A Networked Awari Referee: Specification

Bart Massey
August 10, 2000

The many variants of the African game Awari are among the oldest known games of intellectual skill. In this document, we will describe a networked server to which human and computer players can connect to play the game in a refereed fashion.

1 The Game Of Awari

The rules of awari are complicated and have many variants. This discussion is based on the excellent summary of Schaeffer et al. [3].

Awari is a two player game, played by players conventionally designated as north and south.

\[ \text{PLAYER ::= north} \mid \text{south} \]

\[ \text{opponent : PLAYER \rightarrow PLAYER} \]

\[ \text{opponent north = south} \]

\[ \text{opponent south = north} \]

The board, shown in figure 1, consists of 12 pits into which a number of stones are placed. In addition, two end pits, or awari, contain stones captured by the two sides during play. Starting from the NE corner and moving counterclockwise, the pits are named by the letters a through f, first in lowercase and then in uppercase. This convention is sensible, since the fundamental sowing operation described below moves counterclockwise.

\[ \text{PIT ::= a} \mid b \mid c \mid d \mid e \mid f \mid A \mid B \mid C \mid D \mid E \mid F \]

\[ \text{pit\_order : seq PIT} \]

\[ \text{next : PIT \rightarrow PIT} \]

\[ \text{prev : PIT \rightarrow PIT} \]

\[ \text{pit\_order} = (a, b, c, d, e, f, A, B, C, D, E, F) \]

\[ \forall p : \text{PIT} \quad \text{next} p = \text{pit\_order}((\text{pit\_order}^{-1}) p \mod 12 + 1) \]

\[ \text{prev} = \text{next}^{-1} \]

The board starts with 48 stones, and stones are never lost from the game: all 48 must always be somewhere. It is thus useful to talk about the the total contents of a set of pits: more generically, about the total sum over the range of some relation.
\[
X
\]
\[
\Sigma : (X \leftrightarrow \mathbb{N}) \rightarrow \mathbb{N}
\]
\[
\Sigma \emptyset = 0 \quad \forall r : X \leftrightarrow \mathbb{N}; \quad x : X; \quad n : \mathbb{N} \mid (x \mapsto n) \in r \bullet \quad \Sigma r = n + \Sigma (r \setminus \{x \mapsto n\})
\]

The state of the board at any given time is essentially just the number of stones in each pit. The players alternate moves.

\[
BOARD \leftrightarrow PIT \rightarrow \mathbb{N}
\]
\[
SCORE \leftrightarrow PLAYER \rightarrow \mathbb{N}
\]

\begin{verbatim}
AwariBoard
board : BOARD
score : SCORE
to_move : PLAYER
\Sigma board + \Sigma score = 48
\end{verbatim}

Each pit initially contains 4 stones, as shown in figure 1. The south player moves first.

\begin{verbatim}
InitAwariBoard
AwariBoard'
board = PIT \times \{4\}
score' = PLAYER \times \{0\}
to_move' = south
\end{verbatim}

The players alternate in sowing the stones in a pit of their choice on their side of the board. (north owns pits a-f, and south pits A-F.)

\[
side : PLAYER \mapsto \mathbb{P} PIT
\]
\[
side north = \text{ran}(1 \leq \text{pit_order})
\]
\[
side south = PIT \setminus (\text{side north})
\]

A pit may be sown if it contains one or more stones: the stones are removed from the pit, and placed one at a time into subsequent pits, moving around the board in counterclockwise order. The original pit is
skipped whenever it is encountered.

\[
\text{next\_skip} : \text{PIT} \rightarrow \text{PIT} \rightarrow \text{PIT}
\]

\[
\forall p : \text{PIT} \bullet \text{next\_skip } p = \text{next } \oplus \{\text{prev } p \mapsto \text{next } p\}
\]

Thus, the sequence of pits to be sown is a prefix of the infinite cyclic sequence of pits which repeats every 11 elements after skipping the sown pit. We call this infinite sequence the trace of the sown pit.

\[
\text{trace} : \text{PIT} \rightarrow \text{seq } \text{PIT}
\]

\[
\forall p : \text{PIT} \bullet \text{trace } p = (\lambda n : \mathbb{N}_1 \bullet (\text{iter } n (\text{next\_skip } p)) ) p
\]

Thus we can sow the stones by emptying the pit to be sown and adding a stone to each other pit for each time it is hit by sowing. The last pit filled is the first pit we will examine for possible capture below.

