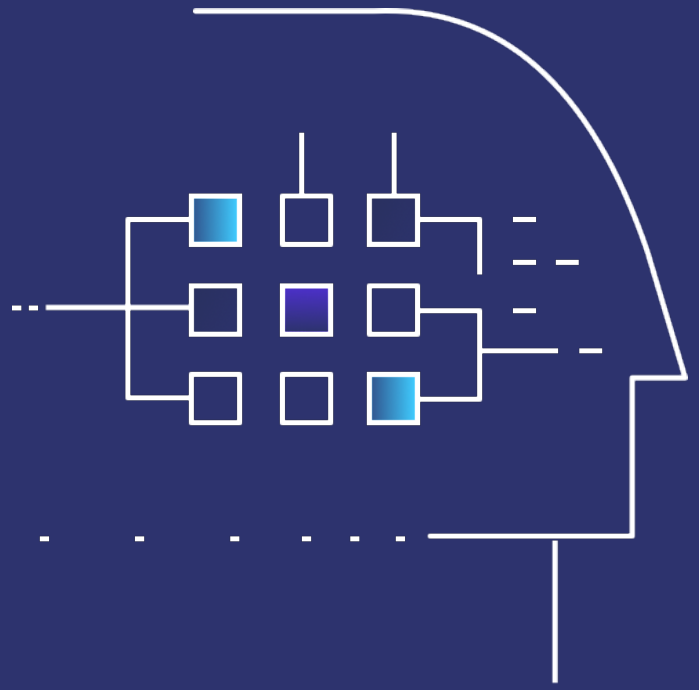
From Here to AGI

via meta-inference, uncertain logic and integrative cognitive architecture

Ben Goertzel

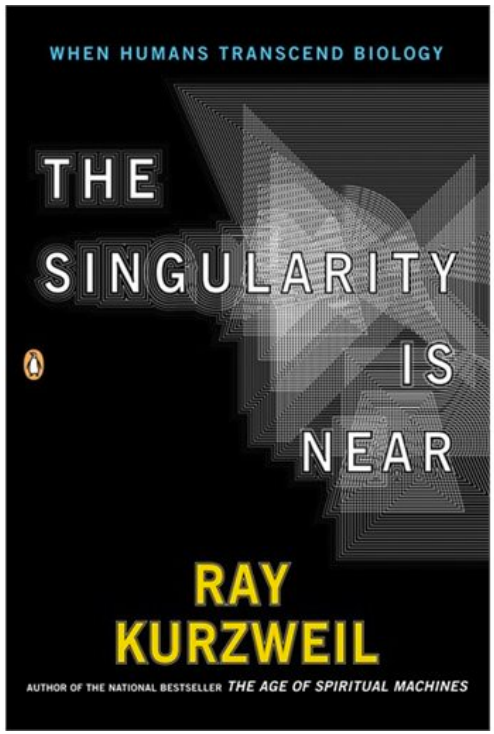


http://goertzel.org/AITP-19/

“I set the date for the Singularity- representing a profound and disruptive transformation in human capability- as 2045.

The nonbiological intelligence created in that year will be one billion times more powerful than all human intelligence today.”

The Singularity is Near
When Humans Transcend
Biology - Ray Kurzweil (2005)



Narrow AI

TRAINED or just **PROGRAMMED**

Can only do one thing

(e.g., Deep Blue can't drive a car; Google Car can't play chess)

- Deep Blue
- AlphaGo
- Facebook Face Recognizer
- Google Self Driving Car
- IBM Watson for Jeopardy
- IBM Watson for Medical

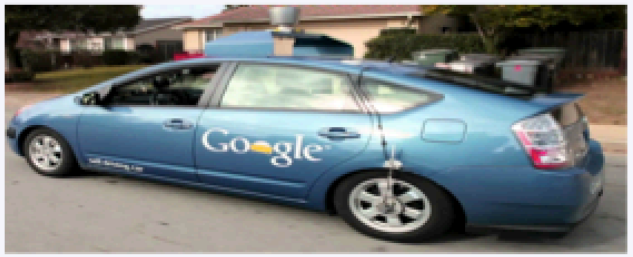
General AI

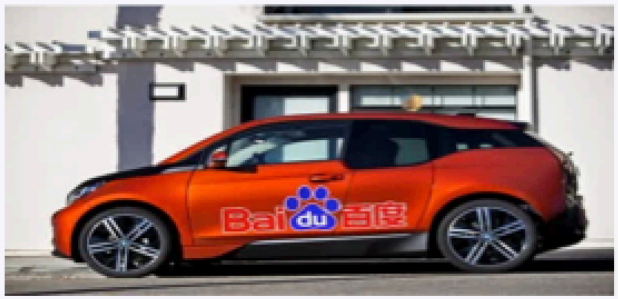
EDUCATED

Can (in principle) learn to do anything a human can learn and more.

Able to generalize dramatically beyond its original programming and training data

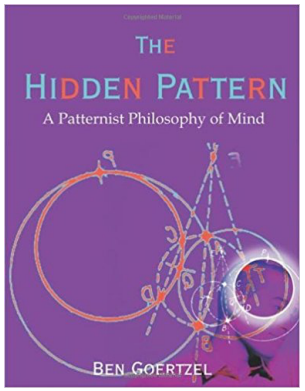
Spends significant resources on generalization rather than strict application of provided functionality

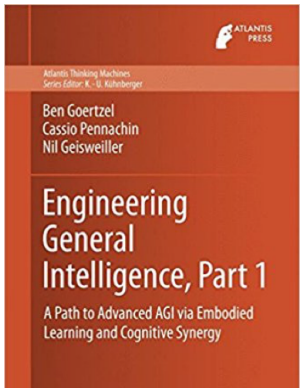


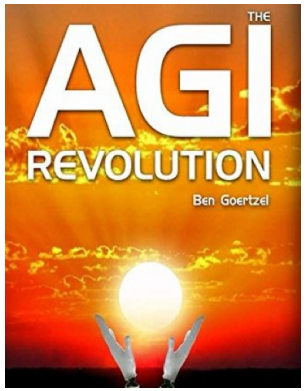


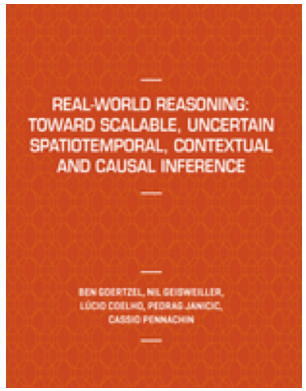
- Self-Driving Cars are a great leap forward –
- but they can't *learn* to drive different types of vehicle besides cars (trucks, boats, motorcycles)
- And they are poor at adapting to unforeseen road conditions

