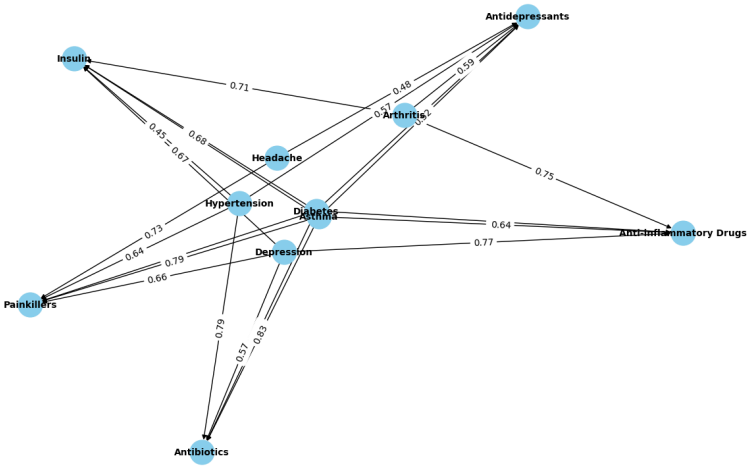
PHOTREK

Architectural Design for Uncertain Knowledge Graphs

Milestone 5 Report



Principal Investigator & Project Manager:

Kenric Nelson

Photrek, LLC

admin@photrek.io

17 June, 2024

PHOTREK

Executive Summary

Photrek has designed and implemented an uncertain knowledge graph for medical information and used it to create UKG facts, including relative risk aversion and coupled probabilities. Probabilities are derived using Bayesian methods. Graph construction is performed with Neo4j. Data analysis and visualizations were implemented in Python. Design details for the development of an interactive interface to integrate a UKG with an open-source Large Language Model (LLM) using Hugging Face's Transformers library are presented. Photrek is now using the New England Research Cloud (NERC) for computing capabilities in ongoing work, future decentralized application (DApp) engagements and aiding in facilitating development activities across a geographically diversified team. Future Plans include producing demonstrations, working with probabilistic programming across groups of KG elements, and integration with OpenCog Hyperon.

Milestone 5 Develop UKG
Total Budget: $ 6,550 USD in AGIX
Objectives:

- Use probabilistic graph foundation to create UKG facts
- Develop small network of UKG facts
- Develop interface (for UKG) with static KG and/or LLM
- Initial use of New England Research Cloud

Deliverable: Report with initial code

1. Use probabilistic graph to create UKG facts

In this section, we outline the process of creating (UKG) facts using a probabilistic graph foundation, specifically focusing on disease-treatment pairs and their associated probabilities.

These probabilities were derived using Bayesian method, taking into account various factors influencing treatment efficacy. Additionally, the Python programming language was used in the process of data analysis and visualization. Libraries such as numpy, and matplotlib.pyplot were employed to create detailed visual representations of the probabilistic relationships.

Methodology and Analysis

A key aspect of our approach is the calculation of coupled probabilities, essential for risk-aware machine intelligence. The calculations account for uncertainty and risk aversion. The following parameters and methodology are used:

