NETWORK LANGUAGES FOR INTELLIGENT CONTROL

D R A F T[*]

Boris Stilman
Department of Computer Science & Engineering, University of Colorado at Denver,
Campus Box 109, Denver, CO 80217-3364, USA. Email: bstilman@cse.cudenver.edu

Abstract - In order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex control systems, and apply them to different systems, we develop a formal theory, the Linguistic Geometry. This research includes the development of syntactic tools for *knowledge representation* and *reasoning* about large-scale hierarchical complex systems. It relies on the formalization of *search heuristics* of high-skilled human experts, which allow to decompose complex system into the hierarchy of subsystems, and thus solve intractable problems reducing the search. The hierarchy of subsystems is represented as a hierarchy of formal attribute languages. This paper includes a brief survey of the Linguistic Geometry and a detailed comparative description of two comprehensive examples of solving optimization problems for military autonomous agents with cooperative and opposing interests operating on surface and in space. These examples include actual generation of the hierarchy of languages and demonstrate the *drastic reduction of search* in comparison with conventional search algorithms.

1. INTRODUCTION

There are many real-world problems where human expert skills in reasoning about complex systems are incomparably higher than the level of modern computing systems. At the same time there are even more areas where advances are required but human problem-solving skills can not be directly applied. For example, there are problems of planning and automatic control of autonomous agents such as space vehicles, stations and robots with cooperative and opposing interests functioning in a complex, hazardous environment. Reasoning about such complex systems should be done automatically, in a timely manner, and often in a real time. Moreover, there are no highly-skilled human experts in these fields ready to substitute for robots (on a virtual model) or transfer their knowledge to them. There is no grand-master in robot control, although, of course, the knowledge of existing experts in this field should not be neglected – it is even more valuable. It is very important to study human expert reasoning about similar complex systems in the areas where the results are successful, in order to discover the keys to success, and then apply and adopt these keys to the new, as yet, unsolved problems. The question then is what language tools do we have for the adequate representation of human expert skills? An application of such language to the area of successful results achieved by the human expert should yield a *formal, domain independent knowledge* ready to be transferred to different areas. Neither natural nor programming languages satisfy our goal. The first are informal and ambiguous, while the second are usually detailed, lower-level tools. Actually, we have to learn how we can formally represent, generate, and investigate a *mathematical model* based on the *abstract images* extracted from the expert vision of the problem.

There have been many attempts to find the optimal (suboptimal) operation for real-world complex systems. One of the basic ideas is to decrease the dimension of the real-world system following the approach of a *human expert in a certain field*, by breaking the system into smaller subsystems. These ideas have been implemented for many problems with varying degrees of success [1, 2, 15]. Implementations based on the formal theories of linear and nonlinear planning meet hard efficiency problems [4, 12, 17, 22, 25]. An efficient planner requires an intensive use of heuristic knowledge. On the other hand, a pure heuristic implementation is unique. There is no general constructive approach to such implementations. Each new problem must be carefully studied and previous experience usually can not be applied. Basically, we can not answer the

[*] The final version was published as Stilman, B., Network Languages for Intelligent Control, *An International Journal: Computers & Mathematics with Applications*, Vol. 31, No. 3,pp. 91-118, 1996.

question: what are the formal properties of human heuristics which drove us to a successful hierarchy of subsystems for a given problem and how can we apply the same ideas in a different problem domain?

In the 1960's a formal syntactic approach to the investigation of properties of natural language resulted in the fast development of a theory of formal languages by Chomsky [5], Ginsburg [10], and others. This development provided an interesting opportunity for dissemination of this approach to different areas. In particular, there came an idea of analogous linguistic representation of images. This idea was successfully developed into syntactic methods of pattern recognition by Fu [8], Narasimhan [16], and Pavlidis [18], and picture description languages by Shaw [23], Feder [6], Rosenfeld [20].

Searching for the adequate mathematical tools formalizing human heuristics of dynamic hierarchy, we have transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems [27]. However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. The origin of such languages can be traced back to the research on programmed attribute grammars by Knuth [11], Rozenkrantz [21], Volchenkov [39].

A mathematical environment (a "glue") for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Nilsson [17], Fikes [7], Sacerdoti [22], McCarthy, Hayes [13, 14], and others based on first order predicate calculus.

To show the power of the linguistic approach it is important that the chosen model of the heuristic hierarchical system be sufficiently complex, poorly formalized, and have successful applications in different areas. Such a model was developed by Botvinnik, Stilman, and others, and successfully applied to scheduling, planning, and computer chess [2].

In order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex control systems, we develop a formal theory, the so-called *Linguistic Geometry* [28-38]. This research includes the development of syntactic tools for *knowledge representation* and *reasoning* about large-scale hierarchical complex systems. It relies on the formalization of *search heuristics*, which allow one to decompose complex system into a hierarchy of subsystems, and thus solve intractable problems, reducing the search. These *hierarchical images* were extracted from the expert vision of the problem. The hierarchy of subsystems is represented as a *hierarchy of formal attribute languages* [28, 32-36]. In this paper after a brief survey of Linguistic Geometry (Sections 2-5), we will consider in comparison two comprehensive examples of the robotic optimization problems for 2-D and 3-D operational districts. Originally, the 2-D example was presented at the 22nd Annual ACM Computer Science Conference in Phoenix, 1994 [37], while the 3-D example – at the 1994 Goddard Conference on Space Applications of Artificial Intelligence at Goddard Space Flight Center, Greenbelt, MD [38].

2. COMPLEX SYSTEMS

A **Complex System** is the following eight-tuple:

$$< X, P, R_p, \{ON\}, v, S_i, S_t, TR>,$$

where

$X=\{x_i\}$ is a finite set of *points*;

$P=\{p_i\}$ is a finite set of *elements*; P is a union of two non-intersecting subsets P_1 and P_2;

$R_p(x, y)$ is a set of binary relations of *reachability* in X (x and y are from X, p from P);

$ON(p)=x$, where ON is a partial function of *placement* from P into X;

v is a function on P with positive integer values; it describes the *values* of elements. The Complex System searches the state space, which should have initial and target states;

S_i and S_t are the descriptions of the *initial* and *target* states in the language of the first order predicate calculus, which matches with each relation a certain Well-Formed Formula (WFF). Thus, each state from S_i or S_t is described by a certain set of WFF of the form $\{ON(p_j) = x_k\}$;

TR is a set of operators, TRANSITION(p, x, y), of transition of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed and added to the description of the state), and of WFF of applicability of the transition. Here,

Remove list: ON(p)=x, ON(q)=y;

Add list: ON(p)=y;

Applicability list: $(ON(p)=x)\wedge R_p(x, y)$,

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out in turn with participation of elements p from P_1 and P_2 respectively; omission of a turn is permitted.

According to definition of the set P, the elements of the System are divided into two subsets P_1 and P_2. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x, y)$ holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from point x to point y. The element q from point y must be withdrawn (eliminated) if p and q belong to the different subsets P_1 and P_2.

The problem of the optimal operation of the System is considered as a search for the optimal sequence of transitions leading from one of the initial states of S_i to a target state S of S_t.

It is easy to show formally that robotic system can be considered as the Complex System (see below). Many different technical and human society systems (including military battlefield systems, systems of economic competition, positional games) which can be represented as twin-sets of movable units (of two or more opposing sides) and their locations, thus, can be considered as Complex Systems.

With such a problem statement for the search of the optimal sequence of transitions leading to the target state, we could use formal methods like those in the problem-solving system STRIPS [7], nonlinear planner NOAH [22], or in subsequent planning systems. However, the search would have to be made in a space of a huge dimension (for nontrivial examples). Thus, in practice no solution would be obtained.

We devote ourselves to the search for an approximate solution of a reformulated problem.

3. DISTANCE MEASUREMENT

To create and study a hierarchy of dynamic subsystems we have to investigate geometrical properties of the Complex System.

A **map of the set X** relative to the point x and element p for the Complex System is the mapping: $\mathbf{MAP_{x,p}}: X \longrightarrow \mathbf{Z_+}$, (where x is from X, p is from P), which is constructed as follows. We consider a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets of X $\{M^k_{x,p}\}$ (Figure 1)**:**

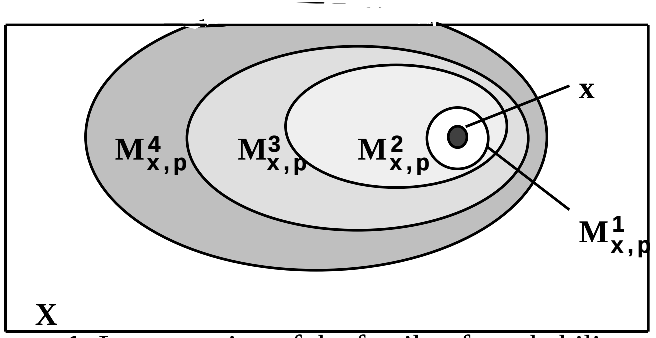

Figure 1. Interpretation of the family of reachability areas

k=1: $M^k_{x,p}$ is a set of points *m reachable in one step* from x: $R_p(x,m)=T$;

k>1: $M^k_{x,p}$ is a set of points *reachable in k steps and not reachable in k-*1 *steps,* i.e., points *m* reachable from points of $M^{k-1}_{x,p}$ and not included in any $M^i_{x,p}$ with numbers *i* less than *k*.

Let $MAP_{x,p}(y)=k$, for y from $M^k_{x,p}$ (*number of steps from x to y*). In the remainder points let $MAP_{x,p}(y)=2n$, if $y \neq x$ (n is the number of points in X)**;** $MAP_{x,p}(y)=0$, if y = x.

It is easy to verify that the map of the set X for the specified element p from P defines an *asymmetric distance function* on X:

1. $MAP_{x,p}(y) > 0$ for $x \neq y$; $MAP_{x,p}(x)=0$;

2. $MAP_{x,p}(y)+MAP_{y,p}(z) \geq MAP_{x,p}(z)$.

If R_p is a symmetric relation,

3. $MAP_{x,p}(y)=MAP_{y,p}(x)$.

In this case each of the elements p from P specifies on X its *own* ***metric.*** Various examples of measurement of distances for robotic vehicles are considered later.

4. LANGUAGE OF TRAJECTORIES

This language is a formal description of the set of lowest-level subsystems, the set of different paths between points of the Complex System. An element might follow a path to achieve the goal "connected with the ending point" of this path.

A ***trajectory*** for an element p of P with the beginning at x of X and the end at y of X ($x \neq y$) with length l is the following string of symbols with parameters, points of X: $t_0=\boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l)$, where $x_l = y$, each successive point x_{i+1} is reachable from the previous point x_i, i.e., $R_p(x_i, x_{i+1})$ holds for $i = 0, 1,\ldots, l-1$; element p stands at the point x: ON(p)=x. We denote $t_p(x, y, l)$ the set of all trajectories for element p, beginning at x, end at y, and with length l. $P(t_0)=\{x, x_1, ..., x_l\}$ is the set of parameter values of the trajectory t_0. A ***shortest trajectory*** t of $t_p(x, y, l)$ is the trajectory of minimum length for the given beginning x, end y and element p.

Properties of the Complex System permit to define (in general form) and study formal grammars for generating the shortest trajectories. A general grammar (Table I) and its application to generating the shortest trajectory for a robotic vehicle will be presented later.

Reasoning informally, an analogy can be set up: the shortest trajectory is analogous with a straight line segment connecting two points in a plane. An analogy to a k-element segmented line connecting these points is called an ***admissible trajectory of degree k***, i.e., the trajectory which can be divided into k shortest trajectories. The admissible trajectories of degree 2 play a special role in many problems. As a rule, elements of the System should move along the shortest paths. In case of an obstacle, the element should move around this obstacle by tracing an intermediate point aside and going to and from this point to the end along the shortest trajectories. Thus, in this case, an element should move along an admissible trajectory of degree 2.

A ***Language of Trajectories*** $L_t^H(S)$ for the Complex System in a state S is the set of all the shortest and admissible (degree 2) trajectories of the length less than H. Different properties of this language and generating grammars were investigated in [32].

5. LANGUAGES OF TRAJECTORY NETWORKS.

After defining the Language of Trajectories, we have new tools for the breakdown of our System into subsystems. According to the ideas presented in [2], these subsystems should be various types of trajectory networks, i.e., the sets of interconnected trajectories with one singled out trajectory called the *main trajectory*. An example of such network is shown in Figure 2.

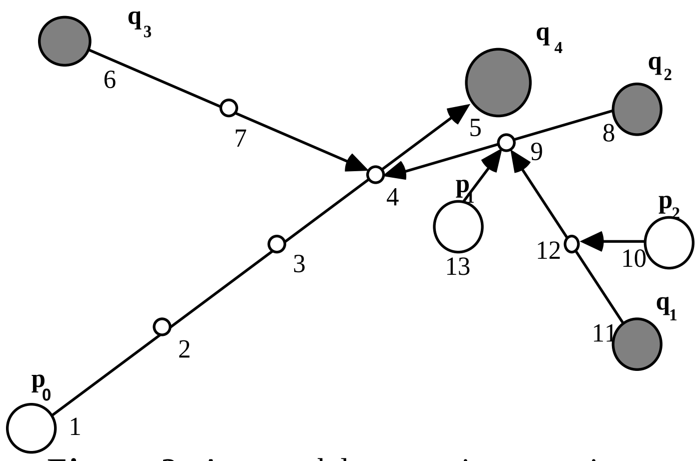

Figure 2. A network language interpretation.

