A Networked *Lines Of Action* Referee: Specification

Bart Massey

November 23, 2000

Lines Of Action (LOA) is a fairly obscure but fascinating game of skill. It is played on an ordinary checkerboard, and has something of the feel of Othello or Go. In this document, we will describe a networked server to which human and computer LOA players can connect to play the game in a refereed fashion.

1 The Game Of *Lines Of Action*

The rules of LOA are complicated and have many variants. This discussion is based partly on Dyer's account on his excellent LOA webpage [1], and partly on the Fifth Computer Olympiad (2000) [2] rules [3], the latter being normative in the case of conflicts.

LOA is a two player game, played by players conventionally designated as *white* and *black*.

$PLAYER ::= black \mid white$

$$
\begin{array}{|l}
opponent : PLAYER \rightarrowtail\!\!\!\!\!\twoheadrightarrow PLAYER \\
\hline
opponent\ white = black \\
opponent\ black = white
\end{array}
$$

The board, shown in figure 1, is an ordinary checkerboard (whose squares are conventionally designated using standard algebraic notation).

$DIGIT == 1 \mathinner{\ldotp\ldotp} 8$
$SQUARE == DIGIT \times DIGIT$

At any given point in the game, the board position can be given by noting whether each square is blank or contains a colored checker.

$SQUAREVAL ::= checker\langle\!\langle PLAYER \rangle\!\rangle \mid blank$
$BOARD == SQUARE \rightarrow SQUAREVAL$

$$
\begin{array}{|l}
\hline
\quad LOAPosition \\
board : BOARD \\
to\_move : PLAYER \\
\hline
\end{array}
$$

As shown in Figure 1, each player starts with 12 checkers, placed on the six central edge squares of opposing sides of the board.

$CDIGIT == 2 \mathinner{\ldotp\ldotp} 7$

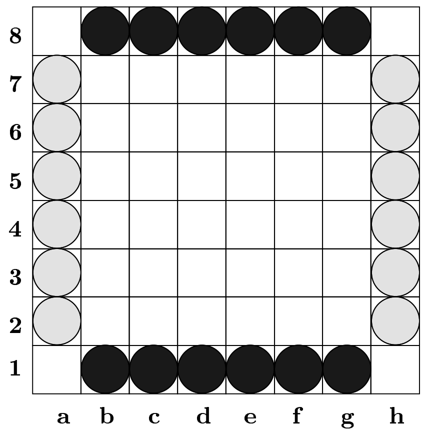

Figure 1: LOA Board In Initial Configuration

$$
\begin{array}{|l}
\hline
\textit{InitLOAPosition} \\
\textit{LOAPosition}' \\
\hline
\textit{board}' = (\textit{SQUARE} \times \{\textit{blank}\}) \oplus \\
\quad (((\{1\} \times \textit{CDIGIT}) \times \{\textit{checker white}\}) \cup \\
\quad ((\{8\} \times \textit{CDIGIT}) \times \{\textit{checker white}\}) \cup \\
\quad ((\textit{CDIGIT} \times \{1\}) \times \{\textit{checker black}\}) \cup \\
\quad ((\textit{CDIGIT} \times \{8\}) \times \{\textit{checker black}\})) \\
\textit{to\_move}' = \textit{black} \\
\hline
\end{array}
$$

The players alternate in moving one of their own checkers a certain distance along its row, column, or diagonal axis ("line of action").

$$
\begin{array}{l}
\textit{DIST} == -8 \mathrel{..} 8 \\
\textit{UNIT} == -1 \mathrel{..} 1 \\
\textit{UVEC} == (\textit{UNIT} \times \textit{UNIT}) \setminus \{(0,0)\} \\
\textit{AXIS} == \textit{SQUARE} \rightarrow \textit{DIST} \rightarrowtail \textit{SQUARE}
\end{array}
$$

$$
\begin{array}{|l}
\textit{line} : \textit{UVEC} \rightarrow \textit{AXIS} \\
\textit{lines} : \mathbb{P}\,\textit{AXIS} \\
\hline
\forall\, ux, uy : \textit{UNIT} \mid (ux, uy) \in \textit{UVEC} \bullet \\
\quad \forall\, x, y : \textit{DIGIT};\ \bullet \\
\quad\quad \forall\, d : \textit{DIST} \mid (x + d * ux, y + d * uy) \in \textit{SQUARE} \bullet \\
\quad\quad\quad \textit{line}\,(ux, uy)\,(x, y)\,d = (x + d, y) \\
\textit{lines} = \textit{line}(\!|\, \textit{UVEC} \,|\!)
\end{array}
$$

A move is illegal unless the number of pieces (including the moving piece) lying along the move axis is equal to the distance moved by the piece. It is also illegal if there is an intervening piece of the other color in a square between the source and destination of the move, or the destination square contains a piece of the same color. (If the destination square contains a piece of opposite color it is captured and removed from the game.)

