NETWORK LANGUAGES FOR COMPLEX SYSTEMS

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—In this paper we describe a research on two-dimensional geometrical structures for complex systems to accomplish applications to robotics, process planning and control, scheduling, decision support, etc. This research includes the development of tools for *representation* and *reasoning* about hierarchical networks of paths that are inherent for complex real-world systems. It relies on the formalization of *search heuristics* of high-skilled human experts which have resulted in the development of successful applications in different areas. The proposed 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* employing the first-order predicate calculus. A performance of implementations is considered in detail.

1. Introduction

Many important practical problems can be considered as optimization problems for complex systems. 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, sometimes exponentially, as a function of dimension [1]. Thus most real-world search problems are not solvable employing exact algorithms in a reasonable amount of time.

A development of approximate algorithms for such problems is a necessity. 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]. These ideas have been implemented for many problems with varying degrees of success, but each implementation was unique. There was no general approach for such implementations. Each new problem must be carefully studied and previous experience usually can not be applied. On the other hand, every attempt to evaluate the computational complexity and quality of a pilot solution requires implementing its program, which in itself is a unique task for each problem.

Here we consider a formal, general approach for a certain class of search problems 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 have 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 proposed approach permits us to study the secondary multi-level system formally, evaluate the complexity and quality of solutions, improve them, if necessary, and generate computer programs for 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 final version was published as Stilman, B., Network Languages for Complex Systems, An International Journal: Computers & Mathematics with Applications, Vol. 26, No. 8, pp. 51-80, 1993.

the same tools to the new problem domain. It actually looks like an application of the methods of a chess expert to a maintenance scheduling problem and vice versa. But what about guaranties of success? The guaranties reside in deeper studies of these methods, in the discovery of inner properties which brought us to a success in a certain class of complex systems.

The main idea of the approach considered in this paper is as follows. 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.

To keep track of the origin of this approach, let us refer to the ideas of syntactic methods of pattern recognition developed by Fu [3, 4], Narasimhan [5], and Pavlidis [6], and picture description languages by Shaw [7], Feder [8], Rosenfeld, Phaltz [9, 29]. The idea of linguistic representation of complex real-world and artificial images was transformed into the idea of similar representation of complex hierarchical systems. 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. An origin of such languages can be traced back to the origin of SNOBOL-4 and the research on programmed formal grammars and languages by Knuth [10], Rozenkrantz [11], and Volchenkov [12]. A mathematical environment for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Nilsson, Fikes [13], Sacerdoti [14], and McCarthy, Hayes [15] 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 hierarchical system be sufficiently complex, poorly formalized, and has successful applications in different areas. The chosen informal model was developed and applied to scheduling, planning, and computer chess by Botvinnik, Stilman, and others [16].

An application of the hierarchy of languages to the chess model was implemented in full as program PIONEER [16]. For power equipment maintenance the hierarchy was implemented in a number of computer programs being used for maintenance scheduling all over the USSR [17, 18, 19].

2. A Discussion of Experimental Results

In order to evaluate the experiments with implementations we present here a brief discussion about search algorithms. We look for approximate algorithms that reduce B, the branching factor [20], especially, those algorithms which make B close to 1. Such algorithms should be considered as extremely goal-driven with *minimal branching to different directions*.

Different search algorithms were designed in order to reduce the branching factor. They are dynamic programming, various types of branch-and-bound algorithms, etc. For opposing games like chess the most popular algorithms are various search algorithms with alpha-beta pruning [20]. They are implemented in the most powerful computer chess programs, e.g., in all the programs which are current and former World Computer Chess Champions. It was proved that these algorithms, in the best case, theoretically can reduce the branching factor to $B^{0.5}$[20]. Supposing that an arbitrary chess position in average contains about 40 moves permitted according to the chess rules, alpha-beta pruning can reduce this number to approximately 6. Still we have an exponential growth with a very *high base* (high branching factor). Thus chess problems that require a deep search, e.g., the search to the depth of 20 or more plies, would require enormous amounts of processing time to be solved. We encounter the same problem but in a greater scope when we apply search algorithms with alpha-beta pruning (or branch-and-bound algorithms) to real-world problems, e.g., when we look for an optimal operation of complex systems. In such problems the number of possibilities in each state usually is far more than 40, so an alpha-beta or

branch-and-bound reduction of the branching factor does not provide a solution in a reasonable processing time.

Returning to the discussion of experiments with the PIONEER chess program, let us consider the values of branching factor as well as some other parameters of the search [16]. The search tree generated by PIONEER while solving the R. Reti endgame contained 54 nodes (T = 54), hence, taking into account that the length of the solution L = 6 here, we have B~1.65. In the Botvinnik-Kaminer endgame the total number of nodes generated by the program was equal to 145, maximum length L =12, hence B~1.34. Although both endgames are solvable by conventional chess programs, these results are very interesting in the framework of substantial reduction of the branching factor.

Among the variety of complex problems solved by the PIONEER, we shall consider two. Both *are not solved yet* by the conventional chess programs: alpha-beta pruning failed to provide a substantial reduction of the branching factor, and so the expected processing time would exceed a reasonable amount.

The first problem is the G.Nadareishvili endgame [16]. The total number of nodes generated was T = 200, while the depth of the search required to find a solution is equal to 25! Consequently, B~1.14. At the initial position of this endgame there are 10 pieces, and the unreduced branching factor might be estimated as B~15. The second complex problem we would like to consider is the middle-game position in a game by Botvinnik-Capablanca. This position contains 19 pieces and the unreduced branching factor might be estimated as B~20! The depth of the search should not be less than 23. The PIONEER generated a search tree of 40 nodes with the branching factor B~1.05.

Let us consider experiments with maintenance scheduling programs. The program for *monthly* scheduling generated different search trees depending on the number of demands in each month and a list of other constraints [17, 18]. The number of demands varied from 118 to 405 in different months. The total number of nodes never exceeded 165. With 31 as the maximum length of the solution, a reduced branching factor in these problems never exceeded 1.06. (To understand these results we should take into account that the program aggregated some of the demands. In spite of this the unreduced branching factor varied from 50 to 100.)

The experiments with the program for *annual* maintenance scheduling showed that even this higher dimensional problem can be solved employing the proposed approach [19]. The power equipment maintenance plan for the USSR United Power System was computed for 1121 demands. Each demand contained 12 parameters, including resources requirements and different types of constraints. Two types of resources were considered: the power reserve and the maintenance personnel. The last one was broken into different specialties. Obviously for the annual plan the length of the solution was 365! The reduced branching factor never exceeded 1.005.

Evaluation of the quality of a solution for the chess problems is not hard. The variant-solution (or subtree) is known. A computer should find it and prove it is optimum. For maintenance scheduling problems the optimal plan is unknown but the results achieved can be evaluated according to the optimum criterion: maximum total demanded power of the units being actually maintained. For monthly scheduling the total demanded power of the solutions varied from 91% to 99% of the theoretical optimum value. For annual scheduling the total demanded power of the solutions was equal to 83% of the total demand while a theoretical optimum was unknown.

The comparison with analogous scheduling programs based on branch-and-bound (or dynamic programming) search strategies showed the advantage of our approach for monthly planning; the quality of the plan was about the same, but the computation time in our case was essentially shorter. In all experiments the branching factor of the trees generated by conventional programs was substantially higher. For yearly planning problems the competition could not even happen, because conventional programs could not overcome in a reasonable time the “combinatorial explosion” for such a higher-dimensional problem [17, 18, 19].

The results shown by these programs 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 a branching factor close to 1. In order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex systems, we develop a formal theory, the so-called Linguistic Geometry [21–26]. While the survey of the whole theory and results on one-dimensional structures were presented in various papers [21–25], the following should be considered as a contribution to the Two-dimensional Linguistic Geometry and applications (see also [26]).

3. Informal Review

The idea of a hierarchy of formal languages has been implemented in full for the problems which can be stated as problems of optimal functioning of a Complex System, a twin-set of *elements* and *points* where elements are units moving from one point to another. The elements are divided into two opposite sides: the goal of each side is to maximize a gain, the total value of opposite elements withdrawn from the system. Such a withdrawal happens if an element comes to the point where there is already an element of the opposite side: in this case opposite element should be withdrawn, e.g., as in a game of chess.

According to [16], a one-goal, one-level system should be substituted for a multi-goal multi-level system by introducing intermediate goals and breaking it down into subsystems striving to attain these goals. The goals of the subsystems are individual but coordinated with the main mutual goal. Each subsystem includes elements of both sides; the goal of one side is to attack and gain some element (a target), while the other side tries to protect it. Thus, a subsystem called a Zone is the set of elements of both sides with their trajectories (paths). The pruning criteria for the search and evaluation function are coordinated with the intermediate subsystem's goals and the main goal of the system. Obviously, problems studied in [16] are not the only class of problems eligible for creating a hierarchy of formal languages.

Let us review the formal linguistic representation. Lower level subsystems are called the *trajectories* of points for moving elements along these points to achieve certain local goals. Trajectories are strings of a lower level formal language, the Language of Trajectories. Higher level subsystems are well-organized *networks* of trajectories for moving elements along them to achieve cooperative goals, specific for each network. These networks, called Zones, are represented as strings of a yet higher level language, the Language of Zones; each symbol of the string represents a trajectory, i.e., the string of a lower level language.

The system functions by moving from one state to another; that is, the movement of an element from one point to another causes an adjustment of the hierarchy of languages. This adjustment can be represented as a mapping (*translating*) to some other 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 is represented as a string of the highest level formal language, the Language of Translations, which itself is a member of the family of languages corresponding to various well-known search algorithms: depth-first search, breadth-first search, alpha-beta and others. Every string of the Language of Translations (corresponding to some search tree) contains a solution to the specific search problem.

Next we consider the specific class of formal grammars to be used for the formal definition and studies of trajectory network languages, in particular, the Language of Zones.

4. 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 sub-patterns. This approach brings to light an analogy between the hierarchic structure of patterns and the syntax of languages. The rules controlling the merging of sub-patterns 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 trajectories [22–25] and trajectory networks, we make use of the theory of formal grammars in the form developed in [10–12]. We begin with the definition of the class of grammars to be used.

Definition 4.1. A ***controlled grammar*** **G** is the following eight-tuple:

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

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* U*Con* U*Var* U*Func* U*Pred* U{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* U*Fvar*). Functions have an attached non-negative integer referred to the *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;

$\boldsymbol{H}$ 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 non empty 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 $V_T \cup V_N$ in 2^{Var} matching with each symbol of the alphabet $V_T \cup V_N$ a set of formal parameters, with $Parm(S)=Var$;

$\boldsymbol{L}$ is a finite set called the set of *labels*;

$\boldsymbol{R}$ is a finite set of *productions*, i.e., a finite set of the following seven-tuples:

$$\boldsymbol{(l,\ Q,\ A\text{—>}B,\ \pi_k,\ \pi_n,\ F_T,\ F_F)}.$$

Here

$\boldsymbol{l}$ (from $\boldsymbol{L}$) 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;

$\boldsymbol{Q}$ 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 $(V_T \cup V_N)^*$ 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 $V_T \cup V_N$ 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	$A(\,,\,,\,) \text{—>} a(\,,\,,\,)b(\,,\,)$			
	V_T = ...	V_N = ... V_{PR} = ...			
	E is ...	*Parm*: ...			

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 values from E, and 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 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 4.2. 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.

5. An Example of Controlled Grammar

In order to make transparent the definition of controlled grammars let us consider a simple example. Next we present a process of generating solutions of the well-known Tower of Hanoi Problem by the specific controlled grammar.

The problem is as follows. There are three pivots ***a***, ***b***, and ***c***. On the first one there is a set of *n* disks, each of different radius. The task is to move all the disks to the pivot ***c*** moving only one disk at a time. In addition, at no time during the process may a disk be placed on top of a smaller disk. The pivot ***c*** can, of course, be used as a temporary resting place for the disks.