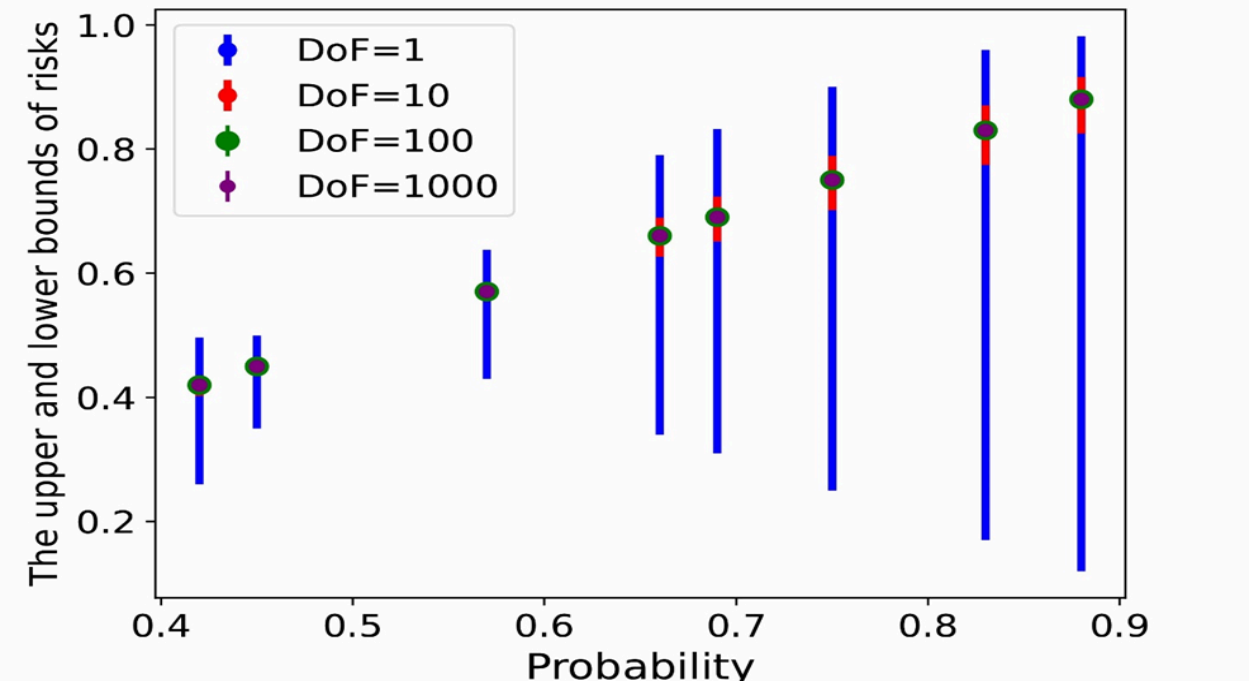
1. Relative Risk Aversion (r):

The relative risk aversion (r) was computed based on the provided parameters, including the degrees of freedom (λ), dimension (d=1) and α=2 of the distribution. The formula used was $r = \frac{\alpha}{\lambda}/(1 + \frac{d}{\lambda})$. In our study, we considered four degrees of freedom: 1000, 100, 10, and 1.

2. Coupled Probability Calculation:

Once the relative risk aversion (r) was determined, we used it to compute coupled probabilities by raising the initial probabilities to the power of r and then renormalizing them. This approach allowed us to capture the uncertainty in the probability estimates and provided a more cautious assessment of the treatment efficacy.

During the study, we plotted the Coupled Probability using different degrees of freedom on the x-axis and the risk aversion (upper bound) and risk tolerance (lower bound) on the y-axis, using vertical lines to represent the confidence intervals, as shown in Figure 1. The analysis revealed several key insights. As the degrees of freedom decreased, the confidence intervals (upper and lower bounds) widened, indicating increased uncertainty in the probability estimates. This trend highlights the importance of sample size and other sources of incomplete information in determining the reliability of probabilistic assessments. Additionally, the plot demonstrated that starting with an uncertain probability results in less influence from coupled probabilities, while a certain probability can be significantly altered given a small degree of freedom. These findings underscore the importance of accounting for uncertainty and risk aversion in probabilistic graph-based models, particularly when dealing with medical data where accurate treatment recommendations are critical.



