Modeling the Game of Arimaa with Linguistic Geometry

José Roberto Mercado Vega and Zvi Retchkiman Königsberg

Abstract— **A computer defeated a chess world champion for the first time in 1997. This event inspired Omar Syed to develop the game of Arimaa. He intended to make it difficult to solve under present search approaches. Linguistic Geometry is a technique that offers a formal method based on the expertise of human chess masters, to make the development of complex heuristics easier. This article introduces a Linguistic Geometry based model for the game of Arimaa. It gives implementation for each of the essential components of Linguistic Geometry: trajectories, zones, translations and searches. A test case is given and it is used as input for a software implementation of the proposed model. The results given by the software are compared against the analysis made by a human player.**

I. Introduction

To study games, they can be represented through large trees. These trees include every possible evolution of the game. Searching through the game's tree for a solution of the game is the current approach in computer sciences. To reduce the execution time of game search algorithms, heuristics are used during the analysis of the game's tree. A heuristic is an estimation mechanism or a rule of thumb. Heuristics were first proposed by Claude Shannon [7]. The most popular search algorithms through general trees are depth-first, breath first and best first, while for search in games trees are minimax and alpha-beta [4].

Games are an interesting area of study because of their complexity, it is believed that techniques used for solving some games can also be used to solve other kind of problems. In particular, the study of chess has excelled. A new game called Arimaa was created by Omar and Aamir Syed [10]. They were inspired by the defeat of Garry Kasparov against the supercomputer Deep Blue. The game of Arimaa is a two player complete information, zero-sum game with no random factors. Arimaa was released in 2002 and it was designed to be complex to play well under traditional game search algorithms. This was done with the purpose of fomenting the development of new, ground breaking techniques. A challenge was published along the rules of the game, it consists of a prize of $10,000 USD for anyone who creates a computer program capable of defeating a human expert in a competition of six games.

Many computer programs have participated in the Arimaa challenge. The vast majority of them are based on conventional alpha-beta search algorithms. An example of an Arimaa design is that of David Fotland: *"Building a World Champion Arimaa Program"* [2]. Fotland's program was champion of the Arimaa tournament in 2004, this is a tournament only for software. Fotland's program was defeated by a human expert (Omar Syed).

R. Mercado and Z. Retchkiman are with Instituto Politécnico Nacional CIC, Mexico, e-mail: dragonslayking@yahoo.co.uk, mzvi@cic.ipn.mx

Haizhi Zhong published his master thesis in 2005 under the title: *"Building a Strong Arimaa-playing Program"* [11]. Zhong's program is based on alpha-beta reductions with some optimizations on the evaluation function based on some Arimaa tactics and selective generation of the movements. It also uses a transposition table and play ordering.

In 2006 Christian-Jan Cox published his master thesis: *"Analysis and Implementation of the Game Arimaa"* [1]. In this work Cox proposes a program called Coxa. It is based on an alpha-beta search algorithm with some optimizations along a transposition table. Cox tests multiple variations of the evaluation function and registers the results. None of the variants of the heuristics represents notable improvements over the search method.

A promising modeling technique that has its own potential is Linguistic Geometry (LG). LG was created by Boris Stilman [8], and has had a slow, though constant, development. From its origin LG was proposed in accordance to the way of thinking of some human chess experts. LG was developed around a class of games called abstract board games (ABG's). Processing along LG is done through a hierarchy of formal grammars. The levels of this hierarchy are: trajectories, webs, translations and searches.

This article presents a Linguistic Geometry based model for the game of Arimaa. The tools of LG are used to create the model, with some variations to make them adequate for the game of Arimaa. The model is used to implement one of the most common tactics of the game of Arimaa. To test the model a software implementation of the same is used and applied in some chosen positions of the game of Arimaa. Up to date, there is no published work which mixes Arimaa and Linguistic Geometry.

II. Arimaa

A. The rules of Arimaa

In order to play Arimaa, only a chess board and a set of chess pieces are needed. However, some changes to the rules have to be made. First off, the 64 cells of the board are equal, except for the cells c3, f3, c6 and f6 (according to chess algebraic notation). In Arimaa these are known as trap cells. Instead of black and white pieces, in Arimaa gold and silver are used respectively. Just as in chess, each player has 16 pieces of 6 different types. In descending weight order these are: elephant, camel, two horses, two dogs, two cats and eight rabbits.

Remark 1: By convention, diagrams presented have gold player on the lower rows (1 and 2) and silver on the upper rows (7 and 8). Also, gold pieces are shown facing to the right, while silver pieces are facing to the left.

The main goal of an Arimaa game is to take any allied rabbit to the opposing final row. A player loses if he has no more valid moves or if he repeats the same position three times. The game is a draw if both players have no more rabbits. According to the online Arimaa gameroom [3], the frequencies of these endings are 2.2%, 0.2% and 0.0% respectively. An Arimaa game starts with an empty board. The first move of each player, starting gold, is to place his 16 pieces on the board in any desired order on his first two rows.

After the initial positioning of pieces the game evolves through turns, each composed of up to four moves. Every piece is capable of moving one square up, down, left or right. Rabbits are not permitted to move backwards and there are no diagonal moves. The four available moves in a turn can be distributed as desired among all the allied pieces on the board.

An additional move in Arimaa is called a dislodge. A dislodge requires two moves. It consists of an allied piece dislodging an opposing one of lower weight. A dislodge can be a push or a pull. In a push, an opposing piece is dislodged to an adjacent cell and the dislodging piece moves to that cell. In figure 1, the gold elephant in d3 could push the silver rabbit from d2 to e2 and move itself to d2. A pull moves the allied piece to an adjacent cell and pulls the opposing one to the cell from where the allied piece moved. In figure 1, the silver elephant on d5 could move to d4 and pull the gold horse from d6 to d5.