The basic idea behind these networks is as follows. Element p_o should move along the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ to reach the ending point 5 and remove the target q_4 (an opposite element). Naturally, the opposite elements should try to disturb those motions by controlling the intermediate points of the main trajectory. They should come closer to these points (to the point 4 in Figure 2) and remove element p_o after its arrival (at point 4). For this purpose, elements q_3 or q_2 should move along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of element p_o at point 4. Similarly, element p_1 of the same side as p_o might try to disturb the motion of q_2 by controlling point 9 along the trajectory $\boldsymbol{a}(13)\boldsymbol{a}(9)$. It makes sense for the opposite side to include the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of element q_1 to prevent this control.

Similar networks are used for the breakdown of complex systems in different areas. Let us consider a linguistic formalization of such networks. The Language of Trajectories describes "one-dimensional" objects by joining symbols into a string employing reachability relation $R_p(x, y)$. To describe networks, i.e., “multi-dimensional" objects made up of trajectories, we use the relation of *trajectory connection.*

A ***trajectory connection*** of the trajectories t_1 and t_2 is the relation $C(t_1,t_2)$. It holds, if the ending link of the trajectory t_1 coincides with an intermediate link of the trajectory t_2; more precisely t_1 is connected with t_2, if among the parameter values $P(t_2)=\{y,y_1,\ldots,y_l\}$ of trajectory t_2 there is a value $y_i = x_k$, where $t_1=\boldsymbol{a}(x_o)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_k)$. If t_1 belongs to a set of trajectories with the common end-point, then the entire set is said to be connected with the trajectory t_2.

For example, in Figure 2 the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ are connected with the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ through point 4. Trajectories $\boldsymbol{a}(13)\boldsymbol{a}(9)$ and $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ are connected with $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$.

To formalize the trajectory networks we define and use routine operations on the set of trajectories: $C_A{}^k(t_1,t_2)$, a ***k-th degree of connection***, and $C_A{}^+(t_1,t_2)$, a ***transitive closure***.

Trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ in Figure 2 is connected degree 2 with trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$, i.e., $C^2(\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9),\ \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5))$ holds. Trajectory $\boldsymbol{a}(10)\boldsymbol{a}(12)$ in Figure 2 is in transitive closure to the trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ because $C^3(\boldsymbol{a}(10)\boldsymbol{a}(12),\ \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5))$ holds by means of the chain of trajectories $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$.

A ***trajectory network* W** relative to trajectory t_o is a finite set of trajectories $t_o,t_1,\ldots,t_k$ from the language $L_t{}^H(S)$ that possesses the following property: for every trajectory t_i from W (i = 1, 2,…,k) the relation $C_W{}^+(t_i,t_o)$ holds, i.e., each trajectory of the network W is connected with the trajectory t_o that was singled out by a subset of interconnected trajectories of this network. If the relation $C_W{}^{\mathbf{m}}(t_i, t_o)$ holds, trajectory t_i is called the ***m negation trajectory***.

Obviously, the trajectories in Figure 2 form a trajectory network relative to the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$. We are now ready to define network languages.

A ***family of trajectory network languages*** $\mathbf{L_C(S)}$ in a state S of the Complex System is the family of languages that contains strings of the form $t(t_1, \mathit{param})t(t_2, \mathit{param})\ldots t(t_m, \mathit{param})$, where *param* in parentheses substitute for the other parameters of a particular language. All the symbols of the string t_1, t_2,…, t_m correspond to trajectories that form a trajectory network W relative to t_1.

Different members of this family correspond to different types of trajectory network languages, which describe particular subsystems for solving search problems. One of such languages is the language that describes specific networks called Zones. They play the main role in the model considered here [2, 26, 33-36]. A formal definition of this language is essentially constructive and requires showing explicitly a method for generating this language, i.e., a certain formal grammar, which is presented in the full paper. In order to make our points transparent, here, we define the Language of Zones informally.

A ***Language of Zones*** is a trajectory network language with strings of the form $Z=t(p_o,t_o,\tau_o)\ t(p_1,t_1,\tau_1)\ldots t(p_k,t_k,\tau_k)$, where $t_o,t_1,\ldots,t_k$ are the trajectories of elements

$p_o, p_2, \ldots, p_k$ respectively; $\tau_o, \tau_1, \ldots, \tau_k$ are positive integer numbers (or 0) which "denote the time allocated for the motion along the trajectories in a correspondence to the mutual goal of this Zone: to remove the target element – for one side, and to protect it – for the opposite side. Trajectory $t(p_o, t_o, \tau_o)$ is called the *main trajectory* of the Zone. The element q standing on the ending point of the main trajectory is called the *target*. The elements p_o and q belong to the opposite sides.

To make it clearer let us show the Zone corresponding to the trajectory network in Figure 2.

$$Z = \boldsymbol{t}(p_o, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 4)\boldsymbol{t}(q_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 3)\boldsymbol{t}(q_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 3)\boldsymbol{t}(p_1, \boldsymbol{a}(13)\boldsymbol{a}(9), 1)$$
$$\boldsymbol{t}(q_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 2)\ \boldsymbol{t}(p_2, \boldsymbol{a}(10)\boldsymbol{a}(12), 1)$$

Assume that the goal of the white side is to remove target q_4, while the goal of the black side is to protect it. According to these goals element p_o starts the motion to the target, while blacks start in its turn to move their elements q_2 or q_3 to intercept element p_o. Actually, only those black trajectories are to be included into the Zone where the motion of the element makes sense, i. e., the *length of the trajectory is less than the amount of time (third parameter* τ) *allocated to it.* For example, the motion along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ makes sense, because they are of length 2 and time allocated equals 3: each of the elements has 3 time intervals to reach point 4 to intercept element p_o assuming one would go along the main trajectory without move omission. According to definition of Zone the trajectories of white elements (except p_o) could only be of the length 1, e.g., $\boldsymbol{a}(13)\boldsymbol{a}(9)$ or $\boldsymbol{a}(10)\boldsymbol{a}(12)$. As far as element p_1 can intercept motion of the element q_2 at the point 9, blacks include into the Zone the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of the element q_1, which has enough time for motion to prevent this interception. The total amount of time allocated to the whole bunch of black trajectories connected (directly or indirectly) with the given point of main trajectory is determined by the number of that point. For example, for the point 4 it equals 3 time intervals.

A language $L_Z(S)$ generated by the certain grammar $\mathbf{G_Z}$ [33-35] in a state S of a Complex System is called the ***Language of Zones***.

Network languages allow us to describe the "statics", i.e., the states of the System. We need a description of the "dynamics" of the System, i.e., the transitions from one state to another. The transitions describe the change of the descriptions of states as the change of sets of WFF. After each transition a new hierarchy of languages should be generated. Of course, it is an inefficient procedure. To improve an efficiency of applications in a process of the search it is important to describe the change of the hierarchy of languages. A study of this change should help us in modifying the hierarchy instead of regenerating it in each state. The change may be described as a hierarchy of mappings – translations of languages. Each language should be transformed by the specific mapping called a *translation.* Translations of Languages of Trajectories and Zones are considered in [34].

6. ROBOT CONTROL MODEL AS COMPLEX SYSTEM.

Such model can be represented as a Complex System naturally (Figure 3).

A set of X represents the operational district which could be the area of combat operation broken into n areas, e.g., squares. It could be a space operation, where X represents the set of different orbits, or a navy battlefield, etc. P is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests; $R_p(x,y)$ represent moving capabilities of different robots: robot p can move from point x to point y if $R_p(x, y)$ holds. Some of the robots can crawl, the other can jump or ride, sail and fly, or even move from one orbit to another. Some of them move fast and can reach point y (from x) in "one step", i.e., $R_p(x, y)$ holds, others can do that in *k* steps only, and many of them can not reach this point at all. $ON(p)=x$, if robot p is at the point x; $v(p)$ is the value of robot p. This value might be determined by the technical parameters of the robot. It might include the immediate value of this robot for the given combat operation; S_i is an arbitrary initial state of operation for analysis, or the starting state; S_t is the set of target states. These might be the states where robots of each side reached specified points. On the other hand S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_j) = x_k\}$ corresponds to the list of robots with their coordinates in each state.

TRANSITION(p, x, y) represents the move of the robot p from square x to square y; if a robot of the opposing side stands on y, a removal occurs, i.e., robot on y is destroyed and removed.

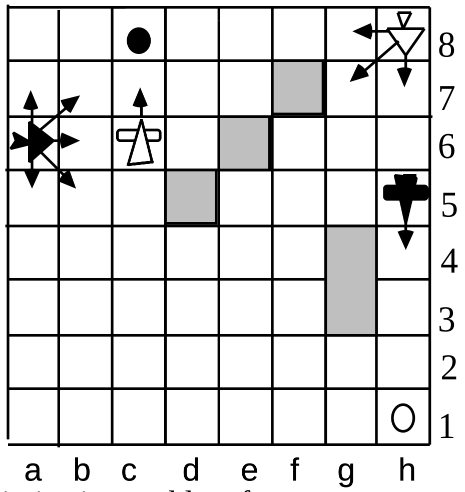

Figure 3. Optimization problem for autonomous robotic vehicles.

Robots with different moving capabilities are shown in Figure 3. The operational district X is the table 8 x 8. Squares g3, g4, d5, e6, f7 representing restricted area, e.g., neutral countries, are excluded. Robot W-FIGHTER (White Fighter) standing on h8, can move to any next square (shown by arrows). The other robot B-BOMBER from h5 can move only straight ahead, one square at a time, e.g., from h5 to h4, from h4 to h3, etc. Robot B-FIGHTER (Black Fighter) standing on a6, can move to any next square similarly to robot W-FIGHTER (shown by arrows). Robot W-BOMBER standing on c6 is analogous with the robot B-BOMER; it can move only straight ahead but in reverse direction. Thus, robot W-FIGHTER on h8 can reach any of the points y {h7, g7, g8} in on step, i.e., $R_{W\text{-}FIGHTER}$(h8, y) holds, while W-BOMBER can reach only c8 in one step.

Assume that robots W-FIGHTER and W-BOMBER belong to one side, while B-FIGHTER and B-BOMBER belong to the opposite side: W-FIGHTER P_1, W-BOMBER P_1, B-FIGHTER P_2, B-BOMBER P_2. Also assume that two more robots, W-TARGET and B-TARGET, (unmoving devices or targeted areas) stand on h1 and c8, respectively. W-TARGET belongs to P_1, while B-TARGET P_2. Each of the BOMBERs can destroy unmoving TARGET ahead of the course; it also has powerful weapons capable to destroy opposing FIGHTERs on the next diagonal squares ahead of the course. For example W-BOMBER from c6 can destroy opposing FIGHTERs on b7 and d7. Each of the FIGHTERs is capable to destroy an opposing BOMBER approaching its location, but it also capable to protect its friendly BOMBER approaching its prospective location. In the latter case the joint protective power of the combined weapons of the friendly BOMBER and FIGHTER can protect the BOMBER from interception. For example, W-FIGHTER located at d6 can protect W-BOMBER on c6 and c7.

The battlefield considered can be broken into two local operations. The first operation is as follows: robot B-BOMBER should reach point h1 to destroy the W-TARGET, while W-FIGHTER will try to intercept this motion. The second operation is similar: robot W-BOMBER should reach point c8 to destroy the B-TARGET, while B-FIGHTER will try to intercept this motion. After destroying the opposing TARGET the attacking side is considered as a winner of the local operation and the global battle. The only chance for the opposing side to avenge itself is to hit its TARGET on the next time interval and this way end the battle in a draw. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Figure 3.

Assume that due to the shortage of resources (which is typical in real combat operation) or some other reasons, each side can not participate in both operations simultaneously. It means that during the current time interval, in case of White turn, either W-BOMBER or W-FIGHTER can move. Analogous condition holds for Blacks. Of course, it does not mean that if one side began participating in one of the operations it must complete it. Any time on its turn each side can switch from one operation to another, e.g., transferring resources (fuel, weapons, human resources, etc.), and later switch back.

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-BOMBER from h5 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. The question is: is there a strategy for the White side to make a draw?

Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Experiments with the computer chess programs showed that for the similar problem (in chess terms - the R.Reti endgame) the search tree includes about a million moves (transitions). It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools.

7. ROBOT TRAJECTORIES ON THE SURFACE

In order to demonstrate generation of the Hierarchy of Languages for this problem, below we consider generation of the Language of Trajectories for the robotic system on example of generation of the shortest trajectory from f6 to point h1 for the robot W-FIGHTER (Figure 4). (This is the location of W-FIGHTER in one of the states of the System in the process of the search.)

