SIAM J. COMPUT.
Vol. 13, No. 2, May 1984

003

N BY N CHECKERS IS EXPTIME COMPLETE*

J. M. ROBSON†

Abstract. The game of Checkers can easily be generalized to be played on an N by N board and the complexity of deciding questions about positions regarded as a function of N. This paper considers mainly the question of whether a particular player can force a win from a given position and also the question of what is the best move in a given position. Each of these problems is shown to be complete in exponential time. This means that any algorithm to solve them must take time which rises exponentially with respect to some power of N and moreover that they are amongst the hardest problems with such a time bound. For instance if there are any problems solvable in exponential time but not in polynomial space, then these two problems are amongst them.

Key words. checkers, two person game, exponential time complete

1. Introduction.

1.1. N by N Checkers. This paper considers the complexity of algorithms to solve certain problems concerning the game of Checkers generalized to an N by N board. It is assumed that the reader is familiar with the rules of 8 by 8 Checkers. The generalization to an N by N board is fairly obvious except for the initial disposition of the pieces which does not concern us. In case the generalization is not obvious, [1] discusses the interesting points fully. It is assumed here that there is no rule which will cause a game to be declared a draw when, apart from this rule, one player could eventually force a win.

The forced capture rule will be of vital importance in the analysis and so it is repeated here: a player who has any capture move available on his/her turn must make one of the available capture moves and such a move is not complete until the capturing piece reaches a square from which no further capture is possible for it. Thus a player may be obliged to capture an indefinite number of pieces on a single move but if several captures are available, there is no obligation to choose the one which captures the most pieces.

1.2. Exptime completeness. The terminology of this paper is taken from [6] which contains a discussion of many related topics as well as the foundations of our arguments. The relevant definitions are:

A language is in exponential time (abbreviated Exptime) if it is recognized by a deterministic Turing machine with running time bounded by $O(c^{p(n)})$ for some c and p a polynomial where n is the length of the input string.

A language is log-complete in exponential time (abbreviated exponential time complete or Exptime complete) if it is in Exptime and every other language in Exptime is reducible to it by a Turing machine using $O(\log(n))$ workspace on inputs of length n. Since some languages are already known to be Exptime complete, to show that a language L is also Exptime complete, it suffices to show that L is in Exptime and that there is a reduction to L from some known Exptime complete language.

If a language L is shown to be Exptime complete, the principal conclusion which follows is that any deterministic algorithm to recognize L must take time $\Omega(c^{n^k})$ for some $c>1$ and $k>0$ since it is known that there are some languages in Exptime with this property. Moreover if it were shown, as most researchers would suspect, that

* Received by the editors August 25, 1981, and in revised form August 20, 1982.
† Department of Computer Science, Australian National University, Canberra, ACT 2600, Australia.

there are languages in Exptime which are not recognizable in polynomial space, then the same would be true of L.

1.3. Results. The main result which will be proved is that the set of all N by N Checkers positions from which White (say) can force a win is Exptime complete. This appears to be a considerable strengthening of the result of [1] that recognizing this set is P space hard.

Since the number of possible positions at N by N Checkers is easily seen to be $<5^{N^2}$, the proof that the language is in Exptime is simple and is omitted. Thus the bulk of the paper is devoted to the description of a reduction from a known Exptime complete language to Checkers positions. Finally §§ 8 and 9 will justify the claim that the reduction is logspace computable and draw some conclusions.

Similar Exptime completeness results are already known for Chess [2] and various artificial games [3], [6]. A proof of the same result for GO is currently in preparation, again strengthening a previous P space hardness result [4].

2. A global view of the Checkers positions. The overall form of the Checkers positions which will be of interest is illustrated in Fig. 1. In the centre a group A consisting of $|A|$ pieces is arranged in a configuration to be described in detail later so that moves there constitute a simulation of a game already known to be Exptime complete. Surrounding A and very distant from it is a spiral whose arms consist of parallel rows of Black and White kings four squares apart. The form of the spiral is shown in more detail in Fig. 2. (Squares denote kings and circles denote single pieces in most of the figures.) The number of circuits of the spiral and the distance from A to the inside of the spiral are $O(|A|)$ and $O(|A|^2)$ respectively. The inside of each arm of the spiral is four squares from the outside of the previous one and each arm has an even number of pieces of each color.

Consider what would happen if one player (Black say) was suddenly left with a large number of possible capture moves in A. White could capture all the Black pieces in the spiral in a small number of moves (independent of the length of the spiral) by

