DRAFT

A Linguistic Geometry of the Chess Model

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

ABSTRACT

In order to discover the inner properties of human expert heuristics, which are successful in computer chess, to investigate these heuristics, improve them and, further, apply to various complex control systems, we develop a formal theory, the so-called Linguistic Geometry. This research includes the development of syntactic tools for *knowledge representation* and *reasoning* about the game of chess as about a hierarchical complex system. It relies on the formalization of *searchheuristics*, which allow to decompose this game into the hierarchy of subsystems, and thus solve this problem reducing the search drastically. The hierarchy of subsystems is represented as a *hierarchy of formal attribute languages*. Syntactic tools generating the hierarchy of languages, the controlled grammars, are introduced in this paper. The performance of the grammar generating paths of pieces is considered in detail. The generation of the entire hierarchy of languages is considered on example of the R.Reti endgame.

INTRODUCTION

The group led by Prof. Botvinnik attacked the computer chess problem by simulating the method of a human chess master. The main idea was to replace a one-level complex system, as the game of chess use to be seen, by a multi-level hierarchical model that could allow us to break the system into subsystems, to search these subsystems separately and in combinations, and then combine optimal solutions for subsystems as approximately optimal for the entire system. This approach reduced the search drastically, down to a hundred of moves. However, the main difficulty was transferred from the development of efficient search methods to the design of methods for implementation of this decomposition.

Basically, the questions are as follows. What are these subsystems? What are the most efficient knowledge representation mechanisms to handle them? Is this decomposition dynamic, i.e., does it change when we move from one position to another during the search? If so, how to generate new subsystems, how to understand that subsystem is obsolete and destroy it, how to re-generate old subsystem by changing it, and, thus, avoid tremendous recomputation. How to organize the search within the subsystem and evaluate the results?

These and many other questions have been answered in our research. However, some answers are ambiguous, some questions are still open. Different subsystems were developed. They are called trajectories, zones, chains of trajectories, and so on. The results were presented by Botvinnik (1970, 1975, 1984) with contributions of members of research team. Of course, many interesting advances were made later.

The language used for the development of the model and presentation of the results was either the language of plausible discussions (for general ideas) or the lower-level algorithmic language close to programming languages (for the description of implementations). There was a huge gap between these two languages. The model itself could not be expressed adequately. The language of plausible discussions is very ambiguous for such a complex subject, while an algorithmic language is too detailed, and, thus, useless as well. This inadequacy caused many problems, and not only for presentation. Following Russian proverb, often "we could not see a forest behind the trees." It seems that each new result was not really *built* into the model but simply added to it violating previous results. The model looked like a collection of bright ideas, complex algorithms, and interesting results. We needed a mathematical skeleton, something like a search tree for conventional search models.

For many years I was looking for adequate mathematical tools. Eventually it resulted in the development of a Hierarchy of Formal Languages (Stilman, 1985) that emerged later into a Linguistic Geometry (Stilman, 1992-1993). These formal, expressive tools are intended for

investigation and development of all the results achieved in the PIONEER project. In particular, employing these tools I answer in a formal way the questions listed above. Based on that I investigate our later results, develop them and go ahead. These new tools allowed not only to present the results formally, but to prove the correctness of algorithms and data structures. One of the newest results on correctness of computation of the piece planning path is considered in this paper. This research already allowed to demonstrate the errors and correct them in algorithms, which had been implemented as subroutines of the PIONEER program and executed for a long time. These were some principle mistakes in the adjustment of the model in the process of search. It is very unlikely they could be found without formal methods. Some advances were made in the evaluation of the computational complexity of this model. I hope in the future to approach a formal evaluation of the accuracy of the solutions. Actually, employing these tools I hope to answer the most intriguing question: why the PIONEER finds solutions of complex positions, why it fails other cases, and what should be corrected.

The application of Linguistic Geometry is far beyond the framework of the chess problem. There are many real-world problems where human expert skills in reasoning about complex systems are incomparably higher than the level of modern computing systems. At the same time there are even more areas where advances are required but human problem-solving skills can not be directly applied. For example, there are problems of planning and automatic control of autonomous agents such as space vehicles, stations and robots with cooperative and opposing interests functioning in a complex, hazardous environment. Reasoning about such complex systems should be done automatically, in a timely manner, and often in a real time. Moreover, there are no high-skilled human experts in these fields ready to substitute for robots (on a virtual model) or transfer their knowledge to them. There is no grand-master in robot control, although, of course, the knowledge of existing experts in this field should not be neglected – it is even more valuable. Due to the special significance of these problems, the quality of solutions must be very high and usually subject to continuous improvement. Thus, it is very important to study human expert reasoning about similar complex systems in the areas where the results are successful, e.g., in computer chess, in order to discover the keys to success, and then apply and adopt these keys to the new, as yet, unsolved problems. It should be considered as investigation, development, and consequent expansion of advanced human expert skills into new areas. The Linguistic Geometry provides formal tools for this investigation and transfer.

THEORETICAL BACKGROUND

In the 1960's a formal syntactic approach to the investigation of properties of natural language resulted in the fast development of a theory of formal languages by Chomsky (1963), Ginsburg (1966), Knuth (1968), Rozenkrantz (1969), and others. This development provided an interesting opportunity for dissemination of this approach to different areas. In particular, there came an idea of analogous linguistic representation of images. This idea was successfully developed into syntactic methods of pattern recognition by Fu (1982), Narasimhan (1966), and Pavlidis (1972), and picture description languages by Shaw (1969), Feder (1971), Phaltz and Rosenfeld (1969). The power of a linguistic approach might be explained, in particular, by the recursive nature and expressiveness of language generating rules, i.e., formal grammars.

Searching for the adequate mathematical tools formalizing human heuristics of dynamic hierarchy for the game of chess, we have transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems (Stilman, 1985). However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. They should describe mathematically all of the essential syntactic and semantic features of the system and search, and be easily generated by certain controlled grammars. The origin of such languages can be traced back to the origin of SNOBOL-4 programming language and the research on programmed attribute grammars and languages by Knuth (1968), Rozenkrantz (1969), and Volchenkov (1979). A mathematical environment (a "glue") for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Fikes (1971), Nilsson (1980),

Sacerdoti (1975), and McCarthy, Hayes (1969), and others (Stefik, 1981, Chapman, 1987, McAllester at al, 1991) based on first order predicate calculus.

CLASS OF PROBLEMS

A *class of problems* to be studied are problems of optimal operation of a complex system. This system is considered as a twin-set of *elements* and *points*(locations) 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 attack and destroy opposite side *elements* and to protect its own. Each side aims to maximize a gain, the total value of opposite *elements* destroyed and withdrawn from the system. Such a withdrawal happens if an attacking *element* comes to the point where there is already an *element* of the opposite side. Obviously, the game of chess is such a problem with pieces substituting for elements and squares of the chess board as points.

A formal definition is as follows. A **Complex System** is the following eight-tuple:

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

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;

$\mathbf{R_p(x,y)}$ is a set of binary relations of *reachability* in X (x and y are from X, p from P);

ON(p)=x, where ON is a partial function of *placement* from P into X;

v is a function on P with positive integer values; it describes the *values* of elements.

The Complex System searches the state space, which should have initial and target states;

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

TR is a set of operators, TRANSITION(p, x, y), of transition of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed and added to the description of the state), and of WFF of applicability of the transition. Here,

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

Add list: ON(p)=y;

Applicability list: (ON(p)=x)^R_p(x,y),

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out in turn with participation of elements p from P_1 and P_2 respectively; omission of a turn is permitted.

According to definition of the set P, the elements of the System are divided into two subsets P_1 and P_2. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., R_p(x,y) holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from point x to point y. The element q from point y must be withdrawn (eliminated) if p and q belong to the different subsets P_1 and P_2.

The problem of the optimal operation of the System is considered as a search for the optimal sequence of transitions leading from one of the initial states of S_i to a target state S of S_t. The target states are described employing 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

$$\mathbf{m}(S)=\sum v(p_i)-\sum v(p_j), \qquad (1)$$

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 the search of the optimal sequence of transitions leading to the target state, we could use formal methods like those in the problem-solving system STRIPS (Fikes, Nilsson, 1971) nonlinear planner NOAH (Sacerdoti, 1975), or in subsequent planning systems. However, the search would have to be made in a space of a huge dimension (for nontrivial examples). Thus, in practice no solution would be obtained.