One approach to AGI is the open-source OpenCog AGI toolkit





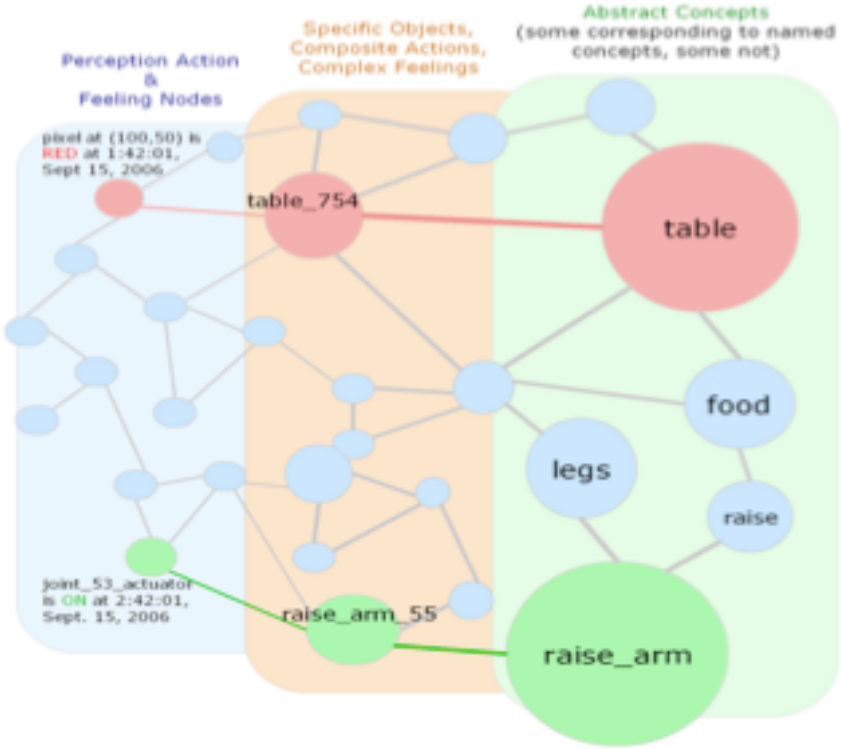




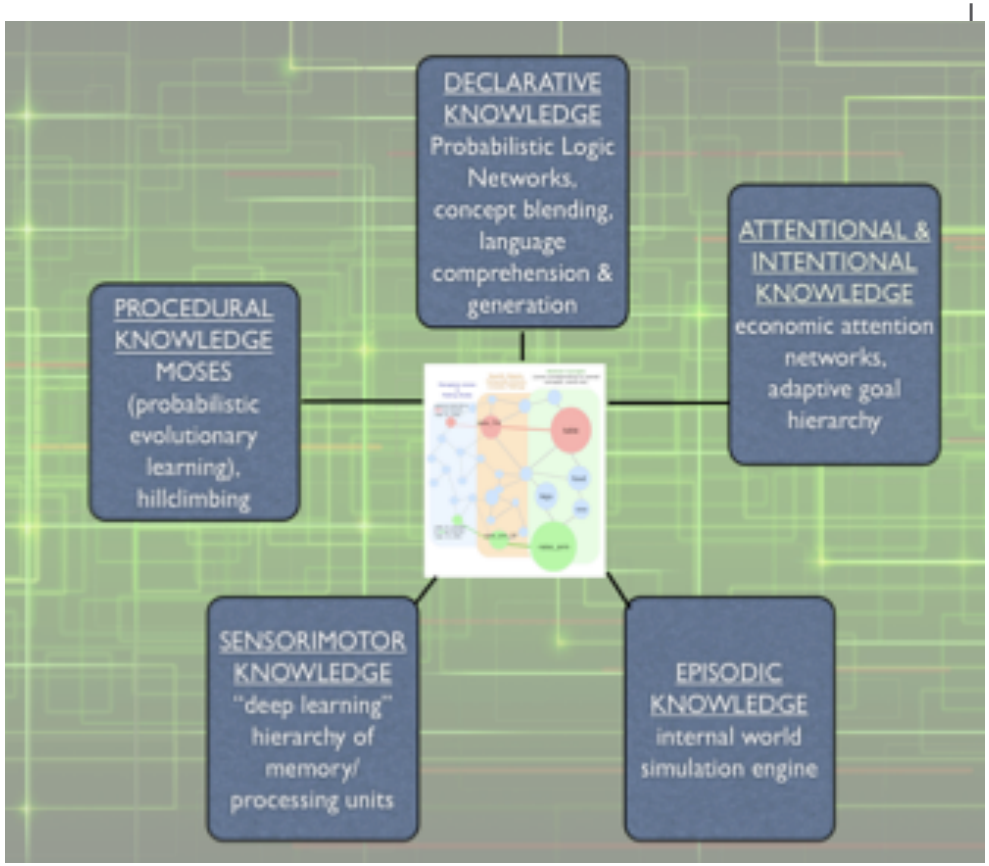
Now being embedded in the SingularityNET framework

Integrated cognitive architecture:

OpenCog uses a hypergraph knowledge store called the Atomspace as the focal point for tight integration of AI algorithms on a common dynamic knowledge representation



OpenCog features multiple cognitive algorithms, each acting on different sorts of knowledge within the common **“Atomspace”** dynamic hypergraph knowledge store - working together via **“cognitive synergy”**

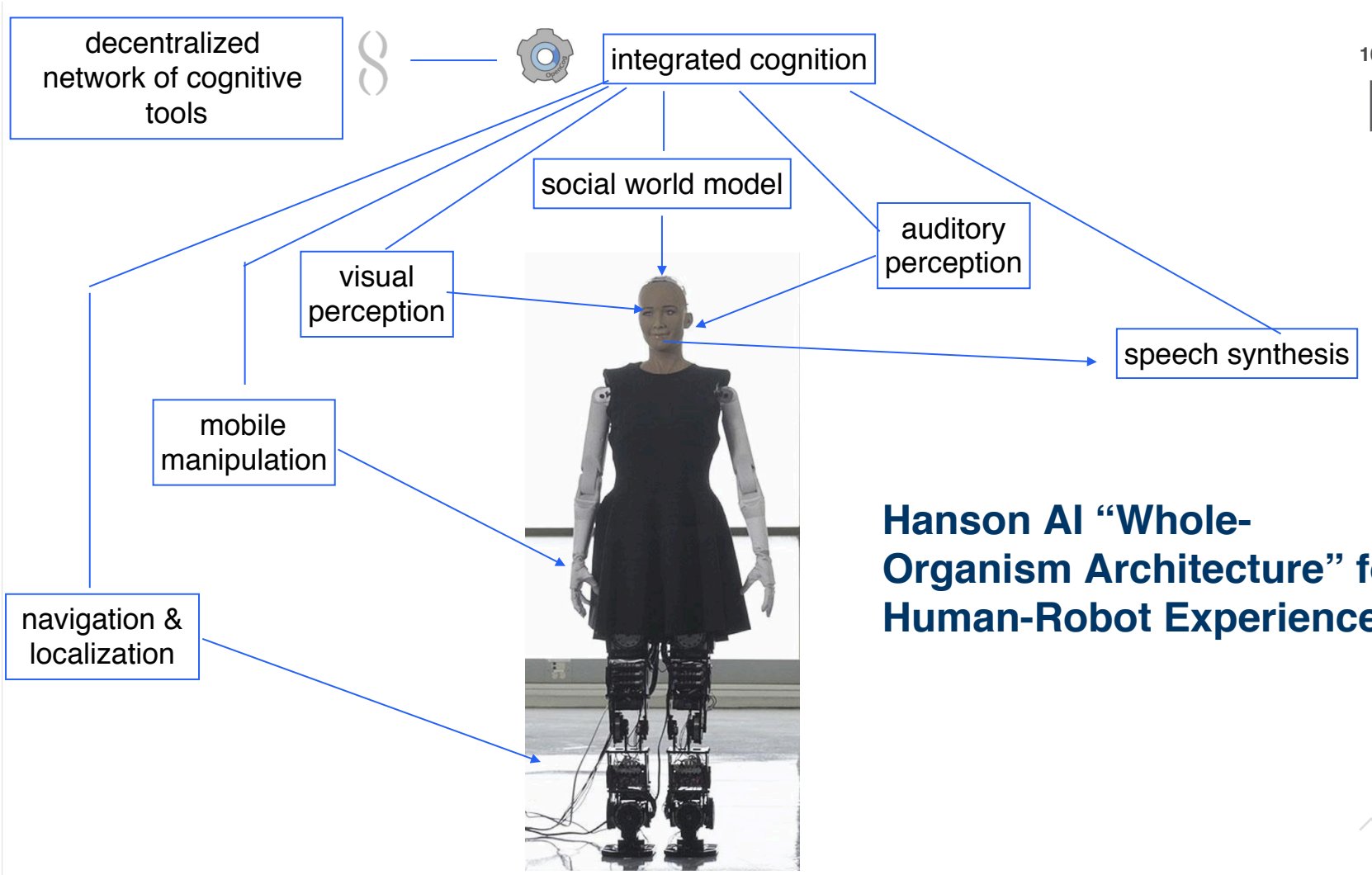


OpenCog applications

applying OpenCog to challenging practical problems
helps bridge the gap between here and AGI

The OpenCog AGI engine is already a core component underlying Sophia, **SingularityNET's humanoid spokesperson**, built by Hanson Robotics and programmed in Hanson AI Lab

OpenCog powers the genomic and biomedical inference engine underlying **Mozi AI Health**, a cutting-edge decentralized cloud service providing advanced intelligence to the discovery of therapeutics and diagnostics and the control of systems biology simulations