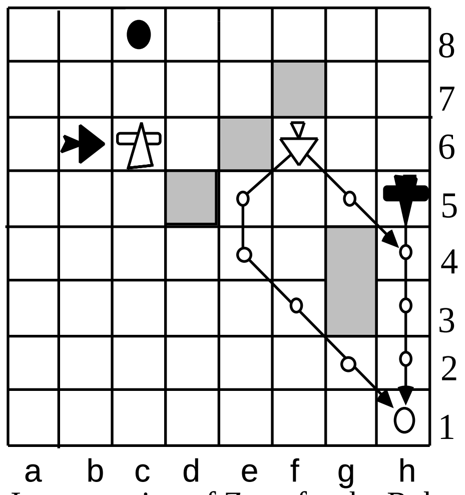

Figure 4. Interpretation of Zone for the Robotic System.

Consider the Grammar of shortest trajectories $\mathbf{G_t}^{(1)}$(Table I).
This is a controlled grammar [32]. Such grammars operate as follows. The initial permissible set of productions consists of the production with label 1. It should be applied first. Let us describe the application of a production in such grammar. Suppose that we attempt to apply production with label $\boldsymbol{l}$ to rewrite a symbol $\boldsymbol{A}$. We choose the leftmost entry of symbol $\boldsymbol{A}$ in the current string and compute the value of predicate $\boldsymbol{Q}$, the condition of applicability of the production. If the current string *does not* contain $\boldsymbol{A}$ or $\boldsymbol{Q}$ $=F$, then the application of the production is ended, and the next production is chosen from the failure section $\boldsymbol{F_F}$; $\boldsymbol{F_F}$ becomes the current permissible set. If the current string *does* contain the symbol $\boldsymbol{A}$ and $\boldsymbol{Q}$=T , $\boldsymbol{A}$ is replaced by the string in the right side of the production; we carry out the computation of the values of all formulas either standing separately (section $\pi_\mathbf{n}$) or corresponding to the parameters of the symbols ($\pi_\mathbf{k}$), and the parameters assume new values thus computed. Then, application of the production is ended, and the next production is chosen from the success section $\boldsymbol{F_T}$, which is now the current permissible set. If the applicable section is empty, the derivation halts.

The controlled grammar shown in Table I can be used for generation of shortest trajectories for robots with arbitrary moving capabilities.

Table I. A grammar of shortest trajectories $G_t^{(1)}$

L	Q	**Kernel, π_k**	F_T	F_F
1	Q_1	$S(x,y,l) \to A(x, y, l)$	two	ø
2_i	Q_2	$A(x,y,l) \to a(x)A(next_i(x,l),y,f(l))$	two	3
3	Q_3	$A(x, y, l) \to a(y)$	ø	ø

V_T ={a} is the alphabet of terminal symbols,
V_N ={S, A} is the alphabet of nonterminal symbols,
V_{PR} = *Truth Pred Con Var Func* {symbols of logical operations} is the alphabet of the first order predicate calculus ***PR***,

Truth ={*T*, *F*}
Pred ={Q_1,Q_2,Q_3} are predicate symbols:
$Q_1(x, y, l) = (MAP_{x,p}(y)=l)$ $(0<l<n)$
$Q_2(l) = (l \geq 1)$
$Q_3 = T$
Var = {x, y, *l*} are variables;
Con = {x_o,y_o,l_o,p} are constants;
Func = *Fcon* are functional symbols;
Fcon ={f,$next_1$,…,$next_n$} (n = |X|, number of points in X), $f(l)=l-1$, $D(f)=\mathbf{Z}_+-\{0\}$
($next_i$ is defined lower)

E =$\mathbf{Z}_+$ U X U P is the subject domain;
Parm: ***S*** –>*Var*, ***A*** –>*Var*, ***a*** –>{x}, is such a mapping that matches each symbol of the alphabet V_T UV_N a set of formal parameters;
L= {1,3} U two, two ={2_1,2_2,…,2_n} is a finite set called the set of labels; labels of different productions are different;
Q_i are the WFF of the predicate calculus ***PR***, the conditions of applicability of productions;
F_T is a subset of L of labels of the productions permitted on the next step derivation if *Q*=T; it is called a permissible set;
F_F is analogous to F_T but these productions are permitted in case of *Q*=F.

At the beginning of derivation: $x = x_o$, $y = y_o$, $l = l_o$, x_o X, y_o X, l_o $\mathbf{Z}_+$, p   P.
<u>***$next_i$*** is defined as follows:</u>

$D(next_i)$= X × $\mathbf{Z}_+$ × X^2 × $\mathbf{Z}_+$ × P (This is the domain of function *next*.)

SUM={v |v   X, $MAP_{xo,p}(v)+MAP_{yo,p}(v)=l_o$}
$ST_k(x)$={v | v from X, $MAP_{x,p}(v)=k$},
$MOVE_l(x)$ is an intersection of the following sets: $ST_1(x)$, $ST_{lo-l+1}(x_o)$ and SUM.

```
If
  $MOVE_l(x)=\{m_1, m_2, \ldots, m_r\} \neq \emptyset$
  then
  $next_i(x, l)=m_i$   for $i \leq r$ ;
  $next_i(x, l)=m_r$   for $r<i \leq n$,
  otherwise
  $next_i(x,l)=x$.
```

Values of $MAP_{F6,W\text{-}FIGHTER}$ are shown in Figure 5.

5	4	3	2	2	2	2	2
5	4	3	2	1	▨	1	2
5	4	3	2	▨	0	1	2
5	4	3	▨	1	1	1	2
5	4	3	2	2	2	▨	2
5	4	3	3	3	3	▨	3
5	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5

Figure 5. Values of $MAP_{f6,FIGHTER}$

8	7	7	7	6	7	7	7
7	7	6	6	6	▨	6	6
7	6	6	5	▨	5	5	5
7	6	5	▨	4	4	4	4
7	6	5	4	3	3	▨	3
7	6	5	4	3	2	▨	2
7	6	5	4	3	2	1	1
7	6	5	4	3	2	1	0

Figure 6. Values of $MAP_{h1,\ W\text{-}FIGHTER}$

Thus, the distance from f6 to h1 for W-FIGHTER is equal to 5. Applying the grammar $\mathbf{G_t^{(1)}}$ we have (symbol $\overset{l}{=>}$ means application of the production with the label *l*):

$$\boldsymbol{S}(f6, h1, 5) \overset{1}{=>} \boldsymbol{A}(f6, h1, 5) \overset{2_1}{=>} \boldsymbol{a}(f6)\boldsymbol{A}(next_1(f6, 5), h1, 5)$$

Thus we have to compute MOVE (see definition of the function $next_i$ from the grammar $\mathbf{G_t^{(1)}}$). First we have to determine the set of SUM, that is, we need to know values of $MAP_{f6,W\text{-}FIGHTER}$ and $MAP_{h1,W\text{-}FIGHTER}$ (shown in Figure 6) on X. Adding these tables (Figure 5 and Figure 6) as matrices we compute

SUM $=\{v \mid v \quad X,\ MAP_{f6,\ W\text{-}FIGHTER}(v) + MAP_{h1,W\text{-}FIGHTER}(v) = 5\}$ (Figure 7, 9).

					▨		
				▨	5		
			▨	5	5	5	
				5	5	▨	5
					5	▨	5
						5	5
							5

Figure 7. The points of X which belong to SUM.

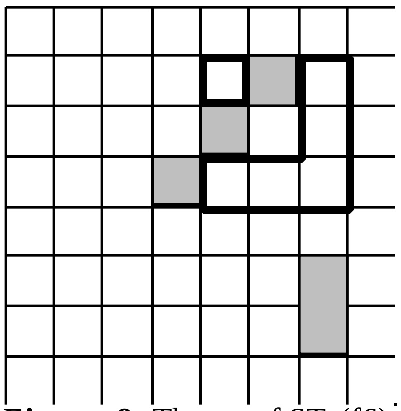

Figure 8. The set of ST_1(f6).

The next step is the computation of $ST_1(f6)=\{v \mid v$ from X, $MAP_{f6,W\text{-}FIGHTER}(v)=1\}$ which can be found in Figure 8. In order to complete computation of the set $MOVE_5(f6)$ we have to determine the following intersection: $ST_1(f6)$, $ST_{5-5+1}(f6)=ST_1(f6)$ and SUM. Consequently, $MOVE_5(f6)=\{e5, f5, g5\}$; and $next_1(f6, 5)=e5$, $next_2(f6, 5)=f5$, $next_3(f6, 5)=g5$. Since the number of different values of *next* is equal to 3 (here r=3, see definition of the function *next*, Table I) we could branch at this step, apply productions 2_1, 2_2 and 2_3 simultaneously, and continue both derivations independently. This could be accomplished in a parallel computing environment. Let us proceed with the first derivation.

$$\boldsymbol{a}(f6)\boldsymbol{A}(e5,h1,4) \overset{2_1}{=>} \boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{A}(next_1(e5,4), h1,3)$$

We have to compute $next_1(e5, 4)$ and, as on the preceding step, have to determine $MOVE_4(e5)$. To do this we have to compute

$$ST_1(e5)=\{v \mid v \quad X,\ MAP_{e5,W\text{-}FIGHTER}(v)=1\}\ \text{, (Figure 10)}$$

$$ST_{5-4+1}(f6) = ST_2(f6) = \{v \mid v \quad X,\ MAP_{f6,W\text{-}FIGHTER}(v)=2\}\text{, (Figure 11).}$$

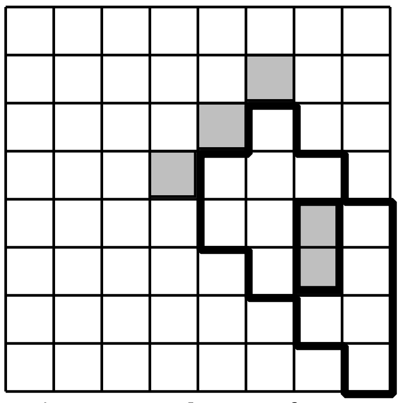
Figure 9. The set of SUM.

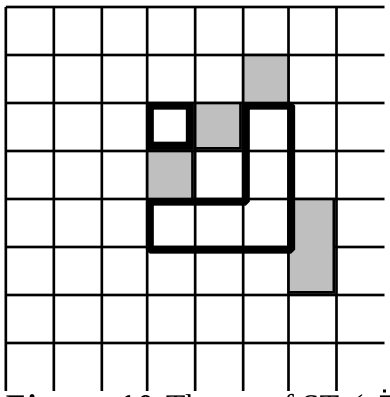
Figure 10. The set of ST_1(e5)

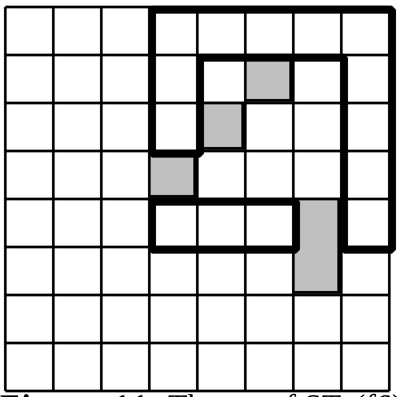
Figure 11. The set of ST_2(f6)

The set of SUM is the same on all steps of the derivation. Hence, $MOVE_4$(e5) is the intersection of the sets shown in Figure 9, 10, 11, $MOVE_4$(e5)= {e4, f4}; and $next_1$(e5,4) = e4; $next_2$(e5, 4) = f4. Thus, the number of different values of the function *next* is equal to 2 (r=2), so the number of continuations of derivation should be multiplied by 2.

Let us proceed with the first one:

$$\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{A}(e4, h1, 3) \ {}^{2}1 \Rightarrow \ldots$$

Eventually, we will generate one of the shortest trajectories for the robot W-FIGHTER from f6 to h1:

$$\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{a}(e4)\boldsymbol{a}(f3)\boldsymbol{a}(g2)\boldsymbol{a}(h1).$$

Similar generating techniques are used to generate higher level subsystems, the networks of paths, i.e., the Language of Zones. For example one of the Zones to be generated in the state shown in Figure 4 is as follows:

$$\boldsymbol{t}(\text{B-BOMBER},t_B,5)\boldsymbol{t}(\text{W-FIGHTER}, t_F,5)\boldsymbol{t}(\text{W-FIGHTER}, t_F{}^1, 2),$$

where

t_B=***a***(h5)***a***(h4)***a***(h3)***a***(h2)***a***(h1), t_F=***a***(f6)***a***(e5)***a***(e4)***a***(f3)***a***(g2)***a***(h1), $t_F{}^1$=***a***(f6)***a***(g5)***a***(h4)

The details of generation of different Zones are considered in [33-35].

8. SEARCH GENERATION FOR ROBOTIC SYSTEM

Consider how the hierarchy of languages works for the optimal control of the Robotic System introduced above (Figure 3). We generate the string of the Language of Translations [34] representing it as a conventional search tree (Figure 12) and comment on its generation.

In fact, this tree is very close to the search tree of the R.Reti endgame generated by program PIONEER in 1977 and presented at the World Computer Chess Championship (joint event with IFIP Congress 77, Toronto, Canada). Later it was published in different journals and books, in particular in [2].

In our comments of this generation we will emphasize the major steps avoiding some sophisticated details that will be considered further for space robotic system (Section 11).