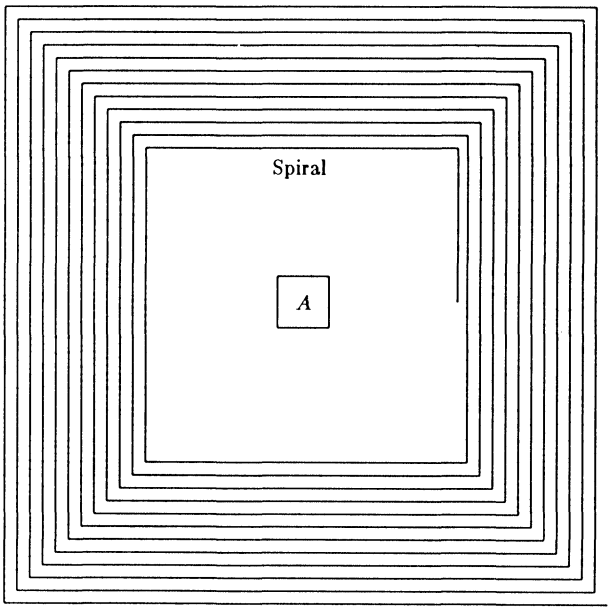


FIG. 1

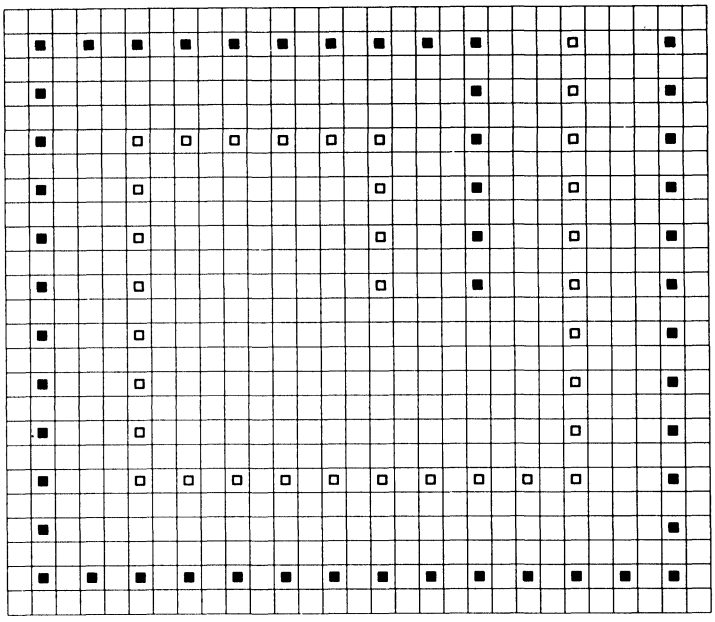


FIG. 2. *The start of the spiral.*

taking advantage of Black's being bound by the forced capture rule. Figure 3 illustrates one way in which White could do this. Having set up the pattern shown, White moves from A to B and then on his next move, whether or not Black captures the piece at B, White starts capturing along the path marked by Ds. This captures all the Black pieces in the spiral apart from one or two of those marked C which can then be mopped up in three or four more moves.

Thus White will have obtained a massive material advantage which we claim will be enough to enable him to force a win. Of course in general no material advantage guarantees a win if the pieces are badly placed but [1] showed that if a group of pieces like A is surrounded by $O(|A|)$ "picket lines" that is rectangles like those shown in Fig. 4 with at least four squares between them, then White can force a win whatever the details of A. Since converting White's spiral to a set of $O(|A|)$ picket lines involves moving $O(|A|)$ pieces, White can complete this process by moving pieces from the outer arms of the spiral in $O(|A|^2)$ moves before any Black pieces from A can approach and interfere with the process.

Of course Black will realize that, if he is left with a number of forced captures, White can force a win in this way. Accordingly Black may try to frustrate the plan by making moves in the spiral. But even if Black makes up to two moves in the spiral, White can still capture all Black's pieces in the spiral in a certain finite number of moves. This is because Black's moves merely split his spiral into three vulnerable sections each of which can be attacked and destroyed as in Fig. 3 and the bounded number of Black pieces remaining can then be mopped up individually. Instead of trying to determine the minimum number of moves which White may ever need after any pair of moves by Black, we simply call it X, and conclude that White can force a win provided, by play in A, he can force Black to make a sequence of X captures in A (with no White captures forced in A) before Black is able to make more than two noncapture moves outside A. In exactly the same way, Black can force a win if

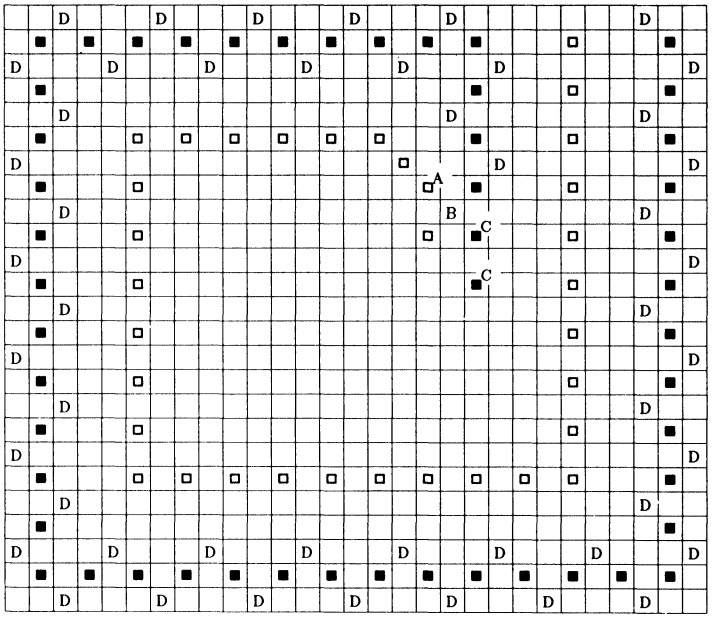


FIG. 3. *Attacking the spiral.*

White is forced to make the X captures in A. The structure of A will be such that there is no other way of forcing a win.

3. A known Exptime complete game. The reduction to Checkers starts from the game G3 shown in [6] to have an Exptime complete decision problem. The notation of that paper has been changed so that the names are more meaningful in the context of Checkers.

G3 is played by changing values of boolean variables W_1 to W_m and B_1 to B_m. Player W (or B) moves by changing the value of exactly one of the W (B) variables. Moves alternate between the two players and the result of a game is decided by two boolean expressions WWIN and BWIN in DNF over the variables. W wins immediately if his expression WWIN is ever true when B has just moved (and vice versa).

The aim of the reduction is to produce, given an instance of G3, a configuration to put at A in Fig. 1, which models the structure of the instance of G3 and is such that certain moves (called "normal play") simulate the playing of G3 and, if W wins at G3, then White can force X Black captures in A (and vice versa).