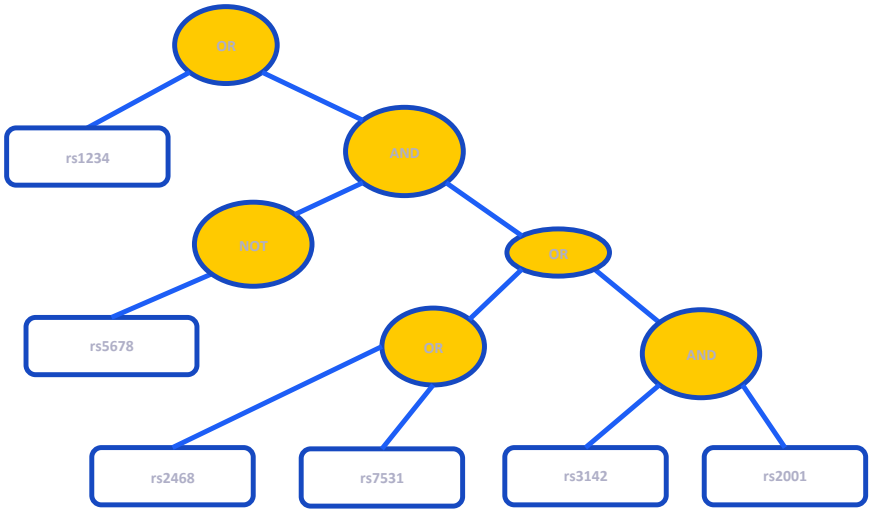


Hanson AI “Whole-Organism Architecture” for Human-Robot Experience

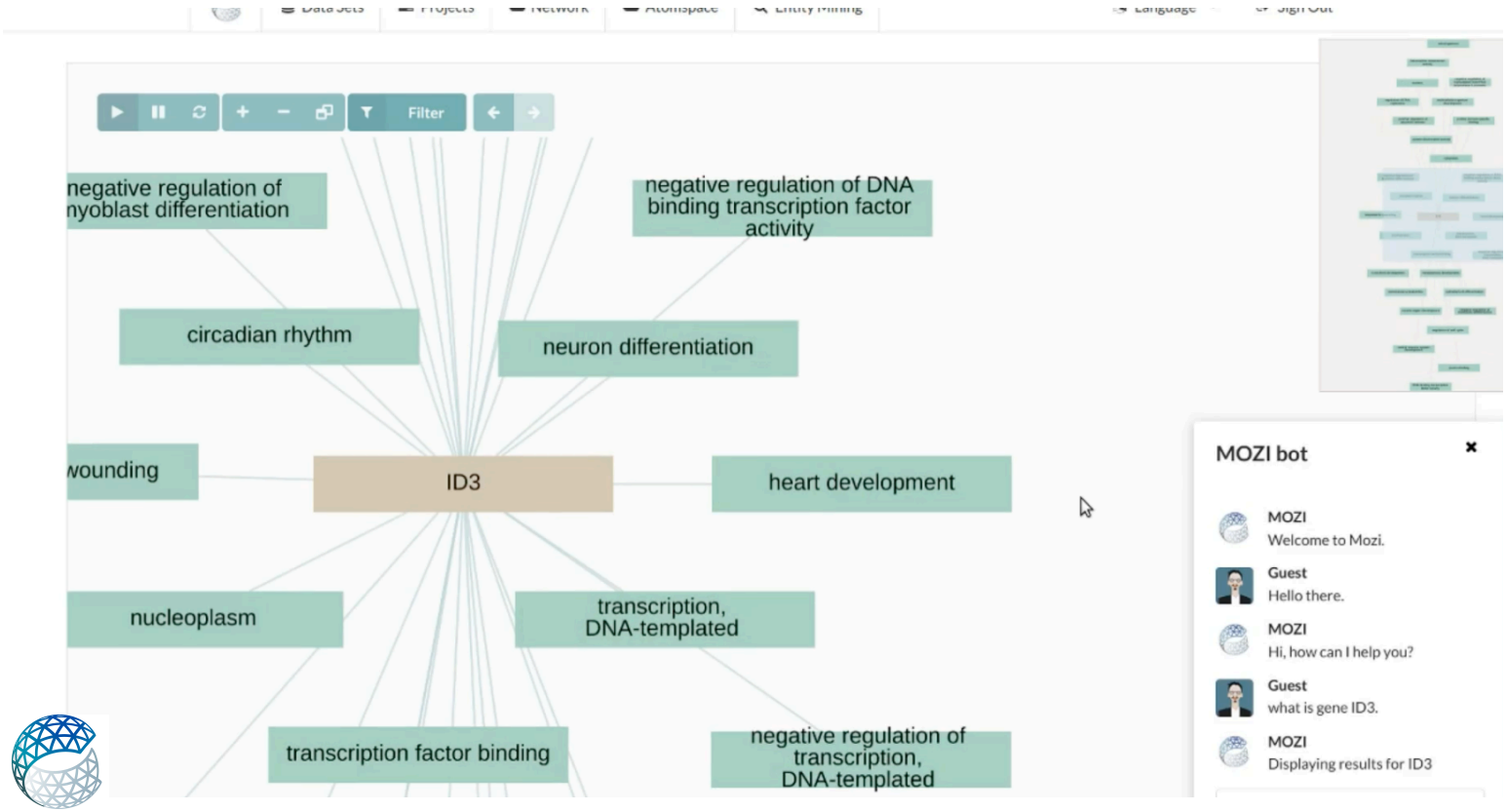
OpenCog for Genetics Data Analysis

- MOSES evolves programs coded in a simple programing language called combo.
- For analyzing categorial SNP data, binary variables are valued “0” if a sample is homozygous for the reference allele and “1” for any alternate alleles for a particular variant. A “true” value indicates “case” status.
- an example boolean combo program applied to genomic data:

or($rs1234 and(!$rs5678 or(or($rs2468 $rs7531) and($rs3142 $rs2001))))

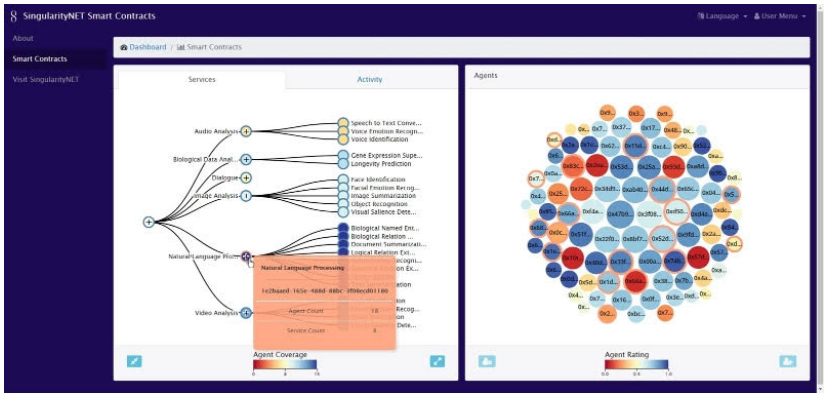


variable	sample 1	sample 2
rs1234	ref (0)	ref (0)
rs2001	ref (0)	alt (1)
rs2468	ref (0)	ref (0)
rs3142	ref (0)	alt (1)
rs5678	ref (0)	ref (0)
rs7531	ref (0)	lt (1)
program value	control (false)	case (true)



MOZI.AI
HEALTH

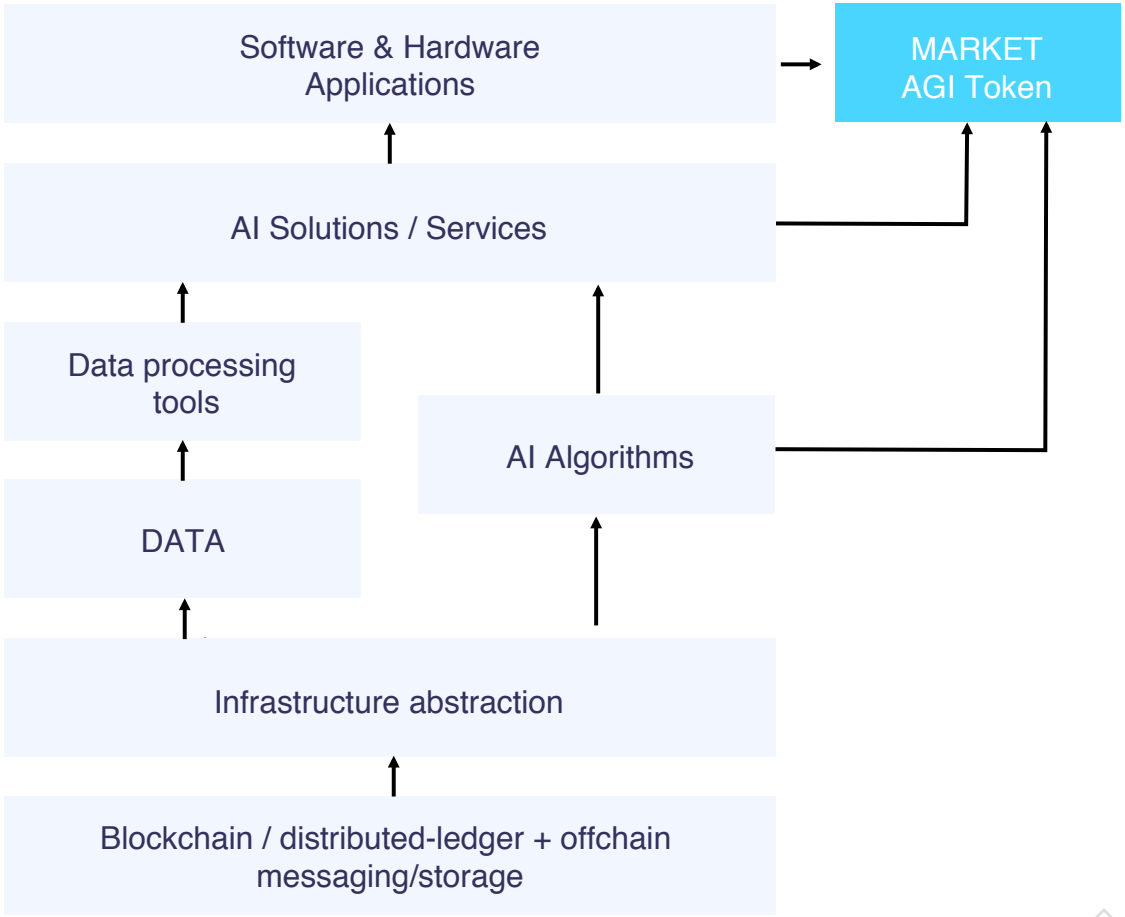
SingularityNET loosely couples multiple AI algorithms, methods and solutions, using a decentralized, blockchain-based framework — enabling them to cooperate as a society and economy of minds



Intelligence exists on the emergent level of the multi-agent network, as well as on the level of the individual AI agents in the network

OpenCog instances, neural networks, and other types of AI agents co-exist and inter-operate within the SingularityNET decentralized cognitive compute fabric

