



Hierarchical networks for space navigation

B. Stilman

*Department of Computer Science and
Engineering, University of Colorado at Denver,
Campus Box 109, CO 80217-3364, USA*

Abstract

This paper reports new results on engineering applications of Linguistic Geometry. This formal theory is intended 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. The Linguistic Geometry relies on the formalization of search heuristics, which allow to decompose complex system into the hierarchy of subsystems, and thus solve intractable problems reducing the search. Currently we investigate heuristics extracted in the form of hierarchical networks of paths. The dynamic hierarchy of networks is represented as a hierarchy of formal attribute languages. This paper includes a brief survey of the Linguistic Geometry, and an example of a solution of optimization problem for space robotic vehicles. This example includes actual generation of the hierarchy of languages and demonstrates the drastic reduction of search in comparison with conventional search algorithms.

It is well known that despite of the universal proliferation of computers 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

340 Artificial Intelligence in Engineering

degrees of success [1, 14, 22]. Implementations based on the formal theories of linear and nonlinear planning meet hard efficiency problems [3, 11, 16, 20, 23]. 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 [4], Ginsburg [9], 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 [7], Narasimhan [15], and Pavlidis [17], and picture description languages by Shaw [21], Feder [5], Rosenfeld [18].

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 [24]. 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 [10], Rozenkrantz [19].

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 [16], Fikes, Nilsson [6], Sacerdoti [20], McCarthy, Hayes [12, 13], 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 [1, 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* [25-29]. 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*.

1 Complex System

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

$$\langle X, P, R_p, \{ON\}, v, S_i, S_t, TR \rangle.$$

where $X=\{x_j\}$ is a finite set of *points*; $P=\{p_i\}$ is a finite set of *elements*; P is a union of two non-intersecting subsets P_1 and P_2 ; $R_p(x, y)$ is a set of binary relations of *reachability* in X (x and y are from X , p from P); $ON(p)=x$, where ON is a partial function of *placement* from P into X ; v is a function on P with positive integer values; it describes the *values* of elements. The Complex System searches the state space, which should have initial and target states; S_i and S_t are the descriptions of the *initial* and *target* states in the language of the first order predicate calculus, which matches with each relation a certain Well-Formed Formula (WFF). Thus, each state from S_i or S_t is described by a certain set of WFF of the form $\{ON(p_j)=x_k\}$; TR is a set of operators, $TRANSITION(p, x, y)$, of transition of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed and added to the description of the state), and of WFF of applicability of the transition. Here,

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

Add list: $ON(p)=y$;

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

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out in turn

with participation of elements p from P_1 and P_2 respectively; omission of a turn is permitted.

According to definition of the set P , the elements of the System are divided into two subsets P_1 and P_2 . They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x, y)$ holds. The current location of each element is described by the equation $ON(p)=x$. Thus, the description of each state of the System $\{ON(p_i)=x_k\}$ 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 [6], nonlinear planner NOAH [20], 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 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 :

$$t_0 = a(x)a(x_1) \dots a(x_l),$$

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 $i = 0, 1, \dots, l-1$; 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 coincide. $\mathcal{P}(t_0) = \{x, x_1, \dots, x_l\}$ is the set of parameter values of the trajectory t_0 .

A *shortest trajectory* t of $t_p(x, y, l)$ is the trajectory of minimum length for the given beginning x , end y and element p .

Properties of the Complex System permit to define (in general form) and study formal grammars for generating the shortest trajectories. A general grammar and its application to generating the shortest trajectories for a robotic vehicle presented in [25, 27].

A *Language of Trajectories* $L_t^H(S)$ for the Complex System in a state S is the set of all the trajectories of the length less than H . Different properties of this language and generating grammars were investigated in [25].

3 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 [1], 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. 1. The basic idea behind these networks is as follows. Element p_0 should move along the main trajectory $a(1)a(2)a(3)a(4)a(5)$ 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. 1) 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 $a(6)a(7)a(4)$ and $a(8)a(9)a(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of element

342 Artificial Intelligence in Engineering

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 $a(13)a(9)$. It makes sense for the opposite side to include the trajectory $a(11)a(12)a(9)$ of element q_1 to prevent this control.

