LINGUISTIC GEOMETRY (LG)
AND ITS APPLICATION TO HISTORICAL CONFLICTS
by
SAMUEL M. GAMMETER
B.S., Computer Science, University of Colorado at Denver, 2003

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Masters of Science
Computer Science
2013
This thesis for the Masters of Science degree by

Samuel M Gammeter

has been approved for the

Computer Science Program

by

Boris Stilman, Chair

Gita Alaghband

Tom Altman

November 6, 2013
Gammeter, Samuel M. (M.S. Computer Science and Engineering)

Linguistic Geometry (LG) and Its Application to Historical Conflicts

Thesis directed by Professor Boris Stilman.

ABSTRACT

This paper begins with a brief historical review of Alexander the Great for contextual purposes, followed by a brief introduction to Linguistic Geometry (LG) theory, which is based upon the idea of constructing solutions to large search problems, similar to how intelligent agents would think through complex problems to find a solution, rather than iterating through vast searches for that same solution. LG accomplishes construction using an Abstract Board Games (ABG) to define the type of problem, trajectories to guide the solution, and zones to formalize the interaction between opposing objectives. This paper also expands upon LG by the application of LG to the historical battle of Alexander the Great crossing the Jaxartes river, by the consideration of zones in this context, how zones might be expanded, how trajectories and zones might be produced in parallel, and how a main, complex objective or series of objectives might be broken up via hyper games into a series of simpler games with single objectives.

The form and content of this abstract are approved. I recommend its publication.

Approved: Boris Stilman
DEDICATION

I dedicate this work to my patient wife Holly – ever so patient; my success is her support.
# Table of Contents

## I. Alexander and Ancient Conflict: A Historical Review

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Alexander the Great in Ancient Conflicts</td>
<td>2</td>
</tr>
<tr>
<td>Selecting a Battle to Review</td>
<td>3</td>
</tr>
<tr>
<td>Alexander Crossing the Jaxartes</td>
<td>5</td>
</tr>
<tr>
<td>Battle Lines in Ancient Time</td>
<td>6</td>
</tr>
</tbody>
</table>

## II. General Considerations of LG in Ancient Conflict

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>Understanding the Location Shape</td>
<td>7</td>
</tr>
<tr>
<td>Location Size</td>
<td>9</td>
</tr>
<tr>
<td>Location Types</td>
<td>10</td>
</tr>
<tr>
<td>Issues in Scaling Reachability Based on Terrain Type</td>
<td>11</td>
</tr>
<tr>
<td>Location Height Map</td>
<td>13</td>
</tr>
</tbody>
</table>

## III. New and Existing Zones in LG for Ancient Conflict

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>15</td>
</tr>
<tr>
<td>Trajectories and Zones in LG</td>
<td>15</td>
</tr>
<tr>
<td>LG Strategies Engine</td>
<td>20</td>
</tr>
<tr>
<td>Possible New Zones Required by Ancient Conflict</td>
<td>23</td>
</tr>
<tr>
<td>Hyper Games</td>
<td>25</td>
</tr>
</tbody>
</table>

## IV. Application of LGTOOLKIT to Alexander the Great’s Battle

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>26</td>
</tr>
<tr>
<td>Manually Building a Battle Line Zone</td>
<td>27</td>
</tr>
<tr>
<td>Setting Up the Jaxarte Scenario</td>
<td>30</td>
</tr>
<tr>
<td>LG-SE Executing the Scenario</td>
<td>32</td>
</tr>
<tr>
<td>Securing the Nearest Bank</td>
<td>32</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

FIGURE

1 Map of Macedonian Kingdom at the Start of Alexander the Great’s Reign as King ...... 2
2 Path Along Which Alexander the Great Travelled During Years of Campaigns .......... 3
3 Historical Depiction of the Crossing of the Jaxartes River by Alexander the Great’s Forces................................................................................................................... 5
4 Various Possible Board Layouts using Square and Hexagon Location Shapes .......... 8
5 At 15 Feet Across, This Hex Location Could hold Up to 25 Soldiers...................... 10
6 Various Location Types Scale the Reachability of Pieces as They Cross the Particular Terrain................................................................................................................ 10
7 Example of Hexagonal Prisms by Adding Height to Hexagonal Locations.............. 13
8 Possible 3D Solution to Using Hex-Volumes Other Than Prism.............................. 14
9 Simple Example of an LG Trajectory........................................................................... 15
10 Simple Example of an LG Zone ................................................................................ 16
11 Two Examples of More Complex LG Zones............................................................. 17
12 Two Examples of Trajectory Bundles ...................................................................... 18
13 Selected Trajectory ................................................................................................. 19
14 Zone with Primary and Secondary Negations ......................................................... 19
15 Red Indicates Troop Locations: Left Show Scattered Locations, Right Shows Troops Deployed as a Formation................................................................. 21
16 Initial Deployment of Alexander and Scythians at Jaxartes.................................... 26
17 Parallel Trajectories (Left) and Final Formation (Right)......................................... 29
18 Historical Depiction of the Crossing of the Jaxartes River by Alexander the Great’s Forces ................................................................................................................. 30
19 Resource Allocation ............................................................................................... 32
20 First Parallel Trajectories....................................................................................... 32
21 Pieces Move Into Position ................................................................. 32
22 Macedonian Army Advances on Near Shore .................................... 33
23 Scythian Retreats ............................................................................. 33
24 Consolidation of Macedonians Forces, Retreat of Scythians ............. 33
25 Holding the Near Shore of the River .............................................. 34
26 First Parallel Crossing Trajectories .................................................... 34
27 Holding Both Sides of the River ....................................................... 34
28 Final Parallel Crossing Trajectories .................................................... 35
29 Crossing Jaxertes Complete ............................................................. 35
30 First Engagement Trajectories ........................................................ 35
31 Partial Scythian Flank ..................................................................... 36
32 Macedonian Counter Flank ............................................................... 36
33 Scythians Surrounded .................................................................... 36
34 Scythian LG Computing Escape Trajectory ...................................... 37
35 Macedonians Press Attack ............................................................... 37
36 Orange LG Presses Attack Trajectories, Purple LG Continues Retreat 37
37 Scythians Off Map, Macedonians to Pursue ..................................... 38
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG</td>
<td>Linguistic Geometry</td>
</tr>
<tr>
<td>ABG</td>
<td>Abstract Board Game</td>
</tr>
<tr>
<td>LG-SE</td>
<td>LG Strategy Engine</td>
</tr>
<tr>
<td>EBO</td>
<td>Effect Based Operations</td>
</tr>
<tr>
<td>X</td>
<td>Finite Set of Points or Locations</td>
</tr>
<tr>
<td>P</td>
<td>Finite Set of Pieces or Elements</td>
</tr>
<tr>
<td>val</td>
<td>A set of numerical values for each piece.</td>
</tr>
<tr>
<td>SPACE</td>
<td>A state space</td>
</tr>
<tr>
<td>$S_i$</td>
<td>A starting state in the SPACE.</td>
</tr>
<tr>
<td>$S_t$</td>
<td>A target state (or states) in the SPACE.</td>
</tr>
</tbody>
</table>
CHAPTER I

ALEXANDER AND ANCIENT CONFLICT: A HISTORICAL REVIEW

Introduction

The success of the application of Linguistic Geometry (LG) has long been based on the persistent software development and redevelopment employing guidance from subject matter experts – anyone from fighter pilots to seasoned generals [10]. Unfortunately, with ancient conflict, we must become the subject matter experts relatively quickly, via the study of historians like Arrian [2] and Ayrault [1]. Then, the historical accounts of Alexander’s well documented and well known battles need to be analyzed. Finally, the application of LG can be considered.

Some of the greatest criticisms of strategy game engines have been that the game has used a fixed battle plan for the enemy and flexible decisions have only been made for the friendlies [13]. Similar criticisms suggest that even with a flexible strategy for both sides, the domain size for an optimal strategy search becomes intractable [4], or an inherent strategy of the friendly is used homogenously for the enemy as well. An additional purpose of exploring the history of Alexander the Great will be to determine the kind of tactics he favored in his attacks, as well as determine the tactics preferred by his enemies.

While covering the following history, consider this: since the 6th century, chess has continued to fascinate the world as an example of strategy and genius, similar to that, which is found on the battle field. However, war has existed ever since the advent of humanity, which was long before the invention of chess.
Therefore, the algorithms that make up LG certainly cannot come from chess alone, but rather something in general human intelligence. Hence, LG must certainly be applicable to ancient conflict.