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limit of five steps. It means that horizon H of the language L_Z(S) is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. Further, on example of space robotic vehicles we will consider reasons and an algorithm for picking the right value of the horizon. All the Zones generated in the start state within the horizon of 5 are shown in Figure 13. Zones for FIGHTERs as attacking elements are shown in the left diagram, while Zones for BOMBERs – in the right one. For example, one of the Zones for W-BOMBER, Z_{WB} is as follows:

Z_{WB}=***t***(P, ***a***(c6)***a***(c7)***a***(c8), 2)***t***(K, ***a***(a6)***a***(b7)***a***(c8), 3)***t***(K,***a***(a6)***a***(b7)***a***(c7), 2)***t***(P, ***a***(c6)***a***(b7), 1)

The second trajectory of B-FIGHTER ***a***(a6)***a***(b6)***a***(c7) leading to the square c7 is included into different Zone; for each Zone only one trajectory from each bundle of trajectories is taken.

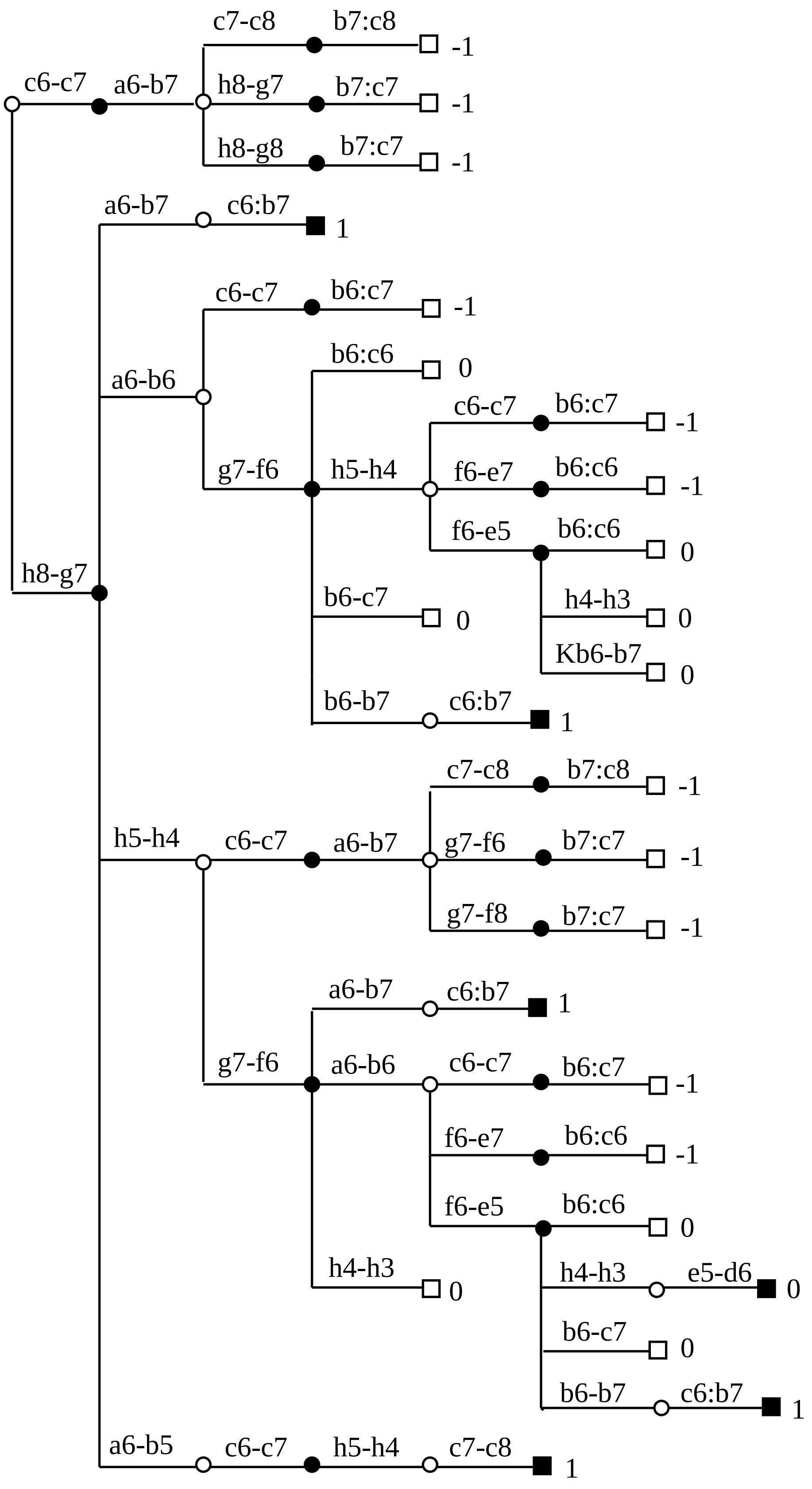

Figure 12. Search tree for the optimization problem for robotic vehicles.

Generation begins with the move 1. c6-c7 in the "white" Zone with the target of the highest value and the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations. The computation of move-ordering constraints is the most sophisticated procedure in this grammar. It takes into account different parameters of Zones, trajectories, and the so-called chains of trajectories.

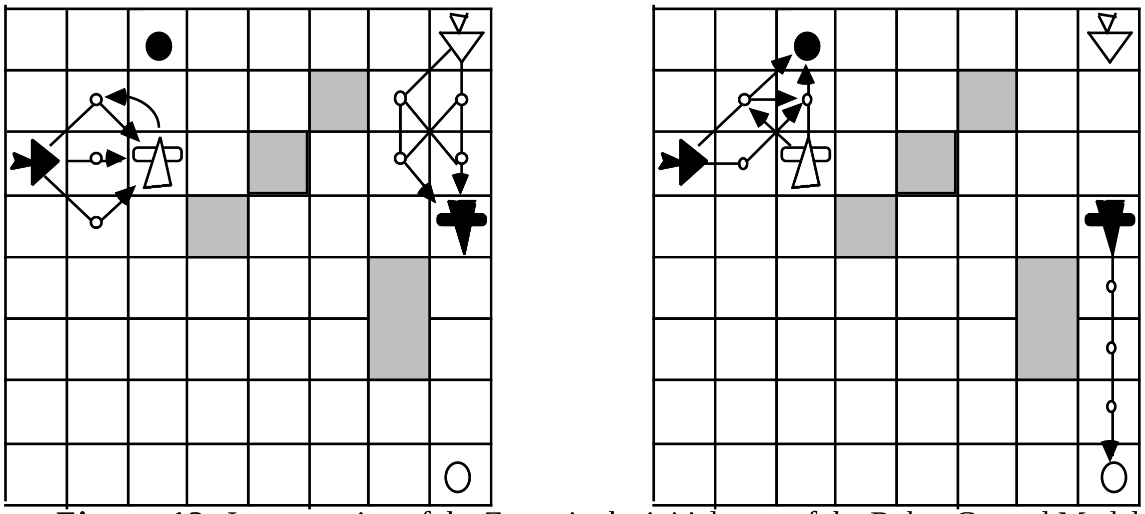
Figure 13. Interpretation of the Zones in the initial state of the Robot Control Model.

Next move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7:c8 (Figure 14, left). Symbol ":" means the removal of element. Here the grammar cuts this branch with the value of -1 (as a win of the Black side). This value is given by the special procedure of "generalized square rules" built into the grammar.

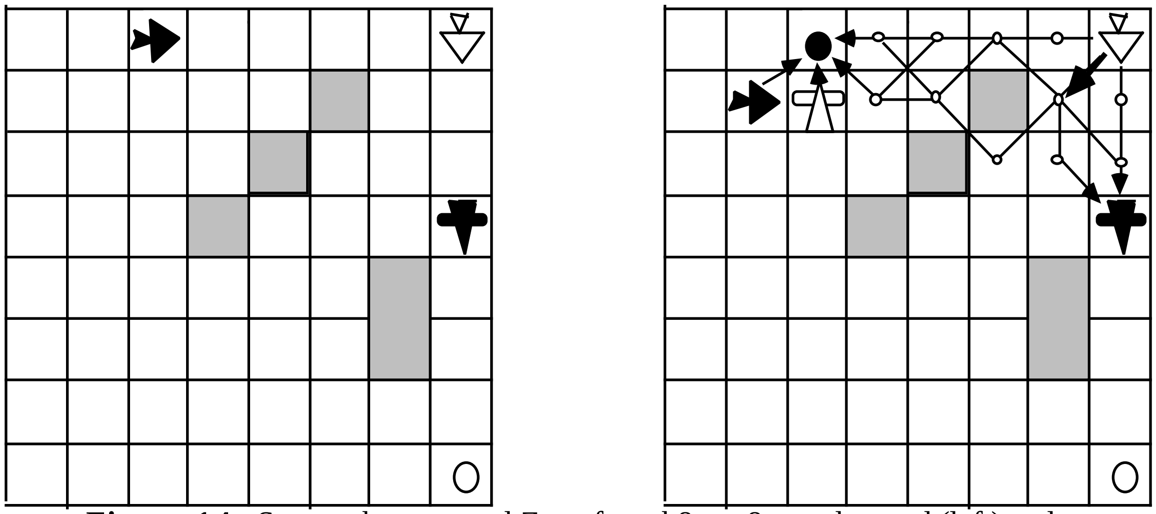
Figure 14. States where control Zone from h8 to c8 was detected (left) and where it was included into the search (right)

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search. After climb up to the move 1. ... a6-b7, the tree to be analyzed consists of one branch (of two plies): 2. c7-c8 b7:c8. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange on c8 (in favor of the White side). This can be achieved by participation of W-FIGHTER from h8, i.e., by generation and inclusion of the new so-called "control" Zone with the main trajectory from h8 to c8. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in Figure 14 (right). The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing W-BOMBER on h5. The motion along such trajectories allows to "gain the time", i.e., to approach two goals simultaneously.

The generation continues: 2. h8-g7 b7:c7. Again, the procedure of "square rules" cuts the branch, evaluates it as a win of the black side, and the grammar initiates the climb. Move 2. h8-g7 is changed for 2. h8-g8. Analogously to the previous case, the inspection procedure determined that the current minimax value (-1) can be improved by the improvement of the exchange on c7. Again, this can be achieved by the inclusion of Zone from h8 to c7. Of course, the best "time-gaining" move in this Zone is 2. h8-g7, but it was already included (as move in the Zone from h8 to c8). The only untested move in the Zone from h8 to c7 is 2. h8-g8. Obviously the grammar does not have knowledge that trajectories to c8 and c7 are "almost" the same.

After the next cut and climb, the inspection procedure does not find new Zones to improve the current minimax value, and the climb continues up to the start state. The analysis of the subtree shows that inclusion of Zone from h8 to c8 in the start state can be useful: the minimax value can be improved. Similarly, the most promising "time-gaining" move is 1. h8-g7. The Black side responded 1. ... a6-b7 along the first negation trajectories ***a***(a6)***a***(b6)***a***(c7) and ***a***(a6)***a***(b6)***a***(c8) (Figure 13 (right)). Obviously, 2. c6:b7, and the branch is terminated. The grammar initiates the climb and move 1. ... a6-b7 is changed for 1. ... a6-b6 along the trajectory ***a***(a6)***a***(b6)***a***(c7). Note, that grammar "knows" that in this state trajectory ***a***(a6)***a***(b6)***a***(c7) is active, i.e., B-FIGHTER has enough time for interception. The following moves are in the same Zone of W-BOMBER: 2. c6-c7 b6:c7. This state is shown in Figure 15(left). The "square rule procedure" cuts this branch and evaluates it as a win of the Black side.

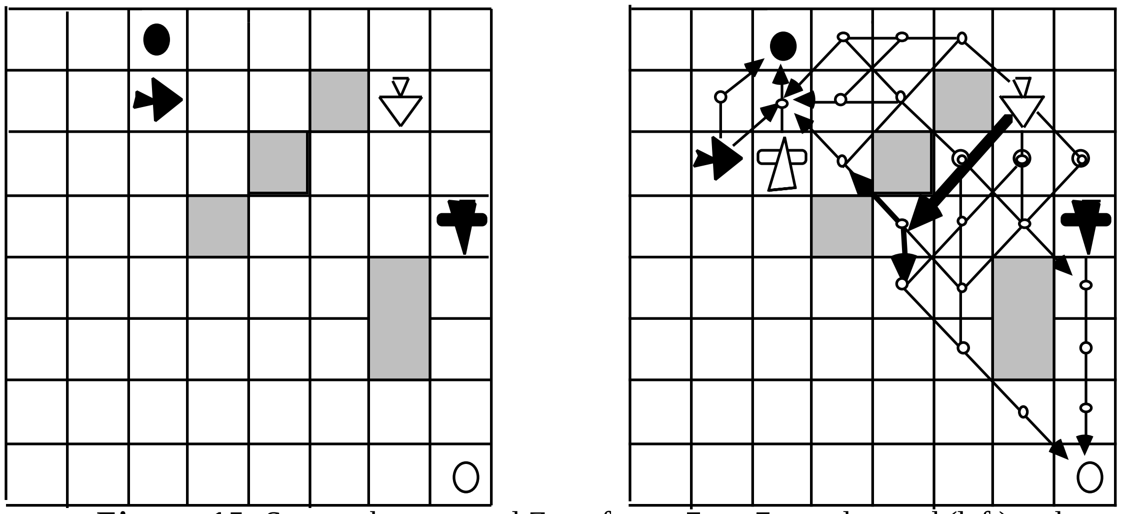
Figure 15. States where control Zone from g7 to c7 was detected (left) and where it was included into the search (right).