Let us designate an elementary step of moving disk number i from the pivot x to the pivot y as ***p***(i, x, y), a terminal symbol with parameters. Thus a solution of the Tower of Hanoi Problem might be represented as the following string of symbols with parameters:

$$\boldsymbol{p}(i_1, x_1, y_1)\boldsymbol{p}(i_2, x_2, y_2)\ldots\boldsymbol{p}(i_m, x_m, y_m).$$

This is the string of the language of all possible sequences of moves. Consider the following controlled grammar:

Table 2. A controlled grammar generating solutions to the Tower of Hanoi Problem

L	Q	Kernel, π_k	π_n	F_T	F_F
1	Q_1	$S(n, x, y)$ –> $A(n, x, y)$		2	ø
2	Q_2	$A(n, x, y)$ –> $A(f_1(n), x, f_2(x, y))$ $p(n, x, y)$ $A(f_1(n), f_2(x, y), y)$		2	3
3	Q_3	$A(n, x, y)$ –> $p(n, x, y)$		2	ø

$V_T = \{p\}$

$V_N = \{S, A\}$

V_{PR}

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

$Q_1 = T$

$Q_2(n) = T$, if n>1; $Q_2(n) = F$, if n = 1.

$Q_3(n) = T$, if n=1; $Q_3(n) = F$, if n > 1.

Var = {n, x, y}

F = *Fcon* U *Fvar*,

Fcon $= \{f_1, f_2\}$

$f_1(n) = n-1$, n = 2,3,...

$f_2(x, y)$ yields the value from $\{a, b, c\}\backslash\{x, y\}$, where values of x, y are from $\{a, b, c\}$

Fvar $= \{3, a, c\}$

$E = Z_+ \cup \{a, b, c\}$

Parm: S –>*Var*, A –>*Var*, p –>*Var*

L = {1, 2, 3}

At the beginning of derivation: x = ***a***, y = ***c***, n = 3.

Consider the derivation for the case of three disks: n =3, x=***a***, y=***c***, i.e., the values of parameters for the starting symbol ***S*** are ***S***(3, ***a***, ***b***). A symbol ^{m}=> means application of the production with the label *m*.

S(3, ***a***, ***b***) ^{1}=> ***A***(3, ***a***, ***c***)^{2}=> ***A***(2, ***a***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
^{2}=> ***A***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***A***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
^{3}=> ***p***(1, ***a***, ***c***)***p***(2. ***a***, ***b***)***A***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
^{3}=> ***p***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
^{2}=>***p***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(1, ***b***, ***a***)***p***(2, ***b***, ***c***)***A***(1, ***a***,***c***)
^{3}=> ***p***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***p***(1, ***b***, ***a***)***p***(2, ***b***, ***c***)***A***(1, ***a***, ***c***)
^{3}=> ***p***(1,***a***, ***c***)***p***(2,***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***p***(1, ***b***, ***a***)***p***(2, ***b***, ***c***)***p***(1, ***a***, ***c***).

6. A Formal Statement of the Problem

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

< X, P, R_p, ON, v, S_i, S_t, TR>,

where

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

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

R_p(x,y) is a set of binary relations of *reachability* in X (x and y 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;

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

TR is a set of operators TRANSITION(p, x, y). They describe transitions of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed 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)^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

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

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 [13], nonlinear planner NOAH [14], or in subsequent planning systems, such as MOLGEN [27] or TWEAK [28]. 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, military operations and robot control problems with two opposing sides might be considered as Complex Systems (see Sections 13, 14, and [22, 23, 25]). But it is interesting that this specific model of the formal linguistic approach is applicable to representing and solving a wide class of practical problems "without obvious opposing side" such as power maintenance scheduling, long-range planning, operations planning, VLSI layout, and various operations research problems. 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 a goal and the other is responsible for the provision of resources (see Section 15, and [17, 24, 25]).

7. A Measurement of Distances

To create and study network languages we have to present briefly some geometrical properties of one-dimensional structures of the Complex System.

Definition 7.1. A ***map of the set*** **X** relative to point x and element p for the Complex System is the mapping:

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

(x is from X, p is from P) which is constructed as follows. We consider a set of the *areas of reachability* from the point x, i.e., 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 then *k*.

Let

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

In the remaining 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 a *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*.**

8. Languages of Trajectories

Here, we define a lower-level language of the hierarchy of languages. It will serve as a building block to create the upper-level languages, in particular, the network 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 of this path.

Definition 8.1. 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:

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

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,…, l–1; element p stands at the point x: ON(p)=x. We denote $t_p(x, y, l)$ the set of trajectories in which p, x, y, and l are the same. $P(t_o)=\{x, x_1, ..., x_l\}$ is the set of parameter values of the trajectory t_o.

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

Definition 8.2. 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. 1, a trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ 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 8.3. 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 $P(t_o)$, $i_1 < i_2 < \ldots < i_{k-1}$, $k \leq l$, such that corresponding substrings

$$\boldsymbol{a}(x_o)\ldots\boldsymbol{a}(x_{i_1}),\ \boldsymbol{a}(x_{i_1})\ldots\boldsymbol{a}(x_{i_2}),\ \ldots,\ \boldsymbol{a}(x_{i_{k-1}})\ldots\boldsymbol{a}(x_l)$$

are the shortest trajectories.

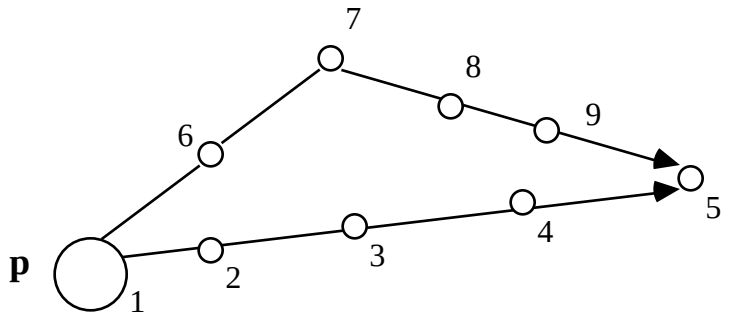

Fig. 1. An interpretation of shortest and admissible trajectories.

The shortest and admissible trajectories of degree 2 play a special role in many problems. An example of such a trajectory $\boldsymbol{a}(1)\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(5)$ is shown in the Fig. 1. 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.1) 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 8.4. A ***Language of Trajectories*** $L_t^H(S)$ for the Complex System in state S is the set of all the shortest and admissible (degree 2) trajectories of the length less than H. This

language also includes the 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. The following theorem holds [25]:

Theorem 8.1. If the distance between points x_o and y_o from X is less or equal l_o for the element p on x_o, i.e., if $ON(p) = x_o$ and $MAP_{x_o,p}(y_o) \le l_o$, where $l_o < 2n$, n is the number of points in X, and 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 there exists a certain controlled grammar, called $\mathbf{G_t^{(2)}}$, which can generate all the shortest and admissible (of degree 2) trajectories $t_p(x_o, y_o, l_o)$ from x_o to y_o of the length less or equal l_o .

9. Languages of Trajectory Networks

After defining the Language of Trajectories, we have the tools for the breakdown of our System into subsystems. According to the ideas presented in [16], these subsystems should be various types of trajectory networks, i.e., some sets of interconnected trajectories with one singled out trajectory called the main trajectory. An example of such a network is shown in Fig. 2.

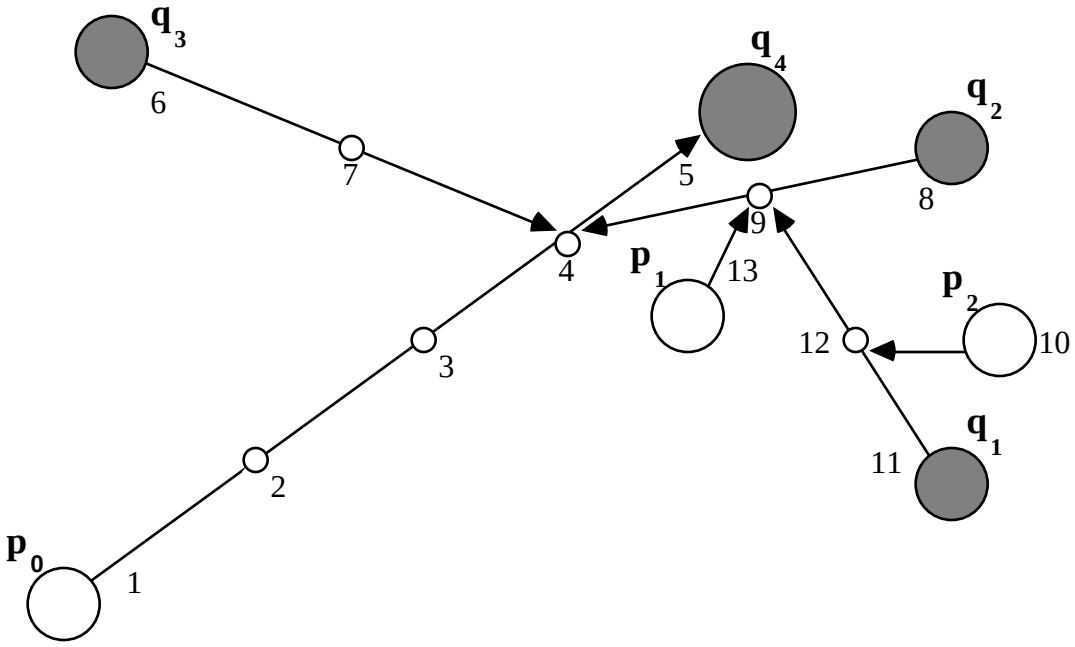

Fig. 2. A network language interpretation.

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

$\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of element q_1 to prevent this control.

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

Definition 9.1. A ***trajectory connection*** of the trajectories t_1 and t_2 is the relation $C(t_1,t_2)$. It holds, if the ending link of the trajectory t_1 coincides with an intermediate link of the trajectory t_2; more precisely t_1 is connected with t_2, if among the parameter values $P(t_2)=\{y,y_1,\ldots,y_l\}$ of trajectory t_2 there is a value $y_i = x_k$, where $t_1=\boldsymbol{a}(x_o)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_k)$. If t_1 belongs to some set of trajectories with the common end-point, than the entire set is said to be connected with the trajectory t_2.

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

Definition 9.2. A set of trajectories $CA_B(t)$ from B, with which trajectory t is connected is called the ***bundle of trajectories*** for trajectory t relative to the set B of trajectories.

To formalize the trajectory networks we should define some routine operations on the set of trajectories: a k-th degree of connection and a transitive closure.

Definition 9.3. A ***k-th degree of the relation C*** on the set of trajectories A (denoted by C_A^k) is defined as usual by induction.

For $k = 1$ $C_A^k(t_1,t_2)$ coincides with $C(t_1,t_2)$ for t_1,t_2 from A.

For $k > 1$ $C_A^k(t_1,t_2)$ holds if and only if there exists a trajectory t_3 from A, such that $C(t_1,t_3)$ and $C_A^{k-1}(t_3,t_2)$ both hold.

Trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ in Fig.2 is connected (degree 2) with trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$, i.e., $C^2(\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5))$ holds.

Definition 9.4. A ***transitive closure of the relation*** **C** on the set of trajectories A (denoted by C_A^+) is a relation, such that $C_A^+(t_1,t_2)$ holds for t_1 and t_2 from A, if and only if there exists $i > 0$ that $C_A^i(t_1,t_2)$ holds.

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

Definition 9.5. A ***trajectory network*** **W** relative to trajectory t_o is a finite set of trajectories $t_o,t_1,\ldots,t_k$ from the language $L_t^H(S)$ that possesses the following property: for every trajectory t_i from W ($i = 1, 2,\ldots,k$) the relation $C_W^+(t_i,t_o)$ holds, i.e., each trajectory of the network W is connected with the trajectory t_o that was singled out by a subset of interconnected trajectories of this network.

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

Definition 9.6. A ***family of trajectory network languages*** $\mathbf{L_C(S)}$ in a state S of the

Complex System is the family of languages that contains strings of the form

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

where *param* in parentheses substitute for the other parameters of a particular language. All the symbols of the string $t_1, t_2,\ldots, t_m$ correspond to trajectories which form a trajectory network W relative to t_1.