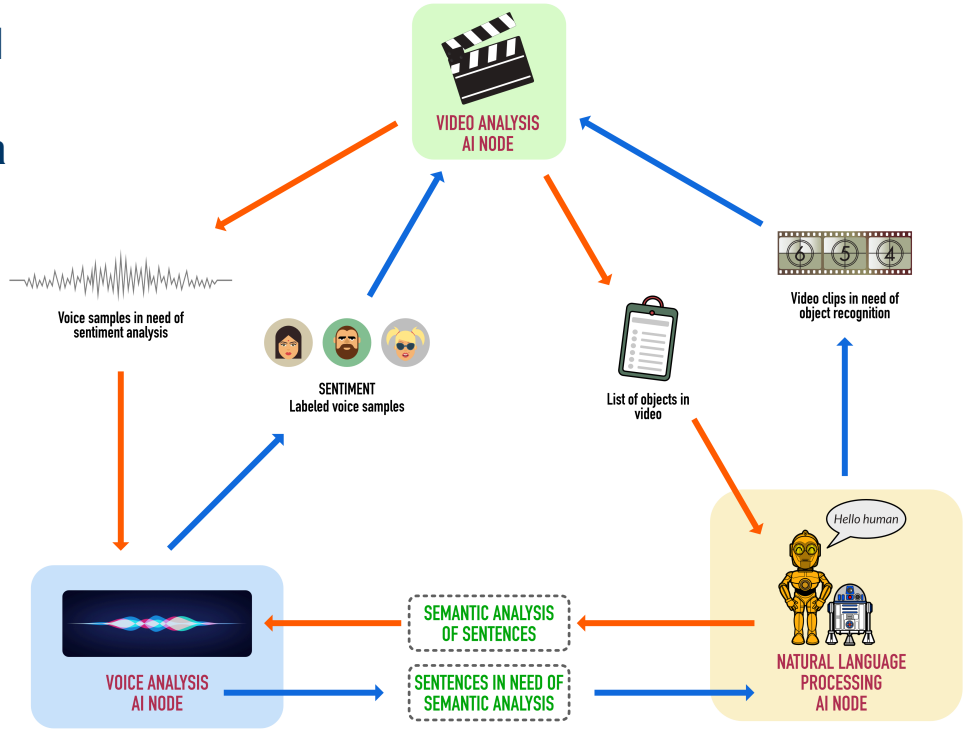
SingularityNET conceptual workflow



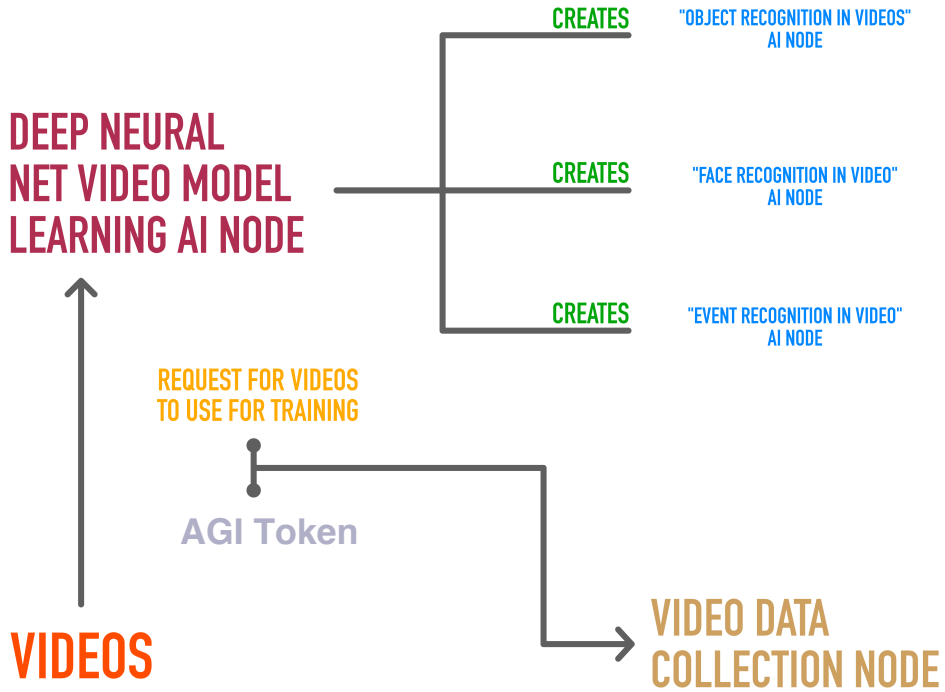
AI Nodes in the SingularityNET carry out diverse cognitive and analytic operations, and exchange diverse types of data

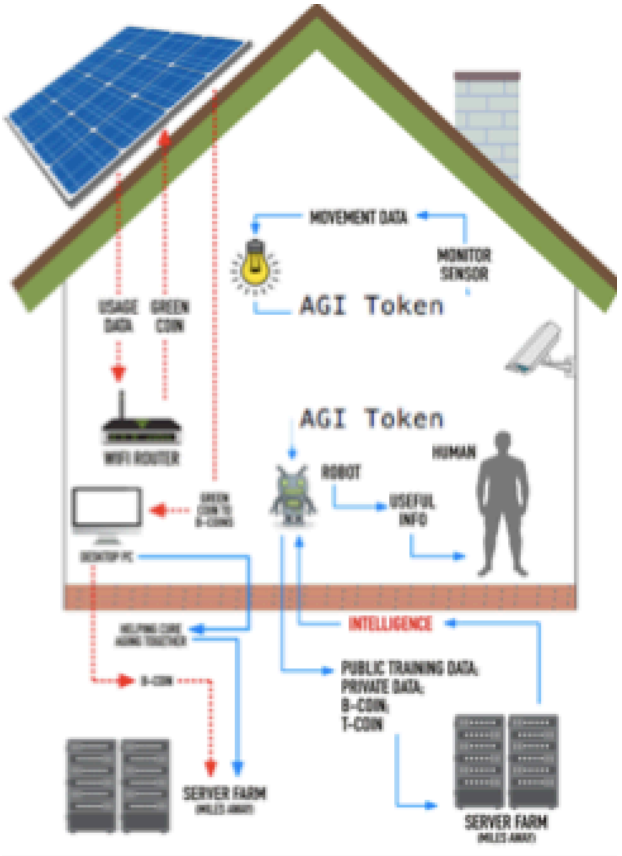
AI Nodes may join together into “federations” – subnetworks of nodes that habitually work together to carry out particular sorts of tasks.

This is a “network effect” from both business and cognitive perspectives



AI Nodes in the SingularityNET can also spawn new AI nodes – the network is **self-growing** and **self-pruning**




MOVEMENT DATA
MONITOR SENSOR
AGI Token
USAGE DATA
GREEN COIN
WIFI ROUTER
AGI Token
HUMAN
ROBOT
USEFUL INFO
DESKTOP PC
INTELLIGENCE
PUBLIC TRAINING DATA;
PRIVATE DATA;
B-COIN;
T-COIN
SERVER FARM
SERVER FARM

Shifting and drifting attention while reading: A case study of nonlinear-dynamical attention allocation in the OpenCog cognitive architecture

Article *in* Biologically Inspired Cognitive Architectures · July 2018 *with* 2 Reads
DOI: 10.1016/j.bica.2018.07.005

Cite this publication

Misgana Bayetta Belachew

Ben Goertzel

Matthew Ikle
13.23 · Adams State University

David Hanson

Abstract

A simple experimental example of the general principle of "cognitive synergy" underlying the OpenCog AGI architecture is explored: An OpenCog system processing a series of articles that shifts from one topic (insects) to another (poisons), and using its nonlinear attention-allocation dynamics (based on the ECAN Economic Attention Networks framework) to spread attention back and forth between the nodes and links within OpenCog's Atomspace knowledge store representing the words in the sentences, and other nodes and links containing related knowledge.With this setup, we study how the ECAN system shifts the attentional focus of the system based on changes in topic - in terms of both the speed of attention switching, and the contextual similarity of the content of attentional focus to the sentences being processed at a given point in time. This also provides an avenue for exploring the effects of particular design choices within the ECAN system.For instance, we find that in this particular example, if the parameters are set appropriately, ECAN indeed causes the system to assign particular importance to nodes and links related to the "insecticide" concept, when it is reading sentences about poisons in a situation where it has been primed by sentences about insects. This is an example of what we call "drifting" attention - the system's attention moves to something suggested by its perceptions, even if not directly presented in them.

Unsupervised Language Learning in OpenCog: 11th International Conference, AGI 2018, Prague, Czech Republic, August 22-25, 2018, Proceedings

Chapter · July 2018 *with* 4 Reads
DOI: 10.1007/978-3-319-97676-1_11
In book: Artificial General Intelligence, pp.109-118

Cite this publication

Authors and Editors

Alex Glushchenko

Andres Suarez

Anton Kolonin
3.82 · Aigents Group

+ 3

Oleg Baskov

Show more authors

Abstract

We discuss technology capable to learn language without supervision. While the entire goal may be too ambitious and not achievable to full extent, we explore how far we can advance grammar learning. We present the current approach employed in the open source OpenCog Artificial Intelligence Platform, describe the cognitive pipeline being constructed and present some intermediate results.