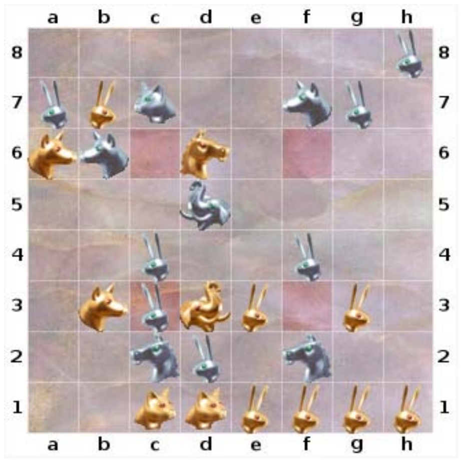

Fig. 1. Basic movements.

A piece is frozen when a heavier opposing piece is adjacent to it as long as no allied piece is adyancent too (regardless of its weight). A frozen piece cannot be moved by his owner, but it can be dislodged by the opponent. In figure 1, the silver rabbit in a7 is frozen , but the one in d2 is not because it is adjacent to another silver piece.

A capture is made when a piece goes inside a trap cell and no allied pieces are adjacent to it. When a piece is captured it is removed from the game. In figure 1, being silver's turn, it could capture the gold horse in d6 by dislodging it to c6 with the silver elephant on d5.

B. Value of pieces

In Arimaa there is no common agreement to measure the relative value of pieces. Without a doubt, the most valuable piece is the elephant but, since it is impossible to capture an elephant, considering its value is of no relevance. For the remaining pieces, a mild approximation of their relative value is as follows:

- A cat is worth more than a rabbit, but not much more.
- A dog is worth about two rabbits.
- A horse is approximately worth a dog and a rabbit.
- A camel is worth more than a horse and a cat, but less than a horse and a dog.

C. Arimaa tactics [6]

When playing board games, some tactics can be used. A tactic has a short-term goal. A tactic is a sequence of moves in one or more turns that can be calculated with precision to force a positive consequence to a player. In general, tactics are no longer than two turns (eight moves). One of the most basic tactics in Arimaa is to to capture an opposing piece.

1) Capture in one turn: To capture a piece in one turn, the easiest way is to dislodge a lighter piece to an uncovered trap. This is possible if the piece we are about to capture is, at most, two cells away from the trap and we have a heavier piece adjacent to it.

Example 1: In figure 2, the silver dog on b6 could pull the gold cat in b7 to c7 and then push it to c6. This captures the cat in c6.

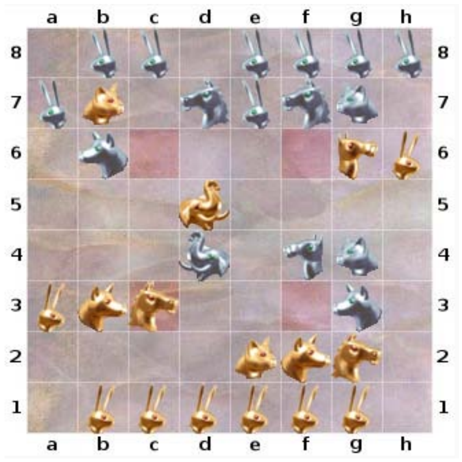

Fig. 2. Example of capture in one turn.

Example 2: If a piece is just a cell away from an unprotected trap, it is even more vulnerable, since it can be captured by non-adjacent pieces. In figure 2, being gold's turn, the gold elephant in d5 could move to c5, b5 and use the remaining move to push the silver dog from b6 to c6, capturing it.

III. Linguistic Geometry [9]

Linguistic Geometry (LG) is a technique for mathematical model construction which represents, to a certain degree, reasoning of human experts about games. LG focuses on a class of games called abstract board games (ABG). It was originally based on the chess expertise of the chess ex-champion Mikhail Botvinnik.

Definition 1: An ABG is an eight-tuple defined as follows:

$$< X, P, R_p, SPACE, v, S_0, S_t, TR >,$$

where: $X = \{x_i\}$ is a finite set of cells. $P = \{p_i\}$ is a finite set of pieces. The set P is the union of two disjoint sets P_1 and P_2, which represent the pieces of each player. $R_p : X \times X \rightarrow \{T, F\}$ is a family of reachability functions indexed by $p \in P$, where $\{T, F\}$ is the set of boolean values, if $R_p(x, y)$ is true then y is reachable from x for the piece p. $SPACE = \{S\}$ is the set of possible states S of the game. S is composed of a partial localization function $ON : P \rightarrow X$ and additional parameters; the value $ON(p) = x$ means that the element p is at cell x in the state S. Each state S in $SPACE$ is described by a list of well formed formulas (WFF): $\{ON(p_j) = x_k\}$. The additional parameters may include, for example, for state S a function $MT(S) \in \{1, 2\}$ that determines if the turn is for the first or second player. $v : P \rightarrow \mathbb{R}^+$ is a function, where $v(p)$ is the value of the piece p. $S_0 \in SPACE$ is the initial state of the game. $S_t = \{S_i\}$ is the set of target states of the game. It represents the endgame conditions, and can represent a victory for any player or a draw. $TR = \{tr\}$ is the set of transitions from one state to another of the game or valid moves. Each transition $tr(p, x, y)$ is described in terms of three lists of WFF: one contains WFF's that will be added to the state's description; other contains WFF's that will be deleted from the description of the state; and the last one contains WFF's that show the applicability restrictions of the transition. In concrete, for a state $S \in SPACE$, the three lists of each transition $tr(p, x, y)$ are given by:
add list: $ON(p) = y$
delete list: $ON(p) = x$
applicability list: $(ON(p) = x) \wedge R_p(x, y)$
with $p \in P$ and $x, y \in X$.

LG uses a formal language hierarchy to represent some of the relationships a human expert would usually find over an ABG. The idea is to create a set of tools that allows one to introduce heuristics in an abstract level. To do this, the LG hierarchy uses some geometrical and spatial relationships among the pieces on the board to create complex structures that are intuitive to the human player. These structures can be used in the creation of heuristics.

