Games of No Chance 3
MSRI Publications
Volume **56**, 2009

Amazons, Konane, and Cross Purposes are PSPACE-complete

ROBERT A. HEARN

ABSTRACT. Amazons is a board game which combines elements of Chess and Go. It has become popular in recent years, and has served as a useful platform for both game-theoretic study and AI games research. Buro showed that simple Amazons endgames are NP-equivalent, leaving the complexity of the general case as an open problem.

Konane is an ancient Hawaiian game, with moves similar to peg solitaire. Konane has received some attention in the combinatorial game theory community, with game values determined for many small positions and one-dimensional positions. However, its general complexity seems not to have been previously addressed.

Cross Purposes was invented by Michael Albert, and named by Richard Guy at the Games at Dalhousie III workshop, in 2004. It is played on a Go board. Cross Purposes is a kind of two-player version of the popular puzzle Tipover: it represents stacks of crates tipping over and blocking others from tipping over.

We show that generalized versions of these games are PSPACE-complete. We give similar reductions to each game from one of the PSPACE-complete two-player formula games described by Schaefer. Our construction also provides an alternate proof that simple Amazons endgames are NP-equivalent.

1. Introduction

Combinatorial game theory is concerned with the attempt to find and analyze winning strategies for combinatorial games, or for tractable families of game positions. However, it is a curious fact that with few exceptions, any game or puzzle that is interesting to humans, and whose worst-case complexity is known,

During the preparation of this paper, the author learned that a group including Buro et al. was simultaneously preparing a paper showing Amazons to be PSPACE-complete [7].

is computationally as hard as possible based on very general characteristics of the game. By hardness here we mean computational complexity of determining the existence of a winning strategy for a given player, from a given position. For example, Minesweeper is a one-player game (puzzle), with a bounded number of moves; it is NP-complete [11]. Sliding-block puzzles do not have a bound on the number of moves; this raises the complexity to PSPACE-complete [10]. Two-player, bounded-move games, such as Hex, are also generally PSPACE-complete [14]. Other games can be even harder: Chess, Checkers, and Go (Japanese Rules), as two-player games with no bound on the number of moves, are EXPTIME-complete [6; 16; 15]. There are harder games still.

Amazons, Konane, and Cross Purposes are all two-player games with a polynomially bounded number of moves. We should therefore expect them to be PSPACE-complete, merely on the grounds that they are interesting games to play, and therefore presumably are as complex as possible given their general characteristics.

In the terminology of combinatorial game theory, all three games also follow the *normal play* convention: the first player who cannot move loses. Additionally, all three games are played on a square grid, with pieces that move, or are captured, or are transformed. These shared characteristics will enable us to use the same proof technique to show all of them PSPACE-hard. Only the specific gadgets differ among the three proofs. Our proof technique seems simpler than that used for many game results, and may have wider applicability. In particular, the generic crossover construction seems likely to simplify new hardness proofs.

As with most hardness results, the hardness of these games only applies directly to particular configurations explicitly constructed to have computational properties. It does not say anything about the difficulty of determining the winner from a standard initial game configuration, or even from reasonable positions that might arise in actual play. Indeed, Hex is PSPACE-complete in general, but a simple strategy-stealing argument shows that from an empty board, it is a first-player win. Nonetheless, a hardness result for a game indicates that there are limits to the degree to which it can be theoretically analyzed.

Conway, Berlekamp, and Guy argue against a tendency to dismiss hard problems as uninteresting [2, page 225]:

> Some people consider a class of problems "finished" when it has been shown to be NP-hard. Philosophically this is a viewpoint we strongly oppose. Some games which are NP-hard are very interesting!

Our view is just the reverse of that argued against: interesting games are almost of necessity hard. Showing a game to be hard is an indication that the game *is* interesting! That is the spirit in which these results are presented.

Outline. Section 2 describes the reduction to be used for each game. Sections 3, 4, and 5 detail the background, history, rules, and hardness proofs for Amazons, Konane, and Cross Purposes, respectively. Section 6 summarizes our results.

2. Reduction framework

Formula game. Schaefer [17] showed that deciding the winner of the following two-person game is PSPACE-complete: Let A be a positive CNF formula (i.e., a propositional formula in conjunctive normal form in which no negated variables occur). Each player on his move chooses a variable occurring in A which has not yet been chosen. After all variables have been chosen, player one wins if A is true when all variables chosen by player one are set to true and those chosen by player two are set to false.

We will refer to this game as the *formula game*. Our hardness reductions consist of constructing game configurations which force the two players to effectively play a given formula game.

Given a positive CNF formula A, we build logic and wiring gadgets corresponding to the variables and the formula, as shown schematically in Figure 1. (We use standard digital logic symbols for AND and OR.) If player one plays first in a variable, a signal is enabled to flow out from it; if player two plays first, that signal is blocked. When a signal arrives at or leaves from a gadget, we will speak of that input or output as *activating*. By splitting the signals, allowing them to cross, and feeding them into a network of logic gates, we may construct a particular signal line that player one may eventually activate only if A is true under the selected variable assignment. For each game, we arrange for player one to win just when he can activate that output signal.