Vision System for AGI: Problems and Directions: 11th International Conference, AGI 2018, Prague, Czech Republic, August 22-25, 2018, Proceedings

Chapter · July 2018 *with* 2 Reads
DOI: 10.1007/978-3-319-97676-1_18
In book: Artificial General Intelligence, pp.185-195

Cite this publication

Authors and Editors


Alexey Potapov
19.42 · ITMO University


Sergey Rodionov
26.34 · Aix-Marseille Université

Maxim Peterson
3.74 · ITMO University

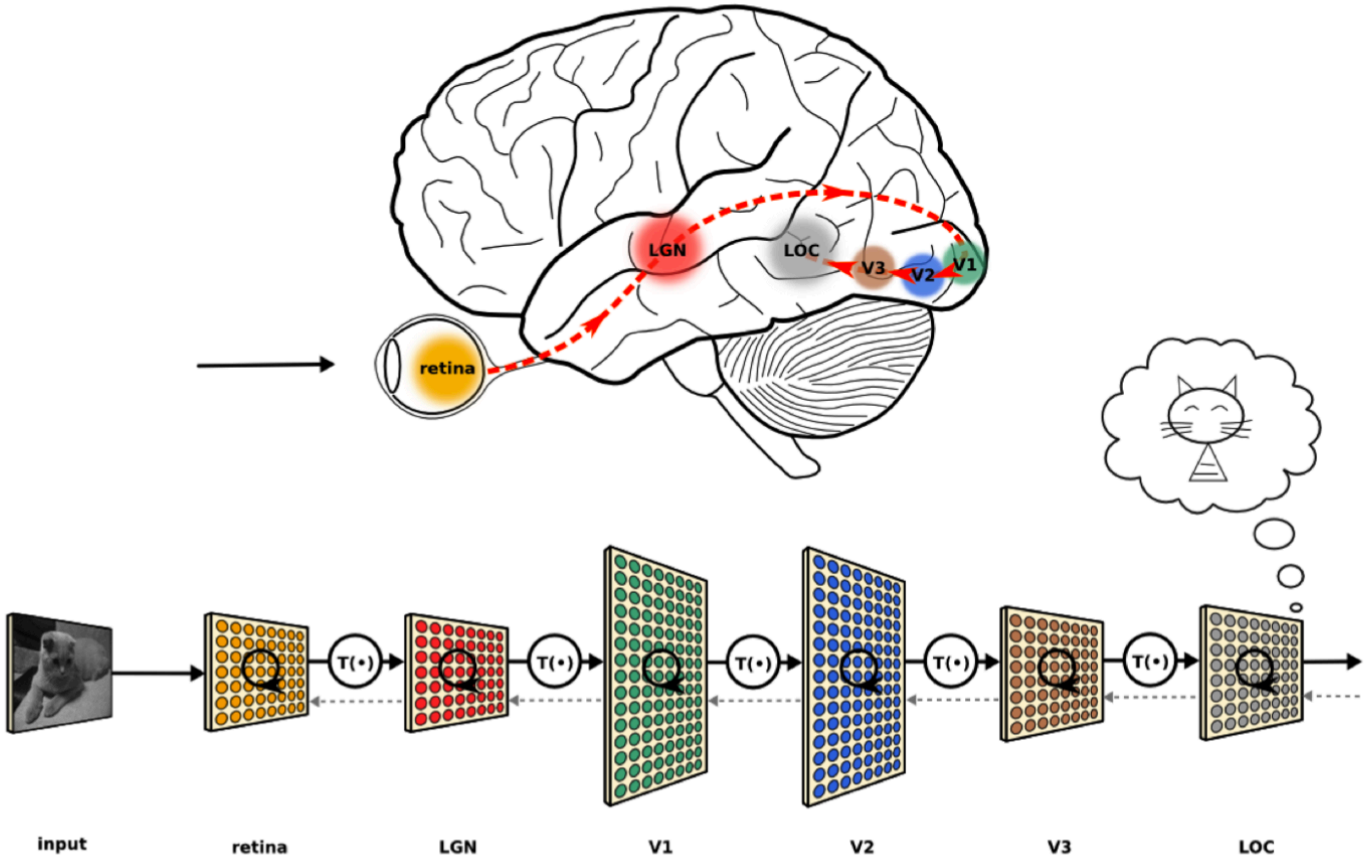
+ 2

Nikolai Skorobogatko

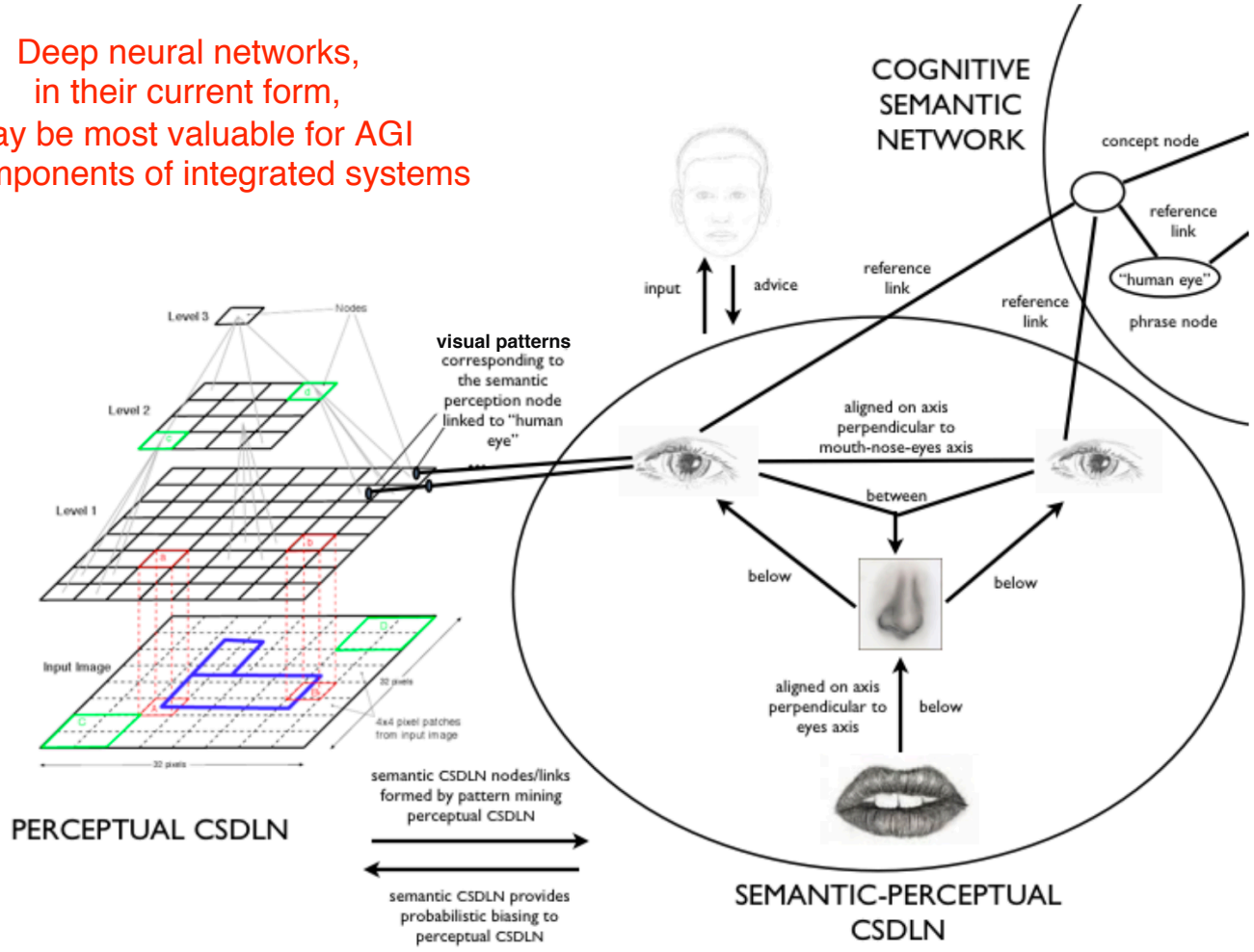
Show more authors

Abstract

What frameworks and architectures are necessary to create a vision system for AGI? In this paper, we propose a formal model that states the task of perception within AGI. We show the role of discriminative and generative models in achieving efficient and general solution of this task, thus specifying the task in more detail. We discuss some existing generative and discriminative models and demonstrate their insufficiency for our purposes. Finally, we discuss some architectural dilemmas and open questions.

