Linguistic Geometry Paradigm: From Fighting Wars ... To Computing Them

Boris Stilman[1,2], Vladimir Yakhnis[1], Oleg Umanskiy[1,2]

[1] STILMAN Advanced Strategies, 1623 Blake Street, Suite 200
Denver CO 80202, USA
{boris, vlad, oleg}@stilman-strategies.com
www.stilman-strategies.com
[2] Department of Computer Science, Campus Box 109,
University of Colorado at Denver, Denver, CO 80217-3364, USA
{boris.stilman}@cudenver.edu

Abstract. This paper describes application of Linguistic Geometry (LG), a new type of game theory, to real life wargaming. LG is compared to conventional gaming approaches with respect to their applicability to wargaming. It is shown that LG generates winning strategies for all sides in a conflict in real time by constructing them out of a limited set of blocks called zones. Several examples of zones are introduced. The paper describes the process of discovery of new zones essential for various types of military operations. The paper includes brief description of scalability of the LG-based applications.

1 Introduction

For centuries, wargames have been used to guide national decision-makers in wars [2, 4, 22]. The ultimate goal was "to compute the war", i.e., to predict the outcome of the war and generate the best strategies for all the sides in a conflict to guide the friendly side. Science fiction novels and movies such as "Star Track" have gone even further and predicted complete elimination of wars if such goal is achieved. In ancient times many existing entertainment games (including chess) have been developed by abstracting from real life wars. Over time, widespread use of these games led to appearance of very strong players, to the development of powerful gaming strategies and to the attempts of using similar strategies in real conflicts. The 19th century wargames of the Prussian General Staff were chess-like in their focus on the mechanics of strategy and movement. The wargames emerged during World War II included detailed modeling based on the attrition factors impacting forces in combat. Subjectively, over the years, some of the military analysts and commanders became accustomed to attrition/movement wargaming and acquired belief that they are "good enough". However, even the most prominent champions admit that these wargames lack truly challenging and unbiased opponent. Often, the Red and Blue teams for combat simulations are chosen from among individuals having very similar military background and training. This creates a natural bias in the strategies employed, that can often completely miss the real-world strategies that would, in fact, be employed

by individuals with decidedly different cultural and military backgrounds. In addition, attrition/movement wargaming ignores specificity of the modern conflicts, which shift toward minimizing casualties of all the sides and involve new types of operations like asymmetric warfare and effects based operations. None of the existing wargaming models has even approached the ultimate goal of wargaming: they are incapable of generating the best strategies for all sides in a conflict.

The idea of improving wargaming by utilizing Artificial Intelligence (AI) has been around since the beginning of the computer era. It is conceivable, that such systems would model an intelligent unbiased opponent in the wargames. In such case, the friendly Blue side would be trying to anticipate and predict the Red action; the Red would actively be trying to do the same with respect to Blue while simultaneously trying to deny Blue sufficient information on which to predict Red actions. To be useful, such models should have an ability to be altered on request of an analyst or a commander to take into account cultural and social idiosyncrasies of different parts of the world. Further, such systems should be able to optimize their actions against various criteria to reflect different types of military operations including asymmetric warfare, effects based operations, and future types of warfare. Such wargaming system should be truly dynamic, adaptable to the strategy and tactics of its human adversary, and be able to discern what concepts work best to defeat the opponent. The adaptability of the simulated adversary would force the human wargamer to avoid predictable patterns of behavior and constantly rethink his/her solutions. In the near term, such wargaming systems would create the best training and simulation environments for human wargamers. In the future, they may allow us to approach the ultimate goal of wargaming.

Unfortunately, the existing AI-based tools do not include faithful and scalable models of intelligent enemy [7, 15]. Such tools usually utilize a fixed battle plan for the enemy while using flexible decisions of human players for the friendly side. This practice results in a narrow construction tied to stereotypical current practices of what is considered the "doctrinally correct behavior". Clearly, such systems lack objectivity by being heavily skewed in favor of the friendly forces. In attempt to avoid combinatorial explosion, i.e., exponential run time ([8] and Section 5), the existing tools do not provide any look-ahead necessary to determine results of a protracted action. Without look-ahead these tools cannot adapt to the opponents behavior and evaluate remote indirect effects of their actions.