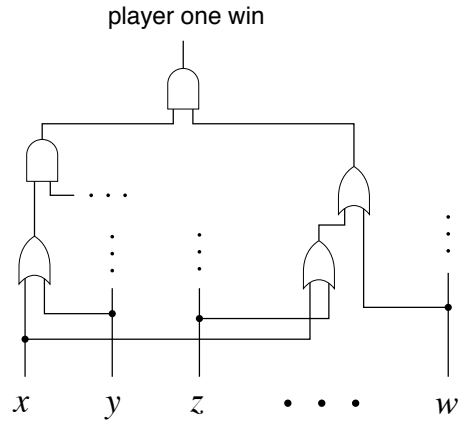
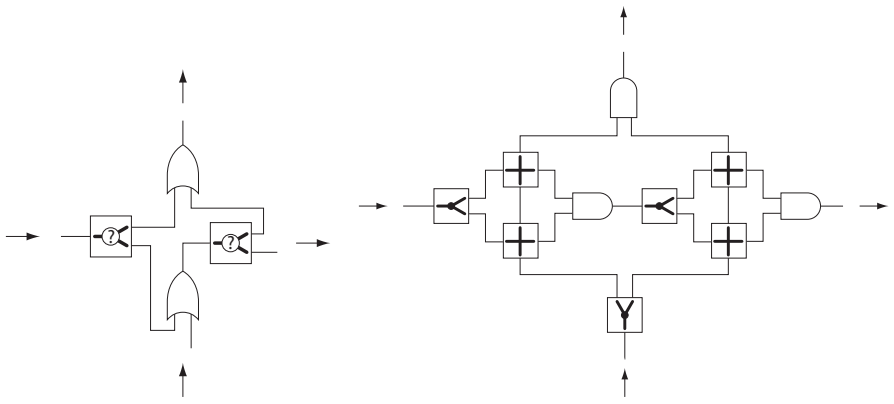


Figure 1. Reduction schematic. The circuit shown corresponds to the positive CNF formula $(x \vee y) \wedge \ldots \wedge (x \vee z \vee w)$.



(a) Half crossover (b) Crossover

Figure 2. Crossover gadgets.

Generic crossover. Crossover gadgets are often among the most complicated and difficult to construct in game hardness reductions. Rather than construct three separate crossover gadgets, we give a generic construction for crossing signals, given the existence of AND, OR, *split*, and *choice* gadgets. This means that in addition to the basic wiring and logic gadgets, we merely need to construct a choice gadget for each game. A choice gadget allows one, but not both, outputs to activate if the input activates. (Note that while there are traditional digital-logic methods for crossing signals in planar circuits, they require inverters, which do not fit well into our problem formalism.)

We develop the ability to cross signals in two steps. The first step is the *half-crossover* gadget, shown in Figure 2(a). Using half crossovers, we can make a full *crossover* gadget, shown in Figure 2(b). Splits are shown with a forking symbol, choice gadgets with a question mark, and half crossovers with a plus symbol. These have the expected properties: e.g., if the left input of the leftmost choice gadget in Figure 2(a) activates, then either, but not both, of its right outputs may activate; similarly, if the left input of the leftmost split gadget in Figure 2(b) activates, then both of its right outputs may activate. We note that this crossover construction is essentially the same as that used in [10]; the only difference is that in [10], the gates are reversible.

The half crossover has the property that if either input activates, either output may activate; if both inputs activate, both outputs may activate. Suppose the left input activates. Then the player propagating the signal may activate the left (i.e., upper) output of the left choice, then the top OR and the top output; or he may choose to activate the right output of the left choice, then the bottom OR, then the right output of the right choice, and the right output. Similarly, if the bottom

input is activated, the signal may be directed to either output. If both inputs are activated, then by making the correct choices both outputs may be activated.

For the crossover gadget, we want the left input to be able to propagate only to the right output, and likewise vertically. First, it is clear that if one input activates, the corresponding output may also activate; simply choose the straight-through path to activate for each half crossover. The splits and ANDs then propagate the signal across the gadget. If both inputs activate, activating both half-crossover outputs allows both crossover outputs to activate.

Suppose the left split's input has not activated. Then at most one input to the left AND may activate, because the bottom-left half crossover can have at most one input, and thus output, activate. Therefore, the left AND's output may not activate. By the same reasoning, the right AND may not activate either. A similar argument shows that if the bottom split's input has not activated, the top AND may not activate. Therefore, the gadget serves to cross signals, as needed.

3. Amazons

Amazons was invented by Walter Zamkauskas in 1988. Both human and computer opponents are available for Internet play, and there have been several tournaments, both for humans and for computers.

Amazons has several properties which make it interesting for theoretical study. Like Go, its endgames naturally separate into independent subgames; these have been studied using combinatorial game theory [1; 18]. Amazons has a very large number of moves available from a typical position, even more than in Go. This makes straightforward search algorithms impractical for computer play. As a result, computer programs tend to incorporate explicit high-level knowledge of Amazons strategy [13; 12]. By showing that generalized Amazons is PSPACE-complete, we provide strong evidence that there is a practical limit to the degree of analysis possible from an arbitrary position.

As mentioned in the footnote on the first page, Furtak, Kiyomi, Uno, and Buro independently showed Amazons to be PSPACE-complete at the same time as the author [7]. (The original version of the present paper, containing only the Amazons proof, is available at [8].) Curiously, [7] already contains two different PSPACE-completeness proofs: one reduces from Hex, and the other from Generalized Geography. The paper is the result of the collaboration of two groups which had also solved the problem independently, then discovered each other. Thus, after remaining an open problem for many years, the complexity of Amazons was solved independently and virtually simultaneously by three different groups, using three completely different approaches, each of which leverages different aspects of the game to construct gadgets. This is a remarkable fact. The reduction from Generalized Geography provides the strongest result:

it shows that Amazons is PSPACE-complete even when each player only has a single Amazon. In contrast, the Hex reduction and the present formula game reduction each require a large number of Amazons.

Amazons rules. Amazons is normally played on a 10×10 board. The standard starting position, and a typical endgame position, are shown in Figure 3. (We indicate burned squares by removing them from the figures, rather than marking them with tokens.) Each player has four *amazons*, which are immortal chess queens. White plays first, and play alternates. On each turn a player must first move an amazon, like a chess queen, and then fire an *arrow* from that amazon. The arrow also moves like a chess queen. The square that the arrow lands on is burned off the board; no amazon or arrow may move onto or across a burned square. There is no capturing. The first player who cannot move loses.

