



GRANDON GILL, BERNARDO RODRIGUES

EMPLOYING DYNAMIC LOGIC IN CYBERSECURITY¹

The dynamic logic algorithm has already demonstrated its ability to identify malware in network traffic. What other opportunities does the emerging cybersecurity space offer for applications?

Dr. Leonid Perlovsky, distinguished physicist and cognitive scientist, pondered this question, which could have a significant impact on his research direction in the years to come. Over the past few decades, he had developed and refined algorithms for distinguishing objects in images, an approach that had found its way into various classified U.S. Department of Defense (DoD) applications. Now he was looking for new potential opportunities to see his research applied, allowing it to evolve further.

One of the most interesting aspects of Perlovsky's approach was that it was very similar to that taken by the human brain in processing sensory information. It began with a very vague model of what might or might not be present in the data being examined. Through successive iterations, analogous to the layers of processing used in human sensory systems, the patterns in the data corresponding to objects would grow more and more distinct until, finally, they became recognizable. Unlike most statistical techniques, this approach—termed “dynamic logic” by Perlovsky—did not require that a model be specified in advance. As such, it was well suited for contexts that required discovery.

One application of dynamic logic that particularly impressed him involved the detection of malware in network packet data. Using an externally provided database of this traffic, his algorithm had successfully identified the presence of malware with almost eerie precision, and with substantially less processing than competing techniques. This suggested that dynamic logic could well become a powerful tool in the arsenal of IT professionals seeking to protect their systems from hackers. What other possible cybersecurity-related opportunities might be well suited to this tool?

Identifying potential opportunities represented only part of the challenge of putting dynamic logic to work. After letting the project lay dormant for several years, he had recently been approached by an energetic Brazilian master's student who had identified ways that DL (dynamic logic) could be used. The student had also established a DL open source project on his own initiative. If that project were to move forward, Perlovsky would need to provide some encouragement and guidance. But he had his own set of questions. Was the open source path the right way to proceed? What potential application should be given highest priority? Should government or commercial funding be pursued? And the big question... Perlovsky readily acknowledged that he was no cybersecurity expert. Given that he was already actively pursuing grants from the DoD and National Institute of Health (NIH), would it really make sense to split his attention further, and look towards tackling an entirely new class of problems?

¹ Copyright © 2017, *Grandon Gill and Bernardo Rodrigues*. This case was prepared for the purpose of class discussion, and not to illustrate the effective or ineffective handling of an administrative situation. This case is published under a Creative Commons BY-NC license. Permission is granted to copy and distribute this case for non-commercial purposes, in both printed and electronic formats.

Leonid Perlovsky

Dr. Leonid Perlovsky was born in Odessa, a resort and port city on the Black Sea, USSR, which later became part of the Ukraine when the Soviet Union broke up. He graduated from Novosibirsk University, where he served as a Professor of Physics. In 1978, he emigrated to the U.S., where he embarked on a stellar and varied academic career that crossed numerous disciplinary lines.

A particular area of study that proved to be of greatest interest to Perlovsky involved the application of cognitive principles to the analysis of problems that were too hard to attack using conventional algorithms. In the course of this research, he created a new area of cognitive mathematical engineering, dynamic logic (discussed in the next section), which modeled the mind's processes, and applied them to a number of problems in engineering and cognitive science that had, for decades, been unsolvable.

As a consequence of the breadth of his interests, Perlovsky's research career had followed a trajectory far different from that of the traditional academic—who might remain at a single institution for decades, or even an entire career. Among his past affiliations were the following:

- Professor at Novosibirsk University and New York University
- Visiting Scholar at Harvard School of Engineering and Applied Sciences
- Researcher at Harvard Medical School's Athinoula Martinos Brain Imaging Center
- Chief Scientist at Nichols Research, a \$500mm high-tech DOD contractor
- Technical Advisor and Principal Research Scientist at the AF Research Lab
- Principal in several commercial startups involved in developing tools for text understanding, biotechnology, and financial predictions

At the time of the case, he was a visiting scholar at Northeastern University, where he was working with faculty to develop grant proposals for future research projects.

Perlovsky's research career had been extraordinarily productive, and was marked by a large number of accomplishments. He had been invited as a keynote plenary speaker and tutorial lecturer across the globe, including most prestigious venues such as the Nobel Forum at Karolinska Institutet. His publications included more than 500 papers, 17 book chapters, and 4 books with Oxford and Springer, both well-known academic publishers. He has also been awarded 2 U.S. patents.

Perlovsky was particularly proud of his work in integrating two of his research passions, physics and cognition. He was the founder and continued to serve as Editor-in-Chief for *Physics of Life Reviews*. Established in 2004, the journal had achieved an impact factor (IF)—a measure of how often articles in the journal are typically cited by other researchers shortly after they are published—of 9.5. (By way of comparison, this was higher than any academic journal in business, which rarely achieved IF levels higher than 6).

Much of Perlovsky's recent research focused on areas far outside of the usual domain of a physicist. He led research projects on mathematical models of the mind, emotionality of languages and cultures, cognitive functions of emotions in language and music, and models of cultures. A number of predictions of these models have been experimentally confirmed. And all were driven by the hierarchy of abstract models that was at the core of dynamic logic.

Dynamic Logic

For at least as long as there have been philosophers, individuals have been intrigued by how the mind works. For example, the early Greek philosophers developed perspectives that still hold considerable validity today. Perlovsky explained (Perlovsky & Ilin, 2010, p. 4):

The relationships between logic, cognition, and language have been a source of longstanding controversy. The widely accepted story is that Aristotle founded logic as a fundamental mind mechanism, and only during the recent decades science overcame this influence. I would like to emphasize the opposite side of this story. Aristotle assumed a close relationship between logic and language. He emphasized that logical statements should not be formulated too strictly and language inherently contains the necessary degree of precision. According to Aristotle, logic serves to communicate already made decisions... The mechanism of the mind relating language, cognition, and the world Aristotle described as forms. Today we call similar mechanisms mental representations, or concepts, or simulators in the mind. Aristotelian forms are similar to Plato's ideas with a marked distinction, forms are dynamic: their initial states, before learning, are different from their final states of concepts... Aristotle emphasized that initial states of forms, forms-as-potentialities, are not logical (i.e., vague), but their final forms, forms-as-actualities, attained in the result of learning, are logical. This fundamental idea was lost during millennia of philosophical arguments.

Dynamic Logic Approach

This notion of “vague to crisp” was central to dynamic logic (DL). In traditional logic-based approaches to object identification in the presence of noise, you typically began with a set of specific alternative models (only one of which could be “true”) that must be mapped to a signal containing a large number of points, any of which could potentially contain an object. To find a best fit, you needed to consider all possible combinations of object configurations (e.g., size, shape, position, orientation) and map these to each of the possible models (e.g., 1 object models, 2 object models, 3 object models, etc.) in order to determine which model was the best fit. This produces what was called a combinatorial explosion or computational complexity (CC) that exceeded the potential of any known computer system. It was sometimes referred to as a bottom up (BU) process, since it started with very specific lower level data (e.g., the signal) and matched it to very specific models. Top down (TD) processes, in contrast, started with an assumed model and then matched it to the assumed data. These were much simpler, but were severely limited by the quality of the initial assumption—since an initial model that did not match the underlying reality would produce a poor result no matter how much mathematical optimization was performed.

