A Formal Model for Heuristic Search
(DRAFT)

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

Abstract: We develop a formal theory, the so-called Linguistic Geometry, 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. 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 an informal survey of the Linguistic Geometry, major formal issues, and a comprehensive example of a solution of optimization problem for military autonomous agents. This example includes actual generation of the hierarchy of languages and demonstrates the drastic reduction of search in comparison with conventional search algorithms.

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 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 [35].

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* [28-34]. 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, 33].

1 Complex System

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

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

where $\mathbf{X}=\{x_i\}$ is a finite set of *points*; $\mathbf{P}=\{p_i\}$ is a finite set of *elements*; **P** is a union of two non-intersecting subsets P_1 and P_2; $\mathbf{R_p(x, y)}$ 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; $\mathbf{S_i}$ and $\mathbf{S_t}$ 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)^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 opposite 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.

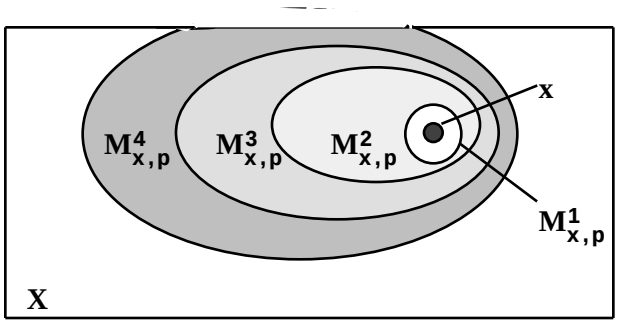
We devote ourselves to the search for an approximate solution of a reformulated problem.

2 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 of X $\{M^k_{x,p}\}$ (Fig. 1):

Fig. 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.

3 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 the y of X ($x \neq y$) with a length l is a following string of symbols with parameters, points of X: $t_o=\boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l)$, 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 $t_p(x, y, l)$ the set of trajectories in which p, x, y, and l are the same. $P(t_o)=\{x, x_1, \ldots, x_l\}$ is the set of parameter values of the trajectory t_o. 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.

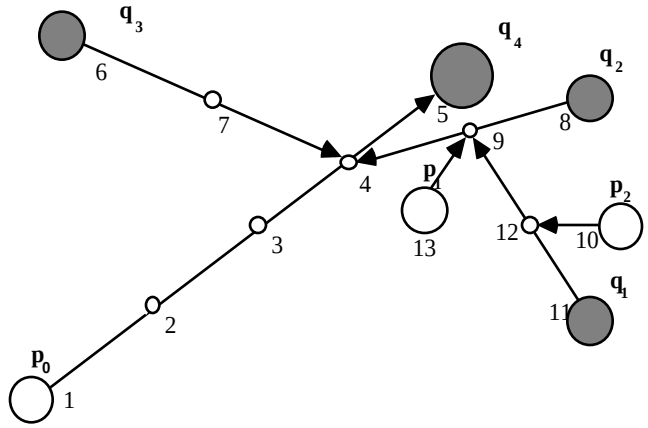
Reasoning informally, an analogy can be set up: the shortest trajectory is analogous with a straight line segment connecting two points in a plane. An analogy to a k-element segmented line connecting these points is called an ***admissible trajectory of degree k***, i.e., the trajectory which can be 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].

4 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 Fig. 2. 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 Fig. 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.

Fig. 2. A 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 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 Fig. 2 the trajectories ***a***(6)***a***(7)***a***(4) and ***a***(8)***a***(9)***a***(4) are connected with the main trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5) through point 4. Trajectories ***a***(13)***a***(9) and ***a***(11)***a***(12)***a***(9) are connected with ***a***(8)***a***(9)***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_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 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 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, 34]. 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 [33, 34]. 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 Fig. 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 ***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_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., ***a***(13)***a***(9) or ***a***(10)***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 ***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 main trajectory is determined by the number of that point. For example, for the point 4 it equals 3 time intervals.

