MULTIAGENT AIR COMBAT WITH CONCURRENT MOTIONS

D R A F T

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

Abstract: This paper is a new step in the development of the Linguistic Geometry. This formal theory is intended to discover the inner properties of human expert heuristics, which have been successful in a certain class of complex control systems, and apply them to different systems. The Linguistic Geometry relies on the formalization of search heuristics of the highly-skilled human experts, which allow for the decomposition of a complex system into a dynamic hierarchy of subsystems, and thus solve intractable problems by reducing the search dramatically. In this paper we report application of the Linguistic Geometry tools to a new example of a solution of simplified 2D optimization problem for the autonomous robotic vehicles in aerospace environment. The novelty of this example with respect to other Linguistic Geometry applications is that in this multiagent problem some agents can move simultaneously.

Keywords: Artificial Intelligence, Multiagent Systems, Linguistic Geometry, Heuristic Search, Network Languages, Concurrent Motions.

1. Introduction

Air Force systems operate in the atmosphere, ionosphere, near-Earth space and through these regions. They include aircraft combat missions, tracking and possible interception of missiles and satellites, surveillance operations, etc. A great number of aerospace problems such as long and short-range mission planning, especially for autonomous navigation, scheduling, aerospace robot control, long-range satellite service, aerospace combat operations control, etc. can be formally represented as reasoning about complex large-scale control systems. The field of efficient aerospace control systems design needs new technology from the science of artificial intelligence (Rodin, 1988; Rodin et al., 1993; Lirov, Rodin et al., 1988). In particular, discrete-event modeling of complex control systems can be implemented as a purely interrogative simulation. These techniques can be based on generating geometrically meaningful states rather than time increments with due respect to the timeliness of actions. As stated in (Lirov, Rodin et al., 1988) this interrogative approach offers much faster execution and clearer simulator definition. For this kind of approach a series of hierarchical dynamic goal-oriented systems should be developed and investigated.

There are many such 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, especially in the aerospace problem domain, where advances are required but human problem-solving skills can not be directly applied. For example, there are problems of tactics planning and automatic control of autonomous agents such as aerospace vehicles, space 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. Due to the special significance of these problems and the fabulous costs of mistakes, the quality of solutions must be very high and usually subject to continuous improvement.

In this respect 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, and first and foremost to the aerospace critical complex systems. It should be considered as investigation, development, and consequent expansion of advanced human expert skills into new areas.

2. Theoretical Background

The difficulties we encounter trying to find the optimal operation for real-world complex control systems are well known. While the formalization of the problem, as a rule, is not difficult, an algorithm that finds its solution usually results in the search of many variations. For small-dimensional "toy" problems a solution can be obtained; however, for most real-world problems the dimension increases and the number of variations increases significantly, usually exponentially, as a function of dimension (Garey and Johnson, 1991). Thus, most real-world search problems are not solvable with the help of exact algorithms in a reasonable amount of time. This becomes increasingly critical for the real-time aerospace autonomous and semiautonomous

vehicles and robots (Lirov et al., 1988; Strosnider and Paul, 1994).

There have been many attempts to find the optimal (suboptimal) operation for real-world complex systems, in particular, for aerospace applications (Leitmann, 1990; Drabble, 1991; Pigeon et al., 1992). Basically, all the approaches for the limited time search can be broken into four categories: the imprecise computation (Chung et al., 1990), real-time search (e.g., Korf, 1990), approximate processing (Lesser et al., 1988), and anytime algorithms (Dean and Boddy, 1988). According to Strosnider and Paul (1994) the correct pruning in its many manifestations is still the only technique that reduces the worst-case execution time without compromising the goal state. But for real-world applications this reduction is usually insufficient: it does not overcome the combinatorial explosion. Another technique, such as approximate processing, scoping, and use of domain knowledge, can reduce execution time significantly but they might compromise the goal state.

One of the basic ideas is to decrease the dimension of the real-world system following the approach of a *human expert in the field*, by breaking the system into smaller subsystems. This process of decomposition can be applied recursively until we end up with a collection of basic subproblems that can be treated (in some sense) independently. These ideas have been implemented for many problems with varying degrees of success (see, e.g., Albus, 1991; Knoblock, 1990; Mesarovich et al, 1989; Botvinnik, 1984). Implementations based on the formal theories of linear and nonlinear planning meet hard efficiency problems (McAllester and Rosenblitt, 1991; Chapman, 1987; Nilsson, 1980; Stefik, 1981; Sacerdoti, 1975). An efficient planner requires an intensive use of heuristic knowledge. Moreover it is possible to use both dynamic and static heuristic knowledge in reducing the search variations. The dynamic knowledge can be acquired during the run time and immediately applied for search reduction (Strosnider and Paul, 1994). On the other hand, a pure heuristic implementation is unique. There is no general constructive approach to such implementations. Each new problem should be carefully studied, and previous experience usually can not be applied. Basically, we can not answer the question: what are the formal properties of the human expert heuristics that drove us to a successful hierarchy of subsystems for a given problem, and how can we apply the same ideas in a an altered or even different problem domain? Moreover, every attempt to evaluate the computational complexity and quality of a pilot solution necessitates implementing its program, which in itself is a unique task for each problem.

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 (1963), Ginsburg (1966), 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 (1982), Narasimhan (1966), and Pavlidis (1977), and picture description languages by Shaw (1969), Feder (1971), and Rosenfeld (1979).

Searching for adequate mathematical tools formalizing human heuristics of dynamic hierarchies, 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 (Stilman, 1985). 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 (1968), Rozenkrantz (1969), and Volchenkov (1979).

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 (1980), Fikes and Nilsson (1971), Sacerdoti (1975), McCarthy (1980), McCarthy and Hayes (1969), and others based on first order predicate calculus.

In the beginning of 80's Botvinnik, Stilman, and others developed one of the most interesting and powerful heuristic hierarchical models. It was successfully applied to scheduling, planning, control, and computer chess. The hierarchical networks were introduced in (Botvinnik, 1984; Stilman, 1977) in the form of ideas, plausible discussions, and program implementations. We consider this model as an ideal case for transferring the developed search heuristics to other domains employing formal linguistic tools.

