Int. J. Mach. Learn. & Cyber. (2014) 5:517–541
DOI 10.1007/s13042-012-0146-0

ORIGINAL ARTICLE

Discovering the discovery of the hierarchy of formal languages

Boris Stilman

Received: 13 September 2012 / Accepted: 18 December 2012 / Published online: 10 February 2013

Abstract The hierarchy of formal languages is a mathematical representation of linguistic geometry (LG). LG is a type of game theory for a class of extensive discrete games called abstract board games (ABG), scalable to the level of real life defense systems. LG is a formal model of human reasoning about armed conflict, a mental reality "hard-wired" in the human brain. LG, an evolutionary product of millions of years of human warfare, must be a component of the primary language of the human brain (as introduced by Von Neumann). Experiences of development of LG must be instructive for solving another major puzzle, discovering the algorithm of discovery, yet another ancient component of the primary language. This paper reports results on discovering mental processes involved in the development of the hierarchy of formal languages. Those mental processes manifesting execution of the algorithm of discovery are called visual streams. This paper reveals the visual streams that were involved in the thought experiments led to the development of the formal theory of LG. Specifically, it demonstrates the streams involved in choosing the formal-linguistic representation of LG; the type of formal languages and grammars, the so-called controlled grammars; the construction of the grammars of shortest trajectories and the grammar of zones. This paper introduces a hypothesis of how we construct and focus visual streams.

B. Stilman (✉)
STILMAN Advanced Strategies, Denver, CO, USA
e-mail: boris@stilman-strategies.com

B. Stilman
University of Colorado Denver, Denver, CO, USA

Keywords Linguistic geometry · Primary language · Artificial intelligence · Algorithm of discovery · Game theory · Formal language · Formal grammar · Visual stream

1 Introduction

A highly intriguing and difficult issue is an algorithm of discovery, i.e., an algorithm of inventing new algorithms and new models. I suggested [36–38] that all the human discoveries from mastering fire more than a million years ago to revealing the structure of DNA and constructing a nuclear reactor in the twentieth century utilized this algorithm. The algorithm of discovery should be a major ancient item "recorded" in the primary language of the human brain (suggested by von Neumann [47]) due to its key role in the development of humanity. This is a line of research that I have been pursuing in [36–38]. It involves investigating past discoveries and experiences of construction of various new algorithms, especially, those I have been involved in. In my opinion, this research would allow me to make a step towards understanding this major puzzle of making discoveries. I investigated several processes that I was involved in while developing important algorithms and representations at different times in the 1980s and 1990s [37, 38]. I also compared my investigations with published introspections of several great scientists [7–13, 20, 45, 47, 48]. It appears that all of them were based on the interplay of the so-called visual streams. These are movie-like visual mental processes. Sometimes, they reflect past reality. More frequently, they reflect artificial mentally constructed reality. This reality includes various artificial worlds with artificial laws of nature including space and time. For example, this could be laws of real or totally different physics, geometry, or game. This

world is populated with realistic and/or artificial animated entities. Some of these entities are mentally constructed (without a real life prototype) while others are images of the real entities. When we run a visual stream we simply mentally observe life events, realistic or artificial, in one of such worlds. Of course, those animated events happen according to the laws of this world and we are "lucky" to be present and "see" what is happening. Usually, scientists have the power to alter this mental world by reconstructing the laws, the entities, etc., according to the problem statement. Then, the visual stream becomes a movie (or play) showing in the end a solution to the problem staged in this world. It is interesting that the theory of LG, itself, the main object of my investigation, is also structured as an animated play and represented naturally in visual streams. These artificial worlds and streams could be modeled employing algorithms, and, consequently, implemented in software.

Linguistic geometry (LG) [34] is a game-theoretic approach that has demonstrated a significant increase in size of problems solvable in real time (or near real time). This paper continues a series of papers [35–37, 43, 44] intended to investigate role of LG in human culture. In here, I will make the next step in utilizing vast experiences of developing LG for discovering the algorithm of discovery. I will focus on the processes that led to the developing formal linguistic representation of LG, the hierarchy of formal languages. I will show that they involved the interplay of the mental visual streams. Some of those streams were wrapped into the formal mathematics and computer programs. Note that multiple figures included in this paper are not just a set of illustrations to formal derivations. It is actually a set of snapshots of the visual streams, the driving engine behind those derivations.

The word *Linguistic* refers to the model of strategies formalized as a hierarchy of formal languages. These languages and their translations are the constructs that permit us describe strategies leading the game from state to state. These languages utilize a powerful class of generating grammars, the controlled grammars ([34] and Section IX), which employ formal semantics of the game to control generation of a string of symbols using mutual influence of the substring generated so far and the grammar's environment.

The word *Geometry* refers to the geometry of the game state space, the **SPACE** ([34] and Section VI), which is a set of all the states resulting from all legal plays (variants) leading from the start state. Every state could be represented as an abstract board X with abstract pieces P, i.e., mobile entities, located on this board and acting upon each other. Thus, different states include the same board with different configurations of pieces resulting from the sequence of moves leading to this state.

In LG, the geometry of the **SPACE** is effectively reduced to the geometry of the board. Thus, the state space is reduced to the "projection of the board $\times$ time" over the "board", by introducing images of planning skirmishes of the pieces reflecting planning physical movements and actions of those pieces over the board (Fig. 2). These images are called zones and trajectories. LG permits constructing highly efficient strategies for the game via routing pieces along those images. Discovering and running the algorithm that guided the development of a mathematical representation of the routing images is the topic of this paper.

LG is a viable approach for solving board games such as the game of chess as well as practical problems such as mission planning and battle management. Historically, LG was developed, beginning from 1972, by generalizing experiences of the most advanced chess players including World Chess Champions and grandmasters [1, 34]. In the 1970s and 1980s this generalization resulted in the development of the computer chess program PIONEER utilized successfully for solving chess endgames and complex chess positions with a number of variants considered in the order of 10^2 while the state spaces of those problems varied from 10^{10} to 15^{25}. The variants constructed by PIONEER were very close to those considered by the advanced chess experts when analyzing the same problems. Further generalization led to development of the new type of game theory, LG, changing the paradigm for solving game problems "from search to construction" [34]. The algorithm of LG was represented in the form of the hierarchy of formal languages with powerful formal semantics. Beginning from 1999, the LG-based technology was applied to more than 30 real life defense-related projects [17]. On multiple experiments, LG successfully demonstrated the ability to solve extremely complex modern military scenarios in real time. The efficacy and sophistication of the courses of action developed by the LG tools exceeded consistently those developed by the commanders and staff members [41, 42].

Forty years of development of LG including numerous successful applications to board games and, most importantly, to a highly diverse set of modern military operations [15–17, 39–44], from cruise missiles to military operations in urban terrain to ballistic missile defense to naval engagements, led us to believe that LG is something more fundamental than simply yet another mathematical model of efficient wargaming.

In 1958, von Neumann suggested that the "external" language (including multiplicity of natural languages as well as mathematics and computer science) that we use in communicating with each other may be quite different from the "internal" language used for computation by the human brain [47]. He argued that we are still totally

unaware of the nature of the primary language for mental calculation. More than 50 years passed since von Neumann hypothesized existence of the primary language. Unfortunately, the nature of this language is still unknown.

A universal applicability of LG in a variety of military domains, especially, in the domain of the ancient warfare [43, 44], its total independence of weaponry, nationality or country, its power in generating human-like strategies suggest that the algorithm of LG utilized by the human brain is "hard-wired" in the primary language [47]. The age of the primary language must be much greater than the age of human natural languages, and so the age of LG [35]. This great age suggests that the primary language as well the language of LG are not symbolic because symbolic languages have been in existence on Earth no more than 10,000 years. The core human skills including the war-fighting skills have been developed as hybrid and, mostly, analog algorithms. During several million years of evolution with constant fight for survival of the human species, the human brain perfected those algorithms as well as language(s) utilized by these algorithms. Eventually, this constant perfection has led to the development of multiple symbolic equivalents. In [35, 43, 44], we concluded that every human brain "speaks" the LG language, though, only well trained commanders and, especially, advanced strategists are able to utilize it to full capacity. Most importantly, they are able to translate from the LG language, i.e., from the primary language, into the secondary (natural) languages to describe strategies in the spoken language terms.

The algorithm of discovery should also be a major ancient item "recorded" in the primary language. By investigating past discoveries, experiences of construction of various new algorithms, and the heritage of LG, we will make a step towards understanding of this puzzle.

In [35, 43, 44], it was suggested that the game of chess served as a means for discovering LG. The first version of the theory of LG was developed by generalizing algorithms implemented in the computer chess program PIONEER [1, 34]. The mathematical framework of LG, the hierarchy of formal languages, constructed in the beginning of the 1980s, was based exclusively on the chess domain, the only application of LG available at that time. While generalizations of LG continued in the following years [27–34], the structure of this framework has never changed.

The key issues of LG were developed in the course of various thought experiments. Some of them were discussed in [36–38]. Most of the constructs of LG were initially conceived and tested in those experiments. The constructs that successfully passed experiments (and some alternative constructions) were programmed and tested employing software applications [2, 34]. Recent investigation [37, 38] permitted to distinguish the core component and the building block of those thought experiments, the mental visual stream. Thought experiments discussed in [37] included two major experiments, the first one related to choosing mathematical tools for construction of the hierarchy of formal languages and the second related to development of the No-Search Approach. The No-Search Approach started as the 2D/4A Forward Pruning experiment with the original goal to evaluate accuracy of the solutions generated by the LG algorithm [38]. The No-Search Approach could be interpreted as an implementation of a macro-model of LG. This macro-model included three thought experiments, the 2D/4A Forward Pruning, the Terminal Set Expansion and the Strategy Construction experiments. Each of those experiments involved various types of visual streams dealing with subsets of the game state space, the **SPACE**. The macro-model demonstrated that the LG algorithm generates solutions employing highly efficient decomposition of **SPACE**.

This paper continues the same line of investigation, in order to understand the nature and, eventually, develop an algorithm of discovery whose execution would model the respective streams. I will investigate the visual streams that drove the development of the mathematical representation of the pictorial LG algorithm, the micro-model of LG. Specifically, I will investigate the three thought experiments related to the development of the hierarchy of formal languages. They include the idea of formal-linguistic representation, the inventions of the language of trajectories and the language of zones. This series of thought experiments started from the experiment described in [37], when I first realized that a planning sequence of steps, a future path, could be represented as a string of symbols. The subsequent experiments involved construction of the appropriate formal languages. In the course of those experiments I had to solve several major problems. The first problem was to learn (or develop) a class of languages with powerful means for handling formal semantics of trajectories and zones. Of course, I did not know if it will be the same class of languages (the controlled languages). The second problem was to construct a grammar for generating trajectories. It appears that those problems presented themselves as a bundle. The third problem was to construct a grammar for generating zones. A number of self-imposed constraints made this problem especially difficult. A continuously subconsciously running visual stream that hit an analogy with pictorial images permitted me to realize that the same class of formal languages is fully capable of representing zones (Section VII). While the grammar of zones was the most sophisticated controlled grammar I ever constructed, this construction was achieved consciously employing a construction visual stream with preliminary modeling generation employing a four-screen example without a grammar (Section XVI).

To make this paper self-contained, I included several introductory sections as well as the material related to the dynamic hierarchy visual streams presented originally in [37]. Note that the hierarchy of formal languages is an optional mathematical model, one of several equivalent formal models reflecting the pictorial LG algorithm. It is likely, that this specific model reflects personal traits of the inventor, me.

2 Neurophysiology of thought experiments

Thought experiments allow us, by pure reflection, to draw conclusions about the laws of nature [5, 7, 14, 18, 21]. For example, Galileo before even starting dropping stones from the Tower in Pisa, used pure imaginative reasoning to conclude that two bodies of different masses fall at the same speed. The Albert Einstein's thought experiments inspired his ideas of the special and general relativity [7, 18]. The efficiency and the very possibility of thought experiments show that our mind incorporates animated models of the reality, natural or artificial, e.g., laws of physics, mathematics, human activities, etc. Scientists managed to decode some of the human mental images by visualizing their traces on the cortex [14]. For example, it was shown that when we imagine a shape "in the mind's eye", the activity in the visual areas of the brain sketches the contours of the imagined object, thus, mental images have the analogical nature [5]. It appears that we simulate the laws of nature by physically reflecting the reality in our brain. The human species and even animals would have had difficulty to survive without even minimal "understanding" of the laws of environment. Over the course of evolution and during development of every organism, our nervous system learns to comprehend its environment, i.e., to "literally take it into ourselves" in the form of mental images, which is a small scale reproduction of the laws of nature. Neuropsychologists discovered that "we carry within ourselves a universe of mental objects whose laws imitate those of physics and geometry" [5]. In [35], I suggested that we also carry the laws of the major human relations including the laws of optimal warfighting. The laws of nature and human relations manifest themselves in many different ways. However, the clearest manifestation is in perception and in action. For example, we can say that the sensorimotor system of the human brain "understands kinematics" when it anticipates the trajectories of objects. It is really fascinating that these same "laws continue to be applicable in the absence of any action or perception when we merely imagine a moving object or a trajectory on a map" [5]. This observation, of course, covers actions of all kinds of objects, natural and artificial. Scientists have shown that the time needed to rotate or explore these mental images follows a linear function of the angle or distance traveled as if we really traveled with a constant speed. They concluded that "mental *trajectory* imitates that of a physical object" [5].

3 Discovering by "seeing"

In this paper, I continue developing a hypothesis [37] that there is a universal algorithm of discovery behind those experiments driving development of LG and major advances in other sciences. Based on the experiences of great scientists [7–13, 20, 45], [47, 48], research in cognitive science and neuroscience [3–14, 18, 21], and from my own experience I suggested that the essence of the discoveries is in the "visual streams" [37]. As I mentioned in Section I, we can think about a visual stream as a movie (or play) showing in the end a solution to the problem staged in the artificial world. Usually, this solution comes without a proof because it was not proved but played in this world. When the solution is known, the proof itself could be broken into small subproblems, staged in the altered mathematical world and eventually discovered formally. This seemingly shocking reality does not look mysterious to me. These artificial worlds could be modeled employing algorithms, and, consequently, implemented in software. In my opinion, this is the road to the algorithm of discovery.