Amazons is a game of mobility and control, like Chess, and of territory, like Go. The strategy involves constraining the mobility of the opponent's amazons, and attempting to secure large isolated areas for one's own amazons. In the endgame shown in Figure 3, Black has access to 23 spaces, and with proper play can make 23 moves; White can also make 23 moves. Thus from this position, the player to move will lose.

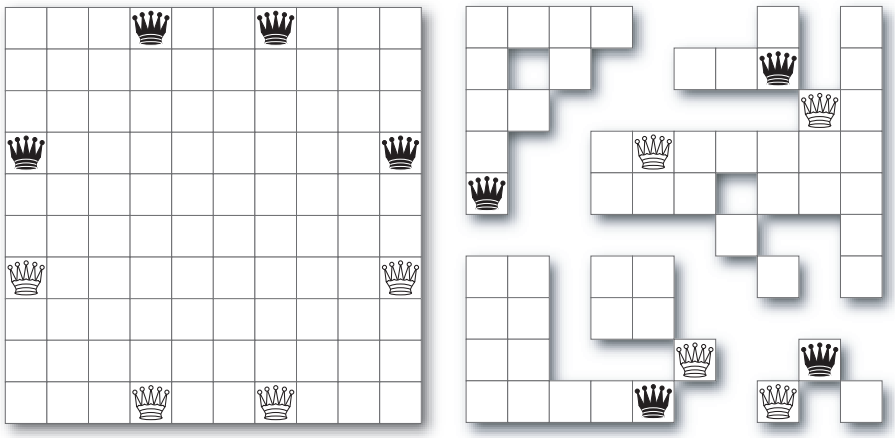


Figure 3. Amazons start position and typical endgame position.

3.1. PSPACE-completeness. We follow the reduction framework outlined in Section 2. The game consists of a variable selection phase, during which all play occurs in variable gadgets, followed by a phase in which White attempts to activate a signal pathway leading to a large supply of extra moves, enabling him to win. Black is supplied with enough extra moves of his own to win otherwise.

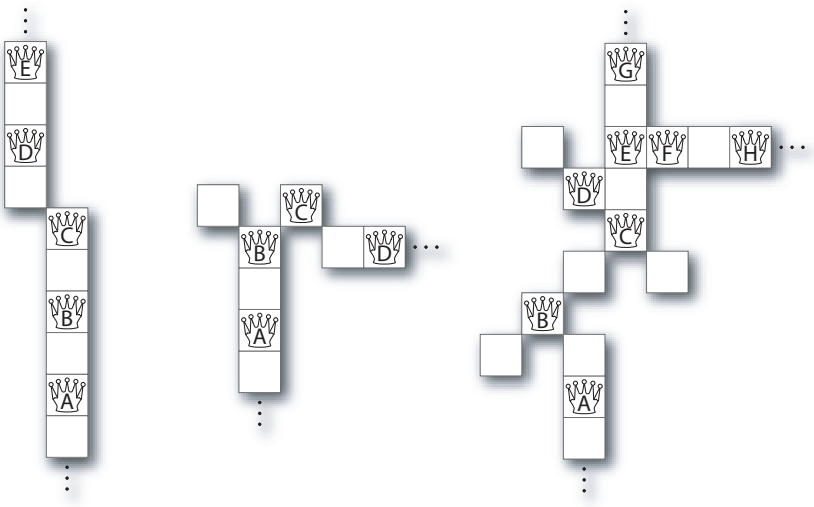
Basic wiring. Signals propagate along *wires*. Figure 4(a) shows the construction of a wire. Suppose that amazon A is able to move down one square and shoot down. This enables amazon B to likewise move down one and shoot down; C may now do the same. This is the basic method of signal propagation. When an amazon moves backward (in the direction of input, away from the direction of output) and shoots backward, we will say that it has *retreated*.

Figure 4(a) illustrates two additional useful features. After C retreats, D may retreat, freeing up E. The result is that the position of the wire has been shifted by one in the horizontal direction. Also, no matter how much space is freed up feeding into the wire, D and E may still only retreat one square, because D is forced to shoot into the space vacated by C.

Figure 4(b) shows how to turn corners. Suppose A, then B may retreat. Then C may retreat, shooting up and left; D may then retreat. This gadget also has another useful property: signals may only flow through it in one direction. Suppose D has moved and shot right. C may then move down and right, and shoot right. B may then move up and right, but it can only shoot into the square it just vacated. Thus, A is not able to move up and shoot up.

By combining the horizontal parity-shifting in Figure 4(a) with turns, we may direct a signal anywhere we wish. Using the unidirectional and flow-limiting properties of these gadgets, we can ensure that signals may never back up into outputs, and that inputs may never retreat more than a single space.

Splitting a signal is a bit trickier. The *split* gadget shown in Figure 4(c) accomplishes this. A is the input; G and H are the outputs. First, observe that



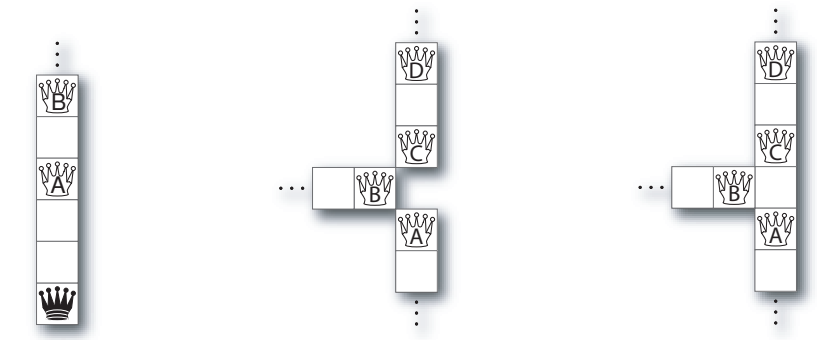
(a) Wire, parity, flow limiter (b) Turn, one way (c) Split

Figure 4. Amazons wiring gadgets.

until A retreats, there are no useful moves to be made. C, D, and F may not move without shooting back into the square they left. A, B, and E may move one unit and shoot two, but nothing is accomplished by this. But if A retreats, then the following sequence is enabled: B down and right, shoot down; C down and left two, shoot down and left; D up and left, shoot down and right three; E down two, shoot down and left; F down and left, shoot left. This frees up space for G and H to retreat, as required.