---

\[
\text{Sow}
\]

\[
\Delta \text{AwrariBoard}
\]

\[
\text{move} : \text{PIT}
\]

\[
\text{to\_capture} : \text{PIT}
\]

\[
\forall p : \text{PLAYER} \bullet \text{score } p < 25
\]

\[
\text{move} \in \text{side to\_move}
\]

\[
\text{board } \text{move} > 0
\]

\[
\text{to\_capture} = (\text{trace } \text{move})(\text{board } \text{move})
\]

\[
\text{board}' =
\]

\[
\text{let } \text{emptied } = \text{board } \oplus \{\text{move } \mapsto 0\};
\]

\[
\text{sown } = (1 \ldots (\text{board } \text{move})) \triangleleft (\text{trace } \text{move}) \bullet
\]

\[
(\lambda p : \text{PIT} \bullet (\text{emptied } p) + \#(\text{sown } \triangleright \{p\}) )
\]

\[
\text{score}' = \text{score}
\]

\[
\text{to\_move}' = \text{to\_move}
\]

---

When sowing a pit, if the last pebble placed makes a group of two or three, then that pit’s stones are captured and scored by placing in the capturing player’s awari. If the previous pit then contains a group of two or three stones, these stones are also captured, and so forth. Thus, the set of pits which are captured is the set of pits on the opponent’s side of the board reachable by captures from the last pit sown: This set is given by by the range of the transitive closure of the next relation restricted to the capturable elements.

\[
[X]
\]

\[
\text{reachable} : (X \leftrightarrow X) \rightarrow \mathbb{P} X \rightarrow \mathbb{P} X
\]

\[
\forall r : X \leftrightarrow X ; \text{xs} : \mathbb{P} X \bullet
\]

\[
\text{reachable } r \text{ xs } = \text{ran}(\text{xs } \triangleleft (r'))
\]
The sowing of stones to capture all stones on the opponent’s side of the board is known as a **grand slam**, clean sweep, or grand coup. Normally, a grand slam ends the game, capturing all stones remaining on the board. The rules governing the grand slam vary widely, however: Schaeffer *et al.* [3] lists a number of variations:

a) Stones may not be sown for a grand slam (unless no other move is possible).

b) The stones may be sown for a grand slam, but no capture results.

c) The stones may be sown for a grand slam, but the last pit is not captured.

d) The stones may be sown for a grand slam, but only the first pit is captured.

e) The stones may be sown for a grand slam, and all captures happen: the remaining stones on the board are awarded to the opponent.

This paper will consider only variant (e), which will be used in the August 2000 Mind Sports Olympiad [2] computer competition. The 1990–1992 Computer Olympiads [1] used variation (a). Many human players prefer the simpler version.

There are a number of possible outcomes of a (attempt to) move.

\[\text{RESULT} ::= \text{win} ⟨\text{PLAYER}⟩ \mid \text{draw} \mid \text{not\_done} \mid \text{illegal\_move}\]

\[\forall s : \text{SCORE} \cdot \text{winner} s =\]

\[\begin{array}{l}
\text{if } s \text{ north } > s \text{ south } \\
\quad \text{then win north} \\
\quad \text{else if } s \text{ south } > s \text{ north } \\
\quad \text{then win south} \\
\quad \text{else draw}
\end{array}\]

A game may be ended by a player being unable to move, in which case the remaining stones on the board belong to the opponent.
However, a player must leave the opponent with a legal move at the start of their turn, if it is possible to do so.

A game will also be ended by repetition of position. (This is generally only true of computer play. In human play, it is more common to end by agreement of both sides.) The most common rule here is that each player captures the stones on their side of the board. One popular variant, used in the 1990-1992 Computer Olympiad, is to not count the stones left on the board at the end. To keep track of repetitions, it is necessary to keep track of the set of positions seen so far.

### Positions

\[ \text{positions : } \mathcal{P} \text{AvariBoard} \]

To summarize the variant of interest here: a move in the game consists of altering the board and recording the resulting board position. Moves alternate between players.

### AvariMove

ΔAvariBoard

ΔPositions

\[ \text{opp_stones : BOARD } \to \mathbb{N} \]

result : RESULT

\[ \text{positions}' = \text{positions} \cup \{ \theta\text{AvariBoard} \} \]

\[ \forall b : \text{BOARD} \bullet \text{opp_stones } b = \Sigma(\text{side(oppponent to_move)}) < b \]

If either player has 25 or more stones at the start of their turn, the player with the most stones wins, and the game is over.

### WinMargin

AvariMove

\[ \exists p : \text{PLAYER} \bullet \text{score } p \geq 25 \]

\[ \text{score}' = \text{score} \]

\[ \text{result} = \text{winner score}' \]

If the sowing is impossible, or the position is a repetition of a previous one, each player claims the stones on their side of the board, and the game is over.