$MOVE == SQUARE \times SQUARE$

$$
\begin{array}{|l}
\text{— } LOAAction \text{ —} \\
LOAPosition \\
naxis : AXIS \rightarrow SQUARE \rightarrow DIST \\
target : AXIS \rightarrow MOVE \\
moves : \mathbb{P}\, MOVE \\
make\_move : MOVE \rightarrow BOARD \\
\hline
\forall f : AXIS \bullet \\
\quad naxis\, f = (\lambda\, x : SQUARE \bullet \\
\quad\quad \#(\mathsf{dom}(board \rhd \{blank\}) \cap ((f\ x)(\!|DIST|\!)))) \\
\forall f : AXIS;\ x : SQUARE;\ d : DIST \mid \\
\quad d = naxis\, f\ x \land \\
\quad board\ x = checker\ to\_move \land \\
\quad board\ (f\ x\ d) \neq checker\ to\_move \land \\
\quad checker\ (opponent\ to\_move) \notin board(\!|(f\ x)(\!|1 \mathrel{..} (d-1)|\!)|\!) \bullet \\
\quad target\, f = (x, f\ x\ d) \\
moves = target(\!|lines|\!) \\
\forall s, d : SQUARE \bullet \\
\quad make\_move\ (s, d) = board \oplus \{s \mapsto blank\} \oplus \{d \mapsto checker(to\_move)\} \\
\hline
\end{array}
$$

The goal of the game is to make a move that connects all of one's pieces. A player's pieces are connected if there is a path from any piece to any other piece via adjacent horizontal, vertical, or diagonal squares.

$$
\begin{array}{|l}
adjacent : SQUARE \leftrightarrow SQUARE \\
connections : \mathbb{P}\,\mathbb{P}\, SQUARE \\
connected : PLAYER \leftrightarrow BOARD \\
\hline
adjacent = \{\, x1, y1, x2, y2 : DIGIT \mid \\
\quad \mathrm{abs}(x2 - x1) \leq 1 \land \mathrm{abs}(y2 - y1) \leq 1 \bullet \\
\quad (x1, y1) \mapsto (x2, y2) \,\} \\
connections = \{\, p : \mathbb{P}\, SQUARE \mid \\
\quad \forall p' : \mathbb{P}\, p \setminus \{\varnothing\} \setminus \{p\} \bullet (p' \lhd adjacent \rhd (p \setminus p')) \neq \varnothing \,\} \\
\forall p : PLAYER;\ b : BOARD \bullet \\
\quad (p, b) \in connected \Leftrightarrow \mathsf{dom}(b \rhd \{checker(p)\}) \in connections
\end{array}
$$

The Computer Olympiad rules list a number of conditions which can terminate the game. Most of these have other variants which are commonly played, but which will not be considered here. Specifically, the Computer Olympiad rules state that:

1. *The game ends when one player (or both) is connected*, and
   (a) *If just one player is connected, that player is the winner.*
   (b) *If both players are connected, the player who moved last wins.*

   Rules 1a and 1b can be understood together in this context as "The winner is the first player to be connected at either the start or end of their turn."

2. *Repeated positions with the same player to move are drawn.* While the language is somewhat ambiguous, it is assumed that this means any repetition, not just immediate repetition.

3. *If a player has no move, that player loses.* It is also noted that this exception was added after the fact, which suggests an advantage of formal characterization and analysis of tournament game rules.

There are a number of possible outcomes of a player's turn.