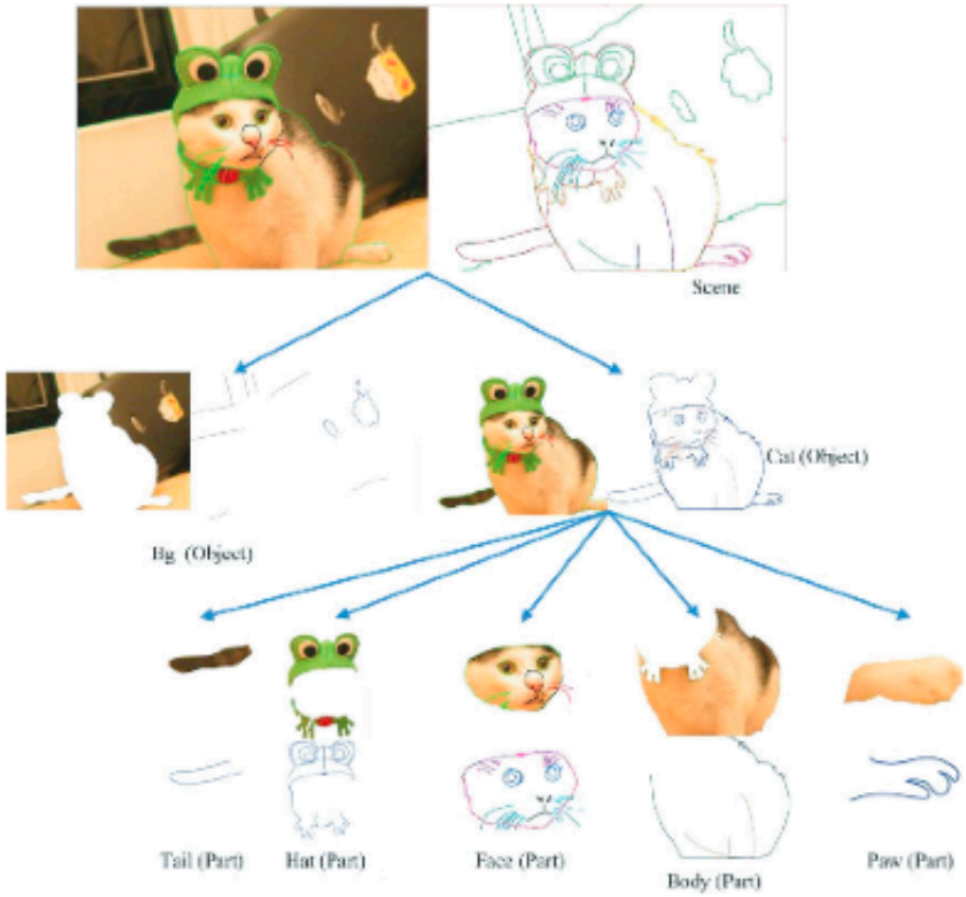

LGN
LOC
V3
V2
V1
retina
T(•)
T(•)
T(•)
T(•)
T(•)
input
retina
LGN
V1
V2
V3
LOC

Deep neural networks,
in their current form,
may be most valuable for AGI
as components of integrated systems



One research goal is to create deep neural networks that internally model observed data in a semantically sensible way —

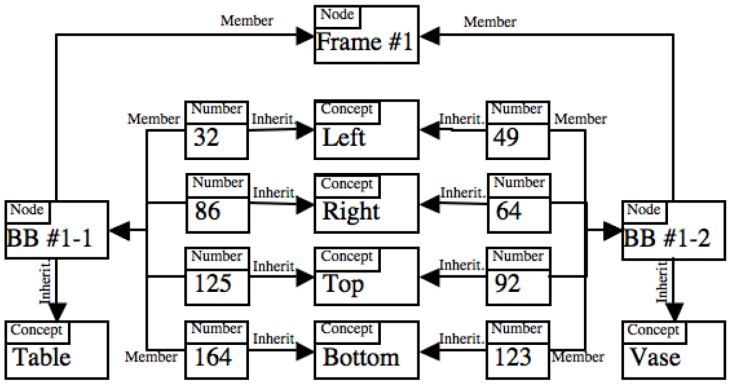
this is important for integrating deep NNs with other AI algorithms and structures effectively



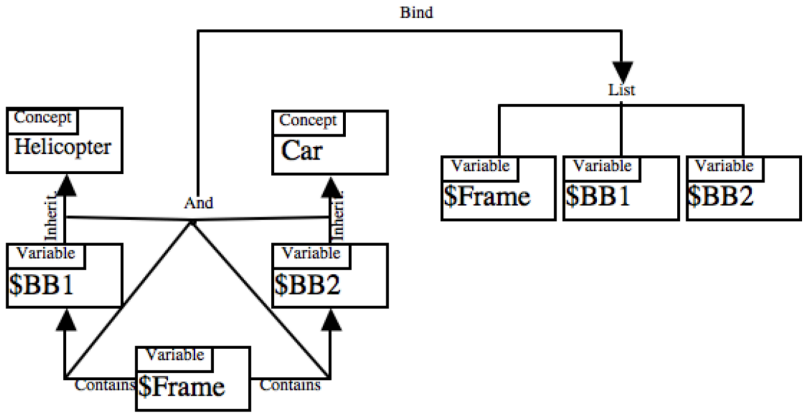
Alexey Potapov Follow
Jul 12 · 7 min read

Deep Semantic Gap to be bridged for SingularityNET

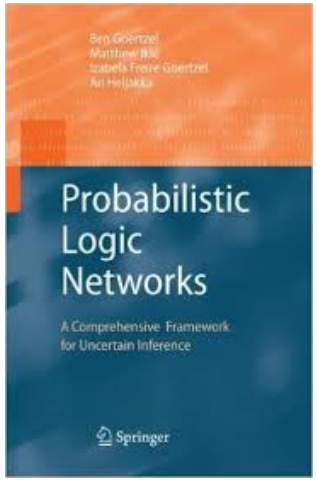
How SingularityNET is being designed to allow for more effective models of higher cognitive functions.



A fragment of the knowledge graph



A (hyper)graph of a query



Observation-based semantics:
an AI should assign meaning
to all its ideas, abstraction
and hypotheses via
extrapolation from its
observations

The AI builds its own subjective world,
self and will, coupled with the flow of observations
coming into it

Logic: very general, flexible framework for carrying out abstract reasoning.

Encompasses both mathematical and commonsense reasoning.

Probability theory: very general, flexible framework for carrying out reasoning based on uncertainty.

Used in a huge variety of areas including data mining, robotics, vision processing, etc.

“**Progic**” = probability + logic

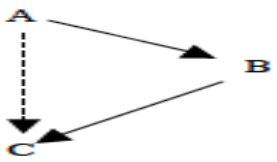
- Various approaches to synthesizing probability and logic exist
- Probabilistic Logic Networks (PLN) is a “progic” framework oriented toward artificial general intelligence.

Probabilistic Logic Networks

- OpenCog represents knowledge in its “Atomspace” in terms of nodes and links of various types
- PLN contains a set of probabilistic logic rules, that transform sets of nodes/links into other sets of nodes/links
- PLN can do deduction, induction, abduction, analogy and other types of reasoning
- PLN can reason on any kind of data, including data-patterns (“combo models”) learned in genomic data by MOSES, or data imported into OpenCog from bio-ontologies
- Due to its ability to process huge amounts of information in subtle ways, PLN can identify data patterns the human mind will miss
- A fundamentally different paradigm than currently popular “machine learning” or “deep learning” architectures, with more capability for abstract symbolic understanding – but can work together with more standard ML algorithms

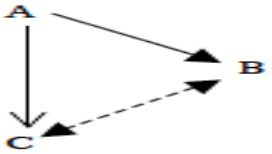
Term Logic

A → B
B → C
|–
A → C



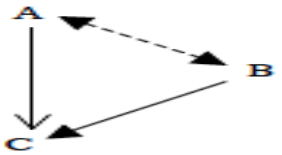
Deduction

A → B
A → C
|–
B → C



Induction

A → C
B → C
|–
A → B



Abduction

Predicate Logic

A

A --> B

|-

B

Multiple PLN Relationship Types

PLN involves more than a dozen logical relationship types, each with particular semantics.

For instance

$$A \rightarrow B$$
$$B \rightarrow C$$
$$\vdash$$
$$A \rightarrow C$$

could be interpreted in many ways including

```
ExtensionalInheritance A B
ExtensionalInheritance B C
|-
ExtensionalInheritance A C
```

```
IntensionalInheritance A B
IntensionalInheritance B C
                |-
IntensionalInheritance A C
```

"Higher-Order" PLN

Following Pei Wang's usage in NARS, in PLN we refer to logic regarding variables or higher-order functions as "higher-order"

```
ImplicationLink
EvaluationLink has($X, mouth)
EvaluationLink eats($X, food)
```

Quantifying Truth Values

Each PLN relationship has a truth value attached to it. PLN supports truth value objects of different types, e.g.

- Single probability
- SimpleTruthValue:
  - (s,c) = (probability, confidence level)
  - (s,n) = (probability, amount of evidence)
- Imprecise truth value
  - (L,U) interval, e.g. (.4,.6)
- Indefinite truth value
  - (L,U,b,k) … interval plus confidence level b, and "personality parameter" k, e.g. (.4,.6,.9,2)
- Distributional truth value
  - first or second order pdf

Example PLN rule+formula: deduction

B $<s_B>$
C $<s_C>$
ExtensionalInheritance A B $<s_{AB}>$
ExtensionalInheritance B C $<s_{BC}>$
|-
ExtensionalInheritance A C $<s_{AC}>$
$s_{AC} = s_{AB}\ s_{BC} + (1-s_{AB})\ (\ s_C - s_B\ s_{BC}\ )\ /\ (1-\ s_B\ )$

As given above, this acts on single-probability truth values. It can be extended to other true value forms.