New climb up to the move 2. ... a6-b6 and execution of the inspection procedure result in the inclusion of the new control Zone from g7 to c7 in order to improve the exchange on c7. The set of Zones with different main trajectories from g7 to c7 is shown in Figure 15 (right). Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same Figure 15. These are "potential" first negation trajectories. It means that beginning with the second symbol ***a***(f6), ***a***(g6) or ***a***(h6) these trajectories become first negation trajectories in the Zone of B-BOMBER h5. Speaking informally, from squares f6, g6, and h6 W-FIGHTER can intercept B-BOMBER (in case of white move). The move-ordering procedure picks the subset of Zones with the main trajectories passing f6. These trajectories partly coincide with the potential first negation trajectories. The motion along such trajectories allows to "gain the time", i.e., to approach two goals simultaneously. Thus, 2. g7-f6.

This way proceeding with the search we will generate the tree that consists of 58 moves. Obviously, this is a drastic reduction in comparison with a million-move trees generated by conventional search procedures.

9. COMPLEX SYSTEM OF SPACE ROBOTIC VEHICLES

There is no specific character in the surface robotic battlefield, i.e., in the 2-D case. The Linguistic Geometry tools work analogously in the multi-dimensional cases. Consider 3-D control optimization problem. A representation of the space robotic system as a Complex System is analogous with the surface model (Figure 3). A set of X represents the operational district which could be the area of combat operation broken into smaller cubic areas, “points”, e.g., in the form of the big cube of 8 × 8 × 8, n = 512. It could be a space operation, where X represents the set of different orbits, or an air force battlefield, etc. P is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests. Analogously with the 2-D case $R_p(x,y)$ represent moving capabilities of different robots: robot p can move from point x to point y if $R_p(x, y)$ holds. ON(p)=x, if robot p is at the point x; v(p) is the value of robot p. It might include the immediate value of this robot for the given combat operation; S_i is an arbitrary initial state of operation for analysis, or the start state; S_t is the set of target states. These might be the states where robots of each side reached specified points. On the other hand S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_j) = x_k\}$ corresponds to the list of robots with their coordinates in each state. TRANSITION(p, x, y) represents the move of the robot p from the location x to location y; if a robot of the opposing side stands on y, a removal occurs, i.e., robot on y is destroyed and removed.

Space robotic vehicles with different moving capabilities are shown in Figure 16.

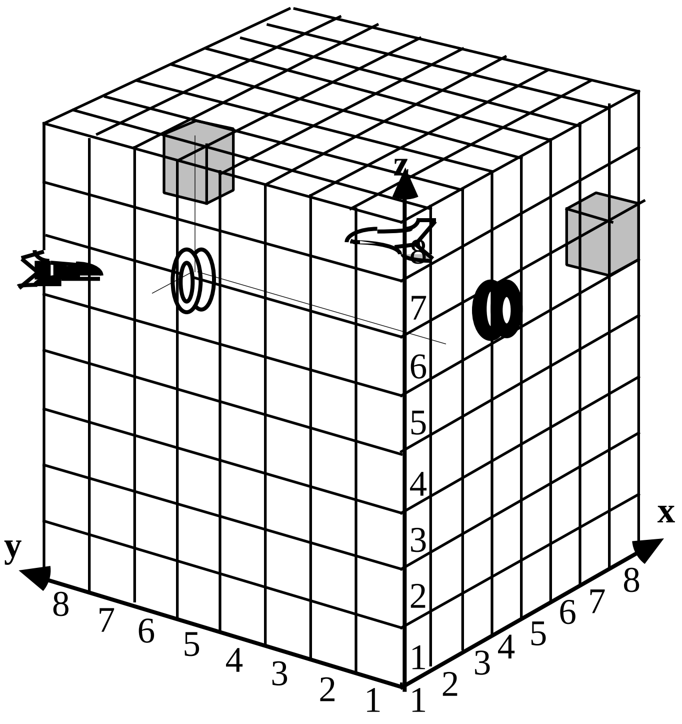

Figure 16. A problem for autonomous space robotic vehicles.

The operational district X is the 3-D table of 8 × 8 × 8. Robot W-INTERCEPTOR (White Interceptor) located at 118 (x=1, y=1, z=8), can move to any next location, i.e., 117, 217, 218, 228, 227, 128, 127. The other robotic vehicle B-STATION (double-ring shape in Figure 16) from 416 can move only straight ahead towards the goal area 816 (shaded in Figure 16), one square at a time, e.g., from 416 to 516, from 516 to 616, etc. Robot B-INTERCEPTOR (Black Interceptor) located at 186, can move to any next square similarly to robot W-INTERCEPTOR. Robotic vehicle W-STATION located at 266 is analogous with the robotic B-STATION; it can move only straight ahead towards the goal area 268 (shaded in Figure 16). Thus, robot W-INTERCEPTOR on 118 can reach any of the points y {117, 217, 218, 228, 227, 128, 127} in one step, i.e., $R_{\text{W-INTERCEPTOR}}(118, y)$ holds, while W-STATION can reach only 267 in one step.

Assume that robots W-INTERCEPTOR and W-STATION belong to one side, while B-INTERCEPTOR and B-STATION belong to the opposite side: W-INTERCEPTOR P_1, W-STATION P_1, B-INTERCEPTOR P_2, B-STATION P_2. Also assume that both goal areas, 816 and 268, are the safe areas for B-STATION and W-STATION, respectively, if station reached the area and stayed there for more than one time interval. Each of the STATIONs has powerful weapons capable to destroy opposing INTERCEPTORs at the next diagonal locations ahead of the course. For example W-STATION from 266 can destroy opposing INTERCEPTORs at 157, 257, 357, 367, 377, 277, 177, 167. Each of the INTERCEPTORs is capable to destroy an opposing STATION approaching its location from any direction, but it also capable to protect its friendly STATION approaching its prospective location. In the latter case the joint protective power of the combined weapons of the friendly STATION and INTERCEPTOR (from any next to the STATION area) can protect the STATION from interception. For example, W-INTERCEPTOR located at 156 can protect W-STATION on 266 and 267.

As in the 2-D case, the battlefield considered can be broken into two local operations. The first operation is as follows: robot B-STATION should reach the strategic point 816 safely and stay there for at list one time interval, while W-INTERCEPTOR will try to intercept this motion. The second operation is similar: robot W-STATION should reach point 268, while B-INTERCEPTOR will try to intercept this motion. After reaching the designated strategic area the (attacking) side is considered as a winner of the local operation and the global battle. The only chance for the opposing side to revenge itself is to reach its own strategic area within the next time interval and this way end the battle in a draw. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Figure 16.

Assume also that due to the shortage of resources (which is typical in real combat operation) or some other reasons, each side can not participate in both operations simultaneously. It means that during the current time interval, in case of White turn, either W-STATION or W-INTERCEPTOR can move. Analogous condition holds for Black. Of course, it does not mean that if one side began participating in one of the operations it must complete it. Any time on its turn each side can switch from one operation to another, e.g., transferring resources (fuel, weapons, human resources, etc.), and later switch back.

As it was in the 2-D problem, it seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-STATION from 418 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. Is there a strategy for the White side to make a draw?

Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Experiments with the computer chess programs showed that for the similar 2-D problem the search tree includes about a million moves (transitions). Of course, in the 3-D case the search would require billions of moves. It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools.

10. SPACE TRAJECTORIES GENERATION

In order to demonstrate generation of the Hierarchy of Languages for this problem, below we consider generation of the Language of Trajectories for the robotic system on example of generation of the shortest trajectory from point 336 to point 816 for the robot W-INTERCEPTOR (Figure 17 - xy projection, see also Figure 31 – xy projection). (Point 336 is the location of W-INTERCEPTOR in one of the states of the System in the process of the search.)

Consider the same Grammar of shortest trajectories $\mathbf{G_t^{(1)}}$(Table I). The controlled grammar shown in Table I can be used for generation of shortest trajectories for robots with arbitrary moving capabilities and different areas X. Values of $MAP_{336,W\text{-}INTERCEPTOR}$ are shown in Figure 18.

Thus, the distance from 336 to 816 for W-INTERCEPTOR is equal to 5. To be transparent we will show generation of trajectories located completely within the plane xy6 only. Thus, for this generation we will use 2-D coordinates.

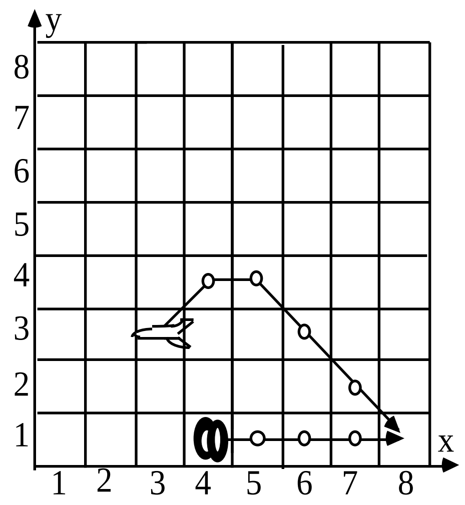

Figure 17. Interpretation of Zone for the Robotic System (projection to xy-plane).

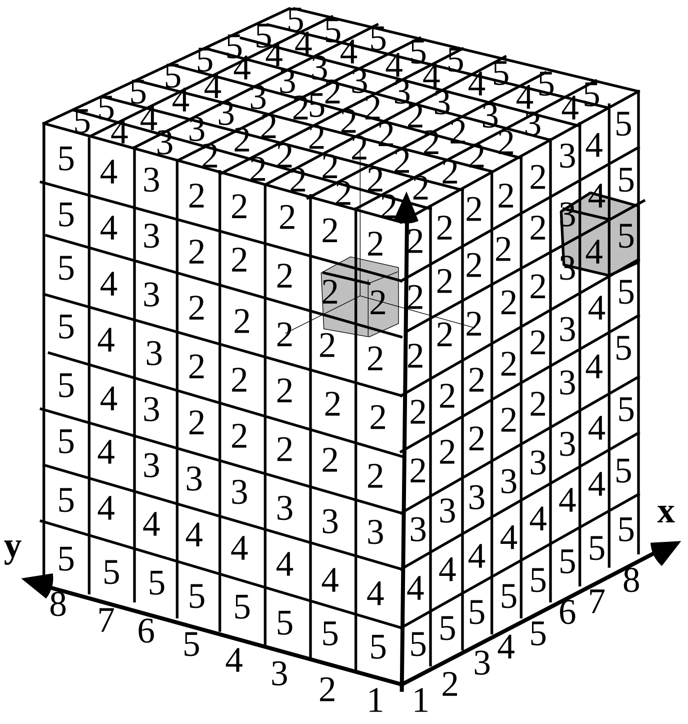

Figure 18. MAP_{336}, INTERCEPTOR

Applying the grammar $\mathbf{G}_t^{(1)}$ we have (symbol ^{l}=> means application of the production with the label l):

$$\boldsymbol{S}(33, 81, 5)^{1}\!\!=>\boldsymbol{A}(33, 81, 5)\ ^{2}1\!\!=>\boldsymbol{a}(33)\boldsymbol{A}(next_1(33, 5), 81, 5)$$

Thus we have to compute MOVE (see definition of the function $next_1$ from the grammar $\mathbf{G}_t^{(1)}$). First we have to determine the set of SUM, that is, we need to know values of

$$MAP_{33,W\text{-}INTERCEPTOR} \text{ and } MAP_{81,W\text{-}INTERCEPTOR}$$

(shown in Figure 19) on X.

5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	5
3	3	3	3	3	3	4	5
2	2	2	2	2	3	4	5
2	1	1	1	2	3	4	5
2	1	0	1	2	3	4	5
2	1	1	1	2	3	4	5
2	2	2	2	2	3	4	5

7	7	7	7	7	7	7	7
7	6	6	6	6	6	6	6
7	6	5	5	5	5	5	5
7	6	5	4	4	4	4	4
7	6	5	4	3	3	3	3
7	6	5	4	3	2	2	2
7	6	5	4	3	2	1	1
7	6	5	4	3	2	1	0

Figure 19. $MAP_{33,W\text{-}INTERCEPTOR}$ (left) and $MAP_{81,W\text{-}INTERCEPTOR}$ (right)

Adding these tables as matrices we compute SUM (Figure 20):

$SUM = \{v \mid v \quad X,\ MAP_{33,\ W\text{-}INTERCEPTOR}(v) + MAP_{81,W\text{-}INTERCEPTOR}(v) = 5\}$.

For the general 3-D case we should add 3-D matrices like those shown in Figure 18.