An application of the developed model to a chess domain was implemented in full as program PIONEER (Botvinnik, 1984). Similar heuristic model was implemented for power equipment maintenance in a number of computer programs being used for maintenance scheduling all over the USSR (Botvinnik et al., 1983; Reznitskiy and Stilman, 1983; Stilman, 1985, 1993a).

3. Heuristic Search Efficiency

Here we discuss the parameters of the search and the criteria for evaluation of results. Such parameters of the system as number of agents, size of the space for their motions, and length of the variant-solution, can be considered as characteristics of the complexity of this class of problems. For example, the length of a solution usually predetermines the depth of the search tree which is necessary to generate and evaluate. Thus, if a 6-move search is required, we

have to generate a search tree of the depth 6. The question is: what is the "average breadth" of this tree, i.e., how many moves (on average) should be included into this tree at each node? For example, applying the brute force search algorithm, we have to include all the moves permitted in every state according to the problem statement. It means in this case we have to generate a search tree of the size:

$$B+B^2+...+B^L=T \quad (1)$$

where B is the average number of moves in each state, L is the depth of the search, and T is the total number of states generated. Following (Nilsson, 1980) parameter B is called a *branching factor*. The computation of B is based on the consideration of a hypothetical search tree with the depth of all branches equal to L, total number of moves equal to T , and a *constant* number of successors of each node. According to (1) this hypothetical constant number is equal to the branching factor B and might be computed as the solution of eq.(1) relative to B. Big values of B correspond to a non-selective search; obviously they indicate an exponential growth of the search (with a big base) as a function of the length of a solution. We look for approximate algorithms that reduce B, especially, those algorithms which make B close to 1. Such algorithms should be considered as extremely goal-driven with *minimal branching to different directions*.

4. Introduction to Linguistic Geometry

To discover the inner properties of human expert heuristics, which have been successful in a certain class of complex control systems, we develop a formal theory, the so-called *Linguistic Geometry* (Stilman, 1992-94). 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 by reducing the search. These *hierarchical images* in the form of networks of paths were extracted from the expert vision of the problem.

The hierarchy of subsystems is represented as a *hierarchy of formal attribute languages* where each "sentence" (a group of "words" or symbols) of the lower level language corresponds to the "word" of the higher level one. Following a linguistic approach each subsystem could be represented as a string of symbols with parameters: $\boldsymbol{a}(x_1)\boldsymbol{a}(x_2)...\boldsymbol{a}(x_n)$, where the values of the parameters incorporate the semantics of the problem domain or lower-level subsystems. The lowest-level language of the hierarchy of languages, the Language of Trajectories (Stilman, 1992b, 1992c, 1993a, 1993c), serves as a building block to create the upper-level languages, the Languages of Networks (Stilman, 1993b, 1993c, 1993f, 1994a, 1994b).

The Language of Trajectories actually is a formalization of the description of the set of various paths between different points of the complex control system. An element might follow a path to achieve the goal "connected with the ending point of this path." The Language of Networks is a formalization of a set of networks of certain paths unified by the mutual goal. For example, in the chess model such a network represents planning for a local fight, in the robot control model an analogous network of planning paths represents a draft short-range plan for approaching local goal in hazardous environment, i.e., getting over mobile and immobile obstacles. In the scheduling problem it corresponds to the maintenance schedule of a certain power unit including the schedule for the provision of resources required.

Network languages allow us to describe the "statics", i.e., the states of the System. In order to describe the "dynamics" of the System, i.e., the motions from one state to another, we have to regenerate the entire hierarchy of languages. Of course, it is an inefficient procedure. To improve the efficiency of applications in the search process it is important to describe the change of the hierarchy of languages (Stilman, 1994a). A study of this change helped us in modifying the hierarchy instead of regenerating it in each state. This change is represented as a mapping (translation) to some other hierarchy (actually, to the new state of the same hierarchy). Thus, the functioning of the system, in a search process, generates a tree of translations of the hierarchy of languages. This tree is represented as a string of the highest level formal language, the Language of Translations (Stilman, 1994b, 1994d).

5. Class of Problems

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 describing 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 transitions of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed from 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)^R_p(x,y),

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out with participation of one *or many elements* p from P_1 and P_2.

According to the 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 do not belong to the same one of the two 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 a robotic system can be considered as a Complex System (see below). Many different technical and human society systems (including military battlefield systems, systems of economic competition, positional games) that can be represented as twin sets of movable units (representing two or more opposing sides) and their locations 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 (Fikes and Nilsson, 1971), nonlinear planner NOAH (Sacerdoti, 1975), 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 finding an approximate solution of a reformulated problem.

6. Geometry of Complex Systems: Measurement of Distances

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 —> $\mathbf{Z_+}$ (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 $\{M^k_{x,p}\}$ of X (Fig.1)**:**

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 *i* less than *k*.

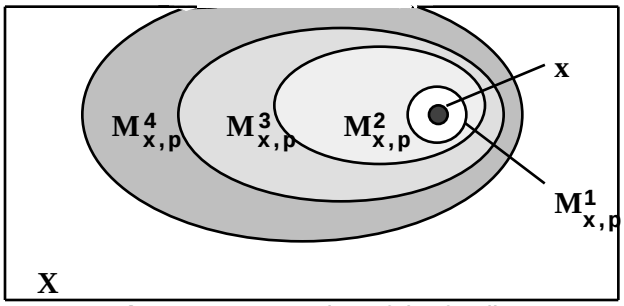

Fig. 1. Interpretation of the family of reachability areas

Let $MAP_{x,p}(y)=k$, for y from $M^k_{x,p}$ (*the number of steps from x to y*). For the remaining 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.

7. Set of Paths: Language of Trajectories

This language is a formal description of the set of lowest-level subsystems, the set of all 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 the y of X ($x \neq y$) with a length *l* is following formal string of symbols ***a***(x) with points of X as parameters:

$$t_0=\boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l),$$

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,…, *l*–1; element p stands at the point x: ON(p)=x. We denote by $t_p(x, y, l)$ the set of all trajectories for element p, beginning at x, end at y, and with length *l*. $\boldsymbol{P}(t_0)=\{x, x_1, ..., x_l\}$ is the set of parameter values of the trajectory t_0. (To avoid confusion we should emphasize that $\boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l)$ is a formal record and does not mean anything else except what is given above.)