However, there is a complication in that a player may choose to "cheat" by making a legitimate Checkers move either in A or in the spiral which is not a normal simulation move. The solution to this is to modify the expressions WWIN and BWIN so that a player who "cheats" by not changing one of his variables on his move loses at once. In other words we must ensure than a normal W move always leaves WWIN true and a normal B move always leaves BWIN true. It is not obvious that this can be done for arbitrary WWIN and BWIN without increasing their length (when expressed in DNF) exponentially.

Fortunately, for the particular instances of G3 reached by the reduction to G3 in [6], this modification can be done. Their expressions WWIN and BWIN use a number of subexpressions a_i' over W_1 to W_m and b_i' over B_1 to B_m which change

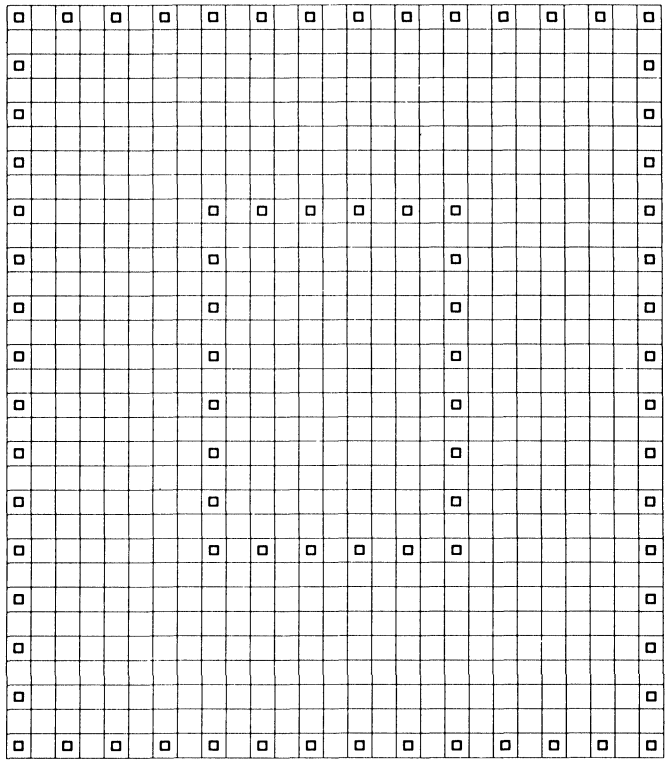


FIG. 4. *Picket lines.*

cyclically; that is as long as no player makes a stupid move which loses at once, every move by W sets exactly one a_i' true, the following move by B sets b_i' true (for the same i) and all other b' false, and the following move by W sets a_{i+1}' true et cetera (all subscripts being taken modulo $2m+2$). Hence after $W(B)$ moves WGONE (BGONE) is always true where

$$\text{WGONE} = \bigvee_{i=1}^{2m+2} (a_{i+1}' \wedge b_i'), \qquad \text{BGONE} = \bigvee_{i=1}^{2m+2} (a_i' \wedge b_i')$$

and moreover no move by $W(B)$ leaves BGONE (WGONE) true. Hence, if we play G3 on the expressions WWIN v WGONE and BWIN v BGONE, the outcome will be the same as the original game but any attempt to cheat by not moving at G3 means that the offending player has lost.

The reduction to Checkers will be from these particular modified instances of G3 with the further trivial restriction that every disjunct in WWIN (BWIN) contains at least two $W(B)$ variables. Even with these restrictions the reduction will prove the Exptime completeness of Checkers.

4. The simulation of G3. This section gives a general description of the Checkers configuration to be placed at A in Fig. 1 to model an instance of G3 and describes the sequence of moves known as "normal play" which first simulates the game of G3 and then allows the winner of this game to force his opponent to make a sequence of X captures allowing the G3 winner to make his winning X moves in the Checkers spiral. Subsequent sections will describe certain parts of the configuration and play

in greater detail and examine the effect of deviation from normal play. We describe the position from the point of view of White who is assumed to play from the bottom of the board; the description from Black's point of view is obtained by interchanging "Black" with "White" and "B" with "W" and rotating all figures through 180 degrees.

Corresponding to each variable W_i is a "boolean controller" where White can move a king between two squares called T (representing true) and F (false). In normal play White and Black simply make moves in their respective boolean controllers representing W and B's moves at G3 until the simulated game of G3 is over.

Corresponding to each disjunct of WWIN is an "attack zone" providing the mechanism by which White can force the sequence of X captures by Black provided the disjunct is true. The W variables in the disjunct are called the "attack variables"; provided they all have the value required for the disjunct to be true, White can mount an "attack" in the zone. The B variables in the disjunct are called the "defence variables"; Black can successfully defend against White's attack provided *any* defence variable does *not* have the value required for the disjunct to be true. Thus the condition that White can mount an attack and Black can not defend against it is precisely the truth of the disjunct.

The connection between boolean controllers and attack zones is provided by a mechanism called "firing" the controller (or the variable). Firing a controller is a move within the controller by the "owner" of the variable which initiates a sequence of forced captures which will eventually affect an attack zone. Which attack zone is affected, is determined firstly by the value of the simulated variable when the controller is fired and then by choices made by the owner of the variable. The detailed structure of the expressions WWIN and BWIN is reflected in the possible paths provided from controllers to attack zones.

When White (about to move) has one of his WWIN disjuncts true, he fires the variables W_i occurring in it in order of increasing i and directs the capture sequences to the attack zone corresponding to the disjunct. Each capture sequence except the penultimate leaves White to make the next move and so able to fire the next variable. Eventually after the penultimate is fired, Black has a nonforced move. Now if Black had a defence variable with the right value, he could fire it and when the capture sequence reached the boolean controller, because of the changes produced by White's attack, Black could use it to produce X forced captures by White. Since Black has no such defence variable, he can only make delaying moves after which White fires his last attack variable, setting up X forced captures by Black and winning.