The LG hierarchy of tools, in order of complexity, is comprised by: trajectories, webs, translations and searches. Trajectories are presented through strings (generated by the grammar of trajectories) which contain an ordered sequence of cells that a piece needs to visit to reach a destination cell. Webs are presented through strings (generated by the grammar of webs) which contain multiple trajectories, these keep a relationship in function of pieces that attack (or intercept) others. Translations are grammars that convert a web into another based on a move made. Searches are strings (generated by the grammar of searches) that represent search trees in LG, its nodes are states and its children are generated through translations.

In LG, the hierarchy of subsystems is presented as a hierarchy of formal languages [5]. These languages use symbols. A symbol is an abstract entity not formally defined. Some examples of symbols are: a, t, $a(x_i)$, $t(p_2, t_2, \tau_2)$, $\pi(i_5)$, etc. An alphabet is a finite set of symbols. A string is a finite sequence of concatenated symbols that belong to an alphabet; for example, $a(x_1)a(x_2)\ldots a(x_n)$ is a string if $a(x_1)$, $a(x_2)$, $\ldots$, $a(x_n)$ are symbols of some alphabet. A formal language is a set of strings of symbols of some alphabet; every language includes the empty string ε. In the hierarchy of languages of LG each string of a low level is a symbol of the next higher level. A formal grammar is a mechanism or description that characterizes a language. Usually, a grammar is presented as a series of rules which generate the strings of the language.

Each level of the hierarchy of subsystems of LG is composed of languages and/or grammars. The strings of the languages are a way for information exchange between the levels of the hierarchy. The grammars are the method for processing the information given by those strings to get some result. Figure 3 shows the organization of these elements in the hierarchy of LG.

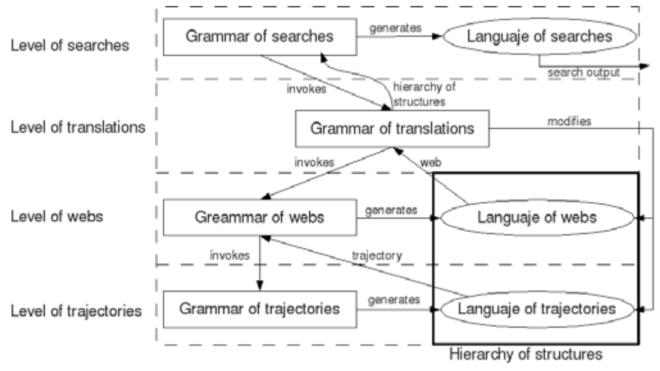

Fig. 3. LG hierarchy of subsystems.

The lowest level of the hierarchy of LG is the level of trajectories. A trajectory t is a string of symbols of the form $t = a(x_1)a(x_2)\ldots a(x_n)$. The string is formed of values x_i that represent the sequence of steps that takes a piece to go from one cell to another. These values are linked through the special symbol a. $L_t^H(S)$ is the set of trajectories of length less than H for a state S of the ABG, and is called language of trajectories. The language of trajectories is generated through the grammar of trajectories.

The second level of the hierarchy of LG is the level of webs. A web w is a string of the form $w = t(p_1, t_1, \tau_1)t(p_2, t_2, \tau_2)\ldots t(p_k, t_k, \tau_k)$; where p_i is a piece, t_i is a trajectory and $\tau_i \in PARAM$ is a list of domain specific parameters. These are linked through the special symbol t. $L_W(S)$ is a set of webs for a state S of the ABG, it is called language of webs. A kind of webs that are of great importance are the zones. Zones define the set $PARAM$ as the natural numbers. In a zone, each value τ_i is a time restriction. The grammar that generates all the strings of a language of zones is the grammar of zones. Figure 4 shows an example of a web (in particular, a zone).

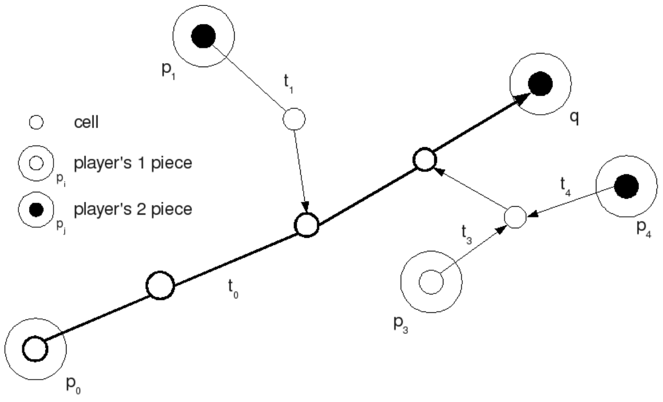

Fig. 4. Example of a web.

The next level of the hierarchy is the level of translations. The job of this level is to transform a hierarchy of structures to match the present state. The process generates a new hierarchy of structures, this is done in the grammar of translations.

The highest level of the hierarchy of LG is the level of searches. A search is a string of the form $(\pi(i_1)\pi(i_2)\ldots\pi(i_m), Child, Sibling, Parent,$ Other-fun); where π has notational purposes, i_k represents a state of the ABG. The parameters $Child$, $Sibling$ and $Parent$ are functions that give information on the structure of the tree. The parameter Other-fun is an n-tuple of domain specific parameters. A search represents an LG search tree.

A. Controlled grammars

The tools used in LG for the generation of languages are a kind of grammars called controlled grammars. Controlled grammars are rule based, they transform input symbols into output symbols through the criteria captured in the rules. A general description of controlled grammars is shown on table I. Each rule of a controlled grammar is called a production. Each production has a label l, an applicability condition $Q(,,)$, a kernel of the form $A(,,) \to B(,,)$, a set of formulas π_k that operates over the kernel parameter symbols, a set of additional formulas π_n (that are not in the kernel) of the form $C(,,) = D(,,)$ and two sets of feasible tags F_T and F_F. The parameters (values and functions) are shown between parenthesis. The values of the parameters change as productions are applied.