$RESULT ::= win\langle\langle PLAYER\rangle\rangle \mid draw \mid not\_done \mid illegal\_move$

Because of rule 2, it is necessary to keep track of all the positions seen in the game. A move in the game consists of altering the board and recording the resulting board position.

$$\begin{array}{|l}
\multicolumn{1}{l}{LOAHistory} \\
positions : \mathbb{P}\, LOAPosition \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{LOARecordHistory} \\
LOAPosition \\
\Delta LOAHistory \\
\hline
positions' = positions \cup \{\theta LOAPosition\} \\
\hline
\end{array}$$

Conditions are checked for the player on move at the start of the turn.

1. If the player is connected, they win.
2. If the current position is a repeat of a previously seen position, they draw.
3. If there is no move available, they lose.

(Note that condition 2 is mutually exclusive with conditions 1 and 3, and the wording of the Computer Olympiad rules implies that condition 1 is supposed to take precedence over condition 3. So all of the special cases are disjoint.)

$$\begin{array}{|l}
\multicolumn{1}{l}{LOAPrecheck} \\
LOAAction \\
LOAHistory \\
precheck : RESULT \\
\hline
precheck = \\
\quad \textbf{if } (to_move, board) \in connected \textbf{ then } win(to_move) \\
\quad \textbf{else if } \theta LOAPosition \in positions \textbf{ then } draw \\
\quad \textbf{else if } moves = \varnothing \textbf{ then } win(opponent\ to_move) \\
\quad \textbf{else } not_done \\
\hline
\end{array}$$

Otherwise, some normal move must be made. In this case, the result of making the move is a win when the position is now connected for the player to move. Otherwise, the game continues.

$LOATurn$

$$
\begin{array}{l}
LOAPrecheck \\
\Delta LOAPosition \\
LOARecordHistory \\
move? : MOVE \\
result! : RESULT \\
\hline
to\_move' = opponent\ to\_move \\
precheck \neq not\_done \Rightarrow \\
\quad result! = precheck \\
precheck = not\_done \land move? \in moves \Rightarrow \\
\quad board' = make\_move\ move? \land \\
\quad result! = \\
\qquad \mathbf{if}\ (to\_move, board') \in connected\ \mathbf{then}\ win(to\_move) \\
\qquad \mathbf{else}\ not\_done \\
precheck = not\_done \land move? \notin moves \Rightarrow \\
\quad result! = illegal\_move
\end{array}
$$

$InitLOATurn$

$$
\begin{array}{l}
InitLOAPosition \\
LOAHistory' \\
\hline
positions' = \{ \theta LOAPosition' \}
\end{array}
$$

Table 1: Greeting

name	response	meaning
greeting	000 LOA ⟨*version-number*⟩	Greeting message

Table 2: Initial Requests

name	request	meaning
want_white	⟨*version*⟩ player white ⟨*optional-name*⟩	Will play white
want_black	⟨*version*⟩ player black ⟨*optional-name*⟩	Will play black
want_side	⟨*version*⟩ player ? ⟨*optional-name*⟩	Will play either
want_observe	⟨*version*⟩ observer ⟨*optional-name*⟩	Will observe

2 Server

The *Lines Of Action* server listens on a port in the range 29057...29067 for a connection.[1] 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.

$$STATE ::= initial \mid seated \mid playing \mid done$$
$$ENTITY ::= player\langle\langle PLAYER\rangle\rangle \mid observer\langle\langle \mathbb{N}_1 \rangle\rangle$$
$$observer \in \operatorname{seq} ENTITY$$

$$\mid cstate : ENTITY \rightarrow\!\!\!\!\!+ \; STATE$$

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 white and black player.

Once both a white player and a black 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 ⟨*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-white*⟩ will

[1] All numbers in this section will be base 10 (decimal) unless otherwise stated.

Table 3: Initial Responses

name	response	meaning
seat_granted	100	Request accepted
seat_granted_tc	101 ⟨*secs*⟩ ⟨*opp-secs*⟩	Request accepted with time controls
	19x	Request not accepted
seat_taken	191	Other player holds requested side
seat_full	192	There are already two players
seat_private	193	Cannot observe
seat_illegal	198	Illegal version number
seat_garbled	199	Request not understood

Table 4: Configuration Messages

name	response	meaning
config_white	341 ⟨*name*⟩	White player is ⟨*name*⟩
config_black	342 ⟨*name*⟩	Black player is ⟨*name*⟩
config_observer	343 ⟨*number*⟩ ⟨*name*⟩	Observer ⟨*number*⟩ is ⟨*name*⟩
config_nobserver	344 ⟨*number*⟩	There are ⟨*number*⟩ observers

be the string ... for a move by white and the empty string for a move by black, 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.

