A LINGUISTIC APPROACH TO GEOMETRIC REASONING

D R A F T[*]

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

Abstract

The objective of the research considered in this paper is to develop a theoretical foundation for the representation of large-scale hierarchical complex systems, so-called Linguistic Geometry. The research relies on the formalization of heuristics of high-skilled human experts which have resulted in the development of successful decision support systems. This approach is based on a broad application of the theory of formal languages and grammars as well as theories of formal problem-solving and planning on the basis of the first-order predicate calculus. This paper reports new results in the investigation of *geometricalproperties* of the first-level subsystems (paths of elements) unified as Surface Linguistic Geometry. Theoretical constructions considered in this paper are illustrated employing comprehensive examples from power maintenance scheduling, robot control, chess game. A program implementation of this approach should generate decision support systems for a wide class of practical problems.

Keywords: complex systems, hierarchical systems, linguistic geometry, formal languages and grammars, search problems, problem representation, planning and scheduling, robot control.

1. Introduction

A great number of real-world problems such as long and short-range planning (especially for autonomous navigation), scheduling, integrated circuits layout, robot control, combat operations control, etc. might be formally represented as problems of reasoning about complex large-scale systems. There are many such problems where human expert skills in reasoning about complex systems are incomparably higher than the level of modern computing systems. Very often human skills show advances in reasoning about geometrical properties of such 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 tactics 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 high-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 with regard to national security 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

[*] The final version was published as Stilman, B., A Linguistic Approach to Geometric Reasoning, An International Journal: Computers & Mathematics with Applications, Vol. 26, No.7, pp. 29-58, 1993.

then apply and adopt these keys to the decision support systems for the new, as yet, unsolved problems, and first and foremost to the critical complex systems. It should be considered as investigation, development, and consequent expansion of advanced human expert skills into new areas.

The difficulties we meet trying to find optimal operation for real-world complex 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 [1]. Thus, most real-world search problems are not solvable with the help of exact algorithms in a reasonable amount of time.

There have been many attempts to design different approximate algorithms. 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 this system down into subsystems, to study these subsystems separately or in combinations, making appropriate searches, and eventually combining optimal solutions for the subsystems as an approximately optimal solution for the whole system [2], [3], [4]. These ideas have been implemented for many problems with varying degrees of success, but each implementation was unique. There was no general constructive approach for such implementations. Each new problem was carefully studied and previous experience usually could not be applied. Basically we could not answer the question: what are the formal properties of human heuristics which drove us to a successful hierarchy for a given problem and how can we apply the same ideas in a different problem domain. On the other hand, 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 early 1960’s a formal syntactic approach to the investigation of properties of the natural language caused fast development of the theory of formal languages by Chomsky [5], Ginsburg [6], and others [7, 8]. This development provided an interesting opportunity of 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 [9, 10], Narasimhan [11], and Pavlidis [12], and picture description languages by Shaw [13], Feder [14], and Phaltz [15]. The power of a linguistic approach might be explained, in particular, by the recursive nature and expressiveness of the language generating rules, i.e., formal grammars.

Searching for the adequate mathematical tools formalizing human heuristics of dynamic hierarchy, we transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems [20-22]. However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. They should describe mathematically all of the essential syntactic and semantic features of the system and search, and be easily generated by certain controlled grammars. The origin of such languages can be traced back to the origin of SNOBOL-4 programming language and the research on programmed attribute grammars and languages by Knuth [7], Rozenkrantz [8], and Volchenkov [16].

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, Fikes [17], Sacerdoti [18], and McCarthy, Hayes [19] on the basis of the first order predicate calculus.

To show the power of this 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. Just that very model was developed by Botvinnik, Stilman, and others and successfully applied to scheduling, planning, and computer chess. The hierarchical constructions were introduced in [4] in the form of ideas and plausible discussions.

An application of the hierarchy of languages to the chess model was implemented in full as program PIONEER [4, 20]. The results shown by this program in solving complex endgames and middle-game chess positions are not been achieved by other well-known computer chess programs based on the alpha-beta search algorithms, e.g., by current and former World Computer Chess Champions. In order to solve these problems PIONEER showed a very deep selective search with the branching factor close to 1, while all the conventional chess programs based on non-selective search algorithms can not “survive after combinatorial explosion”.

The hierarchy of languages was implemented for the power equipment maintenance in a number of computer programs being used for maintenance scheduling all over the USSR [21, 22, 20]. They set up monthly and yearly maintenance plans of good quality and in reasonable processing time. The comparison with analogous programs based on branch-and-bound search strategies showed the advantage of this approach for monthly planning: the quality of the plan was about the same, but the computation time was essentially shorter. In all experiments the branching factor of the search trees generated by conventional programs was substantially higher. For yearly planning problems, the competition failed, because the conventional programs based on branch-and-bound and dynamic programming search algorithms could not overcome the combinatorial explosion for such a higher-dimensional problem.

The results shown by the applications in solving complex chess and scheduling problems indicate that implementations of the hierarchy of languages resulted in the extremely goal-driven algorithms generating search trees with the branching factor close to 1. Thus, formal linguistic tools presented in this paper deserve theoretical investigation and development in order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex systems.

Dynamic hierarchical systems were developed independently for different problems [2, 3]. But their representation was either informal or non-constructive. Still more such systems might be introduced for another real-world search problems pursuing the same general goal of reducing the search by breaking down complex system into subsystems. That is why we see the general need and applicability of this approach, and consider current research as a model for further generalization and applications.

2. Scientific objectives: informal review

The purpose of Linguistic Geometry is to develop a formal and a general approach for a certain class of complex systems that involves breaking down a system into dynamic subsystems. This approach *does not immediately give* us powerful tools for reducing the search in different complex problems. It *does give* us a set of tools to be used for the formal description of problems where successful results had already been achieved due to the informal plausible reasoning of some human expert. This reasoning should involve the decomposition of a complex system into a hierarchy of dynamic interacting subsystems. The set of tools *permits* us to study this hierarchy formally, to investigate general and particular properties of such hierarchies, to prepare a framework for the evaluation of the complexity and quality of solutions, improve them, if necessary, and generate computer programs for specific applications. This approach *provides* us with an opportunity to transfer formal properties and constructions discovered in one problem to a new one and to apply the same tools to the new problem domain. It actually looks like an application of the methods of a chess expert to a robot control or maintenance scheduling and vice versa. But what about guaranties of success? The guaranties reside in the deeper studies of these methods, in the discovery of inner properties that brought us a success for a certain class of complex systems.

The *class of problems* to be studied are problems of optimal operation of a complex system. This system is considered as a twin-set of *elements* and *points* where elements are units moving from one point to another. It is a very general representation, e.g., in robot control problems *elements* are autonomous robots moving along the *points* of the complex hazardous environment on the surface or in space. The *elements* are divided into two opposite sides; the goal of each side is to attack and destroy opposite side *elements* and to protect its own. Each side aims to maximize a gain, the total value of opposite *elements* destroyed and withdrawn from the system. Such a withdrawal happens if an attacking *element* comes to the point where there is already an *element* of the opposite side.

A one-goal, one-level system should be substituted for a multi-goal multi-level system by introducing intermediate goals and breaking the system down into subsystems striving to attain these goals. The goals of the subsystems are individual but coordinated within the main mutual goal. For example, each second-level subsystem includes elements of both sides: the goal of one side is to attack and gain some element (a target), the other side tries to protect it. In the robot control, it means the selection of a couple of robots of opposing sides: one – as an attacking element, and the other – as a local target, generation of the paths for approaching the target, as well as the paths of other robots supporting the attack or protecting the target. The pruning criteria of the search for an optimal operation in such subsystems and evaluation function are coordinated with the intermediate subsystem's goals and the main goal of the system.

3. A survey of the hierarchy of languages

A set of dynamic subsystems might be represented as a hierarchy of formal languages where each "sentence" (a group of "words" or symbols) of the lower level language corresponds to the "word" of the higher level one. This is a routine procedure in our native language. For example, the phrase "A man who teaches students" creates a hierarchy of languages. A lower level language is a native language without the word "professor." The symbols of this language are all the English words (except "professor"). A higher level language might be the same language with one extra word "A-man-who-teaches-students". Instead, we can use the word "professor" which is simply a short designation of this long word.

Following a linguistic approach each first level subsystem should be represented as a string of symbols with parameters:

$$a(x_1)a(x_2)\ldots a(x_n), \qquad (3.1)$$

where values of parameters incorporate the semantics of the problem domain. They form the so-called Language of Trajectories. For example, for the lower level subsystems in the chess model: $x_1, x_2,\ldots, x_n$ are the coordinates of squares of the chess board and $a(x_1)a(x_2)\ldots a(x_n)$ represents a trajectory (a planning path) of a chess piece from the square x_1 to x_n through squares of stops x_2, $x_3,\ldots, x_{n-1}$. For the robot control problem x_i are the coordinates of the basic points of the robot's planning path. For the maintenance scheduling problem an analogous string represents a maintenance schedule variant for a specific power unit, where x_1, x_2, ..., x_n correspond to the particular days of the scheduling period.

The following sections of this paper are devoted to the development of formal linguistic techniques and their application to the investigation of geometrical properties of the Language of Trajectories.

Let us outline briefly a representation of the higher level subsystems of the complex system. (A formal comprehensive survey is presented in [20, 24].) A second level subsystem should be represented as a similar string with parameters:

$$\boldsymbol{t}(p_1, t_1, f_1)\boldsymbol{t}(p_2, t_2, f_2)\ldots\boldsymbol{t}(p_k, t_k, f_k), \qquad (3.2)$$

where values of parameters again incorporate the semantics of the problem domain and *lower level subsystems*. Symbols p_i represent elements of our system (chess pieces, robots, power units, etc.), t_k represent whole trajectories (lower level subsystems) of elements p_i, i.e., strings $a(x_1^{p_i})a(x_2^{p_i})\ldots a(x_n^{p_i})$, included in this subsystem, f_i represent "time allocated for motion along the trajectory t_i."

Thus, using strings of (3.1), we can represent paths of system's elements, and with the strings of (3.2), 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. Strings (3.2) form the Language of Trajectory Nets.

The system functions by moving from one state to another; that is, the motion of an element from one point to another causes an adjustment of the hierarchy of languages. This adjustment can be 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 process of the search, generates a *tree* of translations of the hierarchy of languages. This tree can be represented as a string of the highest level formal language, the Language of Translations.

The search for an optimal (suboptimal) operation (i.e., optimal variant in chess, optimal plan of the robot motion, or optimal maintenance schedule) in the new system is considered as a process of generation and interaction of networks of the form (3.2). This process results in a highly reduced search tree which is represented as a string of the Language of Translations, a member of the Family of Languages of Searches.

The advantages of linguistic representation of the complex hierarchical system become even more apparent when we consider a development of the formal mechanism for generating this representation. A hierarchy of languages should be generated by the hierarchy of formal grammars. These grammars generate strings of symbols with parameters. The lists of parameters incorporate semantics of the string: they are determined by the problem domain, e.g., squares of the chessboard, type of the robot or obstacle, days of planning period, etc. The values of actual parameters should be computed and assigned in a process of derivation. Thus, derivation itself should be controlled by the state of the problem domain. This objective could be achieved by providing the grammar with a control mechanism like subsets of productions admitted for application at each step of derivation and conditions of applicability of the productions, i.e., certain Well Formed Formulas (WFF) of the predicate calculus. During the derivation, this control mechanism in its turn must be controlled by the problem domain through the values of WFF and actual parameters of the substring, that have already been derived on the previous steps.

An approach to investigation of properties of complex hierarchical systems as hierarchies of formal languages was called a Linguistic Geometry [20, 21, 23-25].

4. Complex Systems

DEFINITION 4.1

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

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;

$\mathbf{R_p}$(x,y) is a set of binary relations of *reachability* in X (x and y of X, p of 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 a space of states, hence, it 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 collection 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: $(ON(p)=x) \wedge R_p(x,y)$,
where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out in turn with participation of elements p from P_1 and P_2 respectively; omission of a turn is permitted.

According to definition of the set P, the elements of the System are divided into two subsets P_1 and P_2. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x,y)$ holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from the point x to the point y. The element q from the 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 variant of transitions leading from one of the initial states of S_i to a target state S of S_t. The target states are described with the help of the following function of states **m(S)**.

The values of m(S) for a target state are much bigger than for any other one (they are greater than some constant). In our case we stipulate that

$$(4.1) \quad \mathbf{m(S)}=\sum v(p_i)-\sum v(p_j),$$

where p_i of P_1 and p_j of P_2 which are not withdrawn in a state S. The same function is used to evaluate variants of the search.