The next step is the computation of

$ST_1(33) = \{v \mid v \text{ from } X,\ MAP_{33,W\text{-}INTERCEPTOR}(v)=1\}$

which is shown in Figure 21.

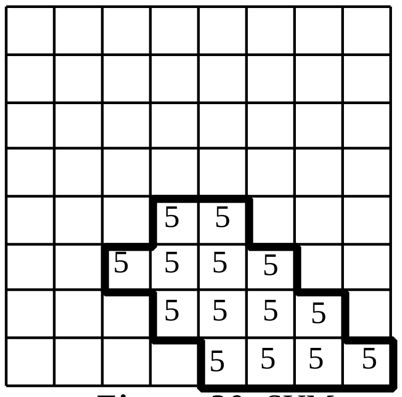

Figure 20. SUM.

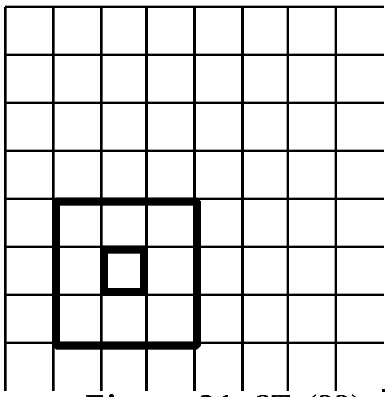

Figure 21. $ST_1(33)$.

In order to complete computation of the set $MOVE_5(33)$ we have to determine the following intersection:

$ST_1(33),\ ST_{5-5+1}(33) = ST_1(33)$ and SUM.

Consequently, $MOVE_5(33)=\{44, 43, 42\}$; and $next_1(33, 5)=44$, $next_2(33, 5)=43$, $next_3(33, 5) = 42$. Since the number of different values of *next* is equal to 3 (here r=3, see definition of the function *next*, Table I) we could branch at this step, apply productions 2_1, 2_2 and 2_3 simultaneously, and continue both derivations independently. This could be accomplished in a parallel computing environment. Let us proceed with the first derivation.

$\boldsymbol{a}(33)\boldsymbol{A}(44, 81,4)\ {}^{2}1\Rightarrow \boldsymbol{a}(33)\boldsymbol{a}(44)\boldsymbol{A}(next_1(44, 4), 81, 3)$

We have to compute $next_1(44, 4)$ and, as on the preceding step, have to determine $MOVE_4(44)$. To do this we have to compute:

$ST_1(44)=\{v \mid v \quad X,\ MAP_{44,W\text{-}INTERCEPTOR}(v)=1\}$,(Figure 22)

$ST_{5-4+1}(33) = ST_2(33) = \{v \mid v \quad X,\ MAP_{33,W\text{-}INTERCEPTOR}(v)=2\}$, (Figure 23).

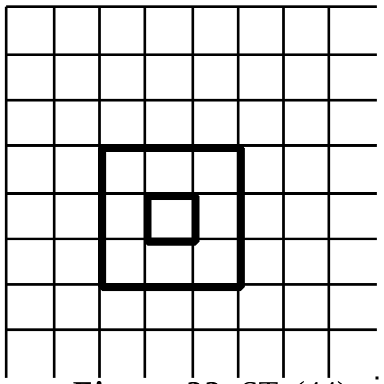

Figure 22. $ST_1(44)$

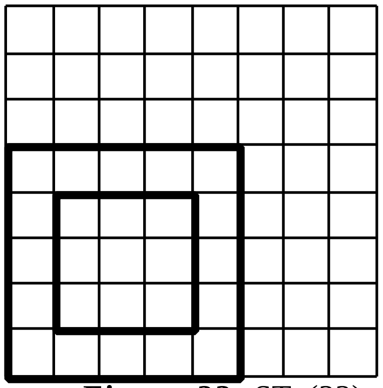

Figure 23. $ST_2(33)$.

The set of SUM is the same for all steps of the derivation. Hence, $MOVE_4(44)$ is the intersection of the sets shown in Figure 20, 22, 23:

$MOVE_4(44)= \{54, 53, 52\}$; and $next_1(44, 4) = 54$; $next_2(44, 4) = 53$, $next_3(44, 4) = 52$.

Thus, the number of different values of the function *next* is equal to 3 (r=3), so the number of continuations of derivation should be multiplied by 3.

Let us proceed with the first one:

$$\boldsymbol{a}(33)\boldsymbol{a}(44)\boldsymbol{A}(54, 81, 3) \ {}^{2}1\Rightarrow \ldots$$

Eventually, we will generate one of the shortest trajectories for the robot W-INTERCEPTOR from 33 to 81:

$$\boldsymbol{a}(33)\boldsymbol{a}(44)\boldsymbol{a}(54)\boldsymbol{a}(63)\boldsymbol{a}(72)\boldsymbol{a}(81).$$

Similar generating techniques are used to generate higher level subsystems, the networks of paths, i.e., the Language of Zones. For example, the incomplete Zone shown in Figure 17 is as follows (in 2-D coordinates):

$$\boldsymbol{t}(\text{B-STATION}, t_B, 5)\boldsymbol{t}(\text{W-INTERCEPTOR}, t_F, 5),$$

where

$$t_B = \boldsymbol{a}(41)\boldsymbol{a}(51)\boldsymbol{a}(61)\boldsymbol{a}(71)\boldsymbol{a}(81),\ t_F = \boldsymbol{a}(33)\boldsymbol{a}(44)\boldsymbol{a}(54)\boldsymbol{a}(63)\boldsymbol{a}(72)\boldsymbol{a}(81).$$

11. SEARCH GENERATION FOR SPACE ROBOTIC SYSTEM WITHIN INSUFFICIENT VIEW RANGE (HORIZON)

Consider how the hierarchy of languages works for the optimal control of the space robotic system introduced above (Figure 16). We will generate the string of the Language of Translations representing it as a conventional search tree (Figure 25, 27) and comment on its generation. We will show that this tree is very close to the search tree of the relative 2-D problem (Section 9).

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limited number of steps which is called a horizon. In general, the value of the horizon is unknown. As a rule, this value can be determined from the experience of solving specific classes of problems employing Linguistic Geometry tools. In absence of such experience, first we have to consider the value of 1 as a horizon, and solve the problem within this limit. According to the model shown in Figure 16 within this horizon no one element can “attack” any other element or goal areas: the model is completely “blind”. Hence, no motion is allowed. If we still have resources available, i.e., computer time, memory, etc., we can increase the horizon by one. After each increase we have to regenerate the entire model. This increase means a new level of “vigilance” of the model, and, consequently, new greater need for resources. All the Zones generated within the horizon 2 in the start state are shown in Figure 24.

For example, one of the Zones for W-STATION, Z_{WS}, is as follows:

$$Z_{WS}=\boldsymbol{t}(\text{W-STATION}, \boldsymbol{a}(266)\boldsymbol{a}(267)\boldsymbol{a}(268), 3)\boldsymbol{t}(\text{B-INTERCEPTOR}, \boldsymbol{a}(186)\boldsymbol{a}(277)\boldsymbol{a}(268), 3)$$
$$\boldsymbol{t}(\text{B-INTERCEPTOR}, \boldsymbol{a}(186)\boldsymbol{a}(276)\boldsymbol{a}(267), 2)\boldsymbol{t}(\text{W-STATION}, \boldsymbol{a}(266)\boldsymbol{a}(277), 1)$$

The other trajectories of B-INTERCEPTOR, e.g., the second trajectory, $\boldsymbol{a}(186)\boldsymbol{a}(177)\boldsymbol{a}(268)$, leading to the point 268, is included into the different Zone; for each Zone only one trajectory from each bundle of trajectories with the same beginning and end is taken.

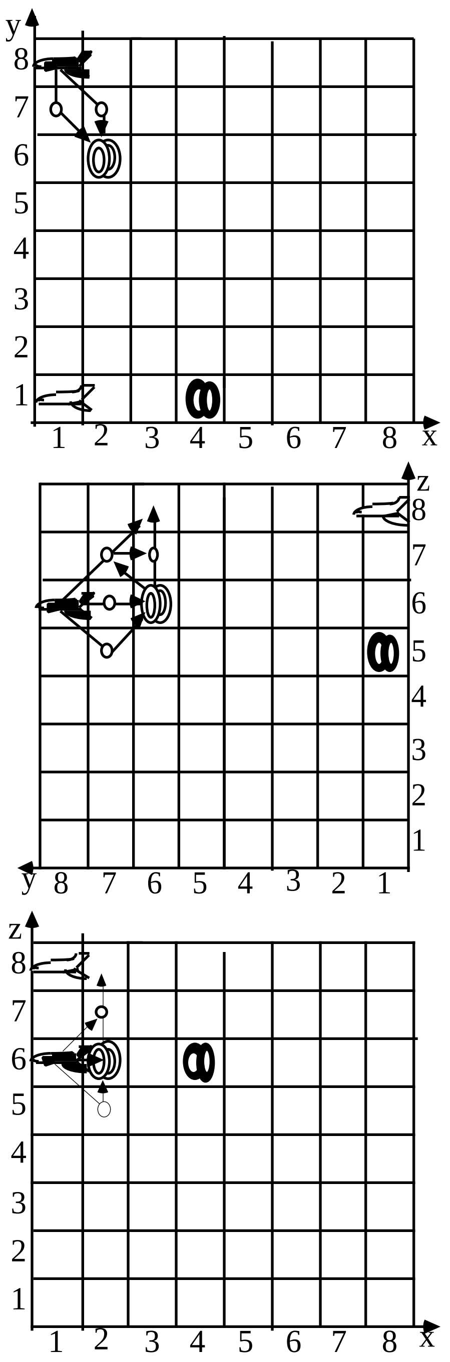

Figure 24. Interpretation of Zones generated within the horizon of 2 in the start state of the space robotic system (3 projections)

The other set of Zones, also shown in Figure 24, is the set of Zones of B-INTERCEPTOR with main trajectories from 186 to 266. As in the case of intercepting trajectories each main trajectory from this bundle is included into the different Zone. Obviously, there are no Zones within the horizon 2 for the main elements W-INTERCEPTOR and B-STATION. It means that B-STATION can not reach the goal area 861. Moreover, neither W-INTERCEPTOR nor B-STATION have intercepting trajectories to participate in some other Zones. Hence, they can not move at all.

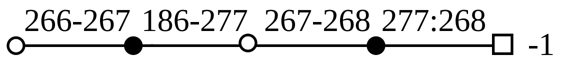

Figure 25. Search tree for the optimization problem within the horizon 2.

The search tree for this model, actually one branch, is presented in Figure 25. (The details of tree generation will be considered later for the model within the horizon 5.) It turns out that the optimal solution within the horizon 2 is a win of the Black side, because W-STATION has been destroyed. At the same time B-STATION would never reach its destination. Obviously this solution can not be considered as reasonable. This model is still “blind”. It is very likely that after this search we still have resources allowing us to increase the horizon.

It is easy to show that within the horizons 1, 2, 3, 4 all the models generated for this problem are “blind” and corresponding searches do not give a reasonable solution. But, again, after application of each of the consecutive values of the horizon we will have a *solution*, which can be considered as an approximate final solution within the available resources.

12. SEARCH GENERATION FOR SPACE ROBOTIC SYSTEM WITHIN THE HORIZON 5

Let the horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of the main trajectories of all Zones must not exceed 5 steps. All the Zones generated in the start state are shown in Fig. 26.

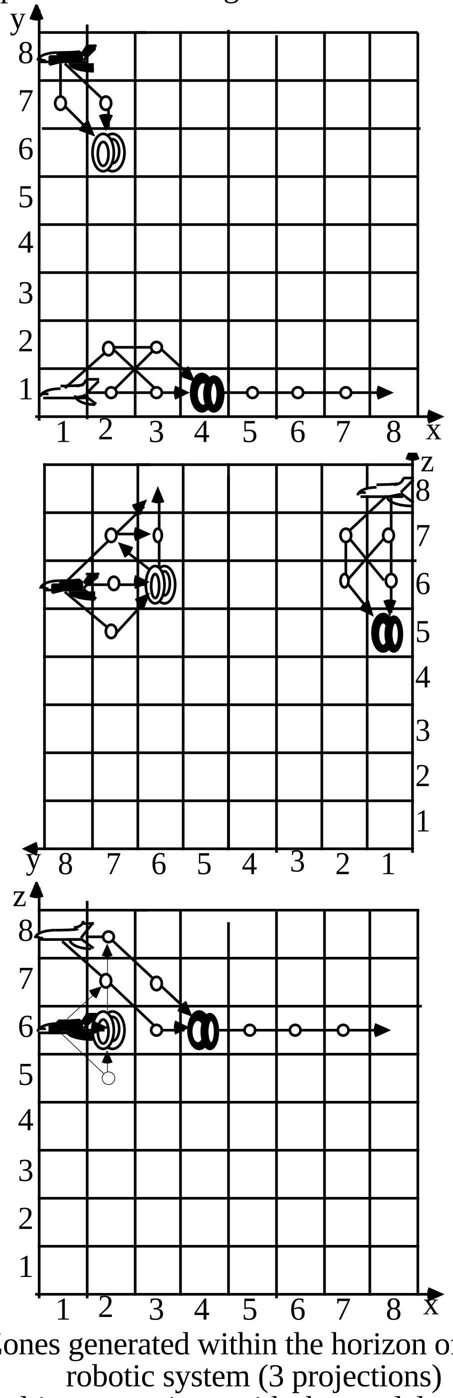

Figure 26. Interpretation of Zones generated within the horizon of 5 in the start state of the space robotic system (3 projections)

Nothing new has been generated in comparison with the model within the horizon of 4. But we should not forget that the same increased horizon will be applied in every state during the search.