Alexander the Great in Ancient Conflicts

After Alexander’s father, Phillip, died, civil unrest and outright rebellion ensued. Alexander’s conquest started by consolidating power within the Greek Commonwealth, which consisted of Macedon, Thessaly, Molossia, Greece, and Thrace (basically, all the neighboring areas except Sparta). Alexander’s first order of business was to ensure the stability of the Macedonian empire that King Phillip had built, without destroying the wealth in infrastructure.

After successfully achieving basic regional unity in Macedon, Alexander headed into Egypt and then into Persia, where he eventually defeated King Darius III, who was not that formidable of an opponent (a coward, really), except that Darius had a large army. After Persia was subdued, Alexander wandered the Eurasia and Indian areas for years, consolidating power, building towns, settlements and infrastructure, and doing all he could NOT to go back to Macedonia and become some kind of royal bureaucrat.
It was during this time of conquest that Alexander had the means to accumulate great wealth (most of which he gave away), including (most of which he kept) elephants. However, after a decades long campaign against nearly the entire known world, the dreaded and feared Macedonian army was relegated to little more than a policing unit in a foreign land, now staffed with a majority of foreign troops.

Figure 2 Path Along Which Alexander the Great Travelled During Years of Campaigns

Alexander’s continued lust for the adventure of conquest led him back to Babylon when his army, desperate and tired, would follow him no longer. Ultimately, Alexander succumbed to an early death of, probably, malaria, complicated by extreme exhaustion and living more in 20 years than most do in 80. Of course, this was not Alexander’s first life-threatening illness; it was just the one that finally caught up with him.

Selecting a Battle to Review

We will attempt to demonstrate a probable LG scenario for one of Alexander’s historic conflict, as it would be useful to see how the LG engine might play out on an actual battle of Alexander the Great, and which zones might be used.
Certainly one of Alexander’s great feats was, in fact, the defeat of Darius’ overwhelmingly large forces, as well as much of the East and India. Alexander was also exceptional at sieging and breaching even the most impregnable fortified cities in the most unforgiving regions (everything from freezing mountains passes to blistering desert wastelands). Since the four main battles of Alexander (Granicus, Issus, Gaugamela, and Hydaspes) have largely already been covered in Dr. Stilman’s previous LG work related to historical conflicts [10][11], it might be more useful to look at a new historical episode. Below is a brief timeline of at least fairly decently documented historical battles (from http://patriot.net/~townsend/GBoH/gboh-alexander.html):

- Lyginus (Alexander vs the Triballians, 335 BC)
- Siege of Pelium (Alexander vs the Illyrians, 335 BC)
- Battle of Granicus (Alexander vs the Persians, 334 BC)
- Battle of Issus (Alexander vs Darius, 333 BC)
- Battle of Gaugamela (Alexander vs Darius, 331 BC)
- Crossing Jaxartes (Alexander vs the Scythians, 329 BC)
- Samarkand (Pharnaces vs Persian rebels, 329 BC)
- Arigaeum (Alexander vs the Indians, 327 BC)
- Hydaspes (Alexander vs Porus, 326 BC)

Alexander observed and exploited the fact that the fortified city could never retreat, giving Alexander as much time as he needed to devise and execute his siege plans. And while LG certainly could be applied to that scenario, as the LG engine is more than capable of determining an efficient siege strategy, an even more dynamic situation is desired.
Thus, through the process of elimination, the event I wish to examine would be the crossing of the Jaxartes, as Alexander would lead an attempt to attack the Scythians, crossing a deep river with steep banks.

**Alexander Crossing the Jaxartes**

After besieging Cyropolis and severely attacking the seven revolting cities, the task of attacking the Scythians would require a trip across the rapidly moving waters. The Scythians, thinking they were in relative safety, taunted and mocked by the Macedonians. Alexander waited until the dry season, where the Jaxartes was passible, near the city of Arbela. Instead of wading through the waters, as was done at Granicus, Alexander’s forces, covered by artillery, crossed the Jaxartes on rafts, which they were able to procure due to their dominance of the region. The archers and slingers were first ferried across to secure the beach head and to keep the pressure on the Scythians through various skirmishes. The Scythians being prevented from approaching the beach, the cavalry and spearmen (phalangites) were ferried across, allowing the horses to swim behind the ferries. Once all were ashore, Alexander attacked. Instead of engaging, the Scythians rode in circles around Alexander’s forces attacking with arrows and bolts (think crossbow).
In any case, Alexander quickly pushed forward with cavalry, which forced the Scythians to form a battle line. With quick maneuvers, the Scythians were constantly out flanked, as Alexander advanced sharply. At this point, the Scythian army unable to cope dispersed. The men were pursued at great length, until thirst forced Alexander and his men to drink from a stagnant pond. They grew ill and abandoned the pursuit. Still, of the original 6000 men engaged, only 160 men were killed (with a thousand wounded). Still, the final operational effect of the battle resulted in tribute and apologies from the King of the Scythians, not to forget the plunder recovered when the Scythians fled, as 1000 of the Scythians were slain and 150 taken prisoner.

**Battle Lines in Ancient Time**

In ancient times, wars were won and lost over position, formation, and holding lines of battle. In modern warfare, land battles are more often fought with tanks and planes, with less emphasis on boots-on-the-ground. Still, even as late as World War II, maintaining strategic battle lines (fronts) was critical to survival, and breaking the enemy’s lines to victory. How does LG form battle lines, if not actual formations? Computations like attrition rates and trajectory qualities are useful ways in guiding which main trajectory to use or which zone to use. How, then, might an efficient zone be created?
CHAPTER II

GENERAL CONSIDERATIONS OF LG IN ANCIENT CONFLICT

Introduction

We need to consider whether changes will be required to the traditional abstract board game (ABG) to implement ancient war strategies (See Appendix B for traditional ABG). We also need to consider whether additional zones need to be created to better represent battle lines or Alexander’s signature “Hammer & Anvil” tactic. The use of heuristics or searches should be minimized; such approaches do not scale well.

A common scenario to consider is a start state where forces are encamped near a battlefield, and a distant army approaches. It is understood that resources will need to be allocated, geography will need to be consider, battle lines will need to be drawn and maintained, while LG will need to force the enemy’s line to collapse. LG already does this very well.

We will attempt to describe exactly how LG would use zones for this ancient conflict. In the process, we will look to see what might need added to the overall ABG definition. In the meantime, we will take time to examine at how the current LG and ABG work, especially with respect to the rudimentary implementation of LG that accompanies this paper.

Understanding the Location Shape

We can consider a location as a tile on a 2D surface, or a prism in a 3D matrix. One consideration is depending on whether pieces move diagonally or horizontally/vertically to the board’s layout, pieces travel different distances to get from one location to another.
For instance, below are five board layouts (where each piece is one unit in height and one unit in width):

![Figure 4 Various Possible Board Layouts using Square and Hexagon Location Shapes](image)

The first board on the left is a typical chess layout, where, if the distance from center to center for horizontal moves, was 1, the distance from center to center for the diagonal moves would be $\sqrt{2}$, or approximately 40% further. The next two boards are brick patterns, which help a little, as the diagonal moves would be $\sqrt{5}/4$, or only approximately 11% further.

The exact number is not as important as the fact that all contiguous locations must lay equidistant from each other. Only the last two boards considered fulfill this requirement. This improvement over the chess board still has the shortcoming of movements not going due north or due east, depending on configuration, being difficult to move in a straight line.

Overall, as the size of the board becomes large enough to support a real-world game, even this drawback would become more inconsequential. In the implementation attached to this thesis, originally an LG engine was constructed to demonstrate reachability for chess pieces on a chessboard. Later as ancient conflict was implemented, the honeycomb board was used. Both are available for future work.
Location Size

Again, the reasoning for the use of hexagonal locations is that the distant from one hexagonal location to any of it six neighboring locations is one unit. How big, then, is a unit? The short answer is that the unit size has to be appropriate for the game application. The long answer is that a unit is distance from the center of one location to the center of another, scaled to be appropriate for the speed of any piece in the game.

For instance, travelling on foot, a particular piece might travel at 3-6 miles an hour, or about 4-8 feet per second. So, if we updated the board every 30 seconds and wanted the piece position to reflect motion, we would want to make unit distance no more than 150 feet (but, probably less).