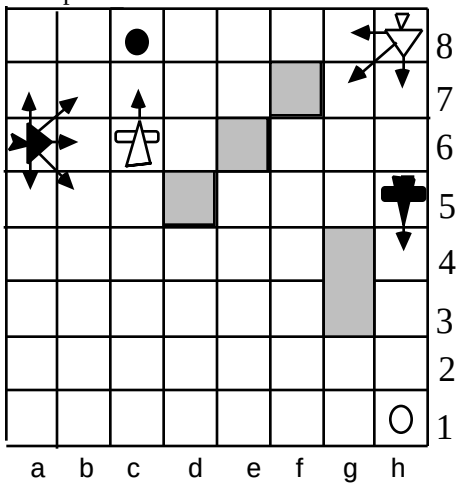
A language $L_Z(S)$ generated by the certain grammar $\mathbf{G_Z}$ [33, 34] in a state S of a Complex System is called the ***Language of Zones***.

Network languages allow us to describe the "statics", i.e., the states of the System. We proceed with the 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].

5 Robot Control Model as Complex System.

Such model can be represented as a Complex System naturally (Fig. 3). A set of **X** represents the operational district which could be the area of combat operation broken into squares, e.g., in the form of the table 8 x 8, n = 64. 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; $\mathbf{R_p(x,y)}$ 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; $\mathbf{S_i}$ is an arbitrary initial state of operation for analysis, or the starting state; $\mathbf{S_t}$ is the set of target states. These might be the states where robots of each sidereached 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.

Fig. 3. A 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. 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-BOMMER; 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 revenge 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 Fig. 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. Is there a strategy for the White side to make a draw?

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*U*Pred*U*Con*U*Var*U*Func*U{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, $D(f)=\mathbf{Z}_+\backslash\{0\}$ ($next_i$ is defined lower)

E =$\mathbf{Z}_+$ U X U P is the subject domain;

Parm: $S \to Var$, $A \to Var$, $a \to \{x\}$, 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.

next_i is defined as follows:

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

$SUM=\{v \mid v \in X, MAP_{xo,p}(v)+MAP_{yo,p}(v)=l_o\}$

$ST_k(x)=\{v \mid v$ 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

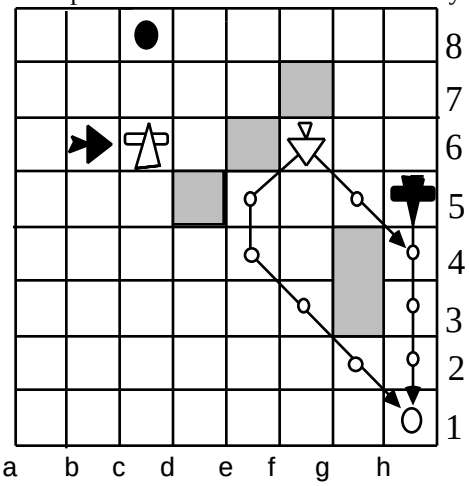
$next_i(x, l)=m_i$ for $i \leq r$;

$next_i(x, l)=m_r$ for $r<i \leq n$,

otherwise

$next_i(x,l)$=x.

Fig. 4. Interpretation of Zone for the Robotic System.



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. 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 (Fig. 4). (This is the location of W-FIGHTER in one of the states of the System in the process of the search.)

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 l to rewrite a symbol A. We choose the leftmost entry of symbol A in the current string and compute the value of predicate Q, the condition of applicability of the production. If the current string *does not* contain A or Q =F, then the application of the production is ended, and the next production is chosen from the failure section F_F ; F_F becomes the current permissible set. If the current string *does* contain the symbol A and Q=T , 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 π_n) or corresponding to the parameters of the symbols (π_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 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. Values of $MAP_{F6,W\text{-}FIGHTER}$ are shown in Fig.5.