We address all these problems by bringing to bear a wargaming technology providing a revolutionary deep *look-ahead* with unmatched *scalability* and a faithful model of *intelligent enemy*. This technology is based on Linguistic Geometry (LG) [16], a new type of game theory for finding winning strategies for multi-player concurrent adversarial games. The word "linguistic" refers to the model of strategies formalized as a hierarchy of formal languages. The word "geometry" refers to the geometry of the game board as well as the abstract relations defining the movements and other actions of the game pieces as well as their mutual potential influence on each other (Section 5). The game board represents the battlefield terrain including land, urban environment, sea, air space, near-Earth space, etc. The abstract relations represent movements of battlespace entities such as ships, tanks, fire teams, aircraft, missiles, etc., and their actions including applications of weapons, sensors, and com-

munications.

LG was developed by Dr. Stilman beginning from 1972. For 16 years, in the former Soviet Union, Dr. Stilman had been involved in the research project PIONEER [1] for modeling strategies of advanced experts in solving complex gaming problems almost without search. This project was led by the former World Chess Champion Professor Mikhail Botvinnik, who, himself, had served as the main source of advanced strategies on this project. In the 90s, in the USA, Dr. Stilman generalized these results in the form of the theory of LG. The LG-based technology was developed by STILMAN Advanced Strategies [11] founded in 1999. Since then, this technology was successfully employed in 30 defense-related projects [11] in the USA and abroad. Over the last three years, LG was validated within one of the DARPA (Defense Advanced Research Projects Agency) flagship programs called RAID (Real-time Adversarial Intelligence and Decision-making) [3, 18], which is currently being transitioned to the US Armed Forces, to the Army Intelligence DCGS-A (Digital Common Ground System – Army) and to the Army Battle Command FBCB2 (Force XXI Battle Command Brigade and Below). Within RAID, LG generates courses of action for MOUT (Military Operations for Urban Terrain). On multiple experiments, LG has successfully demonstrated the ability to solve extremely complex military scenarios in real time. The efficacy and sophistication of the developed courses of action consistently exceeded the level of those developed by the human commanders and staff members.

LG may be structured into two layers: game construction and game solving. Construction involves a hypergame approach based on a hierarchy of Abstract Board Games (ABG). Game solving includes both resource allocation for constructing an advantageous initial game state and strategy generation to reach a desirable final game state in the course of the game.

2 Traditional Gaming Approaches

Consider traditional gaming approaches utilized in various AI-based systems [14]. The games are usually classified as *continuous vs. discrete* and *strategic vs. extensive*. For two major classes of games, continuous and discrete, theoretical results including algorithms have been obtained. However, these algorithms lack scalability and, therefore, are not applicable to solving medium and large scale problems.

Continuous games are usually described mathematically in the form of pursuit-evasion *differential games*. For example, these games are employed for representing several autonomous aerial vehicles planning a route through territory guarded by aggressive counterparts. The classic approach based on the conventional theory of differential games [6] is insufficient, especially in case of dynamic, multi-agent models [5, 10]. It is well known that there exist a small number of differential games for which exact analytical solutions are available. There are a few more differential games for which numerical solutions can be computed in a reasonable amount of time, albeit under very restrictive conditions. However, each of these games must be one-on-one, which is very far from the games modeling real world operations. They are also of the "zero-sum type", which does not allow the enemy to have goals other

than diametrically opposing to those of the friend (which is the case for asymmetric actions). Other difficulties arise from the requirements of the 3D modeling, limitation of the lifetime of the agents, or simultaneous participation of the heterogeneous agents such as land, navy, and aerospace units.