A discovery algorithm manifests itself through its execution, a visual stream. It is executed repeatedly, and, usually, many times. For the new execution it can be changed (morphed) to better represent artificial world. Of course, we could notice just the changed visual stream. Better representation is not the main reason for repeating its execution. For the new run the algorithm may not be changed at all. It appears that multiple runs may be required for a scientist to better observe and make note of the events of the artificial world which have been missed during previous runs. In case of difficult problems the number of runs may be huge and, in my opinion, most of these runs happen subconsciously.

The visual streams mentioned above are the universal components of the staged play (mental movie). Based on the thought experiments in LG as well as [7, 14, 18], and other publications, I identified the three components of the discovery algorithm, observation, construction and validation, which manifest themselves as the following visual streams [37]:

- observation streams;
- construction streams;
- validation streams.

I suggest that the discovery, in general, is based on the interplay of those streams. Apparently, each of the types of

streams can take various shapes and sizes. Indeed, a stream is mentally morphing objects and this morphing is limited by a person's imagination, if at all. However, fortunately or, sometimes, unfortunately, we limit imagination and the respective streams with a set of constraints. These constraints may come in the form of the past knowledge or could be generated by another visual stream. The constraints restrict the form of the artificial world, the laws of the world's nature, the world's players and the morphing of the visual stream. It appears that one of the key principles for terminating the stream, besides getting an expected outcome, is the analogy (or commonality) with a seemingly unrelated object. This analogy could be found within several streams or with an external object. Establishing such analogy should reveal the set of constraints for future mental construction.

In [38], I suggested that a visual stream is the animated interface of the algorithm of discovery executed by the human brain. I also suggested that the only way the algorithm of discovery manifests itself is through the mental visual streams. Moreover, in my opinion, this interface is the only way to control and, possibly, even modify this algorithm. To tune the algorithm of discovery to a different object or even to a completely different domain we would have to activate it with a familiar visual stream and morph this stream to this new object or domain. At first, this morphing may not be satisfactory, but after observing the modified stream we may have to continue morphing until the resulting stream satisfies our expectations. While the algorithm is not directly available, the morphing procedure provides very powerful means for control. We could morph the artificial world, the laws of nature, the entities, etc., practically, without any restrictions except for those imposed by ourselves. This morphing looks like a visual meta-stream which turns one visual stream into another. I could only speculate about the causes of this special type of interface of the algorithm of discovery. Probably, it is based on the underlying brain "hardware", the ancient highly problem-oriented brain structures operating certainly in non-symbolic analogic mode. As a result of evolution of human intelligence these structures were retuned for execution of this extremely important algorithm. This kind of the visual stream interface looks very unusual, which makes it even more interesting for investigation. My task is to discover this algorithm based on its manifestations, i.e., on the recorded signs of its executions, while following various past discoveries.

4 Focusing streams

Based on my personal experience with thought experiments in LG [37, 38], the experiments presented in the current paper, and various publications, including introspections of the great scientists, I concluded that the algorithm of discovery does not involve any search or, at least, what is called in Computer Science a tree-based search. It appears when making discoveries we deal with various visual streams (mental plays) representing real or artificial worlds. One of the important questions that remained unanswered in my previous papers on the algorithm of discovery is how we focus those streams, how we give them direction without search. This question could be reformulated as follows. You have created mentally an artificial world somehow related (or, seemingly, not related) to the object you are concerned about. Let us imagine it as an artificial city where you immersed yourself. A lot of stuff is going on in this city around you. In particular, a number of movies (or plays) are being played simultaneously. You may start watching some of them or even changing them via morphing. Moreover you may get involved in one of those plays. How do you make choices, which play to watch, which play to morph (and if so, in which direction), which play to get involved, etc.?

In my opinion, all those decisions are based on analogy. Even the artificial world (the city) you have created is familiar, analogous to the worlds and objects you dealt in the past. Everything what is happening in the city should be familiar to some degree—this is the product of your imagination, which is based on your experience. And you go in the direction of the closest analogy to the object of your concern.

It is likely that the visual stream world will be characteristic of your past experience. This is the place where you try to familiarize yourself. You may increase this familiarity via morphing some parts of the city. At the first glance many events in the city do not look familiar or analogous to your past. That is why you need to get settled over there, i.e., you have to go around, observe and even get involved. Often, this settling process may take a lot of time; it may not finish at all. However, when it is finished, you begin noticing things, which you did not see before. More formally, you begin noticing non-obvious analogies including closer relevancy to the object in hand. After the leap of noticing, the analogies would start directing your visual stream. I am going to explore this hypothesis in the current and future papers.

5 Constructing visual streams

This object of investigation could be an abstract entity, an event, a process or an algorithm. In formal terms, we could discuss just algorithms because they could model all other types. For the discovery process these are different objects because you have to visualize them. Unfortunately, you

cannot do that directly. For example, you can visualize neither an abstract object nor an abstract, e.g., mathematical, world around it. So, you end up with an instantiation of this abstraction and a respective concrete world around this instantiation. No matter how hard you try to generalize visually this imaginary world, it would still be concrete (not abstract).

For example, you cannot visualize an arbitrary finite set or the n-dimensional Euclidian space. However, you can visualize a concrete, rather small, finite set. In an attempt to visualize a general abstract board of an ABG (Section VI), you can switch to a finite set on a 2D plane and consider a 2D shape, a convex hull of this set. This type of visualization leads to the effect of erasing the particulars, when we drop the specific details while gaining the visual clarity. In case of an ABG we would drop the particulars of the various types of reachabilities (like those of the chess pieces) in favor of the global picture of trajectories and zones. This type of visualization is demonstrated in the visual streams for constructing the grammar of shortest trajectories and the grammar of zones (Sections VII, XI, XV, XVI). Another example of nonvisual objects is the type of objects in the micro-world of physics, such as atoms, nuclei, etc. They are not within the scope of the human visual experience. Yet another example comes from the discovery of the structure of DNA. When constructing (mentally and physically) their model of the DNA molecule, Watson and Crick represented atoms as spheres and bonds as straight lines [48].

This way of erasing the particulars leads to the visual model utilized by a visual stream. The initial visual model may be guided by a very specific prototype, where we actually erase the particulars. Two such models are introduced in this paper. One is the model utilized by the construction stream for function $next_i$, Section XI. It includes ellipses, circles, and rings. In this case, the prototype is the visualization of the algorithm for generating all the shortest trajectories for the chess King from 41 to 48 (Fig. 10). Another model is utilized by the construction visual stream sketching zone generation (without a grammar), Section XVI. It includes four "screens", the duplicates of the enumerated abstract board of the ABG (Table 6). For this model erasing the particulars took place at several stages. Firstly, the specific zone shown in Figs. 2 and 13 is just a sketch of the pictorial LG images of a number of actual LG zones, whose trajectories are replaced by the straight lines. Secondly, the generation in Table 6 is sketched without a grammar with a number of steps skipped.

The visual model drives construction of the formal abstract model so that every item in a visual model has a tag in the respective formal model. At some stage, a running visual stream is accompanied by a formal symbolic shell. The visual stream may run the visual model with symbolic shell several times. One of those runs and all the following runs utilize the same visual model but it will be driven by the abstract constraints of the symbolic shell. The final runs of the construction visual streams of the grammars presented in this paper were driven by the constraints of the ABG, the abstract set theory and the controlled grammars.

In the following Sections, I will analyze the visual streams involved in the development of the hierarchy of formal languages. This development included several stages with different artificial worlds and respective streams. Some of those streams reflect the reality of board games constructed and played by people. Other streams reflect the pictorial LG algorithm for solving those games, which, of course, is an artificial world. However, it is easily representable in a series of pictures. Yet another set of visual streams reflect totally artificial abstract world of various 2D shapes. This world also permits rough, "approximate" visualization and animation, which is sufficient for guiding mathematical reasoning. Those different visual streams are represented in this paper with multiple figures. The reader should not be confused by their role. Typically, in mathematical papers including our past "non-discovery" papers such figures are secondary; they are illustrations to the formal derivations and conclusions. In this paper, they are "primary"; they are the snapshots of the visual streams that drove development of those conclusions.

6 Abstract board games

LG is intended for solving Abstract Board Games (ABG) defined as follows (see full definition in [34]):

$\langle\ X,\ P,\ R_p, val,\ \mathbf{SPACE},\ ON,\ S_0,\ S_t,\ TR\ \rangle,$

where

$X = \{x_i\}$	is a finite set of points (or locations of an abstract board);
$P = \{p_i\}$	is a finite set of pieces; $P = P_1 \cup P_2$ called the opposing sides; P_1 and P_2 are disjoint sets;
$R_p(x, y)$	is a set of binary relations of reachability in X ($x \in X$, $y \in X$, and $p \in P$);
val	is a function on P representing the values of pieces;
SPACE	is the state space. A state $S \in$ **SPACE** consists of a partial function of *placement* ON: $P \rightarrow X$ and additional parameters;
$ON(p) = x$	means that piece $p \in P$ occupies location x at state S. Thus, to describe function ON at state S, we write equalities $ON(p) = x$ for all

pieces p, which are present at S. We use the same symbol ON for each such partial function, though the interpretation of ON may be different at different states. Every state S from **SPACE** is described by a list of formulas $\{ON(p_j) = x_k\}$ in the language of the first order predicate calculus, which matches with each relation a certain formula;

S_0 and S_t are the sets of *Start* and *target* states. Thus, each state from S_0 and S_t is described by a certain list of formulas $\{ON(p_j) = x_k\}$. $S_t = S_t^1 \cup S_t^2 \cup S_t^3$, where all three are disjoint. S_t^1, S_t^2 are the subsets of *win* target states for the opposing sides P_1 and P_2, respectively. S_t^3 is the subset of the *draw* target states. States from S_t^1, S_t^2 and S_t^3 could be evaluated (marked) as $+1$, -1 and 0, respectively;

TR is a set of *transitions* (moves), TRANSITION (p, x, y), of the ABG from one state to another (Fig. 1). These transitions are described in terms of the lists of formulas (to be removed from and added to the description of the state) and a list of formulas of applicability of the transition. These three lists for state $S \in$ **SPACE** are as follows:

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

where $p \in P$. The transitions are defined and carried out by means of a number of pieces p from P_1, P_2, or both. This means that each of the lists may include a number of items shown above. Transitions may be of *two types*. A transition of the *first type* (shown above) occurs when piece p moves from x to y without removing an opposing piece. In this case, point y is not occupied by an opposing piece. A transition of the *second type* occurs if piece p moves from x to y and does remove an opposing piece q. Typically, the opposing piece has to occupy y before the move of p has commenced. In the latter case, the **Applicability list** and the **Remove list** include additional formula $ON(q) = y$. For concurrent systems [34, 36, 37], this is not necessary: piece q may arrive at y simultaneously with p and be removed.

The goal of each side is to reach a state from its subset of the win target states, S_t^1 or S_t^2, respectively, or, at least, a draw state from S_t^3. The problem of the optimal operation of ABG is considered as a problem of finding a sequence of transitions leading from a Start State of S_0 to a target state of S_t assuming that each side makes only the best moves (if known), i.e., such moves that could lead ABG to the respective subset of target states. To solve ABG means to find a *strategy* (an algorithm to select moves) for one side, if it exists, that guarantees that the proper subset of target states will be reached assuming that the other side could make arbitrary moves to prevent the first side from reaching the goal.

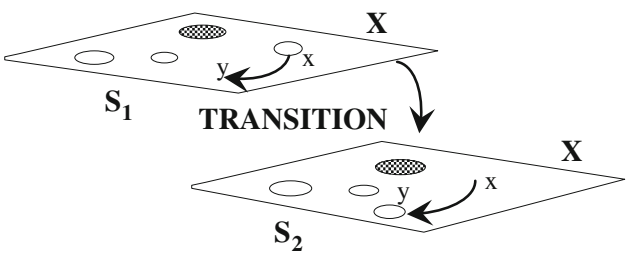


Fig. 1 Interpretation of ABG: TRANSITION(p, x, y)

7 Pictorial LG and hierarchy of languages

In this section I will briefly introduce the reader to the pictorial LG and its formal-linguistic representation. My intent is to provide minimal information required for explaining the construction of the hierarchy of formal languages.

In the pictorial LG, a zone is a trajectory network—the set of interconnected trajectories with one singled out trajectory (called the main trajectory). An example of an attack zone is shown in Fig. 2. Below, I will utilize this image to explain the formal-linguistic representation of LG.

The basic idea behind a zone is as follows. Piece p_0 should move along the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ to reach the ending point 5 and remove the target q_4 (an opposing piece marked with a dark circle). Numbers 1, 2, 3, 4 and 5 represent the identifiers of the respective

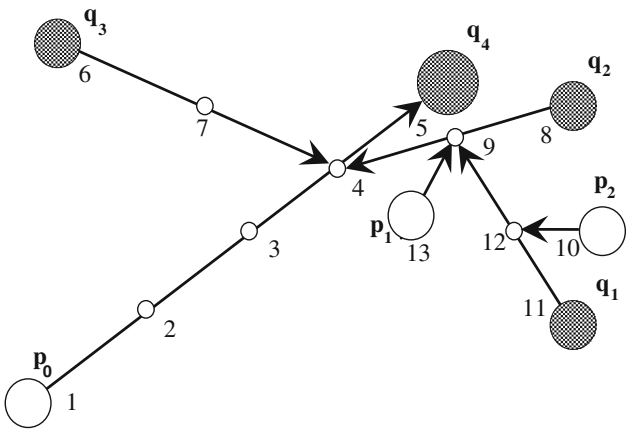


Fig. 2 A pictorial LG image of the attack zone

points or, simply, the coordinates of those points on the abstract board. Naturally, the opposing pieces should try to obstruct these movements by "dominating" the intermediate points of the main trajectory. They should come closer to these points (to point 4 in Fig. 2) and remove piece p_0 after its arrival (at point 4). For this purpose, pieces q_3 or q_2 should move along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of the piece p_0 at point 4. These trajectories are called 1st negation trajectories. Similarly, piece p_1 of the same side as p_0 (i.e., White) might try to obstruct the movement of q_2 by dominating point 9 along the trajectory $\boldsymbol{a}(13)\boldsymbol{a}(9)$. It makes sense for the opposing side to employ the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of piece q_1 to prevent this domination.

