International Journal of Advanced Science and Technology
Volume 5, April, 2009

Reflective Thinking, Machine Learning, and User Authentication via Artificial K-lines

Anestis A. Toptsis
Dept. of Computer Science and Engineering,
York University, Toronto, Ontario, Canada
anestis@yorku.ca

Abstract

Artificial K-lines (AKL) is a structure that can be used to store different types of knowledge, as long as this knowledge is represented by series of events connected by causality. Unlike, and, perhaps, complementary to, Artificial Neural Networks (ANN), AKL can combine inter-domain knowledge and its knowledge base can be augmented dynamically without rebuilding of the entire system. In this paper we demonstrate the diversity of AKL by illustrating, through examples, its workings for three applications across three completely different areas of study. The first example demonstrates that our structure can generate a solution where most other known technologies are either incapable of, or very complicated in doing so. The second example illustrates a novel, human-like, way of machine learning. The third example presents a behavior metrics based method for password authentication.

1. Introduction and Background

Artificial Intelligence (AI) is enjoying a renewed interest which makes its presence welcome in many aspects of our daily life. Even before, and certainly since the appearance of the phrase "AI", the following questions are of utmost importance: How come people are able to learn so much? How come people are creative (i.e., able to perform a new task, different from two or more previously learned tasks, by being "inspired" by their previous experiences and knowledge)? These issues have puzzled philosophers and cognitive scientists for many years, and with the appearance of AI as a field related to those disciplines, they are among the core AI questions as well. Numerous attempts to provide an overall answer to these issues, failed during the past 50 years. However, several "theories of memory" have emerged. Notable examples are [1], [2], [3], and [4]. None of these approaches has been fully implemented to date; however, there have been several reports toward this end. Examples are [5] and [6]. A central theme in [1], [2], and [3] is the concept of K-lines. Quoting from *[1],*

> *When you "get an idea," or "solve a problem" […] you create what we shall call a K-line. […]…When that K-line is later "activated", it reactivates […] mental agencies, creating a partial mental state "resembling the original".*

In some of our previous works (e.g. [7], [11], [12]) we have used a form of K-lines to address media handling issues in affective computing systems. In this paper, inspired by the concept of K-lines, we introduce a structure that can possibly exhibit the caliber of intelligence usually attributed to Artificial Neural Networks (ANN). To the best of our knowledge, no such proposal exists for using K-lines in the way described here. The rest of the paper is organized as follows.

Section 2 describes the proposed structure. This is repetition of our most recent work, as found also in [8] and [9]. Section 3 shows through an example how reflective thinking is inherent to the AKL structure and provides some discussion on how AKL is related to artificial creativity. Section 4 illustrates through an example how AKL can be used for a new type of machine learning. Section 5 describes a behavior metrics based method for password authentication. Section 6 provides a comparison of our method with the essentials of the ANN structure. Section 7 is the conclusion and ideas for future research.

2. Artificial K-lines

We define a K-line to be a *sequence of associated events* $e_1, e_2, \ldots, e_k$, such that e_i and e_{i+1} are connected by *causality*. That is, the occurrence of event e_{i+1} is a direct consequence of the occurrence of event e_i. If two K-lines contain the same event, then they are said to *intersect* at that event. When two K-lines are created, if they do not intersect, they form a graph like the one shown in Figure 1. If the K-lines intersect, then they form a graph like the one shown in Figure 2.

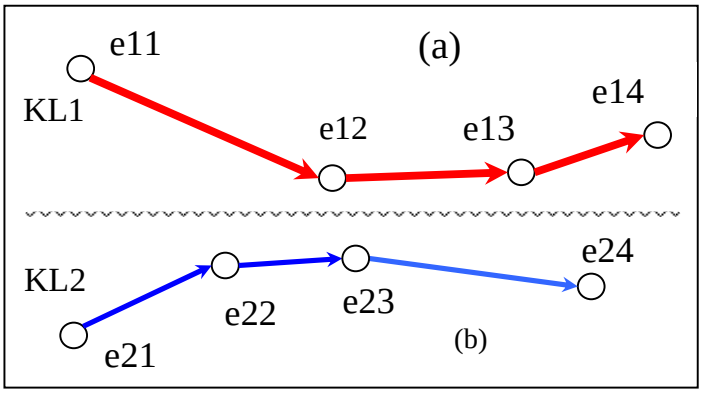

Figure 1. Two K-lines that do not intersect.

In Figure 1, the nodes (circles in the graph) of each K-line represent events. An edge such as (e11, e12) means that event e12 is a consequence of event e11. In figure 1, K-lines KL1 and KL2 do *not* intersect. Consequently, the graph provides two possible "ways to think", as shown in parts (a) and (b) of Figure 1.

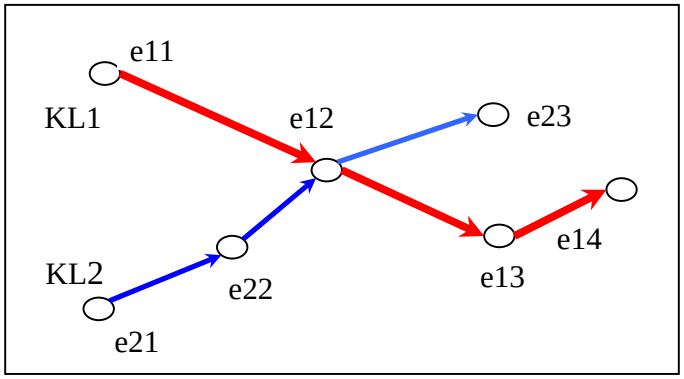

Figure 2. Two intersecting K-lines.

In Figure 2, K-lines KL1 and KL2 intersect. Consequently, the graph provides four possible "ways to think", as shown in Figures 3 (a), (b), (c), and (d).

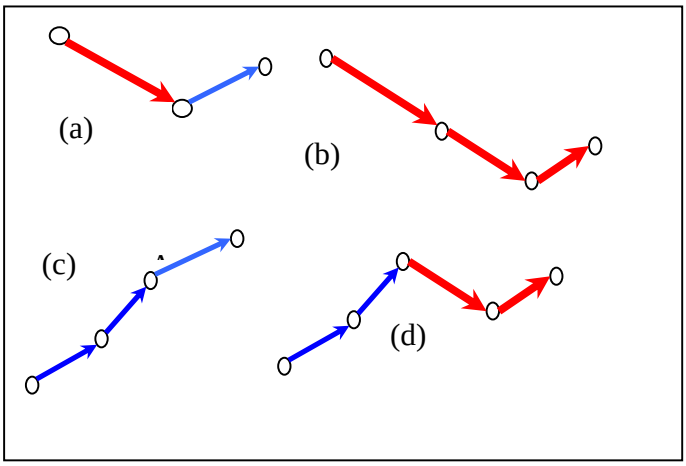

Figure 3. Four possible ways to think: (a), (b), (c), (d).

Each way of thinking is created by starting from one of the available K-lines, and then, once we encounter an intersection, we follow each of the two possible paths. Note, for every intersection that we encounter, the number of "ways to think" is multiplied by the outdegree of the node that is at that intersection. Therefore, the more intersections we have, the more times the number of "ways to think" is multiplied. This is the crucial point in our method. Note, by combining 2 K-lines (KL1 and KL2) we have a system (Fig. 2) capable to "think" in *more* ways than is possible to think if the 2 K-lines are not combined (Fig. 1). By adding more intersecting K-lines, it is expected that the number of "ways to think" is increased *much faster* than the number of the added K-lines. Also note, some of the possible formed "ways to think" are comprised by *parts* of existing K-lines. This is the essence of creativity, i.e., to generate a new idea by using parts of old ideas. The above discussion certainly does not constitute proof, it, nevertheless, fits the essence of how people are able to learn *so much* (by forming intersecting K-lines in their brains) and also how people are *capable of creativity* (by forming new ideas from parts of some of the formed K-lines).

3. Reflective Thinking and Artificial Creativity

In this section we first illustrate through an example (subsection 3.1) how AKL can facilitate reflective (or, alternative ways of) thinking – i.e., the switching from one way of thinking WT1 to another way of thinking WT2 in case that WT1 fails to produce any solution. Then, in subsection 3.2 we discuss the potential of AKL for artificial creativity.

3.1. Robot & Assembly Line Example

We describe an example of how the proposed method can work, and illustrate its benefits. We create two K-lines with events based on two different scenarios that can occur in hypothetical, but realistic environments. Using those problems, we create K-

lines of events and note the possible intersections of these K-lines. Then, we pose a task to be performed and we show that the AKL graph is capable to easily produce a solution to this task, whereas it is not obvious how an easy solution can be produced without the AKL graph. We use this example also in [8], [9]. It is presented here for completeness and to illustrate the diversity of use of AKL. Our setting considers two problems that occur, in general, in different realms.