Discrete strategic games were introduced and investigated by Von Neumann and Morgenstern [21] half a century ago and later developed by multiple followers [14]. This approach allows analyzing full game strategies, representing entire games. It does not allow breaking a game into separate moves and comparing them. Only full strategies and the entire courses of behavior of players can be compared. Each player chooses his/her plan of action once and for all and is not informed about of the plan chosen by another player. This significant limitation makes this approach inadequate for the modern wargaming.

Discrete extensive games specify the possible orders of events; each player can consider his/her plan of action not only at the beginning of the game but also whenever he/she has to make a decision [9, 14]. Extensive games are usually represented as trees, which include every alternative move of every strategy of every player. Application of this class of games to real world problems requires discretization of the problem domain, which can be done with various levels of granularity. The extensive (unlike strategic) games can represent concurrent moves. This is similar to the real world, where all the pieces (ships, tanks, aircraft, etc.) and players (Red and Blue) move and act concurrently. Thus, the extensive games would allow us to adequately represent numerous real world problem domains including modern wargaming. However, the classic game theory considers real extensive games (like chess) trivial for "rational" players because an algorithm exists that can be used to "solve" the game [14]. This algorithm defines a pair of strategies, one for each player that leads to an "equilibrium" outcome: a player who follows his/her strategy can be sure that the outcome will be at least as good as the equilibrium no matter what strategy the other player uses. Finally, the classic theory of extensive games is not focused on the actual *tractability* of this algorithm, which makes it irrelevant to practical wargaming [14].

Practical gaming approaches try to solve discrete extensive games by searching through the game tree. The main difficulty for any practical gaming is scalability, i.e., "the curse of dimensionality" (Section 5). Even for a small-scale conflict, an extensive game would be represented by a game tree of astronomic size, which makes this game intractable employing conventional approaches. Consider, for example, a small concurrent game with 10 pieces total where each piece can make 10 distinct moves at a time. If the game lasts for at least 50 moves (not unusual for real life conflicts), the size of the game tree would be about $(10^{10})^{50} = 10^{500}$ nodes. To be more specific, employing practical gaming approaches the DARPA RAID game would result in the game tree with more than $10^{8,392}$ nodes [18]. No computer can search such trees in a lifetime. The popular search reduction technique, the alpha-beta pruning, is insufficient for the game trees of such magnitude. In the best case, the number of moves to be searched employing alpha-beta algorithms grows exponentially (as the square root of the original exponentially growing game tree) [8]. Moreover, alpha-beta is invalid for concurrent games. Thus, conventional gaming approaches are not applicable to extensive concurrent games modeling real world military operations. In contrast to the other approaches, the LG approach demonstrated the low degree polynomial run

time, which makes it applicable to real life wargaming ([16] and Section 5).

3 LG Zones

The basic building block for the LG strategies is an LG zone. Intuitively, a zone is a network of trajectories drawn in the game board representing action-reaction-counteraction-etc. of the players involved in the game (Fig. 1). Zones represent optional local skirmishes built out of several well organized actions, reactions, counteractions, retreats, etc. In the theory of LG, zones represent the "genetic code" of an ABG. A winning strategy in ABG is not searched over the tree but constructed by generating a hierarchy of high-quality zones and by moving pieces along the trajectories of those zones. For ABG representing military operation a winning strategy can be viewed as a detailed schedule for concurrent movement of entities along the virtual rail-tracks constructed of zones in 3D space.

Consider informally a complete set of different zones for a serial ABG such as the game of chess. Formal definitions of these zones are given in [16]. This set includes just five different types of zones: attack, block/relocation, domination, retreat and unblock. These five types represent a complete set of "amino-acids" of the game of chess. Examples of such zones are shown in Fig. 1 [16].