Remark 2: In table I, the formulas π_k are implicitly presented among the kernel parameters, though formally, π_k is an independent set.

Tag	**Cond.**	**Kernel,** π_k	π_n	F_T	F_F
l	$Q(,,)$	$A(,,) \to B(,,)$	$C(,,) = D(,,)$	L_T	L_F

TABLE I
CONTROLLED GRAMMAR DESCRIPTION.

A controlled grammar works in the following way. At the beginning of the generation of the string it starts with an initial symbol in the production with tag l. After applying the production:

- If condition $Q(,,)$ holds, the production with tag l is applied making the substitution specified by the kernel, and goes to a production with tag in the set L_T.
- If $Q(,,)$ does not hold or the string does not contain the symbol of the left side of the kernel $A(,,)$, the production l is not applied, and goes to a production with tag in the set L_F.

The substitution specified by the kernel is carried over the string (generated at the moment) by replacing the left-side symbol of the kernel $A(,,)$ for the right-side of it. If the condition $Q(,,)$ holds, besides from making the kernel substitution, the operations specified by π_k and π_n are also executed, updating corresponding values. The sets L_T and L_F can be empty. The string generation ends if, when applying a production, holds either $(Q(,,) = T) \wedge (L_T = \emptyset)$ or $(Q(,,) = F) \wedge (L_F = \emptyset)$.

Definition 2: A controlled grammar G is an eight-tuple:

$$G = (V_T, V_N, V_{PR}, E, H, Param, L, R),$$

where: V_T is the terminal symbol alphabet. V_N is the non-terminal symbol alphabet; $I \in V_N$ is the initial symbol. V_{PR} is the first order predicate calculus alphabet PR: $V_{PR} = Truth \bigcup Con \bigcup Var \bigcup Func \bigcup Pred \bigcup LOG$, where: $Truth$ is the set of truth values T and F; Con is the set of constant symbols; Var is the set of variable symbols; $Func = Fcon \bigcup Fvar$ is the set of functional symbols, with constant symbols $Fcon$ and variable functional symbols $Fvar$; $Pred$ is the set of predicate symbols; LOG is the set of logical operators. E is a numerable set called problem's domain. H is an interpretation of the predicate calculus PR over the set E. $Param$ is the function $Param : V_T \bigcup V_N \to 2^{Var}$, that associates each symbol of the alphabet $V_T \bigcup V_N$ to a set of parameters. L is a finite set called tag set. R is a finite set of productions, that is, a finite set of seven-tuples of the form: $(l, Q, A \to B, \pi_k, \pi_n, F_T, F_F)$, where $l \in L$ is the tag of the production; Q is the applicability condition of the production, represented by a well formed formula of the predicate calculus PR; $A \to B$ is an expression called kernel of the production, with $A \in V_N$ and $B \in (V_T \bigcup V_N)^*$; π_k is a set of functional formulas that are among the parameters of the symbols of the kernel; π_n is a set of functional formulas that are not among the parameters of the kernel; $F_T \subseteq L$ is a set of permissible labels in case of success ($Q = T$); $F_F \subseteq L$ is the set permissible labels in case of failure ($Q = F$).

IV. Arimaa model

The Arimaa model is built using the LG tools as a base. An LG based model must have: an ABG modeling the essential characteristics of the problem, a level of trajectories with a corresponding grammar of trajectories, a level of webs with a corresponding grammar of zones, a level of translations with an adequate grammar of translations and a level of searches with a corresponding grammar of searches corresponding to the heuristics of the problem.

The proposed model uses the same grammar of trajectories as the one presented in [9]. The level of webs is composed of a new zone for the game of Arimaa. The level of translations is composed of the grammar of translations presented in [9] with some changes to adapt to the new zones. Finally, the level of searches is formed by a new search which models some basic tactics for the game of Arimaa.

A. *The ABG for Arimaa*

To model the game of Arimaa, it is necessary to capture the most basic characteristics of the game, such as: cells, pieces, reachability relationships, valid moves, etc. The corresponding ABG for the game of Arimaa is the following 8-tuple:

$$< X, P, R_p, SPACE, v, S_0, S_t, TR >,$$

where:

$X = \{1, 2, \ldots, 64\}$
$P = P_1 \bigcup P_2$
  $P_1 = \{G\_EL, G\_CA, G\_HO_1, G\_HO_2, G\_DO_1, G\_DO_2, G\_CT_1, G\_CT_2, G\_RA_1, \ldots, G\_RA_8, \}$
  $P_2 = \{S\_EL, S\_CA, S\_HO_1, S\_HO_2, S\_DO_1, S\_DO_2, S\_CT_1, S\_CT_2, S\_RA_1, \ldots, S\_RA_8, \}$

$$R_p(x,y) = \begin{cases} T & \text{if } ((p \notin SR) \wedge (x \notin TB) \wedge (y = x+8)) \vee \\ & ((p \notin GR) \wedge (x \notin BB) \wedge (y = x-8)) \vee \\ & ((x \notin RB) \wedge (y = x+1)) \vee \\ & ((x \notin LB) \wedge (y = x-1)) \\ F & \text{in other case} \end{cases}$$

where:
  $SR = \{S\_RA_i\}$, with $i = 1, 2, \ldots, 8$
  $GR = \{G\_RA_i\}$, with $i = 1, 2, \ldots, 8$
  $TB = \{57, 58, 59, 60, 61, 62, 63, 64\}$
  $BB = \{1, 2, 3, 4, 5, 6, 7, 8\}$
  $LB = \{1, 9, 17, 25, 33, 41, 49, 57\}$
  $RB = \{8, 16, 24, 32, 40, 48, 56, 64\}$

$SPACE$ is the set of every possible valid position in the game of Arimaa.