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 us to define (in general form) and study formal grammars for generating the shortest trajectories. A general grammar and its application to generating the shortest trajectories for aerospace robotic vehicles are presented in (Stilman, 1993c).

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 that 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 length less than H. Different properties of this language and generating grammars were investigated in (Stilman, 1992b, 1992c, 1993a).

8. Networks of Paths: 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 (Botvinnik, 1984), these subsystems should be various types of trajectory networks, i.e., the sets of interconnected trajectories with one singled out and called the *main trajectory*. An example of such network is shown in Fig. 2. The basic idea behind these networks is as follows. Element p_0 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 opposing element). Naturally, the opposing 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 Fig. 2) and remove element p_0 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_0 at point 4. Similarly, element p_1 of the same side as p_0 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 opposing side to include the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of element q_1 to prevent this control.

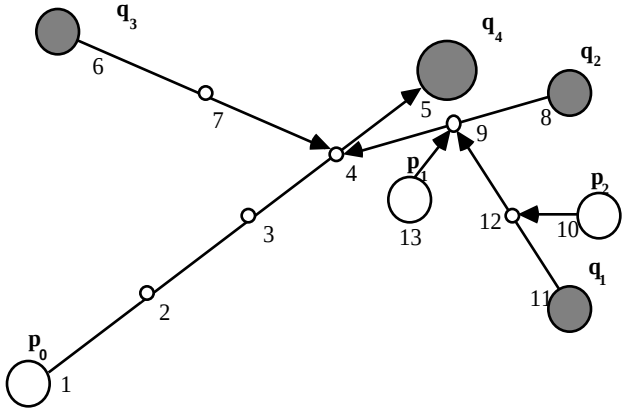

Fig. 2. Network language interpretation.

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 a 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 $\boldsymbol{P}(t_2)=\{y,y_1,\ldots,y_l\}$ of trajectory t_2 there is a value $y_i = x_k$, where $t_1=\boldsymbol{a}(x_0)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_k)$. 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 Fig. 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 ***a***(11)***a***(12)***a***(9) in Fig. 2 is connected degree 2 with trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5), i.e., C^2(***a***(11)***a***(12)***a***(9), ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5)) holds. Trajectory ***a***(10)***a***(12) in Fig. 2 is in transitive closure to the trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5) because C^3(***a***(10)***a***(12), ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5)) holds by means of the chain of trajectories ***a***(11)***a***(12)***a***(9) and ***a***(8)***a***(9)***a***(4).

A ***trajectory network*** **W** relative to trajectory t_0 is a finite set of trajectories $t_0,t_1,\ldots,t_k$ 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 $C_W^+(t_i,t_0)$ holds, i.e., each trajectory of the network W is connected with the trajectory t_0 that was singled out by a subset of interconnected trajectories of this network. If the relation $C_W^{\mathbf{m}}(t_i, t_0)$ holds, i.e., this is the **m**-th degree of connection, trajectory t_i is called the **m** ***negation trajectory***.

Obviously, the trajectories in Fig. 2 form a trajectory network relative to the main trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***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

$$t(t_1, param)t(t_2, param)\ldots t(t_m, param),$$

where *param* in parentheses substitute for the other parameters of a particular language. All the symbols of the string $t_1, t_2,\ldots, t_m$ 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 such language is the language that describes specific networks called Zones. They play the main role in the model considered here (Botvinnik, 1984; Stilman, 1977, 1993b, 1993c, 1994a). 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 (Stilman, 1993b, 1993c,1994a). 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

$$Z=t(p_0,t_0,t_0)\,t(p_1,t_1,t_1)\ldots t(p_k,t_k,t_k),$$

where $t_0,t_1,\ldots,t_k$ are the trajectories of elements $p_0,p_2,\ldots,p_k$ respectively; $t_0,t_1,\ldots,t_k$ are nonnegative integers that "denote the time allotted 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 opposing side. Trajectory $t(p_0,t_0,t_0)$ 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_0 and q belong to the opposing sides.

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

$$Z=t(p_0,a(1)a(2)a(3)a(4)a(5),4)t(q_3,a(6)a(7)a(4),3)$$
$$t(q_2, a(8)a(9)a(4), 3)t(p_1, a(13)a(9), 1)$$
$$t(q_1, a(11)a(12)a(9), 3)\ t(p_2, a(10)a(12), 1)$$

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_0 starts the motion to the target, while black starts in its turn to move elements q_2 or q_3 to intercept element p_0. 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* t) *allocated to it.* For example, the motion along the trajectories ***a***(6)***a***(7)***a***(4) and ***a***(8)***a***(9)***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_0 assuming one would go along the main trajectory without move omission and all the intercepting elements will move *simultaneously* (if necessary). According to definition of Zone, the trajectories of white elements (except p_0) could only be of the length 1, e.g., ***a***(13)***a***(9) or ***a***(10)***a***(12). As element p_1 can intercept the motion of the element q_2 at the point 9, black includes into the Zone the trajectory ***a***(11)***a***(12)***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 the main trajectory is determined by the number of that point. For example, for the point 4, it equals 3 time intervals.

A language $L_Z^H(S)$ generated by the certain grammar $\mathbf{G_Z}$ (Stilman, 1993b, 1993c,1994a) in a state S of a Complex System is called the ***Language of Zones***.

A practicality of the formal constructions considered in Section 4 as well as the entire hierarchy of languages are demonstrated on the following 2D example of the air combat.

9. Air Combat: Problem Statement.

The robotic model can be represented as a Complex System naturally (Fig. 3). The set X represents the operational district, which could be the area of combat operation, broken into smaller square or cubic areas, "points", e.g., in the form of the big square or cubic grid. 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; $R_p(x,y)$ represent

moving capabilities of different robots for different problem domains: robot p can move from point x to point y if $R_p(x, y)$ holds. Some of the robots can crawl, others 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 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.

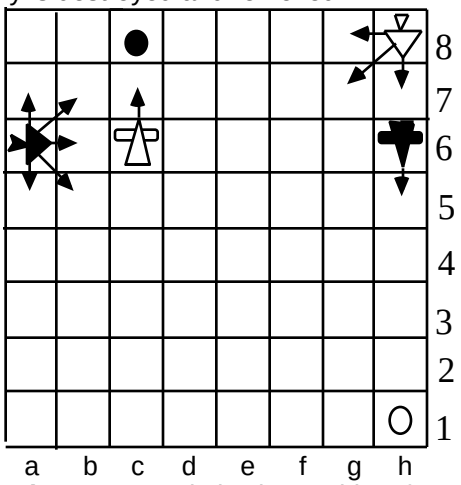

Fig. 3. 2D optimization problem for autonomous robotic vehicles.