The human mind, in contrast to logical approaches, has developed techniques for effectively leveraging both BU and TD approaches in processing signals, such as visual input from the retina. The trick, the same one employed in DL, was to begin with very vague (sketchy) models and process them in parallel, performing processing that caused good fits to become successively more crisp over time. Perlovsky gave the following example (2016c, p. 290-291):

Everyone can conduct a simple 1/2 minute experiment to glimpse into neural mechanisms of representations and BU–TD signal interactions. Look at an object in front of your eyes. Then close your eyes and imagine this object. The imagined object is not as clear and crisp as the same object with opened eyes. It is known that visual imaginations are produced by TD signals,

projecting representations to the visual cortex. Vagueness of the imagined object testifies to the vagueness of its representation. Thus the fundamental DL prediction is experimentally confirmed: representations are vague.

When you open your eyes, the object perception becomes crisp in all its details. This seems to occur momentarily, but this is an illusion of consciousness. Actually the process “from vague to crisp” takes quite long by neuronal measures, 0.6s, hundreds to thousands of neuronal interactions. But our consciousness works in such a way that we are sure, there is no “vague to crisp” process. We are not conscious of this process, and usually we are not conscious about vague initial states either. This prediction of DL has been confirmed in brain imaging experiments. These authors demonstrated that indeed initial representations are vague and usually unconscious. Also the vague to crisp process is not accessible to consciousness. Indeed Aristotle’s formulation of cognition as a process from vague potentialities to logical actualities was ahead of his time.

In the DL approach, vague alternative representations (models) were mapped against a source input signal and similarity scores were computed over time. Each model then effectively competed with other alternative models for the highest similarity score, a dynamic process that naturally placed the greatest weight on those models that exhibited the highest similarity. As the process converges, the highest similarity models made the greatest contribution to an overall score. The process eventually converges when the overall score could not be further increased, at which point (ideally) a single best-fitting model could be identified.

Although the mathematics of the DL algorithm were beyond the scope of the case, its ability to recognize complex patterns in the presence of noise was prodigious. Exhibit 1 shows the results of a simulation where an image that consisted of three crescents (image a) was obscured by random noise to the point where it was completely invisible to the human eye (image b). In its initial passes (images c & d), a single blob model was the best fit. As the model attempts to converge, however, a three blob model becomes a better fit (images e & f). Further convergence finds crescents to be a better fit (image g) and by the time the process ceases to converge further (image h) it has recreated the initial pattern almost perfectly.

What was particularly significant about this illustration was the obvious fact that the algorithm performed even better than the human visual systems—which was an extraordinary information processor in its own right. The obvious implication was that DL could potentially be used to detect patterns that would stymie even human analysts.

Dynamic Logic and Similar Object Identification

The potential application of DL extended far beyond object detection in images. Another type of problem to which it could be applied was the identification of related objects in a large pool of objects that did not exhibit any particular pattern. This type of problem frequently occurred in big data settings where the objective was to extract useful patterns out of a sea of data that was too large to study using traditional database search techniques. Of particular interest here was starting with TD models that were sufficiently vague that when they became crisp they provided insights into relationships that were previously unknown to the researcher.

Exhibit 2 provides an example of this process in action. On the left-hand side, 16,000 observations made in sequence were displayed in temporal sequence (horizontal axis). Each observation could contain any combination of 1000 possible objects (vertical axis). The challenge was to identify relationships between objects. A variation of the DL algorithm was used to sort the observations with the objective of maximizing the similarity of adjacent observations. Displayed in this way, adjacent observations

containing the same object started to form a line. Where parallel lines appeared in the same set of adjacent observations, a relationship between two or more objects has been detected. This was shown on the right-hand figure.

Application of Dynamic Logic

Perlovsky's algorithms have been adapted for use in a number of classified defense systems, primarily for the purpose of image identification and tracking. As a result, he has received the John McLucas Award, the highest US Air Force Award for basic research. It had also produced tactical and strategic breakthroughs endorsed by General Officers, by DepSecDef Hon John Young, and by SecDef Hon Ash Carter. But Perlovsky felt that DL had potential applications that were far beyond tracking systems. For example, his company has predicted the 9/11 market crash a week before the event and supported the subsequent SEC investigation. He was, however, always looking for novel settings in which the DL approach might prove effective.

Broadly speaking, the DL algorithm appeared to be most applicable in settings where the goal was to:

- Identify the presence of an unknown number of vaguely described objects in data sets containing a substantial amount of noise
- Identify previously unsuspected relationships between objects in datasets containing many observations that did not involve the relationship

In both these cases, the goal was to substantially exceed the human's ability to detect the same objects or patterns. DL might also be applied to detect the absence of objects or patterns.

Dynamic Logic and Malware Detection

Several years previously, Perlovsky encountered a specific problem where it appeared that DL might have a practical application. The specific task involved the detection of the presence of malware in network traffic. In a conference paper, Perlovsky and his co-author, described the problem as follows (Perlovsky & Shevchenko, 2014, p. 4056):

The approach to detecting novel attacks is anomaly detection: we develop algorithms learning models of attack-free traffic, and then detect deviations identifying malware. Gradual learning is a fundamental aspect of this approach. We begin assuming that an adequate protection system exists, and we can learn characteristics-models of attack-free traffic. The developed algorithms learn evolution of the malware as it attempts to hide its harmful nature. For the success of this approach, learning of the defensive system must be faster than evolution of the threat.

The defensive system learns to recognize threats as combinations of basic elements, words or n-grams. In principle, this is a most general and universal approach, potentially capable of recognizing any threat. The difficulty of realizing this universal potential is computational complexity and slow learning of most existing algorithms. The reason for these difficulties is fundamental: the number of combinations is very large, even relatively few n-grams can be used to form a very large number of combinations. The number of combinations of only 100 n-grams is 100^{100} , this number exceeds all interactions of all elementary particles in the universe during its entire lifetime. Therefore, even if the entire universe could be made to learn combinations of n-grams, it will not be able to perform its job fast enough.

Perlovsky tested DL on this problem using a publically available database of 119,610 observations characterizing internet protocol (IP) packet data that included both attack free packets and packets generated by different types of malware. The descriptions for each observation consisted of 41 different attributes, each of which might or might not be present in a particular pattern. The goal was to distinguish malware-generated packets from normal traffic and, ideally, to classify different sources of malware.

The results of the test, shown in Exhibit 3, were quite remarkable. Prior to its application, the packet attribute display (not shown) looked much like the left hand side of the earlier Exhibit 2, which is to say random to the human observer. After being sorted based upon the DL assessment of attribute similarity across observations, the malware packets could be readily identified visually. Examined from left to right, a series of lines first mark normal message traffic (67343 observations, more than half the image). Following that, a series of much shorter sets of common parallel lines that indicated (from left to right) the presence of:

- Portsweep malware (2931 observations)
- Warezclient malware (890 observations)
- Satan malware (3633 observations)
- Neptune malware (41214 observations)
- Ipsweep malware (3599 observations, at the far right of the image)

Naturally, as far as DL was concerned, these were just similar patterns—internally, it had no concept of what constituted “malware”. It simply observed similarities between the attributes of the message traffic generated from a particular source, which happened to be malware. What was significant about this achievement, however, was the following:

- A. It was more effective at separating out malware than any other algorithm that had been tried on the particular data set.
- B. The algorithm operated very quickly, meaning it could be applied on a basis that was very close to real time. This was significant because malware developers were known to modify their code immediately once a technique for detecting its presence was identified by security personnel.
- C. Once a baseline for normal traffic was established, the algorithm could be used to monitor traffic and, Perlovsky conjectured, it could identify anomalies that might be the result of new malware sources relatively quickly.