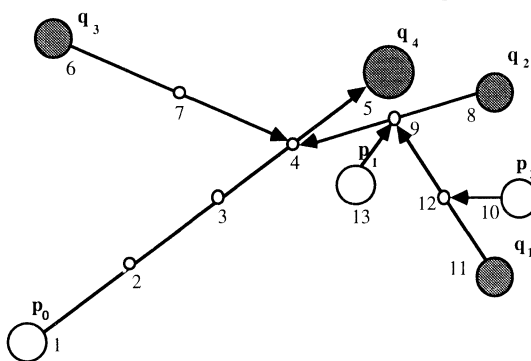


Figure 1: A network language interpretation.

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 $a(8)a(9)a(4)$.

A **Language of Zones** $L_Z(S)$ is a trajectory network language with strings of the form

$$Z = t(p_0, t_0, \tau_0) t(p_1, t_1, \tau_1) \dots t(p_k, t_k, \tau_k).$$

where $t_0, t_1, \dots, t_k$ are the trajectories of elements $p_0, p_1, \dots, p_k$ respectively; $\tau_0, \tau_1, \dots, \tau_k$ are positive integer numbers (or 0) which "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 opposite side. Trajectory $t(p_0, t_0, \tau_0)$ 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 opposite sides. 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 [26-28].

To make it clearer let us show the Zone corresponding to the trajectory network in Fig. 1.

$$Z = t(p_0, a(1)a(2)a(3)a(4)a(5), 4) t(q_3, a(6)a(7)a(4), 3)$$

$$t(q_2, a(8)a(9)a(4), 3) t(p_1, a(13)a(9), 1) t(q_1, a(11)a(12)a(9), 2) t(p_2, a(10)a(12), 1)$$

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 blacks start in its turn to move their 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 τ) allocated to it*. For example, 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. 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 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.

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

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*.

For example, in Fig. 1 the trajectories $a(6)a(7)a(4)$ and $a(8)a(9)a(4)$ are connected with the

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 [28].

4 Space Navigation Problem

This problem can be represented as a Complex System naturally (Fig. 2). A set of X represents the operational district which could be the area of combat operation broken into smaller cubic areas, “points”, e.g., in the form of the big cube of $8 \times 8 \times 8$, $n = 512$. It could be a space operation, where X represents the set of different orbits, or an air force battlefield, etc. P is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests; $R_p(x, y)$ 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, the other can jump or ride, sail and fly, or even move from one orbit to another. Some of them move fast and can reach point y (from x) in “one step”, i.e., $R_p(x, y)$ holds, others can do that in k steps only, and many of them can not reach this point at all. $ON(p) = x$, if robot p is at the point x ; $v(p)$ is the value of robot p . This value might be determined by the technical parameters of the robot. It might include the immediate value of this robot for the given combat operation; S_i is an arbitrary initial state of operation for analysis, or the starting state; S_t is the set of target states. These might be the states where robots of each side reached specified points. On the other hand S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_i) = x_k\}$ 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.

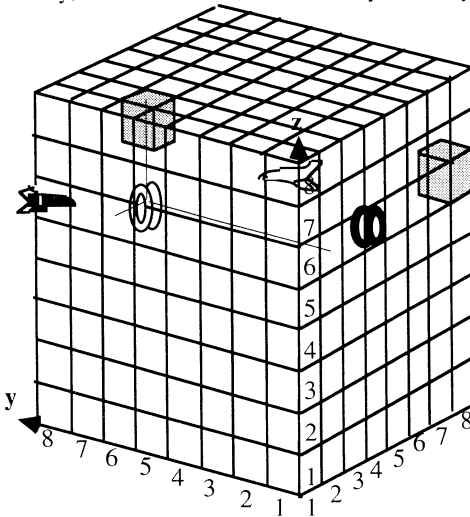


Figure 2: A space navigation problem for autonomous robotic vehicles.