PLN rules

Each rule maps a tuple of relationships into a relationship

Example: deduction rule

```
Subset A B
Subset B C
|-
Subset A C
```

PLN formulas

Each formula maps a tuple of truth values into a truth value

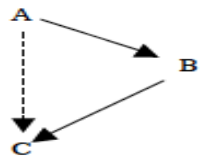
Example: deduction formula

$$s_{AC} = s_{AB}\, s_{BC} + (1 - s_{AB})\, (\, s_C - s_B\, s_{BC}\,) \,/\, (1 - s_B\,)$$

Inversion
(Bayes Rule)

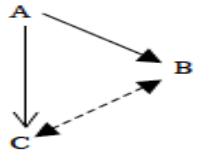
```
A
B
Subset A B
|-
Subset B A
```

A → B
B → C
|-
A → C



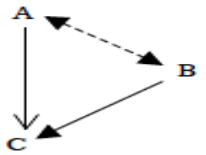
Deduction

A → B
A → C
|-
B → C



Induction

A → C
B → C
|-
A → B



Abduction

In PLN, simple first-order **induction** and **abduction** are obtained by combining deduction and Bayes rule.

More advanced induction and abduction result from using intensional relationships.

Glossary of Link Types

AttractionLink

Indicates the extent to which one concept is a pattern or property helping to characterize another.

(AttractionLink A B) indicates the extent to which B is a property that characterizes A.

ConceptNode

A node representing any concept.

ExecutionOutputLink

Indicates execution of a function with a list arguments to that function. This allows for atomspace representation of the execution of arbitrary code.

GeneNode

A node representing a particular gene.

Glossary of Link Types (cont.)

GroundedSchemaNode

Specifies the name of a predefined procedure that is to be called.

ImplicationLink

Expresses an if...then... relation, or that the truth of one predicate implies the truth of another.

(ImplicationLink A B) denotes that A implies B.

IntensionalEquivalenceLink

Indicates that the properties associated with one predicate being true are similar to the properties associated with another predicate being true.

(IntensionalEquivalanceLink A B) denotes that the properties associated with A being true are similar to the properties associated with B being true.

IntensionalImplicationLink

Expresses an if… then... relation between the *properties* of 2 predicates.

(IntensionalImplicationLink A B) denotes that the properties of A imply the properties of B.

Glossary of Link Types (cont.)

IntensionalSimilarityLink

Indicates that two concepts have similar properties.

(IntensionalSimilarityLink A B) denotes that the properties of A are similar to the properties of B

ListLink

Used for grouping Atoms for some purpose, typically to specify a set of arguments to some function or relation.

MemberLink

Indicates set membership.

(MemberLink x S) denotes that element x is a member of set S. The TruthValue associated with a MemberLink is meant to indicate fuzzy set membership.

NotLink

Corresponds to the negation of a concept or predicate.

Glossary of Link Types (cont.)

PredicateNode

Names the predicate of a relation. Predicates are functions that have arguments and produce a truth value as output.

SetLink

A type of link used to group its arguments into a set

(SetLink x y z) simply indicates that there is a set {x,y,z}

SubsetLink

Denotes extensional inheritance, which is inheritance between sets based on their members. It specifies an “is-an-instance-of” relationship.

(SubsetLink A B) specifies that A is an instance of B.

PLN for commonsense reasoning

Amusing Friend PLN Demo

PLN demo involving deductive and abductive reasoning. Self is looking for an amusing and honest friend and infers that Bob would be one based on his actions and the fact that friends tend to be honest.

Background Knowledge

Bob is a human.
I am a human.
I am honest.
I know Bob.
Friends tend to be honest.
People who told the truth about something are honest.
People who told a joke to someone, somewhere, are funny.
Being funny is loosely equivalent to being amusing.
Bob told Jill the truth about the party.
Bob told Jim a joke at the party.
Probability of two humans being acquaintances: .0002
Probability of a person being honest: .8
Probability of being funny: .69
The probability of random things (typically humans) being friends: .0001
The probablity of turning acquaintance into friendship between humans: .01

show more

Steps leading up to the conclusion: Bob will be an amusing and honest friend.

PLN + deep NNs for visual question answering

Q: What color is the plane?

A: Red

the question "*What color is the plane?*" can be automatically converted to the Relex form:

```
_det(color, _$qVar);_obj(be, plane);_subj(be, color)
```

After which, the following Pattern Matcher query can be constructed:

```
(BindLink
  (VariableList
    (TypedVariable (Variable"$B") (Type "ConceptNode"))
    (TypedVariable (Variable "$X") (Type "ConceptNode")))
  (AndLink
    (InheritanceLink (VariableNode "$B")
                     (ConceptNode "BoundingBox"))
    (InheritanceLink (VariableNode "$X") (ConceptNode "color"))
    (EvaluationLink
      (GroundedPredicateNode "py:recognize")
      (ListLink (VariableNode "$B") (ConceptNode "plane")))
    (EvaluationLink
      (GroundedPredicateNode "py:recognize")
      (ListLink (VariableNode "$B") (VariableNode "$X"))))
  (ListLink (VariableNode "$B") (VariableNode "$X")))
```

Example PLN Biological Inference: Goal

- Through MOSES analysis, we found overexpression of LY96 appears to distinguish Nonagenarians from controls.

- Using PLN, what can we infer about the relationship between LY96 and longevity based on background domain and experimental knowledge?

Our target conclusion is:

```
ImplicationLink
  (ExecutionOutputLink
     (GroundedSchemaNode "scm: make-over-expression-predicate")
     (GeneNode ”LY96"))
  (PredicateNode "LongLived”)
```

Interpretation: “Overexpression of LY96 implies longevity.”

Background Information

There is pre-existing evidence that over-expression of gene TBK1 is associated with increased lifespan (Source: Lifespan Observations Database)

```
(IntensionalImplicationLink (stv 0.3 0.7)
  (ExecutionOutputLink (stv 0.2 0.7)
    (GroundedSchemaNode "scm: make-overexpression-predicate")
    (ListLink
      (GeneNode "TBK1" (stv .0004 0.9))))
  (PredicateNode "LongLived" (stv 0.15 0.8)))
)
```

Interpretation: "Overexpression of TBK1 implies longevity"

Genes are associated with Gene Ontology terms and other categories.

```
((MemberLink
 (GeneNode "TBK1" (stv 0004 0.9))
 (ConceptNode "GO:0005515" (stv 0.001 0.9)))

(MemberLink
 (GeneNode "TBK1" (stv 0004 0.9))
 (ConceptNode "GO:0045087" (stv 0.001 0.9)))
. . .
```

Interpretation: "TBK1 is a member of GO category 0005515,"

"TBK1 is a member of GO category 0045087,"

. . . for each gene category annotation

Inference Chain Steps

(1) Member-to-Subset Rule

(Member A B) |- (Subset (Set A) B)

Premises:

```
(MemberLink
  (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998))
  (ConceptNode "GO:0051607" (stv 0.001 0.89999998))
)
. . .
```

“TBK1 is a member of GO category 0051607”

Conclusions:

```
(SubsetLink
  (SetLink
    (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998))
  )
  (ConceptNode "GO:0051607" (stv 0.001 0.89999998))
)
. . .
```

“The singleton set containing TBK1 is a subset of GO category 0051607”

Intensional Similarity

- We will infer a relationship between the gene LY96 and the predicate LongLived through the similarity of LY96 with gene TBK1, which is already known to be related to longevity.
- Intensional similarity is based on common properties of the genes.
- Steps 2-5 that follow are needed for creating the IntensionalSimilarity relationship.

(2) Compare gene properties

- We are using GO category annotations for gene properties.
- At the start of the inference, we need to get the supersets of {TBK1} and {LY96} and determine the intersection and union of the supersets

LY96: member of 25 GO categories

TBK1: member of 34 GO categories

Common categories (intersection):

GO:0005515 protein binding

GO:0045087 innate immune response

GO:0006954 inflammatory response

GO:0010008 endosome membrane

GO:0002224 toll-like receptor signaling pathway

GO:0002756 MyD88-independent toll-like receptor signaling pathway

GO:0007249 I-kappaB kinase/NF-kappaB signaling

GO:0034138 toll-like receptor 3 signaling pathway

GO:0034142 toll-like receptor 4 signaling pathway

GO:0035666 TRIF-dependent toll-like receptor signaling pathway

(3) Subset NotA B Direct Evaluation

(Inheritance A B) |- (Inheritance (Not A) B)

For each common category relationship (LinkType A B), create (LinkType (Not A) B)

Premises:

```
(SubsetLink (stv 1 0.99999982)
  (SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
  (ConceptNode "GO:0045087" (stv 0.001 0.89999998))
)
. . .
```

“{LY96} is a subset of GO:0045087”

Conclusions:

```
(SubsetLink (stv 0.028667862 0.99999982)
  (NotLink
    (SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
  )
  (ConceptNode "GO:0045087" (stv 0.001 0.89999998)))
. . .
```

“A random gene (exclusive of LY96) belongs to GO:0045087 (with a low probability)”

(4) AttractionRule

(And (Subset A B) (Subset (Not A) B)) |- (AttractionLink A B)

Make AttractionLinks for LY96 and TBK1 for each common relationship (IOW for each relationship in the intersection of the supersets).