The first problem is a robot movement problem. This problem is described in [5] and it is used here to apply our method for its solution. (We believe that the approach presented here is significantly simpler than the one used in [5]). The second problem is a situation that can occur as part of the operation of a typical assembly line. Figure 4(a) illustrates the situation for the first problem and Figure 4(b) illustrates the situation for the second problem. In Figure 4(a), the robot is needed to move the box, from room A to room B, passing it through the opening that connects the two rooms. In Figure 4(b), a box is on a moving belt. As part of the assembly line process, the first worker unfolds the box and the second worker flattens the box further and places it on the moving belt again. Then, the moving belt carries the box through an opening and the flattened box reappears at the other side of the opening. After that, further processing of the box may occur, but this is not relevant to our example.

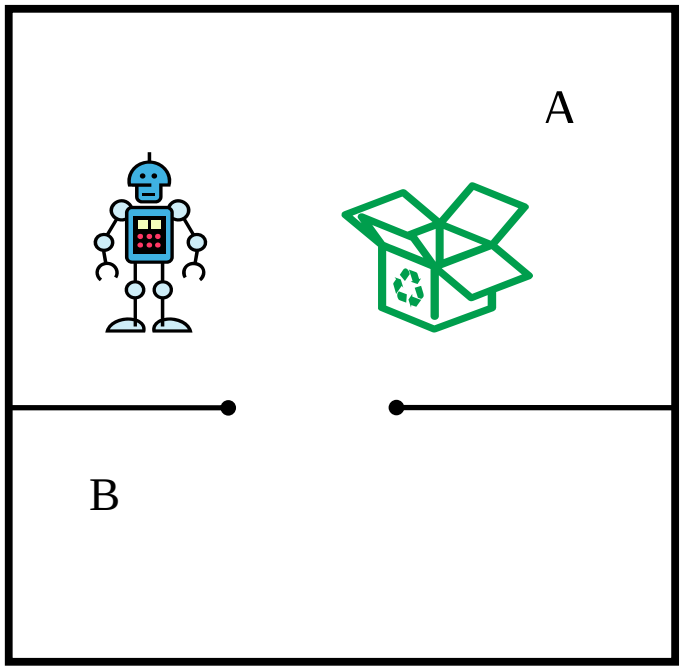

Figure 4.a. (a) Robot has to move box from area A to area B through opening. (b) Assembly line for processing boxes.

A K-line of events that captures a possible process of the robot moving the box through the opening is shown in Figure 5. And a possible K-line of events that captures the process of the assembly line is shown in Figure 6.

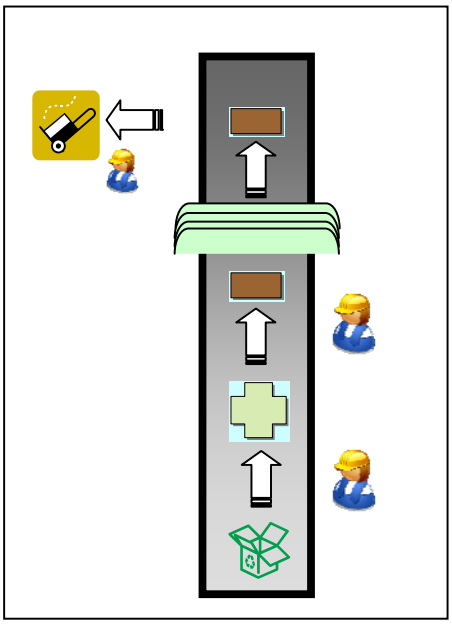

Figure 4.b Assembly line for processing boxes.

K-line KL1	
Event in Node	**Node ID**
(robot) Move toward box	11
(Robot) pick up box	12
Move box toward opening	13
Move box through opening	14
Box passed through opening	15
Box appears at other side of opening	16

Figure 5. Robot K-line.

K-line KL2	
Event in Node	**Node ID**
Box placed on moving belt	21
(Worker) pick up box	22
(worker) unfold box	23
Box passed through opening	24
Box appears at other side of opening	25

Figure 6. Assembly line K-line.

Note, in Figures 5 and 6, some nodes are essentially the same, as shown in Table 1.

Table 1. Node equivalences between 2 K-lines.

Node of KL1	Node of KL2
12	22
15	24
16	25

Each row of Table 1 shows two nodes, one from K-line KL1 and one from K-line KL2. Each row of Table 1 contains nodes that are the same, or essentially equivalent, according to the K-lines of Figures 5 and 6. Based on this observation, the two separate K-lines are said to *intersect*, and they can be merged into a graph, as shown in Figure 7.

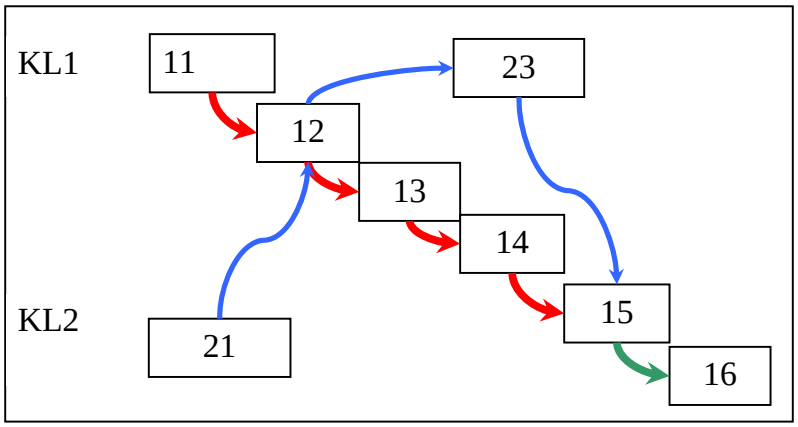

Figure 7. Intersecting K-lines.

Note, in Figure 7, the edge connecting nodes 15 and 16 belongs to both K-lines. We argue that *the knowledge conveyed by Figure 7 is superior to the knowledge conveyed individually by Figures 5 and 6.* Next, we explain why this is the case.

Consider the following problem: In a situation like the one described in Figure 5, we would like the robot to move the box from room A to room B. Also, we assume that the box is too big to fit through the opening that connects the two rooms, but the robot doesn't know this! (The same problem with the same assumption is considered in [5], where a method based on reflective planning is developed to solve it). If we use the *individual* K-lines of Figure 5 or Figure 6 (i.e., without those K-lines being merged), then there are two possible solutions, S1 and S2: (S1) start from K-line KL1 and produce the sequence 11-12-13-14-15-16; (S2) start from K-line KL2 and produce the sequence 21-22-23-24-25. Note, *S1* will *fail* at step 14 since the box does not fit through the opening between rooms A and B; and *S2* is *not applicable*, since the starting point, 21, is outside the domain of the robot problem! Therefore, the system of two *separate* K-lines KL1 and KL2 is incapable of solving this problem.

However, if we consider the same K-lines but with the K-lines being merged, (as shown in Figure 7), then we can generate 4 (instead of 2) possible solutions: (G1) start from K-line KL1 and produce the sequence 11-12-13-14-15-16; (G2) start from K-line KL1 and produce the sequence 11-12-23-15-16; (G3) start from K-line KL2 and produce the sequence 21-12-23-15-16; (G4) start from K-line KL2 and produce the sequence 21-12-13-14-15-16. Note, solutions G1 and G3 are the same as S1 and S2, respectively, and therefore either fail or are not applicable. Solution G4 is also not

applicable, since it starts from K-line KL2. However, solution G2 *is* applicable (since it starts from K-line KL1) *and* it will also *succeed*! Note, G2 is also the result of *combining* 2 K-lines, KL1 and KL2 and G2 is not available as an option if the two K-lines are not merged.

3.2 Artificial K-lines and Artificial Creativity

Creative thinking is one of the holy grails of Artificial Intelligence. Needless to say, we all are in awe of human beings that are deemed as being highly creative or able to make discoveries or inventions. Yet, *no-one* yet has been able to provide a scientific explanation of the mechanisms of creativity. For many, creativity is a gift whose explanation and mechanics are beyond human understanding. During the past 100 years, some of the greatest creative minds have written about *their own* understanding of creative thinking (e.g., [30], [31]), while others (such as the eminent psychologist Mihaly Csikszentmihalyi in [32]) have tried to identify what are the visible characteristics of the most creative people of our time.

An interesting possible application of AKL is the area of *artificial creativity*. Note, an inherent characteristic of AKL is to combine parts of K-lines and form new K-lines comprised of those parts. We argue that this is the essence of creativity, that is, the ability to form new "ideas" by using existing knowledge, or by combining (parts of) old ideas. In the context of the AKL, the newly formed K-lines represent the new ideas, whereas the existing knowledge is the sequence of segments of the existing K-lines that are used to form the new K-lines. In this sense, the AKL might be a suitable candidate structure for artificial creativity.