$$v(p) = \begin{cases} 3 & \text{if } p \in \{G\_RA_i, S\_RA_i\};\ i = 1, 2, \ldots, 8 \\ 4 & \text{if } p \in \{G\_CT_1, G\_CT_2, S\_CT_1, S\_CT_2\} \\ 6 & \text{if } p \in \{G\_DO_1, G\_DO_2, S\_DO_1, S\_DO_2\} \\ 9 & \text{if } p \in \{G\_HO_1, G\_HO_2, S\_HO_1, S\_HO_2\} \\ 14 & \text{if } p \in \{G\_CA, S\_CA\} \\ 20 & \text{if } p \in \{G\_EL, S\_EL\} \end{cases}$$

$S_0 = \emptyset$
$S_t = \{S_i\}$,
  such that the WFF $ON(p) = x$ is in the state S_i and either condition is fullfilled:
  $(p \in GR) \wedge (x \in TB)$; victory for player P_1
  $(p \in SR) \wedge (x \in BB)$; victory for player P_2
$TR = \{tr\}$

Remark 3: The construction of the set of transitions of the system (TR), the set of valid moves, is done according to the rules of Arimaa. Here, such construction is made under informal terms, the reader is referred to [9] and subsection II-A for more information.

In the previous definition, the set of cells X is comprised simply by a numerical sequence whose correspondence is presented in figure 5. The set of pieces P contains silver and gold pieces each one containing itself the 16 pieces for the corresponding player. The reachability relation R_p simply reflects the mobility of the pieces of Arimaa. The function of value of the pieces is simply proposed with some values that conform to the proposals of subsection II-B. The initial state S_0 is the empty board. The set of final states S_t is formed by those states in which a rabbit is on the goal row.

	a	b	c	d	e	f	g	h	
8	57	58	59	60	61	62	63	64	8
7	49	50	51	52	53	54	55	56	7
6	41	42	43	44	45	46	47	48	6
5	33	34	35	36	37	38	39	40	5
4	25	25	27	28	29	30	31	32	4
3	17	18	19	20	21	22	23	24	3
2	9	10	11	12	13	14	15	16	2
1	1	2	3	4	5	6	7	8	1
	a	b	c	d	e	f	g	h	

Fig. 5. Cell distribution for the LG Arimaa model.

Some auxiliary definitions are given to simplify the notation for the remaining of the paper.

Definition 3: The set of trap cells denoted by $TRAPS$ is equal to the set $\{19, 22, 43, 46\}$ (see gray cells in figure 5).

Definition 4: The function $HEAVIER : P \times P \rightarrow \{T, F\}$ is the function of piece domination. It tells if a piece is heavier than other and, as a consequence, if it is capable of freezing it. It is defined as follows:

$$HEAVIER(p_1, p_2) = \begin{cases} T & \text{if } v(p_1) > v(p_2) \\ F & \text{if } v(p_1) \leq v(p_2) \end{cases}$$

B. *Grammar of CUT zones for the game of Arimaa*

The grammar of zones presented in [9] is not compatible with the characteristics of the game of Arimaa. The main reason for this is that it assumes that pieces can attack and take other pieces that are at the destination of the movement. This does not happen in Arimaa.

A grammar of zones is presented for the game of Arimaa. It is called grammar of CUT zones. Its main purpose is to model the interaction of pieces around a trap square, so as to be used to model the capture in one turn tactic.

The grammar of CUT zones is shown in table II; it is based on the grammar of zones presented in [9]. The zone includes two cases of immediate capture (modeled by productions 2_i and 3_j): the first one consists of capturing a piece adjacent to the trap square when its captor is at most two moves away; the second being when that piece is at most two moves away from the trap and its captor is adjacent to it. In both cases the base trajectories of the pieces to their destinations are included. Connected trajectories are also included in a similar way as proposed in [9]. The following definitions complete the grammar of table II.

$V_T = \{t\} \quad V_N = \{I, A\}$
$V_{PR}: Con = \{x_0, y_0, l_0, p_0\}$; $Var = \{x, y, l, p, \tau, \theta, v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_n\}$ (for short, it is denoted $u = (x, y, l)$, $v = (v_1, v_2, \dots, v_n)$, $w = (w_1, w_2, \dots, w_n)$, $zero = (0, 0, \dots, 0)$); $Func = Fcon \bigcup Fvar$,
$Fcon = \{f_x, f_y, f_l, g_1, g_2, \dots, g_n, h_1, h_2, \dots, h_M, h_1^0, h_1^0, \dots, h_M^0, \mathit{DIST}, \mathit{init}, \mathit{ALPHA}, \mathit{CTRL}_p\}$,
(for short, it is denoted $f = (f_x, f_y, f_l)$, $g = (g_1, g_2, \dots, g_n)$), $Fvar = \{x_0, y_0, l_0, p_0, TIME, NEXTTIME\}$;
$Pred = \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7\}$,
$Q_1(u) = (\neg\exists p_1, p_2(\mathit{CTRL}_{p_1}(x) \wedge \mathit{CTRL}_{p_2}(x) \wedge \neg \mathit{OPP}(p_0, p_1) \wedge \neg \mathit{OPP}(p_0, p_2))$
$Q_2(u) = (\exists p_o, p_a(\mathit{CTRL}_{p_o}(x) \wedge \mathit{MAP}_{ON(p_a), p_a}(ON(p_o)) \leq 2)))$
$Q_3(u) = (\exists p_o, p_a(\mathit{CTRL}_{p_a}(ON(p_o)) \wedge (\mathit{MAP}_{ON(p_o), p_o}(x) \leq 2)))$
$Q_4(u) = (x \neq n) \vee (y \neq n)$
$Q_5(u) = (\exists p((ON(p) = x) \wedge (l > 0) \wedge (x \neq x_0) \wedge (x \neq y_0)) \wedge ((\neg \mathit{OPP}(p_0, p) \wedge (\mathit{MAP}_{x,p}(y) = 1)) \vee (\mathit{OPP}(p_0, p) \wedge (\mathit{MAP}_{x,p}(y) \leq l)))$
$Q_6(w) = (w \neq zero)$
$Q_7 = T$
$E = Z_+ \bigcup X \bigcup P \bigcup L_t^{l_0}(S)$
$Parm: \{I, A, t\} \to 2^{Var}$
$Param(I) = \{u, v, w\}$, $Param(A) = \{u, v, w\}$, $Param(t) = \{p, \tau, \theta\}$;
$L = \{1, 4, 6, 7\} \bigcup two \bigcup three \bigcup five$;
$two = \{2_1, 2_2, \dots, 2_M\}$, $three = \{3_1, 3_2, \dots, 3_M\}$, $five = \{5_1, 5_2, \dots, 5_M\}$