For the attack zone, the attack side (including pieces p_O and p_1) is intended to destroy the target q_1 while the intercept side, q_1, q_2, and q_3, is intended to protect it. For the block zone the attack side is intended to block the trajectory of q_1 by relocating p_O to point 4, while the intercept side is intended to prevent this relocation. This zone is linked to the attack zone of the piece q_1. In general, for a relocation zone, p_O is intended to occupy point 4, but the purpose of that might vary from block to other types of participation in an attack zone. For the domination zone, the attack side is intended to intercept q_1 at point 4 by dominating this point from point 3 (employing relocation of p_O), while the intercept side is intended to prevent this domination. This zone is linked to the attack zone of q_1. For the retreat zone, the attack side that includes q_O is intended to save q_O from the attack of p_O by moving it away from the destination of the trajectory of p_O; the intercept side that includes p_1 is intended to prevent this retreat. For the unblock zone, the attack side is intended to support the attack of p_O along its trajectory by moving the blocking piece p_2 away, while the intercept side (piece q_1) is intended to prevent this unblock. Both zones are linked to the attack zone with main piece p_O.

Every class of ABG requires a specific set of types of LG zones. For example, the totally concurrent chess-like ABG are solvable employing a set of five types of zones that are similar to those shown in Fig. 1 [16]. Classes of ABG that represent real world military operations also require sets of specific LG zones. Air Force, Naval, Missile Defense and MOUT wargames - all require different sets of types of zones. It is interesting that all of them are very similar and the number of such types is small: it varies from 5 to 10 for each class of operations. Once a complete set of zones for a class of ABG representing a class of military operations has been discovered and

introduced into the LG tool, this tool will solve all the games from this class, i.e., it will generate advantageous strategies for all the operations from the respective class. In the next section we consider the process of discovery and generalization of the new types of LG zones.

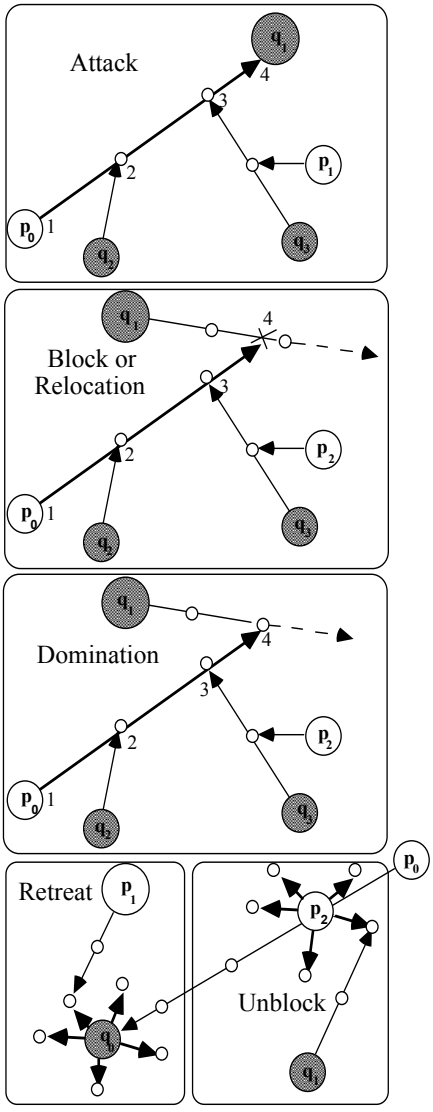

Fig. 1. Various types of serial LG zones

4 Development of New Types of Zones

Every new zone has a development cycle going through several levels of generalization from specific rules to principles to a general conceptual zone. Examples of the development cycles of several new types of zones are considered below.

In computation of strategies, LG relies on a set of zones, serial or concurrent. At the end of the development cycle, new zones are universal abstract models of local skirmishes (strategy building blocks). At the beginning of the cycle, new zones are the existing zones with small differences deduced originally from the ways that human experts employ to solve adversarial problems from a certain class. These differences may include additional rules (and principles) that are specific to a given problem domain or, even, to a particular problem. The goal of the continued development of LG is to discover specific rules and principles applicable to given problems and attempt to convert them, initially, to generic principles, and later to the universal conceptual zones. The universal new zones become components of the theory of LG to be used across a variety of problem domains.