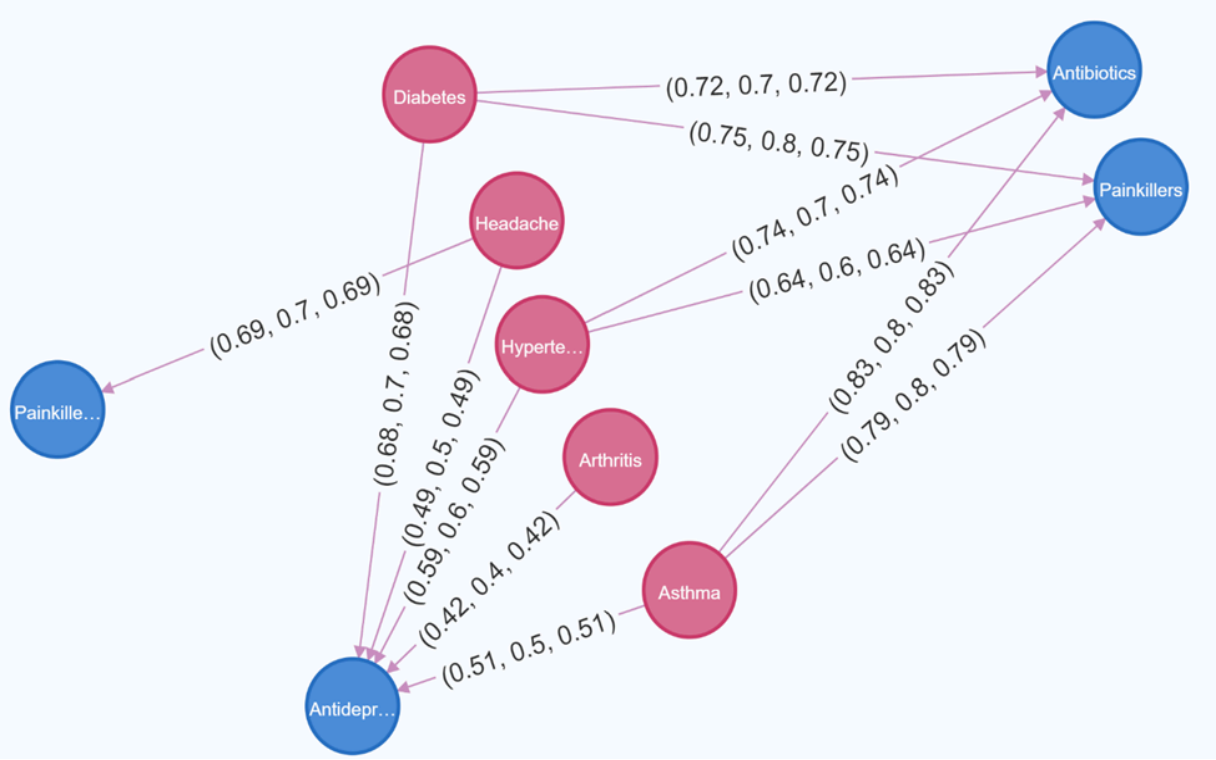
2. Develop small network of UKG facts

To develop a small network of (UKG) facts, we utilized the Neo4j database, which excels in managing graph structures. This approach facilitated efficient querying and analysis of the relationships between diseases and their associated treatments. Data attributes were customized using the SET clause in Cypher, ensuring precise configuration of relationship properties aligned with the imported data.

Furthermore, the edges in the graph were visually represented as parentheses as shown in Figure 2, where the numbers inside indicate the lower bound (risk tolerance), the probability, and the upper bound (risk aversion). Specifically, the graph depicts the scenario where the degree of freedom $\lambda = 1000$, with both the upper and lower bounds closely approximating the probability. This alignment is also evident in Figure 1, where the $\lambda=1000$ and is marked by a purple point.

This structured approach ensures clarity and comprehensibility, illustrating how the UKG facts were developed using graph foundations within the Neo4j framework. This methodological setup not only enhances the understanding of the graph-based

model but also underscores its applicability in managing and analyzing medical data for informed decision-making in treatment recommendations.



3. Develop interface (for UKG) with static KG and/or LLM

In this section, we present design details on development of an interactive interface that integrates a dynamic UKG with a Static KG and an open-source Large Language Model (LLM) using Hugging Face's Transformers library. We will focus on using these tools to execute advanced queries about relationships among medical information.

HuggingFace's Transformers library makes available many large language models, for example GPT-2, making it possible to generate text and answers based on textual queries. Additionally, NetworkX can be used to create graphical representations of the knowledge base. Alternatively, a database can also store diseases and Medication with their relationships instead of using NetworkX.

Our goal is to build a system capable of using the dynamic data stored in an UKG and executing complex queries about relationships among this data using long-range language models. To achieve this, we will integrate several components:

1. Dynamic UKGs to capture temporal data

2. Static KG (optional)

3. Large Language Model (LLM)

4. Probabilistic Programming Language (PPL)

Integration Steps:

1. The UKG representing the knowledge base is constructed using NetworkX. This graph illustrates relationships between diseases, treatments, and other medical information.

2. A language model such as GPT-2 is downloaded from Hugging Face's Transformers library, and the tokenizer is used to convert texts into formats that the model can process.

3. The user's choice between using the knowledge base or the language model is checked. If the entity exists in the knowledge base, relevant information is retrieved. If not, the language model is used to generate the required texts.