Space robotic vehicles with different moving capabilities are shown in Fig. 2. The operational district X is the cubic table of $8 \times 8 \times 8$. Robot W-INTERCEPTOR (White Interceptor) located at 118 ($x=1, y=1, z=8$), can move to any next location, i.e., 117, 217, 218, 228, 227, 128, 127. The other robotic vehicle B-STATION (double-ring shape in Fig. 2) from 416 can move only straight ahead towards the goal area 816 (shaded in Fig. 2), one square at a time, e.g., from 416 to 516, from 516 to 616, etc. Robot B-INTERCEPTOR (Black Interceptor) located at 186, can move to any next square similarly to robot W-INTERCEPTOR. Robotic vehicle W-STATION located at 266 is analogous with the robotic B-STATION; it can move only straight ahead to the goal area 828

(shaded in Fig. 2). Thus, robot W-INTERCEPTOR on 118 can reach any of the points $y \in \{117, 217, 218, 228, 227, 128, 127\}$ in one step, i.e., $RW-INTERCEPTOR(118, y)$ holds, while W-STATION can reach only 267 in one step.

Assume that robots W-INTERCEPTOR and W-STATION belong to one side, while B-INTERCEPTOR and B-STATION belong to the opposite side: $W-INTERCEPTOR \in P_1$, $W-STATION \in P_1$, $B-INTERCEPTOR \in P_2$, $B-STATION \in P_2$. Also assume that both goal areas, 816 and 268, are the safe areas for B-STATION and W-STATION, respectively, if station

344 Artificial Intelligence in Engineering

reached the area and stayed there for more than one time interval. Each of the STATIONS has powerful weapons capable to destroy opposing INTERCEPTORS at the next diagonal locations ahead of the course. For example W-STATION from 266 can destroy opposing INTERCEPTORS at 157, 257, 357, 367, 377, 277, 177, 167. Each of the INTERCEPTORS is capable to destroy an opposing STATION approaching its location from any direction, but it also capable to protect its friendly STATION approaching its prospective location. In the latter case the joint protective power of the combined weapons of the friendly STATION and INTERCEPTOR (from any next to the STATION area) can protect the STATION from interception. For example, W-INTERCEPTOR located at 156 can protect W-STATION on 266 and 267.

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

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

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-STATION from 418 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. Is there a strategy for the White side to make a draw? Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Experiments with the computer chess programs showed that for the similar 2-D problem (in chess terms – the R.Reti endgame) the search tree includes about a million moves (transitions). Of course, in the 3-D case the search would require billions of moves. It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools.

The details of generation of the Languages of Trajectories and Zones for the robotic systems are considered in [27].

5 Search Generation for Space Navigation Problem

Consider how the hierarchy of languages works for the optimal control of the space robotic system introduced above (Fig. 2). We generate the search of the Language of Translations representing it as a conventional search tree (Fig. 7) and comment on its generation. In fact, this tree is close to the search tree of the relative 2-D problem [29]. Moreover, it is 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 [1].

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limited number of steps which is called a horizon. In general, the value of the horizon is unknown. As a rule, this value can be determined from the experience of solving specific classes of problems employing Linguistic Geometry tools. In absence of such experience, first we have to consider the value of 1 as a horizon, and solve the problem within this value. If we still have resources available, i.e., computer time, memory, etc., we can increase the horizon by one. After each increase we have to regenerate the entire model. This increase means a new level of “vigilance” of the model, and, consequently, new greater need for resources.

In our case it is easy to show that within the horizons of 1, 2, 3, 4 all the models are “blind” and corresponding searches do not give a “reasonable” solution. But, again, after application of

each of the consecutive values of the horizon we will have a *solution* which can be considered as an approximate solution within the available resources. Thus, let the horizon H of the language $LZ(S)$ 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. 3.

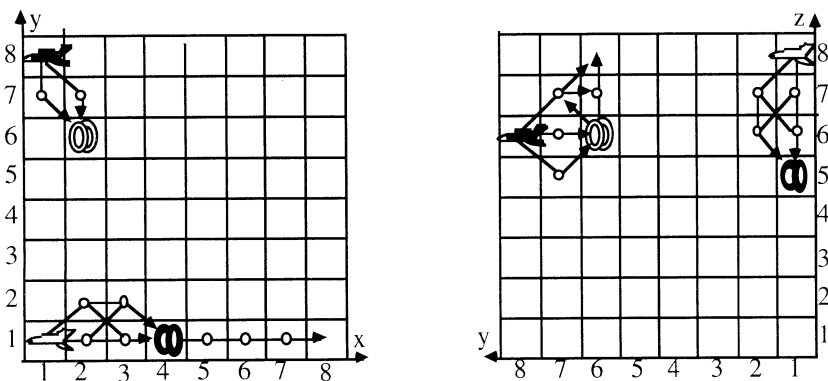


Figure 3: Interpretation of Zones in the initial state of the space robotic system (2 projections)