Recently, a measure for creativity, the *Creativity Quotient*, CQ, has been proposed in [33]. In the context of CQ, the degree of creative thinking depends on how many different ideas (ideation) one can generate as well as on the degree of originality and the number of different uses (or, categories) of each idea (fluency). The former (ideation) is one of the oldest measurements of creativity, as given in [34]. The latter (fluency) is a fairly newer measurement (e.g. [36], [37]) whose purpose is to weigh ideas in terms of their diversity. Specifically, the greater the number of categories that an idea belongs to, the more heavily that idea contributes to the Creativity Quotient. In [33], the CQ is defined as

$$CQ = \sum_{j=1}^{N} \log_2 \left(n_j + 1 \right) \qquad (1)$$

where N is the number of categories, and n_j is the number of ideas in each category.

Based on the above definition, we can easily see that an AKL such as the one described in this section can achieve a very high CQ. This is because due to the K-line intersections in an AKL, n_j (the number of ideas) attains very high values. Note, in any AKL, n_j is roughly

$$n_j = O\left(\prod_{i=1}^{M} \alpha_i \cdot \beta_i \right)$$

where α_i is the in-degree of the i-th K-node in the AKL, β_i is the out-degree of the i-th K-node in the AKL, and M is the number of K-line intersection in the AKL. Of course, in order to express the definite CQ of an AKL based on expression (1), we need to know N, the number of categories for each idea. Unfortunately, we do not know, for the time being, how to express N. However, this in general has been acknowledged as a difficult problem and as one whose answer is highly subjective. Quoting from [35] (pg. 64),

"…it is somewhat difficult to define categories in any truly objective, or unique way."

We hope that future research will provide some insights toward this direction.

4. Artificial K-lines and Machine Learning

AKL can be useful in machine learning scenarios. We illustrate our case for a simple board game – Tic-Tac-To. The method can be applied to any strategy game, such as chess, go, etc. We use this example also in [9]. It is presented here for completeness and to illustrate the diversity of use of AKL.

The game of Tic-Tac-To (TTT) is played by 2 players, PX and PO, using a 3x3 board, as shown in Figure 8.

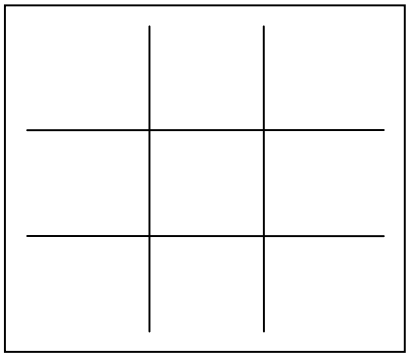

Figure 8. Typical board for Tic-Tac-To game.

Players PX and PO make alternating moves during which each player marks the board with a 'X' or an 'O'. Player PX uses always a 'X", while player PO uses always an 'O'. If any of the two players manages to mark the board with three 'X' or three 'O' to form a straight line (horizontal, or vertical, or diagonal), then that player wins. If the board has been completely marked and none of the two players achieves a 3-same-symbol straight line, then the game is a tie, i.e., none of the two players wins. Figure 9 shows two winning and one tie board configurations.

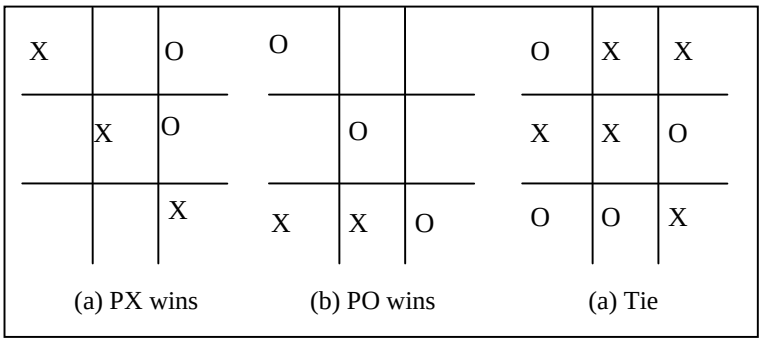

Figure 9. Two winning and one tie Tic-Tac-To board configurations.

Next we describe how the AKL can be utilized to facilitate computer learning for the game of Tic-Tac-To, when one of the players is a computer, C, and the other player is a

human, H. In our setting, C marks the board with the symbol 'C' and H marks the board with the symbol 'H'. If either C or H manage to achieve a straight line of 3-same symbols, (C, C, C or H, H, H) then the corresponding player wins; if the board has been completely marked and none of the two players achieved a straight line with its symbols, the game is a tie and a new game may begin. In our setting, we form a K-line for each game. The assumption is that many games are played, i.e., many K-lines are formed. At the beginning, before any game is played, we assume that the computer knows what are the rules of the game (i.e., it knows that a legal move is to mark any one empty slot of the board with its symbol 'C') but it does not possess any strategic knowledge regarding what constitutes a good move. On the other hand, it is assumed that H is an expert in this game. Note, for the simple game of Tic-Tac-To, any adult human of average intelligence can be considered an expert. The idea is that as each player (either C, or H) makes a move a K-node is formed. The K-nodes that are formed by successive moves are connected to form a K-line. The K-line ends with the win of one of the players, or when a tie occurs. Then a new game begins and a new K-line is formed, and so on. Prior to the first move of the very first game, the computer has no intelligence. Therefore, when it is its turn to move, it just makes a random (but legal) move. But as the game progresses, and especially as more and more games are played, the computer becomes capable of making more intelligent moves by drawing on the knowledge that it has been stored in the AKL that has been formed up to that point. The details of the K-node formation and the AKL formation are described next.

Event Structure. As mentioned earlier, each K-line consists of K-nodes and each K-node contains an event. The events of two consecutive K-nodes are related by causality, i.e., the event of the successor K-node is an immediate consequence of the event of the immediate predecessor K-node. In our setting, we define the event to be a move. Figure 10 shows two consecutive K-nodes K1 and K2 with their events $e^{\langle K1\rangle}$ and $e^{\langle K2\rangle}$.

In Figure 10, each K-node K (K = K1 or K2), contains 2 Tic-Tac-To board configurations, $F_B^{\langle K\rangle}$ and $F_A^{\langle K\rangle}$. $F_B^{\langle K\rangle}$ denotes the Tic-Tac-To board configuration *before* the move during the formation of K-node K and $F_A^{\langle K\rangle}$ denotes the Tic-Tac-To board configuration *after* the move during the formation of K-node K. Note, for two K-nodes K1 and K2 to be connected such that K1 is the predecessor of K2, it must be that $F_A^{\langle K1\rangle} = F_B^{\langle K2\rangle}$. That is, the resulting TTT board in the predecessor K-node K1 must be the starting board configuration in the successor K-node K2. This is the criterion that qualifies events $e^{\langle K1\rangle}$ and $e^{\langle K2\rangle}$ to be associated by causality. Based on the above discussion, we now describe a methodology according to which an AKL is formed and the computer learns, in the context of the game of Tic-Tac-To.

Methodology. Without loss of generality, we assume that the computer makes the first move upon starting the game. At this point, without having any strategic knowledge, the computer places a 'C' mark on the TTT board and the 1st K-node K11 of the 1st K-line L1, is formed.

The event of K11 is

$$e^{\langle K11\rangle} : F_B^{\langle K11\rangle} \rightarrow F_A^{\langle K11\rangle}$$

where $F_B^{\langle K11\rangle} = \varnothing$, the empty TTT board, and $F_A^{\langle K11\rangle}$ is the TTT board containing the 'C' mark placed by the computer-player.