4. Executing Complex Queries using PPL: PPL is employed to execute complex queries, such as selecting the best medication for a specific disease condition based on available information. PPL provides a robust method to handle uncertainty and probabilistic data analysis.

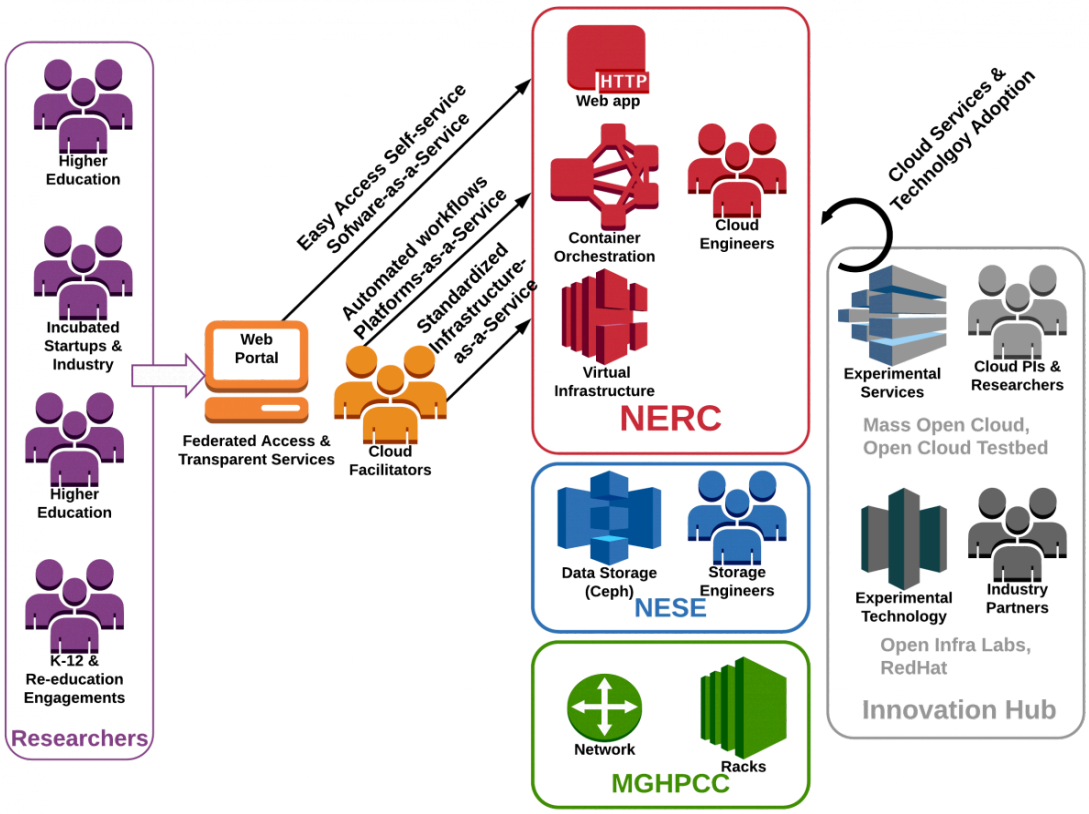
This system is designed to integrate the Static Knowledge Graph with a LLM, enabling it to execute complex queries accurately and efficiently. Using these tools and libraries, the system can achieve high performance and provide precise data-supported answers to users' needs in various fields such as medicine and scientific research.

4. Initial use of New England Research Cloud (NERC)

Photrek conducted an evaluation of multiple hosting platforms and decided upon using NERC for both training models and hosting DApps. This evaluation included computational capabilities, security aspects, uptime/reliability assessment, and operational costs. Photrek believes this choice of a hosting platform will facilitate sustainable developments for future DApp engagements and facilitate development activities across a geographically diversified

team. Photrek is one of the first businesses participating in NERC. NERC is part of the [Massachusetts Green High Performance Computing Center](), which opened in November 2022 at Holyoke, MA to provide HPC capabilities for Harvard, MIT, University of Massachusetts, Boston University, Northeastern University and the other universities in the New England region.