Different members of this family correspond to different types of trajectory network languages which describe particular subsystems for solving search problems. One of such languages is a language which describes specific networks called Zones. They play a main role in the model considered here [16]. The formal definition of this language is essentially constructive and requires showing explicitly a method for generating this language, i.e., a certain formal grammar. This grammar will be discussed later. In order to make our points transparent, first of all, we define the Language of Zones informally.

A *Language of Zones* is a trajectory network language with strings of the form

$$Z=t(p_o,t_o,\tau_o)\ t(p_1,t_1,\tau_1)\ldots t(p_k,t_k,\tau_k),$$

where $t_o,t_1,\ldots,t_k$ are the trajectories of elements $p_o,p_2,\ldots,p_k$ respectively; $\tau_o,\tau_1,\ldots,\tau_k$ are positive integer numbers (or 0) which "denote the time allocated for the motion along the trajectories in a correspondence to the mutual goal of this Zone: to remove the target element – for one side, and to protect it – for the opposite side. Trajectory $t(p_o,t_o,\tau_o)$ is called the *main trajectory* of the Zone. The element q standing on the ending point of the main trajectory is called the *target.* The elements p_o and q belong to the opposite sides.

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

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

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

10. A Grammar of Zones

Here we consider a formal definition of the Language of Zones employing class of controlled grammars.

Definition 10.1. A language $L_Z(S)$ generated by the grammar G_Z (Tables 3, 4) in a state S of a Complex System is called the ***Language of Zones***.

Table 3
A Grammar of Zones G_Z

L	Q	Kernel, π_k	π_n for all z from X	F_T	F_F
1	Q_1	$S(u, v, w) \rightarrow A(u, v, w)$		*two*	ø
2_i	Q_2	$A(u, v, w) \rightarrow t(h_i^0(u), l_o+1)$ $A((0, 0, 0), g(h_i^0(u), w), zero)$	TIME(z)=$DIST(z, h_i^0(u))$	3	ø
3	Q_3	$A(u, v, w) \rightarrow A(f(u, v), v, w)$	NEXTTIME(z)= $init(u$, NEXTTIME(z))	*four*	5
4_j	Q_4	$A(u, v, w) \rightarrow t(h_j(u)$, TIME(y))) $A(u, v, g(h_j(u), w))$	NEXTTIME(z)= $ALPHA(z, h_j(u)$, TIME(y)–l+1)	3	3
5	Q_5	$A(u, v, w) \rightarrow A((0, 0, 0), w, zero)$	TIME(z)=NEXTTIME(z)	3	6
6	Q_6	$A(u, v, w) \rightarrow e$		ø	ø

$V_T = \{t\}$,

$V_N = \{S, A\}$,

V_{PR}

Pred $= \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6\}$

$Q_1(u) = (ON(p_o)=x) \wedge (MAP_{X,p_o}(y) \le l \le l_o) \wedge (\ q((ON(q)=y) \wedge (\chi(p_o, q)=0)))$
$Q_2(u) = T$
$Q_3(u) = (x \neq n) \wedge (y \neq n)$
$Q_4(u) = (\ p_1((ON(p_1)=x) \wedge (l > 0) \wedge (((\chi(p_o, p_1)=1) \wedge (MAP_{X,p_1}(y)=1))$
$((\chi(p_o, p_1)=0) \wedge (MAP_{X,p_1}(y) \le l))))$
$Q_5(w) = (w \neq zero)$
$Q_6 = T$

Var $=$ {x, y, l, τ, θ, $v_1, v_2, \ldots, v_n, w_1, w_2, \ldots, w_n$}; for the sake of brevity:
u=(x, y, l), v=($v_1, v_2, \ldots, v_n$), w=($w_1, w_2, \ldots, w_n$), *zero*=(0, 0, ..., 0)

Con $= \{x_o, y_o, l_o, p_o\}$;

Func=*Fcon* U *Fvar*;

Fcon=$\{f_x, f_y, f_l, g_1, g_2, \ldots, g_n, h_1, h_2, \ldots, h_M, h_1^0, h_2^0, \ldots, h_M^0,$
$DIST, init, ALPHA\}$, $f = (f_x, f_y, f_l)$, $g = (g_{X1}, g_{X2}, \ldots, g_{Xn})$,
$M=|L_t^{l_o}(S)|$ is the number of trajectories $L_t^{l_o}(S)$.

Fvar=$\{x_o, y_o, l_o, p_o$, TIME, NEXTTIME$\}$

$E = Z_+ \cup X \cup P \cup L_t^{l_o}(S)$ is the subject domain;

Parm: $S \rightarrow Var$, $A \rightarrow \{u, v, w\}$, $t \rightarrow \{p, \tau, \theta\}$;

$L = \{1, 3, 5, 6\} \cup$ *two* $\cup$ *four*, *two*=$\{2_1, 2_2, \ldots, 2_M\}$, *four*=$\{4_1, 4_2, \ldots, 4_M\}$

Table 4
A definition of functions of the Grammar of Zones G_Z

D(***init***) = X x X x $\mathbf{Z}_+$ x $\mathbf{Z}_+$

$$\textit{init}(u,r) = \begin{cases} 2n, & \text{if } u = (0,0,0), \\ r, & \text{if } u \neq (0,0,0). \end{cases}$$

D(***f***) = (X x X x $\mathbf{Z}_+$ U{0, 0, 0}) x $\mathbf{Z}_+^n$

$$f(u, v) = \begin{cases} (x+1, y, l), & \text{if } (x \neq n) \quad (l > 0), \\ (1, y+1, \text{TIME}(y+1) * v_{y+1}), & \text{if } (x = n) \quad ((l \leq 0) \quad (y \neq n)), \\ (0,0,0), & \text{if } (x = n) \quad (y = n). \end{cases}$$

D(***DIST***) = X x P x $\mathbf{L_t}^{l_0}$(S). Let t_o $\mathbf{L_t}^{l_0}$(S), $t_o = \boldsymbol{a}(z_0)\boldsymbol{a}(z_1)\ldots\boldsymbol{a}(z_m)$, t_o $t_{p_o}(z_o, z_m, m)$;
If
for some k ($1 \leq k \leq m$) $x = z_k$,
then ***DIST***$(x, p_o, t_o) = k+1$
else ***DIST***$(x, p_o, t_o) = 2n$

D(***ALPHA***) = X x P x $\mathbf{L_t}^{l_0}$(S) x $\mathbf{Z}_+$

$$\textit{ALPHA}(x, p_o, t_o, k) = \begin{cases} \textit{min}(\textit{NEXTTIME}(x), k), & \text{if } \textit{DIST}(x, p_o, t_o) < 2n, \\ \textit{NEXTTIME}(x), & \text{if } \textit{DIST}(x, p_o, t_o) = 2n. \end{cases}$$

D($\boldsymbol{g}_\mathbf{r}$) = P x $\mathbf{L_t}^{l_0}$(S) x $\mathbf{Z}_+^n$, r X.

$$g_r(p_o, t_o, w) = \begin{cases} 1, & \text{if } \textit{DIST}(r, p_o, t_o) < 2n, \\ w_r, & \text{if } \textit{DIST}(r, p_o, t_o) = 2n. \end{cases}$$

D ($\boldsymbol{h}_i^o$) = X x X x $\mathbf{Z}_+$; Denote $\text{TRACKS}_{p_o} = \{p_o\}$ x ($\underset{1 \leq k \leq l}{\text{L}}[\mathbf{G_t^{(2)}}(x, y, k, p_o)]$
If
$\text{TRACKS}_{p_o} = e$
then $\boldsymbol{h}_i^o(u) = e$
else $\text{TRACKS}_{p_o} = \{(p_o, t_1), (p_o, t_2), \ldots, (p_o, t_b)\}, (b \leq M)$ **and** $\boldsymbol{h}_i^o(u) = \begin{cases} (p_o, t_i), & \text{if } i \leq b, \\ (p_o, t_b), & \text{if } i > b. \end{cases}$

D($\boldsymbol{h}_i$) = X x X x $\mathbf{Z}_+$; Denote TRACKS = $\underset{\text{ON}(p)=x}{\text{TRACKS}_p}$, where TRACKS_p is the same as for $\boldsymbol{h}_i^o$
If
TRACKS = e
then $\boldsymbol{h}_i(u) = e$
else $\text{TRACKS} = \{(p_1, t_1), (p_1, t_2), \ldots, (p_m, t_m)\}$, $(m \leq M)$ **and** $\boldsymbol{h}_i(u) = \begin{cases} (p_i, t_i), & \text{if } i \leq m, \\ (p_m, t_m), & \text{if } i > m. \end{cases}$

At the beginning of derivation: $u=(x_o, y_o, l_o)$, $w = zero$, $v = zero$, x_o X, y_o X, l_o $\mathbf{Z}_+$, p_o P, and TIME(z)=2n, NEXTTIME(z)=2n for all z from X.

11. A Geometry of Zones

To study this language formally we need some preliminary definitions.

Definition 11.1. An ***alphabet*** A(Z) ***of the string*** Z of the parameter language L is the set symbols of this language with given parameter values, where each of this symbols with parameters is included at least once in a string Z, and *e* (the empty symbol).

Definition 11.2. A ***trajectory alphabet*** TA(Z) of the zone Z is the set of trajectories from $L_t^H(S)$ that correspond to the actual parameter values of the alphabet A(Z).

Theorem 11.1. For any string Z from $L_Z(S)$ trajectories from TA(Z) form a trajectory network, i.e., $L_Z(S)$ $L_C(S)$.

Proof. Let us consider a string $Z=\boldsymbol{t}(p_o, t_o, \tau_o)\ldots\boldsymbol{t}(p_k, t_k, \tau_k)$. Obviously under the condition that the predicate $\boldsymbol{Q_1}$ is true, the symbol $\boldsymbol{t}(p_o, t_o, \tau_o)$ is attached to the string by applying the productions 1 and 2_i. The following proof is by induction. We assume that all the trajectories $TA(Z_m)$ of the substring $Z_m= \boldsymbol{t}(p_o, t_o, \tau_o)\ldots\boldsymbol{t}(p_m, t_m, \tau_m)$ $(m < k)$ form a trajectory network.

The symbol $\boldsymbol{t}(p_{m+1}, t_{m+1}, \tau_{m+1})$ can be attached to a string only after applying the production with the label 4_j. Among the parameters of the trajectory t_{m+1} $t_p(x, y, l)$ we are interested in the value of y, the parameter value of the last symbol of the trajectory. One can pass to the production with the label 4_j only after a successful application of a production with the label 3, i.e., in F_S case. Here the $\boldsymbol{f}(u, v)$ function changes the value of the parameter $u=(x, y, l)$. It is clear that with the last change of y caused by the substitution of $f(x_o, y_o, l_o, v)$ for $u=(x, y, l)$, some y_o was replaced by $y = y_o+1$ with $l = TIME(y)*v_y \neq 0$. Otherwise the given change of y, and hence of *l*, would not have been the last (before the application of the production with the label 4_j), as all attempts of applying the production 4_j for $l = 0$ would have been unsuccessful ($\boldsymbol{Q_4}=F$ for $l = 0$).

Thus, $TIME(y) \neq 0$ and $v_y \neq 0$. The last change in the course of derivation of the value of v_y could occur only in a successful application of a production with the label 5. Here, after applying the production, v_y was given the value of w_y. Consequently, $w_y \neq 0$.

Finally, such a change of the value of w_y for which it would become different from zero, could take place only in a successful application, earlier in the derivation, of one of the productions with the label 4_j. This means that at some stage of derivation the symbol $\boldsymbol{t}(p_j, t_j, \tau_j)$ was included in the string Z. At the same time, the parameter $w_o=(w_1^o,\ldots,w_n^o)$ was changed under the action of the function $\boldsymbol{g}(\boldsymbol{h_j}(u), w^o)$ in such a way that $w_y = \boldsymbol{g}_y(\boldsymbol{h_j}(u), w^o)$. But $w_y \neq 0$; consequently, $w_y = 1$, i.e., $\boldsymbol{DIST}(y, p_j, t_j) < 2n$, and hence y is included among the parameter values of the t_j trajectory. In addition, obviously this trajectory is included among the trajectories $t_o, t_1, \ldots, t_m$, since the symbol $\boldsymbol{t}(p_j, t_j, \tau_j)$ was included in Z earlier in the course of derivation.