Fig.5. $MAP_{f6,FIGHTER}$

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

Fig.6. $MAP_{h1,W-FIGHTER}$

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

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

$\boldsymbol{S}$(f6, h1, 5) $^{1}\!=\!>\boldsymbol{A}$(f6, h1, 5) $^{2_1}\!=\!>\boldsymbol{a}$(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 $MAP_{f6,W-FIGHTER}$ and $MAP_{h1,W-FIGHTER}$ (shown in Fig. 6) on X. Adding these tables (Fig. 5 and Fig. 6) as matrices we compute

SUM ={v | v X, $MAP_{f6, W-FIGHTER}$(v) + $MAP_{h1,W-FIGHTER}$(v) = 5} (Fig. 7).

Fig. 7. SUM.

					5		
				5	5	5	
				5	5		5
					5		5
						5	5
							5

Fig. 8. ST_1(f6).

The next step is the computation of ST_1(f6)={v | v from X, $MAP_{f6,W-FIGHTER}$(v)=1} which can be found in Fig. 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) $^{2_1}\!=\!>\boldsymbol{a}$(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 | v X, $MAP_{e5,W-FIGHTER}$(v)=1},(Fig. 9)

ST_{5-4+1}(f6) = ST_2(f6) = {v |v X, $MAP_{f6,W-FIGHTER}$(v)=2}, (Fig. 10).

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 Fig. 7, 9, 10; $MOVE_4$(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.

Fig. 9. ST_1(e5)

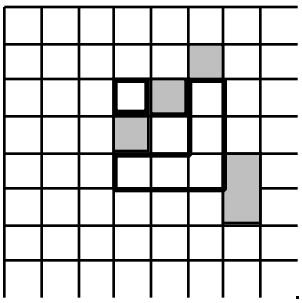
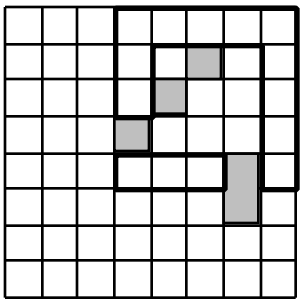


Fig. 10. ST_2(f6).



Let us proceed with the first one: $\boldsymbol{a}$(f6)$\boldsymbol{a}$(e5)$\boldsymbol{A}$(e4, h1, 3) $^{2_1}\!=\!>$ ... 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 Fig. 4 is as follows:

$\boldsymbol{t}$(B-BOMBER,t_B,5)$\boldsymbol{t}$(W-FIGHTER,t_F,5)$\boldsymbol{t}$(W-FIGHTER, $t_F^{\,1}$,2), where t_B=$\boldsymbol{a}$(h5)$\boldsymbol{a}$(h4)$\boldsymbol{a}$(h3)$\boldsymbol{a}$(h2)$\boldsymbol{a}$(h1), t_F=$\boldsymbol{a}$(f6)$\boldsymbol{a}$(e5)$\boldsymbol{a}$(e4)$\boldsymbol{a}$(f3)$\boldsymbol{a}$(g2)$\boldsymbol{a}$(h1), $t_F^{\,1}$=$\boldsymbol{a}$(f6)$\boldsymbol{a}$(g5)$\boldsymbol{a}$(h4)

The details of generation of different Zones are considered in [33, 34].