Robots with different moving capabilities are shown in Fig. 3. The operational district X is the table 8 x 8. Robot W-FIGHTER (White Fighter) standing on h8, can move to any next square (shown by arrows). The other robot B-BOMBER (Black Bomber) from h6 can move only straight ahead, one square at a time, e.g., from h6 to h5, from h5 to h4, 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 (White Bomber) standing on c6 is analogous with the robot B-BOMBER; it can move only straight ahead but in reverse direction. Thus, robot W-FIGHTER on h8 can reach any of the points y Œ{h7, g7, g8} in one step, i.e., $R_{W\text{-}FIGHTER}(h8, y)$ holds, while W-BOMBER can reach only c7 in one step.

Assume that robots W-FIGHTER and W-BOMBER belong to one side, while B-FIGHTER and B-BOMBER belong to the opposing side: W-FIGHTER Œ P_1, W-BOMBER Œ P_1, B-FIGHTER Œ P_2, B-BOMBER Œ 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 Œ P_2. Each of the BOMBERs can destroy unmoving TARGET ahead of the course; it also has powerful weapons able 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 able to destroy an opposing BOMBER approaching its location, but it also able 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 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 Fig. 3.

Assume that motions of the opposing sides alternate and each side can participate in both operations ***simultaneously***. It means, for example, that during the current time interval, in case of White turn, both W-BOMBER and W-FIGHTER, one of them, or none can move. Analogous condition holds for Black. There is one exception. If W-FIGHTER hits B-BOMBER while the latter is fully armed, i.e., it is not at its final destination – square h1, W-BOMBER can not move simultaneously during this time interval in order to avoid possible consequences of the B-BOMBER explosion. Similar restriction holds for B-BOMBER: it can not move at the moment when W-BOMBER is destroyed (not at c8).

In this paper we relaxed the most restrictive requirement of participation of the only element in each motion which holds in our earlier papers for the similar problems (Stilman, 1994b, 1994d). This requirement was imposed in the form of the restriction for each side to participate in both local operations

simultaneously. It was inherited from the original domain, the game of chess, served as a testbed for the development of the Linguistic Geometry tools (Stilman, 1977; Botvinnik, 1984).

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-BOMBER from h6 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. Theoretical evaluations (Nilsson, 1980) of the lower bounds of complexity of this algorithm for the air combat problem show that, at best, it would result in a search tree of 25 million moves (transitions). In practice, even this number is unreachable. It is very interesting to observe the dramatic reduction of search employing Linguistic Geometry tools.

To demonstrate generation of the Hierarchy of Languages for this problem, we have to generate the Language of Trajectories and the Language of Zones in each state of the search. The details of trajectory generation and generation of different Zones are considered in (Stilman, 1993b, 1993c, 1993d, 1994a).

10. Air Combat: Search Generation

Consider how the hierarchy of languages works for the optimal control of the Robotic System introduced above (Fig. 3). We generate the string of the Language of Translations (Stilman, 1994a) representing it as a conventional search tree (Fig. 4) and comment on its generation.

In fact, this tree is close to the search tree of the restricted problem (Stilman, 1994b, 1994d). In our comments on this generation we will emphasize the major steps.

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 Fig. 5. 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:

$$Z_{WB}=\boldsymbol{t}(\text{WB}, \boldsymbol{a}(\text{c6})\boldsymbol{a}(\text{c7})\boldsymbol{a}(\text{c8}), 2)$$
$$\boldsymbol{t}(\text{BF},\boldsymbol{a}(\text{a6})\boldsymbol{a}(\text{b7})\boldsymbol{a}(\text{c8}), 3)\boldsymbol{t}(\text{BF},\boldsymbol{a}(\text{a6})\boldsymbol{a}(\text{b7})\boldsymbol{a}(\text{c7}), 2)$$
$$\boldsymbol{t}(\text{WB}, \boldsymbol{a}(\text{c6})\boldsymbol{a}(\text{b7}), 1)$$

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

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.

Next move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7:c8 (Fig. 6, 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. This procedure determined that W-FIGHTER is out of the Zone of B-BOMBER, thus it can not intercept B-BOMBER.

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" Zones with the main trajectory from h8 to c8. These Zones have been detected (within the horizon 5) in the terminal state after the move 2. ... b7:c8 Fig. 6 (left). Obviously they could not be detected in the initial state of this problem (Fig. 5) because the main element, W-BOMBER, could not "see" the target, B-FIGHTER, within given horizon. However, at the moment of detection it was too late to include them into the search. These Zones have been stored for possible activation at the higher levels of the search tree. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in Fig. 6 (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 h6. The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously.

The generation continues with the simultaneous motion of two agents, the double move, W-FIGHTER and W-BOMBER in their respective Zones: 2. h8-g7/c7-c8. The B-FIGHTER intercepts W-BOMBER at c8: 2. ... b7:c8. Now W-FIGHTER is ready to destroy B-BOMBER moving along the attacking trajectory a(7)a(h6): 3. g7:h6.

