COMBINATORIAL GAMES WITH EXPONENTIAL SPACE COMPLETE DECISION PROBLEMS

J. M. Robson
Australian National University
Canberra, ACT, Australia

1. INTRODUCTION

In recent years, several results have been proved showing that, for a variety of games, it is an exponential time complete problem to decide, given an arbitrary configuration, whether a given player can force a win. The games include a number of artificial 'boolean formula' games [2,10] as well as generalisations to N x N boards of chess [3], checkers [9] and go [8].

The lower bound parts of these proofs have been obtained by reductions from the problem of deciding the outcome of a computation of a linear space bounded Alternating Turing Machine. [2] showed close relationships between Alternating and Deterministic Turing Machine complexities and started the reductions by giving a reduction to a boolean formula game called G_1 described later in this paper. The upper bound parts are trivially based on a 'brute force' method for solving such game problems in exponential time by classifying all configurations as won, lost or drawn and recording this information in a table.

Since the number of distinct configurations which can arise in these games is exponential in the input length (board size or formula length), complexity greater than exponential time will only arise if the outcome from a given configuration depends on the way in which the configuration arose. This happens in chess but only in unimportant ways (rules concerning castling, en passant capture and drawing rules). In the Chinese rules of go, there is a much more significant rule relating the outcome to past play; this rule forbidding a player to play so as to return to any configuration which has occurred previously is a generalisation of the Japanese ko rule. This rule means that the exponential time completeness result referred to above applies only to the Japanese version of go since the 'no-repetitions' rule invalidates both upper and lower bound parts of that proof. The realisation of this fact triggered the investigations reported in this paper which showed that this 'no-repetitions' rule in fact makes some games exponential space complete. It is not yet known whether this fact applies to the Chinese version of go.

Another way in which games with only an exponential number of physical configurations may still be harder than exponential time has been studied in [6,7]. It is shown there that games where the players have imperfect information of the physical configuration may be exponential space complete or harder depending on the number of players and the amount of hidden information. A survey of recent results in complexity of games appears in [4].

2. STATEMENT OF RESULTS

The following description of G_1 is paraphrased from [10]. 'G_1: a position is a triple $(\tau, F(X, Y, \{t\}), \alpha)$ where $\tau \in \{I, II\}$ indicates which player is about to move, F is a boolean formula in 4CNF whose variables have been partitioned into disjoint sets X, Y and {t} and α is an assignment of values to the variables of F. Player I moves by setting t to true, and setting the variables in X to any values (thus producing a new position with τ changed from I to II, F unchanged and α changed according to the new values of X and t); player II moves by setting t to false and setting the variables in Y to any values. A player loses if the formula F is false after his move'.

The major result of this paper is:

Theorem 1

If G_1 is modified to G_1' by introducing the no-repetition rule: 'A player loses if, after his move, $\exists$ some earlier move M such that all variables have the same values as after M.', then deciding, for an arbitrary G_1' configuration, whether player I can force a win is complete in exponential space. The proof of this theorem also shows with a little extra work:

Theorem 2

If G_1 is modified to G_1^* by introducing the conditional no-repetition rule: 'Two special variables $x \in X$ and $y \in Y$ are specified. A player loses if, after his move, $\exists$ some earlier move M such that all variables have the same values as after M and at most one of x and y has been changed since M.', then deciding for an arbitrary G_1^* configuration, whether player I can force a win is complete in alternating exponential space.

Theorem 2 is equivalent to asserting that the decision problem for G_1^* is complete for double exponential time [2]. The lower bound parts of these proofs are described in the next section. The upper bound parts are relatively trivial. For G_1' a simple minimax algorithm runs in exponential space; for G_1^* a double exponential time decision procedure is obtained by applying the brute force method mentioned in the introduction to a game whose configurations are sequences of G_1^* configurations starting with a change of a special variable and not including either another change of that variable or a repeated G_1^* configuration.

3. THE LOWER BOUNDS

3.1 Overview

The lower bound parts of theorems 1 and 2 will be proved by two similar reductions, one from exponential time bounded Alternating Turing Machine (ATM) computations to G_1' and the other from exponential space bounded ATM computations to G_1^*. The result of [2], relating time f on an ATM to space f on a deterministic TM, completes the proof of theorem 1.

To simplify the presentation of the reductions, they will be split into three stages; (i) from a general ATM to a very restricted ATM, (ii) from the restricted ATM to games G_0' and G_0^*

(the variants of a new game G_0 even more basic than G_1) and (iii) from G_0 to G_1. It is hoped that this makes the critical central step from an ATM to a game as easy to understand as possible, though this approach does not give the simplest overall reduction.