Logic. The *variable* gadget is shown in Figure 5(a). If White moves first in a variable, he can move A down, and shoot down, allowing B to later retreat. If Black moves first, he can move up and shoot up, preventing B from ever retreating.

The AND and OR gadgets are shown in Figures 5(b) and 5(c). In each, A and B are the inputs, and D is the output. Note that, because the inputs are protected with flow limiters — Figure 4(a) — no input may retreat more than one square; otherwise the AND might incorrectly activate.



(a) Variable (b) AND (c) OR, choice

Figure 5. Amazons logic gadgets.

In an AND gadget, no amazon may usefully move until at least one input retreats. If B retreats, then a space is opened up, but C is unable to retreat there; similarly if just A retreats. But if both inputs retreat, then C may move down and left, and shoot down and right, allowing D to retreat.

Similarly, in an OR gadget, amazon D may retreat if and only if either A or B first retreats.

Choice. For the generic crossover construction to work, we need a choice gadget. The existing OR gadget suffices, if we reinterpret the bottom input as an output: if if B retreats, then either C or A, but not both, may retreat.

Winning. We will have an AND gadget whose output may be activated only if the formula is true under the chosen assignment. We feed this signal into a *victory* gadget, shown in Figure 6. There are two large rooms available. The sizes are equal, and such that if White can claim both of them, he will win, but if he can claim only one of them, then Black will win.

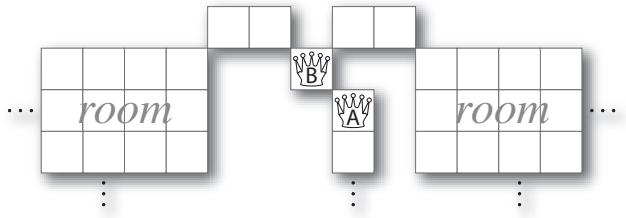


Figure 6. Amazons victory gadget.

If B moves before A has retreated, then it must shoot so as to block access to one room or the other; it may then enter and claim the accessible room. If A first retreats, then B may move up and left, and shoot down and right two, leaving the way clear to enter and claim the left room, then back out and enter and claim the right room.

THEOREM 1. *Amazons is PSPACE-complete.*

PROOF. Given a positive CNF formula A, we construct a corresponding Amazons position, as described above. The reduction may be done in polynomial time: if there are k variables and l clauses, then there need be no more than $(kl)^2$ crossover gadgets to connect each variable to each clause it occurs in; all other aspects of the reduction are equally obviously polynomial.

If the players alternate choosing variables, then when all variables have been chosen, White will be able to activate wires leading from only those variables he has chosen; these are just the variables assigned to true in the formula game. Since A contains no negated variables, White will thus be able eventually to reach both rooms of the victory gadget just if A is true under the variable assignment corresponding to the players' choices. White will then have more moves available than Black, and win; otherwise, Black's extra room will give him more moves than White, and Black will win.

Suppose a player makes a move which does not choose a variable, before all variables have been chosen. This can have no effect on the other player, apart from allowing him to choose two variables in a row, because the Black

and White amazons may only interact within variable gadgets. A player who chooses two variables in a row may finish with at least the same set of variables chosen as he would otherwise. Therefore, not playing in accordance with the formula game does not allow a player to win if he could not otherwise win.

Therefore, a player may win the Amazons game if and only if he may win the corresponding formula game, and Amazons is PSPACE-hard.

Since the game must end after a polynomial number of moves, it is possible to perform a search of all possible move sequences using polynomial space, thus determining the winner. Therefore, Amazons is also in PSPACE, and thus PSPACE-complete. □

3.2. Simple Amazons endgames. A *simple Amazons endgame* is an Amazons position in which the Black and White amazons are completely separated by burned squares. There can thus be no interaction between the amazons, and the winner is determined by which player can make the most moves in his own territory. Buro [3] showed that it is NP-complete to decide whether a player may make a given number of moves from an individual territory containing only his amazons. Buro first proved NP-completeness of the Hamilton circuit problem for cubic subgraphs of the integer grid, and then reduced from that problem. As a result, deciding the outcome of a simple Amazons endgame is NP-equivalent (that is, it can be decided with a polynomial number of calls to an algorithm for an NP-complete problem, and vice versa). Our gadgets provide a simple alternate proof.

THEOREM 2. *Deciding the outcome of a simple Amazons endgame is NP-equivalent*.

PROOF. We reduce SAT to a single-color Amazons position. Given a propositional formula A, we construct the same position as in Theorem 1, with the following modifications. We remove the Black amazons, then connect each variable output to the input of a choice gadget. We connect one choice output path to the non-negated occurrences of the corresponding variable in the formula, and the other output path to the negated occurrences.

Then, White may reach both rooms of the victory gadget if and only if A is satisfiable, by choosing the correct set of choice output paths. Therefore, it is NP-hard to decide whether a player may make a given number of moves from a position containing only his amazons. We may nondeterministically guess a satisfying move sequence and verify it in polynomial time; therefore, the problem is NP-complete. As in [3], it follows automatically that deciding the winner of a simple Amazons endgame is NP-equivalent. □

4. Konane

Konane is an ancient Hawaiian game, with a long history. Captain Cook documented the game in 1778, noting that at the time it was played on a 14×17 board. Other sizes were also used, ranging from 8×8 to 13×20. The game was usually played with pieces of basalt and coral, on stone boards with indentations to hold the pieces. King Kamehameha the Great was said to be an expert player; the game was also popular among all classes of Hawaiians.