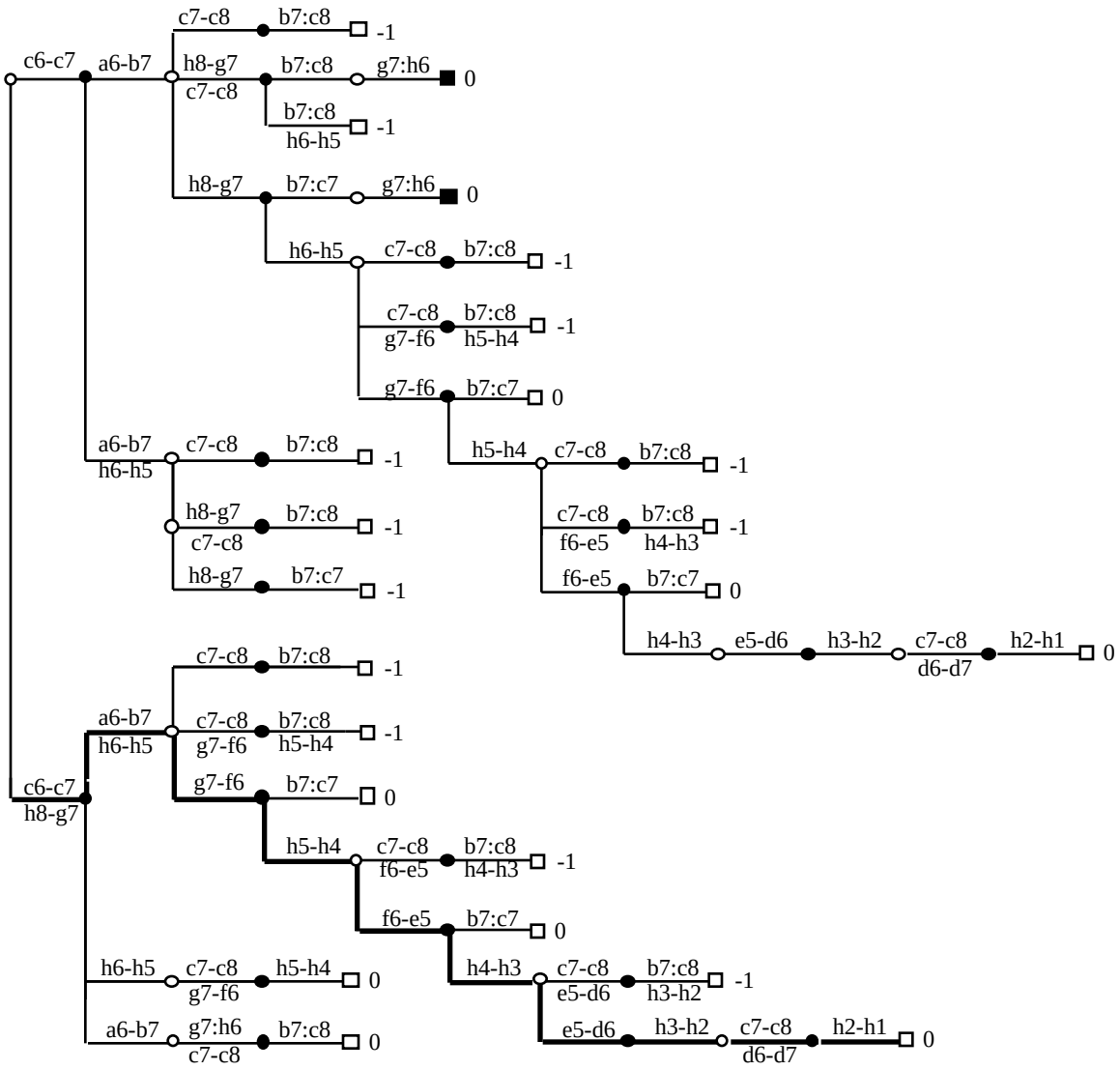

Fig. 4. Search tree for the 2D optimization problem for robotic vehicles.

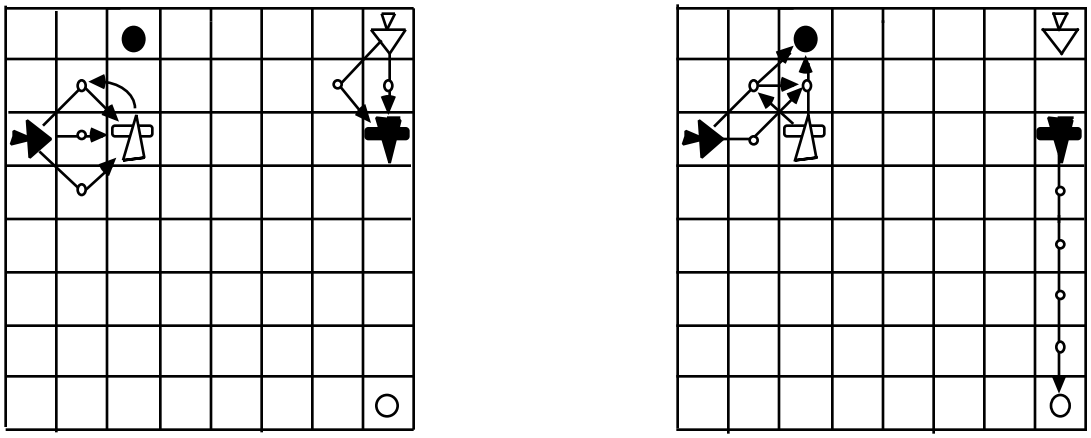

Fig. 5. Interpretation of the Zones in the initial state of the 2D Robot Control Model.

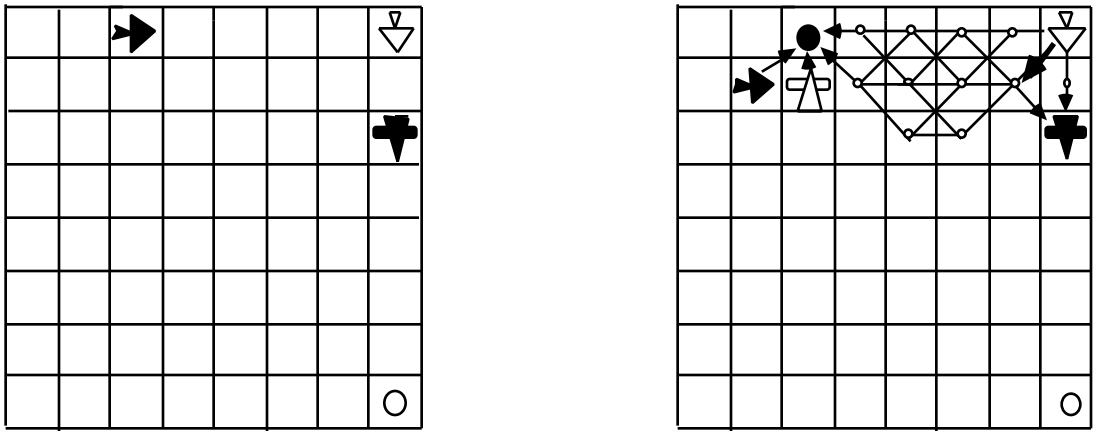

Fig. 6. States where the control Zone from h8 to c8 was detected (left) and where it was included into the search (right)