The general development cycle of new zones begins with conversations with subject matter experts (SME) who usually provide very specific rules that are applicable in a few specific situations. Such rules can be inserted directly into the software to influence generation of the existing zones in very specific cases. Once such rules are better understood they are translated into generic principles of zone generation that are applicable to an entire problem domain rather than to a few specific instances. This means that specific rules are generalized into the domain-level principles for generation of new conceptual zones. At this point, however, it may still be unclear how such principles can be transferred to different problem domains. Further research, interaction with SME from different fields, and work on projects related to different problem domains allow such mid-level principles to be further generalized into conceptual universal zone that can be utilized in multiple problem domains or across the entire domain spectrum. This generalization leads to new concepts of ABG construction such that these types of zones become applicable to any game created employing those concepts. New concepts and universal zones become components of the theory of LG.

In our experience, the new zones developed for one problem domain usually may be applied within a different domain – of course, with potentially unexpected results requiring further research. The final step of abstraction from rules to principles to concepts of ABG construction includes development of tools that allow the user to control how those concepts should be applied based on different needs.

It is usually desirable for LG system to explain its behavior employing the types of zones utilized (and related concepts). While in many situations it is possible, however, in certain cases it may be hard to pinpoint, which type of zone and which concept contributed to a specific behavior within the strategy. In contrast with popular rule-based systems where a specific rule (or a chain of rules) is responsible for a behavior, LG-generated behavior may be caused by a combined interplay of the LG zones produced by the LG strategy generator. In this case, the entire strategy is produced as an interconnected whole. In LG, the original rules are employed only at prototyping stages to test potential principles of new LG zones.

4.1 Zones with Paired/Prerequisite Trajectories

This concept was originally discovered in 2002 during the LG-PROTECTOR project [11, 20] – Defense of Kuwait against cruise missiles employing airborne interceptors and ground based fire units as integrated air defense assets. The specific requirement of the project was integrated fire control – any missile fire from air or ground asset could be guided to the target by any ground or airborne fire control radar (not only by the radar of the asset that fired this missile). A precondition for launch of any missile was that a fire control radar should be illuminating the target at the time of launch. Further, the target had to remain illuminated by a fire control radar (the original one or any other radar within a designated set of radars) for every instance during the entire flight time of the missile.

Initially, the zone generation was altered with a specific rule: "Do not allow an intercepting missile to follow its computed trajectory if no fire control illumination is present". Two types of trajectories were computed: the usual missile intercept trajectory and sensor "fire control" trajectories that would guide a sensor platform to a position to illuminate the target. Only fire control trajectories were allowed to be followed initially, while the missile trajectories were continuously recomputed (as part of a standard zone recomputation) until launch was allowed.

As soon as this requirement was understood by prototyping using the rule above, it was generalized into a principle of the so-called "kill chain" or sensor-weapon pairing. This was done at the core level of LG zone generation. Specifically, the timing horizon for potential missile trajectories, i.e., an upper time limit, was established based on one or more fire control trajectories. Furthermore, a link was maintained that paired missile trajectories to corresponding prerequisite sensor trajectories. Then the rule became subsumed by the zone generation principle as no trajectory would be generated that did not follow the rule.

However, we soon realized that this could be taken a step further. The principle of sensor-weapon pairing was expanded into the abstract concept of a *zone with paired/prerequisite trajectories* and control over employment of such concept was made available at the game definition level, specifically, in Game Development Kit (GDK [11]). When constructing a wargame, the user is now able to define, which piece types require and which piece types can provide such prerequisite trajectories. Zones with paired/prerequisite trajectories may be automatically generated by LG whenever a certain trajectory or action requires a prerequisite trajectory or action. For example, for operations in urban terrain, new type of zones permits establishing a suppressive fire or "bound overwatch" trajectory by one entity before another entity is allowed to perform the main desired action, e.g., cross a street or enter a building. Another example is requiring a reconnaissance team – air, land, or sea – to find the target before an assault team can be sent there. Yet another example is the use of kinetic ballistic interceptor missiles in conjunction with X-band tracking systems for ballistic missile defense.