Search tree generation (Figure 27) begins with the move 1. 266-267 in the “white” Zone with the target of the highest value and the shortest main trajectory.

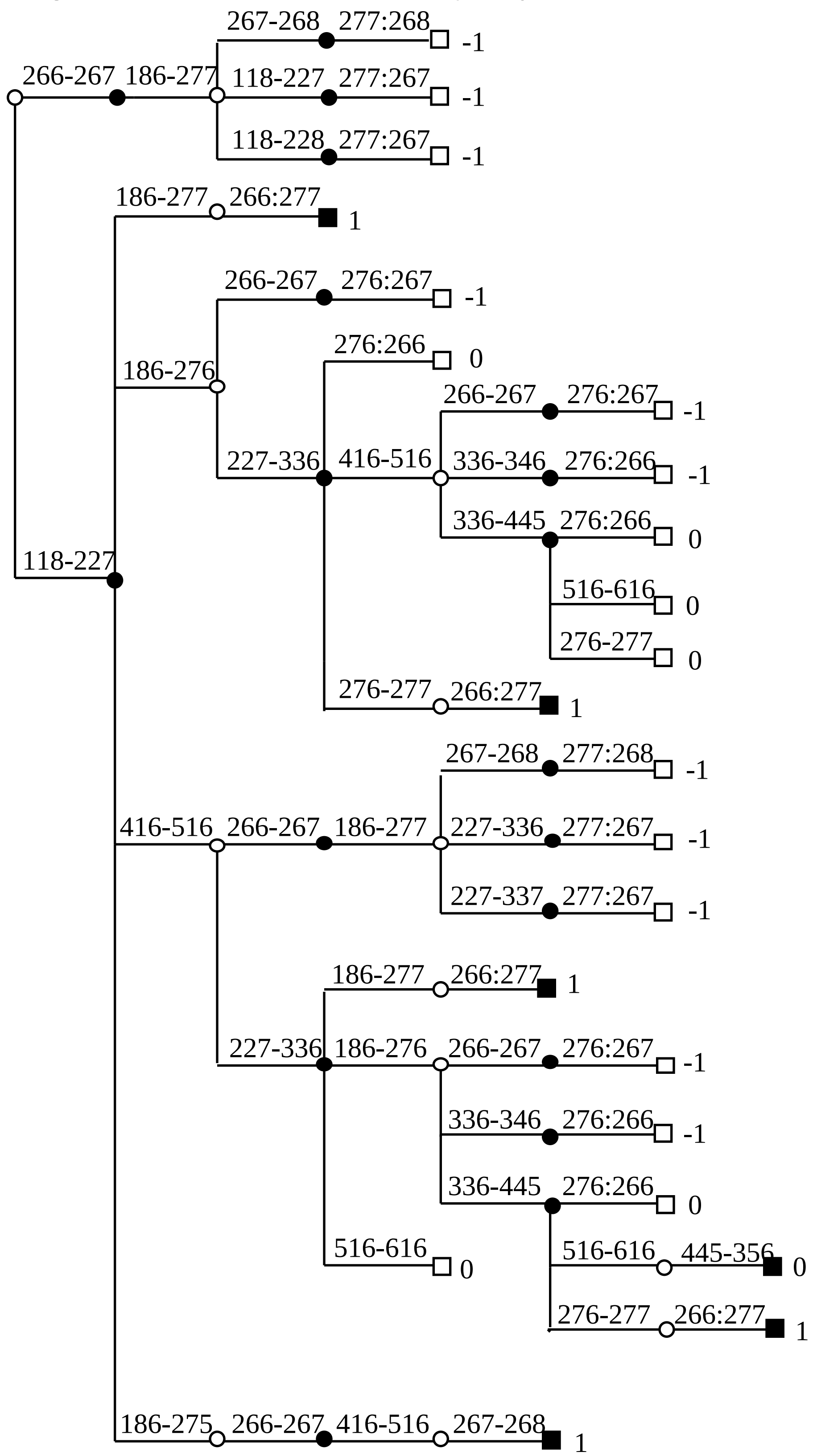

Figure 27. Search tree for the optimization problem within the horizon 5.

The order of consideration of Zones and particular trajectories is determined by the grammar of translations. As it was mentioned Section 11 for 2-D problem, the computation of move-ordering constraints is the most sophisticated procedure in the Grammar of Translations. It takes into account different parameters of Zones, trajectories, and the so-called chains of trajectories. We should keep in mind that after each move the model moves to the new current state S_c, so the entire Language of Zones, $L_Z(S_c)$, must be regenerated. With respect to efficiency of the model it is very important to solve one technical problem relative to the well known Frame Problem [14, 7, 14, 17]. We have to answer how to avoid recomputation of the entire language recomputing only the changing part? An approach to the formal solution of this problem is considered in [34].

Next move, 1. ... 186-277, is in the same Zone along the first negation trajectory. The interception continues: 2. 267-268 277:268. This state is shown in Figure 28.

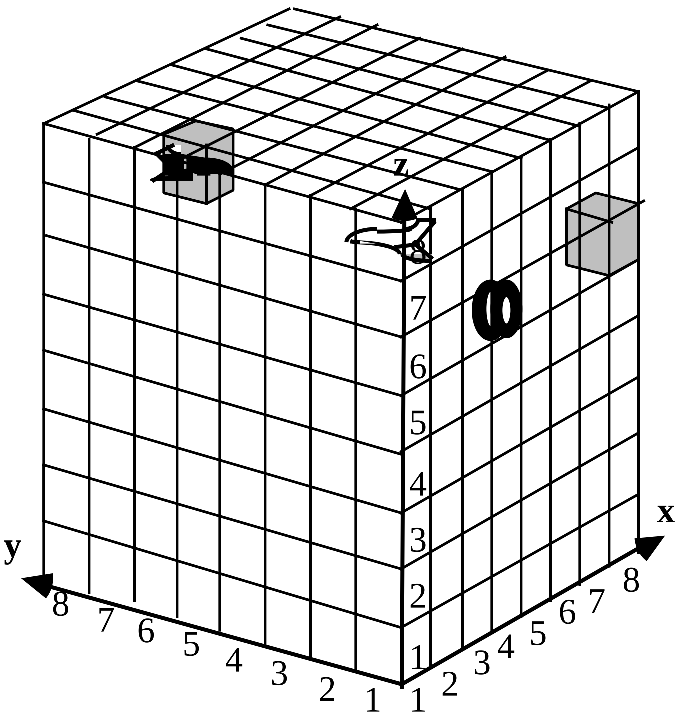

Figure 28. The state where the control Zone from 118 to 268 was detected.

Symbol ":" means the removal of element. Here the grammar terminates this branch with the value of -1 (as a win of the Black side). This value is given by the special procedure of "generalized square rules" built into the grammar.

In this terminal state the grammar detected the whole bundle of Zones with the main trajectories from 118 to 268 within the horizon of 5 (Figure 28). It happened, because this is the first state where new target B-INTERCEPTOR arrived in the area 268 which is reachable within the horizon of 5 from 118 for W-INTERCEPTOR. Inclusion of these new Zones in this state is too late (this is a terminal state), but it could make sense in the "past". Speaking informally, if the grammar "knew" about these Zones in the states it moved through earlier during the search, for example in the start state, these Zones might be included, and, possibly, different moves could be picked for building the search tree. Thus, the grammar learned the new information about the problem in a state, which is in the depth of the search tree. This information should be brought to the upper levels of the search tree; the grammar stores these newly generated Zones as idle for possible activation in different states.

Then, the grammar initiates the backtracking climb. After leaving the state where B-INTERCEPTOR was in the area 268, our idle Zones "lose" the target. Now they are called "control" Zones: they can be activated assuming that target will arrive in the future, during next descent. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search. After the climb up to the move 1. ... 186-277, the tree to be analyzed consists of one branch (of two plies): 2. 267-268 277-268. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange in the area 268 (in favor of the White side). This can be achieved by participation of W-INTERCEPTOR from 118, i.e., by inclusion of the currently idle control Zone with the main trajectory from 118 to 268. The set of different Zones from 118 to 268 (the bundle of Zones) is shown in Figure 29.

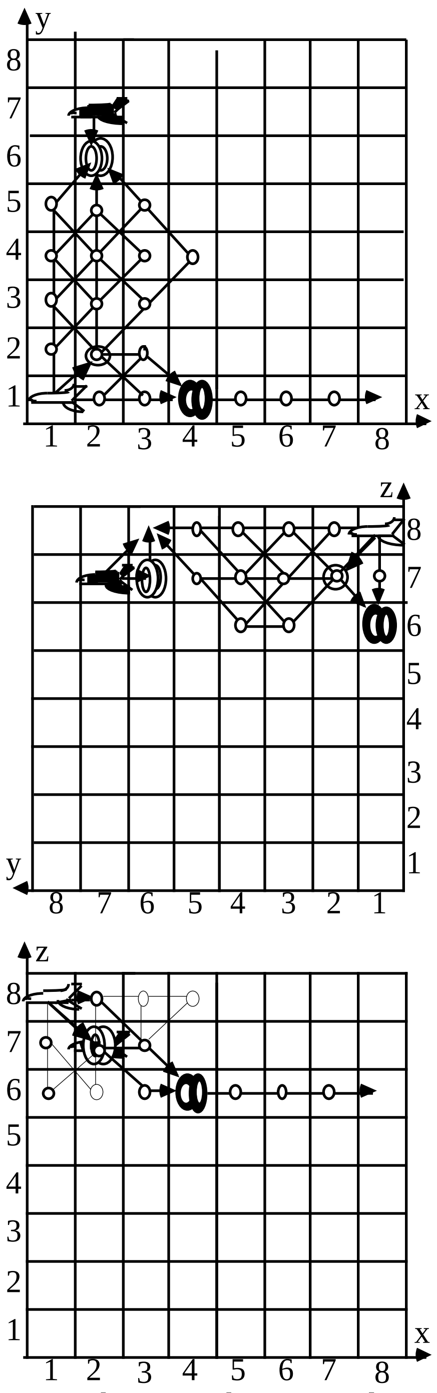

Figure 29. Interpretation of Zones in the state where control Zone from 118 to 268 was included first (3 projections).

The move-ordering procedure picks the subset of Zones with main trajectories passing 227. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing B-STATION on 516. The motion along such trajectories allows to “gain the time”, i.e., to approach two goals simultaneously.

The generation continues: 2. 118-227 277-267. Again, the procedure of “square rules” terminates the branch, evaluates it as a win of the Black side, and the grammar initiates the climb. Move 2. 118-227 is changed for 2. 118-228. Analogously to the previous case, the inspection procedure determined that the current minimax value (-1) can be improved by the improvement of the exchange on 267. Again, this can be achieved by the inclusion of Zone from 118 to 267. Of course, the best “time-gaining” move in this Zone is 2. 118-227, but it was already included (as move in the Zone from 118 to 268). The other untested move in the Zone from 118 to 267 is 2. 118-228. Obviously the grammar does not have knowledge that trajectories to 267 and 268 are “almost” the same.

After the next termination and climb, the inspection procedure does not find new Zones to improve the current minimax value, and the climb continues up to the start state. The analysis of the subtree shows that inclusion of Zone from 118 to 268 in the start state can be useful: the minimax value can be improved. Similarly, the most promising “time-gaining” move is 1. 118-227. The Black side responded 1. ... 186-277 along the first negation trajectories ***a***(186)***a***(277)***a***(267) and ***a***(186)***a***(277)***a***(268) shown in Figure 26 (better see yz-projection). Obviously, 2. 266:277, and the branch is terminated. The grammar initiates the climb and move 1. ... 186-277 is changed for 1. ... 186-276 along the trajectory ***a***(186)***a***(276)***a***(266). Note, that grammar “knows” that in this state trajectory ***a***(186)***a***(276)***a***(266) is active, i.e., B-INTERCEPTOR has enough time for interception. The following moves are in the same Zone of W-STATION: 2. 266-267 276:267. This state is shown in Figure 30. The “square rule procedure” cuts this branch and evaluates it as a win of the Black side. Moreover, this is the state where a set of new control Zones from 227 to 267 was detected but not included.