The size of the distance unit has much to do with reachability (how much distance pieces should cover in one move), which can be determined by knowing the time interval defined in the system and the piece’s velocity. So, how does one define the time interval in the system? Again, it depends on how granularly we want LG to consider the interactions of pieces on the board, and how much time we want to allow the enemy before we re-evaluate the situation. Generally, speaking, 30 seconds is sufficient.

Another consideration for location size has to do with the piece itself. For modern conflict, it is typically sufficient to consider a tank or a plane as a single piece, with an internal count equal to one, in a single physical location. Therefore, each physical location need only be as big as the largest single piece. In contrast, the ancient conflict pieces range in size from as small as single infantry to as large as elephants.
Moreover, each piece has an internal count greater than zero, and that count has to be limited to the max number of pieces allowed per square unit of space. If we assume that for ancient conflict, 15 feet location sizes will be the unit size, and each soldier is something like 3 feet wide and 2 feet thick, we could say maybe 20 soldiers would comfortably be allocable per location. This also implies that two pieces of 10 soldiers in size could join up as one piece, to take up one location, as needed.

**Location Types**

To the right is a color key for the different location types for ancient conflict. This is no means an exhaustive list. Dirt, Grass, and Shallow Water, are all easily passible by all types of pieces. As you move from left to right, the idea is that the locations become less passible, causing the speed of the piece to decrease, or even be hazardous to health and stamina.

For example, an Elephant simply cannot pass through Dense Forrest, while it could pass through mud. However, due to the weight of the animal, it would tend to sink in mud, possibly causing injury to the animal. These properties would reside both with the piece and with the location and would affect the overall reachability of the piece.
Another example would be that infantry might be able to travel further in light forest, than on grassy plain, as shade would minimize the need to consume water. Or, infantry would take some attrition on dirt – especially if following cavalry – as opposed to grass, due to the dust kicked up on open, barren land. Cavalry typically travels fast on dirt and grace, but is slowed down by mud, and cannot base rocky land or dense forest. However, the cavalry, alone, can pass very deep water.

Thus, certain terrain can directly affect the velocity of a piece, i.e., mud slows down everyone and everything, secondary affects can also affect a piece’s reachability. Gravel can cause infantry to exert more stamina than solid ground, which can have the effect of scaling back the infantry’s velocity. Deep water could cause infantry attrition, also causing slow-down in velocity, and so on. For more details, see Appendix C.

The complexity of location types in how piece reachability is affected can seem complicated. However, LG easily handles all of these computations when constructing of zones. These complications better guide the construction of paths and tend to yield routes people naturally would take through the modifying the piece reachability and trajectory quality. Resulting main trajectories tend to be the natural paths that humans, as intelligent agents, would choose as optimal.

**Issues in Scaling Reachability Based on Terrain Type**

With the attached implementation, some interesting practical concerns arose when considering how to handle piece mobility on terrain. As we consider the effect of terrain on piece reachability, it quickly became rather apparent that the historical conflict
configuration of Infantry moving 1 unit of distance per time slice, elephants moving 2 units of distance, and cavalry 3 units, is insufficient granularity.

For instance, if infantry is scaled to 90% of their reachability over a particular location, due to terrain, the options are for it to go 1 unit or no units. As a result, the terrain scaling becomes rather pointless. At the very least, if infantry could go 2 units, scaling would result in the piece going either 1 or 2 units of distance. In reality, we would probably like to see a granularity of 5-10 units of distance per time interval.

Human marches can be as fast as 4 miles an hour, while a sustainable horse’s gallop would be around 8. Thus, if infantry might go 8 units per time interval, cavalry might go 16 units, and Elephants might be somewhere in between (let’s say 12 units). With the reachability basically unchanged, and the time interval still at 30 seconds, now more numerous and smaller unit distances are required. Again, having time intervals that are too large would prevent the LG from being responsive enough to the actions and movements of the opposition, terrain, and so on.

The real impact of the smaller unit distances, in addition to limiting the count for any given piece at that location, would be to massively increase the set of locations that are considered during trajectory computation. This is when pruning via the location and piece properties, as well as other trajectory quality assessments, aids LG in the construction of few and higher quality trajectories. Thus, the overhead of assessing hundreds of properties and conditions is easily justified when closing early millions (or even hundreds of millions) of possible trajectories branches. This becomes one of the key efficiencies in LG computation (hence construction rather than searching).
Location Height Map

Of course, the above discussion assumes that the battle terrain, on which the pieces are moving, is flat. In reality, there is always variation in elevation of terrain— even when the area is considered a plain; you still have the meandering, rolling hills (even the ocean being “perfectly flat” has to, at some point, account for the curvature of the earth).

The conventional way to present a 3D version of the 2D honeycomb game space is to use a kind of height map, an example of which is shown to the right.

Each location is given a height, and a prism is formed. As more complex elevations are rendered, rarely will the tops of the prism be parallel to horizon.

Why consider elevation? When computing distance, the elevation does, indeed, affect the overall distance a piece travels. To increase elevation in height has similar computational limitations as moving within the standard 2D chess layout, namely moving diagonally, the piece travels further than horizontally or vertically. Similarly, using 3D prisms, a piece will move further, if the piece moves up a unit of elevation as it moves a unit in any direction.

A question I briefly considered as a thought experiment was whether a similar solution existed, which might be better than using the prism based height map. This solution
might have each location as a regular geometric volume, equal distant apart in all 3D
direction. Turns out, such a solid does not exist. There are not volumes that can be
densely packed in layers upon layers, where each solid is equidistance from all
continuous neighbors.

If a hexagon is projected into 3D space, and edges
connected as efficiently as possible, the result in
not a regular geometric volume (where all sides are
hexagons). Even if all sides of the hexagon are
coincident with all near neighboring edges, the
closest we can get is still a mix of various regular
polygons (squares, hexagons, pentagons, and
equilateral triangles). Even if the distances from the centers of each volume to the next
are probably one unit, which it is not, the computational complexity of representing
simple topography in these irregular 3D volumes becomes untenable and unintuitive.

*Figure 8 Possible 3D Solution to Using Hex-Volumes Other Than Prism*
CHAPTER III

NEW AND EXISTING ZONES IN LG FOR ANCIENT CONFLICT

Introduction

A brief explanation of how trajectories and zones currently exist in LG is necessary only to aid in the understanding of what a new zone might be. If you want a full explanation of LG in all its control grammars and theoretical complexities, a review of “Linguistic Geometry from Search to Construction” by Dr. Boris Stilman [3] would be appropriate.

Trajectories and Zones in LG

LG begins to build a zone by first constructing a main trajectory for a piece from one location to another, taking into account obstructions.

If we wanted to label locations with their 2D location, a trajectory for the white King could be represented as follows:

\[ T_{WK} = a(1,4)a(2,4)a(3,4)a(4,4) a(5,4)a(6,4)a(7,4) \]

A zone, then, is a kind of collection of trajectories, where each additional trajectory must intersect, within a reasonable time, either with the main trajectory or with a negation.
trajectory. For instance, if another piece, the black Bishop, wanted to block the King, then a trajectory would need to be constructed, such that the Bishop would intersect the King at location (4,4) or (6,4) in 2 time intervals. Such trajectories might appear as follows:

\[ T_{BB1} = a(5,5)a(4,4) \]
\[ T_{BB2} = a(5,5)a(6,4) \]

A zone, Z1, could then be constructed to represent this relationship and be represented as the following:

\[ Z_{WK} = T(WK, a(1,4)a(2,4)a(3,4) a(4,4)a(5,4)a(6,4)a(7,4),7) \]

\[ T(BB1, a(5,5)a(4,4),2) \]
\[ T(BB2, a(5,5)a(6,4),2) \]

In the LGToolkit application provided with this thesis, the only zones generated are the attack and block zone. Other conventional zones exist, of course: block, unblock, dominate, retreat, evade, attack [3],[10],[11]. Here is a more complex example. Here, we have a white King traversing obstructions to reach a location. The two images below represent the only two shortest paths the King can take. Both zones show the first negations (black negating the white trajectories) and the second negation (white negating the black trajectories). There are a couple of things to bear in mind. The first is that the zone that a strategy engine creates is for the piece involved in the main trajectory. It is not necessarily for the other pieces.

Figure 10 Simple Example of an LG Zone
One the strategy engine determines that a negation would occur, a main trajectory for that piece following the negation would be created, and a zone analyzed; this can get very recursive.

The second is that the zone can aid in analysis of trajectory quality. For instance, the zone on the left can be regarded as a better zone than the zone to the right. While both zones allow the white King to find paths that avoid the black bishop, the zone to the right takes the white King closer to the black knight. With the zone to the left, the white rook is able to cover all negation trajectories from the black Knight. We could say that the quality of the left main trajectory of the white King is better than the right.