6 Search generation for Robotic System

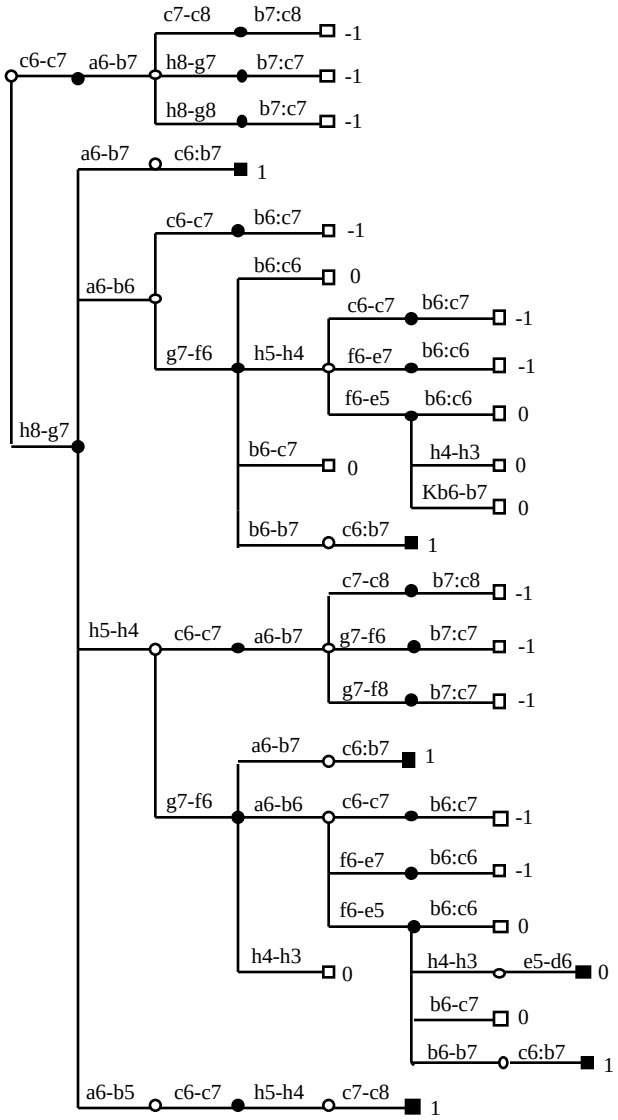
Consider how the hierarchy of languages works for the optimal control of the Robotic System introduced above (Fig. 3). We generate the search of the Language of Translations representing it as a conventional search tree (Fig. 11) and comment on its generation. In fact, this tree is very close to the 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].

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. All the Zones generated in the start state are shown in Fig. 12. Zones for FIGHTERs as attacking elements are shown in the top diagram, while Zones for BOMBERs – in the bottom one. For example, one of the Zones for W-BOMBER, Z_{WB} is as follows:

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

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

Fig. 11. Search tree for the optimization problem for robotic vehicles.

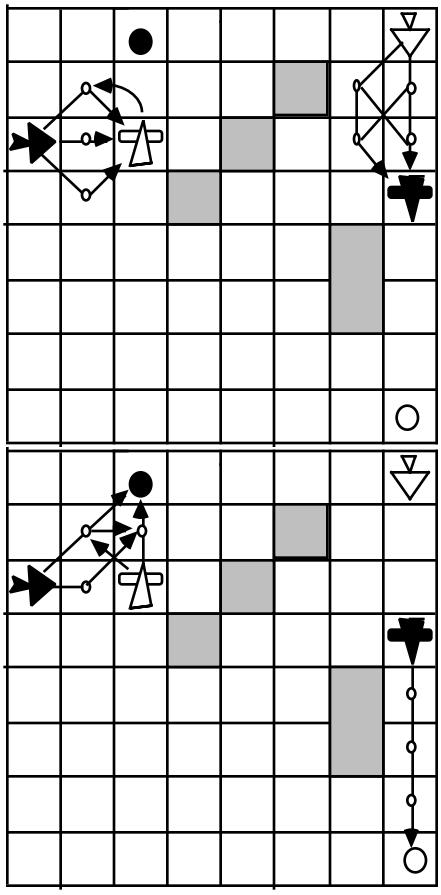


Generation begins with the move 1. c6-c7 in the "white" Zone with the target of the highest value and the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations. The computation of move-ordering constraints is the most sophisticated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

Next move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7:c8 (Fig. 13, top). 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.