Other applications are sure to be discovered during future projects. As such applications are found, GDK will be modified to make it easier for users to specify use of this concept in their particular domains. At some point, the zone itself may be further generalized – perhaps into a longer chain of connected trajectories.

4.2 Zones with Restricted Areas

This concept was originally discovered as a specific requirement to restrict flight of certain aircraft to certain altitudes and corridors. The prototype implementation employed methods to specifically designate desired cells as impassible by the specific types of pieces. The trajectories were, therefore, generated around such impassible areas.

However, after this requirement and its implication were fully understood, this rule was converted to a generalized principle of a *zone with restricted areas*. This principle allowed dynamic designation of impassible areas based on the battle-space properties. For example, areas covered by SAM (surface-to-air missile) sites could be automatically designated as no-go areas. The generation of trajectories and zones was further integrated with this principle to allow more flexible solutions, such as removing flight restrictions if the zone analysis indicated that no winning strategy could be produced with current restrictions. Furthermore, different pieces or groups of pieces could have individually defined restricted areas.

Currently, this principle has been further generalized into a *concept of allowed and disallowed zones which can be dynamically computed* based on the zones themselves. Multiple levels and conditions can be specified. This concept has been employed in the DARPA RAID program to prevent fire-teams within a platoon from spreading far apart from another – they are only allowed to move within a certain distance from the overall main path of the platoon. Similarly, this concept is employed to restrict the movement of all 3 platoons to remain within the company movement boundaries specified in the mission statement. Furthermore, GDK includes means to provide the user with the most flexible control over the use zones with restricted areas so they can be tuned to any desired domain.

4.3 Zones with Synchronized Interception Trajectories

While initial work on this concept was done in the past, the most significant work was accomplished recently, during the DARPA RAID project. The specific requirement is to synchronize movement of multiple entities to achieve massing of fires on the enemy at all points as well as to keep the entire group together.

The first rule of this type was introduced in the past in LG-TCT (Time Critical Targeting) project [11], when the air strike groups were required to move as a single package and synchronize both counter-interception of enemy fighters and arrival at the final target with other groups. This was then generalized into a principle and, later, a concept of a group that operates together in a coordinated fashion. From then on, the missions were defined on per-group basis and cooperation between groups to be controlled by the user. This was integrated into LG zone generation and the way that missions are translated into the types of LG zones.

However, this was not sufficient to accomplish the required intricate level of entity synchronization in a MOUT situation. A new principle was needed. Due to previous experience, we went directly to principle level and implemented synchronization as a general principle of generating *zones with synchronized interception trajectories*. The new zones utilize timing constraints imposed on trajectories and target-to-threat as-

signments to regulate how the units follow their computed trajectories to achieve mutual support and massing of fires. This was used as a key part of the computation of synchronized movements of fire teams within platoons and between platoons.

This principle can be fairly easily transformed into a concept by providing the user with appropriate tools to control its employment. For instance, the user working in a different domain such as "Time Critical cruise missile and bomber attack of ground targets" may wish to have movement synchronized only up to a certain user-defined point. This can be accomplished by the user specifying desired synchronization constraints (which will automatically be translated into the zone generation procedures) as part of the mission definition or rules of engagement.

5 Scalability of the LG Approach

The major difficulty of employing wargaming tools is related to the issue of *scalability* (Sections 1, 2). It means that even *modest* increases in problem complexity, such as adding several tanks, aircraft or platoons, could cause *exponential* increase in computation time to generate plans or make decisions. This is called combinatorial explosion. This is a common problem of all the tools utilizing "look-ahead", that is, an ability to make plans or decisions to achieve certain goals in the future. The problem is considerably aggravated by the fact that the real world military domains are immensely (not modestly) larger than those upon which the majority of the look-ahead tools are being tested.