Consider a complete formal-linguistic representation of a sample zone depicted in Fig. 2 (Table 1):

Below, in analyzing this example, we describe the movements of pieces in the zone. Note that the zone itself is represented in the formal LG as a string of symbols $\boldsymbol{t}$ with parameters $\boldsymbol{t}(\ldots)\boldsymbol{t}(\ldots)\boldsymbol{t}(\ldots)\boldsymbol{t}(\ldots)$, a static (*not dynamic*) object that represents a network of trajectories. It represents a subsystem of ABG in a certain state. With every change of a state, a new zone should be constructed. Some of these zones are considered successors of the original zone. For example, zones with the main trajectories that start at locations 2, 3 and 4 are the successors of the zone shown in Fig. 2. For simplicity, movements of pieces considered below are guided by the pictorial image of the same zone, and the zone itself is considered without changes and isolated from the rest of the ABG.

Let us reconsider these movements to explain reasoning behind the construction of this zone. Assume that the goal of White (Fig. 2) is to remove target q_4, while the goal of Black is to protect it. Also, assume that Black and White alternate turns. Black starts first to move piece q_2 or q_3 to intercept p_0. White follows in its turn by moving piece p_0 to the target. Only those Black trajectories are to be included in the zone where the movement of the piece makes sense, i.e., the length of the trajectory is less than (or equal) the amount of time allotted to it. For example, the movement 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 allotted equals 4. All the above information is reflected in the values of parameters of the symbols of the string Z, $\boldsymbol{t}(q_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$ and $\boldsymbol{t}(q_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)$, respectively. Indeed, each of the pieces, q_2 and q_3, has 4 time intervals to reach point 4 to intercept piece p_0, assuming that Black starts first and p_0 would go along the main trajectory without delay. By definition of a zone, the trajectories of White pieces (except p_0) could only be of the length 1, e.g., $\boldsymbol{a}(13)\boldsymbol{a}(9)$ or $\boldsymbol{a}(10)\boldsymbol{a}(12)$. Because piece p_1 can intercept piece q_2 at point 9, Black trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of the piece q_1 should be included, i.e., q_1 has enough time for movement to prevent this interception. The total amount of time allotted to the whole bundle of the Black trajectories connected (directly or indirectly) with a given point of the main trajectory is determined by the ordinal number of that point. For example, for point 4, it is equal to 4 time intervals. This number gives us the value of time bound for the 1st negation trajectories connected with the main trajectory at point 4. Automatic distribution of the allotted time between the negation trajectories was one of the major requirements for the developing the Grammar of Zones (Sections XV—XVII).

Table 1 Representation of a sample zone shown in Fig. 2

Z =	$\boldsymbol{t}(p_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(q_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$
	$\boldsymbol{t}(q_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(p_1, \boldsymbol{a}(13)\boldsymbol{a}(9), 3)$
	$\boldsymbol{t}(q_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(p_2, \boldsymbol{a}(10)\boldsymbol{a}(12), 2)$

A set of zones generated in every state of a problem is a unique representation of this state. A piece may be involved in several zones and in several trajectories in the same zone. All the trajectories and zones are evaluated with respect to their quality [34]. Only the highest quality trajectories are considered for generating strategies. The quality function is based on the prediction of the "rate of difficulty" for a piece for moving along the trajectory. For example, for the attack zone (Fig. 2) piece p_0 has to pass four locations 2, 3 and 4 to reach destination and destroy its target at 5. This passage may be free or it may be obstructed by the enemy pieces if they approach these locations. For example, piece p_0 can be captured at location 4 by q_2 or q_3 if they approach location 4. The notion of passage through location for the game of chess is based on the values of pieces (surrounding this location, i.e. one step away) and on the result of optimal exchange of these pieces [34]. For the military operations employing trajectories of physical entities (missiles, planes, single soldiers) and shooting, the notion of passage through location is based on the notion of probability-of-kill, which is defined for all the entity-weapon pairs. These probabilities permit calculating qualities of trajectories and zones based on the integrated probabilities of successful passage. For the operations employing trajectories of pieces representing groups of entities that require close encounter with hand-to-hand fighting (like ancient cavalry or infantry) or massive shooting with low probability-of-kill of each shot (like ancient archers), the notion of passage is based on the notion of attrition rate, a statistical outcome of the skirmish, which is defined for all the pairs of antagonistic groups. These attrition rates permit calculating qualities of trajectories and zones based on the integrated attrition resulting from the passage of these trajectories and zones, respectively. In all cases, the less "difficulties" a piece

would experience in passing along a trajectory the higher quality of this trajectory is. Every location along a trajectory, where a piece can be intercepted (for the game of chess), destroyed with high probability (for modern military operations) or suffer high attrition (for ancient operations) reduces quality of this trajectory. A trajectory, which includes at least one such location, is called a trajectory with closed location or a *closed trajectory*. A trajectory without such locations is called an *open trajectory*. An example of an open main trajectory of an attack zone, trajectory for p_0, is shown in Fig. 2. An example of the closed 1st negation trajectory is a trajectory $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ for q_1.

The quality of trajectories reflects a short-term prediction. A long-term prediction for a zone is reflected in its *status*, Protect or Intercept. In general, computation of a status requires search and minimax involving all the pieces inside the zone following zone's trajectories and scheduling constraints (the allotted time). The outcome depends on a number of factors including the values of pieces (function *val* in Section VI). If the optimal variant in the zone is in favor of the attack side, which means that the main piece p_0 can freely reach its destination and destroy the target, the status of this zone is Protect. Otherwise, the status is Intercept. In many cases, zones are simple and their status can be determined without search. Note that status is a prediction for the *current state*. In a different state, status of a similar zone, i.e., a successor of the original one, may be different.

The grammars constructed in the formal theory of LG generate all kinds of trajectories and zones. For example, zone Z (Fig. 2; Table 1) for a specific ABG could be generated by the grammar of zones with the input of just the current state of the ABG, the coordinates of two locations 1 and 5, and the length of the main trajectory, which is 4 steps, Tables 7 and 8. See also Sections XVI and XVII.

In this explanation, for simplicity, I described only one type of zones called *attack* zones. There are several different types of zones. For example, *domination* zone looks similar to the attack zone (Fig. 2) but it would not have a target at location 5. The purpose of this zone would be for p_0 to reach location 4 and guard location 5 (which might belong to other zones).

8 From pictorial images to strings of symbols

Until the beginning of the 1980s, the algorithm of LG existed as an algorithm of the program PIONEER. (Its origin goes back to the "Botvinnik's Algorithm" [1].) However, the ongoing research on LG was conducted in the form of pictorial images of trajectories and zones (Fig. 2) and highly specific thought experiments by mentally solving positions and endgames (employing those images). In the mid-1970s, this research was expanded by inclusion of a number of experiments and printouts reflected program PIONEER runs while solving a growing subset of endgames and positions [2, 25]. Nevertheless, the subset of problems with computer-based solutions was significantly smaller than the set of those with semiformal manual solutions.

When trying to develop the very first version of the formal LG in the beginning of the 1980s, I wanted to map the images of trajectories, zones (Fig. 2) and sequences of chess positions (variants of moves) into a formal mathematical framework. My attempt was to develop a totally general mathematical representation whose 2D chess-like instantiation would "look and behave" exactly as those pictorial images. The problem was to choose the "right kind" of mathematical tools for such mapping. The only guide available to me was this 2D chess-like instantiation. Interestingly, this set of images was already used once for writing and debugging the algorithm and program PIONEER. My thinking of PIONEER was a movie-like visual stream of moving pieces, morphing trajectories and zones when moving from one chess position to another. Note that, in this stream, pieces were not jumping from one square to another but continuously crawling over the board (while trajectories and zones were morphing).

For a number of years, I was constantly running the observation visual stream, consciously and, most likely, subconsciously. This stream was similar to the one which I utilized for representing PIONEER in my mind. I was trying to find the appropriate mathematical tools to represent the essence of the pictorial LG algorithm and the program PIONEER data structures and runs. The Graph Theory did not look very appealing because it did not lend itself for creating an explicit hierarchical structure of trajectories, zones and pieces moving along those "rail tracks". An idea of representing a trajectory as a string of symbols came at one of the conferences during a talk related to the natural language understanding. A presenter manipulated words—strings of symbols, and I realized that a trajectory, i.e., a planning sequence of moves, could be represented as a sequence of symbols, one symbol per move. This was obvious commonality of the object in my visual stream and the linguistic environment. The realization of existence of commonality happened at once, seemingly, without conscious effort. However, I am sure that the pictorial LG observation visual stream was running subconsciously in my brain at that time—it was running all the time, from time to time coming back to the conscious mind.

The observation stream turned into the exploring various options of representing a trajectory as a string of symbols. Except for programming languages, at that time, I was not familiar with the general Theory of formal languages and

grammars. While learning this theory I realized that various types of formal languages could be used to represent trajectories. However, instead of representing planning moves with different symbols it would be convenient to utilize identical symbols. In such case we would have to provide every symbol with additional information specific for each move (such as an identifier of the board location related to this move), e.g., the coordinate of the chess board. In terms of the Theory of formal languages this would mean utilizing languages, which include formal semantics in addition to syntactic structures. The class of appropriate languages was found in [46], which was constructed by Volchenkov at the time when I was searching for the languages with formal built-in semantics. These were strings of symbols with sophisticated parameters generated by the so-called BUPPG grammars [34, 46]. In the next section I will introduce the reader to the controlled grammars, which is a slight modification of those BUPPG grammars.

9 Controlled grammars: introduction

Controlled grammars were developed by the author in 1980. The first paper on controlled grammars was a limited distribution Technical Report [26]; see also [34]. Looking backward, it seems natural to represent a path, a planning sequence of steps, as a string of symbols with parameters. However, at the beginning, it was not clear how to generate these strings, because the grammars developed before BUPPG were not suitable for this task.

I will introduce the class of controlled grammars informally (Table 2) because this type of informal example-based introduction guided the development of the hierarchy of formal languages for LG. For a complete formal definition see [34]. Controlled grammars generate strings of symbols with parameters. The lists of parameters incorporate semantics of the string. This semantics is determined by the problem domain, e.g., by squares of the chessboard, chess pieces, type of the robot or obstacle, days of planning period, etc. The values of actual parameters are calculated and assigned in the process of generation. Moreover, the generation itself is controlled by the state of the problem domain. This is achieved by providing the grammar with a control mechanism. This mechanism includes two major additions to the standard Chomsky structure [4], lists of productions admitted for application at each step of the derivation and conditions of applicability of the productions, i.e., certain formulas of the predicate calculus. During the generation, this control mechanism, in turn, is controlled by the problem domain through the values of formulas and actual parameters of the substring generated so far. Let us consider a rough schema of operation of such grammars (Table 2).

Every production $A(\,,\,,\,) \rightarrow B(\,,\,,\,)$ of a controlled grammar is equipped with the label l, predicate $Q(\,,\,,\,)$, separate formulas $C(\,,\,,\,) = D(\,,\,,\,)$, i.e., assignment operators, and the two sets of labels of transition in case of success and failure, F_T and F_F, respectively. Expressions shown in parenthesis may include variables and functional formulas. The grammar operates as follows (see, for example, Tables 3 and 4).

The initial permissible set of productions consists of the production with label 1. It should be applied first. Let us describe the application of an arbitrary production in a controlled grammar. Suppose that we attempt to apply production with label l to rewrite the symbol A. We have to choose the leftmost entry of symbol A in the current string

Table 2 Informal schema of controlled grammars

Label	Condition	Kernel, π_k	π_n	F_T	F_F
l	$Q(\,,\,,\,)$	$A(\,,\,,\,) \rightarrow B(\,,\,,\,)$	$C(\,,\,,\,) = D(\,,\,,\,)$	L_T	L_F
V_T is the alphabet of terminal symbols V_N is the alphabet of nonterminal symbols V_{PR} is the alphabet of the first order predicate calculus (including *Pred,* the list of predicates Q) E is the subject domain $Parm$ is the finite set of parameters L is the finite set of labels l referencing the productions					
Parameters (variables and functions) are shown in parenthesis. — If Q is true and the current string contains nonterminal A, production with label l is applied and we go to the production with label from the list L_T. — If Q is not true or the current string does not contain nonterminal A, production with label l does not apply and we go to production with label from the list L_F. Values of parameters are changed when we apply productions.					

and compute the value of predicate $\boldsymbol{Q}$ for the subject domain $\boldsymbol{E}$ (the condition of applicability of the production). There are two options for applying this production. If the current string *does not* contain $\boldsymbol{A}$, or if $\boldsymbol{Q} = \boldsymbol{F}$ *(false)*, then the application of the production is ended, and the next production is chosen from the failure section of labels $\boldsymbol{F_F}$, i.e., $\boldsymbol{F_F}$ becomes the current permissible set. If the current string *does* contain $\boldsymbol{A}$ and $\boldsymbol{Q} = \boldsymbol{T}$ *(true)*, $\boldsymbol{A}$ is replaced by the string in the right hand side of the production, $\boldsymbol{B}(\ ,\ ,\ )$; also, we carry out computation of the values of all formulas corresponding to the parameters of the symbols in the right hand side of the production $\boldsymbol{B}(\ ,\ ,\ )$ from section $\boldsymbol{\pi_k}$ and those standing separately $\boldsymbol{D}(\ ,\ ,\ )$ from section $\boldsymbol{\pi_n}$, and the parameters assume their new values. After application of the production is ended, the next production is chosen from the success section $\boldsymbol{F_T}$, which is now the current permissible set. If the permissible set is empty, the derivation halts.

10 Controlled grammar for the Tower of Hanoi

Employing the observation stream I carefully observed a simple example of controlled grammars introduced in [46], (and reproduced later in [34]), a grammar for solving the well-known Tower of Hanoi Problem.