If the participant is an observer, every status message will be followed by two state display messages showing the current state of the game, as in Table 10. The times will be in seconds, and the ⟨*to-move*⟩ value will be either "b", "w", or "." indicating Black, White, or the game is over. The **sdisp_board** message will be immediately followed by 8 lines of 8 printable characters indicating the board state. Each character will be as above: "b", "w", or "." indicating a blank square.

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

[1] Dave Dyer. *Lines Of Action* Home Page. Web document, 2000. URL `http://www.andromeda.com/people/ddyer/loa/loa.html`.

[2] Fifth Computer Olympiad. Web document, 2000. URL `http://www.dcs.qmw.ac.uk/~olympiad/`.

[3] *Lines Of Action* Rules. Web document, 2000. URL `http://www.dcs.qmw.ac.uk/~olympiad/LOA.htm`. See [2].

Table 5: Starting Messages

name	response	meaning
role_white	351	You will play white
role_black	352	You will play black
role_observer	353	You will observe

Table 6: Move Syntax

name	**request**	**meaning**
action_move	⟨*move number*⟩ ⟨*ellipses-if-white*⟩ ⟨*move*⟩	Make a move

Table 7: Alternatives To Moving

name	**request**	**meaning**
action_resign	resign	Player resigns
action_draw_req	draw?	Player offers a draw
action_draw_ack	draw	Player accepts a draw
action_draw_nak	nodraw	Player refuses a draw

Table 8: Responses To Actions

name	**response**	**meaning**
	20x	Action accepted
result_continue	200	Continue playing
result_continue	207 ⟨*secs*⟩	Continue with time left
result_win	201	You win
result_lost	202	You lose
result_drawn	203	You draw
result_resigned	204	Resignation accepted
result_drawreq	205	Received draw request
result_drawnak	206	Received draw refusal
	29x	Action not accepted
result_illegal	291	Illegal request
result_garbled	299	Request not understood

Table 9: Status Messages

name	response	meaning
	31x	Game continues
status_moves_black	311 ⟨*move-number*⟩ ⟨*move*⟩	Black move
status_moves_white	312 ⟨*move-number*⟩ ... ⟨*move*⟩	White move
status_moves_black_tc	313 ⟨*move-number*⟩ ⟨*move*⟩ ⟨*secs*⟩	Black move and time
status_moves_white_tc	314 ⟨*move-number*⟩ ... ⟨*move*⟩ ⟨*secs*⟩	White move and time
	32x, 36x	Game over
status_winsmove_black	321 ⟨*move-number*⟩ ⟨*move*⟩	Black wins by move
status_losesmove_black	322 ⟨*move-number*⟩ ⟨*move*⟩	Black loses by move
status_winsmove_white	323 ⟨*move-number*⟩ ... ⟨*move*⟩	White wins by move
status_losesmove_white	324 ⟨*move-number*⟩ ... ⟨*move*⟩	White loses by move
status_drawsmove_black	325 ⟨*move-number*⟩ ⟨*move*⟩	Drawn by Black move
status_drawsmove_white	326 ⟨*move-number*⟩ ... ⟨*move*⟩	Drawn by White move
status_resigns_white	327	Black wins by resignation
status_resigns_black	328	White wins by resignation
status_draw_agreed	329	Draw accepted
status_flagfell_white	361	Black wins by White time expiring
status_flagfell_black	362	White wins by Black time expiring
	33x	Draw protocol
status_reqsdraw_black	331	Draw requested by Black
status_reqsdraw_white	332	Draw requested by White
status_refsdraw_black	333	Draw refused by Black
status_refsdraw_white	334	Draw refused by White
	34x	(see above)
	35x	(see above)
	39x	Bad status
status_disconnect_black	391	Black disconnected
status_disconnect_white	392	White disconnected
status_garble	399	Unknown problem

Table 10: State Display Messages

name	response	meaning
	38x	state display
sdisp_status	380 ⟨*move-number*⟩ ⟨*to-move*⟩	state
sdisp_status_tc	381 ⟨*move-number*⟩ ⟨*time-b*⟩ ⟨*time-w*⟩ ⟨*to-move*⟩	state and time
sdisp_board	382	board