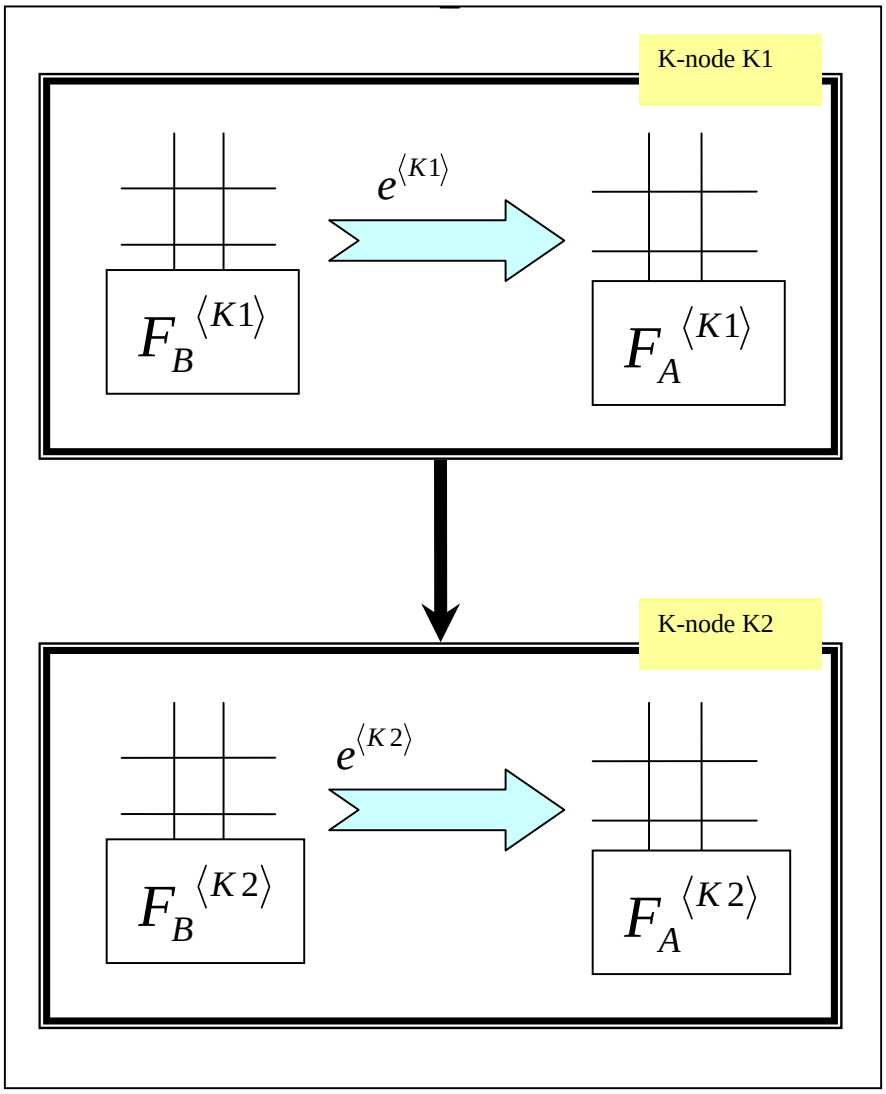

Figure 10. Two consecutive K-nodes and their events.

Then, the human player plays and the system forms a new K-node, K12. The event of K12 is

$$e^{\langle K12\rangle} : F_B^{\langle K12\rangle} \rightarrow F_A^{\langle K12\rangle}$$

where $F_B^{\langle K12\rangle} = F_A^{\langle K11\rangle}$ and $F_A^{\langle K12\rangle}$ is the TTT board configuration resulting from the move of the human player. For the remaining of the 1st K-line, the game continues in this fashion (i.e., the computer makes a legal move without any kind of strategic knowledge, and the human responds with another move) until one of the two players wins (most likely, the human in this case) or the game comes to a tie. At this point, we

record also the number of moves for this game. Then, the 2nd game begins and a new K-line, L2, starts forming. Again, the computer plays first and the 1st K-node K21 is formed, with event

$$e^{\langle K21\rangle}: \varnothing \to F_A^{\langle K21\rangle}.$$

Next, the human moves and K-node K22 is formed, with event

$$e^{\langle K22\rangle}: F_B^{\langle K22\rangle} \to F_A^{\langle K22\rangle}.$$

Recall, $F_A^{\langle K21\rangle} = F_B^{\langle K22\rangle}$. Next, it is the computer's turn to move. Note, since there is now previous knowledge available (from K-line L1), the computer investigates if that knowledge can be utilized to help it in making its next move. In doing so the computer looks if there is already a K-node K, such $F_B^{\langle K\rangle} = F_B^{\langle K22\rangle}$ and such that the K-line L containing that K-node K leads to (i) a win for the computer, or (in the absence of a computer-winning K-line) (ii) to a tie, or (in the absence of (i) and (ii)) (iii) L is the longest K-line that leads to a human-win. In other words, the computer player tries to minimize its chances for defeat, and in doing so it looks for a move that – from past experience, lead to either a computer win, or (if that is not available) to a tie, or (if that is also not available) to a move that will postpone the defeat of the computer player the longest. Assume K_X is the K-node that is selected for the computer player to move, based on the above conditions/criteria. Then, the computer player moves by making the move dictated by the found K-node K_X. Note, at this point an *intersection* is formed in the AKL. Namely, the current K-line L2 intersects with the K-line containing the found K-node K_X. The game resumes, with the human player making the next move. After that, the computer player moves and its move is, again, based on the criteria described above – i.e., by finding (if available) a K-node that leads to an educated guess of postponing its defeat for the longest. The process continues, as described above, and new K-nodes are formed, as applicable, K-line intersections are also formed (as described above), and the AKL is taking shape, for as long as it is desired.

Testing. In testing our system, we judge the computer's learning progress by monitoring the amount of effort that is progressively required by the human to win. As the AKL becomes denser, it is expected that the computer player will choose K-nodes that belong to winning K-lines and, as a result, the moves that correspond to those K-nodes will make it harder for the human player to win. Of course, for the simple game of Tic-Tac-To, the human can always win, or, in the worst case, force a tie. However, for a more complicated game such as chess (or go), it is not at all clear that the human can keep winning after several games have been played. We expect that with enough training of an AKL, the computer can become a better expert in any of these complex games. We intent to test our method and report our findings, in a sequel paper.

Why is this way of machine learning interesting. Our proposed machine learning method (via the formation and utilization of an AKL) is novel and interesting because it does *not require that any knowledge is available up front*. The AKL can be formed progressively, as more games are played (i.e., more knowledge and experience is acquired), and there is also no limit of how much knowledge can be accumulated. As more knowledge and experience is acquired and captured into the AKL, the computer learner's performance is expected to improve. This, we argue, resembles the way that actual human beings learn. That is, as more experiences are acquired with the passage

of time, humans, in general, perform better. Another interesting point in the nature of the learning facilitated by the AKL is that the *quality of learning is proportional to the quality of the "teacher"*. It is expected that an AKL whose K-node are formed by moves that are made by a not-so-intelligent (or, not-so-careful) human being, will be inferior to an AKL whose K-nodes are formed by moves that are made by a highly intelligent (or, skillful, or very careful) human being. As such, a machine that bases its moves on an inferior AKL is expected to underperform a machine that bases it moves on the superior AKL. Note, this analogy also resembles the real-life scenario in which, in general (all other things being equal), students of better teachers tent to become more skillful (i.e., learn more) than students of not-so-good teachers.

5. Artificial K-lines and User Authentication

AKL can be useful in pattern recognition situations. We illustrate this here for a simple scenario of *identity detection/verification.* All previous works that we are aware of for this problem employ neural network (NN) technology (a discussion of such works is found toward the end of this subsection). Here, we describe an authentication mechanism without using NN.

In our case, the setting is that a user attempts to gain access to a computer system (the system could be a computer account, or a bank account accessed through a bank's online service, or any system that requires the user to input an access code, such as a password). Approaches that address this problem today are basically by typing the required password by using a conventional keypad (most commonly used practice), or by identity verification via biometrics (most commonly used biometric today is the fingerprint – see for example [25], [26], [27]. For our example, we assume that the input is provided by the user via a keypad, and the user needs to type his password in order to gain access to the system. Upon typing, the system compares the typed characters against its password database, and if a 100% match occurs then the system grants access to the user. The problem with the above (typical) method of password/identity verification is that the password is vulnerable in at least two ways. First, there is the obvious scenario of password theft. Second, and probably more likely nowadays, is the continuous threat from attacking software (and computer hackers) that attempt to gain access to the user's account by trial and error (by trying 1000's of potential password strings per minute, in hope of trying a string that matches the actual password). The above are some of the main reasons for the popularity gains of biometrics oriented alternatives for password identification. Here we describe how AKL can be utilized to handle password/identity verification. In what follows, we provide a basic description of the proposed method. Similar to the biometrics oriented idea of uniquely matching a person with his computer account, we agree that characteristics which are *unique* to each person should be used to verify the authenticity of the user. For example, in the biometrics oriented approach, the fingerprint of a person is used as such a unique characteristic. Out approach is what we could call behavior-metrics oriented (as opposed to biometrics oriented). That is, we use behavioral characteristics (as opposed to biological characteristics) for authentication. Similar to our discussion in the Tic-Tac-To example of subsection 3.2, we now describe the event structure and the proposed methodology for the authentication problem in this subsection.

Event structure: As usual, the problem is formulated with K-lines, each K-line consists of K-nodes, and each K-node contains an event. The events of two consecutive K-nodes are related by causality i.e., the event of every K-node K is an immediate consequence of the event of the

K-node which is the immediate predecessor of K. In our setting, we define the event to be the input of a character, as part of the character string that comprises the user's password. Figure 11 shows two consecutive K-nodes K1 and K2 with their events $e^{\langle K1\rangle}$ and $e^{\langle K2\rangle}$.