The restrictions on the ATM are as follows:

(1) At each step the ATM will switch from an existential to a universal state or vice versa.

(2) The ATM has a single tape and a binary alphabet.

(3) The ATM starts at the left hand of its input, moves steadily right writing special markers at the left hand end of the tape and at some distance to the right and then never leaves the section of tape between the two markers. The states which may be used in this initial scan of the tape can never again be entered after it.

(4) The ATM's remaining states are partitioned into 'left moving' and 'right moving'. Once in a left (right) moving state the machine continues moving steadily left (right) until it encounters the marker at the left (right) end of the section of tape marked out for the computation which forces the ATM to enter one of two special states in which it must switch into a right (left) moving state.

The effect of restrictions (3) and (4) is that a computation of the ATM consists of alternate left and right scans, each of which covers the same section of tape, and that each tape cell in this section is either always scanned in an existential state or always scanned in a universal state. These restrictions will considerably simplify the expression F; it is fairly straightforward to show that an arbitrary ATM may be replaced by one subject to these restrictions and accepting the same inputs with at most a polynomial increase in its time and space complexity.

The new game G_0 differs from G_1 in that whether a move is legitimate depends on the values of the variables before the move as well as after. More rigorously, the formula F which determines whether a player has lost after his move may include, as well as X, Y and t, X* and Y* indicating the values of the corresponding variables before the move. The reduction from G_0 to G_1 will be described in section 3.4.

3.2 The Game Graph

3.2.1 The variables X and Y

The idea behind the reduction to $G_0^{(*)}$ is exactly the same as that of the reduction to G_1 in [10], namely that in each move of the game the players are expected to set their variables in a way which represents one step of the ATM; player I makes the moves corresponding to existential states and player II makes the moves corresponding to universal states; the formula F is designed so that (i) a player making a move not representing a possible ATM step loses very quickly, (ii) if the ATM computation reaches an accepting state, I wins and (iii) if the ATM computation reaches a rejecting state, II wins. Thus player I can force a win if and only if the ATM accepts its input.

In our case the variables are not sufficient in number to completely encode the current

ATM instantaneous description. Thus the formula F cannot ensure that a player loses instantaneously if he sets his variables to values inconsistent with the symbols written on the tape by the previous computation. Instead, if this happens, the opposing player can set in train a sequence of moves which quickly results in the transgressing player losing by the no-repetitions rule. The way in which this happens will be described first by showing a subgraph of the game graph corresponding to a single ATM step. Section 3.3 will describe the formula F which forces players to play according to this game graph.

The variables in each of the sets X and Y are as follows (subscripts X and Y will be used when necessary to distinguish them):

q_i	$0 \leq i <$ log (number of ATM states)
p_i	$0 \leq i <$ log (upper bound on space used by ATM)
S, A1, A2, A3, C, R	
N	(in the G_0^* case)
n_i	$0 \leq i <$ log (upper bound on number of times ATM traverses its tape) (in the G_0' case)

The intended interpretation of these is: q_i are the bits of an ATM state number called Q; p_i are the bits of P, a position or address on the ATM tape; S is an ATM symbol; A(1,2,3) C, R stand for Accept(1,2,3), Claim, Refute; n_i (at G_0') are the bits of N, the number of times the ATM has switched from a left moving to a right moving state or vice versa; N (at G_0^*) : N_X and N_Y are the special variables which are mentioned in the conditional no-repetition rule of G_0^*, N_X the parity of the number of times the machine has switched from a left moving to a right moving state and N_Y the parity of the number of switches from right to left moving. A quadruple (N,Q,P,S) is interpreted as describing a single step of the ATM in which the symbol S is written and N,Q and P have the appropriate values after the step.

3.2.2 Moves at G_0'

At G_0' corresponding to an instant when the ATM has just made a universal (existential) step, is a G_0' configuration where both (N_X, Q_X, P_X, S_X) and (N_Y, Q_Y, P_Y, S_Y) describe this step and exactly one of the remaining variables in X and Y is true, namely $A3_{X(Y)}$. The game proceeds to the configuration corresponding to the instant one step later in the computation by a sequence of five moves illustrated in the left part of figure 1: first I clears $A3_X$, sets C_X and sets (N_X, Q_X, P_X, S_X) to what he CLAIMs are the correct new values (N', Q', P', S'); then II, provided he ACCEPTs the CLAIM, sets $A1_Y$ and sets (N_Y, Q_Y, P_Y, S_Y) to the same new values; then I clears C_X and sets $A2_X$; II clears $A1_Y$ and sets $A3_Y$ then I clears $A2_X$ completing the sequence. The formula F is designed so that a player loses instantaneously, if he deviates from this pattern except in one of the two ways shown in figure 1.