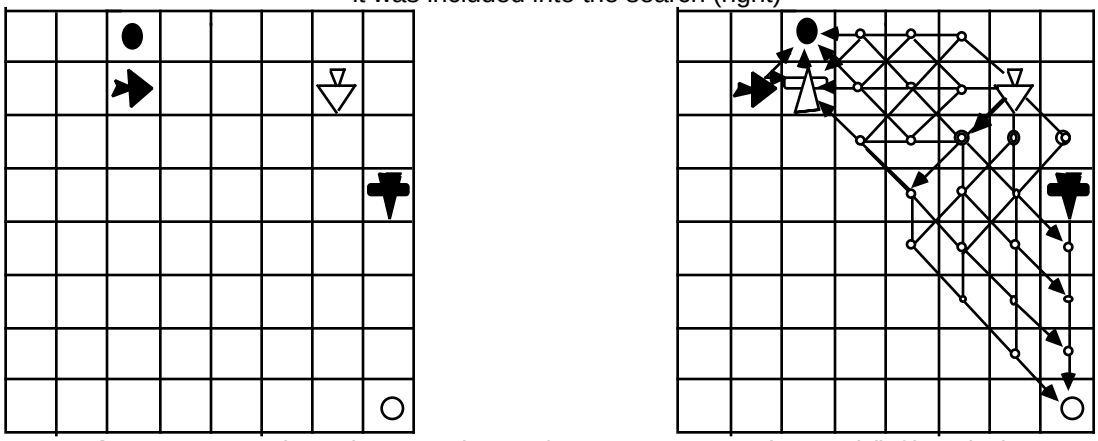

Fig. 7. States where the control Zone from g7 to c7 was detected (left) and where it was included into the search (right).

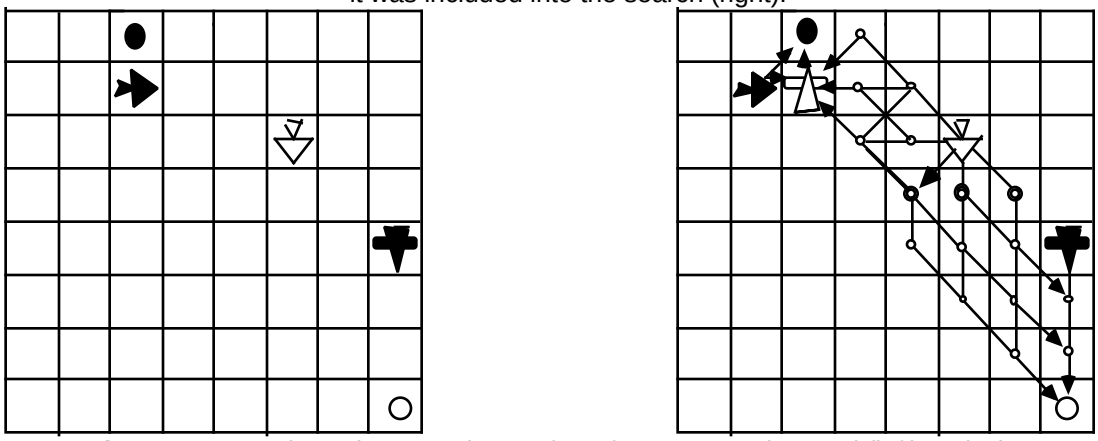

Fig. 8. States where the control Zone from f6 to c7 was detected (left) and where it was included into the search (right).

In this state all the BOMBERs have been destroyed and the grammar evaluates it as a draw (0) and initiates the backtracking climb. Move 2. ... b7:c8 is changed for the double move 2. ... b7:c8/h6-h5. That way the grammar included motion in the Zone of B-BOMBER attacking unmovable target on h1. The “square rule” procedure terminated branch, evaluated it as a win (-1) for the Black side, and initiated backtracking climb. 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 c8. 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, and now the grammar includes this move as a single one 2. h8-g7 postponing motion in the Zone of W-BOMBER. In this state (Fig. 7, left) a set of new control Zones of W-FIGHTER from g7 to c7 have been detected and

stored as idle to be activated later if necessary. In response, Black hit W-BOMBER at c7: 2. ... b7:c7 along the intercepting trajectory a(b7) a(c7). W-FIGHTER continues its attack of B-BOMBER: 3. g7:h6. This state is evaluated as a draw.

After the cut and climb, the inspection procedure included motion in the Zone of B-BOMBER instead of 2. ... b7:c7. New move 2. ... h6-h5 can not be included as a double move simultaneously with the old one because B-BOMBER can not move at the moment when W-BOMBER is being destroyed not at its final destination (c8). The following motion in 3. c7-c8 b7:c8 resulted in the termination of this branch with the value -1 in favor of Black.

New climb up to the move 2. ... h6-h5 and execution of the inspection procedure result in the inclusion of the groups of new control Zones from g7 to c7 and to c8 in order to improve the exchanges at these locations. Both groups of Zones (to c7 and c8) have been detected earlier in the search tree. The set of Zones with different main trajectories from g7 to c7 and from g7 to c8 is shown in Fig. 7 (right). Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same Fig. 7. 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 on h5. Speaking informally, from squares f6, g6, and h6, Zone gateways, W-FIGHTER can intercept B-BOMBER (in case of White turn). 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 time", i.e., to approach two goals simultaneously. Thus, the double move 3. c7-c8/g7-f6 is included. After the response 3. ... b7:c8/h5-h4, the branch was terminated with the value -1. The following climb and branching with inclusion of 3. g7-g6 as a single move resulted in 3. ... b7:c7 with terminal value 0. This state is shown in Fig. 8, left. The draw value has been assigned by the "square rule" procedure which detected that W-FIGHTER is in the Zone of B-BOMBER and thus has enough time for interception.

After the climb Black side continued branching 3. ... h5-h4. The following tree generation until 4. f6-e5 is analogous with the previous one after 2. ... h6-h5. Move 4. f6-e5 is selected by the move ordering procedure as the time-gaining move approaching two goals simultaneously, c7 as a goal of the control Zone of W-FIGHTER and one of the gateways (e5, f5, g5) of the Zone of B-BOMBER (Fig. 8, right).