Specifically, the NERC Deployment Configuration (DC) provides researchers and engineers an easy way to configure and scale a platform for provisioning and assigning compute resources, including A100 GPUs, to Photrek projects on demand. At present we have provisioned a virtual machine hosting individual virtual desktops for Photrek developers facilitating a shared and standardized development environment. Users can store and execute personal workbench routines while having access to NERC's compute resources for additional project runtime and processing needs. NERC's storage costs are far below the industry average providing Photrek with an affordable yet very reliable compute foundation. NERC provides a portal for researchers to take advantage of a suite of services (Infrastructure as a service (IaaS), Platform as a Service (PaaS), Software as a service (SaaS)) not commonly available today.



Code Repository

https://github.com/Photrek/Uncertain-Knowledge-Graphs

Future Project Plans

1. Demos
   We have scheduled a demonstration on UKGs with SingularityNET technical staff on July 12, 2024.

2. Probabilistic Programming across groups of KG elements
   This is also related to the third future plan. A group of KG elements forms a subgraph. The metagraph structure allows connections between these subgraphs, giving the possibility of quantifying the uncertainty of the relationships between subgraphs.

3. Integration with OpenCog Hyperon
   Hyperon currently comprises two core software components: Atomspace and the MeTTa programming language. The atomspace is a distributed neural-symbolic knowledge metagraph, typically a graph where the nodes and edges themselves can also be made up of graphs. In the atomspace edges may connect any number of nodes. It should be possible to induce probabilities on the edges between the graphs used to build the metagraph and give them a natural meaning.

Budget & Schedule

Note: The milestone numbers were revised per SingularityNET's request to start at 1.

M#	Milestone name	Milestone deliverable	Related budget	Status
1	**Contract Signing & Management Reserve**	**Funding put in reserve to address unforeseen requirements** **Contract Signature**	**$ 1,750.00**	**Report and Invoice Submitted on Dec 7th.**
2	**Brief SingularityNET KG & LLM team regarding Objectives**	• **Determine current status of SingularityNETs KG & LLM development** • **Acquire feedback on probability with upper and lower bounds** • **Acquire feedback on use of UKG**	**$ 5,460.00**	**Report & Invoice submitted Dec 22nd.**

		as augmentation to static KG & LLM **Report on refined objectives**		
3	**Draft Architecture Design**	• **Design details of use of probabilistic graph to form UKG** • **Design details of use** • **Draft architectural plan** **Report with architecture design**	**$ 5,870.00**	**Feb 16th Report & Invoice Submitted**
4	**Develop Use Case Plan**	• **Define code base foundation** • **Choose specifics of use case** • **Complete analysis of architecture with use case** **Report on opportunity risk analysis**	**$ 6,550.00**	**Apr 12th Report & Invoice Submitted**
5	**Develop UKG**	• **Use probabilistic graph foundation to create UKG facts** • **Develop small network of UKG facts** • **Develop interface with static KG and/or LLM** **Report with initial code**	**$ 6,550.00**	**Submitted Jun 21**
6	Use Case Demo	• Demo using UKG independently • Demon using upper & lower probabilities • Demo using UKG with static KG and/or LLM Report on Use Case Demo	$ 7,550.00	Plan: July 15th
7	Final Report with Next Steps	• Document strengths and weaknesses of design • Document lessons learned from use case • Document plans for next steps Final Report with next steps	$ 6,270.00	Plan: Aug 10th

References

Chandak, Payal, Kexin Huang, and Marinka Zitnik. 2023. “Building a Knowledge Graph to Enable Precision Medicine.” *Scientific Data* 10 (1): 67. https://doi.org/10.1038/s41597-023-01960-3.

Liu, Qi, Qinghua Zhang, Fan Zhao, and Guoyin Wang. 2024. “Uncertain Knowledge Graph Embedding: An Effective Method Combining Multi-Relation and Multi-Path.”

Frontiers of Computer Science 18 (3): 183311. https://doi.org/10.1007/s11704-023-2427-z.

Navarrete, Francisco J., and Antonio Vallecillo. 2021. “Introducing Subjective Knowledge Graphs.” In *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC)*, 61–70. Gold Coast, Australia: IEEE. https://doi.org/10.1109/EDOC52215.2021.00017.

Nelson, Kenric P. 2020. “Reduced Perplexity: A Simplified Perspective on Assessing Probabilistic Forecasts.” In *Advances in Info-Metrics: Information and Information Processing across Disciplines*, edited by Min Chen, J. Michael Dunn, Amos Golan, and Aman Ullah, 0. Oxford University Press. https://doi.org/10.1093/oso/9780190636685.003.0012.