We devote ourselves to the search for an approximate solution of a reformulated problem. A *hierarchy* to be studied is the hierarchy of subsystems introduced in the problems considered above by the highly-skilled chess experts. This introduction is as follows (Botvinnik, 1975, 1984). A one-goal, one-level system should be substituted for a multi-goal multi-level system by introducing intermediate *goals* and breaking the system down into subsystems striving to attain these goals. The goals of the subsystems are individual but coordinated within the main mutual goal. For example, each second-level subsystem includes elements of both sides: the goal of one side is to attack and gain some element (a target), while the other side tries to protect it. In chess, it means the selection of a couple of pieces of opposing sides: one – as an attacking element, and the other – as a local target, generation of the paths for approaching the target, as well as the paths of other pieces supporting the attack or protecting the target.

A HIERARCHY OF LANGUAGES

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

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

$$\boldsymbol{a}(x_1)\boldsymbol{a}(x_2)\ldots\boldsymbol{a}(x_n), \qquad (2)$$

where values of parameters incorporate the semantics of the problem domain. They form the so-called *Language of Trajectories*. For example, for the lower level subsystems in the chess model $x_1, x_2,\ldots, x_n$ are the coordinates of squares of the chess board and $\boldsymbol{a}(x_1)\boldsymbol{a}(x_2)\ldots\boldsymbol{a}(x_n)$ represents a trajectory (a planning path) of a chess piece from the square x_1 to x_n through squares of stops $x_2, x_3,\ldots, x_{n-1}$.

A second level subsystem should be represented as a similar string with parameters:

$$\boldsymbol{t}(p_1, t_1, \tau_1)\boldsymbol{t}(p_2, t_2, \tau_2)\ldots\boldsymbol{t}(p_k, t_k, \tau_k), \qquad (3)$$

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

Thus, using strings of (2), we can represent paths of system's elements, and with the strings of (3), networks of certain paths unified by the mutual goal. In the chess model such a network represents a network of planning paths for a local fight. Strings (3) form the *Language of Trajectory Networks*.

The system functions by moving from one state to another; that is, the motion of an element from one point to another causes an adjustment of the hierarchy of languages. This adjustment can be represented as a mapping (*translation*) to some other hierarchy (actually, to the new state of the same hierarchy). Thus, the functioning of the system, in a process of the search, generates a *tree* of translations of the hierarchy of languages. This tree can be represented as a string of the highest level formal language, the *Language of Translations*.

The search for an optimal (suboptimal) operation, i.e., optimal variant in chess, in the new

system is considered as a process of generation and interaction of networks of the form (3). This process results in a highly reduced search tree which is represented as a string of the Language of Translations.

A GEOMETRY OF COMPLEX SYSTEM

To create and study a hierarchy of dynamic subsystems we have to investigate and use geometrical properties of the Complex System. Consider the following definition of the function MAP.

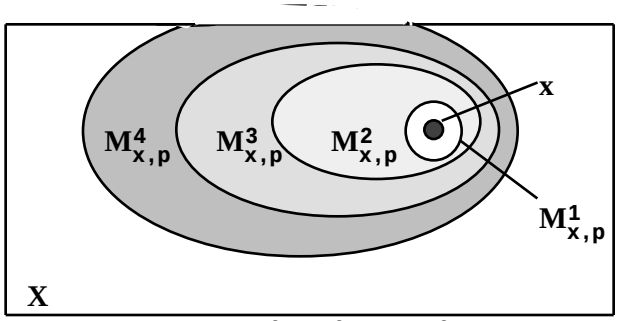

Figure 1. Interpretation of the family of reachability areas

A **map of the set X** relative to the point x and element p for the Complex System is the mapping: $\mathbf{MAP_{x,p}}$: X —> $\mathbf{Z_+}$, (where x is from X, p is from P), which is constructed as follows. We consider a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets of X $\{M^k_{x,p}\}$ (figure 1)**:**

k=1: $M^k_{x,p}$ is a set of points *m reachable in one step* from x: $R_p(x,m)$=T;

k>1: $M^k_{x,p}$ is a set of points *reachable in k steps and not reachable in k*-1 *steps*, i.e., points *m* reachable from points of $M^{k-1}_{x,p}$ and not included in any $M^i_{x,p}$ with numbers *i* less than *k*.

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

In the remainder points let

$MAP_{x,p}(y)=2n$, if $y \neq x$ (n is the number of points in X)**;**

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

It is easy to verify that the map of the set X for the given element p from P defines an *asymmetric distance function* on X:

1. $MAP_{x,p}(y) > 0$ for $x \neq y$; $MAP_{x,p}(x)=0$;
2. $MAP_{x,p}(y)+MAP_{y,p}(z) \geq MAP_{x,p}(z)$.

If R_p is a symmetric relation,

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

In this case each of the elements p from P specifies on X its *own* ***metric*****.**

CHESS AS A COMPLEX SYSTEM

The game of chess is the most transparent example of the Linguistic Geometry application.

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

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

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

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

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

S_i is an arbitrary initial chess position for analysis, or the starting position of the game;

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 in each position.

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 affected.

The chess problem does not completely meet the requirements of the general definition of the Complex System. For simplicity we have neglected such an important chess concept as blockade: in the Complex System several elements (pieces of the same color) can stand on the same point (square). Besides that, we have neglected certain specific chess features, such as castling, capture en passant, pawn promotion, etc. All these chess complications are not crucial for our model; it is not difficult to define a subclass of Complex Systems where all these features will be taken into account.

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

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

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

When implementing the geometrical model for the chess problem, it was necessary to give a tabular specification of the function MAP, in order to increase the efficiency of the program PIONEER (Botvinnik, 1970, 1975 – particularly appendixes). For this, in accordance with the relations R_p (the chess rules of movement of the pieces), seven square tables 15 x 15 were specified (figure 2).

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

Figure 2. A superposition of tables 8 x 8 and 15 x 15 for the Rook standing on c2

Each of the tables was filled with the numbers for one of the chess piece types according to the following principle: the piece is placed on the central square of the table (0 is written there); the remaining squares are filled with the numbers equal to the number of moves necessary for the piece to reach the given square from the central square along the shortest path. These tables may be unified in the form of the following table T15(v_1, v_2, *f*) with the dimension 15 x 15 x 7. (For all x of X, x=(x_1, x_2), x_1=1, 2,..., 8, x_2=1, 2,..., 8, where x_1 and x_2 correspond to files and rows of

the chessboard, respectively.) Then

$$MAP_{x,p}(y)=T15(v_1, v_2, f), \quad (4)$$

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

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

LANGUAGE OF TRAJECTORIES

Here, we define the lowest-level language of the hierarchy of languages, the Language of Trajectories. It serves as a building block to create the upper-level languages. The Language of Trajectories actually is a formalization of the set of lowest-level subsystems, the set of paths between different points of the Complex System. An element might follow a path to achieve the goal "connected with the ending point".

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:

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

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 coincide. $P(t_o)=\{x, x_1, ..., x_l\}$ is the set of parameter values of the trajectory t.

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 figure 3.

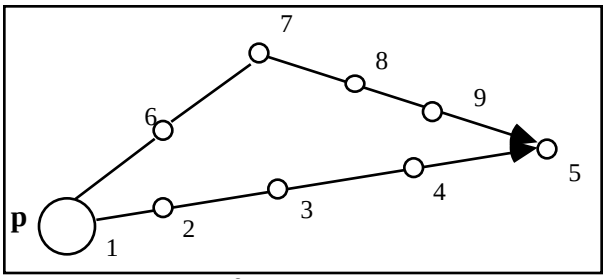

Figure 3. Interpretation of shortest and admissible trajectories

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 figure 3, 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.

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.

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 figure 3. 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 figure 3) and going to and from this point to the end along the shortest trajectories. Thus, in this case, an element should move along an admissible trajectory of degree 2.

A ***Language of Trajectories*** $L_t^H(S)$ for the Complex System in 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 grammar shown in figures 4, 5 is intended to generate shortest trajectories. These type of grammars are called controlled grammars (Stilman, 1992c, 1993b). A formal definition of this class is beyond the scope this paper. We demonstrate the details of the grammar on example of actual generation of trajectories for the chess model (next section).