Table 3 Controlled Grammar for the Tower of Hanoi problem

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

$V_T = \{p\}$
$V_N = \{S, A\}$
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 $n > 1$.
$Var = \{n, x, y\}$
$F = Fcon \cup Fvar$, in which
$Fcon = \{f_1, f_2\}$ (two functions), where
$f_1(n) = n - 1$,
$f_2(x, y)$ yields the value from $\{a, b, c\} - \{x, y\}$,
where values of x, y are from $\{a, b, c\}$;
$Fvar = \{m, a, c\}$ (three arity-zero functions);
$E = Z_+ \cup \{a, b, c\}$;
Parm: $S \in Var$, $A \in Var$, $p \in Var$;
$L = \{1, 2, 3\}$ (the set of labels, referencing the productions)
At the beginning of generation:
$x = a$, $y = c$, $n = m$ (number of disks).

The problem is as follows [46], (Fig. 3). Consider three pivots $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. On the first one there is a set of n disks, each of different size. The task is to move all the disks to the pivot $\boldsymbol{c}$ 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 pivots $\boldsymbol{b}$ and $\boldsymbol{c}$ can, of course, be used as temporary resting places for the disks.

Let us designate an elementary step of moving disk number i from the pivot x to the pivot y as $\boldsymbol{p}$(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:

$$p(i_1,\ x_1,\ y_1)p(i_2,\ x_2,\ y_2)\ldots p(i_r,\ x_r,\ y_r).$$

This is a string of the language of all possible sequences of moves. Consider the controlled grammar shown in Table 3.

Let us apply this grammar for generating a solution for the case of three disks: $n = 3, x = \boldsymbol{a}, y = \boldsymbol{c}$ (Fig. 3). It means that the values of parameters for the start symbol $\boldsymbol{S}$ are $\boldsymbol{S}(3, \boldsymbol{a}, \boldsymbol{b})$.

The generation is shown below. The symbol ^{k} => means application of the production with the label k.

$$\begin{aligned}
\boldsymbol{S}(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c}) &\overset{1}{\Rightarrow} \boldsymbol{A}(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\\
&\overset{2}{\Rightarrow} \boldsymbol{A}(\boldsymbol{2},\boldsymbol{a},\boldsymbol{b})\boldsymbol{p}(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\\
&\overset{2}{\Rightarrow} \boldsymbol{A}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{a},\boldsymbol{b})\boldsymbol{A}(\boldsymbol{1},\boldsymbol{c},\boldsymbol{b})\boldsymbol{p}\\
&\quad(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\\
&\overset{3}{\Rightarrow} \boldsymbol{p}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{2}.\boldsymbol{a},\boldsymbol{b})\boldsymbol{A}(\boldsymbol{1},\boldsymbol{c},\boldsymbol{b})\boldsymbol{p}\\
&\quad(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\\
&\overset{3}{\Rightarrow} \boldsymbol{p}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{a},\boldsymbol{b})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{c},\boldsymbol{b})\boldsymbol{p}\\
&\quad(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\\
&\overset{2}{\Rightarrow} \boldsymbol{p}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{a},\boldsymbol{b})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{c},\boldsymbol{b})\boldsymbol{p}\\
&\quad(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{1},\boldsymbol{b},\boldsymbol{a})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\\
&\overset{3}{\Rightarrow} \boldsymbol{p}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{a},\boldsymbol{b})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{c},\boldsymbol{b})\boldsymbol{p}\\
&\quad(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{b},\boldsymbol{a})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\boldsymbol{A}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\\
&\overset{3}{\Rightarrow} \boldsymbol{p}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{a},\boldsymbol{b})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{c},\boldsymbol{b})\boldsymbol{p}\\
&\quad(\boldsymbol{3},\boldsymbol{a},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{b},\boldsymbol{a})\boldsymbol{p}(\boldsymbol{2},\boldsymbol{b},\boldsymbol{c})\boldsymbol{p}(\boldsymbol{1},\boldsymbol{a},\boldsymbol{c}).
\end{aligned}$$

Let us prove that the grammar generates a solution of this problem in the general case of an arbitrary number of disks. We shall prove this by induction.

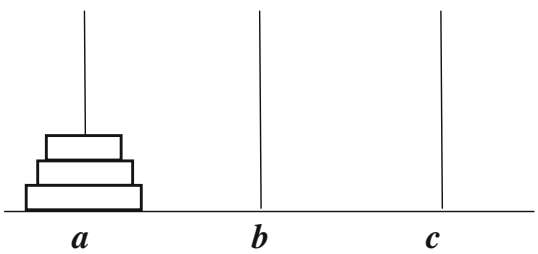


Fig. 3 Start state for the Tower of Hanoi: problem for 3 disks

1. We just proved that the grammar generates a solution of this problem for n = 3.
2. Assume that the grammar generates a solution for n = **m**.
3. Let's prove that it generates a solution for n = **m + 1**.

Consider the generation in case of n = **m + 1**. Interpretation of the start symbol is shown in Fig. 4.

$$S(\mathbf{m}+1,a,c) \overset{1}{\Rightarrow} A(\mathbf{m}+1,a,c) \overset{2}{\Rightarrow} A(\mathbf{m},a,b)p(\mathbf{m}+1,a,c)A(\mathbf{m},b,c).$$

Obviously, the following application of the grammar to the symbol $A(\mathbf{m}, a, b)$ will generate a string of symbols. By inductive assumption 2 this string corresponds to the solution of the Tower of Hanoi problem with **m** disks on the pivot a. These disks must be moved to the pivot b (Fig. 5).

When these disks are moved to b we can apply $p(\mathbf{m} + 1, a, c)$, i.e., we can move disk **m + 1** from a to c (Fig. 6).

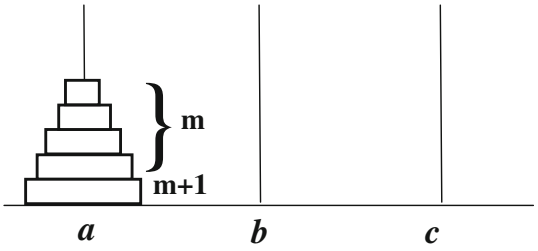


Fig. 4 Interpretation of $S(\mathbf{m} + 1, a, c)$

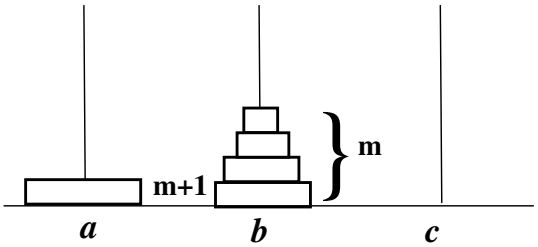


Fig. 5 Interpretation of $A(\mathbf{m}, a, b)$

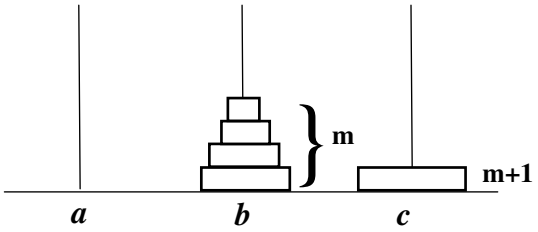


Fig. 6 Interpretation of $p(\mathbf{m} + 1, a, c)$

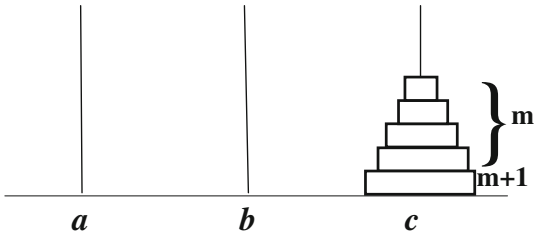


Fig. 7 Interpretation of $A(\mathbf{m}, b, c)$

The following application of the grammar to the symbol $A(\mathbf{m}, b, c)$ will generate a string of symbols. By assumption, this string corresponds to the solution of the Tower of Hanoi problem with **m** disks on the pivot b. These disks must be moved to the pivot c (Fig. 7).

This means that the grammar generated a solution for **n = m + 1** and by induction it generates solutions for all values of **n** $\geq$ 3.

11 Constructing grammar of shortest trajectories

Observation of the class of controlled grammars and, specifically, the Tower of Hanoi example permitted me to initiate the construction, i.e., to construct the grammar for generating the shortest trajectories in the ABG. A trajectory is the following string of symbols, Section VII and [34]:

$$a(x_0)a(x_1)\ldots a(x_m),$$

where all the x_i are the coordinates of the locations of X and for every $k < m$ x_{k+1} is reachable from x_k. In order to generate a string a's we need a simple recursive production

$$A \rightarrow aA,$$

where A is a nonterminal. However, we have to generate the coordinates of the locations along the trajectory, the values of parameters of the string. This could be achieved by introducing function *next* as a parameter of the nonterminal A in the right-hand side of the production as follows:

$$A(x) \rightarrow a(x)A(next(x)).$$

This function should yield the next location along the trajectory given the current location x. The recursive application of this production will result in a string of a's: $aa\ldots a$, while function *next* will be responsible for evaluation of parameters of this string. A parameter value of every next symbol of a shortest trajectory should represent a point (from X) reachable from another point, a value of the parameter of the previous symbol. Moreover, such trajectory must be the shortest string out of all the trajectories with the same begin and end points.

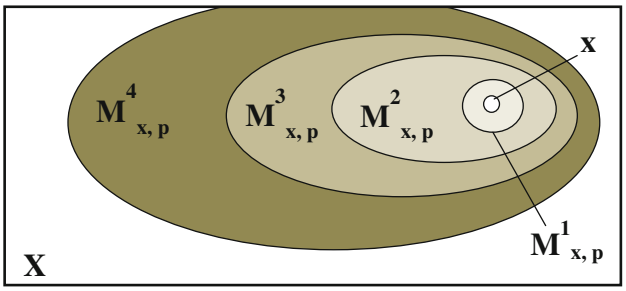


Fig. 8 Construction stream for the function $MAP_{x,\,p}$

The construction of function *next* started from constructing a "ruler" for an ABG, i.e. a function MAP for measuring *distances* between points of the abstract board.

The construction visual stream operated with various 2D shapes, such as those shown in Figs. 8 and 11, etc., that reflected subsets of locations of the abstract board X (Section VI). The construction stream utilized those shapes, i.e., the continuum-type sets, to reflect finite sets. Moreover, the artificial world reflected in this stream utilized the rules of the ABGs. It appears that I reasoned about those 2D shapes employing not the 2D Euclidean geometry but the ABG rules (Section VI) and the Set Theory. This sounds a little strange and the first impression might be that the stream used those shapes as simply illustrations. Contrary to this impression, these images together with the ABG rules guided the construction.

The construction started by defining the *map of the set* X with respect to point x and piece p for the ABG as the following mapping:

$$MAP_{x,p}: \; X \to \mathbf{N}$$

(where x is from X, p is from P, N is the set of non-negative integers), which was constructed as follows. The stream broke X into the set of subsets shown as concentric ellipses in Fig. 8 that represented the sets of locations of equal distances from x. They are called a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets $\{MAP^{k}_{x,\,p}\}$ of X. Note that the elliptical shape of the $\{MAP^{k}_{x,\,p}\}$ for the visual stream was chosen simply by the visual analogy. Indeed, the $\{MAP^{k}_{x,\,p}\}$ are analogous to the subdued convex hulls of the reachability areas for the chess King from 76 shown in Fig. 9. It is obvious that the actual $\{MAP^{k}_{x,\,p}\}$ for a piece with arbitrary reachability relations may have arbitrary shapes.

For $k = 1$: $MAP^{k}_{x,\,p}$ is a set of points *m reachable in one step* from x: $Rp(x, m) = T$ (shown as a ring in Fig. 8); this was achieved by simply checking the value of $Rp(x, m)$ for all the points $m \in X$ for a given x.

For $k > 1$: $MAP^{k}_{x,\,p}$ is a set of points *reachable in k steps but not reachable in k−1 steps,* i.e., points *m* reachable from points of $MAP^{k-1}_{x,\,p}$ and not included in any $MAP^{i}_{x,\,p}$ with *i* less than *k* (shown as nested ellipses in Fig. 8); this was achieved by checking the value of $Rp(x, m)$ for all the points $m \in X$ for a given x excluding those from $MAP^{i}_{x,\,p}$ with *i* less than *k*.

Table 4 Controlled Grammar of Shortest Trajectories $\mathbf{G}^{(1)}_{t}$

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

$V_T = \{a\}$
$V_N = \{S, A\}$
V_{PR}
$Pred = \{Q_1, Q_2, Q_3\}$,
$Q_1(x, y, l) = (MAP_{x,\,p}(y) = l) \quad (0 < l < n)$
$Q_2(l) = (l \geq 1)$
$Q_3 = T$
$Var = \{x, y, l\}$
$F = Fcon \cup Fvar$,
$Fcon = \{f, next_1, \ldots, next_n\}$ $(n = |X|)$,
$f(l) = l - 1$, $D(f) = \mathbf{Z}_+$
$Fvar = \{x_o, y_o, l_o, p\}$
$E = \mathbf{Z}_+ \cup X \cup P$
Parm: $S \to Var$, $A \to Var$, $a \to \{x\}$
$L = \{1, 3\} \cup two$, $two = \{2_1, 2_2, \ldots, 2_n\}$
At the beginning of generation:
$x = x_o$, $y = y_o$, $l = l_o$, $x_o \in X$, $y_o \in X$, $l_o \in \mathbf{Z}_+$, $p \in P$.

$next_i$ is defined as follows:
$D(next_i) = X \times \mathbf{Z}_+ \times X^2 \times \mathbf{Z}_+ \times P$
$SUM = \{v \mid v \text{ from } X, MAP_{x_o,p}(v) + MAP_{y_o,p}(v) = l_o\}$,
$ST_k(x) = \{v \mid v \text{ from } X, MAP_{x,\,p}(v) = k\}$,
$MOVE_l(x)$ is an intersection of the following sets:
$ST_1(x)$, $ST_{l_o-l+1}(x_o)$ and SUM.
if
$MOVE_l(x) = \{m_1, m_2, \ldots, m_r\} \neq \varnothing$
then
$next_i(x, l) = m_i$ for $i \leq r$ and
$next_i(x, l) = m_r$ for $r < i \leq n$,
else
$next_i(x, l) = x$
endif

Employing those reachability areas, I defined

$$MAP_{x,\,p}(y) = k,$$

for y from $MAP^{k}_{x,\,p}$ (*the number of steps from x to y*). For the remaining points, $MAP_{x,\,p}(y) = 2n$, if $y \neq x$ (n is the number of points in X) and $MAP_{x,\,p}(y) = 0$, if $y = x$.