### NoMove

AvariMove

ClaimStones

\[ \forall p : \text{PLAYER} \bullet \text{score } p < 25 \]

\[ \theta\text{AvariBoard} \in \text{positions} \lor \]

\[ \Sigma(\text{side to_move } < \text{board}) = 0 \]

\[ \text{result} = \text{winner score}' \]

If the opponent’s pits are all empty, the player must sow a pit which leaves the opponent a move if possible.
\[
\begin{aligned}
\text{AwariMove} & \quad \text{NormalMove} \\
\text{Sow} \equiv \text{Capture} & \\
\forall b : \text{BOARD} \quad \text{app\_stones\_board} = 0 \Rightarrow \text{app\_stones\_board}' > 0 \\
\text{result} = \text{not\_done}
\end{aligned}
\]

A grand slam capture forfeits all stones still on the board at the end of the turn.

\[
\begin{aligned}
\text{ForfeitStones} & \\
\text{score}, \text{score}' : \text{SCORE} \\
\text{board} : \text{BOARD} \\
\forall p : \text{PLAYER} \quad \text{score}' p = \text{score} p + \Sigma(\text{side}(\text{opponent} p) < \text{board})
\end{aligned}
\]

A grand slam is the capture the rest of the opponent’s stones.

\[
\begin{aligned}
\text{GrandSlam} & \\
\text{AwariMove} & \\
\text{Sow} \equiv \text{Capture} \equiv \text{ForfeitStones} \\
\text{app\_stones\_board} > 0 \\
\text{app\_stones\_board}' = 0 \\
\text{result} = \text{winner score}'
\end{aligned}
\]

A move can only be illegal if a move is available and neither side has won. An illegal move is from a pit on the wrong side of the board, a pit with no stones, or fails to feed an opponent with stones when necessary. (Note that this schema does not assume \text{NormalMove} is deterministic for a given move, although we know it to be.)

\[
\begin{aligned}
\text{IllegalMove} & \\
\text{AwariMove} & \\
\forall p : \text{PLAYER} \quad \text{score} p < 25 \\
\exists p : \text{side\_to\_move} \quad \text{board} p > 0 \\
\quad \expC{\forall \text{move} \neq \text{side\_to\_move} \vee \text{board} \text{move} = 0 \vee \\
\quad \text{app\_stones\_board} = 0 \quad \land \\
\quad (\forall m : \text{NormalMove} | m.\text{move} = \text{move} \quad \\
\quad \text{app\_stones\_m.board}' = 0) \quad \land \\
\quad (\exists m : \text{NormalMove} | m.\text{move} \neq \text{move} \quad \\
\quad \text{app\_stones\_m.board}' > 0)}
\end{aligned}
\]

The referee reports whether a move continues the game, ends the game, or is illegal.

\[
\text{AwariRef} \equiv \text{NormalMove} \vee \text{WinMargin} \vee \\
\text{NoMove} \vee \text{GrandSlam} \vee \text{IllegalMove}
\]

The game starts with the initial board, and no positions yet seen.
Table 1: Greeting

<table>
<thead>
<tr>
<th>name</th>
<th>response</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>greeting</td>
<td>000 (version-number)</td>
<td>Greeting message</td>
</tr>
</tbody>
</table>

Table 2: Initial Requests

<table>
<thead>
<tr>
<th>name</th>
<th>request</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>want_north</td>
<td>(version) player north (optional-name)</td>
<td>Will play north</td>
</tr>
<tr>
<td>want_south</td>
<td>(version) player south (optional-name)</td>
<td>Will play south</td>
</tr>
<tr>
<td>want_side</td>
<td>(version) player ? (optional-name)</td>
<td>Will play either</td>
</tr>
<tr>
<td>want_observe</td>
<td>(version) observer (optional-name)</td>
<td>Will observe</td>
</tr>
</tbody>
</table>

InitAwari

InitAwariBoard

Positions

positions = ∅

2 Server

The Awari server listens on a port in the range 29046...29056 for a connection.\footnote{All numbers in this section will be base 30 (decimal) unless otherwise stated.} All input to the server will be in the form of ASCII text lines, terminated with a CR character (ASCII code 13). All server responses will be in the form of ASCII text lines, terminated with a CR and then an LF character (ASCII code 10). Responses will begin with a 3-digit numerical code, and be followed by whitespace and a (non-standard) explanatory text message. Requests and responses not currently implemented by the server will have their identifier in italics; those implemented will have boldface identifiers.