Full Project References

Chandak, Payal, Kexin Huang, and Marinka Zitnik. 2023. “Building a Knowledge Graph to Enable Precision Medicine.” Scientific Data 10 (1): 67. https://doi.org/10.1038/s41597-023-01960-3.

Liu, Qi, Qinghua Zhang, Fan Zhao, and Guoyin Wang. 2024. “Uncertain Knowledge Graph Embedding: An Effective Method Combining Multi-Relation and Multi-Path.” Frontiers of Computer Science 18 (3): 183311. https://doi.org/10.1007/s11704-023-2427-z.

Navarrete, Francisco J., and Antonio Vallecillo. 2021. “Introducing Subjective Knowledge Graphs.” In 2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC), 61–70. Gold Coast, Australia: IEEE. https://doi.org/10.1109/EDOC52215.2021.00017.

Nelson, Kenric P. 2020. “Reduced Perplexity: A Simplified Perspective on Assessing Probabilistic Forecasts.” In Advances in Info-Metrics: Information and Information Processing across Disciplines, edited by Min Chen, J. Michael Dunn, Amos Golan, and Aman Ullah, 0. Oxford University Press. https://doi.org/10.1093/oso/9780190636685.003.0012

Appendix: OpenCog Hyperon Summary by Raymond Mata

SingularityNET OpenCog Hyperon is an ambitious project aiming to create a complete and open-source Artificial General Intelligence (AGI) system. Here's a breakdown of its key features:

- **Neural-Symbolic AI**: It combines aspects of symbolic AI (using logic and reasoning) and neural networks (inspired by the brain) to achieve human-like intelligence.
- **Scalable Architecture**: OpenCog Hyperon can be distributed across multiple machines and leverage blockchain technology for wider access and processing power.
- **MeTTa Language**: This unique programming language is designed specifically for AGI development and fosters the emergence of complex behaviors in Hyperon.

"MeTTa is unique in its ability to write introspective and self-modifying programs, which can be crucial for recursive self-improvement of AGI. It is like a modeling clay robot that can sculpt itself." — Dr. Alexey Potapov, Chief AGI Officer at SingularityNET

- OpenCog Hyperon Alpha introduces DAS: The Distributed Atomspace: This is a core component that acts as a vast and scalable knowledge base, allowing Hyperon to store and process information efficiently.

Atomspace is a crucial component in SingularityNET's OpenCog Hyperon project, acting as the brain's memory and processing center for this ambitious Artificial General Intelligence (AGI) system.

Key features:

- **In-Memory Database:** Atomspace lives in RAM for use in real-time cognitive processing, and also lives on disk in a distributed form across multiple machines, thanks to the Distributed Atomspace implementation. This is now being optimized to work on both centralized infrastructure (e.g. a server farm) or distributed infrastructure (e.g. NuNet or HyperCycle nodes).
- **Generalized Hypergraphs:** Unlike regular graphs where edges connect only two nodes, Atomspace's hypergraphs allow edges to connect any number of nodes, offering a more flexible way to represent complex relationships between pieces of information.
- **Metagraphs:** Atomspace itself can be represented as a hypergraph, allowing for even more intricate knowledge representation of various types (declarative, linguistic, mathematical, procedural, attentional, goal-related, sensory, motoric, emotional, intuitive, etc.)

Benefits for AGI:

- **Scalability:** The hypergraph structure and potential for distributed storage make Atomspace suitable for handling the massive amounts of information needed for AGI.
- **Flexibility:** It can accommodate various data types and structures, crucial for real-world intelligence that requires handling diverse information.
- **Reasoning Power:** The built-in query engine allows Hyperon to make logical deductions and solve problems based on the stored knowledge.

Overall, OpenCog Hyperon's DAS leverages distributed infrastructure like HyperCycle that is especially suited for intelligent local caching with high transaction speeds along with self-evolving customizable consensus to provide a powerful foundation for OpenCog Hyperon's knowledge processing and reasoning capabilities. Paving the way for the development of a truly intelligent system. To learn more, including jumping into their new MeTTa playground, visit hyperon.opencog.org