At the beginning of derivation: $u = (x_0, y_0, l_0)$, $v = zero$, $w = zero$, $x_0 \in \mathit{TRAPS}$, $y_0 = 0$, $l_0 = 0$, $p_0 \in P$, $n = card(X)$, $M = card(L_t^{l_0}(S))$, $TIME(z) = NEXTTIME(z) = 2n \; \forall z \in X$

$\mathit{CTRL}_p: X \to \{T, F\}$, $\mathit{CTRL}_p(x) = R_p(ON(p), x)$
$init: (X \times X \times Z_+) \times Z_+ \to Z_+$

$$init(u, r) = \begin{cases} 2n, & \text{if } u = (0,0,0) \\ r, & \text{if } u \neq (0,0,0) \end{cases}$$

$f: (X \times X \times Z_+) \times Z_+^n \to (X \times X \times Z_+)$, $f(u, v) =$

$$\begin{cases} (x+1, y, l), & \text{if } ((x \neq n) \wedge (l > 0)) \vee \\ & \quad ((y = n) \wedge (l \leq 0)) \\ (1, y+1, TIME(y+1) \times & \text{if } (x = n) \vee ((y \neq n) \wedge \\ v_{y+1}), & \quad (l \leq 0)) \end{cases}$$

$\mathit{DIST}: X \times P \times L_t^{l_0}(S) \to Z_+$, Let $t_0 \in L_t^{l_0}(S)$ be $t_0 = a(z_0)a(z_1)\dots a(z_m)$, $t_0 \in t_{p_0}(z_0, z_m, m)$;
$\mathit{DIST}(x, p_0, t_0) =$

$$\begin{cases} k+1, & ((z_m = y_0) \wedge (p = p_0) \wedge (\exists k(1 \leq k \leq m) \wedge \\ & (x = z_k))) \vee (((z_m \neq y_0) \vee (p \neq p_0)) \wedge \\ & (\exists k(1 \leq k \leq m-1) \wedge (x = z_k))) \\ 2n, & \text{in other case} \end{cases}$$

$\mathit{ALPHA}: X \times P \times L_t^{l_0}(S) \times Z_+ \to Z_+$,
$\mathit{ALPHA}(x, p_0, t_0, k) =$

$$\begin{cases} max(NEXTTIME(x), k), & \text{if } (\mathit{DIST}(x, p_0, t_0) \neq 2n) \wedge \\ & \quad (NEXTTIME(x) \neq 2n) \\ k, & \text{if } (\mathit{DIST}(x, p_0, t_0) \neq 2n) \wedge \\ & \quad (NEXTTIME(x) = 2n) \\ NEXTTIME(x), & \text{if } \mathit{DIST}(x, p_0, t_0) = 2n \end{cases}$$

$g_r: P \times L_t^{l_0}(S) \times Z_+^n \to Z_+$, with $r \in X$

$$g_r(p_0, t_0, w) = \begin{cases} 1, & \text{if } \mathit{DIST}(r, p_0, t_0) < 2n \\ w_r, & \text{if } \mathit{DIST}(r, p_0, t_0) = 2n \end{cases}$$

$g: P \times L_t^{l_0}(S) \times Z_+^n \to Z_+^n$
$g(p_0, t_0, w) = (g_1(p_0, t_0, w), g_2(p_0, t_0, w), \dots, g_n(p_0, t_0, w))$

$$\mathit{TRACKS}_p = \{p\} \times \left(\bigcup_{1 \leq k \leq l} L\left[G_t^{(2)}(x, y, k, p_a)\right] \right),$$

$h_i^0: (X \times X \times Z_+) \to P \times L_t^{l_0}(S)$
Let $\mathit{TRACKS}_{p_a} = \{(p_0, t_1), (p_0, t_2) \dots, (p_0, t_b)\}$, with $(b \leq M)$,

$$h_i^0(u) = \begin{cases} (p_0, t_i), & \text{if } (\mathit{TRACKS}_{p_0} \neq \emptyset) \wedge (i \leq b) \\ (p_0, t_b), & \text{if } (\mathit{TRACKS}_{p_0} \neq \emptyset) \wedge (i > b) \\ \varepsilon, & \text{in other case} \end{cases}$$

$h_i: (X \times X \times Z_+) \to P \times L_t^{l_0}(S)$
Let $\mathit{TRACKS}_p = \{(p, t_1), (p, t_2) \dots, (p, t_m)\}$, with $(m \leq M)$,

$$h_i(u) = \begin{cases} (p, t_i), & \text{if } (\mathit{TRACKS}_{p_0} \neq \emptyset) \wedge (i \leq m) \\ (p, t_m), & \text{if } (\mathit{TRACKS}_{p_0} \neq \emptyset) \wedge (i > m) \\ \varepsilon, & \text{in other case} \end{cases}$$

Productions 2_i and 3_j are new, with respect to the grammar of zones presented in [9], and intend to reflect the characteristics of these new kind of zones. Conditions Q_1, Q_2 and Q_3 are new and reflect some characteristics of their respective productions. Also, production 3_j is analogous to production 2_i but it applies under different circumstances. The function $\mathit{CTRL}_p(x)$ was added to reduce notation; it is true if piece p is adjacent to cell x.