L	Q	Kernel	F_T	F_F
1	Q_1	$S(x,y,l) \rightarrow A(x, y, l)$	*two*	ø
2_i	Q_2	$A(x,y,l) \rightarrow a(x)A(next_i(x,l),y,f(l))$	*two*	3
3	Q_3	$A(x, y, l) \rightarrow a(y)$	ø	ø

Here

$V_T = \{a\}$ is the alphabet of terminal symbols,

$V_N = \{S, A\}$ is the alphabet of nonterminal symbols,

$V_{PR} = Truth \cup Pred \cup Con \cup Var \cup Func \cup$ {symbols of logical operations and "="} is the alphabet of the first order predicate calculus PR,

$Truth = \{T, F\}$

$Pred = \{Q_1, Q_2, Q_3\}$ are predicate symbols:

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

$Q_2(l) = (l \geq 1)$

$Q_3 = T$

$Var = \{x, y, l\}$ are variables;

$Con = \{x_o, y_o, l_o, p\}$ are constants;

$Func = Fcon$ are functional symbols;

$Fcon = \{f, next_1, \ldots, next_n\}$ (n=|X|, number of points in X),

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

($next_i$ is defined below)

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

Parm: $S \rightarrow Var$, $A \rightarrow Var$, $a \rightarrow \{x\}$, is such a mapping that matches each symbol of the alphabet $V_T \cup V_N$ a set of formal parameters;

$L = \{1,3\} \cup two$, $two = \{2_1, 2_2, \ldots, 2_n\}$ is a finite set called the set of labels; labels of different productions are different;

Q_i are the WFF of the predicate calculus PR, the conditions of applicability of productions;

F_T is a subset of L of labels of the productions permitted on the next step derivation if Q=T; it is called a permissible set;

F_F is analogous to F_T but these productions are permitted in case of Q=F.

At the beginning of derivation:

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

next_i is defined as follows:

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

SUM={v |v X, $MAP_{x_o,p}(v)+MAP_{y_o,p}(v)=l_o$}

$ST_k(x)$={v | v from X, $MAP_{x,p}(v)=k$},

$MOVE_l(x)$ is an intersection of the following sets: $ST_1(x)$, $ST_{l_o-l+1}(x_o)$ and SUM.

If

$MOVE_l(x)=\{m_1, m_2, \ldots, m_r\} \neq \emptyset$

then

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

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

otherwise

$next_i(x,l)=x$.

Figure 4. A grammar of shortest trajectories $\mathbf{G_t^{(1)}}$

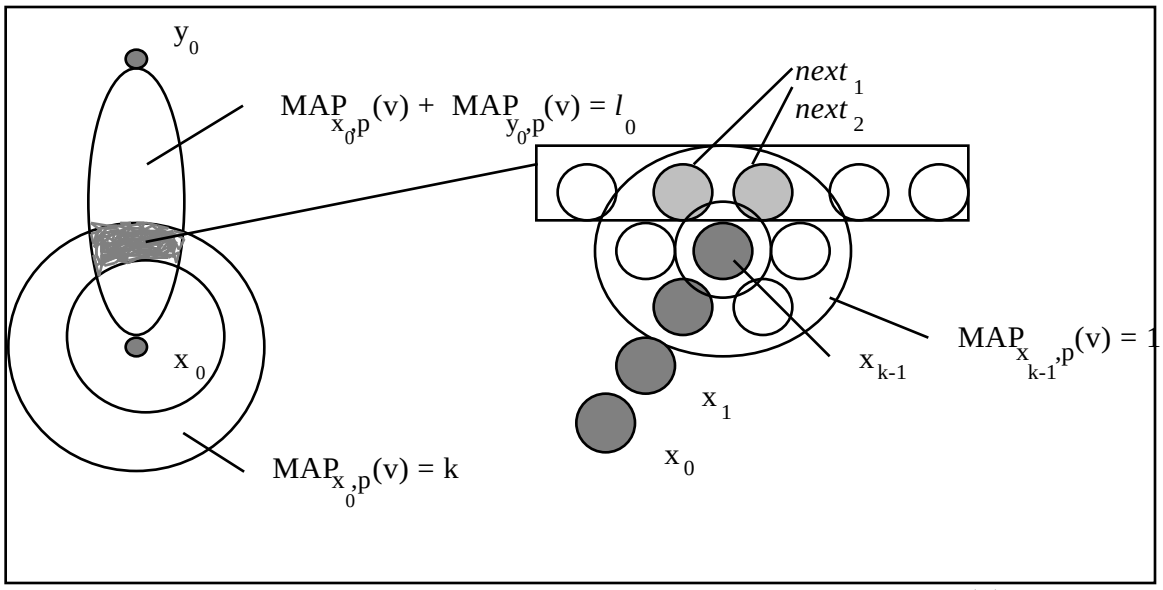

Figure 5. Interpretation of the function ***next_i*** of the grammar $\mathbf{G_t^{(1)}}$.

GENERATION OF TRAJECTORIES ON THE CHESS-BOARD

Let us show the generation of trajectories for the King from f6 to h1. Values of $MAP_{f6,\ K}$ are shown in figure 6. Thus, the distance from f6 to h1 for the King is equal to 5. Applying grammar $\mathbf{G_t^{(1)}}$ (production 1) we have:

$$\boldsymbol{S}(f6, h1, 5) \Rightarrow^1 \boldsymbol{A}(f6, h1, 5) \Rightarrow^{21} \boldsymbol{a}(f6)\boldsymbol{A}(next_1(f6, 5), h1, 5).$$

Thus we have to compute MOVE (see definition of the function $next_i$ from the grammar $\mathbf{G_t^{(1)}}$, figure 4, 5). First, we have to determine the set of SUM, that is, we need to know values of $MAP_{f6,K}$ and $MAP_{h1,K}$(shown in figure 6) on X. Adding these tables as matrices we compute SUM ={v | v X, $MAP_{f6,\ K}(v)+MAP_{h1,K}(v)=5$} (figure 6).

The next step is the computation of $ST_1(f6)=\{v \mid v \text{ from X}, MAP_{f6,K}(v)=1\}$. It is shown in figure 7. In order to complete computation of the set $MOVE_5(f6)$ we have to determine the following intersection: $ST_1(f6)$, $ST_{5-5+1}(f6)=ST_1(f6)$ and SUM.

Consequently, $MOVE_5(f6)=\{e5, f5, g5\}$, and $next_1(f6, 5)=e5$, $next_2(f6, 5)=f5$, $next_3(f6, 5)=g5$.