Figure 11 Two Examples of More Complex LG Zones

To create an attack zone, doing the following:

1. First, take a piece, select a target location, and generate a quality main trajectory.
2. Then, for each enemy, build a first negation trajectory that can reach the main trajectory in time.
3. Then, for each friendly, build a second negation trajectory that can reach the first negation trajectories in time.

4. Then, repeat 2 and 3 until all possible negations have been discovered.

When creating negation trajectories, one must not confuse them with other zones. For instance, a block by the enemy as a first negation is, itself, not a zone, but part of the zone. This information is used by the friendly LG only to adjust, as much as possible, the main trajectories to limit the types and number of negation trajectories by the enemy. This is a critical part of the quality analysis for trajectories.

Similarly, for ancient conflict, a trajectory can be generated.

When trajectories are created, they do tend to get created in what are called Trajectory Bundles. When creating a zone, only a single trajectory is required. As a result, a quality trajectory analyzer is required. This analyzer would consider proximity of enemies and friends, whether paths are blockable, the overall geometry/shape of the trajectory path, and the sustaining of requirements and constraints.

![Figure 12 Two Examples of Trajectory Bundles](image)
For ancient conflict, one of the above trajectories produced for the orange infantry spearman, as shown to the right, would be:

\[ T_{\text{OrInSp}} = a(23,8) \ a(22,8) \ a(20,9) \]

When creating the zone for the aforementioned trajectory, you would get the following zone definition:

\[ Z_{\text{OrInSp}} = T(\text{OrInSp},a(23,8)a(21,8)a(20,9),3) \]

\[ \quad T(\text{PuCaSw},a(21,9)a(20,9),2) \]
\[ \quad T(\text{OrCaSp},a(20,9),1) \]
\[ \quad T(\text{OrCaSp},a(20,8),1) \]
\[ \quad T(\text{PuInSw},a(20,9),1) \]
\[ \quad T(\text{PuInSw},a(20,8),1) \]

The way to interpret this zone is as follows:

1. Main Trajectory is Orange Infantry Spearman, \( T_{\text{OrInSp}} \), of Length 3
2. First Negation is Purple Cavalry Swordsman, \( T_{\text{PuCaSw}} \), of length 2, at \( a(20,9) \)
3. Second Negation is Orange Cavalry Spearman, \( T_{\text{OrCaSp}} \), of Length 1, at \( a(21,9) \)
4. Second Negation is Orange Cavalry Spearman, \( T_{\text{OrCaSp}} \), of Length 1, at \( a(20,9) \)
5. Tertiary Negation is Purple Infantry Swordsman, \( T_{\text{PuInSw}} \), of length 1, at \( a(21,9) \)
6. Tertiary Negation is Purple Infantry Swordsman, \( T_{\text{PuInSw}} \), of length 1, at \( a(20,9) \)

We must also assume the enemy LG would be using the most efficient strategy it can, and would not waste a perfectly optimal attack.
Which is to say, although we do not move the enemy pieces, we assume the enemy will move the pieces to their greatest, long-term advantage. And, even if the enemy chooses some other reallocation of resources, the friendly LG would simply reassess zone construction in real time and adjust accordingly.

**LG Strategies Engine**

One critical component missing from the attached project implementation is a strategy engine. This is the component of software which would determine which zones should be constructed, based on objectives (what does it mean to win?), and restrictions (at what cost?).

The ABG contains all the pieces, reachability, etc…, that is needed to build trajectories and zones. But it is the LG Strategy Engine (LG-SE) that evaluates a list of criteria to determine which kind of zones to create. Is it unacceptable for a piece to be taken? Perhaps a retreat or evade zone is created. Do we want to protect a particular corridor, keeping it free of bad guys?

Perhaps, the LG engine will use a domination zone. Different zone selections for different courses of action can be looked at as the personality of the LG. Do we want the LG punitively aggressive, or overly cautious? We can use this tailoring of personality to better reflect the nature of Alexander (conquest and victory) or Darius (bullying neighboring territories and cowardice retreat). This will yield a more realistic result. Ultimately, though, it is the LG generating zones which will best reflect the intrinsic brilliance found in Alexander.
Now, please, consider figure 15 as a start state and a final state, and consider the two questions posed in Chapter I:

1. *How would LG form battle lines?*

2. *How might an efficient zone be created?*

![Figure 15 Red Indicates Troop Locations: Left Show Scattered Locations, Right Shows Troops Deployed as a Formation](image)

To answer these questions, let us first assume that the nature of the LG-SE will be tailored to emulate more of Alexander the Great. What does that mean? It means, like Alexander the Great, we tailor constraints and restrictions similar to what Alexander the Great would encounter, and observe what zones and main trajectories the LG-SE would generate. If trajectory quality and zone quality is near similar between two different types of zones, and the LG-SE had to choose one, we would have it choose the more Alexandrian choice – perhaps attack or block, dominate over retreat.

The steps of any simulation include allocation of resources and engagement, but these steps have to be in the context of achieving a victory. What is a victory? A victory is the achieving of a final state, while maintaining all mandatory constraints of the game, while not violating any of the restrictions.
For the simple example above, we might say the constraints of this scenario would be as follows (these are made-up constraints with arbitrary numbers, intended as example):

- **Constraints**: Dominate Both Sides of River Banks with All Forces
- **Restriction**: Do Not Allow the Loss of More than 1% of Soldiers
- **Constraints**: Cross the River
- **Restriction**: Do Not Allow the Loss of More than 3% of Soldiers
- **Constraints**: Defeat Opposing Formation
- **Restriction**: Do Not Allow the Loss of More than 10% of Soldiers

With the LG-SE, we will make the assumption that the system will continuously balance allocation or resources and engagement, as necessary. We will also assume that the LG-SE will move seamlessly through the simulation, so long as constraints and restrictions are not violated.

In light of the first constraint and restriction, the LG-SE would initially emphasize deployment and resource allocation over engagement – at least until the battle line is complete. This does not mean that enemy positions will not be attacked. The constraint of “**Dominate Both Sides of River Banks with All Forces**” might be input into the system as domination zones, covering long section of river bank. Then, the LG-SE would begin to determine the deployment of units along the banks.

The restriction of not to losing more than 1% of forces deployed might result in the LG-SE generating a simultaneous convergence on the shore of all forces (what in antiquity might be referred to as a battle line), as well as discrete units of archers and slingers.
acting as point and suppressing enemy locations near the opposite bank. Thus, multiple attack zones would be created to reinforce the numerous domination zones.

**Possible New Zones Required by Ancient Conflict**

As we get closer to a full LG analysis of the Crossing of Jaxarte by Alexander and his forces, we may ask the question as to whether any new types of zone should be created to better emulate the types of movements that were done by Alexander. What simulations have borne out in the more robust Stilman simulation is that the conventional zones are suitable and sufficient to describe 99% (again, arbitrary number) of all scenarios. Even the most complex engagements are described using these conventional zones, which include:

- **Attack:** You Attempt to Occupy Enemy Occupied Location
- **Block/Relocate:** You Occupy Empty Location
- **Dominate/Protect:** You Stay Close to a Piece/Pieces
- **Retreat/Unblock:** You Leave a Currently Occupied Location

There is no restriction as to the types of zones in LG. The types of conventional zones (attack, retreat, block, unblock, dominate) have their origin in Chess. These zones tend to be very universal in nature. As a result, additional zones rarely need to be created, for instance, to emulate Alexander the Great’s brilliance. At most, existing zones might be expanded upon, rather than new ones created. As Alexander’s brilliance is analyzed and re-created, the same conventional zones consistently continue to fall out of that analysis, as LG is used to construct the scenario. This assertion is by no means intuitive. However, it is the fundamental purpose of LG (from search to construction).
Unconventional zones tend to be domain specific, like in urban warfare, where evade zones allow detected pieces to get out of enemy’s line of site, but stay within range for future engagement. Or, a long range attack zone allows an undetected piece to engage from a distance and to move to new locations to mask position. For Ancient Conflicts, similar domain specific zones might be considered, if it would be more efficient a solution.