In accordance with Definition 9.1, trajectory t_{m+1} is connected with trajectory t_i, i.e., $C(t_{m+1}, t_i) = T$ holds, with $i \leq m$. But by the induction assumption $C^+_{TA(Z)}(t_i, t_o) = T$ and , taking into account Definitions 9.3–9.5, we conclude that $C^+_{TA(Z)}(t_{m+1}, t_o) = T$ (because of the transitivity of C^+). Thus all the trajectories $t_o, t_1, \ldots, t_{m+1}$ form a trajectory network.

The theorem is proved.

12. Translations of Languages

The Language of Zones allows us to describe the "statics", i.e., the states of the System. We proceed with the description of the "dynamics" of the System, i.e., the transitions from one state to another. The transitions describe the change of the descriptions of states as the change of

sets of WFF. After each transition a new hierarchy of languages should be generated. Of course, it is an inefficient procedure. To improve an efficiency of applications in a process of the search it is important to describe the change of the hierarchy of languages. A study of this change should help us in modifying the hierarchy instead of regenerating it in each state. The change may be described as a hierarchy of mappings – translations of languages. Each hierarchy's language should be transformed by the specific mapping called a translation.

Definition 12.1. A ***translation relation Tr*** from a language L_1 to a language L_2 is the binary relation *Tr* from L_1 into L_2 for which L_1 is the domain and L_2 is the range. If *Tr*(*a, b*) holds, than the string *b* is called the output for the input string *a.*

In general, for the translation relation for each input string there may be several output strings. However, in our case we can consider the translation relation as a mapping, i.e., "for each input – no more than one output".

Definition 12.2. Let the Complex System move from the state S_1 to the state S_2 by applying the operator T_o= TRANSITION(p, x_o,y_o). ***A Translation of Languages of Trajectories*** is a mapping

$$\prod_{To}: L_t^H(S_1) \text{ —> } L_t^H(S_2),$$

of such a sort that the trajectories of the form ***a***(x)***a***(y)…***a***(z) are transformed as follows:

– are **"shortened"** by the exclusion of the first symbol ***a***(x), if the transition T_o carries out along such a trajectory: $x = x_o$ & $y = y_o$. (If y = z, i.e., y is the ending point, the trajectory is transformed into the empty trajectory *e*.)

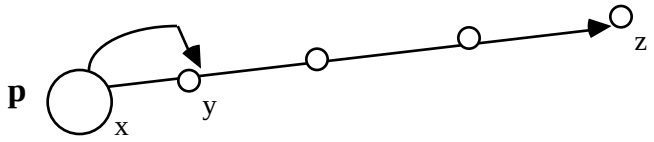

Fig. 3. A "shortening" trajectory.

– are transformed into the **empty trajectory *e***, if

element p moves away from such a trajectory: $x = x_o$ & $y \neq x_1$,

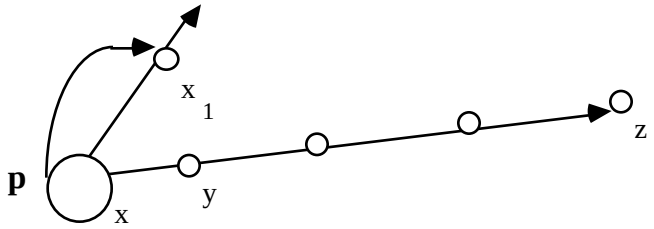

Fig. 4. A trajectory with the element that moves away.

or this element is withdrawn: $x = x_1$ and WFF ON(q) = x_1 comes into the Remove list of the transition T_o (see Section 3.)

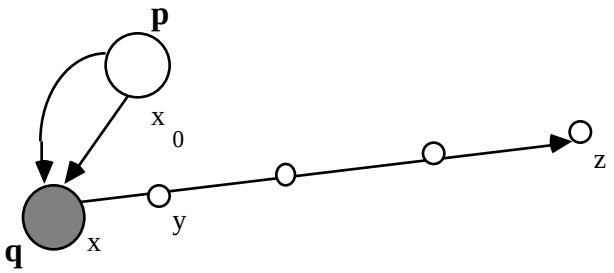

Fig. 5. A trajectory whose element is withdrawn.

– are transformed **into itself** in all the other cases.

Obviously, mapping $\prod_{Mo}$ is not a mapping "onto" and has a non-empty kernel, i.e., a

nonempty co-image of the empty trajectory *e*. For example, in Fig. 2 after transition TRANSITION(p_2, 10, 12) the trajectory ***a***(10)***a***(12) is translated into the trajectory *e* and all the remaining trajectories are translated into itself.

To proceed with the description of the hierarchy change we should define a translation of its next level, the Trajectory Network Languages. Let us consider the definition translation for the Language of Zones.

Definition 12.3. A ***Translation of Languages of Zones*** is a mapping of the following form:

$$\pi_{T_o}: L_Z(S_1) \rightarrow L_Z(S_2),$$

where Zone Z_1 is translated into Zone Z_2, i.e., $\pi_{To}(Z_1) = Z_2$ if and only if the main trajectory t_o^1 of Zone Z_1 is translated into the main trajectory t_o^2 of the Zone Z_2 by the corresponding trajectory translation, $\prod_{T_o}(t_o^1) = t_o^2$.

After transition TRANSITION(p_o, 1, 2) the Zone depicted in Fig.2 is translated into a new Zone with the main trajectory ***a***(2)***a***(3)***a***(4)***a***(5), because this transition causes such a translation of trajectories that trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5) is translated into the trajectory ***a***(2)***a***(3)***a***(4)***a***(5).

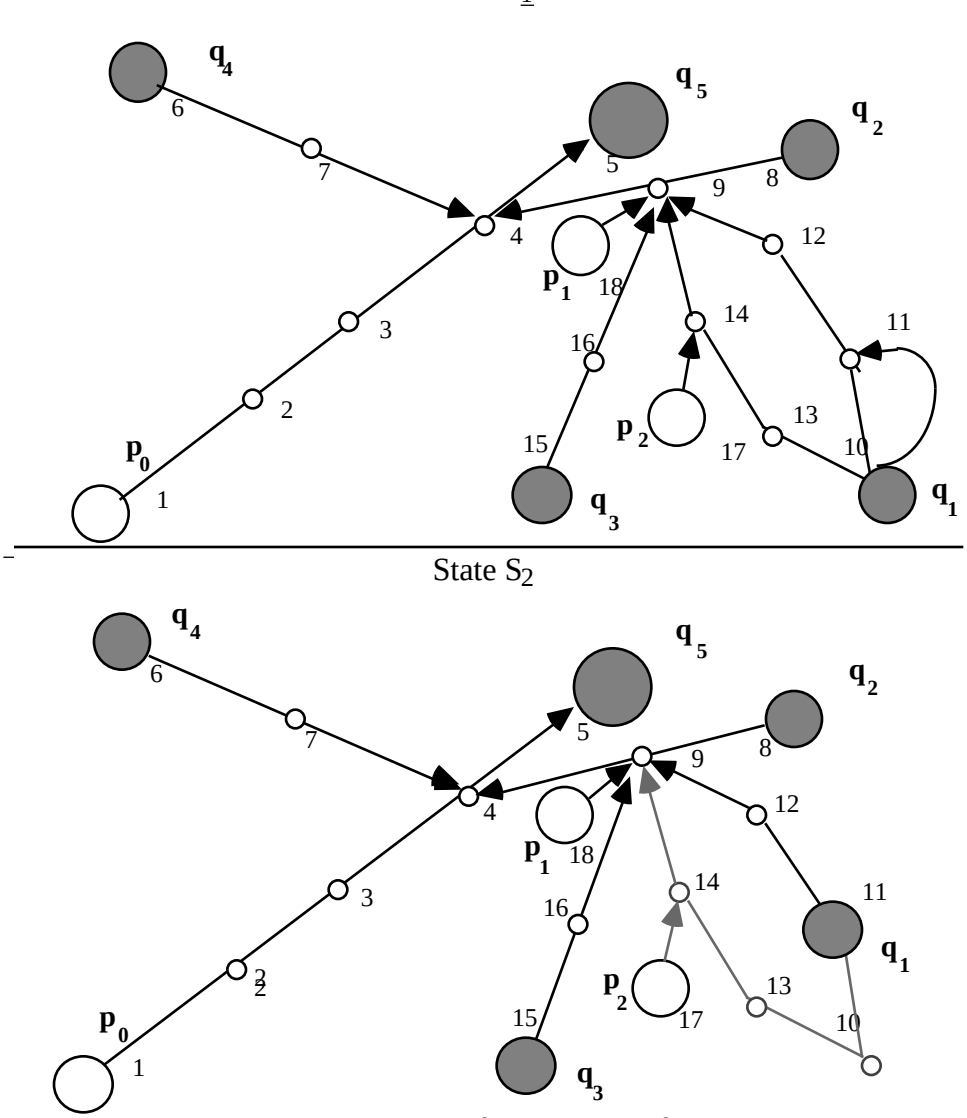

Fig.6. A translation of Languages of Zones.

Let us take a look at the different example (Fig. 6). The Language of Zones in State 1 consists of two Zones with the same main trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5). The difference between these Zones is in the trajectories of element q_1. Trajectory ***a***(10)***a***(11)***a***(12)***a***(9) is included into Zone 1 while ***a***(10)***a***(13)***a***(14)***a***(9) together with ***a***(17)***a***(14) are included into Zone 2. After the TRANSITION(q_1, 10, 11) the Language of Zones in State S1 is translated into the new Language of Zones in State S2. Trajectory ***a***(10)***a***(11)***a***(12)***a***(9) is shortened; it is translated into ***a***(11)***a***(12)***a***(9). This is the only difference between the Zone 1 and its translation. The change for Zone 2 is more essential. It "looses" trajectories ***a***(10)***a***(13)***a***(14)***a***(9) and ***a***(17)***a***(14) at all. (The trajectories and its links that are not included in the Language of Zones in a State S2 are shown by dotted lines in Fig. 6.)

It is very important to show the difference between the Zone and its translation in *general case*, i.e., to describe which trajectories of the old Zone remain unchanged in the new one, which trajectories are shortened, as ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5) in Fig.2 or ***a***(10)***a***(11)***a***(12)***a***(9) in Fig. 6, which are not included, i.e., are translated into the empty trajectory *e*, and finally, what are the new trajectories of the new Zone. This knowledge for every transition would give us a description of the change of the Language of Zones.

A description of the change for the Language of Trajectories is trivial and explicitly follows from the definition of translations of these languages. For the Translation of Languages of Zones it is a problem. It is currently under development. The study of properties of translations allows us to give a formal, constructive solution of the well-known frame problem [13, 15] for this specific system. This is the problem of effective description of boundaries between the actual and outdated information about the system. This information is updated in the process of search for an optimal operation.

13. Zones for the Robot Control Model

A robot control model can be represented as a Complex System naturally (see Section 6). A set **X** represents the operational district which could be the area of combat operation broken into squares, e.g., in the form of the table 8 x 8, n=64. It could be a space operation, where X represents the set of different orbits, etc. **P** is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests; $\mathbf{R_p(x, y)}$ represent moving capabilities of different robots: robot p can move from point x to point y if $R_p(x, y)$ holds. Some of the robots can crawl, the other can jump or ride, 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, and others can do that in *k* steps only, and many of them can not reach this point at all. **ON(p)=x**, if robot p is at the point x; **v(p)** is the value of robot p. This value might be determined by the technical parameters of the robot. It might include the immediate value of this robot for the given combat operation; $\mathbf{S_i}$ is an arbitrary initial state of operation for analysis, or the starting state; $\mathbf{S_t}$ is the set of target states. These might be the states where robots of each side reached specified points. On the other hand S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_j) = x_k\}$ corresponds to the list of robots with their coordinates in each state. **TRANSITION(p, x, y)** represents the move of the robot p from square x to square y; if a robot of the opposing side stands on y, a removal occurs, i.e., robot on y is destroyed and removed.