Any number of observers may connect to the server, as well as the two players. The server will always be in a state determined by the input it has seen. This state will determine which messages it will accept, and which responses it will return. The server may be in different states for different connections: it must synchronize the connections at key points.

\[
\begin{align*}
STATE & := \text{initial} | \text{seated} | \text{playing} | \text{done} \\
ENTITY & := \text{player}⟨\text{PLAYER}⟩ | \text{observer}⟨\text{N}⟩ \\
observer & \in \mathbb{P}(\text{seq ENTITY}) \\
\text{cstate} & : ENTITY \rightarrow STATE
\end{align*}
\]

Upon connection to the server, an entity will receive a greeting in the form indicated by Table 1. The version number is a pair of integers separated by a decimal point. This document describes version 0.9.

The initial message sent to the server must be as shown in Table 2. Responses are shown in Table 3. The ⟨optional-name⟩ is an optional double-quoted string (with the convention that two consecutive double-quotes “” inside the string escape to a single double-quote ”) of up to 31 characters used to identify the entity. The version number is as above, and is used to identify the client version. The client version must be no greater (under the usual ordering) than the server version. If both players indicate “player ?”, the server will randomly select a north and south player.

Once both a north player and a south player have connected, the setup phase will be over. The server may indicate to each entity the other entities involved, by sending messages as shown in Table 4. ⟨name⟩ and
<table>
<thead>
<tr>
<th>name</th>
<th>response</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>seat_granted</td>
<td>100</td>
<td>Request accepted</td>
</tr>
<tr>
<td>seat granted</td>
<td>101 secs</td>
<td>Request accepted with time controls</td>
</tr>
<tr>
<td>seat_jaken</td>
<td>19x</td>
<td>Request not accepted</td>
</tr>
<tr>
<td>seat_full</td>
<td>191</td>
<td>Other player holds requested side</td>
</tr>
<tr>
<td>seat_private</td>
<td>192</td>
<td>There are already two players</td>
</tr>
<tr>
<td>seatIllegal</td>
<td>193</td>
<td>Cannot observe</td>
</tr>
<tr>
<td>seat_garbled</td>
<td>199</td>
<td>Request not understood</td>
</tr>
</tbody>
</table>

<optional-name> are double-quoted strings as described below. <number> is a decimal number. (All entities should be prepared to deal with numbers up to 3 decimal digits, and to discard an arbitrary number).

The server will then signal the start of game by sending a message to each connected entity, as shown in Table 5.

After this, the server will accept moves from players in alternation, of the form shown in Table 6, where the <move number> is a standard decimal number indicating the ply of the move, the <ellipses-if-north> will be the string ... for a move by north and the empty string for a move by south, and <move> will be a move in the notation described above.

Instead of a move, the following inputs may also be accepted as shown in Table 7. Responses to actions are shown in Table 8.

After each accepted action, a message will be sent to each connected entity, as shown in Table 9. Upon termination of the game, the server will close all connections.

The draw protocol is as one would expect: a draw may be offered by a player whose turn it is to move. If it is rejected by the other player, it may not be offered again until the next move.

### References


<table>
<thead>
<tr>
<th>name</th>
<th>response</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_north</td>
<td>341 &lt;name&gt;</td>
<td>North player is &lt;name&gt;</td>
</tr>
<tr>
<td>config_south</td>
<td>342 &lt;name&gt;</td>
<td>South player is &lt;name&gt;</td>
</tr>
<tr>
<td>config_observer</td>
<td>343 &lt;number&gt; &lt;name&gt;</td>
<td>Observer &lt;number&gt; is &lt;name&gt;</td>
</tr>
<tr>
<td>config_observer</td>
<td>344 &lt;number&gt;</td>
<td>There are &lt;number&gt; observers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>response</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>role_north</td>
<td>351</td>
<td>You will play north</td>
</tr>
<tr>
<td>role_south</td>
<td>352</td>
<td>You will play south</td>
</tr>
<tr>
<td>role_observer</td>
<td>353</td>
<td>You will observe</td>
</tr>
</tbody>
</table>
### Table 6: Move Syntax

<table>
<thead>
<tr>
<th>name</th>
<th>request</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_move</td>
<td>⟨move number⟩ ⟨ellipses-if-north⟩ ⟨move⟩</td>
<td>Make a move</td>
</tr>
</tbody>
</table>

### Table 7: Alternatives To Moving

<table>
<thead>
<tr>
<th>name</th>
<th>request</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_resign</td>
<td>resign</td>
<td>Player resigns</td>
</tr>
<tr>
<td>action_draw_req</td>
<td>draw?</td>
<td>Player offers a draw</td>
</tr>
<tr>
<td>action_draw_mak</td>
<td>draw</td>
<td>Player accepts a draw</td>
</tr>
<tr>
<td>action_draw_nak</td>
<td>nodraw</td>
<td>Player refuses a draw</td>
</tr>
</tbody>
</table>

