Ataxx for Wolfe's Toolkit

Karl Chen*

December 14, 2002

Abstract

The game "Ataxx" was implemented for David Wolfe's Gamesman's Toolkit[1]. It is available at
`http://hkn.eecs.berkeley.edu/~quarl/cs294/ataxx.tar.gz`.

1 Ataxx background

Ataxx is a game for two players on a rectangular grid. The board starts with at least one `X` (black, left) and one `O` (white, right) piece. Players alternate either *extending* or *jumping* an existing piece. An extension (also called a *growth*) is played by creating a new piece of one's color next to an existing piece. The original piece in this case stays on the board. A piece can *jump* by moving to a new location that is two positions away. Allowed moves can be visualized thus:

```
j j j j j
j g g g j
j g . g j
j g g g j
j j j j j
```

where `g` denotes a valid *growth* destination from the center, and `j` denotes a valid *jump* destination from the center.

After a new place is placed, whether through extension or jumping, all adjacent opposing pieces flip to the color of the moving player.

One common starting position is

*email: `quarl@hkn.eecs.berkeley.edu`

[1] `http://www.gac.edu/~wolfe/games/`

```
o . . . . . x
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
x . . . . . o
```

Some variations of Ataxx contain *blocks* (we'll color them green) in the starting position, which never move or flip color.

If one player cannot move, then his turn is simply skipped. This deviation from normal form complicates game evaluation and is addressed below. The game ends when no more moves may be made; in general the board will be filled. (Hypothetical exceptions occur only if non-partisan *blocks* "block" off a whole portion of the board from jumping. These aren't interesting nor even possible for small boards.)

1.1 History

Ataxx was invented in 1988 by Dave Crummack and Craig Galley as a derivative of the game Reversi. It is also called Infection (on the Amiga, Spectrum (Sinclair), Amstrad CPC, PC, Commodore 64 and Atari ST) and Spot (for PC, Nintendo, Nintendo Gameboy). A derivative of this game played on a hexagonal grid is called Hexxagon.

2 Implementation

Ataxx was implemented as a C++ module for Wolfe's Toolkit in the same spirit as other games such as Konane and Domineering. The code spans 1000+ lines and would have been much larger in C and probably less efficient if programmed in the style of the core Gamesman's Toolkit.

2.1 Interface

The main entry point `ataxx()` inputs an Ataxx position from standard input and returns its value. It is invoked from the Gamesman's Toolkit commandline with "`ataxx`". There are also options for displaying more verbose output and checking game values; more information is available through "`help ataxx`".

Entering a position on the console is similar to other Gamesman's Toolkit games, but much more robust (a few simple lines of error checking can save hour of headaches from just entering positions!). "`X`"s indicate black/blue/left pieces and "`O`"s indicate white/red/right pieces; green blocks may be placed with "`#`"; spaces denote empty squares. The size of the board is determined by the input size. Input may contain borders consisting of the ASCII characters "`+`", "`-`", and "`|`". These are allowed for aesthetics and input clarity; they do not affect the position. Unlike Konane, boards must be bounded on all sides no matter their size, since the game ends only when the board is filled. Here is an example input sequence:

```
> ataxx   Start an Ataxx position
+---+
|x  |     The "+"s are optional but must be consistent if present.
|   |
|  o|
+---+
```

Pressing return at the end of the board terminates it and begins evaluation.

2.2 Bitboards

Each square in a position can have one of four values (empty, block, black piece, white piece); thus two bits are required per square. Instead of using the bitboards used by other modules and having a clumsy interface of two bitboards per position (and the original C code is inefficient: board positions are hashed by first converting to a linked list), I implemented a new "double bit board" class to handle this. It is optimized with reference counting so that each position uses $(2 * n_{\text{squares}}/8 + 4)$ bytes rounded up to the nearest word (two bits per square, plus one word for size and reference counting information).

2.3 Hashing

Positions are cached using hash tables, but not using Wolfe's global hash table. The global hash table is a bad idea because it simply adds more collisions with data that is known to be different. The custom hash tables also allow us to do more complicated caching such as comparing rotated and rotated/inverted positions. (Rotating a position does not change its value; inverting a position (swapping left/right) negates its value.)

2.4 Scoring

The scoring functions were very complicated to implement because Ataxx is not exactly normal form. The main problem is that if a player cannot move, the other player simply moves until the first can move. We implemented this in our program by redefining an Ataxx "move" option in Gamesman's Toolkit as a sequence of plies until the mover's opponent has options.