With such a problem statement for search for the optimal sequence of transitions into the target state, we could use formal methods like those in the problem-solving system STRIPS [20] , nonlinear planner NOAH [14], or in subsequent planning systems, such as MOLGEN [21] or TWEAK [22]. However the search would have to be made in a space of a huge dimension (for nontrivial examples), i.e., in practice no solution would be obtained. We, thus, devote ourselves to search for an approximate solution of a reformulated problem, considering our Complex System in some sense as *nearly decomposable* [2].

It is easy to show that positional games such as chess and checkers might be considered as Complex Systems (see Section 8). But it is more interesting that this specific model of the formal linguistic approach is applicable to representing and solving a wide class of practical problems such as power maintenance scheduling, long-range planning, operations planning, VLSI layout, and various operations research problems [17, 18]. The idea is that the optimal variant of operation of these real-world systems might be artificially reduced to a two-sides game where one side strives to achieve some goal and the other is responsible for the provision of resources as shown in Sections 9-12.

5. Geometrical Properties of the Complex System

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

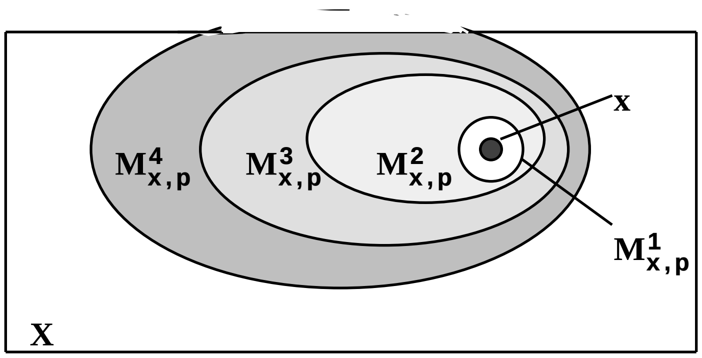


Fig. 1. An interpretation of the family of reachability areas

DEFINITION 5.1

A **map of the set X** relative to the point x and element p for the Complex System is the mapping:

$$\mathbf{MAP_{x,p}}: X \longrightarrow \mathbf{Z_+},$$

(where x is from X, p is from P) which is constructed as follows. We consider a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets of X $\{M^k_{x,p}\}$**:**

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 $y \neq x$, and

$MAP_{x,p}(y)=0$, 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.***

A map of the set X relative to the point x and element p for the Complex System is the mapping:

$\mathbf{MAP_{x,p}}: X \to \mathbf{Z_+}$,

(where x is from X, p is from P) which is constructed as follows. We consider a family of the *areas of reachability* from the point x, i.e.,the following nonempty subsets $\{M^k_{x,p}\}$**:**

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 then 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 $y \neq x$, and

$MAP_{x,p}(y)=0$, if $y=x$.

It is easy to verify that the map of 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.***

6. Chess game as Complex System.

The problem of programming of the game of chess is the most transparent example of the Linguistic Geometry application. This problem domain with the method informally described in [1] was actually the first application and experimental area for the formal linguistic approach. In this model of the Complex System (Definition 4.1):

X represents 64 squares of the chess board, i.e., n=64;

$\mathbf{P_1}$ and $\mathbf{P_2}$ are the white and black pieces;

$\mathbf{R_p(x,\ y)}$ are given by the rules of the game, permitting or forbidding a piece p to make a move from a square x to a square y; thus a point x is *reachable* from a point y for an element p, if a piece p can move from a square x to a square y according to the chess game rules;

ON(p)=x, if piece p stands on the square x;

v(p) is the value of piece p, e.g., pawn - 1, N - 3, B - 3, R - 5, Q - 9, K - 200;

$\mathbf{S_i}$ is an arbitrary initial chess position for analysis, or the starting position of the game;

$\mathbf{S_t}$ is the set of chess positions which can be obtained from all possible mating positions in two half moves by capturing of the King (suppose, this capture is permitted).

The collections of WFF $\{ON(p_j)=x_k\}$ correspond to the lists of pieces with their coordinates in each position.

TRANSITION(p, x, y) represents the move of the piece p from the square x to the square y; if on y there stands a piece of the opposing color, a capture is affected.

The chess problem does not completely meet the requirements of the definition of the Complex System.We have neglected such an important chess concept as blockade: in the Complex System several elements (pieces of the same color) can stand on the same point (square). Besides that, we have neglected certain specific chess features, such as castling, capture en passant, pawn promotion, etc. All these chess complications are not crucial for our model; at the implementation stage of the hierarchy of languages for this model (program PIONEER) all this was taken into account [4].

Investigating the geometry of the chess system we can see that here $MAP_{x,p}(y)$ yields the number of moves necessary for the piece p from the square x to reach the square y along the shortest path. Because of symmetry of the relation R_p in this model, $MAP_{x,p}(y)$ specifies the *metric* on the chessboard, own for each kind of piece. For a pawn the symmetry is more complex:

$R_p(x,y)=R_q(y,x)$,

where p and q are the black and white pawn, respectively. Thus function MAP might be used as a "ruler" to measure *distances* in this system for different elements.

When implementing the geometrical model for the chess problem, it was necessary to give a tabular specification of the function MAP, in order to increase the efficiency of the program PIONEER [4]. For this, in accordance with the relations R_p (the chess rules of movement of of the pieces), seven square tables 15 x 15 were specified. Each of the tables was filled with the numbers for one of the chess piece types according to the following principle: the piece is placed on the central field of the table (0 is written there); on the remaining fields we write numbers equal to the number of moves necessary for the piece to reach the given field from the central field along the shortest path. These tables may be unified in the form of the following table $T15(v_1, v_2, f)$ with the dimension 15 x 15 x 7. (For all x of X, $x=(x_1, x_2)$, $x_1=1, 2,\ldots, 8$, $x_2=1, 2,\ldots, 8$, where x_1 and x_2 correspond to files and rows of the chessboard, respectively.) Then

$$MAP_{x,p}(y)=T15(v_1, v_2, f), \qquad (6.1)$$

where $x=(x_1, x_2)$, $y=(y_1, y_2)$, $v_1=8-x_1+y_1$, $v_2=8-x_2+y_2$, $f=f(p)$ is the type of the piece p (King, Rook, etc.). Seven tables 15x15 specify on X seven different metrics.

Practically we can imagine the following procedure for the computation of (6.1). The array 8 x 8 is superimposed on the array 15 x 15 in such a way that square x coincides with the central square of the array 15 x 15. Further, let us assume that array 8 x 8 is transparent, then on corresponding squares we could see values of $MAP_{x,p}$, i.e., the values of actual distance of these squares from the square x. An example of superposition of tables for x=c2 and p=Rook is shown in Fig. 2. For details see [4].

2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2

Fig. 2. An example of superposition of tables 8 x 8 and 15 x 15 for the Rook standing on c2.

7. Autonomous robots as elements of the Complex System

A robot control model can be represented as a Complex System the similar way as chess.

X represents the operational district. It might be the area of combat operation broken into squares, e.g., in the form of the table 8 x 8, n=64. It might be space operation, then X represents the set of different orbits, etc.

P is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests;

R_p(x, y) represent moving capabilities of different robots: robot p can move from the point x to the point y if R_p(x, y) holds. Some of the robots can crawl, the other can jump or ride, or even sail and fly. Some of them move fast and can reach point y (from x) in "one step", i.e. R_p(x, y) holds, the other can do that in *k* steps only, i.e., $MAP_{x,p}(y)=k$, and many of them can not reach this point at all, $MAP_{x,p}(y)=2n$.

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 {$ON(p_j)=x_k$} correspond to the lists of robots with their coordinates in each state.

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

Without going into details similar to the chess, specific tables can represent moving capabilities of different robots. An example of distances $MAP_{h8,K}(v)$ for for the robot K standing on the point h8 of the set X (X is the table of 8 x 8) is shown in Fig. 3. (The moving capabilities of this robot are identical to the chess King.) For example, the distance from h8 to the point c6 is equal to 5.

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

Fig. 3. Values of $MAP_{h8,K}$

8. A scheduling problem as Complex System

8.1 STATEMENT OF THE PROBLEM

Here we consider a way of transformation of the different real-world system into the Complex System.

Assume that energy-producing company is going to set up a maintenance plan for power-producing equipment for a given planning period T_{max}, e.g., month, year. There exists an array of m demands for maintenance work of power units. The problem is to satisfy these demands. To do that we must include the maintenance work for all the demanded units into the plan, i.e., to schedule maintenance. A maintenance work of a power unit causes turning off of this unit, and, consequently, a fall of generating power in the system. Thus, it is impossible to satisfy all the demands because of problem constraints, which is basically the power reserve, e.i., the amount of power to be lost without turning off customers. This amount varies daily.

Each demand requests maintenance work for one power unit (j-th unit) and contains three attributes: w_j, the demanded power of the unit; h_j, the fall in the operating power of the energy-producing system because of maintenance of this unit (resources requirement); and x_j^{max}, required duration of maintenance. For simplicity, we neglect the rest of the demand parameters. For the same reason we specify the only one type of constraints the function $f(i)$ of power reserve for the energy-producing system, where i is the number of a day of the planning period. On the i-th day of the planning period the total fall in the operating power, because of the maintenance of some power units, can not be greater then the value $f(i)$. The values of all the parameters are positive integer numbers.

The optimum criterion of the plan is the maximum total demanded power of the units being maintained.

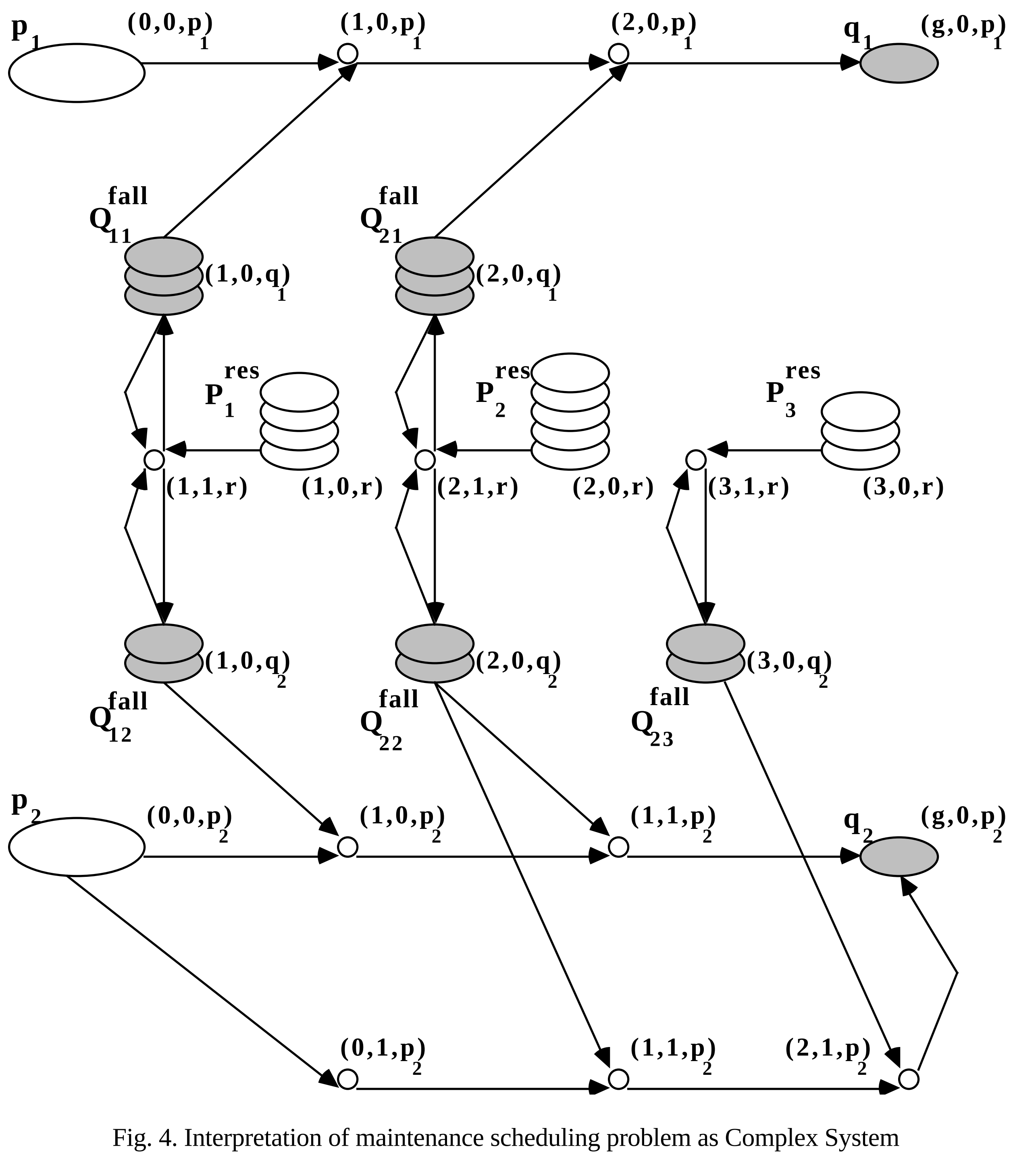


Fig. 4. Interpretation of maintenance scheduling problem as Complex System

In terms of the Complex System, this problem might be represented as a twin-set of *elements* and *points*, as depicted in the Fig. 4. Here *points* form a network which is used by *elements* as a "railroad" to reach certain nodes. There are two classes of *elements*. The first one includes power units, depicted as white discs p_1, p_2, striving to reach nodes (g, 0, p_1) and (g, 0, p_2) and thereby gain opposite elements q_1, q_2(i.e., the ones to be maintained). The other elements of the first class are depicted as pyramids of white disks P_i^{res}: each pyramid represents a daily stock of resources, the power reserve for the energy-producing system. The pyramids of opposite black discs Q_{ij}^{fall} represent requirements of resources, the daily fall in the operating power because of the maintenance of the units p_1 and p_2. The black discs control the nodes of paths for discs p_1, p_2 and able to gain any of them, i. e., maintenance can not take place without provision of resources. It means we are forced to spend white discs of pyramids P_i^{res} exchanging them in the nodes (i, 1, r) with the black discs of Q_{ij}^{fall}. These actions can "clear away" the paths for power units p_1 and p_2. For a closer look at this example see Section 8.3.