Employing the map of the set X for piece p from P, I defined an *asymmetric distance function* on X, the *board distance*: $bd_p(x, y) = MAP_{x,\,p}(y)$.

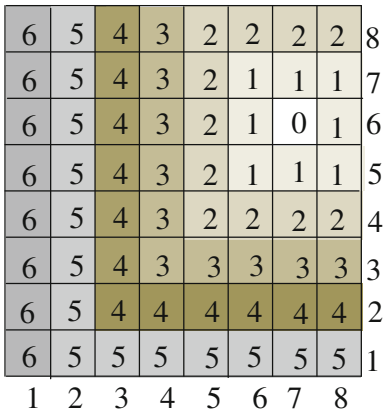


Fig. 9 Reachability areas for the King at 76

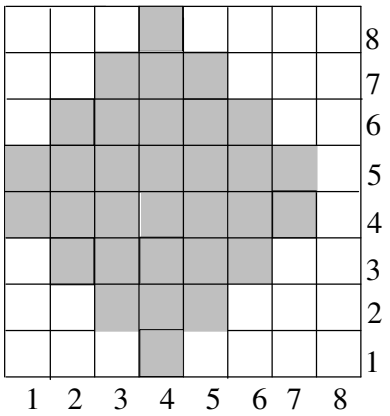


Fig. 10 Set of SUM for the shortest paths for King from 41 to 48

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 our ABG (Section VI) is a game with a symmetric relation of reachability, i.e., $R_p(x, y) = R_p(y, x)$, then

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

In this case, each of the pieces p from P specifies on X its *own metric*. In fact, I defined the distance function between points of the ABG as a *length of the shortest path between these points*. It is interesting that distances between the same two points may be different for different pieces of the ABG. Our distance is actually a reflection of time needed for different pieces to move from one point to another. For example, moving abilities of different types of chess pieces as well as those of different robots or vehicles are different.

Assume that we have to construct all the shortest paths between points x_o and y_o on X. Also assume that the length of these paths is l_o. Consider a set of points inside the ellipse SUM shown in Fig. 11. Note that the elliptical shape of the SUM for the visual stream was chosen simply by visual analogy. Indeed, a subdued convex hull of the SUM for all the shortest paths for the King from 41 to 48 is shown in Fig. 10.

It is obvious that the actual SUM for a piece with arbitrary reachability relations may have an arbitrary shape. However, it appears convenient for the visual stream to deal with elliptical shape to construct an algorithm for the function *next*. These are all the points v from X such that the sum of distances between x_o and v and between y_o and v is equal to l_o. As we know from the above, distances on X can be measured by employing the function MAP. Thus, points inside the ellipse SUM are as follows:

$$\mathrm{SUM} = \left\{ v | v \text{ from X}, \ \mathrm{MAP}_{x_o,\,p}(v) + \mathrm{MAP}_{y_o,\,p}(v) = l_o \right\}.$$

This is a collection of points of all possible shortest paths between x_o and y_o, and no points outside this ellipse can belong to the shortest paths. Having constructed a non-ordered set of the points that belong to the shortest paths, we have to order them by assigning them to different paths. Note that a point may usually belong to more than one path.

Assume that the visual stream has already constructed k-1 points of a shortest path beginning from x_o. All of them belong to the ellipse SUM. These points are shown in Fig. 11 as small dark circles. A magnified picture is shown in the right-hand side of Fig. 11. The point we are going to construct, x_k, is the k-th location along the shortest path, hence the distance between x_o and x_k must be equal to k. This means that this point must belong to the set of points ST_k (shown as a large ring in Fig. 11) such that their distances from x_o are equal to k. From definition of function MAP we know that these are the following points v:

$$\mathrm{ST}_k(x) = \left\{ v | v \text{ from X}, \ \mathrm{MAP}_{x,\,p}(v) = k \right\}.$$

That way, x_k must belong to ST_k and to SUM, i.e., it must belong to the intersection of the large ring and the ellipse—shown as a segment in the left-hand side of Fig. 11 and magnified as a rectangle—in the right-hand side. Let us switch to the magnified picture in the right-hand side. By definition, x_k is going to be the next location along the shortest path, s_o it must be reachable from x_{k-1}. This means that x_k must belong to the small ring (Fig. 11):

$$\mathrm{ST}_1(x_{k-1}) = \{ v | v \text{ from X}, \ \mathrm{MAP}x_{k-1}, \ p(v) = 1 \},$$

a set of points reachable in one step from x_{k-1}. Thus, every next point along the shortest trajectory belongs to the intersection, called MOVE, of three sets, the ellipse SUM, the large ring ST_k, and the small ring ST_1. The intersection shown in Fig. 11 includes two points $next_1$ and $next_2$. This means that the path that has currently been constructed can branch because we have two candidates for x_k. Which candidate, $next_1$ or $next_2$, is chosen should be determined by the grammar utilizing function *next*.

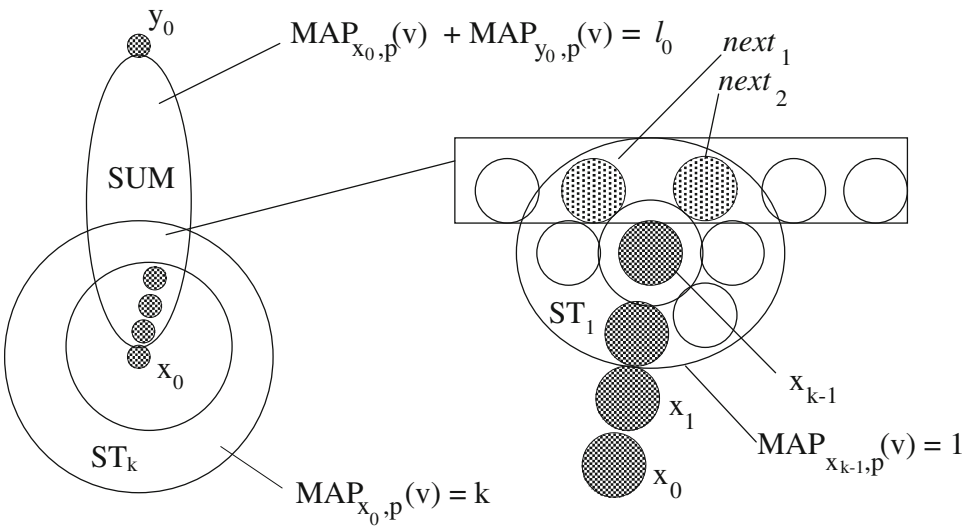


Fig. 11 Construction stream for function $next_i$ for the grammar $\mathbf{G_t^{(1)}}$

The construction stream produced a sketch of the kernel for the controlled grammar generating all the shortest trajectories. This grammar should include at least two nonterminals, ***S*** and ***A***, and three productions:

1 $S \rightarrow A$
2_i $A \rightarrow aA$
3 $A \rightarrow a$

However, production 2 should be included in multiple copies. The *i*-th copy differs from the others by the label 2_i from the set of labels *two* and by the function $next_i$. This function should be included in the parameter list of the nonterminal ***A*** in the right-hand side of the production 2_i as discussed above and in Table 4. The total number of productions 2_i is limited by the value of n, the number of points in X, but some of these productions may be identical. The number of *different* productions 2_i should be determined by the number of different values of the function $next_i$ during each step of generation.

Employing all the preliminary constructs the construction stream generated the controlled Grammar of Shortest Trajectories $\mathbf{G_t^{(1)}}$ (Table 4). It includes formal definitions of all the components including those introduced informally in Table 2, a formal definition of the function *next* (developed above), and the initial values of all the parameters. The validation stream permitted proving correctness and completeness of the grammar $\mathbf{G_t^{(1)}}$ [34]. The respective theorem was proved for the ABG with symmetric relations of reachability R_p, $R_p(x, y) = R_p(y, x)$. The construction stream for generating algorithm of the function *next* was initially utilized in the beginning of the 1970s for constructing algorithm of the chess program PIONEER [2, 34].

Note that, as usual in the grammars, one application of the grammar $\mathbf{G_t^{(1)}}$ generates only one shortest trajectory. Various implementations of $\mathbf{G_t^{(1)}}$, where all the shortest trajectories of a bundle must be evaluated, require that all the productions 2_i be applied. This is achieved either by adding a procedure backtracking along the trajectories generated so far and attempting to branch applying other productions 2_i or by executing all the productions 2_i concurrently (if we have a parallel computing environment).

Grammar of Shortest and Admissible Trajectories $\mathbf{G_t^{(2)}}$, was developed employing a similar construction stream [34]. It generated all the shortest trajectories as well as trajectories built of two shortest components. This grammar is utilized in the Grammar of Zones (Tables 7, 8).

12 Representing zone: no clues

Even after learning about the BUPPG grammars [46], constructing the class of controlled grammars and the Grammar of Shortest Trajectories (Table 4), I had no idea how to represent the networks of trajectories and, specifically, zones (Section VII). A significant leap in my understanding of the nature of the micro-model of LG as a hierarchy of subsystems was required to realize that a hierarchical network of trajectories could be represented as a string as well. Such a leap should have included an idea that whole trajectories and time constraints for movement along those trajectories could be included as parameters of symbols of the strings. This leap was supported by the intent to apply controlled grammars for generation of those strings. However, this intent could not produce a solution for some time because I could not understand how a zone, a nonlinear network with sophisticated hierarchical structure of multiple negations could be represented as a string, a simple linear object. The hierarchical structure of all the examples of zones I dealt with in the pictorial LG

algorithm could be easily "flattened" down to the 2D networks but not to a string, a 1D object.

I was running the observation and construction streams making attempts to represent a zone as a string of a controlled language. Objectively, looking back in time, it is clear that all the necessary patterns were hidden in the grammar for the Tower of Hanoi (Section X). This grammar generated a solution, which could be interpreted as a hierarchical structure of nested trees—solutions to the sequence of the lower order Tower of Hanoi problems. It was certainly a nonlinear structure represented as a string of symbols with parameters. Unfortunately, this structure was implicit and my lack of experience with linguistic representation of those structures prohibited me from realizing that. With lack of experience and in absence of the images of embedded graphs my observation stream simply glanced through the Tower of Hanoi linguistic representation without realizing deep structural analogy. It seems that it is a major problem of all roads to discoveries, i.e., of the respective observation visual streams. A stream observes an object but does not "see" its relevance, its analogy to the problem in hand, without "relevant" visible images. In my case, I needed examples of hierarchical multi-dimensional objects represented linguistically to realize those analogies. It took significant time for the observation stream and conscious effort to understand that a zone could also be effectively represented as a string of symbols employing the same class of controlled languages, so that every symbol of a zone-string would represent a trajectory from this zone.

My observation stream ran through the examples of application of various graph grammars for generation and recognition of physical images published in [8–10, 19, 22–24]. Out of those examples I will introduce the reader to the one class of languages developed by Feder [8]. In the following brief introduction to *plex grammars* I will follow [9, 12]. Note that book [12] was not available to me at the time of developing linguistic representation of zones, i.e., around 1980. I will skip the details of the formal definition of those languages and will focus on several examples, especially, on the example of the natural rubber molecule. As was the case of the controlled grammars, a complete formal definition was not important for the visual stream to realize a powerful analogy.

13 Plex languages

In an ordinary formal language every symbol has 2 "attaching-points" (left and right), which are used to connect with the adjacent symbols (like a chain). ***abc***

Let us consider "symbols" with an arbitrary number of attaching-points to connect with other symbols. Symbol with N attaching-points is called NAPE ("N attaching-point entity").

A plex grammar is the following set:

$(\boldsymbol{V_T}, \boldsymbol{V_N}, \boldsymbol{P}, \boldsymbol{S}, \boldsymbol{Q}, \mathbf{q_o})$

$\boldsymbol{V_T}$ is a finite nonempty set of terminal NAPEs;
$\boldsymbol{V_N}$ is a finite nonempty set of nonterminal NAPEs;
$\boldsymbol{P}$ is a finite set of productions;
$\boldsymbol{S}$ is the start NAPE;
$\boldsymbol{Q}$ is a finite set of symbols called identifiers $q_1, q_2, \ldots, q_m$, which are used as labels of attaching-points. Interconnections of NAPEs can only be made through the specified attaching points; $Q \cap V_T \cap V_N = \varnothing$;
$\mathbf{q_o}$ is an empty identifier, $\mathbf{q_o} \in Q$; it serves as a place marker and is not associated with any attaching points.

A production is as follows:

$$\boldsymbol{\psi}\mathbf{G}_{\psi}\boldsymbol{\Delta}_{\psi} \rightarrow \omega\mathbf{G}_{\omega}\Delta_{\omega},$$

where $\boldsymbol{\psi} = a_1 a_2 \ldots a_m$ and $\boldsymbol{\omega} = b_1 b_2 \ldots b_n$; a_i, b_j are NAPEs; $\mathbf{G}_{\psi}$, $\mathbf{G}_{\omega}$ are the lists of joint-fields for current connections. Joint field shows which attaching-points out of $q_{i1}, q_{i2}, \ldots, q_{ik}$ of k NAPEs should be connected. If q_j is included in the field from $\mathbf{G}_{\omega}$ at the position j it means that the j-th NAPE from $\boldsymbol{\omega}$ b_j participates in this joint-field. The same condition holds for the $\mathbf{G}_{\psi}$ fields. $\boldsymbol{\Delta}_{\psi}$ and $\boldsymbol{\Delta}_{\omega}$ are the lists of joint-fields of attaching-points for future connections. If j-th NAPE a_j (or b_j) is not included in a joint-field then the j-th position of this field is substituted by the empty identifier $\mathbf{q_o}$. Every field must include at least one nonempty identifier. We will use positive integer numbers as identifiers. Symbol "0" will designate $\mathbf{q_o}$. Lists $\mathbf{G}$ and $\boldsymbol{\Delta}$ will be included in parenthesis, fields will be separated by commas, empty fields are denoted by ().