For instance, what kind of zone would be used to represent the movement of complex equipment? When crossing an especially deep and wide river, cavalry would need to dismount their horses and oar across the river in rafts, while towing their horses (who would be swimming behind them). This could be called a Cavalry Water Crossing zone. Because this zone would be nearly defenseless and quite vulnerable to attack, the LG-SE would provide additional units that could protect those pieces. Perhaps archers and slingers would be positioned on the near shore. Perhaps only half the raft would be filled with Cavalry, while the other half filled with archers. Ultimately, however, this overly complex zone is really only a series of attack, domination, protection, retreat, and blocking zones.

Another ancient conflict zone to consider might be a continuous suppression zone. As an army approaches a battle line, and wants it to move back, archers and slingers could be used to encroach upon a battle line, causing them to fall back, while also providing cover for the advancing army. This would be especially useful when approaching a river to be crossed. Again, though, highly restricted attack zones, supported by infantry protection, could be used to accomplish the same effect.
Hyper Games

When LG games become more regionally separated, hyper games can be used to optimize processing. In the case of Ancient Conflict, as forces cross the river, there really becomes two different games; one for each side of the river. Both of these games would be sub games in what is known as a hyper game. The same friendly LG-SE has to govern all the games in the hyper game. Because of that, pieces can move seamlessly between games, as needed, and certainly pieces geographically in one game can be used to attack pieces within another game, as the LG-SE sees fit. Within each game of the hyper game, the games with their constraints and restriction are largely separate from each other, but that is not entirely required.

Other causes of hyper games include a desire for different time intervals. When Alexander sends his Companion cavalry to flank, the speed at which the horses travel could justify their own game, simply because more is happening in less time. Similarly, when a section of the army is marching to an engagement, while other sections are already engaged, that could be represented as two different games. The marching army section has less going on (perhaps 1 min time interval is sufficient), and more computation should be reserved for the engagement (perhaps 15 sec time interval), since much more activity would be going on.

Another use of hyper games, then, might be to define multiple games with single game objectives that could run simultaneously in the LG-SE. This might simplify the computation of complex objectives into single, serial games with a single, simpler objective. In fact, we could repeat this process and have several, if not many, levels of hyper games abstracting details to minimize complexity of any one game or hype game.
CHAPTER IV

APPLICATION OF LGTOOLKIT TO ALEXANDER THE GREAT’S BATTLE

Introduction

Alexander the Great’s Crossing of the Jaxartes was chosen to be modeled in the accompanying LGToolkit project. This last chapter will detail how a mature implementation of LG would handle this particular historical event, while reviewing other limitations of my current LGToolkit implementation, which will be compared and contrasted with the current implementation of LG by Stilman Strategies. Below is the start state of the Crossing of the Jaxartes:

![Figure 16 Initial Deployment of Alexander and Scythians at Jaxartes](image-url)
The orange pieces are Alexander’s forces. The purple would be the Scythians. The colored tiles are genuinely assigned a terrain type, as opposed to just being pretty colors. Unlike Stilman Strategies, this implementation has no LG-SE, so all strategies have to be manually produced, and the terrain plays little into the actual trajectory construction.

The architecture of this implementation starts with an LG engine, which can be extended programmatically to different universes (chess, ancient, and modern conflicts). The UI is strictly for display and control, and contains no LG business logic, with one minor caveat: the pseudo LG-SE could be said to be hard-coded in the UI.

**Manually Building a Battle Line Zone**

Since the above simulation in the current implementation does not have an LG-SE, we cannot explicitly specify constraints or any real restrictions. However, we can demonstrate the creation of main trajectories for the pieces. The system can then take each piece to the battle line formation.

The above simulation begins by selecting a target location that will act as a rally point. Based on the Jaxarte scenario, where Alexander formed his lines before crossing the waters, the rally point selected is set just on the shore of the river. The application then determines a best-fit line from the Scythians to aid in forming the Alexandrian battle line “parallel” to the Scythian formation.

This is accomplished by first generating a best-fit line through all of the enemy’s pieces and projecting that line through the target location. Since each piece as a location in space, we can use the following to get a relatively good line, assuming the enemy army is lining up its forces:
Next, each location at (x,y) in the game space is checked to see if it fits this line, simply using:

```csharp
double sumX = 0;
double sumY = 0;
double sumXX = 0;
double sumXY = 0;
foreach (Piece piece in ViewManager.EnemyPieces)
{
    sumX += piece.CurrentLocation.Position.X;
}
float slope = (float)((sumXY - sumX * sumY / ViewManager.EnemyPieces.Count) / (sumXX - sumX * sumX / ViewManager.EnemyPieces.Count));
slopeIntercept = (float)(ViewManager.targetLocation.Position.Y - slope * ViewManager.targetLocation.Position.X);
```

Those locations are then added to a list of locations that can be used for the friendly battle line. Finally, starting from the geographical center of the battle line locations and working out, trajectories are computed from the nearest piece to that location (see image on next page).

This final part to the above algorithm could greatly be improved, based more on LG zone construction. Instead of generating a best-fit line manually, the domination zone generated by LG should create a similar line. Moreover, that line would be more sensitive and adaptable to the more granular movements of the enemy. Finally, target locations are determined along that battle line and used to create parallel main trajectories for each piece, as seen below:
The starting state indicated above is largely arbitrary and does not reflect any historically concrete positions. As previously discussed, also made obvious by the appearance of the trajectories, the algorithm is simply starting from the rally point (dark pink), finding the closest piece to that location in absolute Cartesian distance and working its way out until all pieces are accounted for. Naturally, this allows the system to enable the pieces to easily follow the trajectories to the final destination.

Clearly, an LG-SE is needed for the LGToolkit, as issues exist for this extremely simple scenario. For instance, not all pieces arrive at the same time. This leaves pieces vulnerable. Moreover, redeployment must be done to match movements by the enemy, which will be making movements to the friendly’s movements, and so on. Of course, without an LG-SE to manage these sorts of complexities for each time interval, what remains is a simple predefined allocation, and the above simulation basically ends.

This is where synchronized parallel zones might be useful. Such zones would make the above trajectories more strategic, with an ending position more greatly organized than is illustrated above, but you would see the final line begin to form long before the pieces
arrived. This gradual formation of a battle line aids in reducing attrition, as neighboring pieces next to each other, facing in the same direction, force the enemy to attack from the front, where defense is strong and attrition less. Thus, as the pieces move, and the line forms, it would become more difficult for a pre-emptive strike by the enemy.

**Setting Up the Jaxarte Scenario**

Let’s look at the Jaxarte crossing from a strictly LG point of view and begin by defining the ABG. For this scenario, we will use the exact same ABG as the manual battle line example above. We will also state for clarification that the initial game time interval is 30 seconds, and the initial location size is 15 feet. We will need to define our constraints and determine a winning end state. A restatement of the constraints and restrictions originally defined above, but more specific for the Alexander campaign, might be as follows:

1. Secured river banks on nearest side
   a. Suppress enemy attack from reaching nearest side
   b. No more than 0.5% casualties of existing forces permitted

2. Secured the river banks on furthest side
   a. Suppress enemy attack from reaching furthest side and from reaching forces crossing.
   b. Transported cavalry reach 100%, artillery reach 60%, and infantry reach 80%.
   c. No more than 1% casualties of existing forces permitted
3. Routed enemy army
   a. Enemy army in state of surrender or 75% destroyed
   b. No more than 10% casualties of existing forces permitted

We will assume these are reasonable constraints and restrictions. A subject matter expert would probably suggest slightly different numbers or aspects. This would be fine, since LG-SE will perform optimally with whatever numbers are provided. Also, notice that constraints and restrictions are never really achieved, but must be maintained, while the end state must be accomplished. Instead of attempting to define three serial end states in one game, it would be better to define three serial games that each has a single end state.

In addition to the Alexander’s army, other assets should also be explicitly made available to the LG-SE. For instance, by this time, the seven cities along the Jaxertes River were under Alexander’s control. Therefore, any useful assets available in from cities should be designated as assets for use by the LG-SE. Specifically important assets include mobility (horses, carts, wagons, boats, and rafts) and logistics (beans, bullets, and Band-Aids).

Another approach might be to create a single game, where the overall end effect is represented as an overarching LG strategy. For instance, moving the army to the river becomes a high-level game, where Alexander’s army is a single, friendly piece, the Scythian army a single, enemy piece, and even the river as a single enemy piece. Then, a zone could be generated to demonstrate the scenario showing movement via main trajectories and negations. This would allow the LG to manage the over-arching game, where each zone could then be resolved by using a sub-game to represent the details.
LG-SE Executing the Scenario

**Securing the Nearest Bank**

LG-SE would begin by determining what kind of domination zone could be used to suppress fire along the nearest banks.