8.2. FORMAL REPRESENTATION

Formal representation of the Complex System for the maintenance problem is as follows:

$\mathbf{X}=(Y\cup\{g\}) \times Y \times (P_{dem}\cup Q_{dem}\cup\{r\})$, where $Y=\{0,1,...,T_{max}\}$,

P_{dem} is the set of power units included in the demands, $|P_{dem}|$ is the number of demands. It is introduced a duplicate set Q_{dem} of the elements q_j, and one-to-one correspondence $q_j \longleftrightarrow p_j$ is established between elements of Q_{dem} and P_{dem} $\mathbf{P}=P_1\cup P_2$, P_1 and P_2 are not intersected and

$$P_1=P_{dem}\cup P_{reserve}, \quad P_2=Q_{dem}\cup Q_{fall}, \quad P_{reserve}=\bigcup_{i=1}^{Tmax} P_i^{res}, \quad Q_{fall}=\bigcup_{i=1}^{Tmax}\ \bigcap_{j=1}^{Qdem} Q_{ij}^{fall}$$

To show the number of elements $|P_{reserve}|$ and $|Q_{reserve}|$ we have to define $\mathbf{v_o}$. It is the quantum of power fall (loss), the common factor of all values *f(i)* of power reserve and all values h_j of power fall (for all demanded units); for example, $\mathbf{v_o}=1$ Megawatt. We can now determine $|P_{reserve}|$ and $|Q_{fall}|$, having given $|P_i^{res}|$ and $|Q_{ij}^{fall}|$. Thus, $|P_i^{res}|=f(i)/\mathbf{v_o}$ and $|Q_{ij}^{fall}|=h_j/\mathbf{v_o}$.

The relation of reachability $R_p(x,y)$ can be given explicitly by setting the values for all the triples of p, x, y:

$$\mathbf{R_p(x,y)}=\begin{cases} ((x=(0, y_2, p)) \wedge (y-(0, y_2+1, p))) \vee ((x=(y_1, y_2, p)) \wedge \\ ((y=(y_1+1, y_2, p))) \wedge ((x=(x_i^{max}, y_2, p)) \wedge (p=p_i) \wedge \\ (y=(g, 0, p))), & \text{if p from } P_{dem}; \\ ((x=((y_1, 0, q_j)) \wedge (y_1>0)) \wedge (((y=(y_1-y_2, y_2, p_j)) \wedge \\ ((y_1-y_2)>0)) \vee (y=(y_1, 1, r))), & \text{if p from } Q_{y_1 j}^{fall}, \\ & \text{a subset of } Q_{fall} \\ (((x=(y_1, 0, r)) \wedge (y=(y_1, 1, r))) \vee ((x=(y_1, 1, r)) \wedge \\ (y=(y_1, 0, q_j)) \wedge (y_1>0) \wedge (q_j \subset Q_{dem}), & \text{if p from } P_{reserve}; \\ F \text{ (false)}, & \text{if p from } Q_{dem}. \end{cases}$$

Note that here the reachability relation R_p is *asymmetric*, i.e., there exist p, x, and y such that $R_p(x, y)\neq R_p(y, x)$. To specify the partial function ON(p), it is sufficient to write out its values in the initial state S_o:

$$\mathbf{ON(p)}=\begin{cases} (0, 0, p), & \text{if p from } P_{dem}; \\ (g, 0, p_j), & \text{if } p=q_j \text{ from } Q_{dem}; \\ (y_1, 0, r) & \text{if p from } P_{y_1}^{res}, \text{ a subset of } P_{reserve}; \\ (y_1,0, q_j) & \text{if p from } Q_{y_1 j}^{fall}, \text{ a subset of } Q_{fall}. \end{cases}$$

Function v(p) for target elements $p=q_i$ is equal to the demanded power of separate power unit p_i; for the elements p_i, striving to reach targets, it is equal to the total power of all the demands, and for elements p of power reserve and fall v(p) equals to v_o, the quantum of power fall (see above).

$$\mathbf{v(p)=}\begin{cases} w_i, & \text{if } p=q_i \text{ from } Q_{dem}; \\ \sum_{i=1}^{|Q_{dem}|} w_i, & \text{if p from } P_{dem}; \\ \mathbf{v_o} & \text{if p from } P_{reserve} \cup P_{fall}. \end{cases}$$

The functioning of the system can easily be described using formulas for the TRANSITION operator.

$\mathbf{S_i}$, the initial state, corresponds to the state of the energy-producing system in the "zero day" of planning period, while the target states

$\mathbf{S_t}$, the target states, correspond to the state of the system with the maximum total demanded power of units being maintained. Thus, states from S_t can be described as states of the energy-producing system by the end of the planning period, in which the WFF ON(p_i)=(g, 0, p_i) are true for numbers i such that $\sum v(q_i)$ is maximum (q_i from Q_{dem}).

T, the set of transitions, consists of the “moves” of the elements along the network. Following are the meanings of some transitions (see Fig. 4):

— TRANSITION(p_i, x, (g, 0, p_i)) with removal of the WFF ON(q_i)=(g, 0, p_i) means completion of the maintenance of the unit p_i.

— TRANSITION(p_i, x, (1, y_2, p_i)) with addition of the WFF ON(p_i)=(1, y_2, p_i) means the unit p_i being taken out for the maintenance work on the day y_2.

— TRANSITION(p_i, (0, y_2, p_i), (0, y_2+1, p_i)) with addition of the WFF ON(p_i)=
(0, y_2+1, p_i) and removal of ON(p_i)=(0, y_2, p_i) means that on the day y_2+1 unit p_i has not yet been taken out for maintenance in the given plan variant.

8.3. DETAILS OF MAINTENANCE PROBLEM

To clarify this problem let us return to the example depicted in Fig. 4 and consider it in details. This is the maintenance planning problem for two units over a period of three days:
w_1=5, w_2=2; h_1=3, h_2=2; $x_1^{max}=x_2^{max}$=2; T_{max}=3; $f(1)$=4; $f(2)$=5; $f(3)$=3.
(A reader should not be confused by the simplicity of the example shown in the Fig. 4. It is cited here only for clarification of our approach. For the practical applications there were considered hundreds and even thousands of power units, and different kinds of resources including those which required some time to be delivered to the places of maintenance.)

From Fig. 4 it is seen that, for setting up the maintenance plan, the elements p_i have to go from the points (0, 0, p_i) to the points (g, 0, p_i). In particular, for element p_2 to get through to the point (g, 0, p_2) along any of the paths

(0, 0, p_2)—>(1, 0, p_2)—>(2, 0, p_2)—>(g, 0, p_2)

or

(0, 0, p_2)—>(0, 1, p_2)—>(1, 1, p_2)—>(g, 0, p_2),

it is necessary to do away with the elements of the set (pyramid) Q_{12}^{fall} at the point (1, 0, q_2), as well as the elements of the pyramids Q_{22}^{fall}, Q_{23}^{fall} at the points (2, 0, q_2), (3, 0, q_2). The elements of these pyramids control the points of the path of the element p_2 to the target. Obviously, pyramids of elements from Q_{fall} correspond to fall of power in the energy-producing system during the time of maintenance of power units.

For liquidation of the elements from Q_{fall} we have three sets (pyramids of discs) P_1^{res}, P_2^{res}, P_3^{res} at the points (1, 0, r), (2, 0, r) and (3, 0, r) corresponding to the power reserves in the system during each particular day. It is necessary to carry out a transition, i.e., to move an element from P_1^{res} to the point (1, 1, r), then move an element from Q_{12}^{fall} to the same point, i. e., to perform a "capture", then move the next element from $P_1^{res,}$ and so forth.

In the given example pyramids are placed at one-step distance from the points of exchange. It means the instantaneous availability of resources in the given problem. For complex real-world problems pyramids of resources have to be placed at several steps from the points of exchange which means that resources delivery should start in advance, in several time intervals.

Returning to our example, if at the point (1, 1, r) it is possible to exchange all the elements from Q_{fall}, then the point (1, 0, p_2) becomes traversable freely for the element p_2. If this, however, is not possible (as is in fact shown in Fig. 4), owing to the fact that three elements of the pyramid P_1^{res} were spent on removing the control from the point (1, 0, p_1), i.e., on liquidating Q_{11}^{fall}, and if the remaining single element is not sufficient for destroying the two elements of Q_{12}^{fall}, the element p_2 is forced to move to the point (0, 1, p_2). Thus, on the first day of the planning period, only one of the power units (p_1, for example) can be taken out for maintenance because of insufficiency of power reserve. The second unit p_2 will be taken out on the second day (displacement (0, 1, p_2)—>(1, 1, p_2)). Different versions of the maintenance plan are matched by different variants of movement of elements from P along points from X.

Let us take a closer look at the geometrical structure of the maintenance model. Due to the asymmetry of the relation R_p for this model (Section 5) function $MAP_{x,p}(y)$ is specified in accordance with definition from the Section 5 as asymmetric function of distance. In particular, from Fig. 4 $MAP_{(0,\ 0,\ p_2),\ p_2}(1, 0, p_2)=1$, $MAP_{(0,\ 0,\ p_1),p_1}(g, 0, p_1)=3$, etc.

9. Controlled grammars

In pattern recognition problems, a linguistic approach was proposed [9-15] for representation of hierarchic structured information contained by each pattern, i.e., for describing patterns by means of simpler subpatterns. This approach brings to light an analogy between the the hierarchic structure of patterns and the syntax of languages. The rules controlling the merging of subpatterns into patterns are usually given by the so-called pattern description grammars, with the power of such description being explained by the recursive nature of the grammars. Using similar approach for generating of the hierarchy of formal languages, we make use of the theory of formal grammars in the form developed in [7, 8, 16]. We begin with the definition of the class of grammars to be used.

DEFINITION 9.1

A **controlled grammar G** is the following eight-tuple:

$$\mathbf{G=(}\boldsymbol{V_T, V_N, V_{PR}, E, H, Parm, L, R}\mathbf{)},$$

where

$\boldsymbol{V_T}$ is the alphabet of *terminal symbols*;

$\boldsymbol{V_N}$ is the alphabet of *nonterminal symbols*, *S* (from V_N) is the start symbol;

$\boldsymbol{V_{PR}}$ is the alphabet of the *first order predicate calculus PR*:

$V_{PR}=Truth \cup Con \cup Var \cup Func \cup Pred \cup \{$symbols of logical operations$\}$,

where

Truth are truth symbols T and F (these are reserved symbols);

Con are constant symbols;

Var are variable symbols;

Func are functional symbols ($Func = Fcon \cup Fvar$). Functions have an attached non-negative integer referred to as *arity* indicating the number of elements of the domain mapped onto each element of the range. A term is either a constant, variable or function expression. A *function expression* is given by a functional symbol of arity k, followed by k terms, t_1, t_2,..., t_k, enclosed in parentheses and separated by commas;

Pred are predicate symbols. Predicates have an associated positive integer referred to as *arity* or “argument number” for the predicate. Predicates with the same name but different arities are considered distinct. An *atom* is a predicate constant of arity n, followed by n terms, t_1, t_2,..., t_n, enclosed in parentheses and separated by commas. The truth values, T and F, are also atoms.*Well-formed formulas* (or WFF) are atoms and combinations of atoms using logical operations;

E is an enumerable set called the *subject domain*;

H is an *interpretation* of *PR* calculus on the set *E*, i.e., a certain assignment of the following form. Each

– constant from *Con* is assigned to an element of *E*;

– variable from *Var* is assigned to a nonempty subset of *E*; these are allowable substitutions for that variable;

– predicate *Q* from *Pred* of arity *n* is assigned to a relation on the set *E* of arity *n*, i.e., to a mapping from E^n into {T, F};

– function *f* of arity *k* is assigned to a mapping *h*(*f*)from *D* into *E*, where *D* belongs to E^k. If *f* is from *Fvar*, then *D* and the mapping *h*(*f*) vary in the process of derivation in the grammar G.