Zones for INTERCEPTORSs and STATIONS as attacking elements are shown in Fig. 3. For example, one of the Zones for W-STATION, Z_{WS} is as follows:

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

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

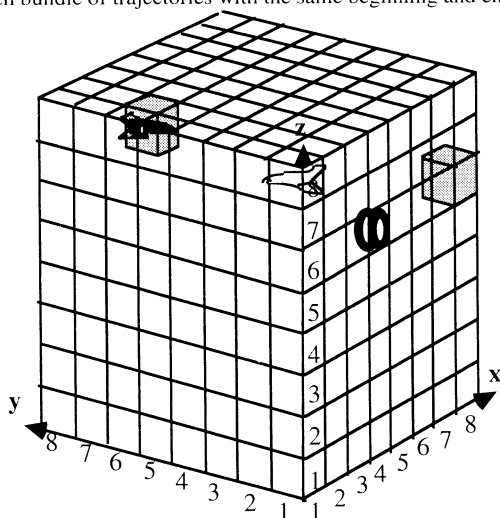


Figure 4: The terminal state where control Zone from 118 to 268 was detected.

Generation begins with the move 1. 266-267 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. ... 186-277, is in the same Zone along the first negation trajectory. The interception continues: 2. 267-268 277:268 (Fig. 4). Symbol ":" means the removal of element. Here the grammar terminates this branch with the value of -1 (as a win of the Black side). This value is given by the special procedure of "generalized

square rules" built into the grammar. 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 the climb up to the move 1. ... 186-277, the tree to be analyzed consists of one branch (of two plies): 2. 267-268 277:268. The inspection procedure determined that the current

346 Artificial Intelligence in Engineering

minimax value (-1) can be “improved” by the improvement of the exchange in the area 268 (in favor of the White side). This can be achieved by participation of W-INTERCEPTOR from 118, i.e., by generation and inclusion of the new so-called “control” Zone with the main trajectory from 118 to 268. The set of different Zones from 118 to 268 (the bundle of Zones) is shown in Fig. 5. The move-ordering procedure picks the subset of Zones with main trajectories passing 227. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing B-STATION on 516. The motion along such trajectories allows to “gain the time”, i.e., to approach two goals simultaneously.

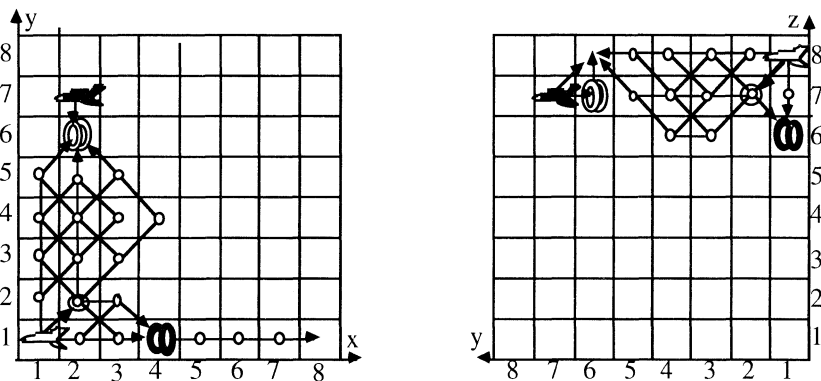


Figure 5: Interpretation of Zones in the state where control Zone from 118 to 268 was included first (2 projections).

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

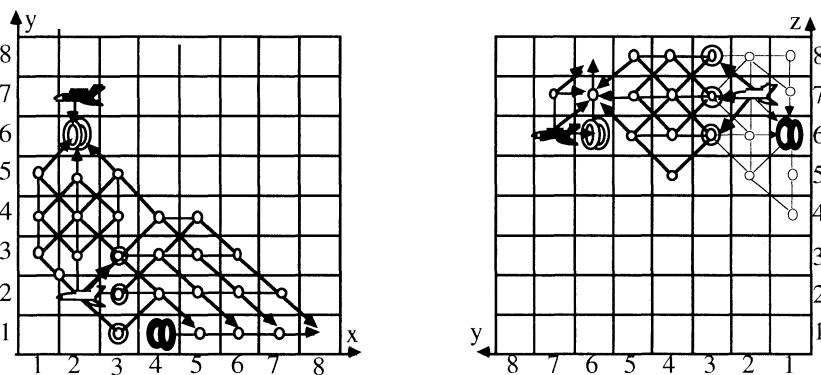


Figure 6: Interpretation of Zones in the state where control Zone from 227 to 267 was included first (2 projections).