Remark 4: The grammar does not include cases for which the piece to capture is already inside the trap cell.

C. Translation grammar for Arimaa

The grammar of translations for Arimaa is based on the one in [9] but, some changes are needed to reflect the changes made to the grammars of zones. The only actual difference is the definition of the function $timer_\pi$. This function controls the times assigned to the translated versions of the trajectories. The modified version of the function $timer_\pi$ is denoted y $timer_\pi^A$ and it is defined as follows:

Definition 5: Let $\pi_{T_0}(Z_1) = Z_2$ be a translation from the zone $Z_1 \in L_Z(S_1)$ to the zone $Z_2 \in L_Z(S_2)$, and let $Z_1 = t(p_0, t_0, \tau_0)t(p_1, t_1, \tau_1)\dots t(p_r, t_r, \tau_r)$. A **time distribution**

L	Q	**Kernel,** π_k	π_n $(\forall z \in X)$	F_T	F_F
1	Q_1	$I(u,v,w) \to A(u,v,w)$	$\emptyset$	two	$\emptyset$
2_i	Q_2	$A(u,v,w) \to t(h_i^0(u'),4)t(p_o,a(ON(p_o))a(x),1)$ $A((0,0,0),g(p_0,h_i^0(u'),w),zero)$	$TIME(z) = DIST(z,h_i^0(u'))$ $u' = (ON(p_a),ON(p_o),4)$	$\{4\}$	$three$
3_j	Q_3	$A(u,v,w) \to t(h_i^0(u'),4)t(p_a,a(ON(p_a))$ $a(ON(p_o)),1)A((0,0,0),g(p_0,h_i^0(u'),w),zero)$	$TIME(z) = DIST(z,h_i^0(u'))$ $u' = (ON(p_o),x,4)$	$\{4\}$	$\emptyset$
4	Q_4	$A(u,v,w) \to A(f(u,v),v,w)$	$NEXTTIME(z) =$ $init(u,NEXTTIME(z))$	$five$	$\{6\}$
5_k	Q_5	$A(u,v,w) \to t(h_j(u),TIME(y))$ $A(u,v,g(p,h_j(u),w))$	$NEXTTIME(z) =$ $ALPHA(z,h_j(u),TIME(y)-l+1)$	$\{4\}$	$\{4\}$
6	Q_6	$A(u,v,w) \to A((0,0,0),w,zero)$	$TIME(z) = NEXTTIME(z)$	$\{4\}$	$\{7\}$
7	Q_7	$A(u,v,w) \to \varepsilon$	$\emptyset$	$\emptyset$	$\emptyset$

TABLE II
GRAMMAR OF CUT ZONES

function $timer_\pi^A$ is the function:

$$timer_\pi^A : Con_{\Pi_{T_0}}(Z_1) \to \mathbb{Z}_+$$

To build $timer_\pi^A$ three cases are considered:

1) If $\Pi_{T_0}(t_0) \neq t_0$; that is, if transition T_0 occurs over the main trajectory t_0 of the zone Z_1, then:

$$timer_\pi(t(p_c,t_c,\tau_c)) = \tau - 1$$

for every symbol $t(p_c,t_c,\tau_c) \in Con_{\Pi_{T_0}}$.

2) If $\Pi_{T_0}(t_k) \neq t_k$ for some $t_k \neq t_0$; that is, if transition T_0 occurs over a trajectory t_k of the zone Z_1 which is not the main trajectory, then $timer_\pi^A$ is defined recursively as:

$$timer_\pi^A(t(p_i,t_i,\tau_i)) = \begin{cases} \tau_i & \text{if } (i=0) \vee (C(t_i,t_0)=T) \\ \max\limits_{t_c \in CA(t_i)} TNEW(t_c,t_i) & \text{in other case} \end{cases},$$

where:

$$CA(t_i) = \{t_c | C(t_c,t_i) = T, \text{ for some } t(p_c,t_c,\tau_c) \in Con_{\Pi_{T_0}}(Z_1)\},$$

$$TNEW(t_c,t_i) = \begin{cases} timer_\pi(t(p_c,t_c,\tau_c)) - len(t_c), & \text{if } C_1 \\ timer_\pi(t(p_c,t_c,\tau_c)) - len(t_c)+1, & \text{if } C_2 \\ timer_\pi(t(p_c,t_c,\tau_c)) - len(t_c)+2, & \text{if } C_3 \end{cases}$$

$C_1 = t_c \neq t_k \wedge \neg HEAVIER(t_c,t_i)$
$C_2 = (t_c \neq t_k \wedge HEAVIER(t_c,t_i)) \vee (t_c = t_k \wedge \neg HEAVIER(t_c,t_i))$
$C_3 = (t_c = t_k) \wedge HEAVIER(t_c,t_i)$

3) If $\Pi_{T_0}(t_m) = t_m \quad \forall t_m \in TA(Z_1)$; that is, if transition T_0 does not affect any of the trajectories of the zone Z_1, then:

$$timer_\pi^A(t(p_m,t_m,\tau_m)) = \tau_m$$

for every $t(p_m,t_m,\tau_m) \in Con_{\Pi_{T_0}}$.

D. Grammar of searches for Arimaa

It is possible to model heuristics that represent common tactics for the game of Arimaa using the proposed grammar of zones and the modified version of the grammar of translations. The formal model of the grammar that must capture those heuristics is the same as in [9].

Capture in one turn: The main goal of this tactic is to capture an opponent's piece. To achieve this, a CUT zone is used along the following criteria to direct the search:

The goal is to capture an opposing piece. A CUT zone is to be generated in each of the board traps. The attacking player will prefer to dislodge the attacked piece towards the trap cell. Defensive player will prefer to move the attacked piece away from the zone. Attacking player shall try to move obstacles away from its trajectory. Defensive player shall try to dominate the trap to avoid capture. Defensive player shall try to block the main trajectory of the zone.