Thus, the interpretation *H* allows us to calculate the value of any function (it lies in *E*) and any predicate (F or T), if the values of all variables contained by them are specified.

Parm is a mapping from V_T UV_N in 2^{Var} matching with each symbol of the alphabet V_T UV_N a set of formal parameters, with *Parm*(*S*)=*Var*;

L is a finite set called the set of *labels*;

R is a finite set of *productions*, i.e., a finite set of the following seven-tuples:

$$(l,\ Q,\ A \longrightarrow B,\ \pi_k,\ \pi_n,\ F_T,\ F_F).$$

Here

l (from ***L***) is the label of a production; the labels of different productions are different, and subsequently sets of labels will be made identical to the sets of productions labeled by them;

Q is a WFF of the predicate calculus *PR* , the *condition* of applicability of productions; *Q* contains only variables from *Var* which belong to *Parm*(*A*);

A—>B is an expression called the *kernel of production*, where

A is from V_N;

B is from (V_T U V_N)* is a string in the alphabet of the grammar G;

π_k is a sequence of functional formulas corresponding to all formal parameters of each entry of symbols from V_T UV_N into the strings *A* and *B* (*kernel actual parameters*);

π_n is a sequence of functional formulas corresponding to all formal parameters of each functional symbol from *Fvar* (*non-kernel actual parameters*);

F_T is a subset of ***L*** of labels of the productions permitted on the next step of derivation if *Q*=T (“true”); it is called a *permissible set in case of success*;

F_F is a subset of ***L*** of labels of the productions permitted on the next step of derivation if *Q*=F(“false”); it is called a *permissible set in case of failure*.

Table 1
A structure of typical controlled grammar

L	Q	Kernel, π_k	π_n	F_T	F_F
l_i	Q_i	$A(\ ,\ ,\ ) \longrightarrow a(\ ,\ ,\ )b(\ ,\ ,)$			
	$V_T = \ldots$ E is ...	$V_N = \ldots$ $Parm$: ... $V_{PR} = \ldots$			

A finite set of strings from V_T* and formulas from π_n , in which each formal parameter (for every entry of a terminal symbol into a string) is attributed with a value from E and each symbol f from *Fvar* is matched with a mapping $h(f)$, serves as a *derivation result.*

Derivation in controlled grammar takes place as follows. A symbol S serves as the start of derivation, where its formal parameters are provided with initial mappings $h(f)$ are specified for all f from *Fvar*. In the role of the *initial permissible set* of productions we take the entire set $\boldsymbol{L}$. To a current string we apply each of the productions of the current permissible set, the symbol A for which enters into the string. As a result of applying a production, a new string and a new permissible set are formed. Later on derivation for each of the strings obtained from a given one takes place independently.

If none of the productions from permissible set can be applied, then derivation of the given string is discontinued. If this string consists only of terminal symbols, then it goes into the set of *derivation results*, otherwise it is discarded.

The application of a production takes place as follows. We choose the leftmost entry of the symbol A in the string. We compute the value of the predicate Q. If Q=F, the F_F becomes the permissible set, and the application of the production is ended. If Q=T, then the symbol A is replaced by the string B; we carry out computation of the values of all formulas from π_k corresponding to the parameters of the symbols, and the parameters assume new values thus computed. New mappings $h(f)$ (f from *Fvar*) are specified by means of formulas from π_n; the permissible set is furnished by F_T, and application of the production is ended. (In the record of the production the formulas from π_n leaving $h(f)$ unaltered are omitted.)

In constructions with which the controlled grammar is provided, it is easy to observe analogies with the programming language SNOBOL-4.

DEFINITION 9.2

A **language L[G] generated by the controlled grammar G** is the union of all the sets which are the derivation results in this grammar.

10. A Linguistic Geometry of Paths

Here, we define the lower-level language of the hierarchy of languages. It will serve as a building block to create the upper-level languages. This language actually formalizes a notion of the path between two points for the certain element of the System. An element might follow this path to achieve the goal connected with the ending point.

DEFINITION 10.1

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:

$$t_o = a(x)a(x_1)\ldots a(x_l),$$

where each successive point x_{i+1} is reachable from the previous point x_i: $R_p(x_i, x_{i+1})$ holds for $i=0,1,\ldots, l-1$; element p stands at the point x: ON(p)=x. We denote $t_p(x, y, l)$ the set of trajectories in which p, x, y, and l are the same. $\boldsymbol{P}(t_o)=\{x, x_1, ..., x_l\}$ is the set of parametric values of the trajectory t_o.

Two trajectories of the element p $a(1)a(2)a(3)a(4)a(5)$ and $a(1)a(6)a(7)a(8)a(9)a(5)$ are shown in fig. 5.

DEFINITION 10.2

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.

For example, in fig. 5, a trajectory $a(1)a(2)a(3)a(4)a(5)$ is the shortest trajectory. Reasoning informally, an analogy can be set up: the shortest trajectory is an analogous to a straight line segment connecting two points in a plane. Let us consider an analogy to a k-element segmented line connecting these points.

DEFINITION 10.3

An ***admissible trajectory of degree k*** is the trajectory which can be divided into k shortest trajectories; more precisely there exists a subset $\{x_{i_1}, x_{i_2}, \ldots, x_{i_{k-1}}\}$ of $\boldsymbol{P}(t_o)$, $i_1<i_2<\ldots<i_{k-1}$, $k \leq l$, such that corresponding substrings $a(x_o)\ldots a(x_{i_1})$, $a(x_{i_1})\ldots a(x_{i_2})$, …, $a(x_{i_{k-1}})\ldots a(x_l)$ are the shortest trajectories.

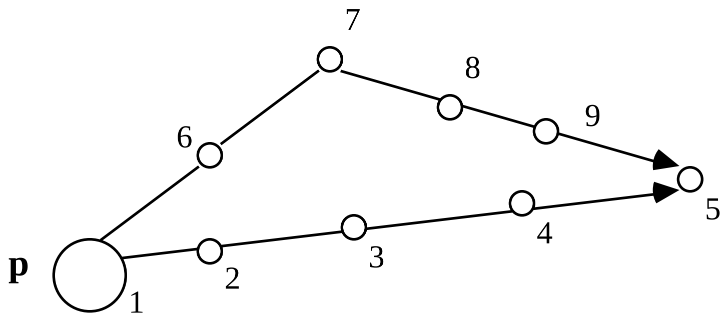


Fig. 5. An interpretation of shortest and admissible trajectories.

The shortest and admissible trajectories of degree 2 play a special role in many problems. Obviously, every shortest trajectory is an admissible trajectory at the same time, but of course, converse statement is not true. There exist admissible trajectories, e.g., of degree 2, which are not shortest. An example of such a trajectory $a(1)a(6)a(7)a(8)a(9)a(5)$ is shown in fig. 5. 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 some intermediate point aside (e.g. point 7 in Fig. 5) 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.

DEFINITION 10.4

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. This language also includes empty trajectory e of the length 0.

Properties of the Complex System permit to define (in general form) and study formal grammars for generating the Language of Trajectories as a whole along with its subsets: shortest and admissible (degree 2) trajectories.

11. Generation of trajectories

Consider the following controlled grammar for the Complex System with symmetric relation of reachability R_p:

Table 2. A grammar of shortest trajectories $G_t^{(1)}$

L	Q	Kernel, π_k	π_n	F_T	F_F
1	$\boldsymbol{Q_1}$	$\boldsymbol{S}(x,y,l) \to \boldsymbol{A}(x,y,l)$		two	Ø
2_i	$\boldsymbol{Q_2}$	$\boldsymbol{A}(x,y,l) \to \boldsymbol{a}(\mathbf{x})\boldsymbol{A}(\boldsymbol{next}_i(x,l), y, f(l))$		two	3
3	$\boldsymbol{Q_3}$	$\boldsymbol{A}(x,y,l) \to \boldsymbol{a}(\mathbf{y})$		Ø	Ø

$\boldsymbol{V_T} = \{\boldsymbol{a}\}$
$\boldsymbol{V_N} = \{\boldsymbol{S}, \boldsymbol{A}\}$
$\boldsymbol{V_{PR}}$

Pred $= \{\boldsymbol{Q_1}, \boldsymbol{Q_2}, \boldsymbol{Q_3}\}$,

$\boldsymbol{Q_1}(x, y, l) = (MAP_{x,p}(y)=l) \quad (0 < l < n)$

$\boldsymbol{Q_2}(l) = (l \geq 1)$

$\boldsymbol{Q_3} = T$

Var $= \{x, y, l\}$

$F = Fcon \cup Fvar$,

$Fcon = \{f, \boldsymbol{next}_1, \dots, \boldsymbol{next}_n\}$ $(n=|X|)$,

$f(l)=l-1$, $D(f)=\mathbf{Z}_+\backslash\{0\}$

($\boldsymbol{next}_i$ is defined in fig. 5)

$Fvar = \{x_o, y_o, l_o, p\}$

$\boldsymbol{E} = \mathbf{Z}_+ \cup X \cup P$

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

$L = \{1,3\} \cup$ two, two$=\{2_1, 2_2, \dots, 2_n\}$

At the beginning of derivation:

$x=x_o$, $y=y_o$, $l=l_o$, x_o from X, y_o from X, l_o from $\mathbf{Z}_+$, p from P.

$next_i$ is defined as follows:

$D(next_i)= X \times \mathbf{Z}_+ \times X^2 \times \mathbf{Z}_+ \times P$

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

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

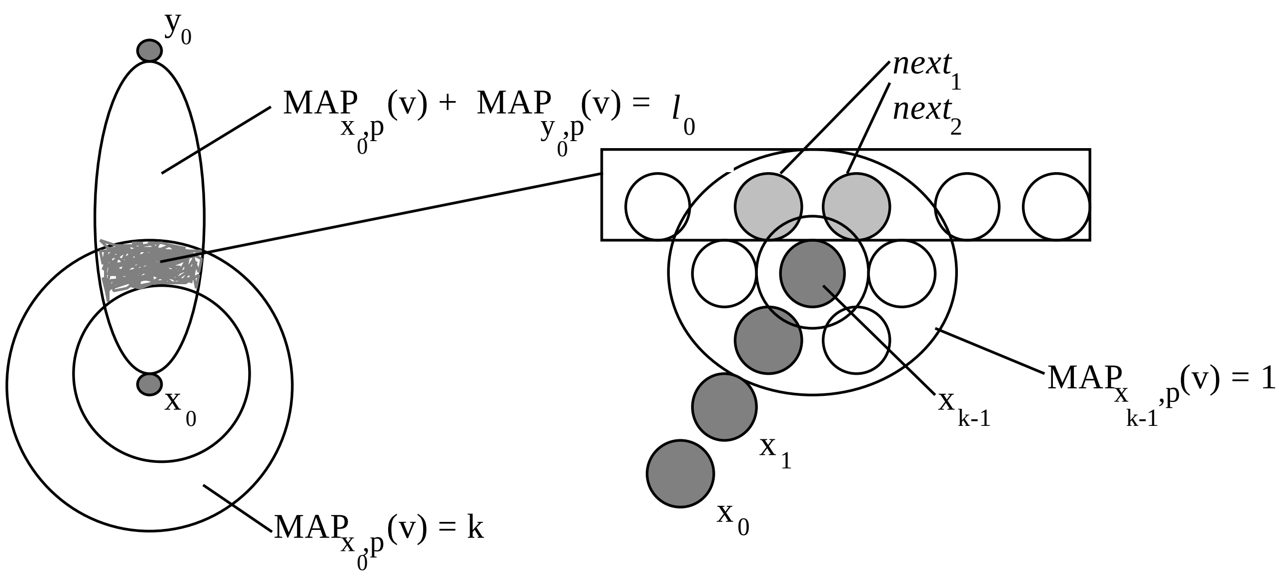


Fig. 6. Interpretation of the algorithm for $next_i$ for the grammar $\mathbf{G_t^{(1)}}$

THEOREM 1
The shortest trajectories from the point x to the point y of the length l_o for the element p on x (i.e., ON(p)=x) exist if and only if the distance of these points is equal l_o:

$$\mathbf{MAP_{x_o,p}(y_o)=l_o}, \qquad (11.1)$$

where $l_o<2n$, n is the number of points in X. If the relation R_p is symmetric, i.e., for all x from X, y from X and p from P $R_p(x, y)=R_p(y, x)$, then all the shortest trajectories $t_p(x_o, y_o, l_o)$ can be generated by the grammar $\mathbf{G_t^{(1)}}$.