More recently, Konane has been the subject of combinatorial game-theoretic analysis [5; 4]. Like Amazons, its endgames break into independent games whose values may be computed and summed. However, as of this writing, even $1 \times n$ Konane has not been completely solved, so it is no surprise that complicated positions can arise.

Konane rules. Konane is played on a rectangular board, which is initially filled with black and white stones in a checkerboard pattern. To begin the game, two adjacent stones in the middle of the board or in a corner are removed. Then, the players take turns making moves. Moves are made as in peg solitaire – indeed, Konane may be thought of as a kind of two-player peg solitaire. A player moves a stone of his color by jumping it over a horizontally or vertically adjacent stone of the opposite color, into an empty space. Stones so jumped are captured, and removed from play. A stone may make multiple successive jumps in a single move, as long as they are in a straight line; no turns are allowed within a single move. The first player unable to move wins.

4.1. PSPACE-completeness. The Konane reduction is similar to the Amazons reduction; the Konane gadgets are somewhat simpler. As before, the game consists of a variable selection phase, during which all play occurs in variable gadgets, followed by a phase in which White attempts to activate a signal pathway leading to a large supply of extra moves, enabling him to win. Black is supplied with enough extra moves of his own to win otherwise.

Basic wiring. A Konane wire is simply a string of alternating black stones and empty spaces. By capturing the black stones, a white stone traverses the wire. Note that in Konane, in contrast with the Amazons reduction, signals propagate by stones moving forwards, capturing opposing stones.

Turns are enabled by adjoining wires as shown in Figure 7; at the end of one wire, the white stone comes to rest at the beginning of another, protected from capture by being interposed be-

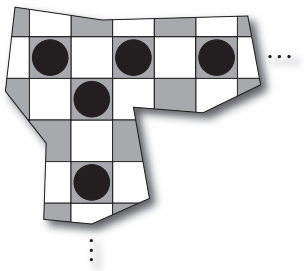


Figure 7. Konane wire, turn.

tween two black stones. If the white stone tried to traverse the turn in the other direction, it would not be so protected, and Black could capture it. Thus, as in the Amazons reduction, the turn is also a one-way device, and we assume that gadget entrances and exits are protected by turns to ensure that signals can only flow in the proper directions.

Conditional gadget. A single gadget serves the purpose of AND, split, and positional parity adjustment. It has two input / output pathways, with the property that the second one may only be used if the first one has already been used. This *conditional gadget* is shown in Figure 8; the individual uses are outlined below.

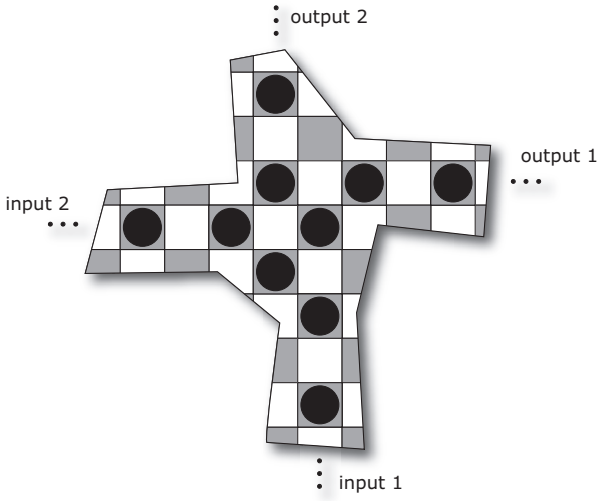


Figure 8. Konane wiring: conditional.

Observe that a white stone arriving at input 1 may only leave via output 1, and likewise for input 2 and output 2. However, if White attempts to use pathway 2 before pathway 1 has been used, Black can capture him in the middle of the turn. But if pathway 1 has been used, the stone Black needs to make this capture is no longer there, and pathway 2 opens up.

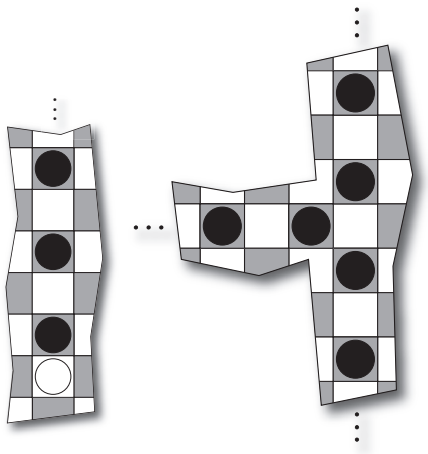
Split, parity. If we place a white stone within the wire feeding input 2 of a conditional gadget, then both outputs may activate if input 1 activates. This splits the signal arriving at input 1.

If we don't use output 1, then this split configuration also serves to propagate a signal from input 1 to output 2, with altered positional parity. This enables us to match signal parities as needed at the gadget inputs and outputs.

Logic. The *variable* gadget consists of a white stone at the end of a wire, as in Figure 9(a). If White moves first in a variable, he can traverse the wire, landing safely at an adjoining turn. If Black moves first, he can capture the white

stone and prevent White from ever traversing the wire.

The AND gadget is a conditional gadget with output 1 unused. By the properties of a conditional gadget, a white stone may exit output 2 only if white stones have arrived at both inputs. The OR gadget is shown in Figure 9(b). The inputs are on the bottom and left; the output is on the top. Clearly, a white stone arriving via either input may leave via the output.