Several zones might be generated for the infantry and cavalry. However, constraint 1.b would be violated, as infantry and cavalry cannot repel slingers and archers from a distance.

Thus, those zones would be quickly abandoned and LG-SE would then determine that long-range ballistics would be required to accomplish suppression. As a result, the LG-SE would begin considering zones affective for long-range assaults and might settle upon a Continuous Suppression zone (CS zone).
Several of the CS zones would be generated, demonstrating optimizations between a continuous advancing line and individual advancing formations. Due to the mobility of smaller formations, the ability for quicker adjustments, and equivalent suppression coverage, the LG-SE would decide to use the smaller would have the archers and slingers (artillery) approaching formations.

The CS zone the near shore, destroying any enemy artillery on the opposite shore. This would continue until no detectable enemy artillery was detected. Concurrent to the CS zone being executed, a protection zone created for the infantry and cavalry. These units would be kept in the green zone (area protected by archers), and advanced slowly with the artillery,

**Figure 22 Macedonian Army Advances on Near Shore**

**Figure 23 Scythian Retreats**

**Figure 24 Consolidation of Macedonians Forces, Retreat of Scythians**
but just behind the archers and slingers, to deter any close-combat assaults. In addition, resource allocation requirements for rafts to cross the Jaxertes, in anticipation of objective 2.b, would be executed. Some infantry and cavalry not protecting the artillery would be burdened with moving the rafts into position, once the bank was secure. Ultimately, due to the Scythians being unable to cross the Jaxertes themselves, their relative separation from logistical support, having inferior tactics and being less experience fighters, they would yield to the LG-SE’s strategy, and the first main objective would be completed.

**Securing the Furthest Bank**

Now, the system replaces constraint 1.b by constraint 2.c as an active restriction. LG-SE would now consider Unit Crossing zones (UC zones) for various

---

**Figure 25 Holding the Near Short of the River**

**Figure 26 First Parallel Crossing Trajectories**

**Figure 27 Holding Both Sides of the River**
combinations of pieces. To fulfill objective 2.a, each combination of pieces for each zone would be analyzed for casualties and other factors. The LG-SE would also consider the existing CS zone in light of the UC zone, protection zone, and existing attack zones. Specifically, the CS zone might need to be maintained or moved to the opposite shore, to fully achieve object 2.a. The LG-SE would determine the optimum strategy and make moves accordingly. At this time, we must note that it is assumed the enemy side has constantly trying to adjust for all troop movements.

Historically speaking, the Scythians used hit and run tactics. So, no major adaptation to the LG-SE’s strategy would likely have been necessary – just maintaining and pursuing the effectiveness of the CS zone and
minimizing casualties. Moreover, the LG-SE would continue optimizing all zones until 2.b is satisfied.

**Routing the Scythians**

With 100% of cavalry, 60% artillery, and 80% infantry, crossed over the Jaxarte, the system replaces restriction 2.c restriction 3.b. The LG-SE will begin to devise strategic attack zones, protection zones, domination zones, etc…, to continue toward the end state. And, for the first time, the constant allocation and reallocation of the Scythian forces (hit and run tactic) will begin to be more of a consideration in the zone construction. We noticed earlier that this LG-SE would have more of the Alexander personality, so it would prefer to use the phalanx in an attack zone, and flank using the Companion cavalry in a blocking zone.
Because of the hit and run of the Scythians on their horses, the LG-SE would switch to using light Cavalry to thrust through Scythians with an initial blocking zone, in an attempt to break up their hit and run tactics, and follow up with additional short-range attack zones with cavalry and infantry, and continuous suppression from artillery (some of which could be mountable as needed).

Historically, this action caused so many Scythians casualties, that they form a battle line to regroup.

The cavalry would then be replaced by the LG-SE in favor of the infantry phalanx.

The companion blocking/attack zones would still be active and being worked by the LG-SE to fully encapsulate the Scythians. Once the
Scythians realized they were surrounded, they dispersed. Because the rapid retreat of the Scythians, a rapid advance of Alexander’s army would result, since constraint 3.a is not satisfied. The LG-SE would continue (as it were, for days), until thirst would have threatened a violation of restriction 3.b, and then Alexander’s army would be turned back. As demonstrated, the LG grammar is sufficient to create efficient trajectories, construct useful zones, and provide strategic transitions. These strategies could be considered at least on par with alexander the great and his contemporaries. Most importantly, advanced LG algorithms and grammars need not necessarily come from chess, though chess was an important contribution.

**Future Efforts: Effect Based Operation**

We talked about many potential avenues, regarding the application of LG to complex situations, including hyper games, complex game objectives, and even a strategy engine. Regardless of which, one has time to implement, the final state of this effort needs to include Effect Based Operation (EBO). EBO is an even higher level strategy that sits even above hyper games, where an open-ended question, such as “How can the will of a regime be crushed and the regime toppled,” can be translated into a hierarchy of hyper games.
This Hierarchy of hyper games, which eventually reduces down to many managed games, which eventually can be boiled down to a strategy, which then is embodied with a set of tactics, which finally is represented by LG as a set of zones. If this sounds rather complex, well, it is. With any field of research, that next avenue of research requires a respectable understanding of the current theoretical base. To tailor and apply LG in such a way that the net effect garnered is answer to the above question may take some time – but inevitably just time.
REFERENCES

1. Alexander - Theodore Ayrault Dodge

   Translated by Brunt, P., 1976.


   Board Games and Linguistic Geometry (LG) Strategies”, pp. 11-23, Proc. of the
   1st Symp.: Advances in Enterprise Control, JFACC Program, DARPA, ISO, Nov.
   15-16, 1999, San Diego, CA, USA.

5. B. Stilman, V. Yakhnis, O. Umanskiy, “Linguistic Geometry: The Age of
   Maturity”, J. of Advanced Computational Intelligence and Intelligent Informatics,
   Vol. 14, No. 6, 2010 (to be published).

   Technology for Simulation Science VI”, pp. 403-414, Orlando, FL, USA, April 1-
   5, 2002.

7. B. Stilman, V. Yakhnis, O. Umanskiy, Linguistic Geometry and Evolution of
   Intelligence, ISAST Trans. on Computers and Intelligent Systems, 16 pp.,
   submitted in April 2010.

8. B. Stilman, V. Yakhnis, O. Umanskiy, R. Boyd, Adversarial Reasoning and
   Technology for Simulation Science IX”, Orlando, FL, USA, 2005.


APPENDIX A

BRIEF HISTORY OF LINGUISTIC GEOMETRY (LG)

Pioneer

The journey from intractable searches through endless trees, to an elegant solution through construction, began by Dr. Mikhail Botvinnik in the late 1950s. Believing that he could discover the algorithm that could arrive at the same conclusions about chess as world famous chess masters, Dr. Botvinnik worked tirelessly through the 1960s to publish his first work on the subject in 1970. From 1972 to 1988, Dr. Boris Stilman, upon completing numerous higher degrees in Mathematics, Computer Science, and Electrical Engineering, assisted Dr. Botvinnik in that research, which eventually became the project known as PIONEER. This project formalized the approach believed to be used intuitively by great chess players into a set of algorithms capable of solving complex chess endgames efficiently through construction and not search.

Linguistic Geometry

Later, Dr. Stilman would formalize the algorithms used in PIONEER into
various Chomsky based control grammars that could be used to prove some solutions to
problems constructed as being optimal. By the early 1990s, Dr. Stilman formalized the
grammars and complex system of sets and functions of the Abstract Board Game, into a
new field of study, coined Linguistic Geometry: Linguistic because of the use of
grammars, and geometry because of the graphs of trajectories and zones that are formed.

LG is known as a hierarchy of languages because of its use of one grammars used to
compute trajectories, another to compute zones, and another to compute transitions. A
famous example used to demonstrate the effectiveness is both Pioneer and LG (not to
mention one of Dr. Stilman’s favorite assignment) is the Richard Reti endgame (can
white force a stalemate?). Pioneer can explain not how white can cause a stalemate, but
LG can explain why it is possible, through the use of various zones.
APPENDIX B

ABSTRACT BOARD GAME DEFINITION

Traditional LG Abstract Board Game

ABG consists of the set, <X, P, RP, SPACE, val, Sᵢ, Sᵢ, TR>:

X: Finite Set of Points or Locations

P: Finite Set of Pieces or Elements, where P is the union of the disjoint sets of \( p₁ \) and \( p₂ \), which we can consider as opposing sides.

val: A set of numerical values for each piece.