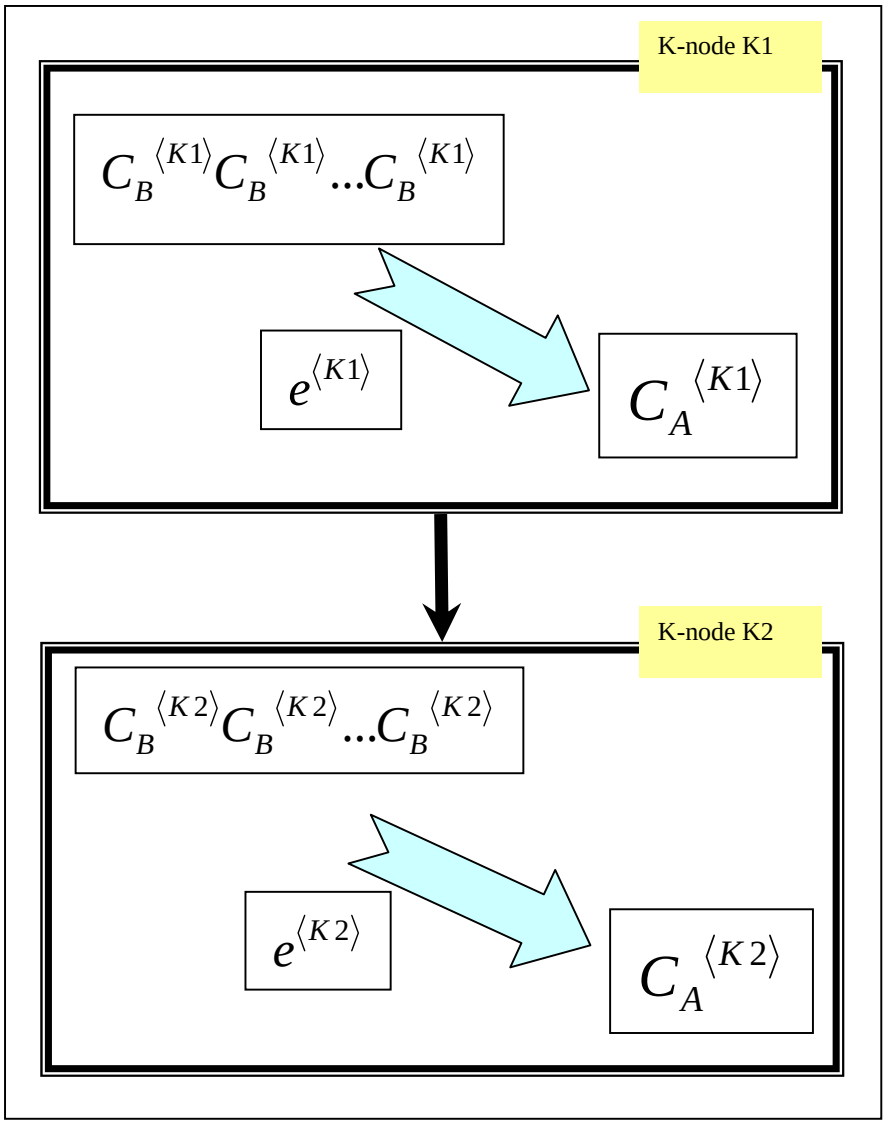

Figure 11. Two consecutive K-nodes for the password authentication problem.

In Figure 11, each K-node K (K = K1 or K2) contains two characters named $C_B^{\langle K\rangle}$ and $C_A^{\langle K\rangle}$ such that character $C_B^{\langle K\rangle}$ is input by the user just before character $C_A^{\langle K\rangle}$, during the process of the user inputting his/her password. As we observe, character $C_B^{\langle K1\rangle}$ is repeated several times in K-node K1, and character $C_B^{\langle K2\rangle}$ is repeated several times in K-node K2. The

reason for this and the number of times that the characters are repeated are explained and discussed next, in the "Methodology" sub-subsection. Note, for the two K-nodes K1 and K2 to be connected so that K1 is the predecessor of K2, it must be that $C_A^{\langle K1\rangle} = C_B^{\langle K2\rangle}$. That is, the input of character $C_A^{\langle K2\rangle}$ must immediately follow the input of character $C_B^{\langle K2\rangle}$. This is the criterion that qualifies events $e^{\langle K1\rangle}$ and $e^{\langle K2\rangle}$ to be associated by causality. We now describe a methodology according to which an AKL is formed so that it captures the behavioral pattern of the user during password input.

Methodology: We assume that a user attempts to gain access to his computer account by typing in his password. The system captures the typing behavior of the user as he types his password. It is assumed that this phase will be repeated for several times so that the system is trained (in the next sub-subsection we describe how a trained system can be used to authenticate a user). We base our method on the assumption that when two people type the same word (or phrase), the *way* that they type that same word differs between the two people. The same claim is made in [10]: "*…the impostor … doesn't type your password the same way you do*". This difference, we argue, is more profound in cases that the word which is typed is very familiar to one person (which is the case for the legitimate owner of a password) and not so familiar to the other person. For example, the author of this paper can type his last name fairly quickly and (usually!) without making any mistakes. However, we argue, if another person types the author's last name, then there will be a noticeable (at least by the system) difference in the speed that the characters of the name are typed, and probably in getting the spelling correct, at least during the first attempt. Based on the above discussion, we use the amount of time that it elapses between inputting consecutive characters during the input of the required password. Algorithm A, below, outlines the interaction between user and system during the input of a user's password.

Algorithm A

Loop

- Step A0: Consider C_B, the most recently entered character.
- Step A1 [System]: `StartClock()` and record time point, t1.
- Step A2 [User]: input character C_A.
- Step A3 [System]: `StopClock()` and record time point, t2.
- Step A4 [System]: `W = t2 – t1`;
- Step A5 [System]: `FormKnode(`C_B`,`C_A`, W );`

End loop.

The main product of Algorithm A is the creation of a K-node, during step A5. Once formed, the newly created K-node is appended to the existing K-line. Figure 12 shows a K-node that is created as a result of the call FormKnode(C_B, C_A, W).

The main ingredients of the K-node shown in Figure 12 are two characters, $C_B^{\langle K\rangle}$, and $C_A^{\langle K\rangle}$, and an event $e^{\langle K\rangle}$ that connects those two characters. Character $C_B^{\langle K\rangle}$ is the "before" character, i.e. the character that has been most recently typed by the user, prior to the typing of character $C_A^{\langle K\rangle}$, the "after" character. Event $e^{\langle K\rangle}$ is the act of typing character $C_A^{\langle K\rangle}$ right after typing character $C_B^{\langle K\rangle}$.

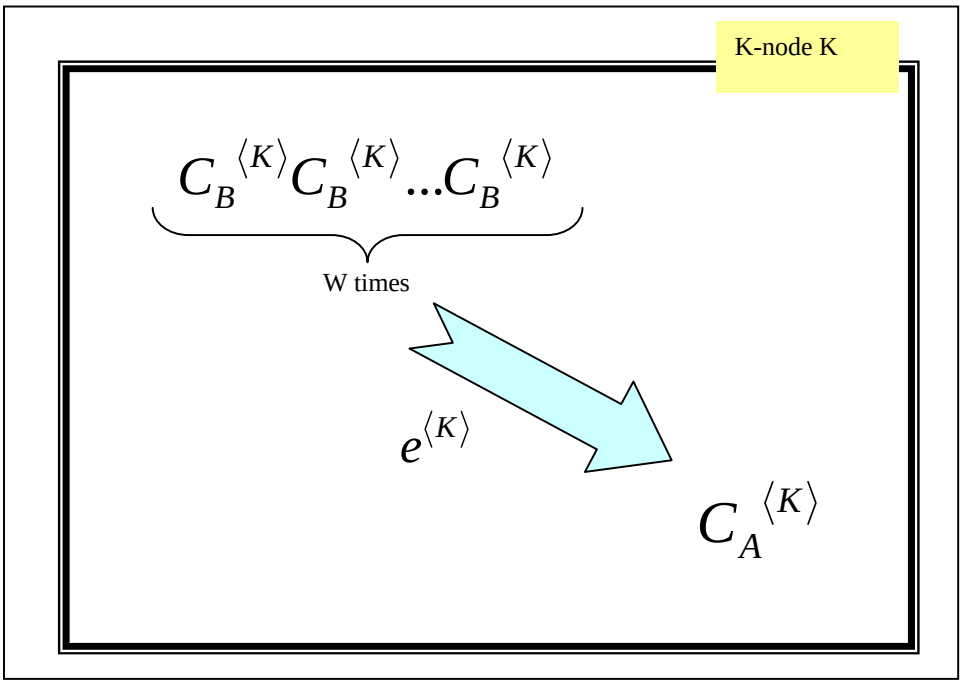

Figure 12. K-node created by Step A5 of Algorithm A.

