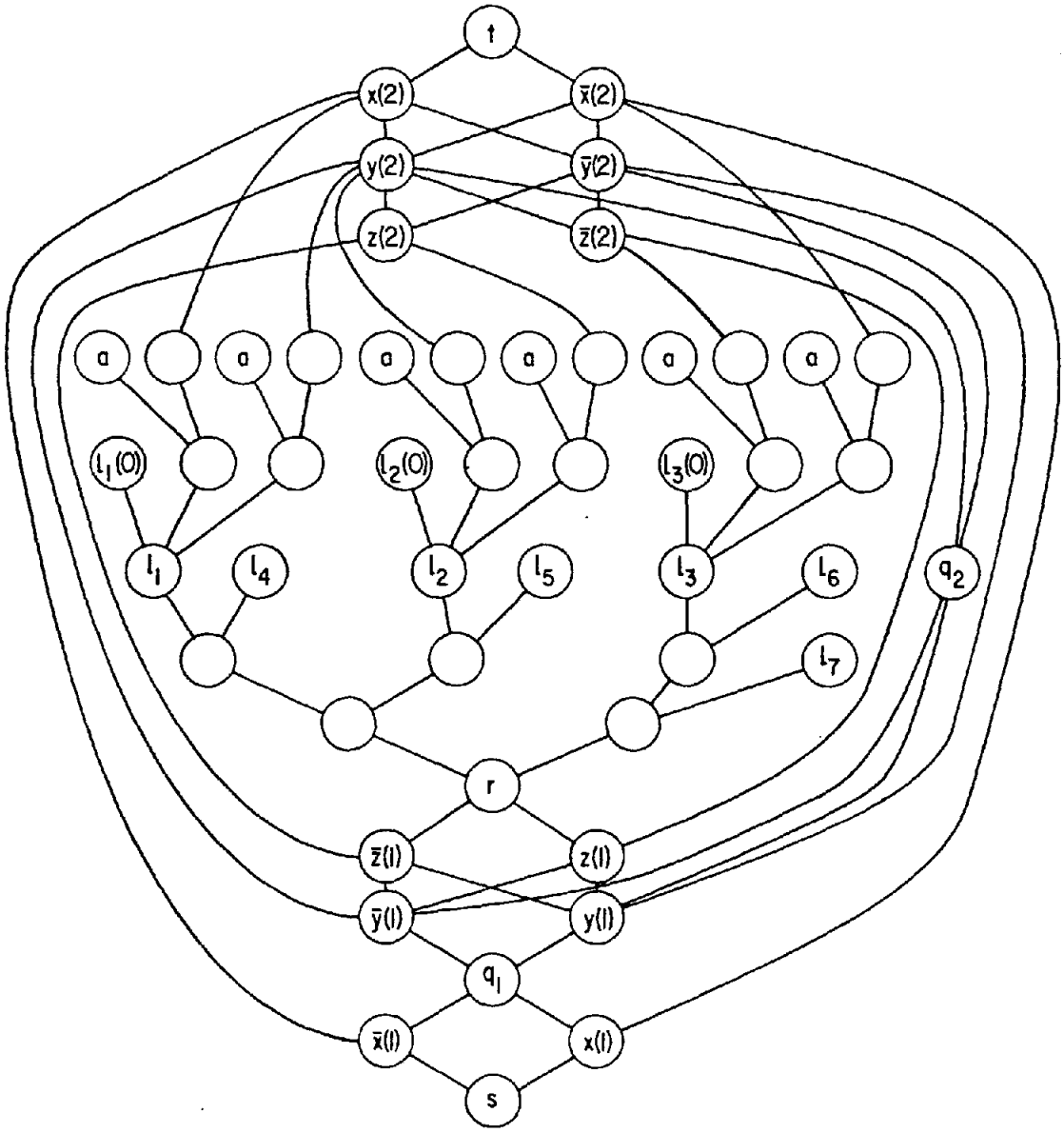

FIGURE 5: Game graph for $\exists x\, \forall y\, \exists z[(x \vee y) \wedge (y \vee z) \wedge (\bar{x} \vee \bar{z})]$.
For clarity, the following corrections are left out:

$$\{z(2), \bar{z}(2)\} \times \{\ell_1(0), \ell_2(0), \ell_3(0), \ell_4, \ell_5, \ell_6, \ell_7, a\}.$$