The success of applying DL to the network traffic test data set showed great promise. But it also left Perlovsky with two important unanswered questions:

1. What steps could he take to move the application of DL to malware detection into practice?
2. Given the rapidly growing recognition of the menace posed by cyber threats to the U.S economy and national defense, were there other potential applications of DL to cybersecurity?

Potential for Cybersecurity Applications

Perlovsky did not feel that he had any particular expertise in cybersecurity; most of his past research, both theoretical and as applied in classified systems, had focused on identifying objects in the presence of high

levels of noise. For this type of task, dynamic logic had proven to be both highly effective and highly efficient in terms of the time and resources required. Particularly in light of the malware test data set where dynamic logic had performed well beyond expectations, it seemed as if cyber threat detection and classification were good potential uses for the algorithm.

Prominent Cyber Threats

Cybersecurity threats could be experienced from many sources. Norton-Symantec, a world leader in anti-virus and firewall software, classified 11 of these as being particularly serious. These were as follows:

1. **Virus:** Software that was installed without the user's knowledge that attached to other programs and could also self-replicate—often “mutating” in the process, so that its signature could change. Viruses could both damage a user's systems directly and, even if it did not, could consume system resources.
2. **Spam/Spim/Spit:** Unsolicited communications of three different types: Junk mail (spam), junk messages (spim), and junk internet phone calls over systems such as Skype (spit). All three forms could consume system resources and the user's time.
3. **Spoofing, Phishing, and Pharming:** Techniques where an intruder masqueraded as a valid user, contact, or potential contact in order to get the target to take an action that would be potentially damaging. This could involve misrepresenting a source address (spoofing), sending a false, but tempting email message (phishing), or redirecting traffic to an unintended website (pharming).
4. **Spyware:** Software, normally installed through a virus or other vector (such as a phishing attachment) that surreptitiously stole information—such as IP or identity information—from a user's system.
5. **Keystroke Logging (Keylogging):** Software, normally installed through a virus or other vector (such as a phishing attachment) that surreptitiously reported a user's keystrokes, a process through which critical information—most frequently logins and passwords—could be acquired.
6. **Adware:** Software, normally installed through a virus or other vector (including the intentional installation of freeware or shareware) that popped up advertisements on the user's system, consuming both system resources and the user's time.
7. **Botnet:** A collection of computers with surreptitiously installed software that could be repurposed to perform various nefarious activities—such as the delivery of spam or a DOS attack—without the knowledge of the owners of the system.
8. **Worm:** Similar to viruses in impact, these were standalone programs that could self-replicate over a network.
9. **Trojan Horse:** An application or file that masqueraded as achieving a desired end while, at the same time, actually performing some malicious purpose.
10. **Blended Threat:** A coordinated combination of threats.

11. **Denial-Of-Service Attack (DOS Attack):** A systematic attempt to prevent a website or other internet service from functioning through overwhelming it with garbage traffic. Often conducted using botnets.

Each of these threats are more fully described in Exhibit 4.

Possible Relevance of Dynamic Logic

As he considered how the eleven threats related to dynamic logic, Perlovsky returned to the core strengths of the approach, namely:

- The ability to use vague patterns as a starting point in order to identify crisp patterns/objects in the presence of large amounts of noise.
- The ability to identify significant commonalities between objects that were described by a set of attributes, so large that traditional techniques—such as cluster analysis—were computationally impractical.

Of particular benefit was dynamic logic's ability to accomplish these tasks in a very short amount of time, using far fewer computational resources than other approaches. Also, unlike pattern matching techniques such as neural networks, the approach was not heavily reliant on previous system training.

The dynamic logic algorithm also had some practical constraints. As currently operationalized, it worked best when each observation was described by a consistent set of attributes. In the case of the earlier examples, this consisted of a set of pixels describing a screen, and a pre-defined set of specific characteristics that described a network packet. The algorithm would be much more difficult to operationalize where observations came in an unstructured form, such as plain text or disk images. The algorithm, particularly in its second form, would also not detect threats per se. Once it had identified self-similar collections of observations, it would need an expert's interpretation to ascertain whether a particular collection represented a threat behavior or a normal behavior. The intervention of experts in selecting attributes used to describe each observation might also be required to ensure that the collections identified were of a type that could capture threats.

Incorporate Dynamic Logic into Cybersecurity Applications?

For several years, Perlovsky had allowed the potential cybersecurity applications of dynamic logic to remain dormant as he pursued other projects. In early 2017, however, Bernardo Rodrigues, an energetic master's degree student from Brazil had become aware of the algorithm. Over a period of less than a month, that student had identified a variety of ways in which DL might be incorporated into cybersecurity tools. The proposals were far ranging. They included both proprietary and open source approaches incorporated into both software components and hardware firmware. Some of the more interesting approaches are now described.

Proprietary Software and Free/Libre Open Source Software

Bringing DL into real world Cyber Security applications followed two different paths: Proprietary Software or Free/Libre Open Source Software (FLOSS). These paths are summarized in Exhibit 5.

Proprietary software was usually owned by the company or individual that developed it. It was generally sold as a product or service, which meant that there would be some sort of customer support, but also that there were restrictions on its use, and that its source code was kept secret. Additionally, proprietary

software was often built in order to meet some specific market need, which meant there was monetary value attached to the final product or service, as well as to the specialized labor force required to write it.

The FLOSS movement employed political and philosophical arguments to focus on the user's freedom and the practical benefits technology could provide to society. Source code was open for anyone to read, modify or improve. Although developers and engineers usually worked on FLOSS projects on a volunteer basis, there was also room for business to be done under this philosophy. In particular, many for-profit organizations have been established to provide service and support for open source libraries. For example, Red Hat, Inc. generated billions of dollars of revenue each year by providing standard distributions and support for the open source Linux operating system.

Although FLOSS applications all provided access to source code, they still were covered by license agreements that users needed to follow. These licenses varied considerably with respect to their terms. At one extreme, public domain open source software could be used without restriction or attribution. At the other extreme, licenses referred to as “copy left” (as opposed to copyright) demanded that any application employing the code must, itself, become governed by the same license—making it entirely inappropriate for any application that also included proprietary code. In between these two extremes were a variety of standard licenses that allowed the open source software to be mingled with other software to different degrees and with different levels of attribution.

DL Software Libraries

The proposed first step in Rodrigues’ plan for bringing DL into real world applications would consist of coming up with the design, architecture, and implementation of software libraries. This would enable software developers and engineers to integrate DL into their solutions.