The lower possible deviation (by I) is fatal, not instantaneously but very quickly: I can set R_X when he clears $A2_X$; II then clears $A3_Y$ and sets $A1_Y$; the formula F would then allow I only one move, namely to clear R_X and set $A2_X$, but the no-repetitions rule ensures that I cannot do this and so loses.

The upper deviation (by II) is more interesting. This deviation is fatal for II if I's CLAIM

represented by (N', Q', P', S') is consistent with the previous computation of the ATM but is fatal for I otherwise. This fact is the crux of the correctness of the reduction.

(i) II REFUTEs I's CLAIM by setting (N_Y, Q_Y, P_Y, S_Y) to the description (N'', Q'', P'', S'') of the last step which wrote a symbol (S'') in the position on the tape read on this current step. F will allow II to do this if and only if the step (N'', Q'', P'', S'') would have made (N', Q', P', S') impossible, that is if N'' and P'' do describe the last step which wrote in the tape cell read by the supposed move (N', Q', P', S') and the supposed ATM transition from reading S'' in state Q to writing S' and changing to state Q' is not possible.

(ii) I's only possible move is now to set all the X variables to the same values as the corresponding Y variables.

(iii) II's only possible move is now to clear R_Y and set $A1_Y$. Thus the game has arrived at a configuration exactly like that already considered on the lower deviation but with the step described (N'', Q'', P'', S'') chosen by II instead of (N', Q', P', S') claimed by I.

If the step (N'', Q'', P'', S'') really did occur earlier in the computation, then I now loses by the no-repetition rule. Otherwise I clears R_X and sets $A2_X$; II is forced to clear $A1_Y$ and set $A3_Y$; I returns to the deviation by clearing $A2_X$ and setting R_X; II now loses by the no-repetition rule.

Thus the illustrated graph of possible moves and deviations ensures exactly as required that II can win by taking the deviation if I makes a false CLAIM but otherwise a player taking a deviation loses.

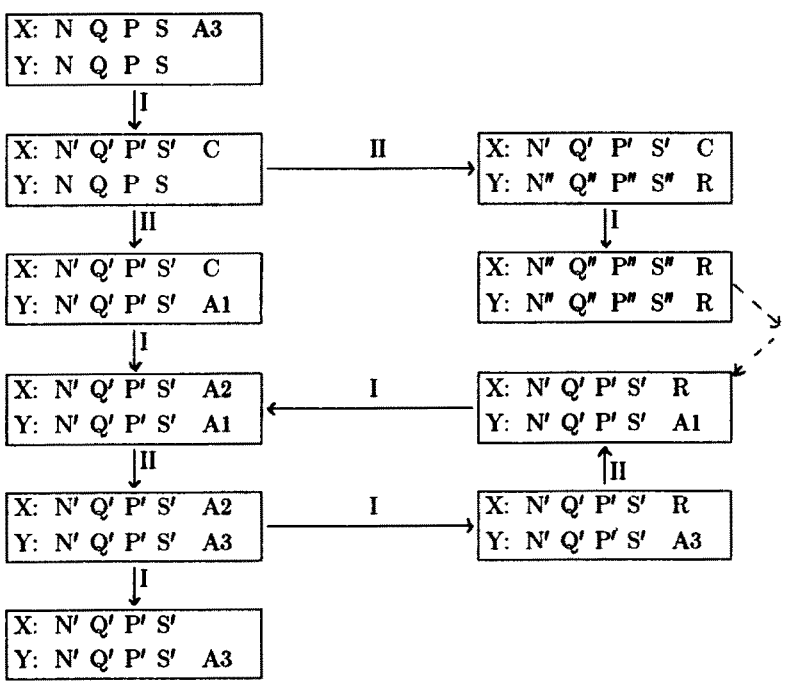

Figure 1 An existential step at G'_0

The graph of all possible moves at G_0' is obtained by taking copies of figure 1 for every possible CLAIM by I and every attempted REFUTATION by II, adding also copies of the corresponding figure for universal steps and finally identifying all nodes with equal values for all variables (including the turn variable t which is implicit in the figure).