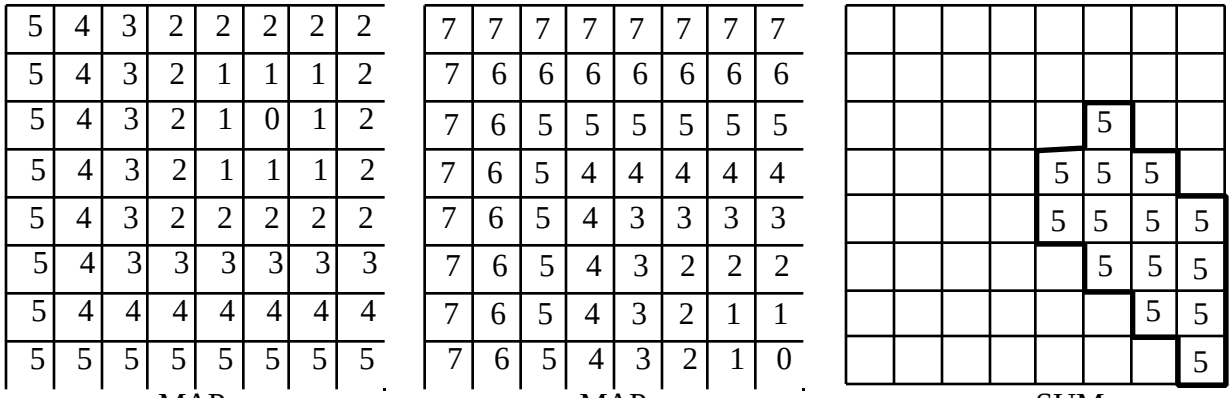

Figure 6. Trajectory generation: computation of SUM

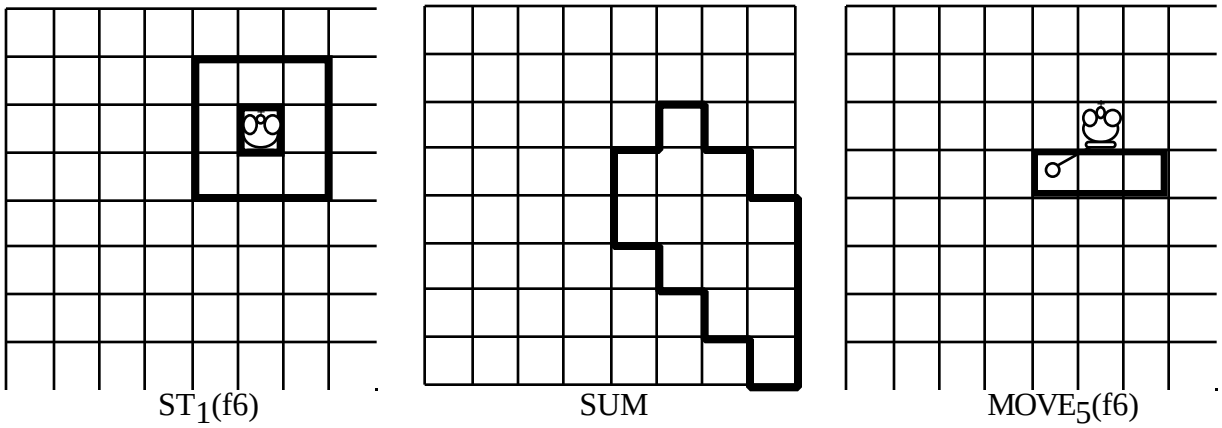

Figure 7. Trajectory generation: the first step

Since the number of different values of *next* is equal to 3 (here r=3, see definition of the function *next*, figures 4,5) we could branch at this step, apply productions 2_1, 2_2 and 2_3 simultaneously, and continue all the derivations independently. This could be accomplished in a parallel computing environment. Let us proceed with the first derivation.

$$\boldsymbol{a}(f6)\boldsymbol{A}(e5,h1,4) \xrightarrow{2_1} \boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{A}(next_1(e5,4),\ h1,3)$$

We have to compute $next_1(e5, 4)$ and, as on the preceding step, we have to determine $MOVE_4(e5)$. To do this we have to compute $ST_1(e5)$ and $ST_{5-4+1}(f6) = ST_2(f6)=\{v \mid v \quad X, MAP_{f6,K}(v)=2\}$

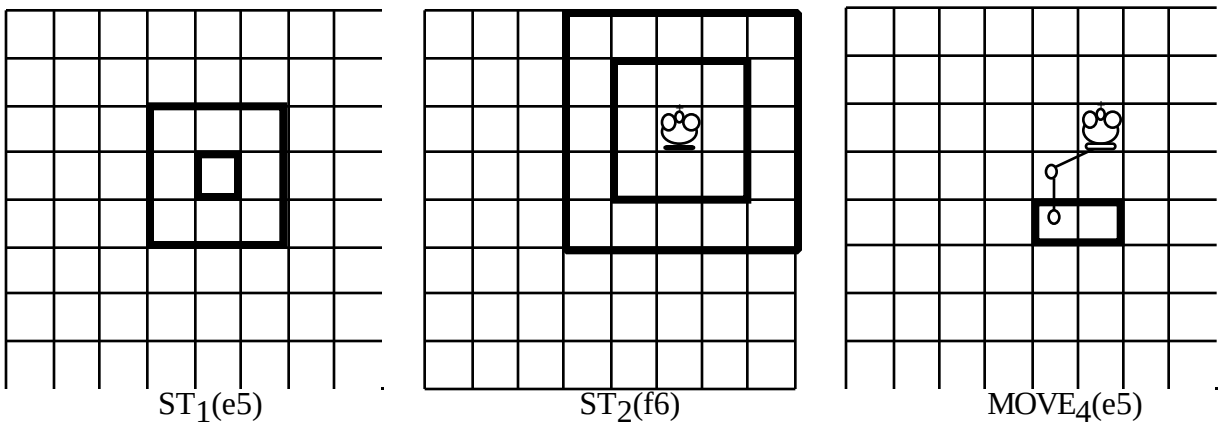

Figure 8. Trajectory generation: the second step

The set of SUM is the same on all steps of the derivation. Hence, $MOVE_4$(e5) is the intersection of the sets shown in figure 8 and SUM, $MOVE_4$(e5)= {e4, f4}; and $next_1$(e5,4) = e4; $next_2$(e5, 4) = f4. This is illustrated in figure 8 (right diagram).

Thus, the number of different values of the function *next* is equal to 2 (r=2), so the number of continuations of derivation should be multiplied by 2. Let us proceed with the first one:

$$\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{A}(e4, h1, 3)\ 2_1 \Rightarrow \ldots$$

This way eventually we will derive one of the shortest trajectories for the King from f6 to h1:

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

Similar generating techniques are used to generate higher level subsystems, the networks of paths.

CORRECTNESS OF GRAMMAR

Employing the Linguistic Geometry tools we can prove the correctness of procedures used for generation and update of the hierarchy of subsystems. The theorem considered below verifies correctness of the grammar of shortest trajectories.

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

$$MAP_{x_o,p}(y_o)=l_o, \qquad (5)$$

where $l_o<2n$, n *is the number of points in* X. *If the relation* R_p *is symmetric, i.e., for all* x *from* X, y *from* X *and* p *from* P $R_p(x, y)=R_p(y, x)$, *then all the shortest trajectories* $t_p(x_o, y_o, l_o)$ *can be generated by the grammar* $\mathbf{G_t^{(1)}}$ (figures 4,5).

Proof. We assume that t_o from $t_p(x_o, y_o, l_o)$ exists and is shortest. We shall prove (5). The proof is carried out by induction with respect to l_o. In the case of $l_o=1$ the statement is easily verified. We assume that for $l_o<m$ the statement is true.

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

Conversely, let (4) *be true*. Let's show that *there exists a trajectory belonging to* $t_p(x_o, y_o, l_o)$, and that *it is the shortest trajectory*.

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

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

Let us construct the shortest trajectory belonging to $t_p(x_o, y_o, m)$. By definition of function MAP there exists x_{m-1} from

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

To complete the proof of the theorem it remains for us to show that *all trajectories* $t_p(x_o, y_o, l_o)$ *are generated by the grammar* $\mathbf{G_t^{(1)}}$ from figure 4, if R_p is symmetric. This grammar, belongs to the class of controlled grammars. Note that the set of functional symbols *Fvar* in it is a set of four zero-arity functions p, x_o, y_o, l_o, i.e., $\mathbf{G_t^{(1)}}$ = G(p, x_o, y_o, l_o). It is obvious that each of the strings generated by $\mathbf{G_t^{(1)}}$ is a trajectory from $t_p(x_o, y_o, l_o)$. Indeed, for each string $a(x_o)a(x_1)\ldots a(y_o)$ thus generated, the elements x_i belong to $ST_i(x_o)=M^i_{x_o,p}$ (see figure 5), consequently, this string is the shortest trajectory.

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

This means that for any shortest trajectory $a(x_o)a(x_1)\ldots a(y_o)$ from $t_p(x_o,y_o,l_o)$ x_i belongs to the intersection of $M^{i}_{x_o,p}$ and $M^{l_o-i}_{y_o,p}$, i.e., $MAP_{x_o,p}(x_i)$=i and $MAP_{y_o,p}(x_i)=l_o$-i,and, consequently,

$$MAP_{x_o,p}(x_i)+MAP_{y_o,p}(x_i)=l_o. \qquad (6)$$

Conversely, if for a certain x from X (6) takes place, then x necessarily enters into the set $\boldsymbol{P}(t_i)$ parametric values of at least one shortest trajectory t_i from $t_p(x_o, y_o, l_o)$. This follows from the fact that $MAP_{x_o,p}(x)\geq 0$ and $MAP_{y_o,p}(x)\geq 0$, while their sum is equal to l_o. That is to say, there exists j ($0\leq j\leq l_o$), such that $MAP_{x_o,p}(x)$=j, $MAP_{y_o,p}(x)=l_o$-j. Then there exist two shortest trajectories t^1 from $t_p(x_o, x, j)$ and t^2 from $t_p(y_o, x, l_o\text{-}j)$. The trajectory t^3 from $t_p(x, y_o, l_o\text{-}j)$ constructed of the same symbols as t_2, but in the reversed order, will also be the shortest trajectory. The concatenation of t^1 and t^2 gives the sought shortest trajectory containing x.