Proprietary: It was expected that new forms of malware would increasingly employ artificial intelligence (AI) techniques to achieve their goals. On the other side of this battle, there was a race to come up with new AI techniques that could prevent such attacks. DL software libraries could potentially help companies build safer systems, and sell new kinds of cyber security products and services. To name a few companies that might consider investing resources into proprietary DL libraries:

- *Microsoft Corporation:* For decades, Microsoft Windows has been one of the main targets for cyber threats because of its ubiquitous presence as the main operating system used in corporate environments. Native DL libraries could potentially help improve Windows’ security features (<https://www.microsoft.com/en-us/research/>).
- *Kaspersky Lab:* Kaspersky Lab was a Russian multinational cybersecurity and antivirus provider that developed and sold antivirus, internet security, password management, endpoint security, and other cybersecurity products and services (<http://www.kaspersky.com>).
- *Intel Security Group:* Formerly known as McAfee Inc, Intel Security Group was the world's largest dedicated security technology company (<http://www.intelsecurity.com/>).

FLOSS: FLOSS software was most commonly created and maintained as projects on source repositories such as GitHub and SourceForge. These repositories allowed many users to collaborate on the same project. They also served to manage stable and developmental releases and provided the capability of managing separate development branches. One example in this context was Google’s TensorFlow, a

Python library for machine learning. Since its transition to the Open Source world (Apache License 2.0) in 2015, TensorFlow democratized education about machine learning and artificial intelligence (AI) all around the world by enabling the use of deep learning neural networks by millions of students, developers, and engineers.

With the goal of turning dynamic logic algorithms into FLOSS libraries and creating a community of DL researchers, Rodrigues had already initiated the Aristotle Project. One of the project's main goals was to develop a set of modules in the Python programming language and libraries in the C/C++ programming language. While Python was a powerful programming language with wide adoption in academic environments, C/C++ was a very low level language that offered both exceptional speed and good integration with hardware. The project complied to the GNU Lesser General Public License v3.0, which meant proprietary projects could also make use of Aristotle's libraries without being forced to become FLOSS. It was important to note that Aristotle's libraries were meant to be agnostic as to which application would be built on top of them, so they could be used in a wide variety of research areas, both for commercial and non-profit purposes. The project was also intended to help educate AI researchers about DL with wikis, videos, and a forum dedicated to discussing DL. One of the project's mottos was: "The best way to learn is by actively promoting knowledge within the community."

As its ultimate goal, Rodrigues envisioned that the Aristotle Project would lead to the creation of a new programming language that could enable ideas to be expressed in terms of DL. Such an achievement had the potential to be an important breakthrough in AI. To name a few examples of research areas that could benefit: music recognition and synthesis, speech recognition and synthesis, computer vision, autonomous vehicles, cyber security, augmented reality, computational neuroscience, computational neurolinguistics, psychology, and artificial intelligence. More information about Aristotle can be found at the website (<http://www.aristotleai.org>).

If Aristotle were to flourish, the project needed mentoring, more human resources, and funding. This could be accomplished through grants for graduate research or through private funding. To list a few organizations that could be interested in helping Aristotle:

- *OpenAI*: A non-profit artificial intelligence research company, associated with business magnate Elon Musk, that aimed to carefully promote and develop friendly AI in such a way as to benefit humanity as a whole (<https://openai.com/research/>).
- *Machine Intelligence Research Institute*: A research institute with the mission of doing foundational mathematical research to ensure smarter-than-human artificial intelligence had a positive impact for mankind (<https://intelligence.org/>).
- *Google*: The ubiquitous search engine company awarded funding for computer science and related fields, with a focus on subjects of interest to the tech giant like machine learning and human-computer interaction. It supported both faculty and PhD students (<https://www.google.com/policyfellowship/>).

R&D: DL + Cyber Threat Detection

Even though preliminary experiments showed promise in applying DL to cyber security, more extensive research & development was still necessary. Carefully designing and performing exhaustive experiments needed to be done in order to allow DL to reveal its true performance in realistic scenarios.

Proprietary: Besides the theoretical and technical aspects, research in the private sector had a bunch of other questions to be answered. For example: What cyber security market niches could DL fill? What was

its real market value? What business models could be applied? What kinds of products and services could be tailored? The same companies mentioned before (Microsoft, Kaspersky, Intel Security Group) were listed as companies interested in doing such research.

FLOSS: One of the most commonly used FLOSS cybersecurity tools was Wireshark, a cross-platform packet analyzer. It was used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark was distributed under the GNU General Public License v2.0, which meant that anyone could freely use it on any number of computers as they liked, without worrying about license keys, fees or such. In addition, all source code was freely available under the GPL.

Amongst Wireshark's main applications:

- Network administrators used it to **troubleshoot network problems**. For example, finding the source of Denial of Service (DoS) attacks.
- Network security engineers used it to **examine security problems**. For example, inside a network a specific host has been sending suspicious unsolicited packets from an unusual port, possibly infecting other machines.
- Developers used it for **debugging in application development**. For example, a developer could employ Wireshark to debug some application that needs to establish a connection with a specific server.
- Students used it to **learn network protocol internals**. For example, many undergraduate and graduate level courses incorporated Wireshark as a teaching tool, providing the students rich insights into how network protocols actually worked.

Networks exchanged information in fixed size chunks of bits known as packets. Wireshark worked by capturing these packets (with help from the libpcap C library) from the machine's network interface and saving them into specific file formats, such as pcap (see Exhibit 6 for a Wireshark packet display). Wireshark then dissected this packet information by trying to identify the packet type and acquiring as much information as possible from it, a process referred to as packet dissection. It was in this stage that Rodrigues felt that DL offered significant potential. He felt that DL might aid Wireshark in processing such information, identifying new attributes, and coming up with higher level representations of the traffic. Because DL's algorithms needed low amounts of processing power, most of these analyses could potentially be performed on-the-fly, without the need for offline processing on expensive hardware. A DL based Wireshark dissector could serve as a tool for a whole new area of cybersecurity research, where academics and companies were able to explore DL by performing a wide range of experiments with malicious software in controlled network environments. Aristotle's libraries would provide key elements for the implementation of these DL based packet dissectors.

Real World Applications of DL

One possible application of DL in the real world was identified in networks. Computer networks could roughly be divided into endpoint machines (called hosts) and various intermediary devices, here referred to as network equipment (NE). Hubs, switches, modems, routers, bridges, and firewalls were a few examples of kinds of NEs.

In a computer network, the interactions occurred at many levels of abstraction, each employing its own protocols. The vast majority of modern computer networks relied on the Open Systems Interconnection (OSI) Model, which established various well-known protocols organized in 7 different layers of abstraction (see Exhibit 8).

Dynamic Logic had already proven its potential for detecting anomalies in the Network Layer (IP) of network traffic. This detection could potentially happen in many layers of the OSI Model, however, producing very accurate and sophisticated models. DL could help NE firmware detect and mitigate unusual traffic, perhaps even preventing it from causing any significant damage. DL had the potential to make the backbones of the Internet safer by providing intelligent protection mechanisms to its main data routes.

Proprietary: NE manufacturers could be interested in applying DL into their firmware because it would stand as a differential feature in the market. A few examples of companies that could be interested in integrating DL algorithms into their NE firmware's security features:

- *Cisco Systems, Inc.:* An American multinational technology conglomerate that developed, manufactured, and sold networking hardware, telecommunications equipment, and other high-technology services and products (<http://www.cisco.com/>).
- *Juniper Networks:* An American multinational corporation that developed and marketed networking products. Its products included routers, switches, network management software, network security products, and software-defined networking technology (<http://www.juniper.net/us/en/>).
- *AT&T:* An American multinational telecommunications conglomerate (<http://www.research.att.com/>).
- *Amazon Web Services:* A subsidiary of Amazon.com that offered on-demand cloud computing platforms (<https://aws.amazon.com/>).