As a consequence, a number of non-LG wargaming tools that are currently used for planning and control of military operations, do not employ look-ahead but provide only a display of the conflict environment. For such tools, the planning of possible courses of actions is either totally scripted or performed by the human experts.

The LG approach overcomes the combinatorial explosion on two levels.

- The first level is theoretical. There is a mathematical proof that LG approach has a low degree polynomial run time [16]. In contrast, for majority of other approaches the complexity is exponential. It means that, unlike LG, the combinatorial explosion is inherent to such approaches and cannot be avoided within the approach itself. As a consequence, many such approaches either employ ad-hoc forward pruning to keep the computations within the required time limits (resulting in generating ad-hoc plans), or employ alternative technologies such as rule-based systems and/or neural networks, which have their own disadvantages. The essence of the contrast between the LG and non-LG approaches to the gaming problems is that LG changes the whole paradigm from search to construction:
  - The *dynamic hierarchical decomposition* within a hypergame and component games is one of the main principles of LG leading to reduction of dimensionality. For example, with LG large-scale military operations are decomposed, via the hypergame approach, into a hierarchy of smaller, homogeneous ABG of various resolution and time scales. Moreover, a hypergame with its hyperlinks between the component games permits us to avoid a Cartesian product representation. Such a product could be thought as a gigantic

game where every move, is a vector including the individual moves from every component ABG. Although convenient mathematically due to simplicity of its definition, such a gigantic product could be an un-scalable obstacle to implementing true concurrency.

- The geometrical *relations of reachability* on the abstract board [16] permit to encode the game description in a highly efficient way since most of the game rules (representing movement, shooting and sensing) are formalized via such relations. Roughly, location *y* of an abstract board is reachable from location *x* for a piece p if p can move from *x* to *y* in one time interval. Relations of reachability permit to efficiently generate *trajectories*, as sequences of steps along the optional planning paths.
- The geometrical *relations of connectivity* on the abstract board [16] permit to define a hierarchy of constructs used to generate desirable strategies. Connectivity of two trajectories roughly means that the end point of one trajectory is an intermediate point of another trajectory (a trajectory of q_3 is connected to trajectory of p_O via point 3, Fig. 1). Various constructs include trajectories, zones, and complex zones, where each subsequent construct is defined as a collection of objects of the preceding construct, linked to each other by certain relation of connectivity.
- The dynamic *hierarchy of formal languages* [16] effectively "redefines" the LG game in a way that every game move represents a *translation* from one hierarchy of languages to another. The hierarchy with translations provides an efficient representation of the hierarchy of constructs, which permits to *translate* (i.e., slightly update and reuse) this hierarchy instead of regenerating it from scratch when moving from state to state during strategy generation.
- Essentially, the hierarchy of languages permits to "project" the game space-time (the game tree) onto the space (on the abstract board), *construct a solution* within the board without searching through the space-time and elevate the solution back into the space-time. For many classes of problems including a variety of defense systems, LG constructs are sufficient to solve the game by constructing advantageous strategies *without employing the search tree*. The rest of the problems are usually those that require highly precise solution; the game of chess is one of such problems. For these problems, construction may lead to a *tiny search tree* in the order of a hundred moves [16];

- The second level is *experimental*. Software implementations could be inefficient, leading to exponential run times despite the theoretical results. Thus the claim of scalability for the LG based software systems must be confirmed experimentally. There have been several lengthy experimental feasibility studies conducted jointly by AFRL, Boeing, Rockwell, and STILMAN in 2000-03 that included hundreds of experiments [20]. These studies concluded that the LG based software tools of mission planning and execution, resource allocation, COA generation and assessment have polynomial run time while several of those tools demonstrated even better, linear, run time growth. Multiple experiments with various

other LG-based systems, including series of comprehensive DARPA RAID experiments in 2005-07, demonstrated exceptional scalability [11, 12, 18, 20].