Besides this, game evaluation follows the same general algorithm as other games: recursively scan all possible moves. Since we use our own data structures we have an extra chance to simplify games by removing dominated options while recursing, before returning the result - this optimizes quite a bit.

Scoring ends when no player can move (the base case of the recursion). The game value then is the signed integer of the difference in the number of pieces on the board.

2.5 Loops: Ignored

Ataxx is a potentially loopy game. Loops can happen if both players jump back and forth ad infinitum. However, loops only occur if both players desire to continue looping. We will assume here that loops (d.u.d.s) are not useful. The player that has the better position (sans the potential loop) will avoid it. Thus, we do not distinguish between tie and draw games (using Dan Garcia's Gamesman terminology).

2.6 Decomposition

Splitting an Ataxx game into multiple components is difficult. A splittable game does not occur except in rare and contrived cases. Canonically, a player can jump out of his own "territory" to connect two otherwise unconnected areas of play, although these moves probably are not orthodox. It was decided to not attempt to decompose games the way Clobber games were decomposed into connected regions, since the advantage was not big for the size of boards possible to solve given today's memory constraints (the machines on which Gamesman's Toolkit with Ataxx was tested had 512 MB RAM).

3 Results

3.1 Resource limits

Unfortunately, large positions could not be evaluated due to memory limits. Perhaps the code could be improved; maybe hashing could be hacked so that intermediate results are discarded and recalculated later if needed.

Attempting to solve the mostly-empty 7x7 board indicated above is futile. A 7x7 board has about $3^{49} = 2.4 \times 10^{23}$ positions, which is huge. Even a 5x5 board is too large, at $3^{25} = 8.5 \times 10^{23}$ positions (5x5 boards take 50 bits). We also run out of total game value storage space very quickly this way. Given Moore's Law though, perhaps in a few years such positions will be child's play. Currently 4x4 boards begin to push the limit at 200 MB - 300 MB of RAM.

3.2 Some games

Here are some easily verifiable game results:

```
+-----+
|x   o|
+-----+
```

0*: second player wins*

If left starts and extends:

```
+-----+
|xx  o|
+-----+
```

```
+-----+
|xoo  |
+-----+
```

```
+-----+
|xoooo|
+-----+
```

-3

If left starts and jumps:

```
+-----+
|x x o|
+-----+

+-----+
|x ooo|
+-----+

+-----+
|xxooo|
+-----+
```

-1

```
+---+
|x  |
|   |
|  o|
+---+
```

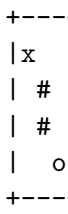
Orthodox: $9|-9$

The canonical value of this game is actually a very long position if written out completely. It looks something like $9, \{9|-9\}|-9, \{9|-9\}$... recursed nine times. Instead of winning right away by extending or jumping to the middle, the moving player could jump or extend to the side and arrive at a similar position. This position also contains d.u.d.s (loops), which we ignore as discussed above.

```
+----+
|x #o|
| ## |
+----+
```

Orthodox: $5|1||-5$

Here, right can win the game easily by jumping left and taking over the board, scoring -5. Left can only block him off by extending into that square; then both players simply fill up their territory.



Orthodox: $10| - 10$

Orthodox: $8| - 8$

It is likely that for all symmetric positions past a certain size, the game is some combination of switches with mean zero. If the position is open (no blocks) then the game value is likely a simple switch. The temperature will vary up to the number of empty squares.

4 Possible Extensions

Ataxx and other processor and memory intensive games in Gamesman's Toolkit could be implemented to run on parallel systems. Coordination between processors would be non-trivial because game values are represented by integer indices into the game hash table; however, if multiple processors each had their separate tables then this easily solves the memory problems simply by partitioning the data space. Only the single result of each computation is needed (results of child recursions are not needed). Overall, implementation would not be difficult. In fact we could even use the High Performance Computing term "embarrassingly parallelizable" here; the obvious implementation would be to use MPI to broadcast and gather data from multiple nodes.

5 References

1. Elwyn Berlekamp, John Conway, & Richard Guy: Winning Ways (vol. I), second edition, A K Peters Ltd, 2001.

2. Ataxx history: `http://www.pressibus.org/ataxx/gen/gborigines.html`