FLOSS: Many NEs relied on some version of the Linux operating system. GNU/Linux was the result of Richard Stallman's GNU efforts to implement a truly Free/Libre and functional version of the Unix operating system combined with Linus Torvalds' kernel called Linux. Rodrigues believed that Aristotle's libraries could potentially help to bring new features, such as big data analytics and improved security, into modern Linux based NE firmware. To name a few projects that could benefit from Aristotle:

- *PNDA:* An open source big data analytics project out of Cisco Systems. The platform took data from various data center devices and applied big data analytics. Customers could mine the data for troubleshooting, performance management, capacity planning, and security use cases (<http://pnda.io>).
- *DPDK:* A Linux Foundation project that provided a set of libraries and drivers for fast packet processing. It was designed to run on any processor. The first supported CPU was Intel x86 and it was now extended to IBM POWER and ARM processors. It ran mostly in the Linux world (<http://dpdk.org>).

- *Open vSwitch*: A production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It was designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (<http://openvswitch.org>).

Perlovsky's Decision

As Perlovsky looked over Rodrigues' various ideas, it began to dawn on him that he needed to make some decisions about DL's cybersecurity future. While he had been very impressed with Rodrigues' energy and initiative, he recognized that the young man's vision was unlikely to be realized without considerable support—some of which would necessarily require Perlovsky's attention.

Among the decisions that he needed to consider:

- *Did he want to support the open source project or keep his options open for proprietary paths?*

While he had no objections to open source projects in general, there were questions in his mind. Specifically, he stated:

We must understand requirements for an open source project to succeed. Can it succeed as a stand-alone software? Which specific applications should it address? What kind of interface will it require?

He also felt that he needed to know more about the question of how open source and proprietary solutions might work together.

- *How much encouragement should he be giving to Rodrigues?*

Perlovsky appreciated Rodrigues' enthusiasm and support for his work. Given that Rodrigues seemed to be the main force pushing the project forward, he was reluctant to put any barriers in the young man's way. He stated:

I would be glad to give advice to Rodrigues, to answer questions he might have. My first advice would be that he should decide how he should proceed and how much effort he is ready to invest.

Perlovsky also recognized that Rodrigues faced two potential sources of risk. First, was the risk that incorporating dynamic logic into existing cybersecurity applications might be more technically challenging than initially thought, and might prove beyond the capabilities of a single individual—even a highly motivated one. Second, the project was in its preliminary stages and the effective “ownership” of the project could shift. Working alone in Brazil, Rodrigues could potentially find himself at the periphery of the project should active work in the U.S. or Europe begin. He wondered if Rodrigues was fully aware of these risks.

- *Should consideration be given to actively pursuing grants to further the software development?*

Perlovsky was very supportive of the possibility of pursuing grants related to the use of dynamic logic in cybersecurity. He asserted:

I would think that it is a good idea to pursue grants for software development. Dynamic Logic can solve many problems unsolvable by other means and this gives a good initial step for a grant application.

He also indicated that, given his interests, he was unlikely to initiate such grant proposals on his own. He wondered how he might entice others to take the lead, while still getting the opportunity to participate as a co-investigator.

- *To what extent should he seek to find a U.S. researcher to whom the oversight of the project could be delegated?*

Perlovsky was very attracted to the idea of having other researchers move forward with the project. He was not convinced of its feasibility, however, stating:

Actually, the question is: How to do it? From my experience, researchers capable of a project oversight tend to concentrate on their own ideas. Only young researchers can notice and develop existing ideas.

In addition to these specific questions, he faced a broader question: To what extent would it make sense to redirect his attention to the potential cybersecurity applications of DL?

References

- Perlovsky, L. (2016a, July). Gödel vs. Aristotle: Algorithmic complexity, models of the mind, and top representations. In *Neural Networks (IJCNN), 2016 International Joint Conference* (pp. 1787-1794). IEEE.
- Perlovsky, L. I. (2016b). Physics of the mind. *Frontiers in Systems Neuroscience*, 10(84). 1-12.
- Perlovsky, L. (2016c). The ANN and Learning Systems in Brains and Machines. In Angelov, P.P., *Handbook on computational intelligence: Volume 1: Fuzzy logic, systems, artificial neural networks, and learning systems* (pp. 281-316). Singapore: World Scientific Publishing Co.
- Perlovsky, L. I., & Ilin, R. (2010). Grounded symbols in the brain, computational foundations for perceptual symbol system. *Webmedcentral Psychology*, 1(12), WMC001357.
- Perlovsky, L., & Shevchenko, O. (2014, July). Cognitive neural network for cybersecurity. In *Neural Networks (IJCNN), 2014 International Joint Conference* (pp. 4056-4061). IEEE.

Acknowledgements

This case study is based upon work supported by the National Science Foundation under Grant No. 1418711.