Figure 5 illustrates the geometry of the position for a trivial instance of G3. The diagonal lines represent paths of possible capture sequences and the places where two such paths cross or diverge are the "crossovers" and "forks" described later. The picture is slightly more complex than has been described above so as to give White a multiplicity of winning methods ensuring that he can still win even if Black "cheats"; there are two attack zones for each disjunct and the last attack variable has two possible capture paths to each of these attack zones; to allow for these four possible capture paths, the boolean controller has up to four exits which may be taken on firing it for each value of the variable.

The case illustrated is

$$\text{BWIN} = B_1 \wedge B_2 \wedge W_1 \wedge \sim W_2, \qquad \text{WWIN} = C_1 \vee C_2,$$

where

$$C_1 = W_1 \wedge W_2 \wedge B_1, \qquad C_2 = W_1 \wedge \sim W_2 \wedge \sim B_2.$$

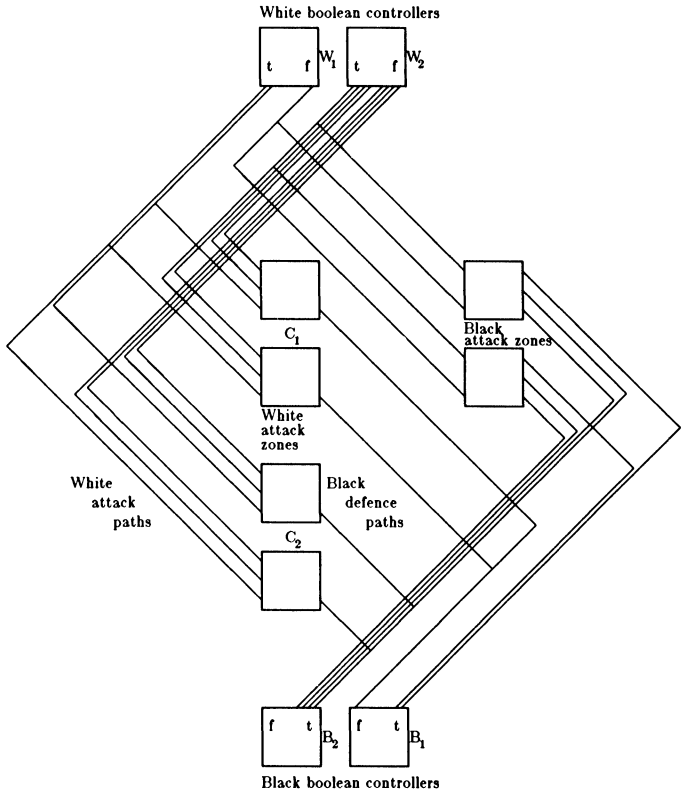


FIG. 5. *The simulation of* G3.

5. Capture sequences. This section starts the detailed description of the configuration outlined in § 4. The discussion continues to be given from White's point of view, so it describes only the White boolean controllers and attack zones and the potential capture paths shown in the top and left parts of Fig. 5 running from White boolean controllers to attack zones of both colors. The form of these potential capture paths is in general like that shown in Fig. 6. Any length of one of these paths can be traversed

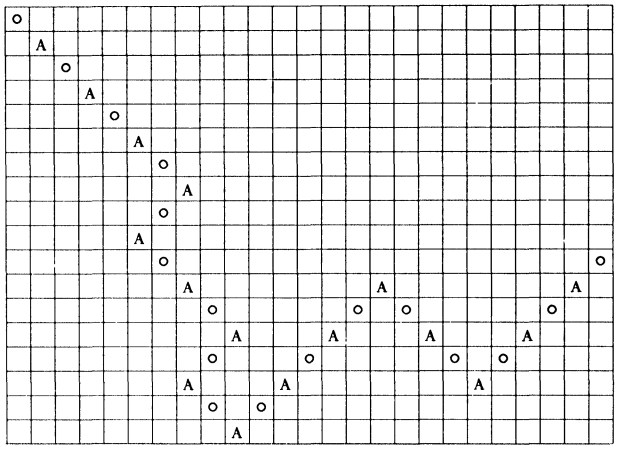


FIG. 6. *Potential capture path.*

(along the "A"s) in a single move by a Black king or by a single Black piece as long as the direction remains downwards. These simple paths are interrupted by a variety of "gadgets" within boolean controllers and attack zones and where they cross and fork. These gadgets are now described in detail.

5.1. Adjusting a capture sequence. Before a capture sequence can behave in the required way in a gadget, it may be necessary to change the color of the piece making the capture. Figure 7a shows how a capture by a Black piece or king, entering the figure at D and coming to rest at C can force a recapture by the White king at B which will leave the figure at A and can then enter a gadget.

It may also be necessary to "shift" a capture sequence to obtain the correct spatial relationship between two sequences. Figure 7b shows a capture by White, entering along the line of Es which forces a recapture by the Black king at I after which White must start a new capture with the king at F which leaves the diagram along the line of Gs. This new sequence is displaced one square diagonally from the line of the original. A combination of up to two of these shifts with the more obvious zigzags of Fig. 6 can cause whatever displacement of a capture path is required.