Proof

We assume that t_o from $t_p(x_o,\ y_o,\ l_o)$ exists and is shortest. We shall prove (11.1). The proof is carried out by induction with respect to l_o.

In the case of l_o=1 the statement is easily verified.

We assume that for $l_o<m$ the statement is true.

Let $l_o=m$ and t_m from $t_p(x_o,\ y_o,\ m)$ be the shortest. We shall prove that $MAP_{x_o,p}(y_o)=m$. Let's consider the the *shortened* trajectory t_{m-1} from $t_p(x_o,\ x_{m-1},\ m\text{-}1)$, $t_{m-1}=a(x_o)a(x_1)\ldots a(x_{m-1})$, which is obtained from t_m after discarding the last symbol. If t_m from $t_p(x_o,\ x_m,\ m)$ is the shortest ($x_m=y_o$), then t_{m-1} is also shortest. But from the assumption it follows that $MAP_{x_o,p}(x_{m-1})=m\text{-}1$. From definition of MAP (see Section 5) it follows that x_{m-1} belongs to $M^{m-1}_{x_o,p}$. Since $R_p(x_{m-1}, y_o)$ is true, y_o belongs to $(\bigcup_{j=1}^{m-1} M^j_{x_o,p}) \cup M^m_{x_o,p}$. If y_o is from $\bigcup_{j=1}^{m-1} M^j_{x_o,p}$, then the trajectory t_m is not the shortest one, since there exists a trajectory t' from

$t_p(x_o,\ y_o,\ j)$ of length $j\leq m\text{-}1$. We have a contradiction. Thus, y_o belongs to $M^m_{x_o,p}$, i.e., $MAP_{x_o,p}(y_o)=m$.

Conversely, let (11.1) *be true.* Let's show that *there exists a trajectory belonging to* $t_p(x_o,\ y_o,\ l_o)$, and that *it is the shortest trajectory.*

The proof will be carried out by induction. For l_o=1 the statement is obvious. Let it be true for $l_o<m$.

Let now $l_o=m$ and $MAP_{x_o,p}(y_o)=m$. The shortest trajectory if exists can not be shorter then m, otherwise there exists $k_o<m$ such that $MAP_{x_o,p}(y_o)=k_o$ (from the direct statement proved above), and we have a contradiction.

Let us construct the shortest trajectory belonging to $t_p(x_o,\ y_o,\ m)$. By definition of MAP there exists x_{m-1} from $M^{m-1}_{x_o,p}$ such that $R_p(x_{m-1},\ y_o)$=T. But from the fact that x_{m-1} belongs to $M^{m-1}_{x_o,p}$, we have $MAP_{x_o,p}(x_{m-1})=m\text{-}1$. Consequently, according to the induction hypothesis, there exists the shortest trajectory $a(x_o)a(x_1)\ldots a(x_{m-1})$ of length m-1. In such a case the trajectory $a(x_o)a(x_1)\ldots a(x_{m-1})a(y_o)$ of length m will also be the shortest one.

To complete the proof of the theorem it remains for us to show that *all trajectories* $t_p(x_o,\ y_o,\ l_o)$ *are generated by the grammar* $G_t^{(1)}$ from Table 2, if R_p is symmetric. This grammar, in accordance with definition 9.1 of controlled grammars, belongs to the class of controlled grammars. Note that that the set of functional letters *Fvar* in it is a set of four zero-placed functions p, x_o, y_o, l_o, i.e., $G_t^{(1)}=G(p,\ x_o,\ y_o,\ l_o)$. It is obvious that each of the strings generated by $G_t^{(1)}$ is a trajectory from $t_p(x_o,\ y_o,\ l_o)$. Indeed, for each string $a(x_o)a(x_1)\ldots a(y_o)$ thus generated, the elements x_i belong to $ST_i(x_o)=M^i_{x_o,p}$ (see Fig. 6), consequently, this string is the shortest trajectory.

To prove that all the shortest trajectories are generated by $G_t^{(1)}$ let us conduct the following preliminary discussion. As was already mentioned above, all substrings of the shortest trajectory are the shortest trajectories with the beginning at x_o and ending at x_i (i=1, 2,..., l_o). Taking into account the symmetry of the relation R_p, all *reversed* substrings with the beginning at y_o and ending at x_i ($i=l_o\text{-}1,\ l_o\text{-}2,\ldots,1,\ 0$) will also be the shortest trajectories. Consequently, x_i belongs to $M^{l_o-i}_{y_o,p}$. This means that for any shortest trajectory $a(x_o)a(x_1)\ldots a(y_o)$ from $t_p(x_o,y_o,l_o)$

x_i belongs to the intersection of $M^{i}_{x_o,p}$ and $M^{l_o-i}_{y_o,p}$, i.e., $MAP_{x_o,p}(x_i)=i$ and $MAP_{y_o,p}(x_i)=l_o-i$, and, consequently,

$$MAP_{x_o,p}(x_i)+MAP_{y_o,p}(x_i)=l_o. \qquad (11.2)$$

Conversely, if for a certain x from X (11.2) takes place, then x necessarily enters into the set $\boldsymbol{P}(t_i)$ parametric values of at least one shortest trajectory t_i from $t_p(x_o, y_o, l_o)$. This follows from the fact that $MAP_{x_o,p}(x)\geq 0$ and $MAP_{y_o,p}(x)\geq 0$, while their sum is equal to l_o. That is to say, there exists j ($0\leq j\leq l_o$), such that $MAP_{x_o,p}(x)=j$, $MAP_{y_o,p}(x)=l_o-j$. Then there exist two shortest trajectories t^1 from $t_p(x_o, x, j)$ and t^2 from $t_p(y_o, x, l_o-j)$. The trajectory t^3 from $t_p(x, y_o, l_o-j)$ constructed of the same symbols as t_2, but in the reversed order, will also be the shortest trajectory. The concatenation of t^1 and t^2 gives the sought shortest trajectory containing x.

Thus, any element of the set X enters into the collection of parametric values $\bigcup_{t^i}\boldsymbol{P}(t^i)$ for all the shortest trajectories t_i from $t_p(x_o, y_o, l_o)$ if and only if (11.2) is true. These arguments lay a basis for the algorithm for calculating the function $next_i(x, l)$ (Fig. 6).

Next we shall use induction again. Obviously, the grammar of trajectories generates the first symbol $a(x_o)$ of all shortest trajectories from $t_p(x_o, y_o, l_o)$. Assume that it generates the m first symbols of any shortest shortest trajectory from $t_p(x_o, y_o, l_o)$. We shall show that it generates also the (m+1)st symbol $a(x_m)$.

We have: $MOVE(x_{m-1})$ is an intersection of $ST_1(x_{m-1})$, $ST_m(x_o)$ and SUM. Since $t_p(x_o, y_o, l_o)$ are the shortest trajectories, x_m belongs to $ST_m(x_o)=M^m_{x_o,p}$. But x_m also belongs to SUM, because of (11.2), and x_m belongs to $ST_1(x_{m-1})=M^1_{x_{m-1},p}$ since $R_p(x_{m-1}, x_m)=T$ by definition of trajectory. Thus, x_m belongs to $MOVE(x_{m-1})$, i.e., the (m+1)st symbol is generated by the grammar $G_t^{(1)}$.

The theorem is proved.

12. Generation of trajectories for the game of chess

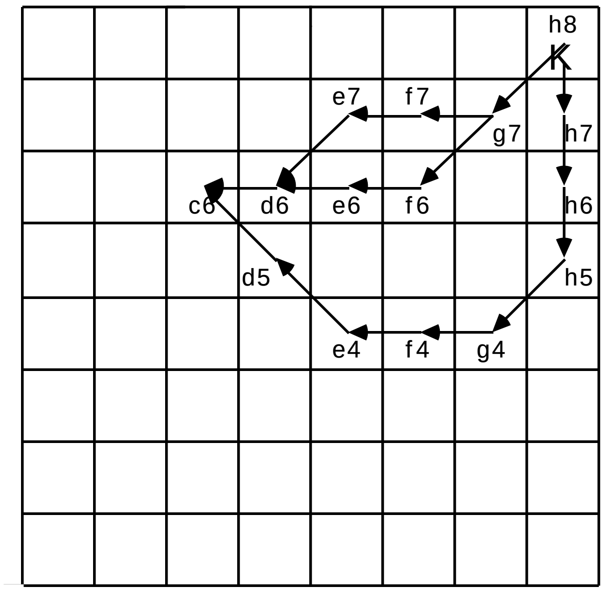


Fig. 7. Interpretation of the Language of Trajectories for the chess model.

The trajectory in this model is furnished by a string of symbols with parameters, i.e., the coordinates of the chessboard squares on which a piece must stand when moving from initial square to the ending one.

Three examples of trajectories for the King are shown in Fig. 7. The trajectories *a*(h8)*a*(g7)*a*(f6)*a*(e6)*a*(d6)*a*(c6) and *a*(h8)*a*(g7)*a*(f7)*a*(e7)*a*(d6)*a*(c6) are the *shortest trajectories* from t_{King}(h8, c6, 5) of the length 5. The trajectory
a(h8)*a*(h7)*a*(h6)*a*(h5)*a*(g4)*a*(f4)*a*(e4)*a*(d5)*a*(c6)
is an *admissible trajectory of degree* 2, because it consists of two shortest trajectories *a*(h8)*a*(h7)*a*(h6)*a*(h5)*a*(g4) and *a*(g4)*a*(f4)*a*(e4)*a*(d5)*a*(c6) tracing to the square g4 and from it.

Due to symmetry of the relation R_p in this model, $MAP_{x,p}(y)$ specifies the metric on X, and the theorem 1 of shortest trajectories holds (Section 11). The grammar $G_t^{(1)}$ (Table 3) generates the shortest trajectories for the movement of pieces on the empty chess board. Each trajectory is generated as a list of stopping squares; remember that some chess pieces can “jump”, e.i., to cross some squares without stop. While shortest trajectories for the Pawn, Knight, King can be of 6 or 7 moves length, for the long-range pieces these trajectories never exceed 2 moves, because such a piece can reach every square on the empty chessboard in 1 or 2 moves.

13. Generation of trajectories for robot control problem

Let us consider the robot control problem (Section 7). We shall show the computation of a planning path for the robot called K. Let us consider the derivation of the shortest trajectory from h8 to the point c6 for the robot K. Values of $MAP_{h8,K}$ are shown in Fig. 3. Thus the distance from h8 to c6 is equal to 5. Applying the grammar $G_t^{(1)}$ we have (symbol $\overset{l}{\longrightarrow}$ means application of the production with the label *l*):

$$\boldsymbol{S}(h8, c6, 5) \overset{1}{\longrightarrow} \boldsymbol{A}(h8, c6, 5) \overset{2_1}{\longrightarrow} \boldsymbol{a}(h8)\boldsymbol{A}(next_1(h8, 5), c6, 5)$$

Thus we have to compute MOVE (see the definition of the function $next_i$ from the grammar $G_t^{(1)}$). First we have to determine set SUM, i.e., we need to know values of $MAP_{h8,K}$ and $MAP_{c6,K}$ (shown in Fig. 8) on X. Adding these tables (Fig. 3 and 8) as matrices we can compute

$$SUM = \{v \mid v \text{ from } X, MAP_{h8,K}(v)+MAP_{c6,K}(v)=5\}$$

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

Fig. 8. Values of $MAP_{c6,K}$)

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

Fig. 9. Points of X which belong to SUM

The next step is the computation of $ST_1(h8)=\{v \mid v \text{ from } X, MAP_{h8,K}(v)=1\}$ which can be found from the table in Fig. 3. The result is shown in Fig. 10. In order to complete computation of

the set $MOVE_5(h8)$ we have to determine the following intersection:
$ST_1(h8)$, $ST_{5-5+1}(h8)=ST_1(h8)$ and SUM (see Fig. 11)

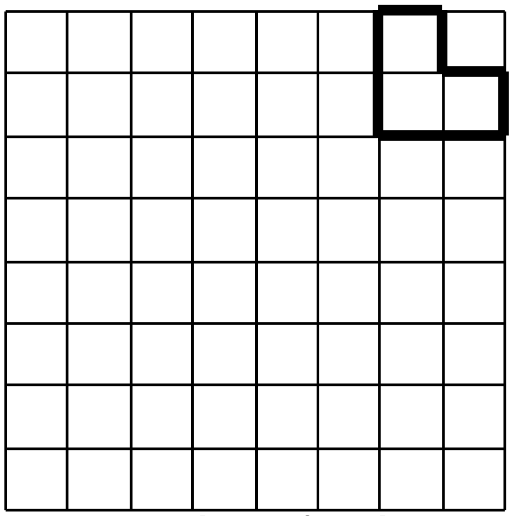


Fig. 10. The set of ST1(h8).