V. TEST CASE AND RESULTS

A test case is proposed, it is analyzed by a human and the results are compared to those given by a software implementation of the model. It must be clarified that through computing strength it is possible to consider some positions that are usually not considered by human players; though, at the same time, some possibilities that the human player would take into account are excluded due to weaknesses in the heuristics proposed. It is possible to reduce this effect by means of modifying the heuristics.

The test case is based on the capture in one turn tactic. The introduced examples are academic and could or could not come from actual Arimaa games. The figure shown presents simplified versions of the generated zone. The straight lines represent the main trajectories, while the arcs represent second or superior negation trajectories. The analysis ot the example consists of two parts, one arount trap c6 and the other one around trap f3. Just the second one is discused here.

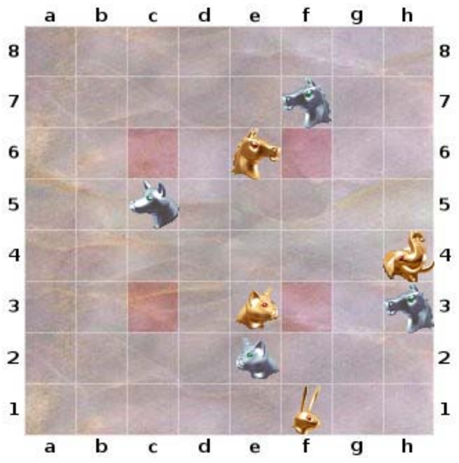

Fig. 6. Test case - starting board position.

A. Test case

In case it is gold's turn, the gold elephant in h4 can capture, in one turn, the silver horse in h3; to achieve this he only needs to push it twice to g3 and f3. The situation is a little more complicated if it is silver's turn since the silver horse is frozen and cannot come out of the dominion of the gold elephant. To cover the silver horse, silver has the following options: take the silver dog from c5 to f4 (following c4, d4, e4, and f4) to cover the trap square f3, in this case the gold elephant could capture the silver dog instead of the silver horse (moving from h4 to g4 and then pushing the silver dog from f4 to f3); silver could also move the silver cat from e2 to f2 as a mean to cover f3; or take the silver cat from e2 to g3 (following f2, g2, and g3), where the silver cat covers the trap, unfreezes the silver horse in h3 and blocks g3 to avoid a dislodge of the silver horse to the trap. It can be seen that silver has options to avoid the capture of the silver horse only if silver is to move first.

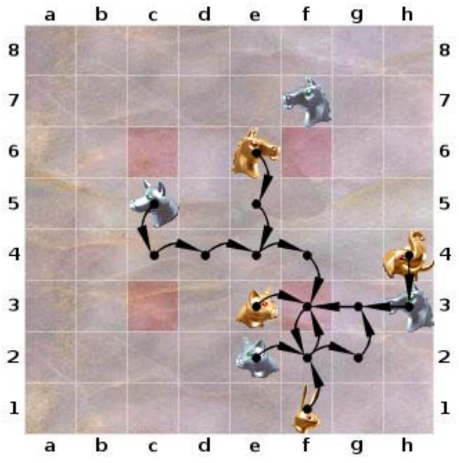

Fig. 7. Test case - CUT zone for the trap square f3.

The CUT zone around f3 is shown in figure 7. It includes all the pieces in the figure except from the silver horse in f7 since it has no active action. The inclusion of the silver dog in c5 and the silver cat in e2 is direct due to the possibility of controlling trap f3 in four or less moves. The gold horse in e6 is included in the zone because of the possibility of blocking or freezing the silver dog in c5 (when it crosses e4). The same happens with the gold rabbit in f1 and the gold cat in e3, but these do not have relevance over the trajectory of the silver cat in e2.

When gold plays first, the sequence of movements needed to capture the silver horse are direct and, given the priorities of the heuristics (subsection IV-D), it is the first sequence to be generated. The solution generated is: gold elephant in h4 pushes silver horse from h3 to g3 and f3.

When silver plays first, the software's proposed solution is: move the silver cat from e2 to f2, this avoids immediate capture of the silver horse. The program returns this solution for being the first to fulfill the requirements. Hence, for this case, the proposed model is also satisfactory.

VI. Conclusions

The proposed Arimaa model is adequate for modeling some of the most common tactics of the game of Arimaa. The model simplifies the formulation of heuristics with particular objectives. The zones and heuristics proposed in this work (level of webs and searches) are quite restrictive. The modeling of a greater number of tactics results in a greater number of zones and a greater complexity on the level of searches. In the test case shown in this work, the results are in accordance to the analysis of the positions. There are some cases where the results are not optimal and the proposed model shows some weaknesses. To correct them to some degree, it is proposed to extend the level of searches by using the tools introduced in this work.

References

[1] Christ-Jan Cox. Analysis and implementation of the game arimaa. Master's thesis, MICC-IKAT, 2006.
[2] David Fotland. Building a world-champion arimaa program. In *Computers and Games*, pages 175–186. Springer Berlin / Heidelberg, 2004.
[3] Arimaa gameroom server. http://arimaa.com/arimaa/gameroom/.
[4] Timothy Hart and Daniel Edwards. The alpha-beta heuristic. Technical report, Cambridge, MA, USA, 1963.
[5] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
[6] On line arimaa book. http://en.wikibooks.org/wiki/arimaa.
[7] Claude Elwood Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.
[8] Boris Stilman. A linguistic approach to geometric reasoning. *Computers and Mathematics with Applications*, 26(7):29–58, 1993.
[9] Boris Stilman. *Linguistic geometry: from search to construction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
[10] Omar Syed and Aamir Syed. Arimaa - a new game designed to be difficult for computers. *International Computer Games Association Journal*, 26:138–139, 2003.
[11] Haizhi Zhong. Building a strong arimaa-playing program. Master's thesis, University of Alberta, 2005.