Biographies

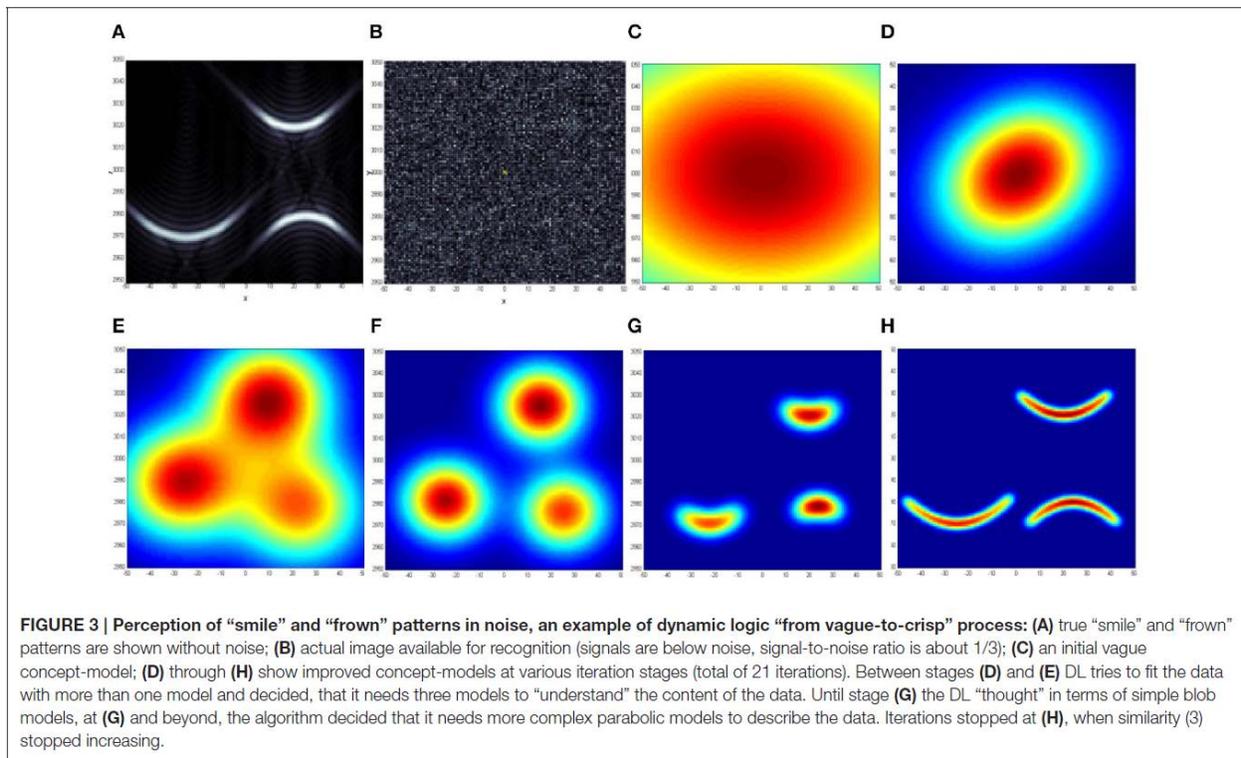


Grandon Gill is a Professor in the Information Systems and Decision Sciences department at the University of South Florida. He holds a doctorate in Management Information Systems from Harvard Business School, where he also received his M.B.A. His principal research areas are the impacts of complexity on decision-making, the diffusion of academic research findings and applying the case method to STEM education. He is currently Editor-in-Chief of the *Journal of IT Education: Discussion Cases* and of the Muma Business Review.



Bernardo Rodrigues is a Hardware Engineer at Data Traffic, a Brazilian company that specializes in smart road traffic control. He received his Bachelor's in Electronics Engineering at University of São Paulo, and is currently working towards his Master's Degree in Computer Engineering, where he is researching Machine Learning applications to Password Strength metrics.

Exhibit 1: Dynamic Logic Used to Identify Objects Obscured by Noise



Source: Perlovsky (2016b, p. 9).

Exhibit 2: Using Dynamic Logic to Uncover Related Objects

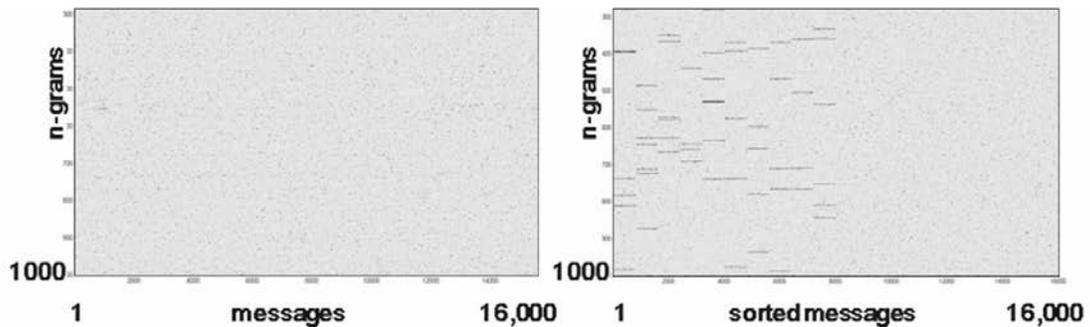


Fig.1. On the left are 16,000 messages arranged in their sequential order along the horizontal axis, n . The total number of possible n -grams in the network is 1000, they are shown along the vertical axis, j . Every message has or does not have a particular n -gram as shown by a white or black dot at the location (n,j) . This figure looks like random noise corresponding to pseudo-random content of messages. On the right figure, messages are sorted so that messages having similar n -grams appear next to each other. These similar n -grams appear as white horizontal streaks. Most of message contents are pseudo-random n -grams; about a half of messages have several similar n -grams. These messages with several specific n -gram values have specific contents, they belong to certain models, and could be malware codes. If one attempts to use any standard sorting algorithm to sort the left image along the horizontal axis until the streaks will become visible, the combinatorial complexity will be on the order of 16000!

Source: Perlovsky (2016a, p. 1790). Images inverted for visual clarity.

Exhibit 3: Malware Detection Using Dynamic Logic

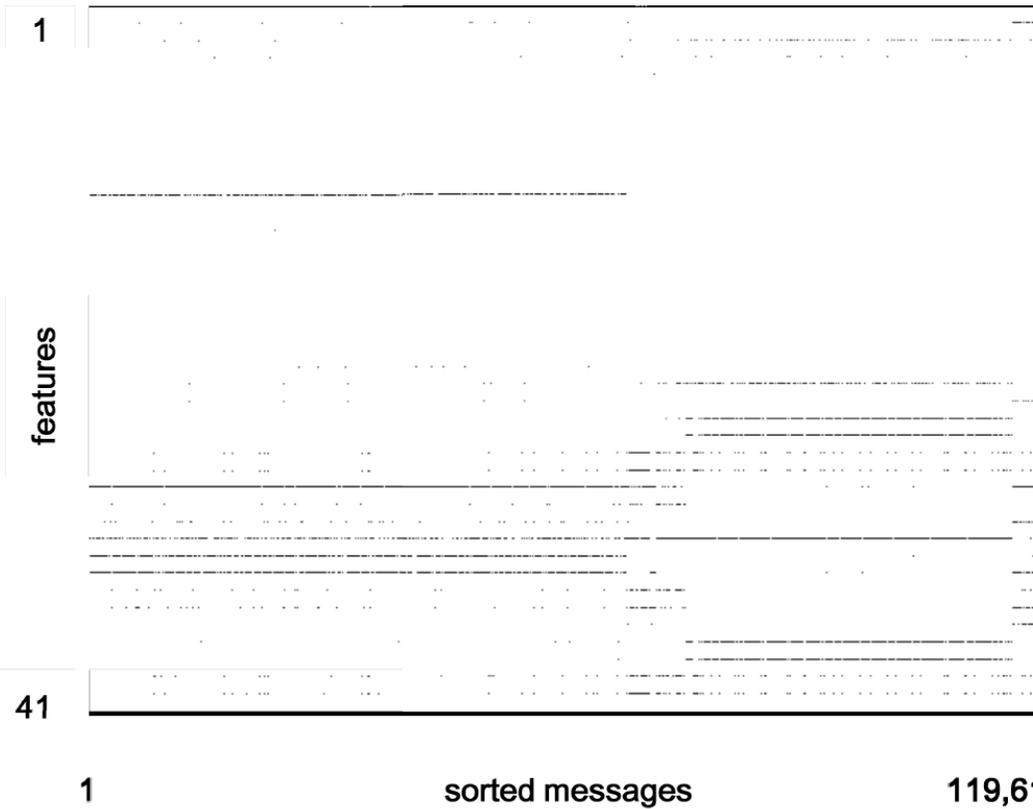


Fig.3. This figure shows sorted messages from 6 groups (1 normal and 5 malware) of the KDD data set corresponding to Fig.2 right. (We do not show unsorted messages, similar to Fig.2 left). Along the horizontal axis: 67343 normal, 2931 portsweep, 890 warezclient, 3633 satan, 41214 neptune, 3599 ipsweep. Groups are significantly different in size, and not all features important for a group are necessarily present in each vector belonging to the group, therefore the look of the figure is not as clear cut as Fig.2 right. Nevertheless all vectors are classified without errors, and without false alarms.

Source: Perlovsky & Shevchenko (2014, p. 4059). Images inverted for visual clarity.