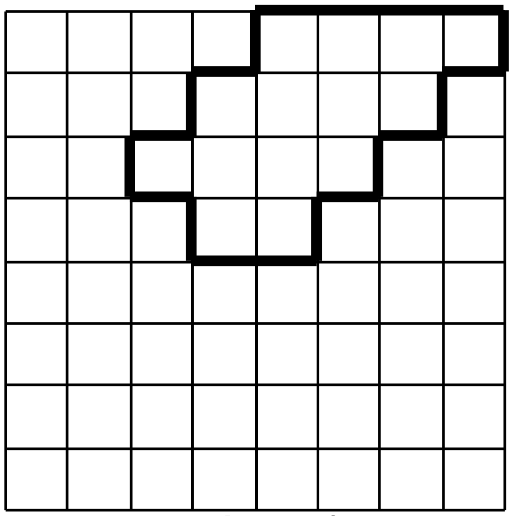


Fig. 11. The set of SUM.

Consequently, $MOVE_5(h8)=\{g7, g8\}$; and $next_1(h8, 5)=g7$, $next_2(h8, 5)=g8$. Since the number of different values of $next$ $_i$s equal to 2 (here r=2, see definition of the function *next*) we could branch at this step, i.e., apply productions 2_1 and 2_2 simultaneously, and continue both derivations independently. It can be accomplished in parallel computing environment. Let us proceed with the first derivation.

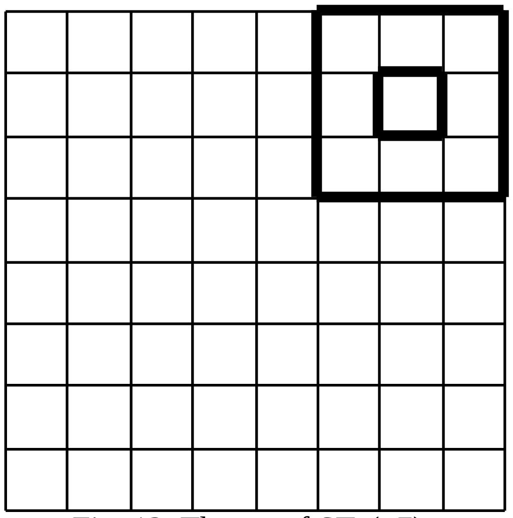


Fig. 12. The set of $ST_1(g7)$.

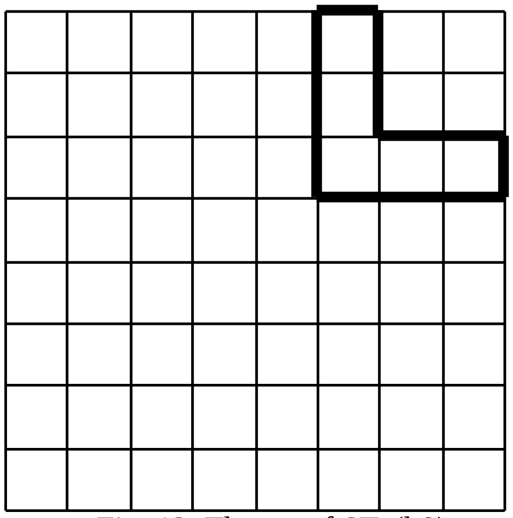


Fig. 13. The set of $ST_2(h8)$.

$$\boldsymbol{a}(h8)\boldsymbol{A}(g7, c6, 4) \xrightarrow{2_1} \boldsymbol{a}(h8)\boldsymbol{a}(g7)\boldsymbol{A}(next_1(g7, 4), c6, 3)$$

We have to compute $next_1(g7, 4)$. As on the preceding step have to determine $MOVE_4(g7)$. In order to do that we have to compute $ST_1(g7)=\{v \mid v \text{ from } X, MAP_{g7,K}(v)=1\}$ and $ST_{5-4+1}(h8)$ $=ST_2(h8)=\{v \mid v \text{ from } X, MAP_{h8,K}(v)=2\}$. The set of SUM is the same on all steps of the derivation. Hence, $MOVE_4(g7)$ is the intersection of the sets shown in Fig. 11, 12, 13, $MOVE_4(g7) = \{f6, f7, f8\}$; and

$$next_1(g7, 4) = f6;\ next_2(g7, 4) = f7;\ next_3(g7, 4) = f8.$$

Thus, the number of different values of the function *next* is equal to 3 (r=3), so the number of continuations of derivation should be multiplied by 3. Let us proceed with the first one:

$$\boldsymbol{a}(h8)\boldsymbol{a}(g7)\boldsymbol{A}(f6, c6, 3) \overset{2}{\underset{1}{\longrightarrow}} \ldots$$

This way eventually we will derive one of the shortest trajectories for the robot K from h8 to c6:

$$\boldsymbol{a}(h8)\boldsymbol{a}(g7)\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{a}(d5)\boldsymbol{a}(c6).$$

14. Generation of trajectories for scheduling problem

Following Sections 8.1-8.3 we can construct a Hierarchy of Languages for this problem. Here we consider several examples from the Language of Trajectories. As an example of *shortest trajectory* here we have

$$t_{p_2}((0, 0, p_2), (g, 0, p_2), 3)=\boldsymbol{a}(0, 0, p_2)\boldsymbol{a}(1, 0, p_2)\boldsymbol{a}(2, 0, p_2)\boldsymbol{a}(g, 0, p_2),$$

i.e., the unit p_2 being taken out for maintenance on the first day and this maintenance was completed on the second day. Another example:

$$t_{p_{res}}((1, 0, r), (1, 0, q_2), 2)=\boldsymbol{a}(1, 0, r)\boldsymbol{a}(1, 1, r)\boldsymbol{a}(1, 0, q_2),$$

The trajectory of the elements of power reserve, the pyramid P_1^{res}, is targeted to liquidate the elements of the pyramid Q_{11}^{fall}.Of course, movement along this trajectory will not be included into the optimal variant of the System: the opposite side (elements from Q_{11}^{fall}) would not be waiting for being captured on the point $(1, 0, q_2)$; after the

$$\text{TRANSITION}(p_{res}, (1, 0, r), (1, 1, r))$$

one of the opposite elements q_{fall} should move to the point (1, 1, r) —

$$\text{TRANSITION}(q_{fall}, (1, 0, q_2), (1, 1, r)),$$

and remove the element p_{res}, starting the exchange.

The trajectory

$$t_{p_2}((0, 0, p_2), (g, 0, p_2), 4)=\boldsymbol{a}(0, 0, p_2)\boldsymbol{a}(0, 1, p_2)\boldsymbol{a}(1, 1, p_2)\boldsymbol{a}(2, 1, p_2)\boldsymbol{a}(g, 0, p_2)$$

can serve as an example of an *admissible trajectory of degree* 2. The movement along this trajectory corresponds to the variant of the plan with unit p_2 being taken out for maintenance on the second day.

Here show a grammar of shortest trajectories which is the same as $G_t^{(1)}$ (Table 2) except function *next*. In view of complexity we do not present here the definition of the function *next* for the general case. Note only that here it does not depend on *i*. For example, in Fig. 4 function $next_{u_o,p}(x, l)$for all (x, l) from $X \times \mathbf{Z}_+$ is given by the arrows of the network.

15. Geometrical constructions: obstacles and roundabout trajectories

Consider the following controlled grammar for the Complex System with *symmetric* relation of reachability R_p (Table 3, 4):

THEOREM 2

All the admissible trajectories $t_p(x_o, y_o, l_o)$ of degree 2 from the point x to the point y of the length l_o for the element p on x can be generated by the grammar $\mathbf{G_t^{(2)}}$.

Table 3
A grammar of shortest and admissible trajectories $G_t^{(2)}$)

L	Q	**Kernel,** π_k	π_n	F_T	F_F
1	Q_1	$S(x,y,l) \rightarrow A(x,y,l)$		*two*	ø
2_i	Q_2	$A(x,y,l) \rightarrow A(x,\ med_i(x,y,l),\ lmed_i(x,y,l))$ $A(med_i(x,y,l),\ y,\ l-lmed_i(x,y,l))$		*three*	*three*
3_j	Q_3	$A(x,y,l) \rightarrow a(x)A(next_j(x,l),\ y,\ f(l))$		*three*	4
4	Q_4	$A(x,y,l) \rightarrow a(y)$		*three*	5
5	Q_5	$A(x,y,l) \rightarrow e$		*three*	ø

$V_T = \{a\}$,
$V_N = \{S, A\}$,
V_{PR}

Pred $=\{Q_1, Q_2, Q_3, Q_4, Q_5\}$

$Q_1(x, y, l) = (MAP_{x,p}(y) \leq l < 2MAP_{x,p}(y)) \wedge (l < 2n)$
$Q_2(x, y, l) = (MAP_{x,p}(y) \neq l)$
$Q_3(x, y, l) = (MAP_{x,p}(y) = l) \wedge (l \geq 1)$
$Q_4(y) = (y = y_o)$
$Q_5(y) = (y \neq y_o)$

Var $= \{x, y, l\}$;
Con $= \{x_o, y_o, l_o, p\}$;
Func $=$ *Fcon* U *Fvar*;

Fcon $=\{f,\ next_1, \ldots, next_n,\ med_1, \ldots,\ med_n,$
$lmed_1, \ldots,\ lmed_n\}$ $(n=|X|)$,
$f(l)=l-1$, $D(f)=\mathbf{Z}_+ \setminus \{0\}$
functions $next_i$, med_i *and* $lmed_i$ are defined in Table 4.

Fvar $=\{x_o, y_o, l_o, p\}$

$E = \mathbf{Z}_+$ U X U P is the subject domain;
Parm: $S \rightarrow$ *Var*, $A \rightarrow$ *Var*, $a \rightarrow \{x\}$;
$L = \{1,4\}$ U *two* U *three*, *two* $=\{2_1, 2_2, \ldots, 2_n\}$, *three* $=\{3_1, 3_2, \ldots, 3_n\}$

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.

Table 4
A definition of functions *med, lmed, next*

med_i is defined as follows:

```
        $D(med_i)$= X x X x $\mathbf{Z}_+$ x P
        DOCK={v | v from X, $MAP_{x_o,p}(v)+MAP_{y_o,p}(v)=l$},
        If
          $DOCK_l(x)=\{v_1, v_2, \ldots, v_m\} \neq \emptyset$
          then
          $med_i(x, y, l)=v_i$ for $1 \leq i \leq m$ and
          $med_i(x, y, l)=v_m$ for $m < i \leq n$,
        otherwise
          $med_i(x, y, l)=x$.
```

$lmed_i$ is defined as follows:
$D(med_i)$= X x X x $\mathbf{Z}_+$ x P
$lmed_i(x, y, l)=MAP_{x,p}(med_i(x, y, l))$

$next_i$ is defined as follows:

```
        $D(next_i)$= X x $\mathbf{Z}_+$ x $X^2$ x $\mathbf{Z}_+$ x P
        SUM={v | v from X, $MAP_{x_o,p}(v)+MAP_{y_o,p}(v)=l_o$},
        $ST_k(x)$={v | v from X, $MAP_{x,p}(v)=k$},
        $MOVE_l(x)$ is an intersection of the following sets:
$ST_1(x)$, $ST_{l_o-l+1}(x_o)$ and SUM.
        If
          $MOVE_l(x)=\{m_1, m_2, \ldots, m_r\} \neq \emptyset$
          then
          $next_i(x, l)=m_i$ for $i \leq r$ and
          $next_i(x, l)=m_r$ for $r < i \leq n$,
        otherwise
          $next_i(x, l)=x$.
```

Proof

It is obvious that if l_o is such that the WFF $\boldsymbol{Q_3}$ is true then all the shortest trajectories from x_o to y_o are generated by productions with labels 1, 3_j, and 4, 5 (Tables 3, 4).