(a) Variable (b) OR, choice

Figure 9. Konane logic gadgets.

Choice. For the generic crossover to work, we need a choice gadget. As was the case with Amazons, the OR gadget suffices, if we relabel the bottom input as an output: a white stone arriving along the left input may exit via either the top or the bottom. (For Konane, it turns out that crossover is a trivial gadget to make in any case.)

Winning. We will have an AND gadget whose output may be activated only if the formula is true under the chosen assignment. We feed this signal into a long series of turns, providing White with enough extra moves to win if he can reach them. Black is provided with his own series of turns, made of white wires, with a single black stone protected at the end of one of them, enabling Black to win if White cannot activate the final AND.

THEOREM 3. *Konane is PSPACE-complete.*

PROOF. Given a positive CNF formula A, we construct a corresponding Konane position, as described above. As in the Amazons construction, the reduction is clearly polynomial. Also as in Amazons, White may reach his supply of extra moves just when he can win the formula game on A.

Therefore, a player may win the Konane game if and only if he may win the corresponding formula game, and Konane is PSPACE-hard. As before, Konane is clearly also in PSPACE, and therefore PSPACE-complete. □

5. Cross Purposes

Cross Purposes was invented by Michael Albert, and named by Richard Guy, at the Games at Dalhousie III workshop, in 2004. It was introduced to the author by Michael Albert at the 2005 BIRS Combinatorial Game Theory Workshop.

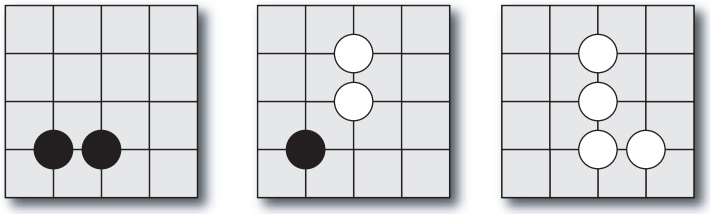


Figure 10. An initial Cross Purposes configuration, and two moves.

Cross Purposes is a kind of two-player version of the popular puzzle Tipover, which is NP-complete [9]. It is easy to construct many interesting combinatorial game values from Cross Purposes positions.

Cross Purposes rules. Cross Purposes is played on the intersections of a Go board, with black and white stones. In the initial configuration, there are some black stones already on the board. A move consists of replacing a black stone with a pair of white stones, placed in a row either directly above, below, to the left, or to the right of the black stone; the spaces so occupied must be vacant for the move to be made. See Figure 10. The idea is that a stack of crates, represented by a black stone, has been tipped over to lie flat. Using this idea, we describe a move as *tipping* a black stone in a given direction.

The players are called *Vertical* and *Horizontal*. Vertical moves first, and play alternates. Vertical may only move vertically, up or down; Horizontal may only move horizontally, left or right. All the black stones are available to each player to be tipped, subject to the availability of empty space. The first player unable to move loses.

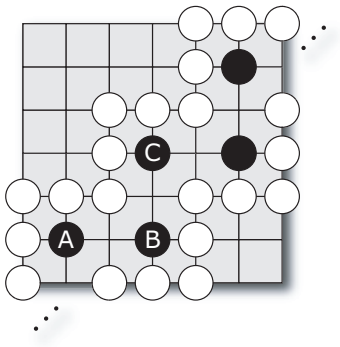
5.1. PSPACE-completeness. The Cross Purposes construction largely follows those used for Amazons and Konane; we build the necessary gadgets to force the two players to effectively play a formula game.

One new challenge in constructing the gadgets is that each player may only directly move either horizontally or vertically, but not both. Yet, for formula game gadgets to work, one player must be able to direct signals two dimensionally. We solve this problem by restricting the moves of Horizontal so that, after the variable selection phase, his possible moves are constrained so as to force him to cooperate in Vertical's signal propagation. (We assume that the number of variables is even, so that it will be Vertical's move after the variable selection phase.) An additional challenge is that a single move can only empty a single square, enabling at most one more move to be made, so it is not obviously possible to split a signal. Again, we use the interaction of the two players to solve this problem.

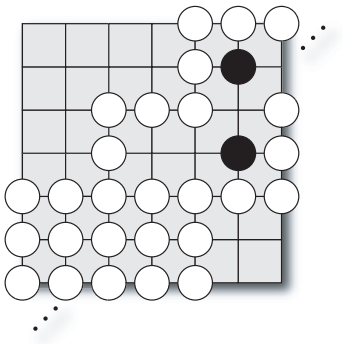
We do not need a supply of extra moves at the end, as used for Amazons and Konane; instead, if Vertical can win the formula game, and correspondingly activate the final AND gadget, then Horizontal will have no move available, and lose. Otherwise, Vertical will run out of moves first, and lose.

Basic wiring. Signals flow diagonally, within surrounding corridors of white stones. A *wire* is shown in Figure 11(a). Suppose that Vertical tips stone A down, and suppose that Horizontal has no other moves available on the board. Then his only move is to tip B left. This then enables Vertical to tip C down. The result of this sequence is shown in Figure 11(b).

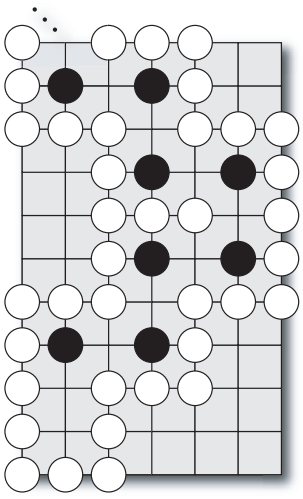
The turn gadget is shown in Figure 11(c); its operation is self-evident. Also



(a) Wire



(b) Wire, after three moves



(c) Turn, free input

Figure 11. Cross Purposes wiring.