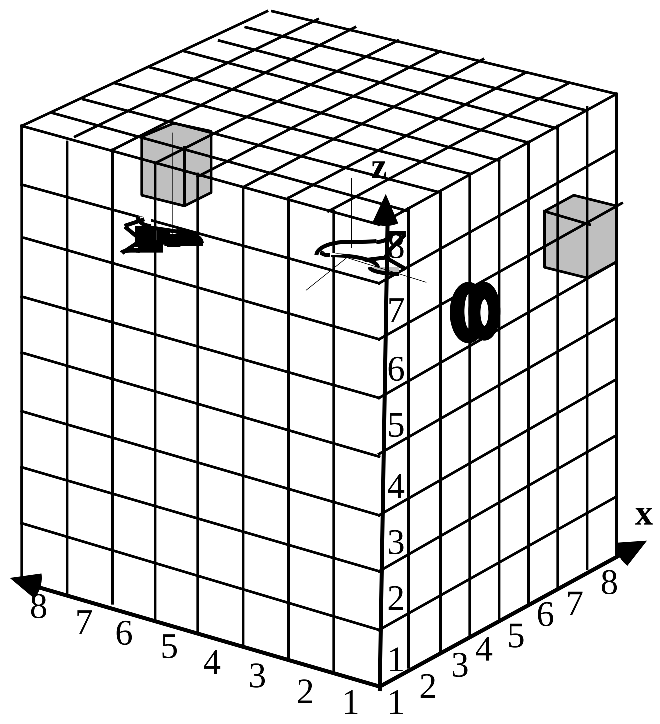

Figure 30. The state where control Zone from 227 to 267 was detected.

New climb up to the move 2. ... 186-276 and execution of the inspection procedure result in the inclusion of the new control Zone from 227 to 267 in order to improve the exchange in the area 267. The set of Zones with different main trajectories from 227 to 267 is shown in Figure 31. Besides that, the trajectories from 227 to 516, 616, 716, and 816 are shown in the same Figure 31. These are “potential” first negation trajectories. It means that beginning with the second symbol ***a***(336), ***a***(337), ***a***(338), or ***a***(326), ***a***(327), ***a***(328), or ***a***(316), ***a***(317), ***a***(318), these trajectories become first negation trajectories in the Zone of B-STATION on 461. Speaking informally, from the areas listed above W-INTERCEPTOR can intercept B-STATION (in case of white move). The main trajectories of control Zones passing one of three points, 336, 337, or 338, partly coincide with the potential first negation trajectories.

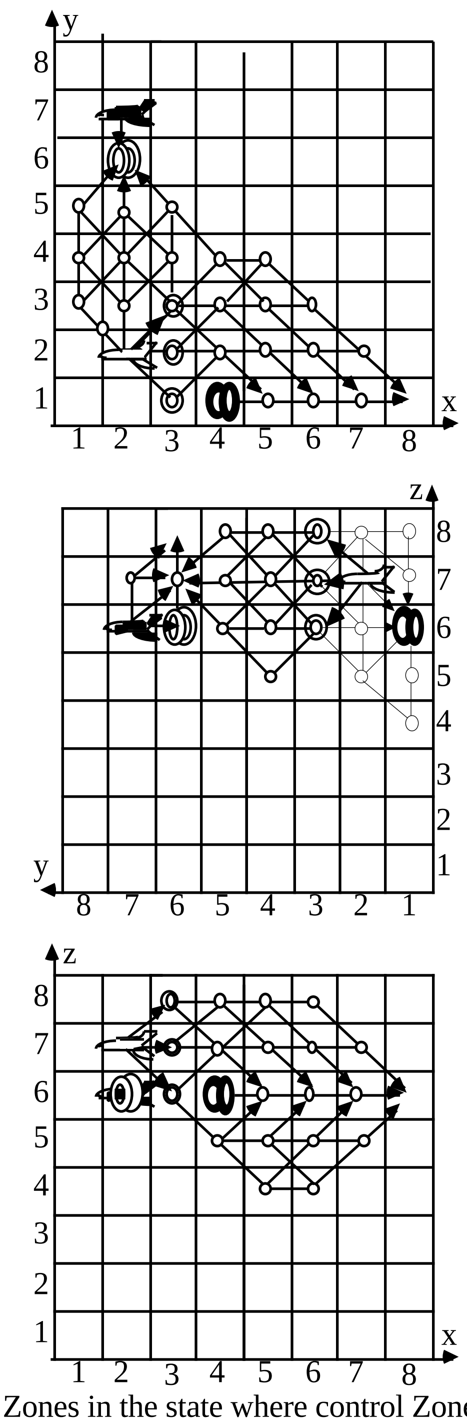

Figure 31. Interpretation of Zones in the state where control Zone from 227 to 267 was included first (3 projections).

The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously. The move-ordering procedure picks the subset of Zones with the main trajectories passing 336. Thus, 2. 227-336.

This way proceeding with the search we will generate the tree that consists of 56 moves. Obviously, this is a drastic reduction in comparison with a billion-move trees generated by conventional search procedures.

13. CONCLUSION

Robotic examples considered in this paper demonstrate the power of the Linguistic Geometry tools that allowed to transfer heuristics discovered in one problem domain, specifically, in the game of chess, to another domain of the surface and space robotic navigation. It is even more interesting that search reduction achieved in the original domain multiplied tremendously in the newest one. Indeed, the conventional approaches employing search algorithms with alpha-beta pruning require a billion move tree to solve 3-D problem, while the tree presented in this paper (Figure 27) consists of about 50 moves and is very similar with the tree from the 2-D case (Figure 12). In both problems the branching factor [17], i.e., the "average number" of branches in each node of the search tree, is very close to 1 (1.65). This means that the algorithm is actually goal-oriented, i.e., it approaches the goal almost without branching. Looking at the complexity of the hierarchy of languages which represents each state in the search process and at the size of the search trees, it is very likely that the growth from the 2-D case to 3-D is linear with the factor close to one. This means that the complexity of the entire algorithm may be about linear with respect to the length of the input.

At the same time the space navigation problem considered here is still very close to the original chess problem, and, of course, to the surface system. It is possible to predict that the power of the Linguistic Geometry goes far beyond these limits. The definition of Complex System (see Section 2) is generic enough to cover a variety of different problem domains. The core component of these definition is the triple X, P, and R_p. Thus, considering a new problem domain we have to define X, the finite set of points – locations of elements. We do not impose any constraints to this set while the surface and space operational districts X considered in this paper as well as the original chess board have different extra features, e.g., 2-D or 3-D connectivity, which is totally unimportant for these problems. For example, coming closer to the real world problems, we can consider X as a set of orbits where the elements are in constant motion with respect to each other. The set of elements P, i.e., the set of movable units, in our problems is quite small, while their moving capabilities, the binary relations of R_p, are non-sophisticated. Indeed, during one time interval our robotic vehicles, i.e., fighters, stations and space interceptors can move only to the next area. Even in the game of chess the moving capabilities of different pieces are much more advanced. This is exactly the place for introduction of the variable speed, the gravity impact, the engine impulse duration, etc.

Application of the Linguistic Geometry tools to the new, yet to be predicted, problem domains, should allow for the expansion of advanced human heuristic methods discovered in different complex systems to other real-world systems where existing methods are not sufficient.

REFERENCES

1. Albus, J. (1991) "Outline for a Theory of Intelligence", *IEEE Trans. on Systems, Man and Cybernetics*, 3:473-509.
2. Botvinnik, M.M. (1984) *Computers in Chess: Solving Inexact Search Problems.* Springer Series in Symbolic Computation, Springer-Verlag, New York.
3. Botvinnik, M., Petriyev, E., Reznitskiy, A., et al. (1983) "Application of New Method for Solving Search Problems For Power Equipment Maintenance Scheduling", *Economics and Mathematical Methods*, 6:1030-1041 (in Russian).
4. Chapman, D. (1987) "Planning for conjunctive goals", *Artificial Intelligence* 32(3).
5. Chomsky, N. (1963) "Formal Properties of Grammars", in *Handbook of Mathematical Psychology*, eds. R.Luce, R.Bush, E. Galanter., vol. 2, John Wiley & Sons, New York, pp. 323-418.
6. Feder, J. (1971) "Plex languages", *Information Sciences*, 3: 225–241.
7. Fikes, R.E. and Nilsson, N.J. (1971) "STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving", *Artificial Intelligence* 2: 189–208.
8. Fu, K.S. (1982) *Syntactic Pattern Recognition and Applications*, Prentice Hall, Engl. Cliffs.
9. Garey, M.R. and D.S.Johnson D.S. (1991) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco.
10. Ginsburg, S. (1966) *The Mathematical Theory of Context-Free Languages*, McGraw Hill, New York.
11. Knuth, D.E. (1968) "Semantics of Context-Free Languages", *Mathematical Systems Theory* 2: 127–146.
12. McAllester, D. and Rosenblitt, D. (1991) Systematic Non-Linear Planning, *Proc. of AAAI-91*, pp. 634-639.
13. McCarthy, J. (1980) "Circumscription–A Form of Non-Monotonic Reasoning", *Artificial*

Intelligence 13:27-39.

14. McCarthy, J. and Hayes, P.J. (1969) "Some Philosophical Problems from the Standpoint of Artificial Intelligence", *Machine Intelligence* 4: 463–502.
15. Mesarovich, M.D., Macko, D., and Takahara Y. (1970) *Theory of Hierarchical Multilevel Systems*., Academic Press, New York.
16. Narasimhan, R.N. (1966) "Syntax–Directed Interpretation of Classes of Pictures", *Communications of the ACM* 9: 166–173.
17. Nilsson, N.J. (1980) *Principles of Artificial Intelligence*, Tioga Publ., Palo Alto, CA.
18. Pavlidis, T. (1977) *Structural Pattern Recognition*, Springer-Verlag, New York.
19. Reznitskiy, A.I. and Stilman, B. (1983) "Use of Method PIONEER in Automating the Planning of Maintenance of Power-Generating Equipment," *Automatics and Remote Control*, 11: 147-153, (in Russian).
20. Rosenfeld, A.(1979) *Picture Languages, Formal Models for Picture Recognition,* Acad. Press.
21. Rozenkrantz, D.J. (1969) "Programmed Grammars and Classes of Formal Languages," *J. of the ACM*, 1:107–131.
22. Sacerdoti, E.D. (1975) "The Nonlinear Nature of Plans," *Proc. Int. Joint Conference on Artificial Intelligence.*
23. Shaw, A.C. (1969) "A Formal Picture Description Scheme as a Basis for Picture Processing System," *Information and Control* 19: 9-52.
24. Simon, H.A. (1980) *The Sciences of the Artificial*, 2-nd ed., MIT Press, Cambridge, MA.
25. Stefik, M. (1981) Planning and meta-planning (MOLGEN: Part 2)," *Artificial Intelligence,* 2: 141-169.
26. Stilman, B. (1977) "The Computer Learns", in Levy, D.,*1976 US Computer Chess Championship,* Computer Science Press, Woodland Hills, CA, pp. 83-90.
27. Stilman, B. (1985) "Hierarchy of Formal Grammars for Solving Search Problems," in *Artificial Intelligence. Results and Prospects, Proceedings of the International Workshop,* Moscow, 63–72, (in Russian).
28. Stilman, B. (1992) "A Linguistic Geometry of Complex Systems, *Abstr. Second Int. Symposium on Artificial Intelligence and Mathematics*," Ft. Lauderdale, FL. The full paper was submitted to *Annals of Math. & Artificial Intelligence.*
29. Stilman, B. (1992) "A Syntactic Structure for Complex Systems", *Proc. Second Golden West Int. Conf. on Intelligent Systems,* Reno, NE, June, 269-274.
30. Stilman, B. (1992) "A Geometry of Hierarchical Systems: Generating Techniques", *Proc. Ninth Israeli Conference on Artificial Intelligence and Computer Vision,* pp. 95-109, Dec., Tel Aviv, Israel.
31. Stilman, B. (1992) "A Syntactic Approach to Geometric Reasoning about Complex Systems," *Proc. Fifth Int. Symp. on Artificial Intelligence*, Cancun, Mexico, Dec., 115-124.
32. Stilman, B. (1993) "A Linguistic Approach to Geometric Reasoning," *Int. J. Computers and Mathematics with Applications*, 26(7), 29-57.
33. Stilman, B. (1993) "Network Languages for Complex Systems," *Int. J. Computers and Mathematics with Applications*, 26(8), 51-79.
34. Stilman, B. (1994) "Translations of Network Languages," *Int. J. Computers and Mathematics with Applications,* 27(2), 65-98.
35. Stilman, B. (1993) "Syntactic Hierarchy for Robotic Systems," *Integrated Computer-Aided Engineering,* 1(1), 57-81.
36. Stilman, B. (1993) "A Formal Language for Hierarchical Systems Control," *Languages of Design,* 1(4), 333-356.
37. Stilman, B. (1994) "A Formal Model for Heuristic Search," *Proc. of the 22nd Annual ACM Computer Science Conf.*, Phoenix, AZ, March 1994, 380-389.
38. Stilman, B., (1994) "A Linguistic Geometry for Space Applications", *Proc. of 1994 Goddard Conf. on Space Applications of Artificial Intelligence.*, NASA Goddard Space Flight Center, Greenbelt, MD, May 1994, 87-101.
39. Volchenkov, N.G. (1979) "The Interpreter of Context-Free Controlled Parameter Programmed Grammars," in L.T. Kuzin, Eds., *Cybernetics Problems. Intellectual Data Banks*, The USSR Academy of Sci., Moscow, pp. 147–157, (in Russian).