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 118 to 268 in the start state can be useful: the minimax value can be improved. Similarly, the most promising “time-gaining” move is 1. 118-227. The Black side responded 1. ... 186-277 along the first negation trajectories

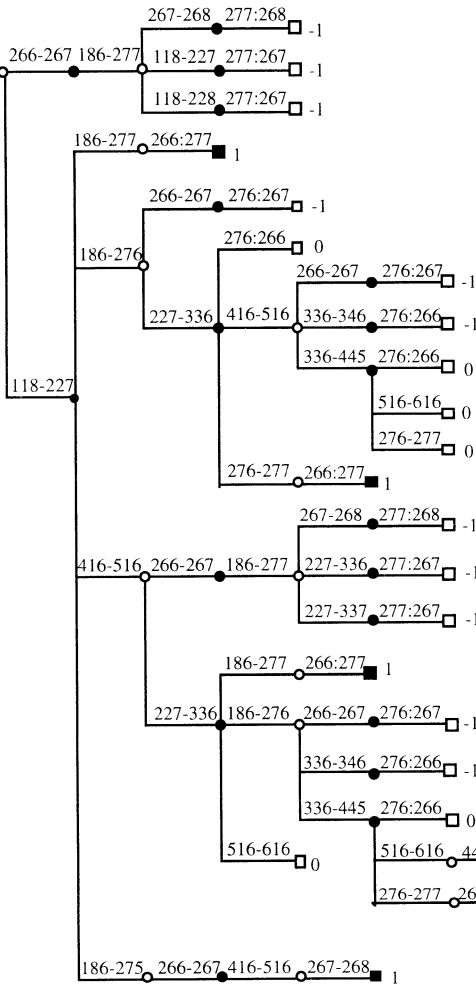


Figure 7: Search tree for the space navigation problem.

informally, from the areas listed above W-INTERCEPTOR can intercept B-STATION (in case of white move). The main trajectories of control Zones passing one of three points, 336, 337, or 338, partly coincide with the potential first negation trajectories. The motion along such trajectories allows to “gain the time”, i.e., to approach two goals simultaneously. The move-ordering procedure picks the subset of Zones with the main trajectories passing 336. Thus, 2. 227-336.

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

6 Conclusion

The example considered in this paper 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 space robotic navigation. It is even more interesting that search reduction achieved in the original domain multiplied tremendously in the new domain.

The following research will lead to the development of *efficient applications* to autonomous

$a(186)a(277)a(267)$ and $a(186)a(277)a(268)$ shown in Fig. 3 (better see yz-projection). Obviously, 2. 266:277, and the branch is terminated. The grammar initiates the climb and move 1. ... 186-277 is changed for 1. ... 186-276 along trajectory $a(186)a(276)a(266)$. Note, that grammar “knows” that trajectory $a(186)a(276)a(266)$ is active in this state, i.e., B-INTERCEPTOR has enough time for interception. The following moves are in the same Zone of W-STATION: 2. 266-267 276:267. The “square rule procedure” cuts this branch and evaluates it as a win of the Black side.

New climb up to the move 2. ... 186-276 and execution of the inspection procedure result in the inclusion of the new control Zone from 227 to 267 in order to improve the exchange in the area 267. The set of Zones with different main trajectories from 227 to 267 is shown in Fig. 6. Besides that, the trajectories from 227 to 516, 616, 716, and 816 are shown in the same Fig. 6. These are “potential” first negation trajectories. It means that beginning with the second symbol $a(336)$, $a(337)$, $a(338)$, or $a(326)$, $a(327)$, $a(328)$, or $a(316)$, $a(317)$, $a(318)$, these trajectories become first negation trajectories in the Zone of B-STATION h5. Speaking

348 Artificial Intelligence in Engineering

navigation in hazardous environment, robot control, combat operations planning as well as applications in different nonmilitary engineering areas.