SPACE: A set of the current state of all the locations, on which location which pieces reside. In addition, there exists a function, ON(p), which will return a location \( x, x \in X \), for any \( p, p \in P \).

Sᵢ: A state in the set SPACE, which indicates the starting state for a particular game configuration.

Sᵢ: A state (or states) in the set SPACE, which indicate the end of a particular game.

\( R_P(x,y) \): A function that returns the number of moves of a piece \( (p \in P) \) can make from location \( x \in X \) to location \( y \in X \).

TR: A function that manages the removal of a piece from one location and the addition of that piece to another location. This function will remove the piece at the target location. This function, TR(p, x, y), has the precondition that \( ON(p) = x \) and \( R_P(x,y) = 1 \).
Historical Conflict Abstract Board Game – Definition

In the warring days of Greece, consider the following for an ABG definition:

X:       Finite Set of Points or Locations

A standard hex honey-comb pattern of locations would be used. The distance from one piece to the other can be scale appropriately to the pieces.

P:       Finite Set of Pieces or Elements

P is the union of the disjoint sets of \( p_1 \) and \( p_2 \), which we can consider as opposing sides.

With respect to ancient battles, pieces need to include not only soldiers and chariots, but formations like the phalanx, where the piece may be converted to smaller, autonomous units during battle. For example, if a phalanx was decimated and broke apart, based on its health at the time of explosion, we could substitute a number of soldiers for the phalanx at the phalanx’s previous location. In addition, as soldiers congregate (and the piece density on a location increase), there could be a minimum threshold for a group of pieces to be replaced with a larger formation.

\( \text{val}: \) A set of numerical values for each piece.

Each piece will have a set of attributes, related to health, strength, speed, compass orientation, and so on. As a piece engages the enemy, it also needs to track the damage it receives. The more damage, e.g., the worse the health, the more the strength and speed of the piece is inhibited, which affects the pieces motility and affective attack. One might also consider the notion of general conditioning and current stamina, which decreases slowly as the piece engages in fighting. As a piece fights, it gets winded, which affects its ability to inflict damage and protect against attack.
If the general conditioning of the piece is good, the current stamina decreases slowly as fighting ensues, but increases rapidly during periods of rest. For example, as a piece fights, its stamina begins to decline. The piece fights for several time periods. As the piece is victorious, it may take several moves (time steps) to get to the next engage-able target, which provides time for the stamina to increase.

**SPACE:** A set of the current state of all the locations, on which location which pieces reside. In addition, there exists a function, ON(p), which will return a location \( x, x \in X \), for any \( p, p \in P \).

But there is more that will affect health, speed, and strength, than just the piece’s attribute. Several values need to be considered, especially when considering how to form a battle line, related to locations. For instance, there should be a small attrition value assigned to each location, as a piece moves across it. For example, it is entirely believable for a piece to experience damage as it travels – especially along distances – whether it is a formation or a chariot.

Each location, then, should have a mapping to several attributes for each type of piece. For each piece, there should be costs associated with traversing the location. The various costs should affect the attributes of the piece. For example, there is a possibility cost, which will affect the piece’s speed and stamina. There should be an attrition cost that affects the piece’s health. The net result is to affect the equations that are used to compute a piece’s trajectory and zone. For instance, on a battle field, chariots would naturally migrate to a more flat location, where the possibility cost less and the damage to the chariot would be less, especially as compared to a more rocky terrain. Dry locations would more preferable to wet obstructions or even muddy locations.
In doing this, we can also simplify the relationship between the location and obstructions. Up to now, in LG, an obstruction was considered either hard, or impassible, or soft, optionally considered during zone computation. To that, we have already considered adding a new obstruction called a wet obstruction. However, perhaps this is not needed. Perhaps, all the attributes of a wet obstruction can be emulated in a location’s attributes mapping for each piece. Then, the idea of hard obstructions is not required. Rather, you would just set the cost of the location you deem impassible to relatively high, if not an arbitrary high, value.

\[ S_i, S_f: \] **A state in the set SPACE, which indicates the starting state and the finishing state or states (respectively) for a particular game configuration.**

The start state and end state also require some reconsideration. For instance, in Ancient Conflict the start state could be anything from pieces encamped to pieces already in formations to formations already in a battle line. The finish state would be, ideally, a state in which the enemy is routed or decimated. This latter notion is hard to articulate, as each battle is unique. Thus, for a victory to be declared, the finishing states would be subject to the current situation, where the final state would need to represent the realization of a set of strategic states.

To that end, we need to introduce a convention of sorts, with regards to defining what a battle line is. A battle line cannot consist of adjacent soldiers, but rather adjacent formations. But being adjacent is insufficient. Each formation (at least on a hex board configuration) can have at most six orientations. By convention, if we can keep the orientations of formations, in adjacent, contiguous locations, in parallel, then we can say that a battle line exists between those two locations.
As more formations, whether phalanxes or chariot groups, fall into the battle line, then we can say the battle line becomes longer. When two adjacent locations of a battle have less than a formation, or a formation that is not parallel, we can say that a whole has developed in the battle line. Holes in the battle line allow enemy forces to go in behind the battle line, where the attrition is higher. Lastly, we can say that a battle line that has no holes is impassibly by the enemy.

\textbf{R}_P(x,y): \ A \ function \ that \ returns \ the \ number \ of \ moves \ of \ a \ piece \ (p \in \ P) \ can \ make \ from \ location \ x \in X \ to \ location \ y \in X.

How do pieces move from the start state to the end state? In order for a battle line to be formed and be maintained, especially before engagement with the enemy, staging needs to occur. Staging is an intermediate strategic state that allows pieces to assembly, congregate, and group, a ways away from the enemy. Once a battle line is formed, pieces will also need to adjust their internal velocity to maintain their relative positions. This will not always be possible, especially as pieces move across terrain that affects their motility. A battle line may compress, with holes in the line showing up for a time step or two, as well as stretch. But, it must never be allowed to break – certainly not in non-combat situations.

\textbf{TR}: \ A \ function \ that \ manages \ the \ removal \ of \ a \ piece \ from \ one \ location \ and \ the \ addition \ of \ that \ piece \ to \ another \ location. \ This \ function \ will \ remove \ the \ piece \ at \ the \ target \ location. \ This \ function, \ TR(p, x, y), \ has \ the \ precondition \ that \ ON(p) = x \ and \ R_P(x,y) = 1.

With regards to staging, we might say that a battle line is created as a sub-target.
This could be a series of wet obstruction, non-parallel and mostly orthogonal to the main, central trajectory, a narrowing of the plain by hard obstructions, like steep embankments, or just some arbitrarily chosen line. Each piece would take the shortest route to the battle line. The largest number of moves, call it mMax, for some piece to reach the battle line would then be the target number of moves for all pieces to cover their distance to the battle line.

In other words, while, perhaps, over half of the pieces could arrive with time to spare, it would be more beneficial for all pieces to arrive together, should the enemy begin the attack before the battle line was fully formed. Thus, after each concurrent turn, the battle line would become more defined. This can be accomplished by taking mMax, and only covering a ratio of the remaining distance for a given piece. This means that more advanced pieces might skip a turn, or simply adjust their velocity vector to move slower, etc....
Extending ABG Definition Properties

Given the above, what do the actual changes look like to the underlying LG engine? We will start by expanding the original ABG definition, where ABG consists of the set, \(<X, P, R_P, SPACE, S_i, S_t, TR, GRP, MAP, ATT>:\n
<table>
<thead>
<tr>
<th>Piece</th>
<th>Macedonia Icon</th>
<th>Enemy Icon</th>
<th>MAP_v(P,X)</th>
<th>MAP_A(P,X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archers</td>
<td></td>
<td></td>
<td>1</td>
<td>.05</td>
</tr>
<tr>
<td>Slingers</td>
<td></td>
<td></td>
<td>1</td>
<td>.06</td>
</tr>
<tr>
<td>Spearmen</td>
<td></td>
<td></td>
<td>1</td>
<td>.07</td>
</tr>
<tr>
<td>Swordsmen</td>
<td></td>
<td></td>
<td>1</td>
<td>.08</td>
</tr>
</tbody>
</table>

X: Finite Set of Points or Locations, where each location contains the following attributes: **Type, Passibility, Size, Position, and Attrition**

P: Finite Set of Pieces or Elements, where P is the union of the disjoint sets of \(p_1\) and \(p_2\), which we can consider as opposing sides, and where P contains the following attributes: **direction (D), motility (M), strength (ST), velocity (V), stamina (S)**,
health (H), effective lethality ($L_E$), range of lethality ($L_R$), armor (AR), minimum grouping count (MC).