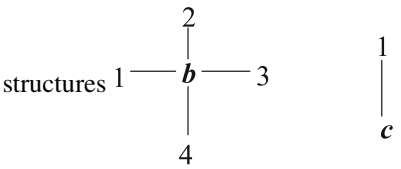
The observation visual stream ran through several examples of plex grammars.

Consider the following plex production:

$\boldsymbol{A}(1, 2, 3)$ —> $\boldsymbol{bc}$ (41)(10, 20, 30)

$\boldsymbol{A}$ is a nonterminal NAPE,

$\boldsymbol{b}$ and $\boldsymbol{c}$ are terminal NAPEs describing the following structures



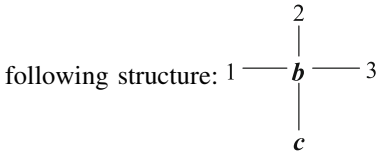
In this case

$\boldsymbol{\psi} = \boldsymbol{A}$ $\quad$ $\boldsymbol{\omega} = \boldsymbol{bc}$

$\mathbf{G}_{\psi} = \varnothing$ $\quad$ $\mathbf{G}_{\omega} = (41)$

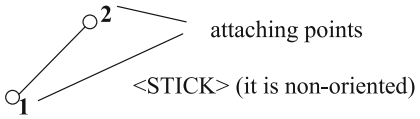
$\boldsymbol{\Delta}_{\psi} = (1, 2, 3)$ $\quad$ $\boldsymbol{\Delta}_{\omega} = (10, 20, 30)$

When the production is applied, ***A*** is rewritten as the following structure:



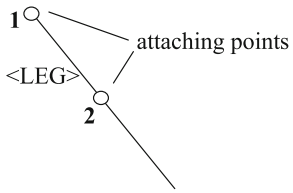
Below NAPE names are enclosed in brackets <>. Consider a plex grammar generating letters "A" and "H":

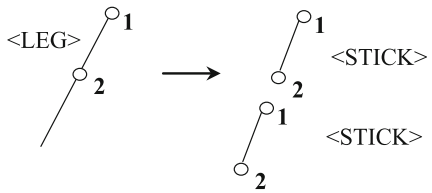
$\mathbf{V_T} = \{<\text{STICK}>\ (1, 2)\}$



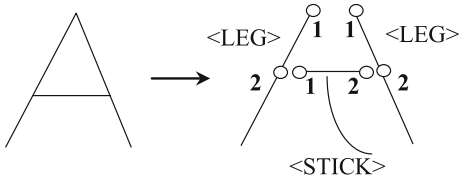
$\mathbf{S} = <\text{LETTER}>$

$\mathbf{V_N} = \{<\text{LETTER}>, <\text{A}>, <\text{H}>, <\text{LEG}>\ (1, 2)\}$

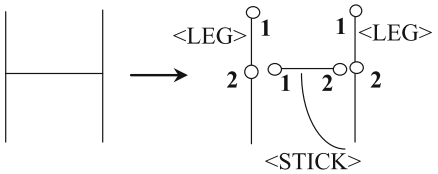




4) <LEG> (1, 2) → <STICK> <STICK> (21) (10, 21)



3) <A> → <LEG> <LEG> <STICK> (110, 201, 022)



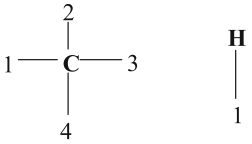
4) <H> → <LEG> <LEG> <STICK> (201, 022)

A complete grammar of two letters is as follows:

1) <LETTER> → <A> | <H>
2) <A> → <LEG> <LEG> <STICK> (110, 201, 022)
3) <H> → <LEG> <LEG> <STICK> (201, 022)
4) <LEG> (1, 2) → <STICK> <STICK> (21) (10, 21)

Consider a plex grammar generating chemical structure of the molecule of natural rubber (Fig. 12).

We introduce two terminal NAPEs:



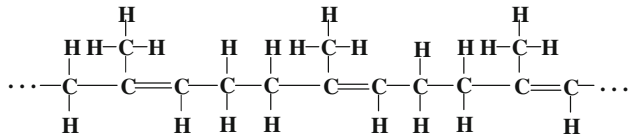
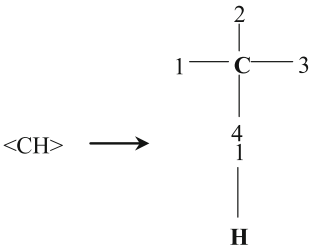
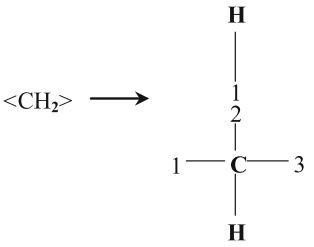


Fig. 12 Chemical structure of the molecule of natural rubber

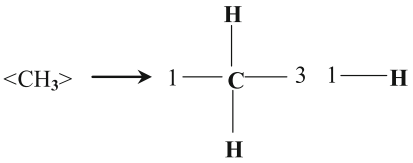
Let us construct nonterminal NAPEs <CH>, <CH_2> and <CH_3>.



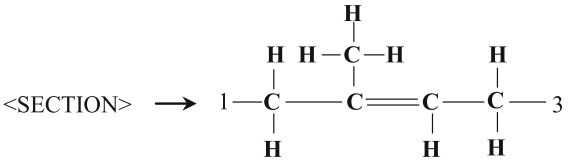
5) <CH> (1, 2, 3) → <C> <H> (41)(10, 20, 30)



4) <CH_2> (1, 2) → <CH> <H> (21)(10, 30)



3) <CH_3> (1) → <CH_2> <H> (31)(10)



2) <SECTION> (1, 3) → <CH_2> <C> <CH_3> <CH> <CH_2>
(31000, 02100, 03020, 04010, 00031) (10000, 00003)

A complete grammar of the molecule of natural rubber (Fig. 12) is shown in Table 5.

Table 5 Plex grammar of the molecule of natural rubber

1.	<CHAIN> (1, 2) → <SECTION> <CHAIN> (31)(10,03) | < SECTION > ()(1,3)
2.	<SECTION > (1, 3) → <CH_2> <C> <CH_3> <CH> <CH_2> (31000, 02100, 03020, 04010, 00031) (10000, 00003)
3.	<CH_3> (1) → <CH_2> <H> (31)(10)
4.	<CH_2> (1, 2) → <CH> <H> (21)(10, 30)
5.	<CH> (1, 2, 3) → <C> <H> (41)(10, 20, 30)

14 From plex languages to zones

The grammar of natural rubber was the turning point. The observation stream produced a clue. This was an analogy I was looking for. It was a nonlinear 2D structure folded into a string employing nonterminal NAPEs and lists of joint-fields. Although, the final string generated by the grammar in Table 5 does not look like a string of terminal symbols typical for the Chomsky grammars, it was clear that it could be unfolded into such a string if required. When switching the observation stream back to the controlled grammars I realized several important analogies. A terminal symbol of the future grammar of zones could be associated with a whole trajectory via a parameter list. Thus, every trajectory of a zone could be reflected by the same terminal symbol with a parameter evaluated as a string generated for this trajectory by the grammar $\mathbf{G}_t^{(1)}$ or $\mathbf{G}_t^{(2)}$. In such case, the final string to be generated would include the number of terminal symbols equal to the number of different trajectories in this zone. In addition, the zone's network structure could be preserved employing the analog of the joint-fields of the plex languages. The analog of those fields for trajectories of a zone could be created by marking the location of the abstract board where one trajectory has to be connected to another and storing the coordinate of this location. (Note that an abstract board is a finite set, so the locations can be marked, enumerated or assigned a coordinate.) This type of the "joint-field" based on the coordinates should have provided precision and respective rigidness to the resulting structure as opposed to various kinds of structural grammars including plex grammars observed by the observation stream [8–10, 19, 22–24]. Indeed, those grammars generated structures, which could be stretched in all directions. For example, the rubber molecule reflected by the plex string could be squeezed or expanded in many different ways. Essentially, I would guess that a plex string reflects a class of the homotopy equivalent structures with terminal symbols as fixed points. For example, for the grammar of natural rubber (Table 5) every bond line could be considered as a class of the homotopy equivalent lines. This would be inappropriate for the zones, especially, those reflecting real world systems. The observation stream concluded that the coordinate-based joint-fields supported by the class of controlled grammars would provide the required rigidness. By the way, this kind of rigidness was already achieved for the trajectories employing the controlled grammar $\mathbf{G}_t^{(1)}$ (Table 4) based on the coordinates of trajectory links. In fact, a trajectory generated by this grammar cannot be stretched at all.

15 Time distribution

The observation stream turned to the several examples of zones generated by the pictorial LG algorithm for various chess-like ABGs (Section VI). I will discuss observations on example of the zone sketched in Fig. 2 (repeated in Fig. 13). The main issue observed was the allotted time distribution introduced in Section VII. A formal definition of a zone did not exist at that time and the notion of time distribution was always explained via the permission rule. (Note that a formal definition of a zone independent of the algorithm of construction was first introduced 20 years later, in 2000 [34].) This permission rule means that only those opposing (to the main piece) trajectories are to be included in the zone where the movement of the piece makes sense, i.e., the length of the trajectory is less than (or equal) the amount of time allotted to it. Thus, when a zone is constructed, every trajectory has to be assigned a number, the amount of time allotted to it. The observation stream concluded that for every trajectory this number could be generated employing simple algorithm utilized already for drawing examples of zones via the pictorial LG algorithm as well as the program PIONEER. The time allotted to the second and higher negation trajectories is equal to $TIME(y) - l + 1$, where $TIME(y)$ is the time already allotted to the lower negation trajectory to which the current one is attached (at the location y); l is the length of the current trajectory. For the 1st negation trajectories the time allotted is just $TIME(y)$, where $TIME(y)$ is the ordinal number of the location y of the main trajectory to which the current trajectory is attached. This algorithm is recursive and requires step-by-step zone construction, i.e.,

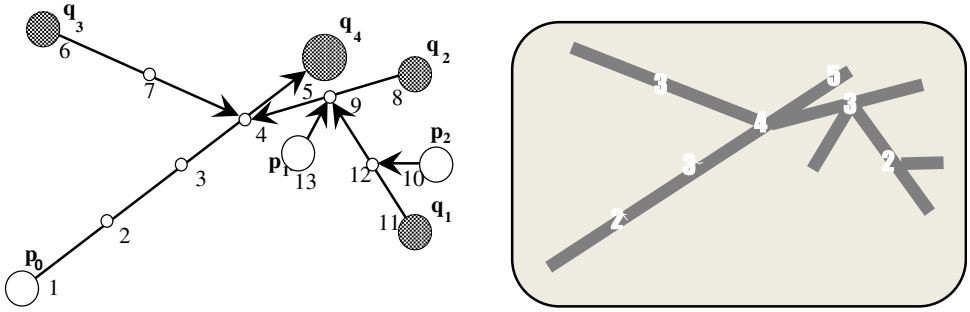


Fig. 13 Construction stream sketching the time distribution for a zone generation grammar

beginning from the main trajectory, then, 1st negation trajectories, following with all the subsequent higher negations sequentially. The current negation should be started when the previous (lower) negation is finished. Every inclusion of a trajectory in a zone should be accompanied by assigning the time allotted to it according to the algorithm considered above. The main trajectory is the special case in a way that such assignment should be performed for every location along this trajectory—they should be assigned just their ordinal numbers. As long as the trajectories are firmly assigned to the abstract board via the coordinates of their locations, the value of time allotted to a trajectory in a zone could also be assigned to the appropriate location of the board, specifically, to the end location of the trajectory. For the main trajectory the time assignment should be made for all the stop locations along the trajectory (except for the start location). The time distribution for the sample zone (shown in Fig. 13, left), is in Fig. 13, right. The numbers shown are the values of time assigned to the respective board locations. I assumed that this assignment is made via the array *TIME*(z) whose index z represents the coordinate of every location of the abstract board. In Fig. 13, right, this array *TIME* is shown as a gray rectangle with rounded edges meaning that it represents a screen showing the locations of the abstract board where the time distribution numbers are assigned. The dark bold lines are present just for the reader to understand which trajectories these numbers represent. Otherwise, I could use a clear gray screen (as an area representing abstract board) with the time distribution numbers (shown in white).

16 Sketching zone generation without a grammar

I initiated the construction stream to generate sample zone shown in Fig. 2 (and in Fig. 13, left) as a hierarchical network of trajectories. Most importantly, I intended to sketch generation of this zone as a string of symbols (Table 1) without a grammar. A grammar should be developed based on this sketch.

The stream started from introducing a dummy Start Symbol $\boldsymbol{S}(u, v, w)$, where the second and third parameters are shown as gray screens (Table 6). These gray screens are the models of the abstract board of the ABG considered above (Fig. 13, left and right). In addition, two more duplicates of the abstract board were introduced, *TIME* and *NEXTTIME* (Table 6). All the four arrays were to be updated in the course of generation. At the beginning of the generation all the four arrays are empty—clear gray screens. To generate all the required trajectories I could employ the Grammar of Shortest Trajectories $\mathbf{G}_{\mathbf{t}}^{(1)}$ (Table 4). In the complete Grammar of Zones I utilized a more advanced grammar of shortest and admissible trajectories $\mathbf{G}_{\mathbf{t}}^{(2)}$ [34], and Table 8. The first parameter u in such case could carry the values of input parameters for this grammar $u = (x, y, l)$.

The stream produced the first terminal symbol $\boldsymbol{t}(p_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)$ with the main trajectory as a value of the second parameter (Table 6, 2nd line). The key part of this generation step is the nonterminal $\boldsymbol{A}(u, v, w)$ as well as the current values of *TIME* and *NEXTTIME*, Table 6, 3d line.