According to the way that we structure our K-node, character $C_B^{\langle K \rangle}$ appears *multiple* times (W times) in the K-node of Figure 12. This is due to step A4 of Algorithm A, which captures the *delay* of the user in typing character $C_A^{\langle K \rangle}$ after having typed character $C_B^{\langle K \rangle}$. This delay is captured by steps A1 and A3 of Algorithm A. Step A1 is essentially the beginning of Algorithm A and the system is waiting for the user to type a character. At that time, the system starts a clock (call to method StartClock, in step A1) and keeps track of the time elapsed until the user eventually types something (in step A2). As soon as the user types a new character, $C_A^{\langle K \rangle}$, the system stops the clock (call to method StopClock, in step A3) and calculates the elapsed time (step A4). Then, the amount of elapsed time serves as a measurement of how many times character $C_B^{\langle K \rangle}$ will be repeated in the K-node that is about to be formed (in step A5). To convert the amount of elapsed time W into number of times that character $C_B^{\langle K \rangle}$ is repeated, we divide W by a number k such that $k < W$ (and preferably $k << W$) and k is a predetermined system constant and it is the same for all K-nodes and it represents an amount of time much smaller than any amount of time W that is being observed throughout the formation of all K-nodes. The configuration of the K-node as shown in Figure 12 is used to verify/authenticate a user, as follows.

As the user continues typing characters, the system continues forming K-nodes, by invoking Algorithm A for each typed character. Each K-node that is being formed is appended to the *current* K-line. At the end of the formation of the current K-line, the K-line consists of N K-nodes where N is the number of characters that the user typed during his current attempt to gain access to the system. One K-line, L1, is formed as soon as the user finishes typing her password. K-line L1 consists of N K-nodes and each K-node has the structure and configuration shown in Figure 11 (and Figure 12). At this point, we can consider, in principle, the AKL to have been formed and be ready for usage. However, to increase its sensitivity and accuracy, we can consider forming new K-lines, as follows. The

system prompts for the user's password again, and starts forming a new K-line (it is understood that the *same* user – the rightful owner of the account, is used for the formation of this K-line as well). Similar to the process of the formation of K-line L1, the user enters her password again, and the system records the characters that are typed, as well as the delays between consecutive keystrokes, and forms K-nodes by invoking Algorithm A – the same way as done during the formation of K-line L1. However, *in addition* to this, the system also checks how the time delays between typing consecutive characters during the formation of the current K-line, L2 compare with the time delays recorded during the formation of the previous K-line, L1. This is done as follows: as the user enters her password (for second time, for the formation of K-line L2), and K-nodes are formed as in the case of L1, it is assumed that the typed characters are the same for all corresponding K-nodes (i.e., all K-nodes Kj1, Kj2), where Kj1 is the j-th K-node of K-line L1 and Kj2 is the j-th K-node of K-line L2. However, the time delays Dj1 and Dj2 (i.e., the number of occurrences of character C_B of K-node Kj1 and the number of occurrences of character C_B of K-node Kj2) may *vary* slightly! This is expected due to human nature. That is, we do *expect* that the same user exhibits *similar typing behavior* in typing her password during different times, but we *do not require* that she exhibits *identical behavior*. As such, we maintain a *Delay Tolerance Table* (DTT) for each time delay that corresponds to every pair of consecutively typed characters. Figure 13 shows such a tolerance table.

K-node	C_B	**Delay Tolerance**	C_A
K1	$C_B^{\langle K1\rangle}$	$DTL^{\langle K1\rangle}$	$C_A^{\langle K1\rangle}$
K2	$C_B^{\langle K2\rangle} = C_A^{\langle K1\rangle}$	$DTL^{\langle K2\rangle}$	$C_A^{\langle K2\rangle}$
K3	$C_B^{\langle K3\rangle} = C_A^{\langle K2\rangle}$	$DTL^{\langle K3\rangle}$	$C_A^{\langle K3\rangle}$
…	…	…	…
Kj	$C_B^{\langle Kj\rangle} = C_A^{\langle Kj-1\rangle}$	$DTL^{\langle Kj\rangle}$	$C_A^{\langle Kj\rangle}$
…	…	…	…
Kn	$C_B^{\langle Kn\rangle} = C_A^{\langle Kn-1\rangle}$	$DTL^{\langle Kn\rangle}$	$C_A^{\langle Kn\rangle}$

Figure 13. Sample Delay Tolerance Table.

In Figure 13, it is assumed that each K-line has n K-nodes. The values $DTL^{\langle K1\rangle}$, $DTL^{\langle K2\rangle}$, …, $DTL^{\langle Kj\rangle}$, …, $DTL^{\langle Kn\rangle}$ represent intervals

$$I^{\langle Kj\rangle} = \left[\min D^{\langle Kj\rangle}, \max D^{\langle Kj\rangle}\right]$$

where $\min D^{\langle Kj\rangle}$ is the minimum observed delay time for K-node Kj among all the user's corresponding inputs for the same K-node so far, and $\max D^{\langle Kj\rangle}$ is the maximum observed

delay time for K-node Kj among all the user's inputs for K-node Kj, so far. The characteristic of every DTL-value in the Tolerance Table is that for every

$$DTL^{\langle Kj\rangle} = \left[\min D^{\langle Kj\rangle}, \max D^{\langle Kj\rangle}\right],$$

we require that

$$\left|\min D^{\langle Kj\rangle} - \max D^{\langle Kj\rangle}\right| \le \varepsilon$$

for a small value ε, determined by the system. The idea is that if the user's typing pattern exhibits varying degrees of delay for the same 2-character sequence, then we require that those differences in delays are within an acceptable restricted threshold. When a new K-line L_Y is formed and one of its K-nodes K_{Yj} has a delay time $D^{\langle K_{Yj}\rangle}$ which is *close enough* to a delay time $D^{\langle K_{Xj}\rangle}$ of the same-rank K-node K_{Xj} of another K-line L_X, then we consider K-lines L_X and L_Y to *intersect*. In such a case, K-nodes K_{Xj} and K_{Yj} are combined to form a new K-node $K_{(XY)j}$ and this new K-node is considered to be at the intersection of the two K-lines L_X and L_Y. The delay time $D^{\langle K_{(XY)j}\rangle}$ of the new K-node is set to

$$D^{\langle K_{(XY)j}\rangle} = \frac{D^{\langle K_{Xj}\rangle} + D^{\langle K_{Yj}\rangle}}{2}$$

i.e, the average of the delay times of the two amalgamated K-nodes K_{Xj} and K_{Yj}; and the delay tolerance DTL of the new K-node is set to

$$DTL^{\langle K_{(XY)j}\rangle} = \left[\frac{\min D^{\langle K_{Xj}\rangle} + \min D^{\langle K_{Yj}\rangle}}{2}, \frac{\max D^{\langle K_{Xj}\rangle} + \max D^{\langle K_{Yj}\rangle}}{2}\right],$$

i.e., we set the delay tolerance interval of the new K-node to an interval whose boundaries are the *averages* of the boundaries of the delay tolerance intervals of the two combined K-nodes K_{Xj} and K_{Yj}. Also, the number $W^{\langle K_{(XY)j}\rangle}$ of repetitions for the "before" character of the new K-node $K_{(XY)j}$ is set to

$$W^{\langle K_{(XY)j}\rangle} = \frac{W^{\langle K_X\rangle} + W^{\langle K_Y\rangle}}{2}$$

where $W^{\langle K_X\rangle}$ and $W^{\langle K_Y\rangle}$ are the number of repetitions of the two "before" characters of the two combined K-nodes K_{Xj} and K_{Yj}, respectively. Note, the above process of the formation of the AKL may result into an AKL that consists of multiple K-lines. It is expected that those K-lines intersect at many of their K-nodes and, in fact, many of them intersect at *all* of their K-nodes. This is normal because, after all, a user is expected to have a pretty stable typing behavior when typing her password at different times. However, there may be also some K-lines for which some of their K-nodes have not been found to be close enough (per the criterion of the delay tolerance table, as described above). Those K-nodes are not at the intersection of any K-lines. Such cases arise for any two two K-nodes K_{Xj} and K_{Yj} of the same rank j and belonging to two different K-lines L_X and L_Y, and for which it has been found that

$$\left| W^{\langle K_{Xj} \rangle} - W^{\langle K_{Yj} \rangle} \right| > \varepsilon .$$

In effect, this means that the AKL contains two K-lines each of which represents a different typing pattern of the *same* user, while that user enters the *same* password at different times! At first, this may seem somewhat peculiar. We argue that this is possible and in fact it is probably a desirable feature of the AKL. In other words, we welcome the case that the same user can exhibit different typing patterns when typing her password during different times. The reason for that is that the same user may be at different *states of mind* during different times. For example, when the user is tired, she may exhibit different typing behavior from when she is well-rested. Or, when the user is under pressure of time (e.g., a deadline) she tends to exhibit yet a different typing behavior; and so on. We are not aware of any research of how the state of mind of a person may affect his/her typing behavior and it our intention to investigate this issue and report our findings in a sequel paper. In some of our previous works (e.g., [7], [11], [12]) we found evidence that the affective state of a person influences his/her performance in the area of media handling. It will not be surprising if the same is true for the case of identity authentication.