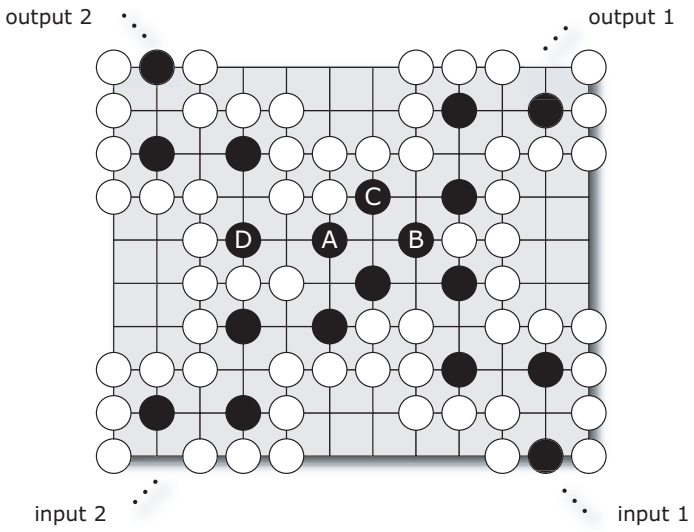


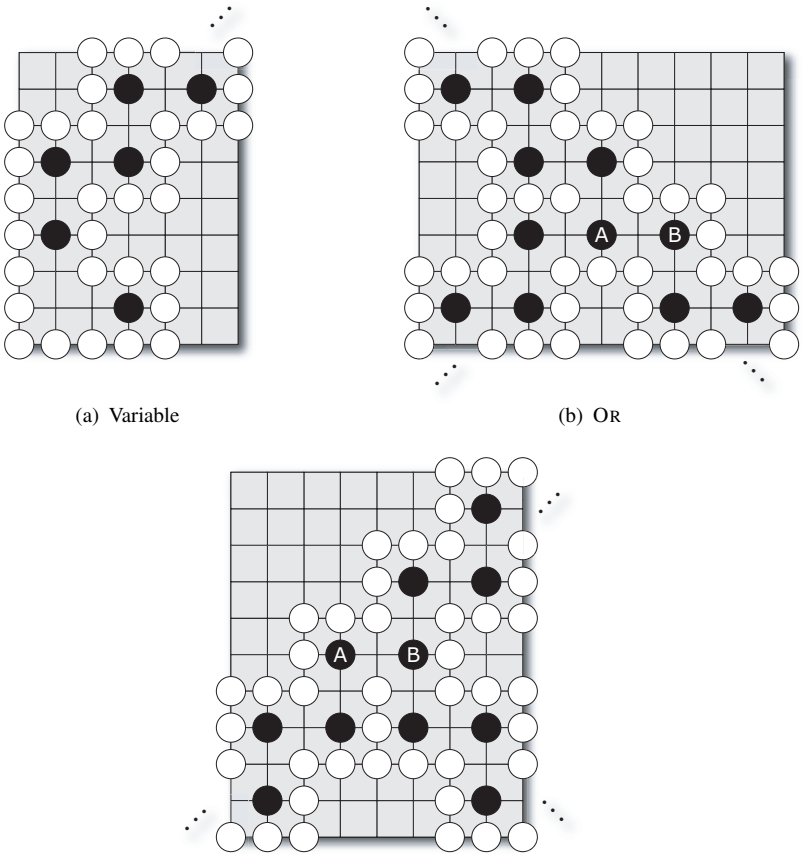
Figure 12. Cross Purposes conditional gadget.

shown in Figure 11(c) is a *free input* for Vertical: he may begin to activate this wire at any time. We will need free inputs in a couple of later gadgets.

Conditional gadget. As with Konane, a single *conditional gadget*, shown in Figure 12, serves the role of split, parity adjustment, and AND. A signal arriving along input 1 may only leave via output 1, and likewise for input 2 and output 2; these pathways are ordinary turns embedded in the larger gadget. However, if Vertical attempts to use pathway 2 before pathway 1 has been used, then after he tips stone A down, Horizontal can tip stone B left, and Vertical will then have no local move. But if pathway 1 has already been used, stone B is blocked from this move by the white stones left behind by tipping C down, and Horizontal has no choice but to tip stone D right, allowing Vertical to continue propagating the signal along pathway 2.

Split, parity. As with Konane, if we give Vertical a free input to the wire feeding input 2 of a conditional gadget, then both outputs may activate if input 1 activates. This splits the signal arriving at input 1.

If we don't use output 1, then this split configuration also serves to propagate a signal from input 1 to output 2, with altered positional parity. This enables us to match signal parities as needed at the gadget inputs and outputs. We must be careful with not using outputs, since we need to ensure that Vertical has no free moves anywhere in the construction; unlike in the previous two constructions, in Cross Purposes, there is no extra pool of moves at the end, and every available move within the layout counts. However, blocking an output is easy to arrange;



(a) Variable (b) OR

(c) Choice

Figure 13. Cross Purposes logic gadgets.

we just terminate the wire so that Horizontal has the last move in it. Then Vertical gains nothing by using that output.

Logic. The *variable* gadget is shown in Figure 13(a). If Vertical moves first in a variable, he can begin to propagate a signal along the output wire. If Horizontal moves first, he will tip the bottom stone to block Vertical from activating the signal.

The AND gadget is a conditional gadget with output 1 unused. By the properties of the conditional gadget, output 2 may activate only if both inputs have activated.