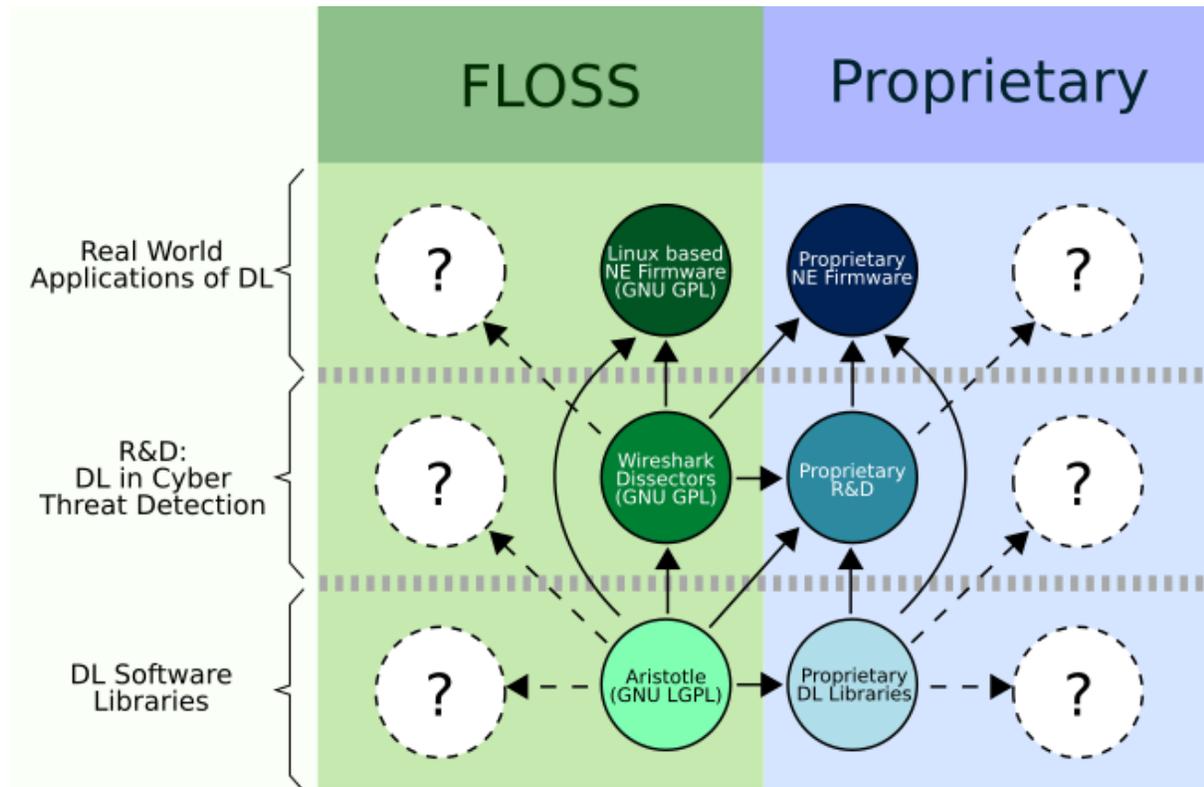
Exhibit 4: Norton-Symantec Cybersecurity Threat Ratings

This list profiles the 11 most common security threats and classifies them by prevalence in descending order.

Threat	Description	Prevalence
Virus	Piece of software able to infect a computer without the user permission or knowledge, as well as to replicate itself. Usually spread over networks or removable media by attaching itself to other files or programs.	Extremely High
Spam, Spim, Spit	Spam is junk email. Spim is similar to spam, but distributed through instant messaging services. Spit is also similar to spam, but the media is internet telephony.	Extremely High
Spoofing, Phishing and Pharming	Spoofing is an attack where a person or program impersonates someone else (usually with a spoofed URL). Phishing is a form of spoofing where the attacker seeks to steal the victim's credentials. Pharming is a form of spoofing where the a group of users is targeted, usually through some malicious modifications to popular DNS servers.	Extremely High
Spyware	Software secretly installed without user's consent with the intent of monitoring their activity.	High
Keylogging	Piece of software that captures, records, and reports every key stroked by the user without their consent or knowledge.	High
Adware	Software that shows advertising on the computer, usually without the user's consent or as a condition for installing some desired software.	High
BotNet	A group of software robots that run automated tasks over networks. Frequently used to launch Distributed Denial-of-Service (DDoS) attacks.	High
Worm	A self-replicating malicious piece of software that send copies of itself over the network. Does not need to attach itself to other programs.	Moderate
Trojan Horse	A piece of software that appears to be legitimate but carries a malicious payload.	Moderate
Blended Threat	A combination of multiple malicious components, such as a worm, a virus, and a trojan horse.	Moderate
Denial of Service (DoS) Attack	An attempt to make a computer resource unavailable by starvation.	Low

Source: Adapted from: http://www.symantec-norton.com/11-most-common-computer-security-threats_k13.aspx

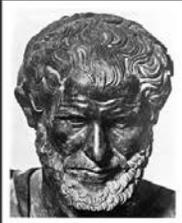
Exhibit 5: Alternative Possible Paths to Incorporate DL



Source: Developed by case writer.

Exhibit 6: Aristotle Project's Website

Aristotle Project



[GitHub](#)

About

In Leonid Perlovsky's words:

"The widely accepted story is that Aristotle founded logic as a fundamental mind mechanism, and only during the recent decades science overcame this influence. I would like to emphasize the opposite side of this story.

Aristotle thought that logic and language are closely related. He emphasized that logical statements should not be formulated too strictly and language inherently contains the necessary degree of precision. According to Aristotle, logic serves to communicate already made decisions. The mechanism of the mind relating language, cognition, and the world Aristotle described as forms. Today we call similar mechanisms mental representations, or concepts, or simulators in the mind. Aristotelian forms are similar to Plato's ideas with a marked distinction, forms are dynamic; their initial states, before learning, are different from their final states of concepts.

Aristotle emphasized that initial states of forms, forms-as-potentialities, are not logical (i.e., vague), but their final states, forms-as-actualities, attained in the result of learning, are logical. This fundamental idea was lost during millennia of philosophical arguments. It is interesting to add Aristotelian idea of vague forms-potentialities has been resurrected in fuzzy logic by Zadeh; and Dynamic Logic described here is an extension of Fuzzy Logic to a process from vague to crisp".

Aristotle Project's main goal is to turn Perlovsky's Dynamic Logic algorithms into Free Software. We aim to provide **Python modules** and **C/C++ libraries**. Python is a powerful programming language with wide adoption in academic environments, while C/C++ is an ISO Standard and provides good integration with hardware. We want to write libraries that work agnostically to which application is built on top of them, which means they could be used in a wide variety of research areas. We also want them to be usable as freely as possible by anyone, so the project complies to the **GNU Lesser General Public License v3.0**, which means Aristotle's libraries could be used for either commercial or non-profit purposes.

Aristotle Project also aims to educate AI researchers about Dynamic Logic with wikis and a forum dedicated to discussing DL. We really believe that the best way to learn is by sharing knowledge with the community.

As its ultimate goal, Aristotle Project envisions the creation of a **new programming language that enables ideas to be expressed terms of Dynamic Logic**. Such achievement has the potential to be an important breakthrough in Artificial Intelligence. To name a few examples of research areas that could benefit from Aristotle: **Music Recognition and Synthesis; Speech Recognition and Synthesis; Computer Vision; Autonomous Vehicles; Cyber Security; Augmented Reality; Computational Neuroscience; Computational Neurolinguistics; Psychology; and Artificial Intelligence**. But we need a community to make that happen. If you're an AI researcher and you're interested in collaborating with Aristotle, don't hesitate to **contact us**. At the moment, we also need help to properly structure a wiki, a forum, and set up SSL certificates for the website.