Thus, any element of the set X enters into the set of parametric values $\bigcup_{t^i} P(t^i)$ for all the shortest trajectories t_i from $t_p(x_o, y_o, l_o)$ if and only if (6) is true. These arguments lay a basis for the algorithm for computing the function $next_i(x, l)$ (figure 5).

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

We have: $MOVE(x_{m-1})$ is an intersection of $ST_1(x_{m-1})$, $ST_m(x_o)$ and SUM. Since $t_p(x_o, y_o, l_o)$ are the shortest trajectories, x_m belongs to $ST_m(x_o)=M^m_{x_o,p}$. But x_m also belongs to SUM, because of (6), and x_m belongs to $ST_1(x_{m-1})=M^1_{x_{m-1},p}$ since $R_p(x_{m-1}, x_m)$=T by definition of trajectory. Thus, x_m belongs to $MOVE(x_{m-1})$, i.e., the (m+1)st symbol is generated by the grammar $\mathbf{G_t^{(1)}}$.

The theorem is proved.

LANGUAGES OF TRAJECTORY NETWORKS

After defining the Language of Trajectories, we have new tools for the breakdown of our System into subsystems. According to the ideas presented in (Botvinnik, 1975, 1984), 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 figure 9.

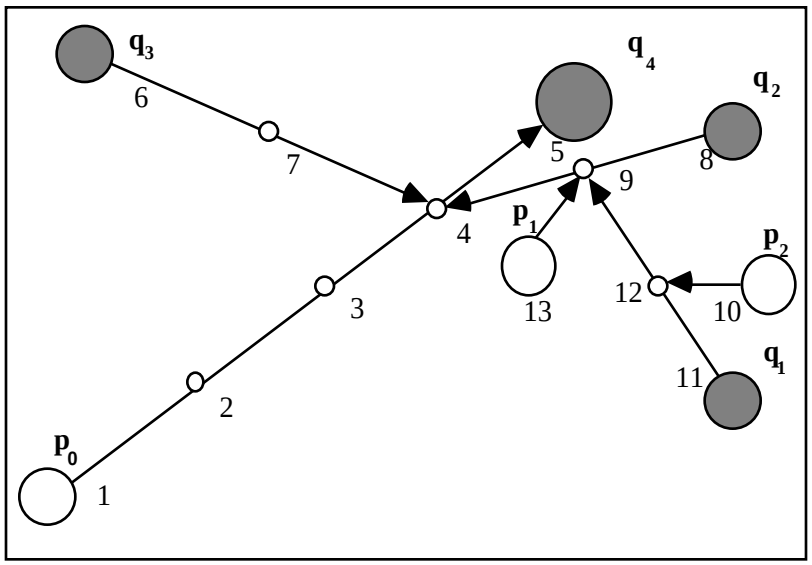

Figure 9. Network language interpretation

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 figure 9) 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 ***a***(11)***a***(12)***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 syntactic representation of such networks. The Language of Trajectories describes "one-dimensional" objects by joining symbols into a string employing reachability relation $R_p(x, y)$. To describe networks, i.e., "multi-dimensional" objects made up of trajectories, we use the relation of *trajectory connection.*

A ***trajectory connection*** of the trajectories t_1 and t_2 is the relation $C(t_1,t_2)$. It holds, if the ending link of the trajectory t_1 coincides with an intermediate link of the trajectory t_2; more precisely t_1 is connected with t_2, if among the parameter values $P(t_2)=\{y,y_1,\ldots,y_l\}$ of trajectory t_2 there is a value $y_i = x_k$, where $t_1=$***a***(x_o)***a***$(x_1)\ldots$***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 figure 9 the trajectories ***a***(6)***a***(7)***a***(4) and ***a***(8)***a***(9)***a***(4) are connected with the main trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5) through point 4. Trajectories ***a***(13)***a***(9) and ***a***(11)***a***(12)***a***(9) are connected with ***a***(8)***a***(9)***a***(4).

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.

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 ***a***(11)***a***(12)***a***(9) in figure 9 is connected (degree 2) with trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5), i.e., C^2(***a***(11)***a***(12)***a***(9), ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5)) holds.

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 ***a***(10)***a***(12) in figure 9 is in transitive closure to the trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5) because C^3(***a***(10)***a***(12), ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5)) holds by means of the chain of trajectories ***a***(11)***a***(12)***a***(9) and ***a***(8)***a***(9)***a***(4).

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

Obviously, the trajectories in figure 9 form a trajectory network relative to the main trajectory ***a***(1)***a***(2)***a***(3)***a***(4)***a***(5). We are now ready to define network languages.

A ***family of trajectory network languages*** $\mathbf{L_C(S)}$ in a state S of the Complex System is the family of languages that contains strings of the form

$$t(t_1, \textit{param})t(t_2, \textit{param})\ldots t(t_m, \textit{param}),$$

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

Different members of this family correspond to different types of trajectory network languages, which describe particular subsystems for solving search problems. One of such languages is the language that describes specific networks called Zones. They play the main role in the model considered here (Botvinnik, 1984), (Stilman, 1977, 1993c, 1993d). A formal definition of this language is essentially constructive and requires showing explicitly a method for generating this language, i.e., a certain formal grammar. The definition of this grammar is beyond the scale of this paper (see Stilman, 1993c). Below 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.

Consider Zone corresponding to the trajectory network shown in figure 9:

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

Assume that the goal of the white side is to remove target q_4, while the goal of the black side is to protect it. According to these goals element p_o starts the motion to the target, while blacks start in its turn to move their elements q_2 or q_3 to intercept element p_o. Actually, only those black trajectories are to be included into the Zone where the motion of the element makes sense, i. e., the

length of the trajectory is less than the amount of time (third parameter τ*) allocated to it.* For example, the motion along the trajectories ***a***(6)***a***(7)***a***(4) and ***a***(8)***a***(9)***a***(4) makes sense, because they are of length 2 and time allocated equals 3: each of the elements has 3 time intervals to reach point 4 to intercept element p_0 assuming one would go along the main trajectory without move omission. According to definition of Zone the trajectories of white elements (except p_0) could only be of the length 1, e.g., ***a***(13)***a***(9) or ***a***(10)***a***(12). As far as element p_1 can intercept motion of the element q_2 at the point 9, blacks include into the Zone the trajectory ***a***(11)***a***(12)***a***(9) of the element q_1, which has enough time for motion to prevent this interception. The total amount of time allocated to the whole bunch of black trajectories connected (directly or indirectly) with the given point of main trajectory is determined by the number of that point. For example, for the point 4 it equals 3 time intervals.

Formalization of the Trajectory Network Languages allowed us to state formally and approach a solution of the problem of efficient adjustment of the hierarchy of languages when system moves from one state to another (Stilman, 1993d). For the chess model this problem is as follows. What Zones and trajectories should be changed (entirely or in part) and what – should not when model moves from one position to another during the search? This problem is relative to the notorious Frame Problem in Artificial Intelligence (McCarthy, Hayes, 1969).

ZONES IN THE 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 figure 10.

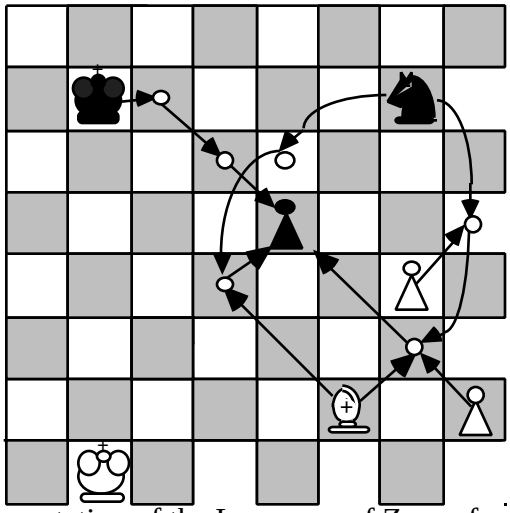

Figure 10. Interpretation of the Language of Zones for chess model.