3.2.3 Differences at G_0^*

Figure 2 illustrates the G_0^* moves corresponding to an ATM step together with possible deviations exactly as in figure 1. The discussion is identical except for the treatment of the special variables N_X and N_Y. N_X and N_Y are no longer required to be equal in the first and last configurations; instead I changes N_X or II changes N_Y at the appropriate end of the tape used. On the upper deviation II sets N_Y to the correct value for the step with which he tries to REFUTE I's CLAIM; then I is required to set N_X to the appropriate value (note that N_X is implied by N_Y and the knowledge of whether Q is a left or right moving state). Between two adjacent occasions when a tape cell is scanned, exactly one of N_X and N_Y changes whereas between two non-adjacent occasions, both N_X and N_Y change at least once. Thus with the conditional no-repetition rule of G_0^*, N_X and N_Y ensure that a refutation succeeds only if it relies on the last symbol written in the relevant tape cell, avoiding the need, as in G_0', to have enough n_i variables to count scans over the tape.

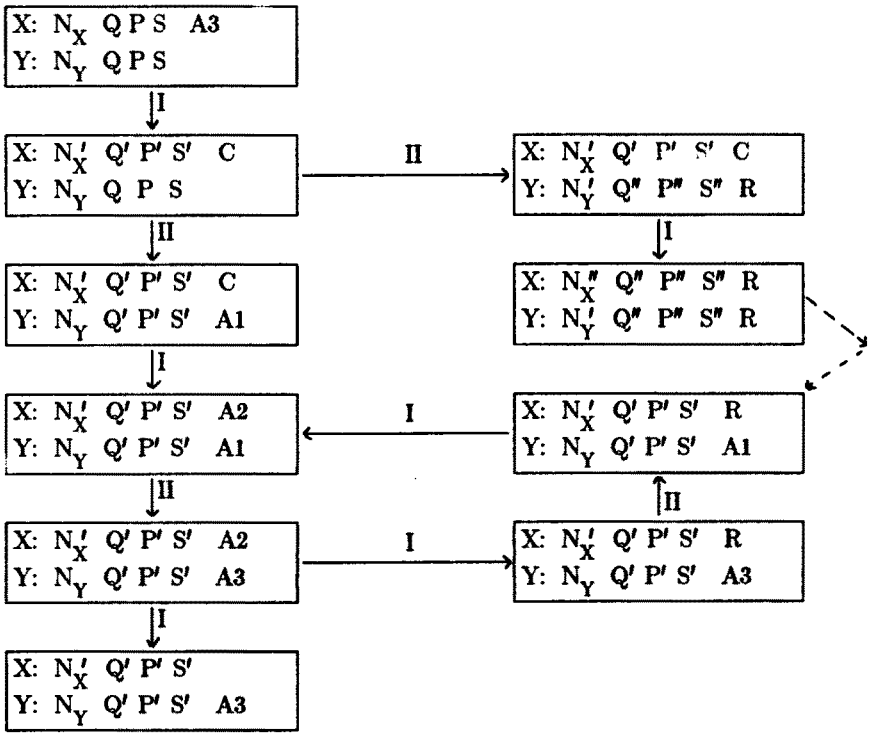

Figure 2 An existential step at G_0^*

3.3 The Boolean Formula

3.3.1 Semantics

The boolean formula F over the variables X, Y, t (in fact t is not needed in G_0), and X* and Y* is required mainly to ensure that any move not corresponding to the informal description above loses at once. (F must also ensure that the symbols read on the first scan of the tape do correspond to the input, that I loses if the computation rejects and that II loses if it accepts. These are trivial given the restriction (3) on the ATM and are not discussed further.) We distinguish between variables indicating the 'type' of a configuration, namely A(1,2,3) C and R and those imparting information, about the progress of a computation (N,Q,P,S).

Figure 1 (or 2) contains eleven arcs so that with the figure for universal steps, there are 22 types of move. With each move type i ($1 \leq i \leq 22$) we associate a type condition, namely an expression T_i indicating that the type variables before and after the move are correct for that type, and a semantic condition S_i. Then F can be written as $\forall i(S_i \vee \neg T_i) \wedge \exists i(T_i)$.

For nine of the eleven move types in figure 1, the semantic condition is simply, as indicated in the diagram, that $N_X=N_Y \wedge Q_X=Q_Y \wedge P_X=P_Y \wedge S_X=S_Y$.