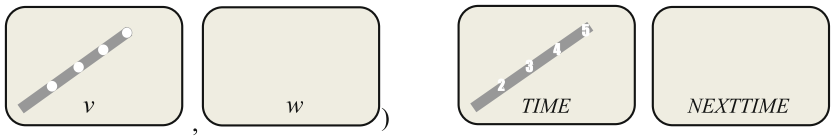
Screen v includes markings (small white circles) at the locations 2, 3, 4 and 5 of the main trajectory; screen *TIME* includes values of the allotted time assigned to the same locations. These locations should be considered as potential end points for the 1st negation trajectories. This means that v and *TIME* store information for generating 1st negation trajectories on the following generation steps. Indeed, on the next step, the stream produced another terminal symbol $\boldsymbol{t}(q_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)$ with the first negation trajectory, Table 6, 4th line. The nonterminal symbol $\boldsymbol{A}(u, v, w)$ produced on this step includes screen w where location 9 of the trajectory $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ is marked, Table 6, 5th line. Screen *NEXTTIME* includes value of the allotted time at the same location—it is 3. This means that w and *NEXTTIME* store information for generating the 2nd negation trajectories on the following generation steps. Screens v and *TIME* are left unchanged. This generation step should not be considered as the second generation step as it is shown in Table 6. The reader should understand that the construction stream produced a sketch of the generation, i.e., the major steps, while several intermediate steps could have been skipped. The following lines 4 and 5 (Table 6) show generation of the terminal symbol $\boldsymbol{t}(q_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$ which should complete generation of the 1st negation trajectories. The next generation step should pass information about the end points and the allotted time for generating 2nd negation trajectories. This is shown in lines 6 and 7 (Table 6) where no new terminal symbols were generated. However, the previous values of the screens w and *NEXTTIME* were assigned to v and *TIME*, respectively, while the w and *NEXTTIME* were cleaned after that. This way, the same productions, which were utilized for generating 1st negation trajectories, could be reused for generating 2nd negations. After generating 2nd negations in lines 8 through 13 (Table 6), the same procedure was utilized in lines 14-16 to pass information about potential 3d negation trajectories to v and *TIME* for reusing the same productions for the 3d negation. This process stopped at the 3d negation because generation of those 3d negation trajectories did not produce information for generating higher negation trajectories. This happened because the 3d negation trajectories were too short (just 10–12 of length 1, Fig. 13, left) to leave any marks in the screens w and *NEXTTIME* (Table 6, line 3 from the bottom of the page). Emptiness of v after

Table 6 Construction stream sketching zone generation including the trajectories generation and the time distribution

$\boldsymbol{S}(u,$ 

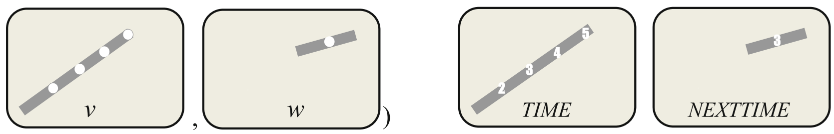
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)$

$\boldsymbol{A}(u,$ 

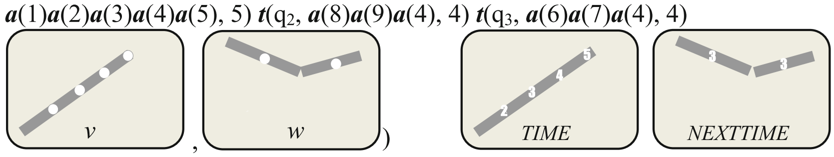
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)$

$\boldsymbol{A}(u,$ 

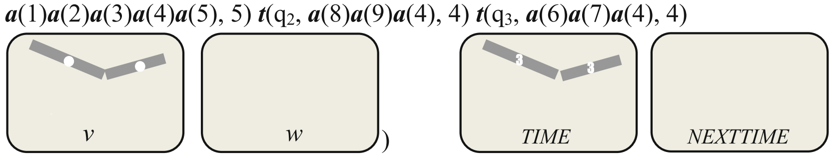
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$

$\boldsymbol{A}(u,$ 

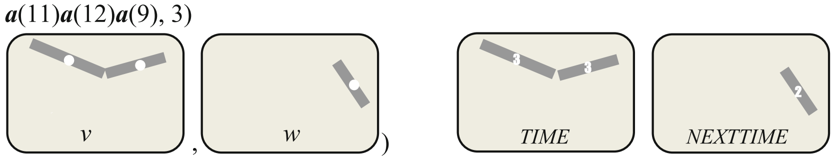
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$

$\boldsymbol{A}(u,$ 

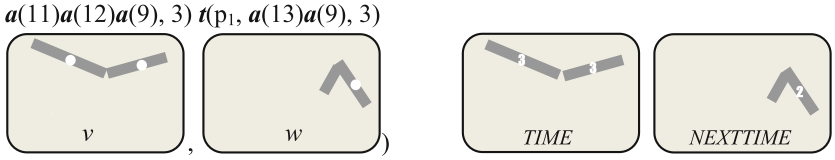
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$
$\boldsymbol{t}(\mathrm{q}_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 3)$

$\boldsymbol{A}(u,$ 

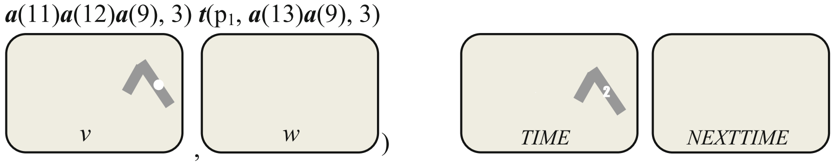
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$
$\boldsymbol{t}(\mathrm{q}_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(\mathrm{p}_1, \boldsymbol{a}(13)\boldsymbol{a}(9), 3)$

$\boldsymbol{A}(u,$ 

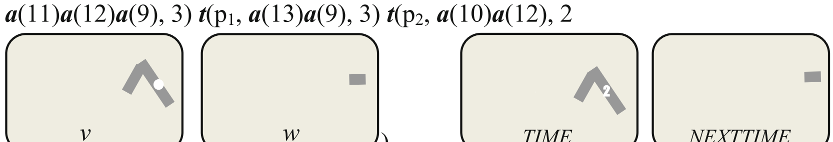
$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$
$\boldsymbol{t}(\mathrm{q}_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(\mathrm{p}_1, \boldsymbol{a}(13)\boldsymbol{a}(9), 3)$

$\boldsymbol{A}(u,$ 

$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$
$\boldsymbol{t}(\mathrm{q}_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(\mathrm{p}_1, \boldsymbol{a}(13)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(\mathrm{p}_2, \boldsymbol{a}(10)\boldsymbol{a}(12), 2$

$\boldsymbol{A}(u,$ 

$)$

$\rightarrow \boldsymbol{t}(\mathrm{p}_0, \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5), 5)\ \boldsymbol{t}(\mathrm{q}_2, \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4), 4)\ \boldsymbol{t}(\mathrm{q}_3, \boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4), 4)$
$\boldsymbol{t}(\mathrm{q}_1, \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(\mathrm{p}_1, \boldsymbol{a}(13)\boldsymbol{a}(9), 3)\ \boldsymbol{t}(\mathrm{p}_2, \boldsymbol{a}(10)\boldsymbol{a}(12), 2)$

completion of generating the 3d negation trajectories was considered as a stop signal for completion of the entire generation sketch. This sketch demonstrated a possibility of generating a zone as a rigid network of trajectories with allotted time distribution.

17 Complete grammar of zones

I employed the construction stream for constructing a complete grammar of zones (Tables 7, 8). The screens *v, w, TIME* and *NEXTTIME* were implemented as n-vectors, where n is the number of locations of the abstract board X. Functions *g, h, h^0, init, ALPHA, DIST* were introduced to implement the trajectory marks and time assignments (for the screens), sketched visually in Section XVI. Function *f*, combined with multiple cyclic applications of productions 3 and 4*j*, was introduced to implement the scan through all locations of the abstract board X to register those, which could serve as the start locations for potential negation trajectories.

Grammar of zones $\mathbf{G_Z}$ generates the following strings of symbols:

$$Z = \boldsymbol{t}(p_o, t_o, t_o)\boldsymbol{t}(p_1, t_1, t_1)\ldots\boldsymbol{t}(p_k, t_k, t_k).$$

Symbol $\boldsymbol{t}(p_o, t_o, t_o)$ related to the main trajectory of the zone is generated employing productions 1 and 2*i*, a substring $\boldsymbol{t}(p_1, t_1, t_1)\ldots\boldsymbol{t}(p_{m1}, t_{m1}, t_{m1})$ related to the 1st negation trajectories is generated next employing productions 3 and 4*j*. Production 5 allows the grammar to switch to the next negation, to generation of the substring of symbols related to the trajectories of the 2nd negation. The second parameter *v* of the nonterminal $\boldsymbol{A}((0, 0, 0), w, zero)$ assumes the value of *w*, while vector *NEXTTIME* is explicitly assigned to *TIME* in the section $\boldsymbol{\pi_n}$ of this production. These two assignments pass information about the end points and the allotted time for generating 2nd negation trajectories stored in *w* and *NEXTTIME* on the previous steps. Production 5 performs generation step shown in the sketch in lines 6 and 7 (Table 6). This way, the 2nd negation trajectories are generated employing the same series of productions 3, 4*j*, and 5 as the 1st negation trajectories. All the trajectories of higher negations are generated employing the same series. Generation is stopped by applying production 6 after production 5 fails—predicate $\boldsymbol{Q}_5$ shows that no more trajectories of higher negation exist.

18 Complete hierarchy of languages

To complete the hierarchy of languages I had to formalize movement. It was obvious that movement along a trajectory could be implemented by shortening the respective string of symbols, i.e., by dropping symbols one by one (one symbol per move), from the front of this string. It was much harder to understand what is happening to the string representing a zone when pieces are moving inside this zone.

To understand this movement I was running the observation stream in the form of several pictorial LG mental movies for different chess problems with the pieces moving over the chess board. The formal representations of all the trajectories and zones, the strings of symbols, were mentally and often visually (on paper) attached as tags to the respective visual images. The purpose was to represent those morphing pictorial structures of trajectories and zones via changes to their formal representations. The key analogy for representing morphing was as follows. When a piece moves along a trajectory, this trajectory shortens—the part of the route left behind disappears; so does the respective string of symbols by dropping symbols from the front when a piece moves from one square to another. When pieces move inside a zone this zone shrinks; so does the string, representing this zone, by dropping some of the symbols, though, not necessarily from the front. The part of a zone that was dropped on the way forward was called a freezing part, to unfreeze on the way backward. To reveal formally the structure of the freezing part required substantial analysis employing formal representation. In the end, the entire movement process was formalized as a sequence of translations of languages of trajectories and zones. I will investigate the respective streams in the future papers.

The only remark I will make here about the complete hierarchy of languages is related to its inventive power and the respective validation visual stream. From the very beginning, I could see that the result of this construction was not continuous and not even dynamic as was my original pictorial LG visual stream, Sections VII, VIII and [37]. As described briefly above, to introduce dynamics into the construction stream, I represented movement (morphing) as a sequence of snapshots, i.e., different languages. When the game moves from one state to another the hierarchy of languages (that describes the first state) is translated into the different hierarchy of languages (that describes the second state). Thus, a variant of moves corresponds to a sequence of translations of languages. At this point I started the stream of dynamic hierarchy to validate the newly constructed hierarchy of formal languages. I envisioned an exotic movement as a sequence of hopper jumps between multiple parallel plains in 3D Euclidian space. Every plain represented a game state with corresponding two languages, trajectories and zones "located on this plain". Thus, when analyzing mentally a variant, I was hopping from one plain to another, which represented translation from the languages "of one plain to another".

Table 7 Grammar of Zones $\mathbf{G_Z}$

L	Q	Kernel, π_k ($\forall z \in X$)	π_n ($\forall z \in X$)	F_T	F_F
1	Q_1	$S(u, v, w) \rightarrow$ $A(u, v, w)$		*two*	∅
2_i	Q_2	$A(u, v, w) \rightarrow$ $t(h_i^o(u), l_o+1)$ $A((0, 0, 0), g(h_i^o(u), w), zero)$	$TIME(z) = DIST(z, h_i^o(u))$	3	∅
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	$A(u, v, w) \rightarrow e$		∅	∅

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

V_{PR}

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

$Q_1(u) = (ON(p_o) = x) \wedge (MAP_{x,p_o}(y) \le l \le l_o) \wedge (\exists q\ ((ON(q) = y) \wedge (OPPOSE(p_o, q))))$

$Q_2(u) = T$

$Q_3(u) = (x \ne n) \vee (y \ne n)$

$Q_4(u) = (\exists p\ ((ON(p) = x) \wedge (l > 0) \wedge (x \ne x_o) \wedge (x \ne y_o)) \wedge ((\neg OPPOSE(p_o, p) \wedge (MAP_{x,p}(y) = 1)) \vee (OPPOSE(p_o, p) \wedge (MAP_{x,p}(y) \le l))))$

$Q_5(w) = (w \ne zero)$

$Q_6 = T$

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

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

$Func = Fcon \cup Fvar$;

$Fcon = \{f_x, f_y, f_l, g_1, g_2, ..., g_n, h_1, h_2, ..., h_M, h_1^o, h_2^o, ..., h_M^o, DIST, init, ALPHA\}$,
$f = (f_x, f_y, f_l)$, $g = (g_{x_1}, g_{x_2}, ..., g_{x_n})$,
$M = |\mathbf{L_t}^{l_o}(S)|$ is the number of trajectories in $\mathbf{L_t}^{l_o}(S)$;

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

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

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

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

At the beginning of generation:

$u = (x_o, y_o, l_o)$, $w = zero$, $v = zero$, $x_o \in X$, $y_o \in X$, $l_o \in \mathbf{Z_+}$, $p_o \in P$,
$TIME(z) = 2n$, $NEXTTIME(z) = 2n$ for all z from X.

However, to really understand how the strings change, what they are translated into, I had to mentally run a version of the "classic" pictorial LG visual stream concurrently with the hopper jumps. The new formal LG did not bring additional inventive power. The strings were convenient mathematically but the strings-based hopper jumps did not show visually the required inter-dependences between the trajectories inside a zone while the pictorial

Table 8 Definition of functions of the Grammar of Zones $\mathbf{G_Z}$

$\mathrm{D}(init) = \mathrm{X} \times \mathrm{X} \times \mathbf{Z}_+ \times \mathbf{Z}_+$

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

$\mathrm{D}(f) = (\mathrm{X} \times \mathrm{X} \times \mathbf{Z}_+ \cup \{0, 0, 0\}) \cup \mathbf{Z}_+^{\mathrm{n}}$

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

$\mathrm{D}(DIST) = \mathrm{X} \times \mathrm{P} \times \mathbf{L}_{\mathbf{t}}^{\boldsymbol{l_0}}(\mathrm{S})$.