### Table 8: Responses To Actions

<table>
<thead>
<tr>
<th>name</th>
<th>response</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>result_continue</td>
<td>20x</td>
<td>Action accepted</td>
</tr>
<tr>
<td>result_continue</td>
<td>200</td>
<td>Continue playing</td>
</tr>
<tr>
<td>result_continue</td>
<td>207 ⟨sees⟩</td>
<td>Continue with time left</td>
</tr>
<tr>
<td>result_win</td>
<td>201</td>
<td>You win</td>
</tr>
<tr>
<td>result_lost</td>
<td>202</td>
<td>You lose</td>
</tr>
<tr>
<td>result_drawn</td>
<td>203</td>
<td>You draw</td>
</tr>
<tr>
<td>result_resigned</td>
<td>204</td>
<td>Resignation accepted</td>
</tr>
<tr>
<td>result_drawn req</td>
<td>205</td>
<td>Received draw request</td>
</tr>
<tr>
<td>result_drawn nq</td>
<td>206</td>
<td>Received draw refusal</td>
</tr>
<tr>
<td>29x</td>
<td></td>
<td>Action not accepted</td>
</tr>
<tr>
<td>result Illegal</td>
<td>291</td>
<td>Illegal request</td>
</tr>
<tr>
<td>result Garbled</td>
<td>299</td>
<td>Request not understood</td>
</tr>
<tr>
<td>name</td>
<td>response</td>
<td>meaning</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>status_moves_south</td>
<td>31x</td>
<td>Game continues</td>
</tr>
<tr>
<td>status_moves_north</td>
<td>311 (move-number) (move)</td>
<td>South move</td>
</tr>
<tr>
<td>status_moves_south_tc</td>
<td>312 (move-number) ... (move)</td>
<td>North move</td>
</tr>
<tr>
<td>status_moves_north_tc</td>
<td>313 (move-number) (move) (secs)</td>
<td>South move and time</td>
</tr>
<tr>
<td>status_moves_north_tc</td>
<td>314 (move-number) ... (move) (secs)</td>
<td>North move and time</td>
</tr>
<tr>
<td>status_winsmove_south</td>
<td>321 (move-number) (move)</td>
<td>South wins by move</td>
</tr>
<tr>
<td>status_winsmove_south</td>
<td>322 (move-number) (move)</td>
<td>South loses by move</td>
</tr>
<tr>
<td>status_winsmove_north</td>
<td>323 (move-number) ... (move)</td>
<td>North wins by move</td>
</tr>
<tr>
<td>status_winsmove_north</td>
<td>324 (move-number) ... (move)</td>
<td>North loses by move</td>
</tr>
<tr>
<td>status_drawsmove_south</td>
<td>325 (move-number) (move)</td>
<td>Drawn by South move</td>
</tr>
<tr>
<td>status_drawsmove_north</td>
<td>326 (move-number) ... (move)</td>
<td>Drawn by North move</td>
</tr>
<tr>
<td>status_resigns_north</td>
<td>327</td>
<td>South wins by resignation</td>
</tr>
<tr>
<td>status_resigns_south</td>
<td>328</td>
<td>North wins by resignation</td>
</tr>
<tr>
<td>status_drawagreed</td>
<td>329</td>
<td>Draw accepted</td>
</tr>
<tr>
<td>status_flagfell_north</td>
<td>361</td>
<td>South wins by North time expiring</td>
</tr>
<tr>
<td>status_flagfell_south</td>
<td>362</td>
<td>North wins by South time expiring</td>
</tr>
<tr>
<td>status_reqdraw_south</td>
<td>331</td>
<td>Draw requested by South</td>
</tr>
<tr>
<td>status_reqdraw_north</td>
<td>332</td>
<td>Draw requested by North</td>
</tr>
<tr>
<td>status_refsdra_south</td>
<td>333</td>
<td>Draw refused by South</td>
</tr>
<tr>
<td>status_refsdra_north</td>
<td>334</td>
<td>Draw refused by North</td>
</tr>
<tr>
<td>status_disconnect_south</td>
<td>391</td>
<td>South disconnected</td>
</tr>
<tr>
<td>status_disconnect_north</td>
<td>392</td>
<td>North disconnected</td>
</tr>
<tr>
<td>status_garble</td>
<td>399</td>
<td>Unknown problem</td>
</tr>
</tbody>
</table>