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 figure 10 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 first 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 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)).

LANGUAGES OF SEARCHES

To describe the search for the optimal operational variant of the System, we define a family of languages of searches. Each search, the string of this language, represents a search tree of variants created to find a solution of some search problem. Assume there is one-to-one correspondence between searches and search trees, neglecting possible repetitions of states.

*A **Family of Languages of Searches*** is the following five-tuple:

$(\pi(i_1)\pi(i_2)\ldots\pi(i_m),$ *Son*, *Brother*, *Father*, *other functions*),

where $\pi(i_k)$ denote branches of some tree and for each i_n:

Son(i_n)=0 if $\pi(i_n)$ represents a leaf branch, *Son*(i_n)= i_{n+1} in the other cases; it means that $\pi(i_{n+1})$ is the left-most child branch for $\pi(i_n)$;

Father(i_n)=0 if $\pi(i_n)$ is a root branch (without parent), for the remaining n *Father*(i_n) = i_r, where $\pi(i_r)$ is parent branch for $\pi(i_n)$;

Brother(i_n)=0 if the branch π(Father(i_n)) has the only child branch $\pi(i_n)$, Brother(i_n) = $\pi(i_r)$, where $\pi(i_r)$ is the next right child branch of the same parent π(*Father*(i_n)).

The list of certain *other functions* singles out a specific language of searches, a member of this family (or sub-family of languages).

The tree shown in figure 11 corresponds to the following search:

$(\pi(1)\pi(2)\pi(3)\pi(4)\pi(5)\pi(6)\pi(7)\pi(8)\pi(9)$, *Son*, *Father*, *Brother*)

Son(1)=2, *Father*(1)=0, *Brother*(1)=3;

Son(2)=0, *Father*(2)=1, *Brother*(2)=0,

and so on.

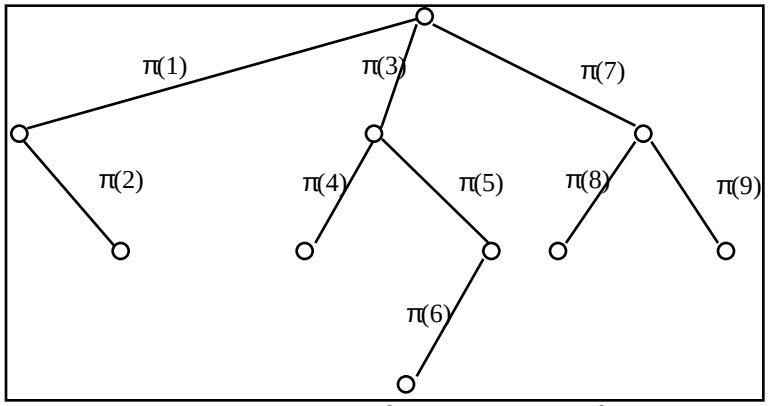

Figure 11. Interpretation of the Language of Searches

Assume that we have a Complex System, searching a space of states.

A ***Family of Languages of Transitions*** is the sub-family of Languages of Searches of the following form:

$(\pi(i_1)\pi(i_2)\ldots\pi(i_m),$ *Son, Brother, Father,* TRANSITIONS, *other functions*),

where TRANSITIONS denotes the set of operators T_j=TRANSITION(p_j,x_j,y_j) of the Complex System, and each T_j corresponds one-to-one to the $\pi(i_j)$.

Thus, the states of the Complex System correspond to the nodes of search trees. Varying the list of *other functions* we can define members of this family which represent various well-known

methods of search. First, it is important to single out a subfamily of languages of reduced searches whose members represent different algorithms of selective search. Among the members of this subfamily there are languages corresponding to *brute force search down to some depth, depth-first search with forward pruning, alpha-beta search, dynamic programming* and others. For all these languages it is necessary to write generating grammars.

As a member of the same family a Language of Translations is defined. It is the highest-level language of the Hierarchy of Languages considered here. This language contains searches generated by the following search procedure (Botvinnik, 1984). We describe it informally.

The search procedure operates by moving the elements of the Complex System along the trajectories of Zones. The Language of Zones is adjusted by translations. The current variant of the search lasts until there still exists possibility to withdraw at least one target element of Zones. However, the variant can not be continued if current losses in this variant, i.e., the value of m(S) from (1), is greater than the expected gain, the sum of values $v(p_i)$ of targets of Zones. This is a depth-first search procedure. Thus transition-ordering constraints are established. If one transition is judged to be superior to its siblings it should be searched first. The transitions with low scores should be pruned. These scores are based on the notion of "vulnerability" of trajectories and Zones, i.e. on a forecast of achieving local goals along trajectories and in entire Zones. Pruning conditions depend on the same forecast and the results of the "inspection procedure" of visiting the subtree already generated, to discover the impossibility of improvement their minimax value (by considering new searches in combinations of Zones). All those procedures are built into the grammar generating the Language of Translations. Next we demonstrate the generation in the Language of Translation on example of the Reti endgame.

RETI ENDGAME SEARCH

The R.Reti endgame is shown in figure 12. The search tree shown in figure 13 was generated by program PIONEER in 1977 and presented at the World Computer Chess Championship (joint event with IFIP Congress 77, Toronto, Canada). Later it was published in different journals and books, in particular, in (Botvinnik, 1984). We consider this tree as a string of the Language of Translations and comment on its generation.

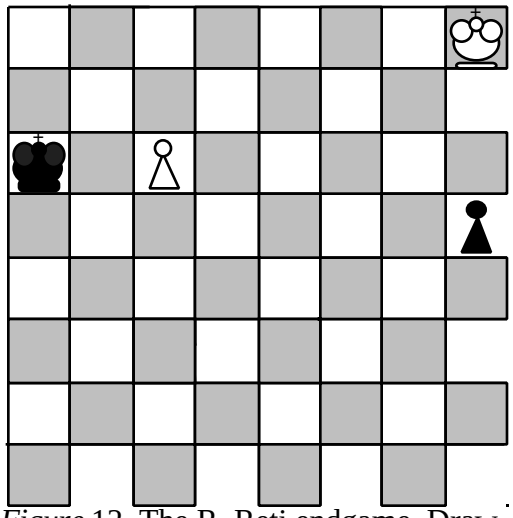

Figure 12. The R. Reti endgame. Draw

First, the Language of Zones in the start position is generated. The targets for attack are determined within the limit of five steps. It means that the horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. All the Zones generated in the start position are shown in figures 14: Zones for Kings as attacking pieces are shown in the left diagram, while Zones for Pawns – in the right one. For example, one of the