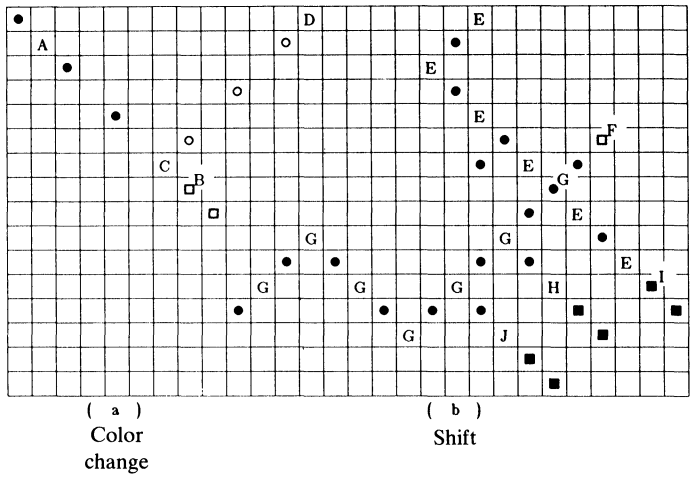


FIG. 7

These two "components", color changes and shifts, are to be added before gadgets as necessary to enable capture sequences to enter gadgets on the correct squares and with the correct color capturing. The reverse color change may also be necessary but the shift shown in Fig. 7b is the only one to be used on paths from White boolean controllers. (If the corresponding component to shift a Black capture sequence were used here, Black could take one of the "side turnings" to H or J nullifying the effect White wants from firing his controller. It is never to White's advantage to take these side turnings in normal play; the appendix points out why they are necessary.)

5.2. Crossovers and forks. Figure 8a shows how two potential captures from A to a and from B to b can cross each other without permitting the player making the capture to switch from one path to the other. The fact that after one of the captures has been made the other is impossible, is not significant in normal play.

Figure 8b shows a simple fork where a White king entering at C can choose between two possible exit routes D and E.

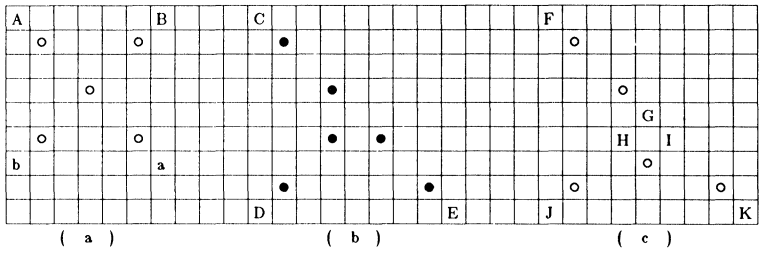


FIG. 8

Figure 8c shows a more complex fork to be used only in White boolean controllers. A single Black piece enters the fork by F and stops at G; now a White piece can go to either H or I and force a continuation in either of two directions leaving by J or K. Of course White also has the option of not moving to H or I and halting this capture sequence.

5.3. Enablers and delays. Figure 9 shows how one capture (crossing the figure diagonally from A) makes it possible for a later one to proceed (from B). This gadget will be used in attack zones.

Figures 10a and 10b show how a capture by Black may leave behind a forced capture by White or Black respectively. In each case, Black captures down the line of "A"s and this leaves a piece (B or C) able to capture by jumping into a newly vacant square. These will be called respectively White and Black "delays" and lines of X of them will be used in boolean controllers and attack zones. Such a line of X

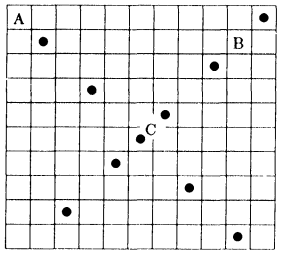


FIG. 9. *Enabler.*

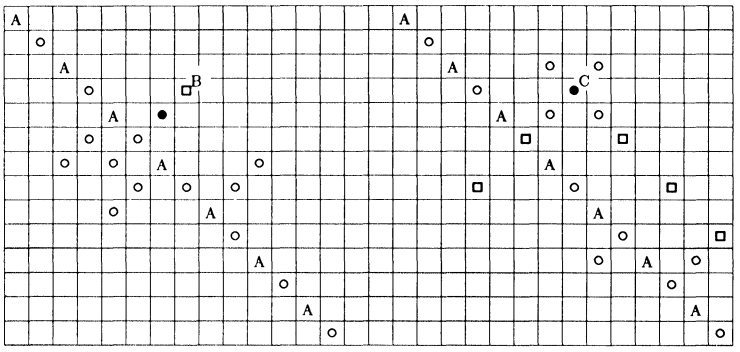


FIG. 10

delays will be represented in diagrams by MBD (Multiple Black Delay) or MWD (Multiple White Delay).

6. Boolean controllers and attack zones. Figure 11 illustrates a White boolean controller and the symbolism used for it in Fig. 5. The lines in the middle part of the figure represent lines of White pieces able to be captured in a single move, broken only by forks of the type shown in Fig. 8c. A White king is at either T or F.