It remains for us to consider the case where the WFF $\boldsymbol{Q_2}$ is true, i.e., the case of non-shortest admissible trajectories of degree 2 and length l_o. Let $t_o=\boldsymbol{a}(x_o)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(y_o)$ be a non-shortest admissible trajectory of degree 2 and length l_o. Then there exists x_m $P(t_o)$ such that $\boldsymbol{a}(x_o)\ldots\boldsymbol{a}(x_m)$ and $\boldsymbol{a}(x_m)\ldots\boldsymbol{a}(y_o)$ are the shortest trajectories. According to Theorem 1, $MAP_{x_o,p}(x_m)=m$, and $MAP_{x_m,p}(y_o)=l_o-m$. Then in fact $MAP_{y_o,p}(x_m)=l_o-m$, hence

$$MAP_{x_o,p}(x_m)+MAP_{y_o,p}(x_m)=l_o.$$

Thus, x_m DOCK. Consequently, there exists v_i from DOCK, $v_i=x_m$. Thus, one of the productions with labels from the set *two*, e.g. 2_i, is sure to be applied in the derivation process. It means that the “attaching point” a(xm) of two shortest trajectories will be generated. These shortest trajectories themselves, however, will be generated by application of the productions 1, 3_j and 4, 5 in accordance with Theorem 1.

Conversely, let the grammar from Table 3 generates a certain string of symbols with parameters. From the definition of the function $next_i$(x, l) it follows that this string is a trajectory. If in the derivation process we did not apply productions with labels from *two*, even once, then from Theorem 1 it follows that the given trajectory is the *shortest* trajectory. We now assume that these productions were applied. From Table 3 it follows that such an application was made once only: a production with the label 2_i can be used only after a production with label 1, which is used first and once only; if set of *two* contains more than one element, derivation of several strings takes place simultaneously. Thus, a single application of a production with the label 2_i generates the following nonterminal symbols (Tables 3, 4):

$$\boldsymbol{A}(\text{ x, } med_i\,(\text{x, y, } l),\ lmed_i(\text{x, y, } l))\boldsymbol{A}(med_i\,(\text{x, y, } l),\ \text{y, } l—lmed_i(\text{x, y, } l)),$$

where for v_i=med_i (x, y, l) we have $\text{MAP}_{\text{xo,p}}(\text{v}_i)+\text{MAP}_{\text{yo,p}}(\text{v}_i)=l\}$. Consequently, $\boldsymbol{a}(v_i)$ is an "attaching point" of two shortest trajectories whose total length equals l_o.

The theorem is proved.

16. Examples of trajectories in case of visible and invisible obstacles

For the implementation of the chess model considered in [4] for all pieces, except Queen, admissible trajectories of degree 2 are generated. It means that for short-range pieces such trajectories can reach 12-14 moves, while for long-range pieces they do not exceed 4 moves. Thus, the implementation of the Hierarchy of Languages for this model is based on admissible trajectories of degree 2 (except for the Queen). In order to utilize all the capabilities of this mobile piece we have to generate admissible trajectories of degrees 2, 3 and 4. In practice, it does not mean the increase of the length of the trajectories considered: Queen's trajectories never exceed 4 moves. The point is that all the admissible trajectories of the other long-range pieces of the length less or equal 4 are admissible trajectories of degree 2 only. The generation of admissible trajectories of higher degrees for the Queen was implemented by modification of the grammar $G_t^{(2)}$ (Tables 3,4) [4].

In order to make our discussion about grammars generating admissible trajectories transparent, let us consider in detail application of such a grammar to generating trajectories of the elements of the Complex System in cases of visible and invisible obstacles. The difference between these types of obstacles is as follows. Visible obstacles can be considered beforehand and represented as restricted areas (Fig. 14). Invisible obstacles display themselves only during the motion of elements along the trajectories, and after being encountered, require new (usually longer) trajectories to be generated and examined.

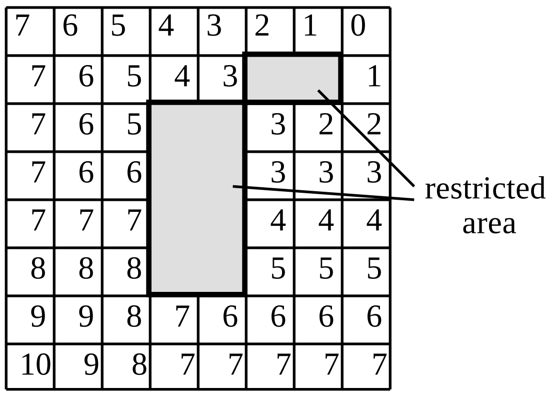


Fig. 14. Values of $\text{MAP}_{\text{h8,K}}$

We shall apply this grammar for generating trajectories for the robot from the point h8 to c6 (Fig. 14). The motion space for this robot is the square table of 8x8 with the restricted area. Let us consider the derivation of the shortest trajectory from h8 to the point c6 for the robot K. Values of

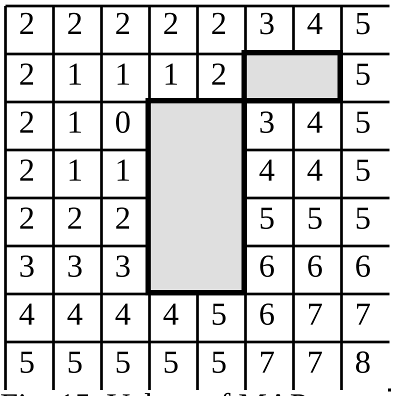
$MAP_{h8,K}$ are shown in Fig. 14. The restricted area shown in Fig. 14 represents visible obstacles. Thus, the distance from h8 to c6 is equal to 5. Applying the grammar $G_t^{(2)}$ we have (symbol $\xrightarrow{l}$ means application of the production with the label *l*):

$$\boldsymbol{S}(h8, c6, 5) \overset{1}{\longrightarrow} \boldsymbol{A}(h8, c6, 5) \overset{3_1}{\longrightarrow} \boldsymbol{a}(h8)\boldsymbol{A}(next_1(h8, 5), c6, 5)$$

Thus we have to compute MOVE (see definition of the function $next_i$ from the grammar $G_t^{(2)}$). First we have to determine set SUM, i.e., we need to know values of $MAP_{h8,K}$ and $MAP_{c6,K}$ (shown in Fig. 15) on X. Adding these tables (Fig. 14 and Fig. 15) as matrices we compute

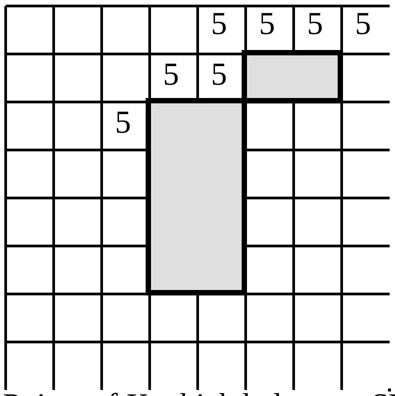
$$SUM = \{v \mid v \text{ from } X, MAP_{h8,K}(v)+MAP_{c6,K}(v)=5\}.$$

It is shown in Fig. 16.



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

Fig. 15. Values of $MAP_{c6,K}$



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

Fig. 16.Points of X which belong to SUM

The next step is the computation of $ST_1(h8)=\{v \mid v \text{ from } X, MAP_{h8,K}(v)=1\}$ which can be found from the Fig. 14. A result is the only point g8. In order to complete computation of the set $MOVE_5(h8)$ we have to determine the following intersection:

$$ST_1(h8),\ ST_{5-5+1}(h8)=ST_1(h8) \text{ and SUM (Fig. 16)}$$

Consequently, $MOVE_5(h8)=\{g8\}$; and $next_1(h8, 5)=g8$. Since the number of different values of *next* is equal to 1 we can not branch here. Let us proceed with the derivation.

$$\boldsymbol{a}(h8)\boldsymbol{A}(g8, f8, 4) \overset{2_1}{\longrightarrow} \boldsymbol{a}(h8)\boldsymbol{a}(g8)\boldsymbol{A}(next_1(g8, 4), c6, 3)$$

We have to compute $next_1(g8, 4)$. Obviously $next_1(g8, 4)=f8$.

$$\boldsymbol{a}(h8)\boldsymbol{a}(g8)\boldsymbol{A}(f8, c6, 3) \overset{2_1}{\longrightarrow} \boldsymbol{a}(h8)\boldsymbol{a}(g8)\boldsymbol{a}(f8)\boldsymbol{A}(next_1(f8, 3), c6, 2)$$

As on the preceding step have to determine $MOVE_3(f8)$. In order to do that we have to compute $ST_1(f8)=\{v \mid v \text{ from } X, MAP_{f8,K}(v)=1\}$ (Fig. 17) and $ST_{5-3+1}(h8)=ST_3(h8)=\{v \mid v \text{ from } X, MAP_{h8,K}(v)=3\}$ (Fig. 18).

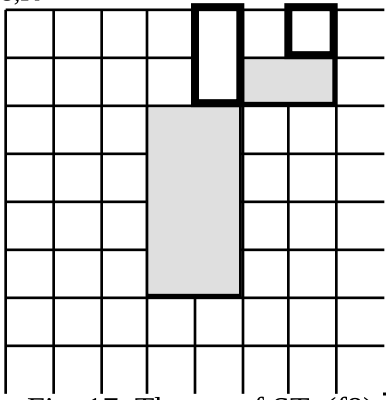


Fig. 17. The set of $ST_1(f8)$

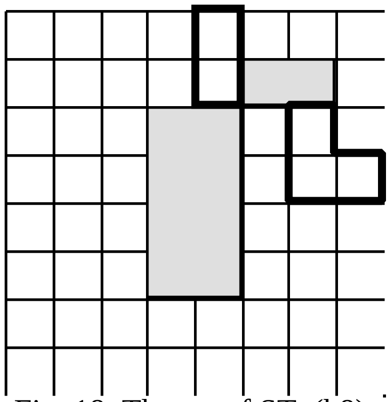


Fig. 18. The set of $ST_3(h8)$

The set of SUM is the same on all steps of the derivation. Hence, $MOVE_3(f8)$ is the intersection of the sets shown in Fig.16, 17, 18, $\mathrm{MOVE}_4(\mathrm{f8}) = \{\mathrm{e7}, \mathrm{e8}\}$; and

$$next_1(\mathrm{f8}, 3) = \mathrm{e7};\ next_2(\mathrm{f8}, 3) = \mathrm{e8}.$$

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; there exist two shortest trajectories. This way, eventually, we will derive both of the shortest trajectories for the robot K from h8 to c6:

$$\boldsymbol{a}(\mathrm{h8})\boldsymbol{a}(\mathrm{g8})\boldsymbol{a}(\mathrm{f8})\boldsymbol{a}(\mathrm{e7})\boldsymbol{a}(\mathrm{d7})\boldsymbol{a}(\mathrm{c6})$$

and

$$\boldsymbol{a}(\mathrm{h8})\boldsymbol{a}(\mathrm{g8})\boldsymbol{a}(\mathrm{f8})\boldsymbol{a}(\mathrm{e8})\boldsymbol{a}(\mathrm{d7})\boldsymbol{a}(\mathrm{c6}).$$

Let us assume that after examining these shortest paths robot found out that both of them are not satisfactory because of invisible obstacles. So our robot is looking for different paths: they can be longer trajectories, only. Applying the grammar $G_t^{(2)}$ with l=6 we have

$$\boldsymbol{S}(\mathrm{h8}, \mathrm{c6}, 6) \xrightarrow{1} \boldsymbol{A}(\mathrm{h8}, \mathrm{c6}, 6)$$

Then we try to apply one of the productions 2_i, check $\boldsymbol{Q_2}$=($\mathrm{MAP}_{\mathrm{h8,K}}(\mathrm{c6})$=5), i.e., $\boldsymbol{Q_2}\neq 6$. Hence

$$\boldsymbol{A}(\mathrm{h8}, \mathrm{c6}, 6) \xrightarrow{2_1} \boldsymbol{A}(\mathrm{h8}, med_i(\mathrm{h8}, \mathrm{c6}, 6),\ lmed_i(\mathrm{h8}, \mathrm{c6}, 6))$$
$$\boldsymbol{A}(med_i(\mathrm{h8}, \mathrm{c6}, 6), \mathrm{c6}, 6\text{-}lmed_i(\mathrm{h8}, \mathrm{c6}, 6))$$