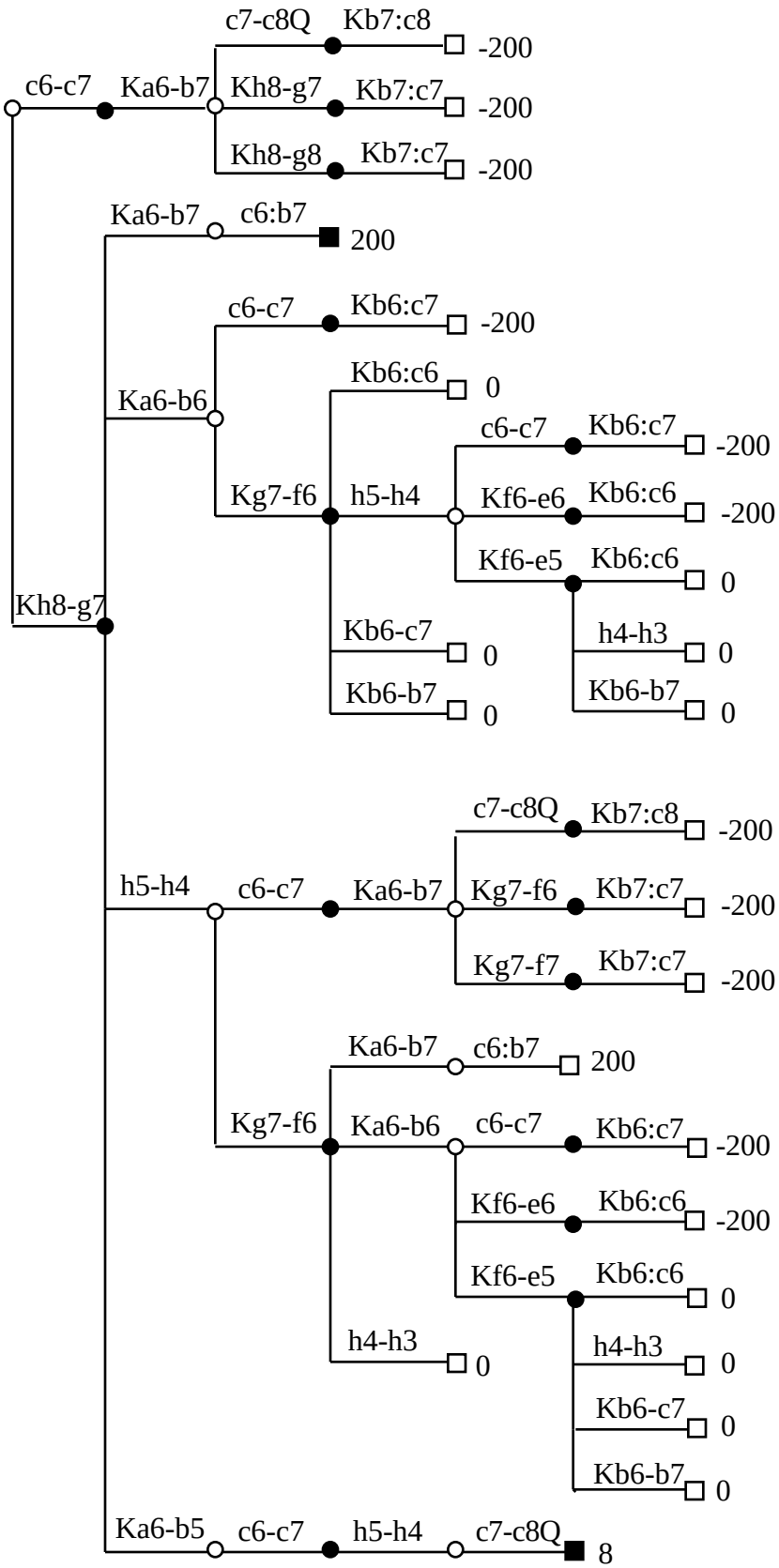

Figure 13. Search tree for the R.Reti endgame

Zones for pawn promotion Z_{WP} is as follows:
Z_{WP}=***t***(P, ***a***(c6)***a***(c7)***a***(c8), 2)***t***(K, ***a***(a6)***a***(b7)***a***(c8), 3)***t***(K, ***a***(a6)***a***(b7)***a***(c7), 2)***t***(P, ***a***(c6)***a***(b7), 1)
The second trajectory of the King ***a***(a6)***a***(b6)***a***(c7) leading to the square c7 is included into different Zone; for each Zone only one trajectory from each bundle of trajectories is taken.

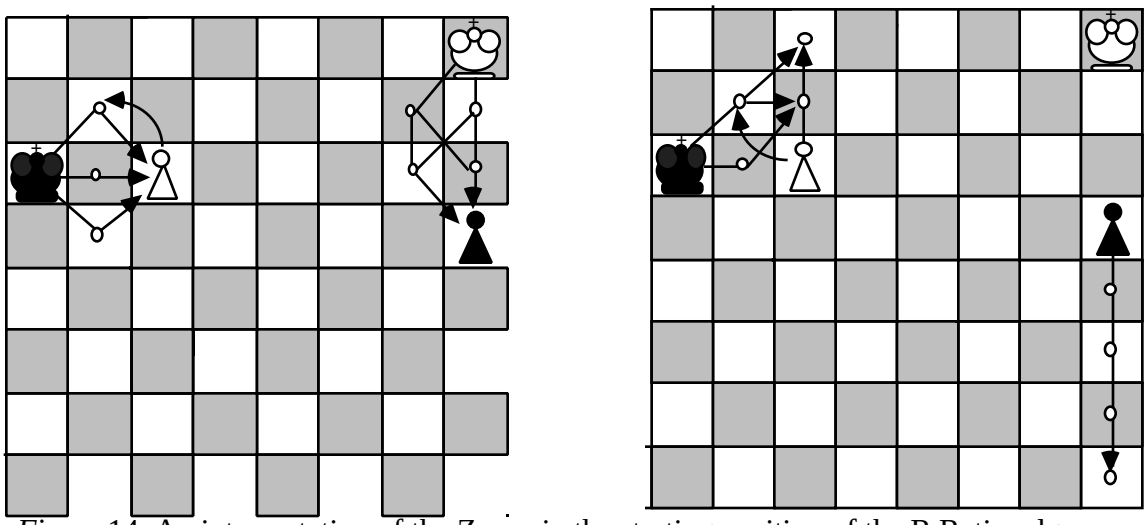

Figure 14. An interpretation of the Zones in the starting position of the R.Reti endgame

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

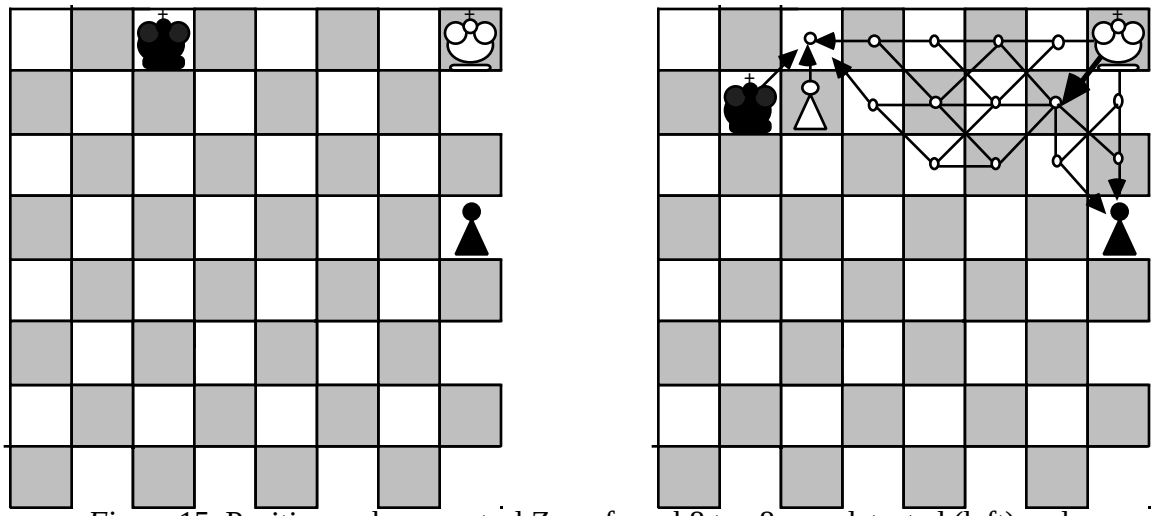

Figure 15. Positions where control Zone from h8 to c8 was detected (left) and where it was included into the search (right)

Next move, 1. ... Ka6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8Q Kb7:c8 (figure 15, left). Here the grammar cuts this branch with the value of -200 (as a win of the black side). This value is given by the special procedure of "generalized square rules" built into the grammar.

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search.

After climb up to the move 1. ... Ka6-b7, the tree to be analyzed consists of one branch (of two plies): 2. c7-c8Q Kb7:c8. The inspection procedure determined that the current minimax value (-200) can be improved by the improvement of the exchange on c8 (in favor of white side). This can be achieved by participation of King from h8, i.e., by generation and inclusion of the new so-called "control" Zone with the main trajectory from h8 to c8. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in figure 15 (right). The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking Pawn on h5. The motion along such trajectories allows to "gain the time", i.e., to approach two goals simultaneously.