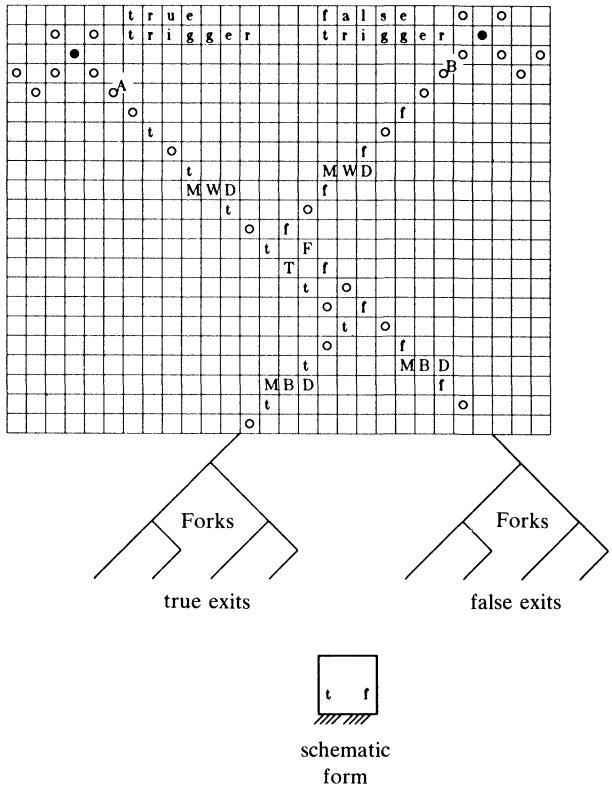


FIG. 11. *White boolean controller.*

In normal play White moves his king between T and F to simulate changes in the corresponding G3 variable between true and false. To fire the controller when the king is at T, White moves the piece at A in the "true trigger." This allows (forces) Black to capture down the path marked t, through White delays, the king at T and Black delays before coming to rest at the first fork in the true side of the controller. Now White and Black are forced to spend their next X moves making the captures in their respective delays after which White makes his moves in the forks forcing a Black capture move to emerge from the controller at whichever of the true exits White chooses. The process of firing the controller when the king is at F is similar but uses the false trigger to produce a capture emerging from one of the false exits.

Once a Black capture energes from the controller, a sequence of forced captures continues through forks and crossovers until eventually it reaches an attack zone. Since White makes the choices at all the forks, he can choose to direct the capture sequence to any of the attack zones to which a path exists.

From the true exits of the controller for W_i, there will be paths to all attack zones for disjuncts in WWIN which contain W_i (in nonnegated form) and to zones for disjuncts in BWIN containing $\sim W_i$. Conversely from false exits there will be paths to WWIN disjuncts containing $\sim W_i$ and BWIN disjuncts containing W_i.

Figure 12 illustrates the form of an attack zone for a WWIN disjunct with three attack variables and two defence variables. The figure incorporates several enablers

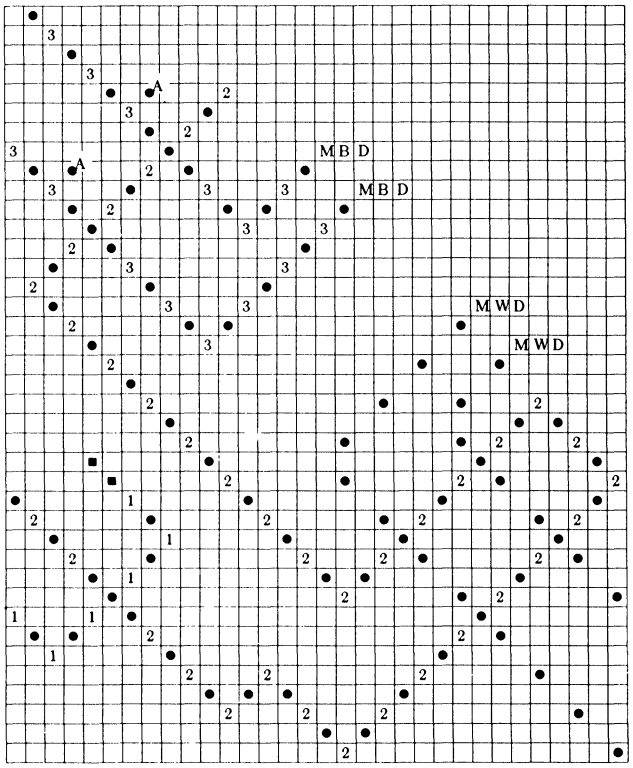


FIG. 12

as discussed in § 5.3. All capture sequences reaching an attack zone cause a White king to enter the zone along one of the lines of Black pieces. The delays in this zone, being astride lines of Black pieces are those obtained by reversing colors and directions from those of Fig. 10.

We recall from § 4 that the attack is supposed to succeed if every attack variable has the correct value to fire at the zone and no defence variable has the correct value. Thus the attack zone is constructed so that *all* attack variables must fire for the attack's success but *any* defence variable firing is sufficient to prevent it. The way in which the illustrated zone achieves this is as follows.

Each attack variable, except the last two, traverses a path like that marked "1" enabling its successor and terminating in a simple Black delay. The penultimate attack variable traverses a path like that marked "2" enabling all the defence variables and the last attack variable before terminating leaving Black to move. Now if Black had defence variable able to fire at this zone, he could fire it and win by activating one of the Multiple White Delays. Otherwise White will win by firing his last attack variable

and causing a Multiple Black Delay: Black cannot use his free move to sabotage the path of this last attack variable because we have provided White with two separate paths.

7. Abnormal play. From now on it is assumed that W has a forced win at G3. It should be clear from the above discussion that, as long as Black plays moves which simulate G3 moves by B, White will eventually have some attack zone where he is ready to fire all his attack variables and Black has no defence variable ready to fire. Thus White will win unless Black finds some "unexpected" move which either disrupts the simulation of G3 or sabotages the attack in the attack zone. This section will rule out that possibility, completing the proof of the correctness of the reduction.

Moves by Black other than those simulating G3 moves will be classified as either "irrelevant" moves which White can ignore since they do not hinder his winning strategy, or "catastrophic" moves which enable White to win immediately by forcing X Black captures, or the majority of "cheating" moves which can be regarded as not altering a G3 variable and so enable White to win by using the disjunct of WGONE which was true after his move.

7.1. Irrelevant moves. These are the very small number of possible moves which, after a sequence of forced captures by each side, leave Black again with the next nonforced move so that he can resume the G3 simulation. They consist of firing a Black boolean controller with the variable having the correct value (e.g. using the true trigger with the king at T) followed by either halting the captures in the forks of the controller or allowing capture sequences to reach Black attack zones in the correct enabling orders but not as far as the penultimate attack variable in any zone.

These moves in no way interfere with White's ability eventually to mount a successful attack. They merely prevent Black from later altering the values of the affected G3 variables, an effect which cannot be to Black's advantage.

7.2. Catastrophic moves. These are two types of move by Black which are punished by the immediate activation of a Multiple Black Delay.