Thus we have to compute med_i and $lmed_i$ (the definitions of these functions are included into the definition of the grammar $G_t^{(2)}$). First we have to determine the set of DOCK, i.e., we have to know values of $\mathrm{MAP}_{\mathrm{h8,K}}$ and $\mathrm{MAP}_{\mathrm{c6,K}}$ (shown in Fig. 14, 15) on X. Adding these tables as matrices we compute

$$\mathrm{DOCK} = \{v \mid v \text{ from X}, \mathrm{MAP}_{\mathrm{h8,K}}(v)+\mathrm{MAP}_{\mathrm{c6,K}}(v)=6\}.$$

It is shown in Fig. 19. This set represents the attaching points of admissible trajectories

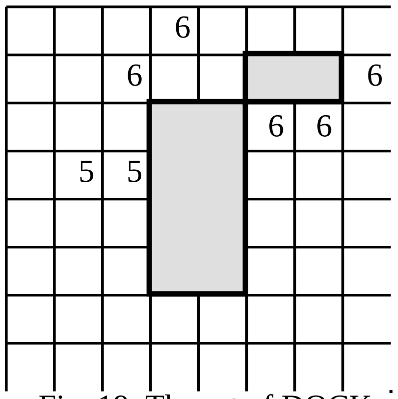


Fig. 19. The set of DOCK

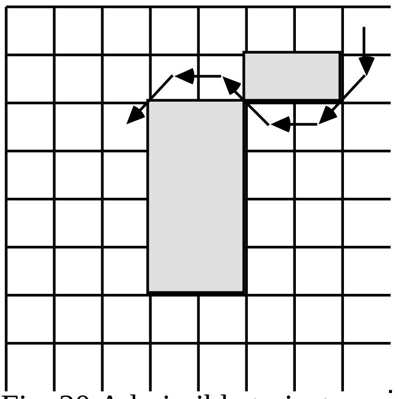


Fig. 20.Admissible trajectory from h8 to c6 through h7

of degree 2. There are five attaching points: DOCK={c7, d8, f6, g6, h7}, and

$$med_1(\mathrm{h8}, \mathrm{c6}, 6)=\mathrm{c7},\ med_2(\mathrm{h8}, \mathrm{c6}, 6)=\mathrm{d8},\ med_3(\mathrm{h8}, \mathrm{c6}, 6)=\mathrm{f6},$$
$$med_4(\mathrm{h8}, \mathrm{c6}, 6)=\mathrm{g6},\ med_5(\mathrm{h8}, \mathrm{c6}, 6)=\mathrm{h7}.$$

Thus, the number of different values of the function med is equal to 5 (m=5), so the number of continuations of derivation should be multiplied by 5; there exist five bundles of admissible trajectories coming through points c7, d8, f6, g6, and h7. Some of the may be the same. Let us continue the derivation of the trajectories coming through h7. According to definition of *lmed* and $\mathrm{MAP}_{\mathrm{h8,K}}$ (Fig. 14):

$$lmed_5(\mathrm{h8}, \mathrm{c6}, 6)= \mathrm{MAP}_{\mathrm{h8,K}}(med_5(\mathrm{h8}, \mathrm{c6}, 6))=1;$$
$$l\text{–}lmed_5(\mathrm{h8}, \mathrm{c6}, 6)=5.$$

It means that shortest trajectories forming admissible trajectories from h8 to c6 through h7 are as follows: the trajectories which come from h8 to h7 are of length 1, and from h7 to c6 — are of length 5. Proceeding with the derivation we have:

$$\boldsymbol{A}(\mathrm{h8}, \mathrm{c6}, 6) \xrightarrow{2_1} \boldsymbol{A}(\mathrm{h8}, \mathrm{h7},\ 1)\boldsymbol{A}(\mathrm{h7}, \mathrm{c6}, 5),$$

i.e., each of nonterminals $\boldsymbol{A}$ corresponds to the bundle of shortest trajectories coming to and from the attaching point h7. Next we apply one of the productions 3_j:

$$\boldsymbol{A}(\text{h8, h7, 1})\boldsymbol{A}(\text{h7, c6, 5}) \xrightarrow{3_j} \boldsymbol{a}(\text{h8})\boldsymbol{A}(next_j(\text{h8, 1}), \text{h7, 0})\boldsymbol{A}(\text{h7, c6, 5})$$

Obviously, the number of different values of *next* is equal to 1 and $next_1$(h8, 1) = h7:

$$\boldsymbol{a}(\text{h8})\boldsymbol{A}(\text{h7, h7, 0})\boldsymbol{A}(\text{h7, c6, 5}).$$

Then we try to apply production 3_1 again to nonterminal $\boldsymbol{A}$ (h7, h7, 0), but fail because $\boldsymbol{Q_3}$(h7, h7, 0)=F (l=1), we go to the production 4. Due to y=h7, i.e. y≠y_o (y_o=c6), $\boldsymbol{Q_4}$=F, and we fail applying this production. So according to the set of$\boldsymbol{F_F}$ we go to the production 5:

$$\boldsymbol{a}(\text{h8})\boldsymbol{A}(\text{h7, h7, 0})\boldsymbol{A}(\text{h7, c6, 5}) \xrightarrow{5} \boldsymbol{a}(\text{h8})\boldsymbol{A}(\text{h7, c6, 5})$$

Next we apply one of the productions 3_j:

$$\boldsymbol{a}(\text{h8})\boldsymbol{A}(\text{h7, c6, 6), c6, 5}) \xrightarrow{3_j} \boldsymbol{a}(\text{h8})\boldsymbol{a}(\text{h7})\boldsymbol{A}(next_j(\text{h7, c6, 5}), \text{c6, 4}).$$

Thus, one of the shortest components is done. This is the trajectory $\boldsymbol{a}$(h8)$\boldsymbol{a}$(h7).
Obviously, function $next_j$(h7, c6, 4) yields only one value g6, and proceeding with derivation we have:

$$\boldsymbol{a}(\text{h8})\boldsymbol{a}(\text{h7})\boldsymbol{A}(next_1(\text{h7, c6, 4}), \text{c6, 5}) = \boldsymbol{a}(\text{h8})\boldsymbol{a}(\text{h7})\boldsymbol{A}(\text{g6, c6, 4}).$$

At the conclusion of derivation we will get the admissible trajectory (Fig. 20):

$$\boldsymbol{a}(\text{h8})\boldsymbol{a}(\text{h7})\boldsymbol{a}(\text{g6})\boldsymbol{a}(\text{f6})\boldsymbol{a}(\text{e7})\boldsymbol{a}(\text{d7})\boldsymbol{a}(\text{c6}).$$

This trajectory is quite different of the trajectories generated earlier. It might be free of invisible obstacles. If it is not true, robot K should generate longer trajectories and try to move along them.

17. A discussion of results

This paper reports the results on investigation of geometrical properties of complex systems. It explores properties of the first-level subsystems, paths of elements, so-called trajectories. These results are considered as contribution to the Surface Linguistic Geometry. The impact of these results on the applications was considered in details for the chess game, autonomous robot control and maintenance scheduling problem.

The investigation resulted in definition of a distance function between two points of the system as a “length of the shortest path between these points”. It is very interesting that distances between the same two points are different for different elements of the system. It takes place because usually paths for different elements are different, i.e., moving capabilities of different robots as well as maintainability of different power units are different.

The distance measurement allowed us to build the general formal grammar generating all the shortest paths between two points for the given element of the system, the shortest trajectories. Moreover there was proved the theorem which gives necessary and sufficient conditions for existence of a path (trajectory) between two points (for the given element); if such path does exist the theorem shows the actual length of the shortest path and confirms that grammar $G_t^{(1)}$ generates all the shortest paths. Analogous results were obtained in case of obstacles: visible and invisible. In this case so-called “admissible trajectories of degree 2”, i.e., constructed of two shortest ones, can be generated by the $G_t^{(2)}$ grammar to go around the obstacles. The application of the Surface Linguistic Geometry to the game of chess, robot control, and maintenance scheduling allowed for efficient implementation of the Language of Trajectories in these models.

The same generating tools can be used to generate higher level subsystems, the networks of paths, i.e., the Language of Trajectory Nets [20, 23, 24]. Even the Language of Translations [20, 24] describing the process of search of a complex system can be generated by a similar type of grammars. Consequently, the investigation of the control of the search for an optimal operation of the complex system can be reduced to the investigation of properties of the specific formal grammars.

Subsequent studies of geometrical properties of trajectory networks should allow us formally and constructively to describe the adjustment of the whole hierarchy, i.e., the difference between the representations of two states, while the system moves from one state to another one in the process of the search. An efficient and constructive description of the hierarchy adjustment is very important for the design of efficient applications in different fields.

This research began in Moscow, USSR, continued at McGill University, Montreal, Canada, and currently proceeds at the University of Colorado at Denver. The previous stages of this research were mainly funded in the former USSR by the USSR National Committee for Science and Technology and the USSR Department of Energy. In Canada funding was provided jointly by McGill University, Montreal, and NSERC of Canada.

References

[1] M.R. Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness* (W.H.Freeman and Co., San Francisco, 1979).
[2] H. Simon, *The Sciences of the Artificial* (The MIT Press, Cambridge, MA, 1969).
[3] M.D. Mesarovich, D. Macko, Y. Takahara Y.,*Theory of Hierarchical Multilevel Systems,* (Academic Press, New York, 1970).
[4] M.M. Botvinnik, *Computers in Chess: Solving Inexact Search Problems. Springer Series in Symbolic Computation* (Springer-Verlag, New York , 1984).
[5] N. Chomsky, Formal Properties of Grammars, in *Handbook of MathematicalPsychology*, ed. R. Luce, R. Bush, E. Galanter., vol. 2, 323-418 (John Wiley & Sons, New York, 1963).
[6] S. Ginsburg, *The Mathematical Theory of Context-Free Languages* (McGraw Hill, New York, 1966)
[7] D.E. Knuth, Semantics of Context-Free Languages, Mathematical Systems Theory, 2-2 (1968) 127–146.
[8] D.J. Rozenkrantz, Programmed Grammars and Classes of Formal Languages, J. of the ACM, 16-1 (1969) 107–131.
[9] K.S. Fu, *Syntactic methods in pattern recognition, Mathematics in Science and Engineering,* Vol. 112, edited by Richard Bellman (Academic Press, New York, 1974).
[10] K.S. Fu, *Digital pattern recognition* (Springer-Verlag, New York, 1980).
[11] R.N. Narasimhan, Syntax–Directed Interpretation of Classes of Pictures, Comm. of the ACM, 9 (1966) 166–173.
[12] T. Pavlidis, Linear and Context-Free Graph Grammars, J. of the ACM, 19 (1972) 11-22.
[13] A.C. Shaw, A Formal Picture Description Scheme as a Basis for Picture Processing System, Information and Control, 19 (1969) 9–52.
[14] J. Feder, Plex languages, Information Sciences, 3 (1971) 225-241.
[15] J.L. Pfaltz and A. Rosenfeld, WEB grammars, in *Proc. of the 1st International Joint Conf. Artificial Intelligence* (Washington, D.C., May 1969) pp. 609–619.
[16] N.G. Volchenkov, The Interpreter of Context-Free Controlled Parametric Programmed Grammars, in: Cybernetics Problems. Intellectual Data Banks, ed. L.T. Kuzin (The USSR Academy of Sciences, Moscow, 1979) pp. 147–157 [in Russian].
[17] R.E. Fikes and N.J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, Artificial Intelligence, 2 (1971) 189–208.
[18] E.D. Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, Artificial Intelligence, 5-1 (1974) 115-135.
[19] J. McCarthy and P.J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, Machine Intelligence, 4 (1969) 463–502.
[20] B. Stilman, Introduction to Linguistic Geometry and Applications, Technical Rep. TR–12, Dept. of Computer Science, University of Colorado at Denver, Denver, CO (March, 1992)
[21] B. Stilman, Hierarchy of Formal Grammars for Solving Search Problems, in: *Proceedings of the International Workshop, Artificial Intelligence. Results and Prospects* (Moscow, 1985) 63–72,[in Russian].
[22] A.I. Reznitskiy and B.M. Stilman, Use of Method PIONEER in Automating the Planning of Maintenance of Power-Generating Equipment, Automatics and Remote Control, 11 (1983)

147-153, [in Russian].
[23] B. Stilman, A Syntactic Structure for Complex Systems, *Proc. of the Second Golden West International Conf. Artificial Intelligence* (Reno, NE, June 1992).
[24] B. Stilman, A Linguistic Geometry of Complex Systems, Annals of Mathematics and Artificial Intelligence (1992) (submitted; extended version of paper presented at the 2nd Int. Symposium on AI and Math., Ft. Lauderdale, FL, Jan. 1992).
[25] B. Stilman, A Linguistic Approach to Geometric Reasoning, Technical Rep. TR-19, Dept.of Computer Science, University of Colorado at Denver, Denver (July, 1992).