For the tenth move type, where I sets C, the semantic condition simply states that the new values N_X and P_X are correctly related to the old ones N_Y and P_Y. Since, by the restrictions on the ATM, Q_Y determines whether the ATM is moving left or right and whether it is about to reverse direction, this can be written, for some expressions $f_i(Q_Y)$ $1 \leq i \leq 4$, as $(f_1(Q_Y) \vee N_X=N_Y) \wedge (f_2(Q_Y) \vee N_X=N_Y+1) \wedge (f_3(Q_Y) \vee P_X=P_Y+1) \wedge (f_4(Q_Y) \vee P_X=P_Y-1)$. An extra conjunct $f_5(Q_X,Q_Y)$ ensures correct state changes at ATM direction changes.

For the last move type, where II sets R attempting to refute I's claim by setting new values N" etc., there are two semantic conditions: firstly N_Y and P_Y must refer to the previous occasion when this cell was scanned and secondly in state Q_Y^* scanning symbol S_Y the machine does not move into state Q_X and write symbol S_X. These can be written respectively as $(N_Y = N_X-1) \wedge (f_6(Q_Y^*) \vee P_Y = P_Y^*+2) \wedge (f_7(Q_Y^*) \vee P_Y = P_Y^*-2)$ and as $f_8(S_Y, Q_Y^*, S_X, Q_X)$.

For the eleven other move types, in the corresponding figure for universal steps, the conditions are obtained by interchanging subscripts X and Y.

3.3.2 The CNF form

To convert the expression F described above into CNF, we first convert various subexpressions to CNF. For some expressions such as the type conditions and the f_i whose exact form was not discussed, this clearly gives expressions whose length depends only on the ATM and not on the computation length or space. For the remaining arithmetic subexpressions, $N_X = N_Y$, for instance, is equivalent to $\forall i\ N_{X,i} = N_{Y,i}$ and $P_X = P_Y + 1$ is equivalent to $(P_{X,0} \neq P_{Y,0}) \wedge \forall(i>0)\{(P_{X,i}=P_{Y,i}) \blacksquare (P_{X,i-1} \vee \neg P_{Y,i-1})\}$. Next writing these simpler subexpressions involving only 0(1) variables in CNF and applying the distributive law, we obtain a CNF expression F whose length is bounded by a linear function of the number of variables (the

linear function depending on the ATM).

Now F is a conjunction of disjunctions each of the form $E_1(X) \vee E_2(Y)$ or equivalently $(X\# = E_1(X)) \wedge (Y\# = E_2(Y)) \wedge (X\# \vee Y\#)$. Using the complement of the method of [1], a subexpression such as $(X\# = E_1(X))$ can be replaced by a 3CNF form $E_1'(X, X\#, X')$ which has a satisfying X' assignment iff the original subexpression was true. Adding the X# and X' variables into the set X and similarly for Y, gives a form of the expression F which is in 3CNF and has a length linear in the number of variables introduced in section 3.2.1. This completes the polynomial reduction from the original ATM problems to G_0' and G_0^*.

3.4 From G_0 to G_1

The reduction from G_0 to G_1 is most easily explained as a modification to the game graph described in section 3.2. The translation from the modified graph to the CNF form of F is then very similar to that described in section 3.3. The modification gives each player extra variables which can be used to record the state of the G_0 variables before the move as well as after and also three new 'timing' variables t^0, t^1 and t^2; at any time exactly one of these will be true and will indicate the number of moves made by the player (mod 3).

Figure 3 illustrates, for a move by player I, how a single arc in the G_0 game graph becomes a path of length 3 in the G_1 graph. Recall that the G_0 expression could use X* and Y* to refer to the values of the variables before the move. Initially I has two sets of copies of the initial G_0 variables X* and one set of copies of Y*; II has two sets of copies of Y* and one of X*. I moves by changing one of his copies of X* to the new value X; II responds by changing his single copy of X* to X and I finally changes his other copy. Note that if the G_0 move loses by causing a repetition, the repetition at G_1 occurs on I's second move so that the responsibility for the repetition remains correctly ascribed to I. In G_1^* the special variables are those corresponding to the G_0^* special variables in the second copy of the G_0^* variable sets; that is they are the ones changed in the third move of the sequence described.

Now the type conditions for a position in G_1^* can be written in a form like $t^2_X \wedge t^1_Y \wedge \neg t^0_X \wedge \neg t^1_X \wedge \neg t^0_Y \wedge \neg t^2_Y \wedge t$ for the last position in figure 3. (Remember t is the turn variable always set true by I).