Two robots with different moving capabilities are shown in Fig. 7. One of them, robot FIGHTER standing on f6, can move to any next square. The other robot BOMBER from h5 can move only straight ahead, e.g., from h5 to h4, from h4 to h3, etc. Thus robot FIGHTER on f6 can reach all the following points y {e5, e6, e7, f7, g7, g6, g5, f4} in on step, i.e., $R_{FIGHTER}(f6, y)$ holds, while robot BOMBER standing on h5 can reach only h4. Obviously, moving capabilities of these robots are similar to well-known chess pieces King and Pawn, respectively. Assume that robots FIGHTER and BOMBER belong to opposite sides: FIGHTER P_1 while BOMBER P_2. Also assume that there is one more robot TARGET (or unmoving device) standing on h1.

TARGET belongs to P_1 which means that characteristic function χ(BOMBER, TARGET)=0. (Function χ(p, q) is defined on P x P and equals 1 if p and q both belong to P_1 or P_2; χ(p, q) = 0 in the remaining cases.) Thus robot BOMBER should reach point h1 to destroy the Target while FIGHTER will try to intercept this motion. Make sure that X corresponds to the area 8 x 8 excluding points g3, g4 which are restricted.

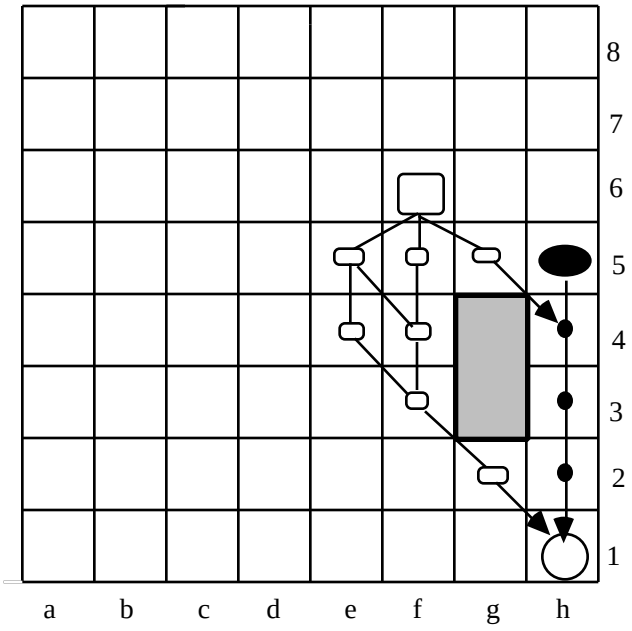

Figure 7. Interpretation of the Language of Zones for the robot control model.

Let us generate the Language of Zones. Here we identify points of X with their ordinal numbers, thus a1 corresponds to 1, a2 to 2, etc., h8 corresponds to 62 (g3, g4 are excluded). We shall use both notations, algebraic and numerical, where it is convenient.

Let us apply grammar $\mathbf{G_Z}$ (Table 3, 4) for different values of *u*. Production 1 is applicable for *u* = (h5, h1, 5) = (40, 8, 5), $l = l_o = 4$ because

$$\boldsymbol{Q_1}(u) = (\mathrm{ON}(\mathrm{BOMBER})=\mathrm{h5}) \wedge (\mathrm{MAP}_{\mathrm{h5,BOMBER}}(\mathrm{h1}) \le 4 \le 4) \wedge ((\mathrm{ON}(\mathrm{Target})=\mathrm{h1}) \wedge (\chi(\mathrm{BOMBER}, \mathrm{TARGET})=0)) = \boldsymbol{T}.$$

Thus,

$$\boldsymbol{S}(u, zero, zero) \overset{1}{\Rightarrow} \boldsymbol{A}(u, zero, zero)$$

and $\boldsymbol{F_T} = two$ is a permissible set. Therefor next we have to apply one of the productions 2_i two. $\mathbf{Q_2}(u)$ is always true so

$$\boldsymbol{A}(u, zero, zero) \overset{2_i}{\Rightarrow} \boldsymbol{t}(\boldsymbol{h_i^0}(\mathrm{u}), 5)\, \boldsymbol{A}((0, 0, 0), \boldsymbol{g}(\boldsymbol{h_i^0}(u), zero), zero)$$

In order to compute $\boldsymbol{h_i^0}(u)$ we have to generate all the shortest and admissible (of degree 2) trajectories from h5 to h1 for the robot BOMBER (Table 4). The length of these trajectories should be less or equal $l = 4$.

$$\mathrm{TRACKS}_{\mathrm{BOMBER}} = \{\mathrm{BOMBER}\} \times \Big(\bigcup_{1 \le k \le 4} L[\, \mathbf{G_t^{(2)}}(\mathrm{h5}, \mathrm{h1}, k, \mathrm{BOMBER})].$$

According to grammar $\mathbf{G_t^{(2)}}$ [25] only one such trajectory t_1 exists, and it is generated by this grammar :

$$t_B = \boldsymbol{a}(\mathrm{h5})\boldsymbol{a}(\mathrm{h4})\boldsymbol{a}(\mathrm{h3})\boldsymbol{a}(\mathrm{h2})\boldsymbol{a}(\mathrm{h1}).$$

Thus TRACKS ={(BOMBER, t_B)}, the number of trajectories b = 1 and $\boldsymbol{h_1^0}(u)$ = (BOMBER, t_B). In that way we generated the main trajectory of the Zone:

$$\boldsymbol{t}(\mathrm{BOMBER}, t_B, 5).$$

Next we have to compute $\boldsymbol{g}(\boldsymbol{h_1^0}(u),zero) = \boldsymbol{g}$ (BOMBER, t_B, *zero*). According to Table 4, for all r X the r-th component of function $\boldsymbol{g}$ is as follows:

$$\boldsymbol{g_r}(\text{BOMBER},t_B, zero) = \begin{cases} 1, & \text{if } \boldsymbol{DIST}(r,\text{BOMBER},t_B) < 2n, \\ 0, & \text{if } \boldsymbol{DIST}(r,\text{BOMBER},t_B) = 2n, \end{cases}$$

The value of the function $\boldsymbol{DIST}$ (x, BOMBER, t_B) = k+1, where k is the number of symbol of the trajectory t_1, whose parameter value equals x. Consequently

$\boldsymbol{DIST}$ (h4, BOMBER, t_B) = 2
$\boldsymbol{DIST}$ (h3, BOMBER, t_B) = 3
$\boldsymbol{DIST}$ (h2, BOMBER, t_B) = 4
$\boldsymbol{DIST}$ (h1, BOMBER, t_B) = 5

For the rest of x from X $\boldsymbol{DIST}$ (x, BOMBER, t_1) = 2 x 62 =124. Thus for r {h1, h2, h3, h4} = {8, 16, 23, 30} $\boldsymbol{g_r}$(BOMBER, t_B, *zero*) = 1, for the rest of r $\boldsymbol{g_r}$= 0.

Now we can complete application of production 2_1:

$\boldsymbol{A}(u, zero, zero) \Rightarrow \boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{A}((0, 0, 0), \boldsymbol{g}(\text{BOMBER},t_B,zero), zero)$.

Non-kernel functional formula from $\pi_{\mathbf{n}}$ remains for computation:

$$\text{TIME}(z) = \boldsymbol{DIST}(z, \text{BOMBER}, t_B).$$

Symbol "=" in these formulas should be considered as an assignment, i.e., the current value of the right side expression should be assigned to the left side. The computation of $\boldsymbol{DIST}$(z, BOMBER, t_B) for all z from X has been performed above, so TIME(z) equals 124 for all z X except {h1, h2, h3, h4}, where TIME(z) equals 5, 4, 3, 2, respectively own.

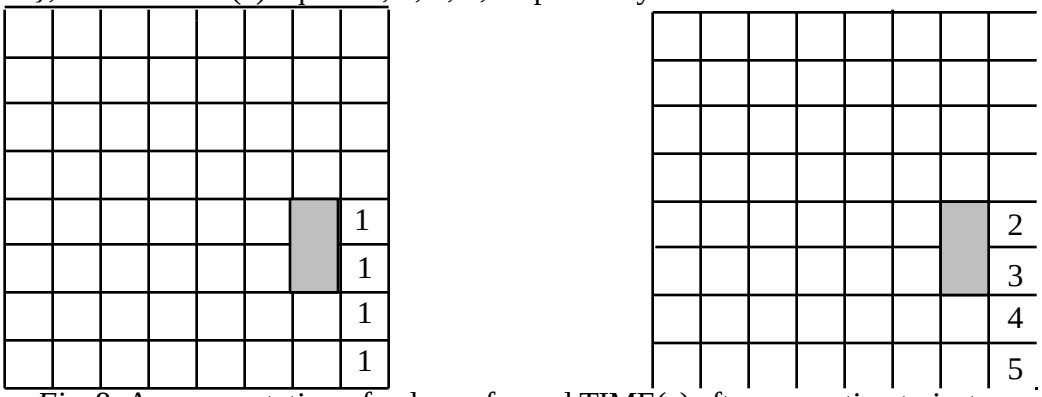

Fig. 8. A representation of values of *v* and TIME(z) after generating trajectory $\boldsymbol{a}$(h5)$\boldsymbol{a}$(h4)$\boldsymbol{a}$(h3)$\boldsymbol{a}$(h2)$\boldsymbol{a}$(h1).

Values of function $\boldsymbol{g}$ and, consequently, values of the components of vector *v* (Fig. 8), different from zero, mark ending points of prospective trajectories of robots from P_1 which could intercept motion of BOMBER along the main trajectory: points h1, h2, h3, h4. Values of TIME (Fig. 8) for the same points designate maximum lengths of those prospective trajectories. These trajectories are called the*1-st negation* trajectories. Points h1, h2, h3, h4 are considered as targets by the other side, P_2, as well. It means that the grammar should generate trajectories of robots (if they exist) which could support motion of BOMBER by preventing its interception, the so-called *own* trajectories. By definition of the Grammar of Zones (Table 3, predicate $\boldsymbol{Q_4}$) the length of such trajectories is restricted by 1. Obviously, there are no own trajectories in the problem shown in Fig. 7.

Let us continue derivation of Zone. Production 2_1 was applied successfully, so we have to go to the production with label 3 and try to apply it to the left-most entry of nonterminal $\boldsymbol{A}$. This production is applicable because $\boldsymbol{Q_3}$ ((0, 0, 0)) = (0 ≠ 64) ^ (0 ≠ 64). Thus,

$\boldsymbol{t}(\text{BOMBER}, t_1, 5)\boldsymbol{A}((0, 0, 0), v, zero) \overset{3}{\Rightarrow} \boldsymbol{t}(\text{BOMBER}, t_1, 5)\boldsymbol{A}(\boldsymbol{f}((0, 0, 0), v), v, zero)$.

Next we have to compute value of the function $\boldsymbol{f}$. According to Table 4 for u = (x, y, *l*) = (0, 0, 0) and $v_{y+1} = v_1 = 0$:

$$\boldsymbol{f}(u, v) = (1, y+1, TIME(y+1) * v_{y+1}) = (1, 1, 0).$$

Therefor,

$$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}((1, 1, 0), v, zero)$$

It remains to compute values of the functional formula from $\pi_{\mathbf{n}}$.

$$NEXTTIME(z) = init((0, 0, 0), NEXTTIME(z)) = 2n = 128 \text{ for all z from X.}$$

Application of the production 3 was successful so next we have to apply one of the productions 4_j to the left-most entry of the nonterminal $\boldsymbol{A}(u, v, w)$. Here $u = (x, y, l) = (1, 1, 0)$, i.e., $l = 0$ and consequently $\boldsymbol{Q_4} = \boldsymbol{F}$. Thus, productions 4_j cannot be applied, so $\boldsymbol{F_F}$ is a permissible set here and we have to go back to the production 3.