References

1. Botvinnik, M.M., Computers in Chess: Solving Inexact Search Problems, Springer-Verlag, 1984.
2. Brewer, G.D., Shubik, M., The War Game: A Critique of Military Problem Solving, Harvard University Press, 1979.
3. DARPA RAID program, 2004-07, http://dtsn.darpa.mil/ixo/programs.asp?id=43
4. Dunnigan, J., The Complete Wargames Handbook, William Morrow & Co., 1992.
5. Garcia-Ortiz, A. et al. Application of Semantic Control to a Class of Pursue-Evader Problems, *Computers and Mathematics with Applications*, 1993, 26(5): 97-124.
6. Isaacs, R. *Differential Games*, Wiley, New York, NY, 1965.
7. Jones, R.M. Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. Automated Intelligent Pilots for Combat Flight Simulation, *AI Magazine*, Spring 1999.
8. Knuth, D., Moore, R., An Analysis of Alpha-Beta Pruning, *Artificial Intelligence*, 293-326, 6(4), 1975.
9. Kuhn, H. W., Tucker, A. W., eds. *Contributions to the Theory of Games*, Volume II (Annals of Mathematics Studies, 28), Princeton University Press, 1953.
10. Lirov Y., Rodin, E.Y., McElhaney, B.G., and Wilbur, L.W. Artificial Intelligence Modeling of Control Systems, *Simulation*, 1988, 50(1): 12-24.
11. *Linguistic Geometry Tools: LG-PACKAGE*, with Demo DVD, 50 pp., STILMAN, 2007. This brochure and recorded demos are also available at www.stilman-strategies.com
12. *Linguistic Geometry Workshop, with STILMAN's Comments*, 17 pp., REPORT, Dstl, Ministry of Defence, Farnborough, UK, Feb. 25-26, 2003.
13. Lirov, Y., Rodin, E.Y., McElhaney, B.G., Wilbur, L.W. Artificial Intelligence Modeling of Control Systems, *Simulation*, pp.12-24, 1, 1988.
14. Osborn, M., Rubinstein, A., A Course in Game Theory, MIT Press, Cambridge, MA, 1994.
15. Pugh, D., Captain, USAF, "A Validation Assessment of the STORM Air-to-Air Prototype Algorithm", AFIT.
16. Stilman, B., *Linguistic Geometry: From Search to Construction*. Kluwer Acad. Publishers (now Springer-Verlag), 416 pp., 2000.
17. Stilman, B., Linguistic Geometry for Symmetric and Asymmetric War Games, pp. 431-449, *Proc. of the 2nd Int. Conference on Computers and Games* – CG'2000, Oct. 26-28, 2000, Hamamatsu, Japan.
18. Stilman, B., Yakhnis, V., Umanskiy, O., Chapter 3.3. Strategies in Large Scale Problems, in: *Adversarial Reasoning: Computational Approaches to Reading the Opponent's Mind*, Ed. by A. Kott and W. McEneaney, Chapman & Hall/CRC, pp. 251-285, 2007.
19. Stilman, B., Yakhnis, V., Umanskiy, O., Knowledge Acquisition and Strategy Generation with LG Wargaming Tools, *Int. J. of Comp. Intelligence and Appl.*, pp. 385-410, Vol. 2, No. 4, 2002.
20. Stilman, B., Yakhnis, V., Umanskiy, O., Hearing, J., Operational Level Decision Aids with LG-based Tools, pp. 403-414, *Proc. of the SPIE Conference "Enabling Technology for Simulation Science VI"*, April 1-5, 2002, Orlando, FL, **USA**.
21. Von Neumann, J., Morgenstern, O. Theory of Games and Economic Behavior, Princeton Univ. Press, 1944.
22. Williams, J.D., The Complete Strategist, Dover Publications, 1986.