The OR gadget is shown in Figure 13(b). The inputs are on the bottom; the output is on the top. Whether Vertical activates the left or the right input, Horizontal will be forced to tip stone A either left or right, allowing Vertical

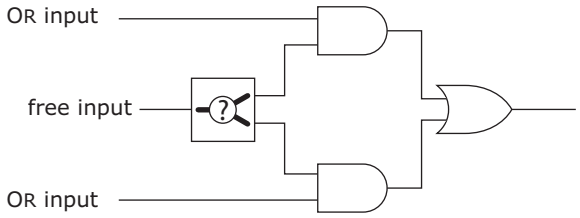


Figure 14. Protected OR.

to activate the output. Here we must again be careful with available moves. Suppose Vertical has activated the left input, and the output, of an OR. Now what happens if he later activates the right input? After he tips stone B down, Horizontal will have no move; he will already have tipped stone A left. This would give Vertical the last move even if he were unable to activate the final AND gadget; therefore, we must prevent this from happening. We will show how to do so after describing the choice gadget.

Choice. For the generic crossover to work, we need a choice gadget. As with Amazons and Konane, the existing OR gadget suffices, if we reinterpret it. This time the gadget must be rotated. The rotated version is shown in Figure 13(c). The input is on the left, and the outputs are on the right. When Vertical activates the input, and tips stone A down, Horizontal must tip stone B left. Vertical may then choose to propagate the signal to either the top or the bottom output; either choice blocks the other.

Protecting the OR Inputs. As mentioned above, we must ensure that only one input of an OR is ever able to activate, to prevent giving Vertical extra moves. We do so with the circuit shown in Figure 14. Vertical is given a free input to a choice gadget, whose output combines with one of the two OR input signals in an AND gadget. Since only one choice output can activate, only one AND output, and thus one OR input, can activate. Inspection of the relevant gadgets shows that Vertical has no extra moves in this construction; for every move he can make, Horizontal has a response.

Winning. We will have an AND gadget whose output may be activated only if the formula is true under the chosen assignment. We terminate its output wire with Vertical having the final move. If he can reach this output, Horizontal will have no moves left, and lose. If he cannot, then since Horizontal has a move in reply to every Vertical move within all of the gadgets, Vertical will eventually run out of moves, and lose.

THEOREM 4. *Cross Purposes is PSPACE-complete.*

PROOF. Given a positive CNF formula A, we construct a corresponding Cross Purposes position, as described above. As before, the reduction is clearly polynomial. Also as before, Vertical may activate a particular AND output, and thus gain the last move, just when he can win the formula game on A.

Therefore, a player may win the Cross Purposes game if and only if he may win the corresponding formula game, and Cross Purposes is PSPACE-hard. As before, Cross Purposes is clearly also in PSPACE, and therefore PSPACE-complete. □

6. Conclusion

We have shown that generalized versions of Amazons, Konane, and Cross Purposes are PSPACE-complete, indicating that it is highly unlikely that an efficient algorithm for optimal play exists for any of them. Their hardness is also additional evidence, if any were needed, that the games are interesting – they are sufficiently rich games to represent abstract computations.

Additionally, we have demonstrated a simple proof technique for showing planar, two-player, bounded move games hard. The generic crossover, in particular, seems likely to make further proofs along these lines easier. It would be interesting to revisit some classic game hardness results, to see whether the proofs can be simplified with these techniques.

Acknowledgments

We thank Michael Albert for introducing us to Cross Purposes, and for his invaluable assistance in constructing the Cross Purposes signal propagation mechanism.

References

[1] Elwyn R. Berlekamp. Sums of $N \times 2$ Amazons. In *Lecture Notes - Monograph Series*, volume 35, pages 1–34. Institute of Mathematical Statistics, 2000.

[2] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways, 2nd edition*. A. K. Peters, Ltd., Wellesley, MA, 2001–2004.

[3] Michael Buro. Simple Amazons endgames and their connection to Hamilton circuits in cubic subgrid graphs. In *Proceedings of the 2nd International Conference on Computers and Games*, Lecture Notes in Computer Science, Hamamatsu, Japan, October 2000.

[4] Alice Chan and Alice Tsai. $1 \times n$ konane: a summary of results. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 331–339, 2002.

[5] Michael D. Ernst. Playing Konane mathematically: A combinatorial game-theoretic analysis. *UMAP Journal*, 16(2):95–121, Spring 1995.

[6] Aviezri S. Fraenkel and David Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n. *Journal of Combinatorial Theory*, Series A, 31:199–214, 1981.

[7] Timothy Furtak, Masashi Kiyomi, Takeaki Uno, and Michael Buro. Generalized amazons is PSPACE-complete. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 132–137. Professional Book Center, 2005.

[8] Robert A. Hearn. Amazons is PSPACE-complete. Manuscript, February 2005. http://www.arXiv.org/abs/cs.CC/0008025.

[9] Robert A. Hearn. Tipover is NP-complete. *Mathematical Intelligencer*, 28(3):10–14, 2006.

[10] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, October 2005.

[11] Richard Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer*, 22(2): 9–15, 2000.

[12] Jens Lieberum. An evaluation function for the game of Amazons. *Theoretical Computer Science*, 349(2):230–244, December 2005.

[13] Martin Müller and Theodore Tegos. Experiments in computer Amazons. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 243–257. Cambridge University Press, 2002.

[14] Stefan Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15:167–191, 1981.

[15] J. M. Robson. The complexity of Go. In *Proceedings of the IFIP 9th World Computer Congress on Information Processing*, pages 413–417, 1983.

[16] J. M. Robson. N by N Checkers is EXPTIME complete. *SIAM Journal on Computing*, 13(2):252–267, May 1984.

[17] Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16:185–225, 1978.

[18] Raymond Georg Snatzke. New results of exhaustive search in the game Amazons. *Theor. Comput. Sci.*, 313(3):499–509, 2004.

ROBERT A. HEARN
NEUKOM INSTITUTE FOR COMPUTATIONAL SCIENCE, DARTMOUTH COLLEGE
SUDIKOFF HALL, HB 6255
HANOVER, NH 03755
UNITED STATES
robert.a.hearn@dartmouth.edu