Let $\mathrm{t_o} \in \mathrm{L}_{\mathrm{t}}^{l_o}(\mathrm{S})$, $\mathrm{t_o} = \boldsymbol{a}(\mathrm{z_o})\boldsymbol{a}(\mathrm{z_1})...\boldsymbol{a}(\mathrm{z_m})$, $\mathrm{t_o} \in \mathrm{t}_{\mathrm{p_o}}(\mathrm{z_o}, \mathrm{z_m}, \mathrm{m})$;

If $((\mathrm{z_m} = \mathrm{y_o}) \wedge (\mathrm{p} = \mathrm{p_o}) \wedge (\exists\, \mathrm{k}\ (1 \leq \mathrm{k} \leq \mathrm{m}) \wedge (\mathrm{x} = \mathrm{z_k}))) \vee$
$(((\mathrm{z_m} \neq \mathrm{y_o}) \vee (\mathrm{p} \neq \mathrm{p_o})) \wedge (\exists\, \mathrm{k}\ (1 \leq \mathrm{k} \leq \mathrm{m} - 1) \wedge (\mathrm{x} = \mathrm{z_k})))$
then $DIST(\mathrm{x}, \mathrm{p_o}, \mathrm{t_o}) = \mathrm{k}+1$
else $DIST(\mathrm{x}, \mathrm{p_o}, \mathrm{t_o}) = 2\mathrm{n}$.

$\mathrm{D}(ALPHA) = \mathrm{X} \times \mathrm{P} \times \mathbf{L}_{\mathbf{t}}^{\boldsymbol{l_0}}(\mathrm{S}) \times \mathbf{Z}_+$

$$ALPHA(\mathrm{x}, \mathrm{p_o}, \mathrm{t_o}, \mathrm{k}) = \begin{cases} \max(NEXTTIME(\mathrm{x}), \mathrm{k}), & \text{if } (DIST(\mathrm{x}, \mathrm{p_o}, \mathrm{t_o}) \neq 2\mathrm{n}) \\ & \quad \wedge (NEXTTIME(\mathrm{x}) \neq 2\mathrm{n}); \\ \mathrm{k}, & \text{if } DIST(\mathrm{x}, \mathrm{p_o}, \mathrm{t_o}) \neq 2\mathrm{n}) \\ & \quad \wedge (NEXTTIME(\mathrm{x}) = 2\mathrm{n}); \\ NEXTTIME(\mathrm{x}), & \text{if } DIST(\mathrm{x}, \mathrm{p_o}, \mathrm{t_o}) = 2\mathrm{n}). \end{cases}$$

$\mathrm{D}(g_{\mathrm{r}}) = \mathrm{P} \times \mathbf{L}_{\mathbf{t}}^{\boldsymbol{l_o}}(\mathrm{S}) \times \mathbf{Z}_+^{\mathrm{n}}$, $\mathrm{r} \in \mathrm{X}$

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

$\mathrm{D}(h_i^o) = \mathrm{X} \times \mathrm{X} \times \mathbf{Z}_+$; Let $\mathrm{TRACKS}_{\mathrm{p_o}} = \{\mathrm{p_o}\} \times (\bigcup_{1 \leq \mathrm{k} \leq l} \mathrm{L}[\mathbf{G}_{\mathbf{t}}^{(2)}(\mathrm{x}, \mathrm{y}, \mathrm{k}, \mathrm{p_o})]$;

If $\mathrm{TRACKS}_{\mathrm{p_o}} = e$
then $h_i^o(u) = e$
else $\mathrm{TRACKS}_{\mathrm{p_o}} = \{(\mathrm{p_o}, \mathrm{t}_1), (\mathrm{p_o}, \mathrm{t}_2), ..., (\mathrm{p_o}, \mathrm{t_b})\}, (\mathrm{b} \leq \mathrm{M})$ **and**

$$h_i^o(u) = \begin{cases} (\mathrm{p_o}, \mathrm{t}_i), & \textbf{if } i \leq \mathrm{b}, \\ (\mathrm{p_o}, \mathrm{t_b}), & \textbf{if } i > \mathrm{b}. \end{cases}$$

$\mathrm{D}(h_i) = \mathrm{X} \times \mathrm{X} \times \mathbf{Z}_+$; Let $\mathrm{TRACKS}_{\mathrm{p}} = \{\mathrm{p}\} \times (\bigcup_{1 \leq \mathrm{k} \leq l} \mathrm{L}[\mathbf{G}_{\mathbf{t}}^{(2)}(\mathrm{x}, \mathrm{y}, \mathrm{k}, \mathrm{p})]$;

If $\mathrm{TRACKS}_{\mathrm{p}} = e$
then $h_i(u) = e$
else $\mathrm{TRACKS}_{\mathrm{p}} = \{(\mathrm{p}_1, \mathrm{t}_1), (\mathrm{p}_1, \mathrm{t}_2), ..., (\mathrm{p_m}, \mathrm{t_m})\}, (\mathrm{m} \leq \mathrm{M})$ **and**

$$h_i(u) = \begin{cases} (\mathrm{p}_i, \mathrm{t}_i), & \textbf{if } i \leq \mathrm{m}, \\ (\mathrm{p_m}, \mathrm{t_m}), & \textbf{if } i > \mathrm{m}. \end{cases}$$

Trajectories $\mathrm{t_i}$ should not be embedded (as sub-trajectories) in the trajectories of the same negation.

LG visual stream did. Unfortunately, with all its advantages, the hierarchy of formal languages did not increase the inventive power of the LG algorithm—the hopper jumps of the visual stream of dynamic hierarchy were non-sophisticated.

In a sense, rigorous mathematical formalization dried out the original pictorial LG visual stream. Though all further generalizations of LG were written in the hierarchy of formal languages, the new examples, the first results in

new domains and even the new approaches in LG (such as the No-search approach [38]) were based on the thought experiments with the same, original pictorial LG visual stream. Here, I mean different instantiations of this stream for 2D/4A, 3D/4A, TC 2D/4A, etc. [37].

The constructed hierarchy permitted to prove correctness of the grammars generating trajectories and zones, and, the algorithm of translations [34]. Consequently, this proved the correctness of the 2D/Chess implementation, i.e., of the related major procedures of program PIONEER. Later, it permitted to evaluate run time of those procedures and the entire algorithm of LG [34]. Most importantly, it opened up the doors for the new implementations, i.e., for the different problem domains.

19 Conclusion

This paper is a step in our research, discovering the algorithm for inventing new algorithms. The ultimate goal of this research is to program this algorithm of discovery. In this paper I investigated this algorithm via analysis of the development of the hierarchy of the formal languages, a formal representation of linguistic geometry. This analysis and our previous results [37, 38] show that this algorithm manifests itself during execution via the mental visual streams. The streams run consciously and unconsciously, from time to time coming back to the conscious mind. This paper confirms the hypothesis that those streams are the only interface available for this brain phenomenon. The visual streams can run concurrently and can exchange information between each other. This information may include data structures and algorithms. The visual stream interface of the algorithm of discovery seems to be very unusual and very powerful at the same time.

This paper reveals the greater role of analogy in organizing visual streams. Not only the observation stream is terminated when the essential analogy is found, the stream itself is morphed around analogous events and objects until it reaches its final form most familiar (analogous) to the past experience. The construction stream takes over from the observation stream and operates in a familiar environment prepared by the observation stream. Employing the construction goal, also inherited from the observation stream, it builds new object which is a reconstruction of the familiar components. On examples of development of the two LG formal grammars this paper demonstrates the hypothesis of constructing the construction visual stream. It shows a route how we move by erasing the particulars from constructing a simple concrete example to a visual construction model and further to a symbolic theory-based shell attached to the visual model in the form of tags. This retagging provides a set of theoretical constraints to the execution of the visual model employing the laws of a respective theory. It seems this is the way the streams are built and focused in the right direction.

References

1. Botvinnik M (1970) Chess, computers, and log-range planning, Springer, Berlin
2. Botvinnik M (1984) Computers in chess: solving inexact search problems, Springer, Berlin
3. Brown J (2011) The laboratory of the mind: thought experiments in the natural sciences, 2nd edn, Routledge. Taylor & Francis Group, New York
4. Chomsky N (1957) Syntactic structures. Mouton Publishers, The Hague
5. Deheaene S (2007) A few steps toward a science of mental life. Mind Brain Educ 1(1):28–47
6. Deheaene S (2009) Edge In Paris, Talk at the Reality Club: Signatures of Consciousness. http://edge.org/3rd_culture/dehaene09/dehaene09_index.html
7. Einstein A (1991) Autobiographical notes. Open Court, La Salle
8. Feder J (1971) Plex languages. Inform Sci 3:225–241
9. Fu KS (1974) Syntactic methods in pattern recognition. Academic Press, New York
10. Fu KS (1982) Syntactic pattern recognition and applications. Prentice Hall, Englewood Cliffs
11. Gleick J (1992) Genius: the life and science of Richard feynman, pantheon books, a division of Random House, New York
12. Gonzalez R, Thompson M (1978) Syntactic pattern recognitionL: an introduction. Addison-Wesley, Wesley
13. Hadamard J (1996) The mathematician's mind: the psychology of invention in the methematical field. Princeton University Press, Princeton
14. Kosslyn S, Thompson W, Kim I, Alpert N (1995) Representations of mental images in primary visual cortex. Nature 378:496–498
15. Kott A (ed) (2004) Advanced Technology Concepts for Command and Control, Xlibris Corporation
16. Kott A, McEneaney W (eds) (2007) Adversarial Reasoning: computational approaches to reading the opponent's mind, Chapman & Hall, London
17. Linguistic Geometry Tools: LG-PACKAGE, with Demo DVD, 60 pp., STILMAN Advanced Strategies 2010. This brochure and 8 recorded demonstrations are also available at www.stilman-strategies.com
18. Miller A (1996) Insights of genius: imagery and creativity in science and art, Copernicus, Springer, Berlin
19. Narasimhan R (1966) Syntax–directed interpretation of classes of pictures. Comm ACM 9:166–173
20. Nasar S (2001) A beautiful mind. Touchstone, New York
21. Nersessian N (2009) Conceptual change: creativity, cognition, and culture, in the book. In: Meheus J, Nicles T (eds) Models of discovery and creativity, Springer, Berlin, pp 127–166
22. Pavlidis T (1977) Structural pattern recognition. Springer, New York
23. Rosenfeld A (1979) Picture languages formal. Models for picture recognition. Academic Press, London
24. Shaw AC (1969) A formal picture description scheme as a basis for picture processing system. Inf Control 19:9–52
25. Stilman B (1975) Formation of the set of trajectory bundles, appendix 1 to the book: on the cybernetic goal of games. In: Botvinnik MM (ed) Soviet Radio, Moscow (in Russian), pp 70–77
26. Stilman B Ierarhia formalnikh grammatik dla reshenia prebornikh zadach (Hierachy of Formal Grammars for Solving Search Problems), Tech. Report, VNIIE, Moscow (in Russian), pp 105

27. Stilman B (1993) A formal language for hierarchical systems control. Int J Lang Design 1(4):333–356
28. Stilman B (1993) A linguistic approach to geometric reasoning. Int J Comput Math Appl 26(7):29–58
29. Stilman B (1993) Network languages for complex systems. Int J Comput Math Appl. 26(8):51–80
30. Stilman B (1994) Linguistic geometry for control systems design. Int J Comput Their Appl 1(2):89–110
31. Stilman B (1994) Translations of network languages. Int J Comput Math Appl 27(2):65–98
32. Stilman B (1997) Managing search complexity in linguistic geometry. IEEE Trans Syst Man Cybernet 27(6):978–998
33. Stilman B (1997) Network languages for concurrent multi-agent systems. Int J Comput Math Appl 34(1):103–136
34. Stilman B (2000) Linguistic Geometry: From Search to Construction. Kluwer Academic Publishers (now Springer), Dordrecht, pp 416
35. Stilman B (2011) Linguistic geometry and evolution of intelligence. ISAST Trans Comput Intell Syst 3(2):23–37
36. Stilman B (2011) Thought experiments in linguistic geometry. In: Proceedings of the 3d International Conference on Advanced Cognitive Technologies and Applications—COGNITIVE'2011, Rome, 25–30 Sep 2011, pp 77–83
37. Stilman B (2012) Discovering the discovery of linguistic geometry, Int J Mach Learn Cybernet (accepted for publ)
38. Stilman B (2012) Discovering the discovery of the no-search approach, Int J Mach Learn Cybernet (accepted for publ)
39. Stilman B, Yakhnis V, Umanskiy O (2000) Winning strategies for robotic wars: defense applications of linguistic geometry. Artif Life Robotics 4(3)
40. Stilman B, Yakhnis V, Umanskiy O (2002) Knowledge acquisition and strategy generation with LG wargaming tools. Int J Comput Intell Appl 2(4):385–409
41. Stilman B, Yakhnis V, Umanskiy O (2007) Chapter 3.3. Strategies in Large Scale Problems, in [16], pp 251–285
42. Stilman B, Yakhnis V, Umanskiy O (2010) Linguistic geometry: the age of maturity. J Adv Comput Intell Intell Informat 14(6):684–699
43. Stilman B, Yakhnis V, Umanskiy O (2010) Revisiting history with linguistic geometry. ISAST Trans Comput Intell Syst 2(2):22–38
44. Stilman B, Yakhnis V, Umanskiy O (2011) The primary language of ancient battles. Int J Mach Learn Cybernet 2(3):157–176
45. Ulam S (1976) Adventures of a mathematician. Charles Scribner's Sons, New York
46. Volchenkov N (1979) Interpretator beskontekstnikh upravliaemikh parametricheskikh programmnikh grammatik (Interpreter of Context-free Controlled Programmed Grammars with Parameters) in Voprosy Kibernetiky: Intellektualnie banky dannikh (Proc. on Cybernetics: Intelligent Data Banks), USSR Academy of Sciences, Sci. Board on Complex Problem Cybernetics, pp 147–157 (in Russian)
47. Von Neumann J (1958) The Computer and the Brain. Yale University Press, London
48. Watson J (1998) The Double Helix. Scribner Classics, New York