We try to apply production to the nonterminal $\boldsymbol{A}(u, v, w)$ with $u = (x, y, l) = (1, 1, 0)$, v shown in Fig. 5, and $w = zero$. Obviously, $\boldsymbol{Q_3}(1, 1, 0) = \boldsymbol{T}$ and this production is applicable:

$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}(\boldsymbol{f}((1, 1, 0), v), v, zero)$.

As far as $(l = 0) \wedge (y = 1)$ and $v_{y+1} = v_2 = 0$,

$$\boldsymbol{f}(u, v) = (1, y+1, TIME(y+1) * v_{y+1}) = (1, 2, 0).$$

Therefor,

$$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}((1, 2, 0), v, zero)$$

A computation of function NEXTTIME takes place as follows:

$$NEXTTIME(z) = init((1, 1, 0), NEXTTIME(z)).$$

To prevent misunderstanding we have to remind that symbol "=" here means that value of the right side should be assigned to the left side, i.e., the new values of NEXTTIME are computed basing on the current values. Thus,

$$NEXTTIME(z) = 124 \text{ for all z from X.}$$

Application of the production 3 was successful so next again we will try to apply one of the productions 4_j. But $\boldsymbol{Q_4}(1, 2, 0) = \boldsymbol{F}$ and again we have to go back to production 3. $\boldsymbol{Q_3}(1, 2, 0) = \boldsymbol{T}$, this production is applicable, and this loop continues until u changes either way:

$$l = TIME(y+1) * v_{y+1} \neq 0 \quad \text{or} \quad y = 124.$$

In our case $v_{7+1} = 1 (\neq 0)$. Thus 8-th application of production 3 will result in the following string:

$$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}((1, 8, 5), v, zero)$$

because for $u = (1, 7, 0)$ y+1 corresponds to h1, $TIME(y+1) * v_{y+1} = TIME(h8) * 1 = 5$.

This means that point h1 is determined as the ending point for generating trajectories of robots which intercept motion of the BOMBER. The following derivation steps would allow us to find possible starting points of such trajectories.

The next attempt of applying production 4_j will result in failure because there no robots at point x = 1, i.e., at point a1, and $\boldsymbol{Q_4}(1, 8, 5) = \boldsymbol{F}$. Again we return to production 3 but with $l > 0$ and $x \neq 62$. This means the beginning of a new loop which consists of multiple applications of production 3 after failures of attempts to apply one of productions 4_j .

$$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}((2, 8, 5), v, zero)$$

$$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}((3, 8, 5), v, zero)$$

……

$$^{3}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\, \boldsymbol{A}((44, 8, 5), v, zero)$$

With $u = (44, 8, 5)$ this loop will be terminated because

$$\boldsymbol{Q_4}(44, 8, 5) = (ON(FIGHTER) = 44) \wedge (5 > 0) \wedge (\chi(BOMBER, FIGHTER) = 0) \wedge$$
$$(MAP_{f6,FIGHTER}(h1) = 5) = \boldsymbol{T}$$

which means that productions 4_j are applicable. These productions will generate intercepting trajectories from f6 to h1.

$$^{4j}\Rightarrow \boldsymbol{t}(BOMBER, t_B, 5)\boldsymbol{t}(\boldsymbol{h_j}(44, 8, 5), TIME(8))\boldsymbol{A}((44, 8, 5), v, \boldsymbol{g}(\boldsymbol{h_j}(44, 8, 5), zero))$$

In order to compute $\boldsymbol{h_j}(44, 8, 5)$ we have to generate all the shortest and admissible (of degree 2) trajectories from point f6 to h1 for robot FIGHTER (Table 4). The length of these trajectories should be less or equal $l = 5$.

$$TRACKS_{FIGHTER} = \{FIGHTER\} \times \Big(\bigcup_{1 \le k \le 5} L[\, \mathbf{G_t^{(2)}}(f6, h1, k, FIGHTER)].$$

$$TRACKS = \{(FIGHTER, t_1), (FIGHTER, t_2), (FIGHTER, t_3)\},\ m = 3 \text{ and}$$

$$\boldsymbol{h_1}(44, 8, 5) = (\text{FIGHTER}, t_1)$$
$$\boldsymbol{h_2}(44, 8, 5) = (\text{FIGHTER}, t_2)$$
$$\boldsymbol{h_2}(44, 8, 5) = (\text{FIGHTER}, t_3)$$

According to [25] there are three such trajectories (Fig. 7), and they are generated by the grammar $\mathbf{G_t^{(2)}}$. (Of course, there is one more trajectory, ***a***(f6)***a***(g5)***a***(h4)***a***(h3)***a***(h2)***a***(h1), which partially coincides with the main trajectory of the Zone and thus should be rejected.) Beginning with this step the derivation can be continued with three strings depending on the production applied on this step: 4_1, 4_2 or 4_3. It means we can derive *three* Zones with the same main trajectory and different intercepting trajectories from f6 to h1. Let us apply production 4_1 and continue derivation of Zone with the following trajectory

$$t_F = t_1 = \boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{a}(e4)\boldsymbol{a}(f3)\boldsymbol{a}(g2)\boldsymbol{a}(h1).$$

Thus, taking into account that TIME(8) = 5, we have

$$^{41}\Rightarrow \boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}((\text{FIGHTER}, t_F), 5)\boldsymbol{A}((44, 8, 5), v, \boldsymbol{g}(\text{FIGHTER}, t_F, zero)).$$

Next we have to compute $\boldsymbol{g}(\text{FIGHTER}, t_F, zero)$. According to Table 4, for all r X the r-th component of function $\boldsymbol{g}$ is as follows:

$$\boldsymbol{g_r}(\text{FIGHTER}, t_F, zero) = \begin{cases} 1, & \text{if } \boldsymbol{DIST}(r, \text{FIGHTER}, t_F) < 2n, \\ 0, & \text{if } \boldsymbol{DIST}(r, \text{FIGHTER}, t_F) = 2n, \end{cases}$$

The value of function ***DIST*** (x, FIGHTER, t_F) = k+1, where k is the number of symbol of the trajectory t_F, whose parameter value equals x. Consequently

$$\boldsymbol{DIST}(e5, \text{FIGHTER}, t_F) = 2$$
$$\boldsymbol{DIST}(e4, \text{FIGHTER}, t_F) = 3$$
$$\boldsymbol{DIST}(f3, \text{FIGHTER}, t_F) = 4$$
$$\boldsymbol{DIST}(g2, \text{FIGHTER}, t_F) = 5$$
$$\boldsymbol{DIST}(h1, \text{FIGHTER}, t_F) = 6$$

For the rest of x from X ***DIST*** (x, FIGHTER, t_F) = 2 x 62 =124. Thus for r {e5, e4, f3, g2, h1}={35, 28, 21, 15, 8} $\boldsymbol{g_r}(\text{FIGHTER}, t_F, zero) = 1$, for the rest of r $\boldsymbol{g_r} = 0$.

Now we can complete application of production 4_1. It remains to compute values of functional formula:

$$\text{NEXTTIME}(z) = \text{ALPHA}(z, (\text{FIGHTER}, t_F), 5–5+1).$$

As we know from previous steps NEXTTIME(x) = 124 for all x from X. Therefor according to Table 4

$$\boldsymbol{ALPHA}(x, \text{FIGHTER}, t_F, 1) = \begin{cases} \boldsymbol{min}(NEXTTIME(x), 1), & \text{if } \boldsymbol{DIST}(x, \text{FIGHTER}, t_F) < 124 \\ NEXTTIME(x), & \text{if } \boldsymbol{DIST}(x, \text{FIGHTER}, t_F) = 124. \end{cases}$$

Thus, for x {e5, e4, f3, g2, h1} ***ALPHA***(x, FIGHTER, t_F, 1) = 1, while for other x ***ALPHA***(x, FIGHTER, t_F, 1) = 124. The same values should be assigned to NEXTTIME(z).

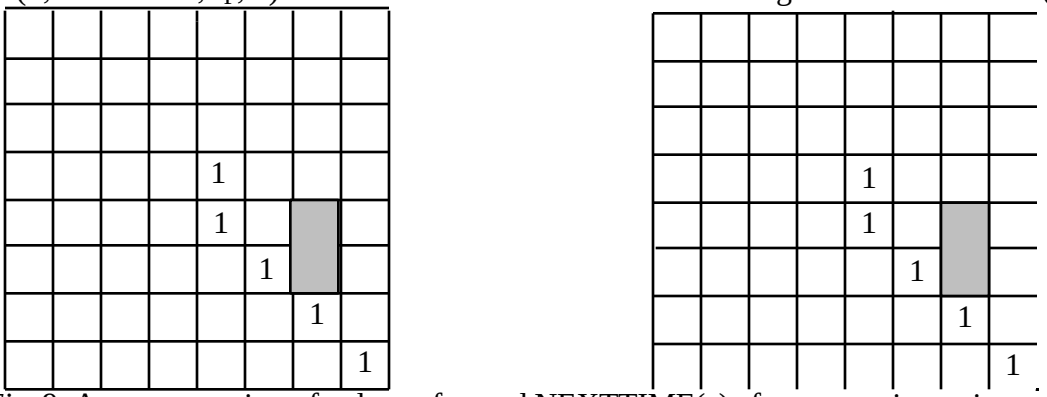

Fig. 9. A representation of values of *w* and NEXTTIME(z) after generating trajectory ***a***(f6)***a***(e5)***a***(e4)***a***(f3)***a***(g2)***a***(h1.

Values of function $\boldsymbol{g}$ and, consequently, values of components of vector w, different from zero, mark ending points of prospective trajectories of robots from P_1 which could support interception of BOMBER by protecting points the 1-st negation trajectories, points e5, e4, f3, g2, h1 in Fig. 9. These trajectories are called the *2-nd negation* trajectories. Values of NEXTTIME for the same points (Fig. 9) designate maximum lengths of those prospective trajectories. These values are equal 1 because trajectory t_F is an intercepting trajectory of maximum length (5). It means that no one robot has enough time to intercept BOMBER at point h1 while moving along the trajectory of a greater length. Thus there is no extra time for robots from P_1 to approach points of trajectory t_F (for possible protection) while robot FIGHTER is moving along t_F. Values of w and NEXTTIME are computed employing productions 3 and 4_j, while 1-st negation trajectories are generated. After completion of this generation these values will be assigned to v and TIME respectively (production 5) to be used for generation of the 2-nd negation trajectories.

Points e5, e4, f3, g2, h1 are considered as targets by the other side P_2 as well. It means that the grammar should generate trajectories of robots (if they exist) which could intercept motion of FIGHTER, and thus prevent interception of BOMBER, the *own* trajectories. By definition of the Grammar of Zones (Table 3, predicate $\boldsymbol{Q_4}$) the length of such trajectories is restricted by 1. (Obviously, there are no own trajectories in the problem shown in Fig. 7.)

Let us continue derivation of Zone. Production 4_1 was applied successfully, so we have to go to the production with label 3 and proceed with searching possible starting points of the trajectories with h1 as the ending point. We return to production 3 but with u =(44, 8, 5), i.e., with $l > 0$ and $x \neq 62$. This means the beginning of a new loop which consists of multiple applications of production 3 after failures of attempts to apply one of productions 4_j .

$$^{3}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}(\text{FIGHTER}, t_F, 5)\,\boldsymbol{A}(45, 8, 5),\ v,\ w)$$
$$^{3}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}((\text{FIGHTER}, t_F, 5)\,\boldsymbol{A}(46, 8, 5),\ v,\ w)$$
$$\ldots\ldots\ldots$$
$$^{3}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}(\text{FIGHTER}, t_F, 5)\,\boldsymbol{A}(62, 8, 5),\ v,\ w).$$

Computations of NEXTTIME(z) in production 3 will not change its values. With u = (62, 8, 5) this loop is terminated which means that no other starting points are found. Then a new loop begins. The grammar changes ending point of prospective trajectories:

$$^{3}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}(\text{FIGHTER}, t_F, 5)\,\boldsymbol{A}(1, 9, 0),\ v,\ w)$$
$$^{3}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}((\text{FIGHTER}, t_F, 5)\,\boldsymbol{A}(1, 10, 0),\ v,\ w)$$
$$\ldots\ldots\ldots$$

and eventually

$$^{3}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}(\text{FIGHTER}, t_F, 5)\,\boldsymbol{A}(1, 16, 4),\ v,\ w),$$

because for u = (1, 15, 0) y+1 corresponds to h2, TIME(y+1) * v_{y+1} = TIME(h2) * 1 = 4.