After the change of 4. ... b7:c7 for 4. ... h4-h3, the following branch is pretty straightforward. After 6. c7-c8/d6-d7 h2-h1, it is terminated with the value of 0. The following climb with activation of the inspection procedure in every black node ended at 1. ... a6-b7 which has been changed for the double move 1. ... a6-b7/h6-h5. It seems that this move almost depreciated previous search. The minimax value brought to the top of the tree generated so far is -1. However, the tree generation followed after the change of 1. c6-c7 for the double move 1. c6-c7/h8-g7 showed that previous search was very important. As a result of this search the grammar *learned key networks*, Zones of W-FIGHTER with main trajectories from g8 to c8, from g7 to c7 and c8, and from f6 to c7. These networks have been used successfully in a different context after 1. c6-c7/h8-g7. The optimal branch is shown in Fig. 4 with bold lines.

The total number of moves included into this tree is 63. The maximum depth reached is 12. This means that the branching factor (Nilsson, 1980) of this tree is 1.24, i.e., the search is highly goal-oriented. The unreduced branching factor for this problem is about 17, taking into account the allowance of simultaneous moves. Obviously, 63 is a drastic reduction in comparison with a 17^{12} move trees that would have to be generated by conventional search procedures, or even with the theoretical minimum of the minimax search with alpha-beta cut-offs $(17^{12})^{1/2}= 17^{6}$ ª (25 million).

11. Discussion

Example considered in Sections 9, 10 demonstrates 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 simplified aerospace robotic vehicles. It is more interesting that search reduction achieved in the original domain with one-at-a-time motion of every agent (see, for example, Stilman, 1994b) multiplied tremendously in the new domain with the allowance of concurrent moves. While the total number of moves to be included in the search tree in order to solve new problem (17^{12}) is by far greater than a million required for the original problem, the actual number of moves generated by the grammar of searches is almost the same (about 60), and the branching factor even decreased from 1.65 to 1.24. Looking at the complexity of the hierarchy of languages which represents each state in the search process, it is very likely that the growth from the serial motion case to the concurrent one is limited by multiplication to a constant factor close to one. This means that the complexity of the entire algorithm did not change after allowing concurrent moves, and this allowance is, probably, inherent to the Complex Systems and the entire Hierarchy of Languages.

At the same time the simplified air combat problem considered here is still very close to the original chess domain. It is possible to predict that the power of Linguistic Geometry goes far beyond these limits. The definition of the Complex System (see Section 5) is generic enough to cover a variety of different problem

domains. The core component of this definition is the triple X, P, and R_p. Thus, looking at the new problem domain we have to define X, the finite set of points – locations of elements. We do not impose any constraints on this set while the aerospace operational district X considered in this paper as well as the original chess board have different extra features, e.g., 2D space connectivity, which is totally unimportant for these problems. Thus, we can consider X, for example, as a set of orbits where the elements are in constant motion with respect to each other. The set of elements P, e.g., movable units, in our problem is quite small, while their moving capabilities, binary relations of R_p, are non-sophisticated. Indeed, during a one time interval our aircrafts 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.

The development of Linguistic Geometry toward aerospace applications will encompass the discovery of geometrical properties of subsystems, details of interactions between the elements within subsystems and between different subsystems, the effect of this complex hierarchical structure on the search reduction.

References

Albus, J. (1991). Outline for a Theory of Intelligence. *IEEE Trans. on Systems, Man and Cybernetics* (pp. 473-509), 3.

Boddy, M. and Dean, T. (1989). Solving Time-Dependent Planning Problems, *Proc. of the 11th Int. Joint Conf. on AI*, (pp. 979-984).

Botvinnik, M.M. (1984). *Computers in Chess: Solving Inexact Search Problems.* Springer Series in Symbolic Computation, New York: Springer-Verlag.

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* (pp. 1030-1041), 6, (in Russian).

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence* 32(3).

Chomsky, N. (1963). Formal Properties of Grammars. in *Handbook of Mathematical Psychology*, eds. R.Luce, R.Bush, E. Galanter., vol. 2 (pp. 323-418). New York: John Wiley & Sons.

Chung, J., Liu, J. and Lin, K., (1990). Scheduling Periodic Jobs That Allow Imprecise Results. *IEEE Transactions on Computers*, (pp. 1156-1174), 39(9).

Drabble, B., (1991) Spacecraft Command and Control Using Artificial Intelligence Techniques, *J. of the British Interplanetary Society*, Vol. 44, (251-254).

Durfee, E.H. and Lesser, V.R. (1987). Incremental Planning to Control a Blackboard-Based Problem Solver. *Proceedings of the Workshop on Space Telerobotics*, Vol. 3, NASA Jet Propulsion Lab, July, (pp. 91-99).

Feder, J. (1971). Plex languages. *Information Sciences*, 3: 225–241.

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.

Fu, K.S. (1982). *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs.

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.

Giarratano, J.C., (1991). CLIPS User's Guide, NASA Johnson Space Center, Information Systems Directorate (JSC-25013).

Ginsburg, S. (1966). *The Mathematical Theory of Context-Free Languages*, McGraw Hill, New York.

King, R. D. (1993). Rule Based Approach to Hierarchical Grammars for Geometrical Reasoning, Master Thesis, Dept. of Computer Science, Univ. of Colorado at Denver.

Knoblock, C.A. (1990). Learning Abstraction Hierarchies for Problem Solving, *Proc. of the 8th AAAI Conf.*, (pp.923-928), Menlo Park, CA.

Knuth, D.E. (1968). Semantics of Context-Free Languages. *Mathematical Systems Theory*, (pp. 127–146), 2.

Korf, R.E. (1990). Real-Time Heuristic Search, *Artificial Intelligence*, (pp. 189-211), 42(2-3).

Leitmann, G., (1990). *Optimization Techniques with Applications to Aerospace Systems*, Academic Press.

Lesser, V.R., Pavlin, J., and Durfee, E. (1988). Approximate Processing in Real-Time Problem Solving, *AI Magazine*, (pp. 49-62), 9(1).

Lirov Y., Rodin, E.Y., McElhaney, B.G., and Wilbur, L.W. (1988). Artificial Intelligence Modeling of Control Systems, *Simulation*, (pp. 12-24), 50(1).