The generation continues: 2. Kh8-g7 Kb7:c7. Again, the procedure of "square rules" cuts the branch, evaluates it as a win of the black side, and the grammar initiates the climb. Move 2. Kh8-g7 is changed for 2. Kh8-g8. Analogously to the previous case, the inspection procedure determined that the current minimax value (-200) can be improved by the improvement of the exchange on c7. Again, this can be achieved by the inclusion of Zone from h8 to c7. Of course the best "time-gaining" move in this Zone is 2. Kh8-g7, but it was already included (as move in the Zone from h8 to c8). The only untested move in the Zone from h8 to c7 is 2. Kh8-g8. Obviously the grammar does not have knowledge that trajectories to c8 and c7 are "almost" the same.

After the next cut and climb, the inspection procedure does not find new Zones to improve the current minimax value, and the climb continues up to the start position. The analysis of the subtree shows that inclusion of Zone from h8 to c8 in the start position can be useful: the minimax value can be improved. Similarly, the most promising "time-gaining" move is 1. Kh8-g7. The black side responded 1. ... Ka6-b7 along the first negation trajectories Ka6-b6-c7 and Ka6-b6-c8 (figure 14 (right)). Obviously, 2. c6:b7, and the branch is cut. The grammar initiates the climb and move 1. ... Ka6-b7 is changed for 1. ... Ka6-b6 along the trajectory Ka6-b6-c7. Note, that grammar "knows" that in this position trajectory Ka6-b6-c7 is active, i.e., the King has enough time for interception. The following moves are in the same Zone of Pawn promotion: 2. c6-c7 Kb6:c7. This position is shown in figure 16 (left). The "square rule procedure" cuts this branch and evaluates it as a win of the black side.

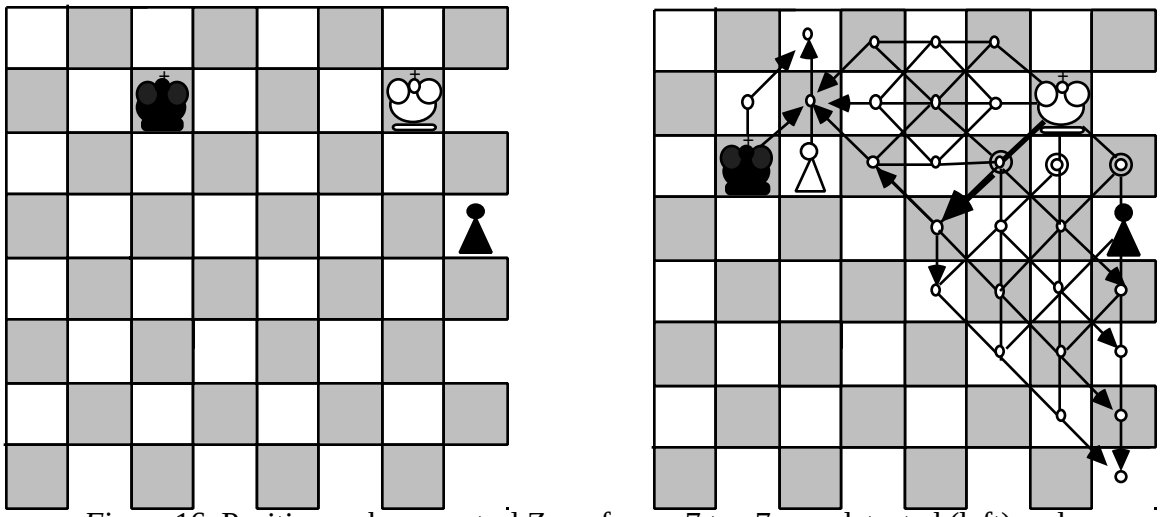

Figure 16. Positions where control Zone from g7 to c7 was detected (left) and where it was included into the search (right).

New climb up to the move 2. ... Ka6-b6 and execution of the inspection procedure result in the inclusion of the new control Zone from g7 to c7 in order to improve the exchange on c7. The set of Zones with different main trajectories from g7 to c7 is shown in figure 16 (right). Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same figure. These are "potential" first negation trajectories. It means that beginning with the second symbol ***a***(f6), ***a***(g6) or ***a***(h6) these trajectories become first negation trajectories in the Zone of promotion of the Pawn

h5. Speaking informally, from squares f6, g6, and h6 the King can intercept the Pawn (in case of white move). The move-ordering procedure picks the subset of Zones with the main trajectories passing f6. These trajectories partly coincide with the potential first negation trajectories. The motion along such trajectories allows to “gain the time”, i.e., to approach two goals simultaneously. Thus, 2. Kg7-f6.

CONCLUSION

The approach to understanding of the game of chess considered here encompasses the discovery of geometrical properties of subsystems and details of interactions between the elements within subsystems, and between different subsystems. We can understand the details of influence of this complex hierarchical structure on the dramatic reduction of the search. Most importantly, the following development of this approach should allow a better understanding of the solution quality.

This contribution to the development of formal language tools for the representation and analysis of human search heuristics should allow for the expansion of advanced human heuristic methods discovered in different complex systems to other real-world systems where existing methods are not sufficient.

References

Botvinnik, M.M. (1970) *Chess, Computers and Long-Range Planning*, Springer-Verlag, New York.

——(1975) *On the Cybernetic Goal of Games*, Soviet Radio, Moscow, (in Russian).

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

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

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

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

Fikes, R.E. & N.J. Nilsson, N.J. (1971) STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, *Artificial Intelligence* 2, 189–208.

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

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

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

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

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

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

Narasimhan, R.N. (1966) Syntax–Directed Interpretation of Classes of Pictures, *Communications of the ACM* 9, 166–173.

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

Pavlidis, T. (1972) Linear and Context-Free Graph Grammars, *Journal of the ACM* 19, 11-22.

Pfaltz, J.L. & A. Rosenfeld, A. (1969) WEB Grammars, *Proceedings of the 1-st International Joint Conference on Artificial Intelligence*, Washington, D.C., 609–619.

Rozenkrantz, D.J.,(1969) Programmed Grammars and Classes of Formal Languages, *Journal of the ACM* 16(1), 107–131.

Sacerdoti, E.D. (1975) The Nonlinear Nature of Plans, *Proceedings of the International Joint Conference on Artificial Intelligence*.

Shaw, A.C. (1969) A Formal Picture Description Scheme as a Basis for Picture Processing

System, *Information and Control* 19, 9-52.
Stefik, M. (1981) Planning and meta-planning (MOLGEN: Part 2), *Artificial Intelligence*, 16(2), 141-169.
Stilman, B. (1977) The Computer Learns, in the book: *1976 US Computer Chess Championship*, pp. 83-90, by Levy, D., Computer Science Press, Woodland Hills, CA.
——(1985) Hierarchy of Formal Grammars for Solving Search Problems, in *Artificial Intelligence. Results and Prospects, Proceedings of the International Workshop*, Moscow, 63–72, (in Russian).
——(1992a) A Syntactic Structure for Complex Systems, *Proc. of the Second Golden West International Conference on Intelligent Systems*, Reno, NE, 269-274.
——(1992b) A Geometry of Hierarchical Systems: Generating Techniques, *Proc. of the Ninth Israeli Conference on Artificial Intelligence and Computer Vision*, Tel Aviv, Israel, 95-109.
——(1992c) A Syntactic Approach to Geometric Reasoning about Complex Systems, *Proc. of the Fifth International Symposium on Artificial Intelligence*, Cancun, Mexico, 115-124.
——(1993a) Linguistic Tools for Intelligent Systems, *Proc. of the Seventh Int. Symp. on Methodologies for Intelligent Systems* (*Poster Session*), Trondheim, Norway, 125-139.
——(1993b) A Linguistic Approach to Geometric Reasoning, *Int. J. Computers and Mathematics with Applications*, (to appear).
——(1993c) Network Languages for Complex Systems, *Int. J. Computers and Mathematics with Applications*, (to appear).
——(1993d) Translations of Network Languages, *Int. J. Computers and Mathematics with Applications*, (to appear).
——(1993e) A Syntactic Hierarchy for Robotic Systems, *Integrated Computer-Aided Engineering*, (to appear).
Volchenkov, N.G. (1979) The Interpreter of Context-Free Controlled Parameter Programmed Grammars, in L.T. Kuzin, ed., *Cybernetics Problems. Intellectual Data Banks* ,The USSR Academy of Sciences, Moscow, 147–157 (in Russian).