The first type is for Black to fire a boolean controller using the "wrong" trigger; for instance to use the true trigger with his king at F. The resulting capture by White goes through a Multiple Black Delay but stops before T and so does not activate the compensating Multiple White Delay. (Note that these delays are essential to the whole argument; if they were not there, Black could now move his king from F to T.)

The second type of catastrophic move may occur if Black fires attack variables in the correct enabling order so that eventually the penultimate attack variable enters some attack zone and enables the last attack variable. Now by the assumption that W has a winning strategy at G3 (and that White has been simulating it), either White has a defence variable ready to fire or Black does not have his last attack variable ready. In the first case, Black's move has been catastrophic and White will fire his defence variable and activate a Multiple Black Delay. In the other case, Black's last move has simply wasted time and White treats it as a "cheating" move of the type discussed below.

7.3. Cheating moves. This category covers all other Black moves. That is, all moves in the spiral, in boolean controllers, attack zones and other gadgets or in the capture paths connecting them except for (i) normal simulation of G3 moves and (ii) the irrelevant and catastrophic moves described above. In all these cases White responds by mounting an attack in an attack zone whose disjunct was true before Black's move.

The fact that White can always do this depends on three properties of moves in the various gadgets etc., which can only be verified by tedious checking of many cases:

(1) Any cheating move loses a tempo. That is although it may result in a sequence of forced captures, after this sequence White is the first player to have a nonforced move. (This may depend on White making the correct choice when forced to choose between two possible captures.)

(2) Any such move, together with its following forced captures, causes at most localized damage to the configurations enabling White to mount an attack. To be more precise, although it may destroy one potential capture path outside a boolean controller, it cannot

(a) prevent White from firing a boolean controller and choosing which exit is taken,

or

(b) prevent a Black delay from working,

or

(c) damage two potential capture paths from White boolean controllers (except at a fork where the two diverge).

Properties (1) and (2) are sufficient to ensure that after any cheating move by Black, White can still choose an undamaged attack zone corresponding to his true disjunct, which has all the capture paths to it also undamaged. Thus White can fire the attack variables for that attack zone at least up to the penultimate. Then Black has one last opportunity to stave off defeat. Property (1) may now no longer hold; Black may be able to make a move close to his original cheating move which forces a White capture after which Black has a free move and may repeat this process. Calling such a sequence of moves a "supplementary sequence" we now state the last required property:

(3) One cheating move by Black together with a supplementary sequence cannot

(a) prevent White from firing a boolean controller and choosing which exit is taken,

or

(b) damage more than two potential capture paths, from White boolean controllers,

or

(c) regain the lost tempo by causing two forced White captures,

or

(d) affect the type of Black delay occurring in White boolean controllers.

Since White has two potential capture paths to his chosen attack zone for his last attack variable, this ensures that he can eventually activate one of the Multiple Black Delays and win.

The apparent over-complexity of many of the gadgets stems from the need to ensure that (1), (2) and (3) hold. Most of the checking of these properties has been done by a program which did detect one error in an earlier version of Fig. 10b. The appendix points out a few ways in which simpler gadgets are inadequate.

Two points should be mentioned here since they concern the general structure of the attack zone rather than the details of the gadgets within the zone: to ensure that 3(c) holds, the path of a defence variable within attack zones passes through two enablers each of which is opened by an attack variable; otherwise Black could win by moving the piece at C of Fig. 9 (in the enabler for one of his defence variables) as his initial cheating move and later firing that defence variable as his supplementary sequence. Secondly the orientation of the enablers within the attack zone ensures that

either player firing a variable which has not yet been enabled loses a tempo as required by property (1) and the Black pieces marked "A" in Fig. 12 prevent a tricky play by White to try to regain that tempo.

8. Summary. Apart from some minor details concerning the exact placement of the gadgets on the Checkers board and the necessary gaps between various gadgets and the spiral, the Checkers position corresponding to a G3 position has been fully described. Moreover it has been shown that the first player to deviate, except in a few insignificant ways, from the moves simulating G3 loses at Checkers and that, if the simulated game of G3 ends, the winner of that game wins at Checkers. This completes the demonstration of the reduction from G3 to N by N Checkers.

If the length of the description of the instance of G3 is n, then both the number of variables and the number of disjuncts in WWIN and BWIN are $O(n)$. Hence the area A in which the G3 simulation takes place can be enclosed in a square of side $O(n)$ containing $O(n^2)$ pieces. Hence from the discussion in § 2, the whole Checkers position can certainly be placed on a board of side $O(n^4)$.

The regular way in which the placement of pieces on the board depends on the disjuncts of WWIN and BWIN means that the Checkers position could be output by a RAM program using $O(1)$ variables each restricted to a range of $O(n^4)$ values. Finally any reasonable Turing machine simulation of this RAM program will yield the required $O(\log n)$ bound on the workspace complexity of the reduction.

9. Conclusions and some open questions. This completes the proof of the main result that the set of positions at N by N Checkers from which White can force a win is Exptime complete. Thus any algorithm to recognize these positions must have running time $\Omega(c^{N^k})$ for some $c > 1$ and $k > 0$. This conclusion unfortunately reveals nothing about the difficulty of such a recognition algorithm for the 8 by 8 case; an 8 by 8 board would not even hold one boolean controller. This lack of information on particular finite cases is of course common to such lower bound complexity results.