User Authentication. Once the AKL has been formed – as described above, it can be used to authenticate a user, as follows. A user arrives and types her password. As soon as the user types the first character, the *closest* K-line is selected from the AKL. Assuming that the first typed character is the same as the first character of all K-nodes of all K-lines, the closest K-line is a K-line L such that the first K-node of L has a W-value that is within the acceptable tolerance threshold ε from the W-value of the K-node that would have been formed out of the user's first typed character. Note, if there is no matching K-line (either because the typed character does not match the character of any first K-node of any of the K-lines, or because the created W-value does not match any of the W-values of the first K-node of any of the K-lines) then the user's attempt is rejected and the user is not granted access to the system. Provided that the matching succeeds, assume that the K-line against which the password is matched consists of n K-nodes, K1, K2, …, Kn. Denote by $C_A^{\langle K1 \rangle}, C_A^{\langle K2 \rangle}, ..., C_A^{\langle Kn \rangle}$ the "after" characters of those K-nodes, and by $DTL^{\langle K1 \rangle}, DTL^{\langle K2 \rangle}, ..., DTL^{\langle Kn \rangle}$ the corresponding delay tolerances of the Tolerance Table of that K-line. Assume that the password input by the user is the string $C_1, C_2, ..., C_n$ with corresponding delays $D_1, D_2, ..., D_n$, should K-nodes were to be formed out of that password. From the recorded delays $D_1, D_2, ..., D_n$ calculate the W-values the same way that they would have been calculated if K-nodes were to be formed from the supplied input password. Let $W_1, W_2, ..., W_n$ are the calculated W-values. Then, the system grants access to the user (i.e., the system considers the supplied password to be valid), if and only if

$$Cj \equiv C_A^{\langle Kj \rangle}, \text{ for all j=1, …, N } \textbf{and } Dj \subseteq DTL^{\langle Kj \rangle} \text{ (or, equivalently, } W_j \le W^{\langle Kj \rangle}\text{),}$$
$$\text{for all j = 1, …, N.}$$

That is, the system grants access to the user if and only if there is a complete match, character-by-character, in the characters of the supplied password and the one that is being stored in the K-line, *and also*, if the typing pattern of the supplied password resembles the typing behavioral pattern of the user as it has been recorded in the matching K-line from previous experiences.

Why is this way of authentication interesting? The proposed method for user password authentication has the following interesting characteristics.

1. It provides added security which comes from string matching (the traditional password verification method in current systems) *and* from matching of behavioral characteristics of the user.
2. It can be used potentially for user training. In addition to having the user training the system to understand the user's typing pattern as described above (during the discussion of the formation of the Tolerance Table), the opposite may be possible as well, i.e., having the system to train a user to adopt (or, conform to) a certain behavior, by "insisting" that the user adopts a certain typing pattern (and rejecting typing pattern variations that are outside the tolerance values of the tolerance table. Note, the tolerance table in this case may be set from past experiences, or may be created artificially by the system, if the purpose is to (re-)train a user to a completely new typing pattern). Such a mode of operation of the system could be used stand alone to train a user to a completely new typing pattern, or it can be used in conjunction with the "normal" mode to fine-tune an existing system during the phase that the user enters his password and the system forms the delay tolerance table. In case that the user's observed delays are too "liberal" (or, too "erratic"), i.e., $\left| D^{\langle K_{Lj} \rangle} - D^{\langle K_{Mj} \rangle} \right| > \varepsilon$ for two K-nodes of the same rank, j, at different K-lines, L and M.
3. It is potentially more secure than biometrics oriented authentication. Note, although biometrics guarantee 100% uniqueness, they are still vulnerable to theft (for example, in the extreme scenario that a thief makes a mold of a fingerprint, then the password is both stolen and, moreover, there is little hope for the rightful owner to change that password!). However, even though that could be somewhat difficult, any user can change his/her typing behavior! Therefore, if someone steals your password together with your current typing pattern (we consider the latter to be very difficult) you can still protect your access privilege to the system by altering your typing behavior (and, of course, retraining the system to recognize the new behavior).
4. Additional behavioral attributes can be added, besides the time delay approach that is outlined in this paper, as modern computer systems become more sensitive to various behavioral characteristics. For example, the angle that a user tends to hit a particular key while typing his password, or the amount of skidding that a user's finger tends to make on the typing surface, could be some of those behavioral attributes. Note, certain input devises (e.g., the iPhone) are already sensitive to such behavioral characteristics. For example, the iPhone can recognize how much a user slides her finger while touching the screen.
5. Simplicity. The proposed method is easy to implement and it does not require any additional or new hardware (such as fingerprint readers, or iris scanners), which is required by most/all biometrics based methods.

Related work. Typing behavior is an almost century-old issue, identified since at least as early as the first quarter of the previous century [13], [14], and there is a fair amount of research on the issue of typing behavior and of password verification based on typing behavior. See for example, [15], [16], [17], [18], [19], [20], [21], [22], [23], and [24]. All previous works identify typing behavior attributes and patterns and/or devise strategies for user authentication. The preferred attribute typing behavior that is used is the time duration between keystrokes (e.g., in [15], [16], [17], [24], [18]. Without claiming that this is the only

typing behavior characteristic, we use the same attribute in the description of our method here. The preferred strategy for password authentication is by means of neural network (NN). Several flavors of NN are explored, such as the ADALINE Neural Element in [19], the Backpropagation NN in [19], [21], [18], the Kohonen NN in [19], the Counterpropagation NN in [21], the Fuzzy ARTMAP in [21], the Radial Basis Network in [21], the Learning Vector Quantization NN in [21], the Sum-Of-Product NN in [21], [18], and the Hybrid-Sum-Of-Product NN in [21], [18]. Interestingly, the most recent password authentication work that we are aware of is a commercial product from a recent startup company named Delfigo [28]. As reported in [10], the design of this product is inspired by studies that one of the company's co-founders recently completed at MIT during studies with professor M. Minsky. It has been also stated in [29] that the company's identity identification algorithm employs NN technology. (The details of the NN mechanism are, of course, Delfigo's intellectual property and secrets and, as such, are not disclosed to the general public).

6. Comparison with Neural Networks

We provide a general comparison between the essence of AKL and ANN. Note, this is meant to be only a very general, although accurate, contrast between the proposed structure, AKL, and the long-existing well-known structure of ANN. This comparison has been offered also in our most recent work in [8] and [9].

ANN is a structure that aids in decision making. After training, it answers, essentially, "yes"-"no" questions, upon presentation of input for a task. The task has to be within the domain that the ANN has been trained. That is, the ANN does not have general knowledge, neither does it have the ability for creativity and for combining multi-domain knowledge. In this sense, ANN is a *specialist (rather than a polymath) that continuously hones its skill to perform a certain task, without any creative abilities*. An ANN may increase its knowledge, but the newly acquired knowledge is *strictly confined* within the boundaries of performing better the *same* task that it used to perform before.

AKL is a structure that accommodates learning, but no decision making. An AKL does not point to a single piece of knowledge; on the contrary, it spans *several domains* of knowledge and this knowledge is expanded. The expansion is done with the incorporation of new K-lines into the existing AKL graph, regardless of the knowledge domain from where those K-lines come from. By incorporating more K-lines, AKL builds on existing knowledge and expands its ability of many different alternative "ways to think". As such, it facilitates creativity (i.e., allowing the generation of new "ways to think"), by allowing the formation of paths comprised from edges from *different* K-lines (as for example, cases (a) and (d) of Figure 4). The number of new "ways to think" is, in most cases, significantly greater than the number of the original "ways to think" that were used to form the AKL graph due to the periodic fun-outs that we encounter at K-line intersections. In this sense, AKL is the *creative polymath that continuously expands its knowledge and increases its chances for creative thinking*.

7. Conclusion

We present AKL, a novel approach in how to use K-lines. We illustrate via three applications drawn from widely diverse domains, how AKL can solve some problems. In section 4, we illustrate how the proposed method can solve a problem that involves reflective thinking, that no other known method can solve as easily. In section 5, we use a simple board game to illustrate how AKL can be used to facilitate machine learning.

In section 5, we use a simple scenario of computer security to illustrate how AKL can be used to facilitate identity authentication. It is hoped that the proposed structure is a potential tool for building intelligent systems, complementary to ANN.