This means that point h2 is determined as the next ending point for generating trajectories of robots which can intercept motion of the BOMBER. The following derivation steps would allow us to look for possible starting points of such trajectories. Obviously, nothing will be found, except $\boldsymbol{a}$(f6)$\boldsymbol{a}$(g5)$\boldsymbol{a}$(h4)$\boldsymbol{a}$(h3)$\boldsymbol{a}$(h2), which will be rejected. The same negative result will be achieved with the next ending point, h3. The only intercepting trajectory to be found and accepted is as follows:

$$t_F^1 = t_1 = \boldsymbol{a}(\text{f6})\boldsymbol{a}(\text{g5})\boldsymbol{a}(\text{h4})$$

We have

$$^{4_1}\!=\!>\boldsymbol{t}(\text{BOMBER},t_B,5)\boldsymbol{t}(\text{FIGHTER},t_F,5)\boldsymbol{t}(\text{FIGHTER},t_F^1,2)\boldsymbol{A}((44,30,2),v,\ \boldsymbol{g}(\text{FIGHTER},t_F^1,w)),$$
$$\text{NEXTTIME(z)} = \boldsymbol{ALPHA}(\text{z, FIGHTER, } t_F^1,\ 2{-}2{+}1).$$

Application of production 4_1 will result in the change of the values of w and NEXTTIME shown in Fig. 10. Then we continue applying production 3 returning to it each time after unsuccessful attempt of applying production 4_j. This loop will be terminated when $\boldsymbol{Q_3}(u)$ = (x ≠ 62) (y ≠ 62) = $\boldsymbol{F}$. Next we have to go to production 5. This production is applicable because $\boldsymbol{Q_5}(w)$ = ($w \neq 0$) = $\boldsymbol{T}$ (current values of w are shown in Fig. 10). Thus,

$$^{5}\!=\!>\boldsymbol{t}(\text{BOMBER}, t_B, 5)\boldsymbol{t}(\text{FIGHTER}, t_F, 5)\boldsymbol{t}(\text{FIGHTER}, t_F^1, 2)\boldsymbol{A}((0, 0, 0),\ w,\ \textit{zero})$$
$$\text{TIME(z)} = \text{NEXTTIME(z)}$$

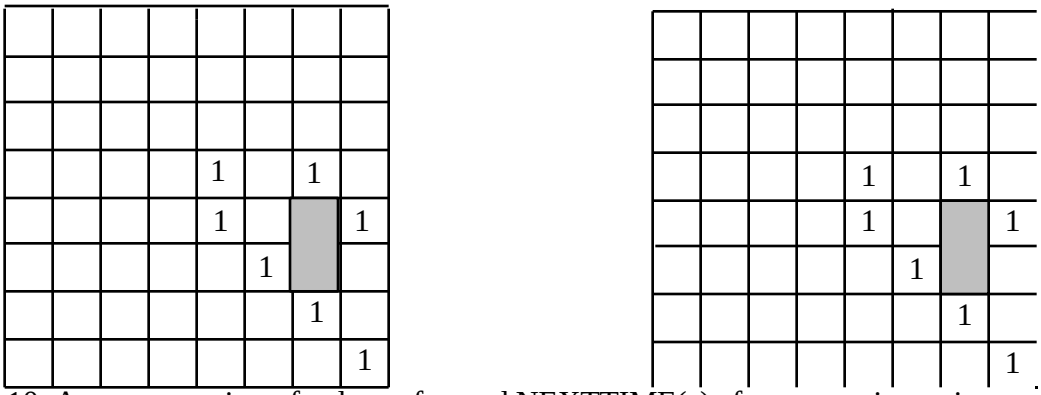

Fig. 10. A representation of values of *w* and NEXTTIME(z) after generating trajectory ***a***(f6)***a***(g5)***a***(h4).)

This is the completion of generation of the 1-st negation trajectories, so production 5 performs the assignment we promised above. Values of *w* are assigned to *v* while NEXTTIME(z) are assigned to TIME(z). All the steps, 3 and 4_j, which have been executed (or tried) for generating 1-st negation trajectories will be repeated for generating 2-nd negation. No one such trajectory should be found. The next return to production 5 will happen with *w* = *zero* (nothing is found). It means this production is not applicable, and we complete derivation applying production 6:

^{6}=>***t***(BOMBER, t_B, 5)***t***(FIGHTER, t_F, 5)***t***(FIGHTER, t_F^1, 2).

14. Zones for the Game of Chess.

The problem of programming the game of chess, is the most transparent example of a Linguistic Geometry application. This problem domain with the method informally described in [16] was the first application and experimental area for the formal linguistic approach. In this model of the Complex System (see Section 6) **X** is represented by 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 square x to square y; thus a point x is *reachable* from a point y for an element p, if a piece p can move from square x to square y according to the chess game rules; **ON(p)=x**, if a piece p stands on square x; **v(p)** is the value of piece p, e.g., P – 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 the King (suppose, this capture is permitted). The sets of WFF $\{ON(p_j) = x_k\}$ correspond to the lists of pieces with their coordinates. **TRANSITION(p, x, y)** represents the move of the piece p from square x to square y; if a piece of the opposing color stands on y, a capture is made.

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 a 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 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 [16].

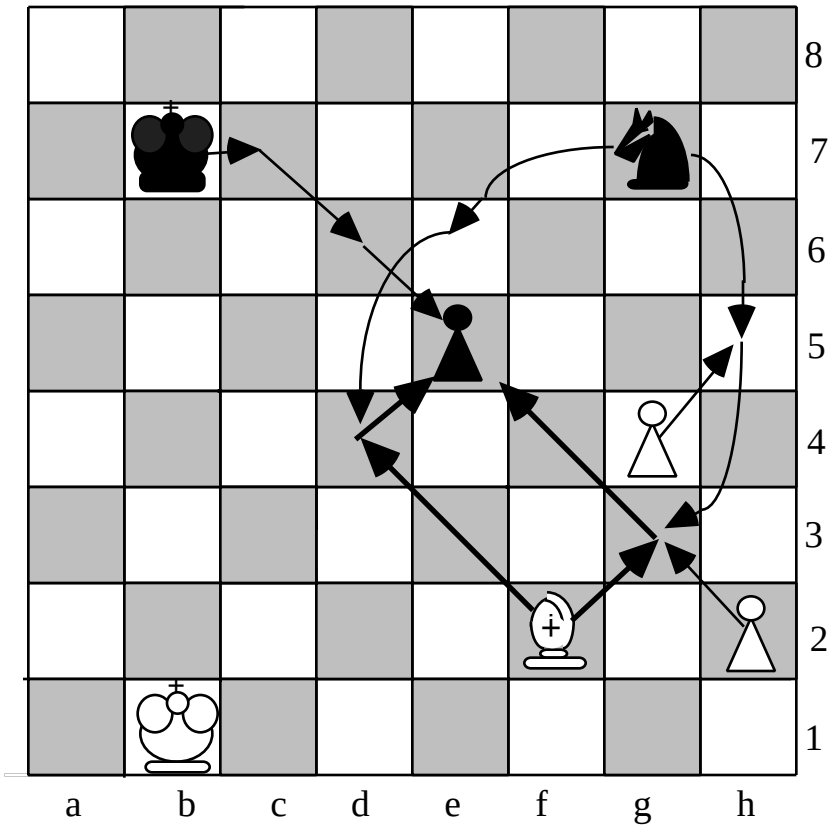

Fig. 11. Interpretation of the Language of Zones for chess model.

Let us consider an example of the Language of Zones for the chess model. We are going to present this language informally, listing Zones and trajectories, without explicit generating by the Grammar of Zones. An artificial chess position is shown in Fig. 11. Assuming that, the so-called horizon, H = 2 steps, in this range of lengths the only couple of attacking and attacked pieces are the Bishop on f2 and Pawn on e5, respectively. Thus, only such Zones can be generated. Trajectories ***a***(f2)***a***(g3)***a***(e5) and ***a***(f2)***a***(d4)***a***(e5) for the Bishop are the main trajectories of these Zones. They are shown by bold lines. All the other lines shown in Fig. 11 single out one Zone of the bundle of Zones generated by the grammar. The black side can intercept the Bishop employing one of the various intercepting trajectories, the 1-st negation trajectories. For example, the interception on square g3 can be accomplished by the black pieces located in the range of two steps from g3. (By definition of Zone it is generated in assumption that the protecting side is to move.) Thus one of the Knight's trajectories from g7 to g3, ***a***(g7)***a***(f5)***a***(g3) or ***a***(g7)***a***(h5)***a***(g3), should be included into the this Zone. Similarly either ***a***(g7)***a***(e6)***a***(d4), or ***a***(g7)***a***(f5)***a***(d4) can be included to intercept Bishop on d4. The last chance for interception is to approach the target, Pawn on e5, in 3 steps. It can be done by the King on b7 along one of two trajectories, ***a***(b7)***a***(c6)***a***(d5)***a***(e5) or ***a***(b7)***a***(c7)***a***(d6)***a***(e5). There are no other trajectories to prevent the attack. White side should include its own trajectories to support the attack, i.e., the motion of the Bishop along one of the main trajectories. By definition of Zone they are in the range of one step only. They are ***a***(h2)***a***(g3), ***a***(g4)***a***(h5) (if ***a***(g7)***a***(h5)***a***(g3) was included) or ***a***(g4)***a***(f5) (in case of ***a***(g7)***a***(f5)***a***(g3)).

15. Zones for the Scheduling Problem

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., a month, a 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 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, i.e., the amount of power to be lost without turning off customers. This amount varies daily.