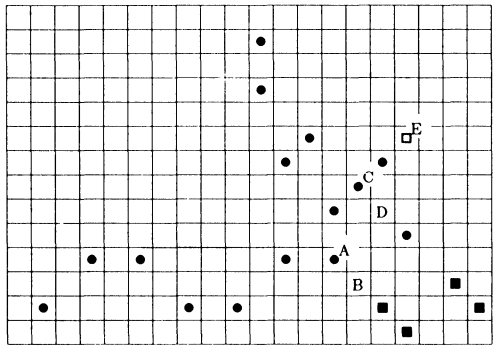
Two other conclusions follow. Firstly any algorithm to decide an optimal move must also require exponential time (by a fairly obvious argument given in [5]). Secondly by following the chain of reductions to Checkers from an arbitrary exponential time Turing machine computation [6], it follows that the number of moves needed to force a win against optimal delaying defensive play also rises exponentially with N. Thus a conventional minimax algorithm has exponential space complexity; of course that does not of itself preclude the possibility of a polynomial space algorithm.

Having shown that Checkers has intractable decision problems despite its simplicity in each player having only two types of piece, one naturally wonders whether the problems would remain intractable if only one type of piece existed. For a game with only kings, it seems likely that a reduction similar to the one given here, but with different gadgets, would prove the same results. On the other hand, in a game played only with single pieces which, if they reached the far end of the board, were not promoted to kings and so became immobile, the length of a game would be bounded by a polynomial in the board size giving decision problems solvable in polynomial space. Another interesting question is whether the forced capture rule, which has been used so heavily, is in fact essential to the conclusion; the methods used in this paper do not appear to throw much light on that question.

Appendix. This appendix discusses how the complexity of some of the gadgets used in the reduction is related to the need for them to satisfy properties (1), (2) and (3) of § 7. In fact they were designed to satisfy a slightly stronger version of (3b),

namely that no cheating move followed by any supplementary sequence within a gadget should cause any repercussions outside the gadget. This "locality" property makes the checking of properties (1) to (3) much easier whether it is done by hand or by machine. No claim is made that the gadgets used are the simplest possible with the required properties. However four examples will be given of how apparently superfluous pieces within gadgets prevent breaches of the properties. This explanation is not part of the proof of the results of the paper but it may make § 7 clearer.

The first example is in Fig. 13 which shows a shift modified by removing one of the side turnings of the original. Now Black can move from A to B removing the second side turning and later as a supplementary move, move from C to D. The resulting capture by the White king at E will cause a capture sequence which may pass through several crossovers certainly invalidating property (3b) and possibly (3c).



Shift

FIG. 13

The remaining three examples are all contained in Fig. 14 which shows the delays slightly modified. The White delay has had two White pieces removed from A and B. The removal of the piece from A enables White to move from C to A forcing a capture by the Black piece D, invalidating property (1). Of course White would not do this in a White delay in a White boolean controller but if White can do it there, then Black can make a similar move in all the Black delays in White attack zones.

Returning to the White delay, Black can move from D to E (cheating) and then E to F (supplementary). The absence of the White piece at B now forces White to capture with G invalidating property (3a).

Finally in the Black delay, Fig. 14 has dropped a White piece from H. The need for this piece at H was discovered by computer testing which found that this Black delay violated the locality property. White can make cheating and supplementary moves which cause nonlocal effects as follows:

(i) move from I to J,

(ii) move out from K forcing Black to capture from L to K and I leaving White still able to move,

(iii) move from M to N forcing Black to capture from I to M,

(iv) move from P to H forcing Black to capture from M to P,

(v) move from Q to R forcing Black to capture from P to Q, et cetera.

In fact it appears that this last modification does not invalidate the reduction. However, it seems easier to design gadgets which satisfy the locality property than to

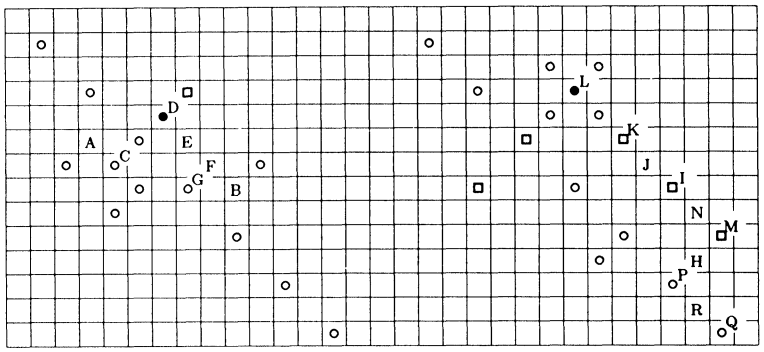


FIG. 14

justify the claim that it is irrelevant by arguments about how one gadget may affect its neighbors.

10. Acknowledgment. I would like to thank the anonymous referee of an earlier version of this paper for many helpful comments, particularly for the suggestion of a simplified version of the boolean controller.

Note added in proof. The Exptime completeness result for GO referred to in § 1.3 has been presented at the 1983 IFIP World Computer Congress.

REFERENCES

[1] A. S. FRAENKEL, M. R. GAREY, D. S. JOHNSON, T. SCHAEFER AND Y. YESHA, *The complexity of Checkers on an N by N board, preliminary report*, Proc. 19th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1978, pp. 55–64.

[2] A. S. FRAENKEL AND D. LICHTENSTEIN, *Computing a perfect strategy for n by n chess requires time exponential in n*, J. Combin. Theory, 31 (1981), pp. 199–214.

[3] T. KASAI, A. ADACHI AND S. IWATA, *Classes of pebble games and complete problems*, this Journal, 18, (1979), pp. 574–586.

[4] D. LICHTENSTEIN AND M. SIPSER, *Go is polynomial-space hard*, J. Assoc. Comput. Mach., 27 (1980), pp. 393–401.

[5] J. M. ROBSON, *Optimal storage allocation decisions require exponential time*, Australian Computer Science Communications, 3 (1981), pp. 162–172.

[6] L. J. STOCKMEYER AND A. K. CHANDRA, *Provably difficult combinatorial games*, this Journal, 8 (1979), pp. 151–174.