References

1. Botvinnik, M.M. *Computers in Chess: Solving Inexact Search Problems*. Springer Series in Symbolic Computation, Springer-Verlag, New York, 1984.
2. Botvinnik, M., Petriyev, E., Reznitskiy, A., et al. Application of New Method for Solving Search Problems for Power Equipment Maintenance Scheduling, *Economics and Mathematical Methods*, 1983, **6**, 1030-1041 (in Russian).
3. Chapman, D. Planning for conjunctive goals, *Artificial Intelligence*, 1987, **32**(3).
4. Chomsky, N. Formal Properties of Grammars, *Handbook of Mathematical Psychology*, (ed. R.Luce, R.Bush, E. Galanter), Vol.2, J. Wiley, New York, 323-418, 1963.
5. Feder, J. Plex languages, *Information Sciences*, 1971, **3**, 225-241.
6. Fikes, R.E. & Nilsson, N.J. STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, *Artificial Intelligence* 1971, **2**, 189-208.
7. Fu, K.S. *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs, 1982.
8. Garey, M.R. & D.S.Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1991.
9. Ginsburg, S. *The Math. Theory of Context-Free Languages*, McGraw Hill, New York, 1966.
10. Knuth, D.E. Semantics of Context-Free Languages, *Mathem. Systems Theory*, 1968, **2**, 127-146.
11. McAllester, D. & Rosenblitt, D. Systematic Non-Linear Planning, *Proc. of AAAI-91*, 1991, 634-639.
12. McCarthy, J. Circumscription – A Form of Non-Monotonic Reasoning”, *Artificial Intelligence*, 1980, **13**, 27-39.
13. McCarthy, J. and Hayes, P.J. Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence*, 1969, **4**, 463-502.
14. Mesarovich, M.D., Macko, D., and Takahara Y. *Theory of Hierarchical Multilevel Systems*, Academic Press, New York, 1970.
15. Narasimhan, R.N. Syntax-Directed Interpretation of Classes of Pictures, *Communications of the ACM*, 1966, **9**, 166-173.
16. Nilsson, N.J. *Principles of Artificial Intelligence*, Tioga Publ., Palo Alto, CA, 1980.
17. Pavlidis, T. *Structural Pattern Recognition*, Springer-Verlag, New York, 1977.
18. Rosenfeld, A. *Picture Languages, Formal Models for Picture Recognition*, Academic Press, 1979.
19. Rozenkrantz, D.J. Programmed Grammars and Classes of Formal Languages, *J. of the ACM*, 1969, **1**, 107-131.
20. Sacerdoti, E.D. The Nonlinear Nature of Plans, *Proc. Int. Joint Conference on Artificial Intel.*, 1975.
21. Shaw, A.C. A Formal Picture Description Scheme as a Basis for Picture Processing System, *Information and Control*, 1969, **19**, 9-52.
22. Simon, H.A. *The Sciences of the Artificial*, 2-nd ed., The MIT Press, Cambridge, 1980.
23. Stefik, M. Planning and meta-planning (MOLGEN: Part 2), *Artificial Intelligence*, 1981, **2**, 141-169.
24. Stilman, B. Hierarchy of Formal Grammars for Solving Search Problems, in *Artificial Intelligence. Results and Prospects, Proceedings of the International Workshop*, Moscow, 63-72, 1985, (in Russian).
25. Stilman, B. A Linguistic Approach to Geometric Reasoning, *Int. J. Computers and Mathematics with Applications*, 1993, **26**(7), 29-57.
26. Stilman, B. Network Languages for Complex Systems, *Int. J. Computers and Mathematics with Applications*, 1993, **26**(8), 51-79.
27. Stilman, B. Syntactic Hierarchy for Robotic Systems, *Integrated Computer-Aided Engineering*, 1993, **1**(1), 57-82.
28. Stilman, B. Translations of Network Languages, *Int. J. Computers and Mathematics with Applications*, 1994, **27**(2), 65-98.
29. Stilman, B. A Formal Model for Heuristic Search, *Proceedings of the 22nd Annual ACM Computer Science Conf.*, Phoenix, AZ, March, 1994, 380-389.