Our future plans include refining the proposed method and investigating its potential for *machine learning* (such as the TTT example outlined in section 4), for *user authentication* as outlined in section 5, and for *artificial creativity*. For *machine learning*, our immediate plan is to formulate a way to *automate* the testing of the TTT example described earlier. Note, testing of our methodology as described in section 4 is possible, but it is cumbersome since it relies on continuous human participation. Since it is desirable that there are a *large* number of moves and games played in order for the AKL to acquire a *substantial* amount of knowledge that can be used by the machine, it is impractical to utilize an actual human for this purpose. Therefore, the input of a human participant has to be automated. Fortunately, for the TTT game outlined here, the *entire* knowledge space can be generated since this particular game is fairly small. We are currently working on formulating the problem of how this space can be utilized to automate the input of a human expert. In addition to the game of Tic-Tac-To, it is interesting to formulate the automation of human expert input for more complex games, such as chess and go. For user authentication, our plan is to (a) formulate a way to automatically test the method described in section 5 and (b) to enhance our method with additional behavior based attributes, besides the successive keystroke elapsed time attribute. We believe that task (a) can be performed by generating the same password a large number of times and simulating a user's several keystroke delay patterns which are then used to form an AKL. We believe that the main difficulty is in finding an optimal ε-value for the DTL values. Needless to say, due to the nature of the problem (computer security is a very sensitive issue nowadays) the results will have to be near-perfect, or else the method can be deemed as impractical or undesirable. The neural network based approaches to the same problem that are referenced earlier in this paper (section 5) report user authentication success rates of well-above 90%. Task (b) is, perhaps, more interesting and more useful in practice, albeit more intellectually challenging. A main problem in this context is how to incorporate the many behavioral attributes that affect the typing pattern of a user into a K-node and in particular to express them as events. In the method described in section 5, we have only one attribute – the time elapsed between two successive keystrokes, and that is mapped fairly naturally to an event by converting the elapsed time to a string of repeated characters. When, however, the number of affective behavioral attributes is greater than one, such a mapping is not obvious. We are in the process of developing an appropriate structure, similar to AKL, which can hopefully address the problem. Details of our work will be reported in a sequel paper. Another interesting research direction as an application of AKL is the area of *artificial creativity*. Note, an inherent characteristic of AKL is to combine parts of K-lines and form new K-lines comprised of those parts. We argue that this is the essence of creativity, that is, the ability to form new "ideas" by using existing knowledge, or by combining (parts of) old ideas. In the context of the AKL, the newly formed K-lines represent the new ideas, whereas the existing knowledge is the sequence of segments of the existing K-lines that are used to form the new K-lines. In this sense, the AKL might be a suitable candidate structure for artificial creativity. One major obvious issue is how to judge the suitability of the newly generated K-lines, i.e., how to zero-in into *only* the meaningful generated ideas among all generated ideas, since the latter can be too many and, among those, many/most of them might not make sense. Note, again, the same phenomenon is encountered with natural creativity. That is, for

every new idea that is generated by a person, there is always the issue of how sensible (or applicable) that idea is. We will be delighted if the AKL can shed some light to the artificial creativity problem, which is a many-decades old and unsolved problem in Artificial Intelligence.

References

[1] M. Minsky, "K-Lines: A Theory of Memory", Cognitive Science, 4, 1980, pp. 117-l33.
[2] M. Minsky, The Society of Mind, Simon & Schuster, 1986.
[3] M. Minsky, The Emotion Machine, Simon & Schuster, 2006.
[4] A. Sloman, GRAND CHALLENGE 5: The Architecture of Brain and Mind: Integrating Low-Level Neuronal Brain Processes with High-Level Cognitive Behaviours in a Functioning Robot, Technical Report COSY-TR-0607, July 2006. http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0607
[5] P. Singh, "Failure Directed Reformulation", M. Eng. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, March 10, (1998).
[6] P. Singh, "EM-ONE: An Architecture for Reflective Commonsense Thinking", Ph.D. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2005.
[7] A. A. Toptsis and A. Dubitski, "Iterative K-line Meshing via Non-Linear Least Squares Interpolation of Affectively Decorated Media Repositories", The Open Artificial Intelligence Journal, Vol. 2, 2008, pp. 46-61.
[8] A. A. Toptsis, and A. Dubitski, "Artificial K-lines", Proc. Advanced Science and Technology 2009 (AST2009), IEEE CS Press., 2009, (in press).
[9] A. A. Toptsis and A. Dubitski, "Artificial K-lines and Applications", Proc. Advanced Science and Technology 2009 (AST2009), Communications in Computer and Information Science (CCIS), Springer-Verlag, (in press).
[10] The journal of New Englad Technology, "Delfigo thwarts hackers with typing behavior security", http://www.masshightech.com/stories/2009/02/09/weekly5-Delfigo-thwarts-hackers-with-typing-behavior-security.html (last accessed April 11, 2009).
[11] A.A. Toptsis and A. Dubitski, "Affective Space Calibration in Action-rich Media Affective Systems", Journal of Convergence Information Technology (JCIT), Vol. 3, No. 4, 2008, pp. 3-14.
[12] A. A. Toptsis and A. Dubitski, "On the Affective Bias of Emotional and Physiological States During Handling Media Content in K-line Mesh-Driven Repositories", Journal of Digital Content Technology and its Applications (JDCTA), Vol. 3, No. 1, March 2009, pp. 4-15.
[13] J. E. Coover, "A Method of Teaching Typewriting Based on a Psychological Analysis of Expert Typing", National Education Association, Addresses and Proceedings, Vol. 61, 1923, pp. 561-567.
[14] A. Dvorak, N.L. Merrick, W.L. Dealev, and G.C. Ford, "Typewriting Behavior", American Books, New York, 1936.
[15] D. Umphress and G. Williams, "Identity verification through keyboard characteristics", Int. J. Man-Machine Studies, Vol. 23, 1985, pp. 263-273.
[16] J. Leggett and G. Williams, "Verifying identity via keystroke characteristics", Int.. J. Man-Machine Studies, Vol. 28, 1988, pp. 67-76.
[17] S. Bleha, C. Slivinsky, and B. Hussien, "Computer-access security systems using keystroke dynamics", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 12, December 1990, pp. 1217-1222.
[18] M. S. Obaidat and D.T. Macchiarolo, "An online neural network system for computer access security", IEEE Transactions on Industrial Electronics, Vol. 40, No. 2, April 1993, pp. 235-242.
[19] M. Brown and S.J. Rogers, "User identification via keystroke characteristics of typed names using neural networks", Int.. J. Man-Machine Studies, Vol. 39, 1993, pp. 999-1014.
[20] J. Leggett, G. Williams, and M. Usnick, "Dynamic identity verification via keystroke characteristics", Int.. J. Man-Machine Studies, Vol. 35, 1991, pp. 859-870.
[21] M. S. Obaidat and B. Sadoun, "Verification of Computer Users Using Keystroke Dynamics", IEEE Transactions on Systems, Man and Cybernetics, Part B Cybernetics, Vol. 27, No. 2, April 1997, pp. 261-269.
[22] F. Berganado, D. Gunetti, and C. Picardi, "User Authentication through Keystroke Dynamics", ACM Transactions on Information and System Security, Vol. 5, No. 4, November 2002, pp. 367–397.
[23] D. Gunetti and C. Picardi, "Keystroke Analysis of Free Text", ACM Transactions on Information and System Security, Vol. 8, No. 3, August 2005, Pages 312–347.
[24] R. Stockton Gaines, W. Lisowski; S. James Press; N. Shapiro, "Authentication by Keystroke Timing: Some Preliminary Results", RAND Report R-2526-NSF, Santa Monica, CA, May 1980.

[25] D. Kumar, Y. Ryu; and D. Kwon, "A survey on biometric fingerprints: The cardless payment system", IEEE- International symposium on Biometrics & security technologies, ISBAST '08, Islamabad, April 23–24, 2008, pp.1-6.
[26] A.K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, Issue 1, January 2004, pp. 4 – 20.
[27] D. Kumar and Y. Ryu, "A Brief Introduction of Biometrics and Fingerprint Payment Technology", International Journal of Advanced Science and Technology, Vol. 4, March, 2009, pp. 25-38.
[28] Delfigo Home Site, http://www.delfigosecurity.com/wwwdelfigosecuritycom/. (last accessed April 4, 2009).
[29] Delfigo Products, "DSGateway: Context-based Identity Assurance", http://www.delfigosecurity.com/products, (last accessed April 14, 2009).
[30] H. Poincaré , "Creativity: Building the New", in The Foundations of Science, Paris, 1908. Available at http://www.is.wayne.edu/drbowen/crtvyw99/poincare.htm (last accesses April 14, 2009).
[31] G. Polya, How to Solve It: A New Aspect of Mathematical Method, Princeton University Press, 1945.
[32] M. Csikszentmihalyi, Creativity : Flow and the Psychology of Discovery and Invention, Harpercollins, 1996.
[33] A. Snyder, D. Mitchell, T. Bossomaier, and G. Pallier, "The Creativity Quotient, an Objective Scoring of Ideational Fluency", Creativity Research Journal, Vol. 16, No. 4, 2004, pp. 415-420.
[34] L.L. Thurstone, The nature of intelligence, 1923. Newer edition published by Routledge, 1999.
[35] T. Bossomaier, M. Harre, A. Knittel, and A. Snyder, "A Semantic Network Approach to the Creativity Quotient (CQ)", Creativity Research Journal, Vol. 21, No. 1, January 2009, pp. 64 – 71.
[36] J.W. Getzels, and P.W. Jackson, Creativity and intelligence, PW Jackson, 1962.
[37] J. Guilford, Personality, McGraw Hill, 1959.