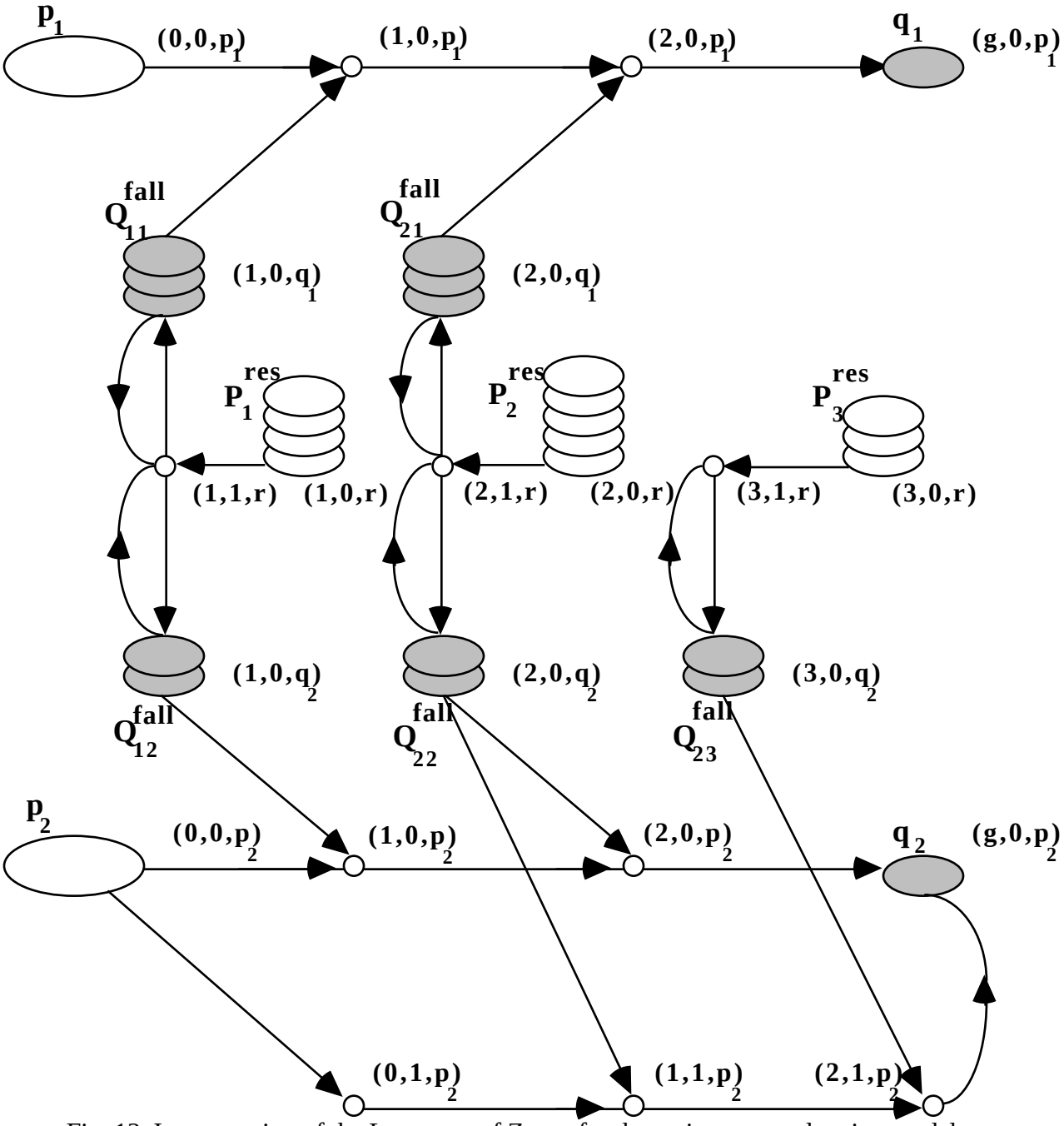

Fig. 12. Interpretation of the Language of Zones for the maintenance planning model.

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 constraint, the function *f(i)* of the 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.

In terms of the Complex System, this problem might be represented as a twin-set of *elements* and *points*. To avoid a long formal definition [19] which is far beyond the scope of this paper we present here a simplified example depicted in the Fig. 12. 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 are able to gain any of them, i.e., maintenance can not take place without provision of resources. This means we are forced to spend white discs of pyramids P_i^{res} exchanging them at the points (i, 1, r) with the black discs of Q_{ij}^{fall}. These actions can "clear" the paths for power units p_1 and p_2.

To clarify this problem and the network language representation let us consider the example depicted in Fig. 12 in detail. This is the ”toy” maintenance planning problem for two units over a period of three days:

$$w_1=5,\ w_2=2;\quad h_1=3,\ h_2=2;\quad x_1^{max}=x_2^{max}=2;\quad T_{max}=3;\quad f(1)=4;\ f(2)=5;\ f(3)=3.$$

(A reader should not be confused by the simplicity of the example shown in the Fig. 12. It is cited here only for clarification of our approach. For the practical applications described in Section 2 hundreds and even thousands of power units were considered; different kinds of resources were taken into account, including those which required some time to be delivered to the places of maintenance [18, 19].)

From Fig. 12 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

$$(0,0,p_2) \text{ –> } (1,0,p_2) \text{ –> } (2,0,p_2) \text{–>} (g,0,p_2) \text{ or}$$
$$(0,0,p_2) \text{–>} (0,1,p_2) \text{–>} (1,1,p_2) \text{–>} (2,1,p_2) \text{–>} (g,0,p_2),$$

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 the fall in power of the energy-producing system during the time of the power units’ maintenance.

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 the 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 the pyramids of resources have to be placed several steps from the points of exchange which means that resource 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. 12), 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 the insufficiency of the 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.

Due to the instantaneous availability of resources mentioned above all the Zones generated for this example are very "simple." They have 1-st negation trajectories of the length 1 only. Nevertheless this example is interesting because the Language of Zones here corresponds to the network of Zones subordinate to each other. The highest level of this hierarchy consists of two Zones Z_1 and Z_2 of the power units p_1 and p_2. The first one Z_1 includes the main trajectory

$$\boldsymbol{a}(0, 0, p_1)\boldsymbol{a}(1, 0, p_1)\boldsymbol{a}(2, 0, p_1)\boldsymbol{a}(g, 0, p_1)$$

and 1-st negation trajectories

$\boldsymbol{a}(1, 0, q_1)\boldsymbol{a}(1, 0, p_1)$ for the elements from Q_{11}^{fall} and $\boldsymbol{a}(2, 0, q_1)\boldsymbol{a}(2, 0, p_1)$ from Q_{21}^{fall}.

The second Zone Z_2 includes two main trajectories

$$\boldsymbol{a}(0, 0, p_2)\boldsymbol{a}(1, 0, p_2)\boldsymbol{a}(2, 0, p_2)\boldsymbol{a}(g, 0, p_2) \text{ and}$$
$$\boldsymbol{a}(0, 0, p_2)\boldsymbol{a}(0, 1, p_2)\boldsymbol{a}(1, 1, p_2)\boldsymbol{a}(2, 1, p_2)\boldsymbol{a}(g, 0, p_2)$$

and 1-st negation trajectories:

$\boldsymbol{a}(1, 0, q_2)\boldsymbol{a}(1, 0, p_2)$ for the elements from Q_{12}^{fall} ,

$\boldsymbol{a}(2, 0, q_2)\boldsymbol{a}(2, 0, p_2)$ and $\boldsymbol{a}(2, 0, q_2)\boldsymbol{a}(1, 1, p_2)$ for the elements from Q_{22}^{fall},

$\boldsymbol{a}(3, 0, q_2)\boldsymbol{a}(2, 1, p_2)$ for the elements from Q_{23}^{fall}.

The next level of this hierarchy includes many Zones Z_{ij}^{res} which provide resources for maintenance work. For example, Zones Z_{i1}^{res} include the main trajectories $\boldsymbol{a}(1,0,r)\boldsymbol{a}(1,1,p_2)\boldsymbol{a}(1,0,q_1)$ for different elements from P_1^{res} while Z_{i2}^{res} include $\boldsymbol{a}(1,0,r)\boldsymbol{a}(1,1,p_2)\boldsymbol{a}(1,0,q_2)$. Zones Z_{i1}^{res} and Z_{i2}^{res} are intended for liquidation of the elements from Q_{11}^{fall} and Q_{12}^{fall}, respectively, which means that providing of resources is required. First negation trajectories for Z_{i1}^{res} are $\boldsymbol{a}(1,0,q_1)\boldsymbol{a}(1,1,r)$ while for Z_{i2}^{res} $\boldsymbol{a}(1,0,q_2)\boldsymbol{a}(1,1,r)$. The trajectories supporting the attack include $\boldsymbol{a}(1,0,r)\boldsymbol{a}(1,1,r)$. These Zones are subordinate to Zones Z1 and Z2. Similar Zones are generated for P_2^{res} and P_3^{res}.

For complex real-world problems subordinate Zones usually have longer main trajectories which means that resources delivery in this case requires some time.

16. Conclusions and expectations

The results presented in this paper outline the main ideas of description and generating techniques of Two-dimensional Linguistic Geometry. There are many points of growth and research in this field.

The most exciting results are expected in the field of deeper study of the properties of translations. These studies should allow us to give a formal and constructive solution of the Frame Problem [13, 15] for the Complex System. It means that eventually we are going to give an effective description of the "dynamic" boundaries between the actual and outdated information about the Complex System in a process of search.

An efficient program implementation of the general network grammar, like the Grammar of Zones, especially in parallel environment, would permit us to generate applications by turning this general grammar for the specific practical problems. Theoretical problems of complexity and accuracy of solutions are of great interest as well. Of course, very interesting results should be expected in the field of transfer of the linguistic approach to different complex hierarchical systems.

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, (1991).
2. H.A.Simon, *The Sciences of the Artificial*, *2-nd ed*., The MIT Press: Cambridge, MA, (1980).
3. K.S. Fu, *Syntactic methods in pattern recognition*, *Mathematics in Science and Engineering*, vol. 112, edited by Richard Bellman, Academic Press: New York, (1974).
4. K.S. Fu, *Digital pattern recognition*, Springer-Verlag, New York, (1980).
5. R.N. Narasimhan, Syntax–Directed Interpretation of Classes of Pictures, *Communications of the ACM*, vol. 9, 166–173, (1966).
6. T. Pavlidis, Linear and Context-Free Graph Grammars, *Journal of the ACM*, vol. 19, 11–22, (1972).
7. A.C. Shaw, A Formal Picture Description Scheme as a Basis for Picture Processing System, *Information and Control*, vol. 19, 9–52, (1969).
8. J. Feder, Plex languages, *Information Sciences*, vol. 3, 225–241, (1971).
9. J.L. Pfaltz and A. Rosenfeld, WEB Grammars, in *Artificial Intelligence, Proceedings of the 1-st International Joint Conference* , Washington, D.C., 609–619, (May 1969).
10. D.E. Knuth, Semantics of Context-Free Languages, *Mathematical Systems Theory*, vol. 2-2, 127–146, (1968).
11. D.J. Rozenkrantz, "Programmed Grammars and Classes of Formal Languages", *Journal of the ACM*, vol. 16-1, 107–131, (1969).
12. N.G. Volchenkov, The Interpreter of Context-Free Controlled Parameter Programmed Grammars, in *Cybernetics Problems. Intellectual Data Banks*, ed. by L.T. Kuzin, The USSR Academy of Sciences: Moscow, 147–157, (1979) [in Russian].
13. R.E. Fikes and N.J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, *Artificial Intelligence*, vol. 2, 189–208, (1971).
14. E.D. Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence*, vol. 5-1, 115-135, (1974).
15. J. McCarthy and P.J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence*, vol. 4, 463–502, (1969).
16. M.M. Botvinnik, *Computers in Chess: Solving Inexact Search Problems*. Springer Series in Symbolic Computation, Springer-Verlag: New York , (1984).
17. B.M. Stilman, Hierarchy of Formal Grammars for Solving Search Problems, in *Artificial Intelligence. Results and Prospects, Proceedings of the International Workshop*, Moscow, 63–72, (1985), [in Russian].
18. 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, 147-153, (1983) [in Russian].
19. M. Botvinnik, E. Petriyev, A. Reznitskiy, et al., Application of New Method for Solving Search Problems For Power Equipment Maintenance Scheduling", *Economics and Mathematical Methods*, vol. 19-6, 1030-1041, (1983) [in Russian].
20. N.J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, (1980).
21. B. Stilman, A Syntactic Structure for Complex Systems, *Proc. of the Second Golden West International Conference on Intelligent Systems*, Reno, NE, 269-274, (June 1992).
22. B. Stilman, A Geometry of Hierarchical Systems: Generating Techniques, *Proc. of the Ninth Israeli Conference on Artificial Intelligence and Computer Vision*, Tel Aviv, Israel, (Dec. 1992) (to appear).
23. B. Stilman, A Syntactic Approach to Geometric Reasoning about Complex Systems, *Proc. of the Fifth International Symposium on Artificial Intelligence*, Cancun, Mexico, (Dec. 1992) (to appear).
24. B. Stilman, A Linguistic Geometry of Complex Systems, *Annals of Mathematics and Artificial Intelligence*, (1992), (submitted).

25. B. Stilman, A Linguistic Approach to Geometric Reasoning, *Int. J. Computers and Mathematics with Applications*, (1992), (to appear).
26. B. Stilman, Two-dimensional Structures in Linguistic Geometry, Dept. of Computer Science, University of Colorado at Denver, Denver, CO, *Technical Report* TR-20, (Sept. 1992).
27. M. Stefik, Planning and meta-planning (MOLGEN: Part 2), *Artificial Intelligence*, 16-2, 141-169, (1981).
28. D. Chapman, Planning for conjunctive goals, *Artificial Intelligence*, 32-3, (1987).
29. A. Rosenfeld, *Picture Languages: Formal Models for Picture Recognition*, Academic Press, (1979).