Dynamic Logic

For a theoretical foundation about Dynamic Logic:

- Perlovsky, Leonid I. "Neural Networks, Fuzzy Models And Dynamic Logic". Aspects of Automatic Text Analysis (2007): 363-386. Web. 13 Apr. 2017. <http://bit.ly/2pclTJx>
- Perlovsky, Leonid I. "Emotional Cognitive Neural Algorithms With Engineering Applications". 1st ed. Springer-Verlag Berlin An, 2014. <http://bit.ly/2paysPg>

Solutions for the problems from Perlovsky's book "Emotional Cognitive Neural Algorithms with Engineering Applications" can be found in the form of Python scripts [here](#).

Mission

Actively and Responsibly contribute to the invention AI.

Vision

Promote education about Dynamic Logic and AI
Build and encourage a community of Dynamic Logic researchers.
Be the reference implementation of Dynamic Logic algorithms by 2019.
Have a new Dynamic Logic programming language by 2020.

Values

Freedom.
Equality.
Community.
Asmov's 3 Laxes of Robotics.

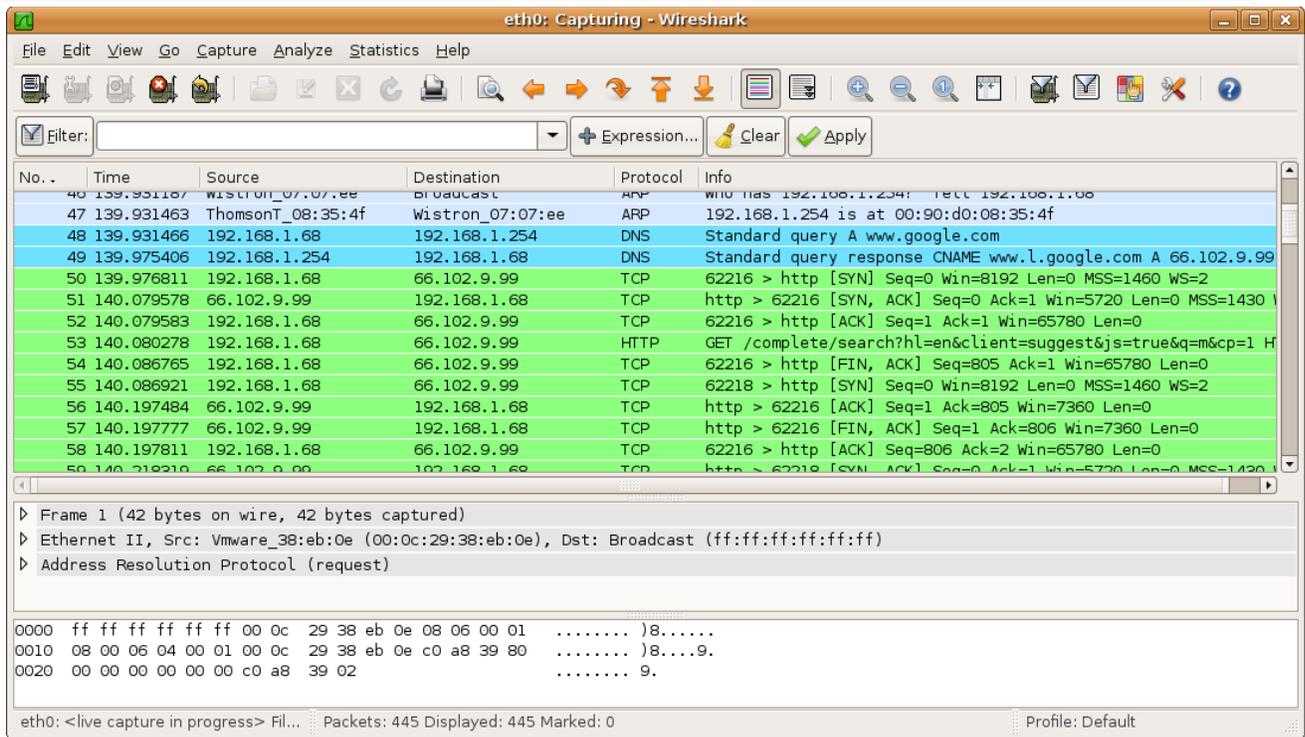
Contact

aristotle@aristotleai.org

Maintained by Bernardo Rodrigues.

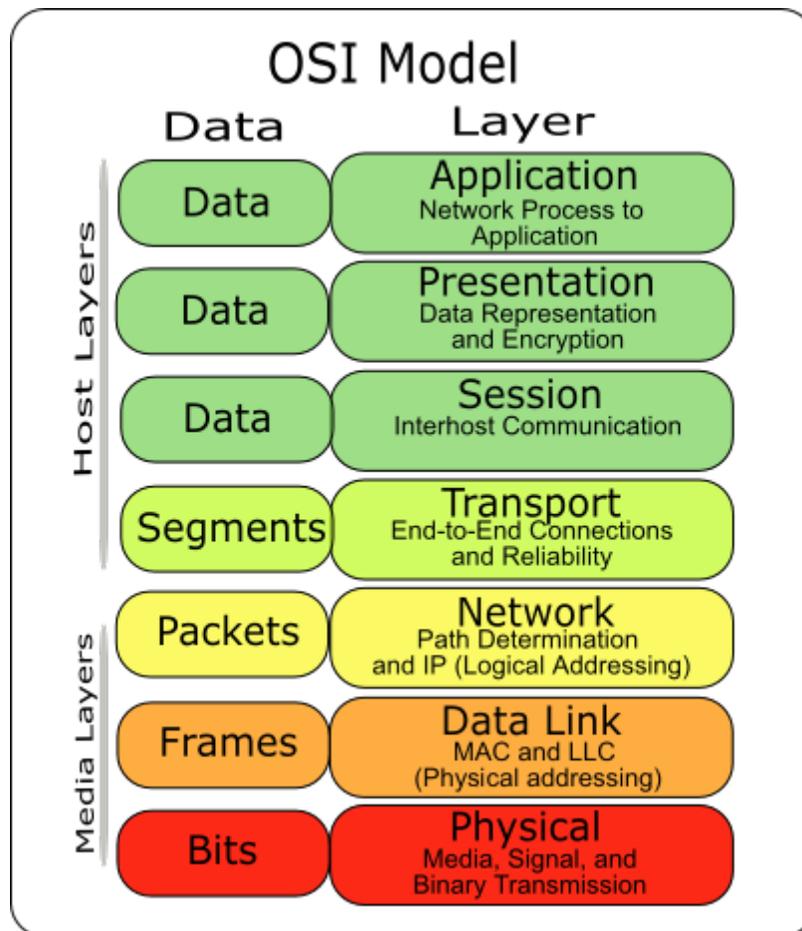
Source: <http://www.aristotleai.org/> (2017)

Exhibit 7: Wireshark Screenshot



Source: Wikimedia Commons (2017)

Exhibit 8: OSI Model



Source: Wikimedia Commons (2017)