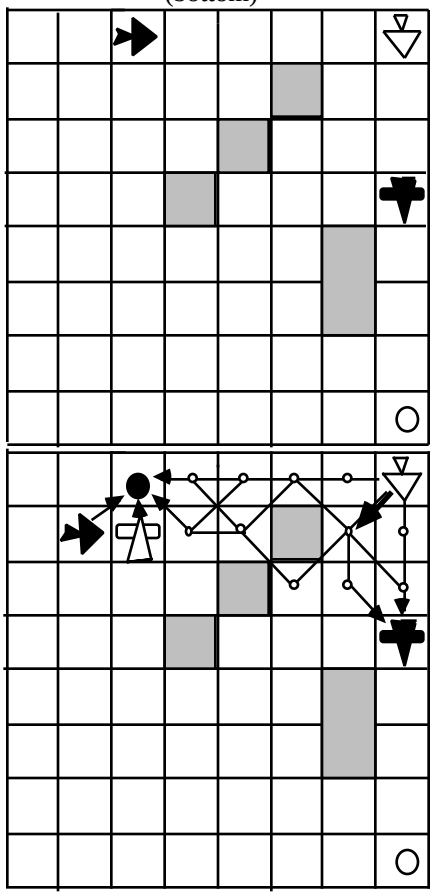
Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search. After climb up to the move 1. ... a6-b7, the tree to be analyzed consists of one branch (of two plies): 2. c7-c8 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 Fig. 13 (bottom). 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.

Fig. 12. An interpretation of the Zones in the initial state of the Robot Control Model.



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.

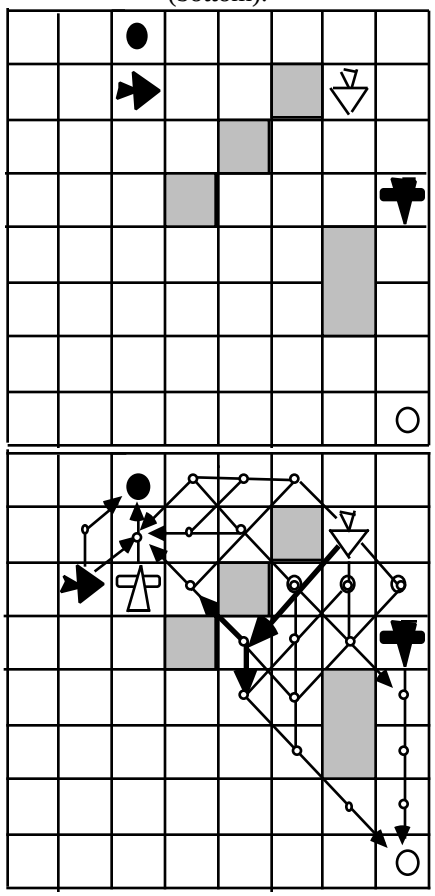
Fig. 13. States where control Zone from h8 to c8 was detected (top) and where it was included into the search (bottom)



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) (Fig. 12(bottom)). 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 Fig. 14(top). The "square rule procedure" cuts this branch and evaluates it as a win of the Black side.

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 Fig. 14 (bottom). Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same Fig. 14. 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.

Fig. 14. States where control Zone from g7 to c7 was detected (top) and where it was included into the search (bottom).



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.

7 Discussion

The approach to understanding of dynamic hierarchical systems considered here will encompass the discovery of geometrical properties of subsystems and details of interactions between the elements within subsystems and between different subsystems. We will understand the details of influence of this complex hierarchical structure on the reduction of the search for suboptimal operation. Most importantly, it should allow a better understanding of the evaluation and control of the solution quality.

This contribution to the formalization and generalization of human search heuristics 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. The research will lead to the development of *efficient applications* to autonomous navigation in hazardous environment, robot control, combat operations planning as well as applications in different nonmilitary areas. The development of applications will be accomplished by the design of separate programs, and, later on, by the program implementation of the general hierarchy of formal grammars and applying it to a given problem.

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, Englewood 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*, Academic 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., The 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, 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, (to appear).
35. 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).