The semantic conditions for the various positions consist of the obvious conditions that various copies of variable sets are in fact equal and the clauses inherited from G_0^* F(X[copy 1], X[copy 2], Y[copy 1], Y[copy 2]) in the two positions where one player has taken the first move of the three move sequence.

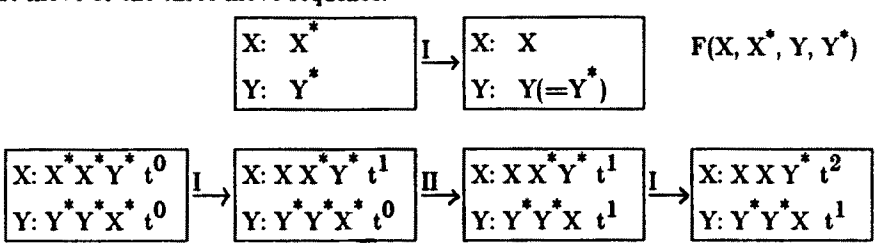

Figure 3 The reduction from G_0 to G_1

4. CONCLUSIONS

The reductions described in section 3, and the theorems of [2] on the relationships between deterministic and alternating time and space complexities complete the proof of the theorems 1 and 2.

Theorem 1 would apply also to the modified version X' of any other game X, if there were a polynomial time reduction from G_1 to X satisfying a few simple conditions which force players at X' to mimic play at G_1'.

In fact such reductions would exist using the already published reductions to G_2, G_3 [10], chess [3] and checkers [9] except for one minor snag: the original reduction from G_1 to G_2 produces multiple G_2 configurations for one at G_1. However this problem is easily removed (for each variable in the original set X, the reduction produces four variables $x_{i,1}$, $x_{i,2}$, $y_{i,1}$, and $y_{i,2}$; when player I changes $x_{i,1}$ or $x_{i,2}$, player II must change one of $y_{i,1}$ or $y_{i,2}$ but it is immaterial which he chooses; removing $y_{i,2}$ and requiring player II to change $y_{i,1}$ leaves a valid reduction; similar comments apply to the variables $x_{i,2}$ corresponding to original Y variables), establishing exponential space completeness for G_2', G_3', chess' and checkers'.

The list of games whose ' version is known to be exponential space complete does not contain go whose version go' as played in China started this investigation. The reductions to go from G_3 make it easy for one player to force the other to break the no-repetition rule.

Finally it is of interest to note that instances of X' for any game X are simply instances of the 'generalised geography game' [5], which is known to be complete in polynomial space. Thus the exponential space completeness results could be interpreted as implying that the geography game is still very hard (requiring space $\Omega(e^{\log^c x})$ for $c > 0$) even when restricted to apparently simple instances which can be described in logarithmic space and even when this short description is provided as part of the input.

REFERENCES

[1] Bauer M., Brand D., Fischer M., Meyer A. and Paterson M., A note on disjunctive form tautologies, SIGACT news, April 1973, pp 17-20.

[2] Chandra A.K., Kozen D.A. and Stockmeyer L.J., Alternation, J. ACM, vol. 28, 1981, pp 114-133.

[3] Fraenkel A.S. and Lichtenstein D., Computing a perfect strategy for n x n chess requires time exponential in n, J. Combinatorial Theory series A, vol. 31, 1981, pp 199-213.

[4] Johnson D.S., The NP-Completeness Column: an ongoing guide, Journal of Algorithms, vol. 4, 1983, pp 397-411.

[5] Lichtenstein D. and Sipser M., Go is polynomial-space hard, J. ACM, vol. 27, 1980, pp 393-401.

[6] Reif J.H. and Peterson G.L., Multiple person alternation, in Proceedings of 20th annual symposium on foundations of computer science, IEEE Computer Society, 1979, pp 348-363.

[7] Reif J.H., Universal games of incomplete information, in Proceedings of 11th annual ACM symposium of theory of computing, ACM, 1979, pp 288-308.

[8] Robson J.M., The complexity of go, in Information Processing 83, R.E.A. Mason ed., IFIP, 1983, pp 413-418.

[9] Robson J.M., N by N checkers is exptime complete, SIAM J. Comput., to appear.

[10] Stockmeyer L.J. and Chandra A.K., Provably difficult combinatorial games, SIAM J. Comput, vol. 8, 1979, pp 151-174.