P is the initial set of pieces by both their kinds and counts. Moreover, P can be modified by TR and GRP.

For ancient battles, there are 4 basic pieces: Archers, Slingers, Spearmen, and Swordsmen, all of which could be Soldiers, Cavalry, or on Elephant, where each of the above attributes must be carefully defined for each of the above pieces, after much trial and error. Notice that defensibility and mobile might increase dramatically with being cavalry, but accuracy would be reduced, for instance.

On a historical note, Alexander did develop a fascination of elephants in his later years (late twenties, early thirties). He collected huge herds. Unfortunately, he never really had

<table>
<thead>
<tr>
<th>Piece</th>
<th>Macedonia Icon</th>
<th>Enemy Icon</th>
<th>MAP$_V$(P,X)</th>
<th>MAP$_A$(P,X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cavalry</td>
<td>Cavalry</td>
<td>8</td>
<td>.01</td>
<td></td>
</tr>
<tr>
<td>Archers</td>
<td>Slingers</td>
<td>7</td>
<td>.02</td>
<td></td>
</tr>
<tr>
<td>Spearmen</td>
<td>Cavalry</td>
<td>6</td>
<td>.03</td>
<td></td>
</tr>
<tr>
<td>Swordsmen</td>
<td>Cavalry</td>
<td>5</td>
<td>.04</td>
<td></td>
</tr>
</tbody>
</table>
an opportunity to deploy them in battle. Alexander relied largely on his Companion Calvary, as well as his phalanx. Though, after Alexander headed east into open plain of Eurasia; he discovered that the vast territory negated the benefits of his phalanx.

**SPACE**: A set of the current state of all the locations, including on which location which pieces reside. In addition, there exists a function, ON(p), which will return a location \( x, x \in X \), for any \( p, p \in P \).

**\( S_i \)**: A state in the set SPACE, which indicates the starting state for a particular game configuration.

**\( S_e \)**: A state (or states) in the set SPACE, which indicate the end of a particular game.

<table>
<thead>
<tr>
<th>Piece</th>
<th>Macedonia Icon</th>
<th>Enemy Icon</th>
<th>MAP(_1(P,X))</th>
<th>MAP(_A(P,X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elephant Archers</td>
<td><img src="image" alt="Macedonia Icon" /></td>
<td><img src="image" alt="Enemy Icon" /></td>
<td>1</td>
<td>.01</td>
</tr>
<tr>
<td>Elephant Slingers</td>
<td><img src="image" alt="Macedonia Icon" /></td>
<td><img src="image" alt="Enemy Icon" /></td>
<td>1</td>
<td>.02</td>
</tr>
<tr>
<td>Elephant Spearmen</td>
<td><img src="image" alt="Macedonia Icon" /></td>
<td><img src="image" alt="Enemy Icon" /></td>
<td>1</td>
<td>.03</td>
</tr>
<tr>
<td>Elephant Swordsmen</td>
<td><img src="image" alt="Macedonia Icon" /></td>
<td><img src="image" alt="Enemy Icon" /></td>
<td>1</td>
<td>.04</td>
</tr>
</tbody>
</table>
TR: A function that manages the removal of a piece from one location and the addition of that piece to another location. This function will remove the piece at the target location. This function, TR(p, x, y), has the precondition that ON(p) = x and Rp(x,y) = 1.

Note that we list each elephant as if it had a rider like a cavalry piece, when, in reality, it would hold several persons, generally archers or slingers, and it would be a mix. Another inaccuracy is that Alexander the Great never really used elephants for most of his campaigns. However, for the sake of symmetry and completeness of the LG engine, we have included this breakout.

Adding New ABG Definition Properties

GRP(P) The group function will evaluate each piece after a turn has occurred, and decide if the strength of the piece warrants breaking the piece into smaller units, if possible, or whether neighboring pieces of like kind, which are consolidatable, should be consolidated. Group will run just before each turn.

MAPV(P,X): P is a piece, and X is a location. Based on the piece and the location, this mapping returns the affective scalar for adjusting velocity for a piece over this location (where un-scaled velocity is 1).

MAPA(P,X): P is a piece, and X is a location. Based on the piece and the location, this mapping returns the affective attrition of a piece from crossing this location, a structure containing the cost for Health, Stamina
ATT(p_a, p_d): Attrition takes 2 pieces: attacking piece and defending piece. The attrition
takes into account all piece attributes, and generates a reasonable damage
that gets applied to the piece’s properties. If the pieces continue engaging,
ATT will be called for each time step. If one piece flees, direction and
position of attack will be accounted for. The idea of attacking and
defending annotation is only for distinction, and would play no part in the
actual attrition computation.

MAP_5(P,A): P is a piece, and A is an attribute. Based on the piece and the attribute, this
mapping returns a scaled value for attrition of a piece over this location.
For instance, if a given piece is computing its velocity over a terrain, it
passes its velocity attribute, and the resulting output is a scaled velocity
which will take into account the piece, the terrain, and how that affects the
velocity

As a quick aside, notice, in the previous chart, how the Persians and the Macedonians
have the same values for the various types of pieces. However, we know that these values
would be different, based on the particular battle scenario. Moreover, these values would
likely, but not always, slightly be adjusted in favor of the Macedonians.

In addition, the values listed should be considered initial values, which eventually would
constantly be changing. Some properties would go down, like stamina, slowly when
travelling or quickly in battle. Some properties would go up, like health, slowly when
resting, or stamina, quickly when resting. And so on.
APPENDIX D

LG TOOLKIT APPLICATION ARCHITECTURE

Introduction

The Following is a summary of the LGToolKit application architecture. The full source code can be downloaded at http://www.gammeter.com/LGToolKit.zip. Note: the use of XNA, and the fact XNA is now deprecated by Microsoft, make this solution only compatible with Visual Studios 2010.

Project Overview

LGToolKitUI – XNA Application

This is the XNA Application that ties everything else below into a usable program. It relies heavily on the correctness of the GameLibrary and the LGEngine. This application is simply a collection of Views, which basically includes Play, Load, Edit, New, and Save boards. The views are intended to be used and managed via the ViewManager class.

GameLibrary – XNA Class Library

This game library is a full 3D XNA engine, which supports audio, UI (keyboard, mouse, picking), 3D rendering (camera, lighting, fog, skybox, model management, ), 2D rendering (sprites, animated sprites, text), physics (collision detection, particle engine), and basic controls (button, toolbar, dialog box). This library is intended to be used and controlled via the GameManager class.
**LGToolKitXnaContent – XNA Content Library**

All media is stored in this library. This includes all xna font definitions and textures. This could also include any audio or 3D models, if desired.

**LGEngine – C# Class Library**

The LGEngine consists of the GameSpace, which can be thought of as the Abstract Board Game, and everything else defined and used in the Abstract Board Game (Locations, Pieces, Trajectories, and Reachability). Some external UI API support is provided through the details class. Location and LocationHex support square and hexagonal location shapes. The Piece class defines all common LG piece properties and functionality, while the Obstruction class is a type of piece definition support by all LG ABGs. The Trajectory class supports most of the LG trajectory and zone logic, which the reachability class handling the computation of the reachability table for a piece and a set of start and end locations. The Piece class actually stores a hash table of multiple reachabilities, to avoid reprocessing.

**LGAncientConflict – C# Class Library**

This library implements the Ancient Conflict Abstract Board Game. In addition, each Ancient Conflict piece (with reachability) is defined in the Pieces folder, as well as a AncientConflictPiece base class inherited by the pieces.

**LGModernConflict – C# Class Library**

This library implements the Modern Conflict Abstract Board Game. In addition, each Modern Conflict piece (with reachability) is defined in the Pieces folder, as well as a ModernConflictPiece base class inherited by the pieces.
**LGChess – C# Class Library**

This library implements the Chess Abstract Board Game and the Chess Square. In addition, each chess piece (with reachability) is defined in the Pieces folder, as well as a ChessPiece base class inherited by the pieces.

**SharedUtilities – C# Class Library**

This is a collection of classes originally used to support the Microsoft MVVM architecture. Specifically, the Relay class, the NotifyPropertyChanged class, and the NullImageConverter class, are stored here. Since moving to XNA, they are no longer used. Also, here is stored the EnumHelper class, which provides access to reading enumeration description attributes, as well as the Error Notification class used for logging and debugging purposes.