Mathews, E. R. (1993). An Implementation of the Grammars of Trajectories and Zones in C Language, Master Thesis, Dept. of Mathematics, Univ. of Colorado at Denver.

McAllester, D. and Rosenblitt, D. (1991). Systematic Non-Linear Planning, *Proc. of AAAI-91*, (pp. 634-639).

McCarthy, J. (1980). Circumscription–A Form of Non-Monotonic Reasoning. *Artificial Intelligence* , (pp. 27-39), 13.

McCarthy, J. and Hayes, P.J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence* (pp. 463–502), 4.

Mesarovich, M.D. and Takahara Y. (1989). *Abstract Systems Theory*, Berlin: Springer-Verlag.

Narasimhan, R.N. (1966). Syntax–Directed

Interpretation of Classes of Pictures. *Comm. of ACM* (pp. 166–173), 9.

Nilsson, N.J. (1980). *Principles of Artificial Intelligence*, Palo Alto, CA: Tioga Publ.

Pavlidis, T. (1977). *Structural Pattern Recognition*, New York: Springer-Verlag.

Pigeon, A., Howard, G., and Seaton B. (1992) Operational Aspects of Space craft Autonomy, *J. of the British Interplanetary Society*, Vol. 45, (pp. 87-92).

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* (pp. 147-153), 11, (in Russian).

Rodin E. (1988). Semantic Control Theory, *Applied Mathematical Letters*, (pp. 73-78), 1(1).

Rodin E., Garsia-Ortiz et al. (1993). Application of Semantic Control to a Class of Pursue-Evader Problems, *Computers and Mathematics with Applications*, 26(5).

Rosenfeld, A. (1979). *Picture Languages, Formal Models for Picture Recognition*, Academic Press.

Rozenkrantz, D.J. (1969). Programmed Grammars and Classes of Formal Languages, *J. of the ACM* (pp. 107–131), 1.

Sacerdoti, E.D. (1975). The Nonlinear Nature of Plans, *Proc. Int. Joint Conference on Artificial Intelligence*.

Shaw, A.C. (1969). A Formal Picture Description Scheme as a Basis for Picture Processing System, *Information and Control* (pp.9-52), 19.

Simon, H.A. (1980). *The Sciences of the Artificial*, 2-nd ed., The MIT Press, Cambridge, MA.

Stefik, M. (1981). Planning and meta-planning (MOLGEN: Part 2),"*Artificial Intelligence* (pp. 141-169), 2.

Stilman, B. (1977). The Computer Learns. in Levy, D., *1976 US Computer Chess Championship* (pp. 83-90). Computer Science Press, Woodland Hills, CA.

Stilman, B. (1985). Hierarchy of Formal Grammars for Solving Search Problems. In *Artificial Intelligence. Results and Prospects*, *Proceedings of the International Workshop* (pp. 63-72), Moscow, (in Russian).

Stilman, B. (1992a). A Syntactic Structure for Complex Systems. *Proc. Second Golden West Int. Conf. on Intelligent Systems* (pp. 269-274), Reno, NE, June.

Stilman, B. (1992b). A Geometry of Hierarchical Systems: Generating Techniques. *Proc. Ninth Israeli Conference on Artificial Intelligence and Computer Vision* (95-109), Tel Aviv, Israel, Dec.

Stilman, B. (1992c). A Syntactic Approach to Geometric Reasoning about Complex Systems," *Proc. Fifth Int. Symp. on Artificial Intelligence* (pp. 115-124), Cancun, Mexico, Dec.

Stilman, B. (1993a). A Linguistic Approach to Geometric Reasoning," *Int. J. Computers and Mathematics with Applications* (pp. 29-57), 26(7).

Stilman, B. (1993b). Network Languages for Complex Systems, *Int. J. Computers and Mathematics with Applications* (51-79), 26(8).

Stilman, B. (1993c). Syntactic Hierarchy for Robotic Systems, *Integrated Computer-Aided Engineering* (pp. 57-81), 1(1).

Stilman, B. (1993d). A Formal Language for Hierarchical Systems Control, *Languages of Design* (pp. 333-356), 1(4).

Stilman, B., (1993e). Hierarchical Network for Systems Control. *Proc. of the Tenth Israeli Symposium on Artificial Intelligence and Computer Vision* (pp. 141-153), Ramat Gan, Israel, Dec.

Stilman, B., (1993f). Knowledge Representation in Linguistic Geometry. *Proc. of the Fifth Int. UNB Artificial Intelligence Symposium* (pp. 219-229), Fredericton, New Brunswick, Canada, August.

Stilman, B. (1994a). Translations of Network Languages. *Int. J. Computers and Mathematics with Applications* (pp. 65-98), 27(2).

Stilman, B. (1994b). A Formal Model for Heuristic Search. *Proc. of the 22nd Annual ACM Computer Science Conf.*, (pp. 380-389), March 8-10, Phoenix, AZ.

Stilman, B., (1994c). A Linguistic Geometry for Intelligent Autonomous Systems. *Cybernetics and Systems-94*. *Proc. of the Twelfth European Meeting on Cybernetics and Systems Research* (pp. 1483-1490), Vienna, Austria, April.

Stilman, B., (1994d). A Linguistic Geometry for Space Applications, *Proc. of the 1994 Goddard Conference on Space Applications of Artificial Intelligence*, pp. 87-101, NASA Goddard Space Flight Center, Greenbelt, MD, USA, May 1994.

Stilman, B.(1994e).Heuristic Networks for Space Exploration, *Telematics and Informatics, An Int. Journal on Telecommunications & Information Technology*, 11(4), (to appear).

Stilman, B., (1994f). A Linguistic Geometry of the Chess Model, *Advances in Computer Chess 7*, (pp. 91-117), (to appear).

Stilman, B., (1994g). Network Languages for Intelligent Control, *An International Journal: Computers & Mathematics with Applications*., (to appear).

Strosnider, J.K. and Paul, C.J. (1994). A Structured View of Real-Time Problem Solving, *AI Magazine*, (pp. 45-66), 15(2).

Volchenkov, N.G. (1979). The Interpreter of Context-Free Controlled Parameter Programmed Grammars. In L.T. Kuzin, Eds., *Cybernetics Problems. Intellectual Data Banks* (147–157), USSR Academy of Sci., Moscow, (in Russian).