Premises:

```
(SubsetLink (stv 1 0.99999982)
  (SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
  (ConceptNode "GO:0045087" (stv 0.001 0.89999998))
)
(SubsetLink (stv 0.028667862 0.99999982)
  (NotLink
    (SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
  )
  (ConceptNode "GO:0045087" (stv 0.001 0.89999998)))
. . .
```

{LY96} is a subset of "GO:0045087,"

"A random gene not in {LY96} is a subset of GO:0045087 (with a low probability)"

Conclusions:

```
(AttractionLink (stv 0.97133213 0.99999982)
  (SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
  (ConceptNode "GO:0045087" (stv 0.001 0.89999998)))
. . .
```

"GO:0045087 is a property of/pattern in {LY96)"

(5)IntensionalSimilarity Direct Evaluation

(And (Attraction P A) (Attraction P B) (Attraction (Q A) (Attraction (Q B) …) |- (IntensionalSimilarity A B)

Premises:

(AttractionLink (stv 0.97133213 0.99999982)
(SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
(ConceptNode "GO:0045087" (stv 0.001 0.89999998)))

(AttractionLink (stv 0.97133213 0.99999982)
(SetLink (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998)))
(ConceptNode "GO:0045087" (stv 0.001 0.89999998)))
. . .

“GO:0045087 is a property of {LY96}”
“GO:0045087 is a property of {TBK1}”
Etc. . . .

Conclusion:

(IntensionalSimilarityLink (stv 0.19570713 0.99999982)
(SetLink (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998)))
(SetLink (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))

“{TBK1} properties are similar to {LY96} properties”

(6) Singleton-Similarity-Rule

(Similarity {A} {B}) |- (Similarity A B)

Premise:

```
(IntensionalSimilarityLink (stv 0.19570713 0.99999982)
  (SetLink
    (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998)))
  (SetLink
    (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))
```

“{TBK1} properties are similar to {LY96} properties”

Conclusion:

```
(IntensionalSimilarityLink (stv 0.19570713 0.99999982)
    (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998))
    (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))
  )
```

“TBK1 properties are similar to LY96 properties”

(7) Gene-Similarity-to-Overexpression-Equivalence

(Similarity (Gene A) (Gene B)) |- (Equivalence (A-overexpressed) (B-overexpressed)

Premise:

```
(IntensionalSimilarityLink (stv 0.19570713 0.99999982)
    (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998))
    (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))
  )
```

“TBK1 properties are similar to LY96 properties”

Conclusion:

```
(IntensionalEquivalenceLink (stv 0.19570713 0.99999982)
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998))))
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))))
```

“Properties associated with over-expression of TBK1 are similar to properties associated with overexpression of LY96”

(8) Equivalence-Transformation Rule

(Equivalence A B) |- (And (Implication A B) (Implication B A))

Premise:

```
(IntensionalEquivalenceLink (stv 0.19570713 0.99999982)
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998))))
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "LY96" (stv 4.1666666e-05 0.89999998)))))
```

" 'Overexpression of TBK1 properties' is similar to 'overexpression of RYR1 properties' "

Conclusion:

```
(IntensionalImplicationLink (stv 0.3273496 0.99999982)
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))))
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998)))))
```

"Having properties associated with over-expression of LY96 implies having properties associated with overexpression of TBK1"

(9) Implication Deduction Rule

(And (Implication A B) (Implication B C) |- (Implication A C)
(Part 1)

Premises:

```
(IntensionalImplicationLink (stv 0.3273496 0.99999982)
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))))
    (ExecutionOutputLink (stv 0.2 0.69999999)
      (GroundedSchemaNode "scm: make-overexpression-predicate")
      (ListLink
        (GeneNode "TBK1" (stv 4.1666666e-05 0.89999998)))))
```

“Having properties associated with overexpression of LY96, implies having properties associated with overexpression of TBK1”

```
(IntensionalImplicationLink (stv 0.3 0.7)
  (ExecutionOutputLink (stv 0.2 0.7)
    (GroundedSchemaNode "scm: make-overexpression-predicate")
    (ListLink
      (GeneNode "TBK1" (stv .0004 0.9))))
  (PredicateNode "LongLived" (stv 0.15 0.8)))
)
```

“Having properties associated with overexpression of TBK1, implies having properties associated with longevity”

(9) Implication Deduction Rule

(And (Implication A B) (Implication B C) |- (Implication A C)
(Part 2)

Conclusion:

```
(IntensionalImplicationLink (stv 0.17387806 0.69999999)
   (ExecutionOutputLink (stv 0.2 0.69999999)
     (GroundedSchemaNode "scm: make-overexpression-predicate")
     (ListLink
       (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))
     )
   )
   (PredicateNode "LongLived" (stv 0.15000001 0.80000001))
 )
```

“Having properties associated with ‘Overexpression of LY96’ implies having properties associated with longevity”

(10) Implication Conversion Rule

(IntensionalImplication A B) |- (Implication A B)

Premise:

```
(IntensionalImplicationLink (stv 0.17387806 0.69999999)
   (ExecutionOutputLink (stv 0.2 0.69999999)
     (GroundedSchemaNode "scm: make-overexpression-predicate")
     (ListLink
       (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))))
   (PredicateNode "LongLived" (stv 0.15000001 0.80000001))
 )
```

“Having properties associated with ‘Overexpression of LY96’ implies having properties associated with longevity”

Conclusion:

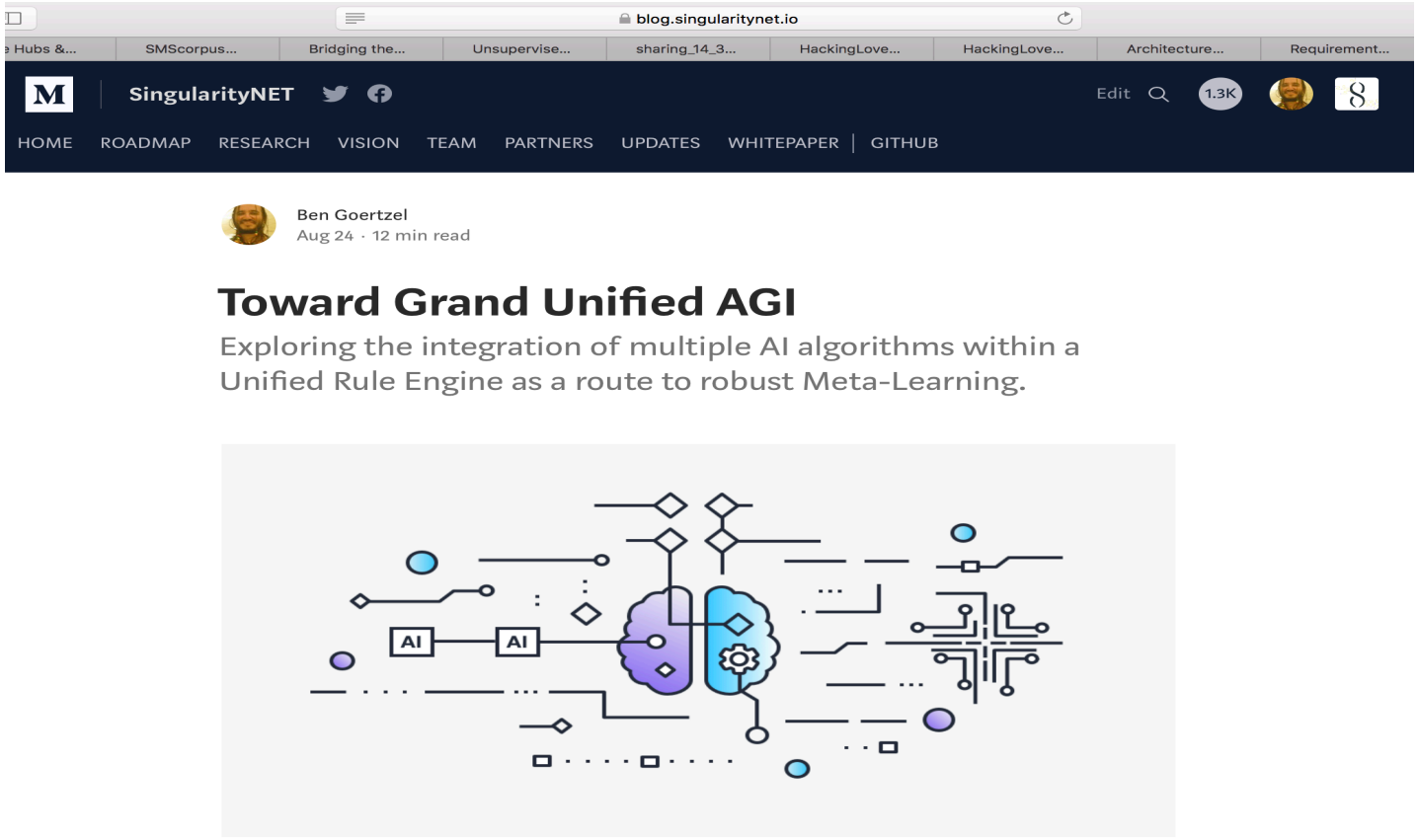
```
(ImplicationLink (stv 0.17387806 0.48999998)
   (ExecutionOutputLink (stv 0.2 0.69999999)
     (GroundedSchemaNode "scm: make-overexpression-predicate")
     (ListLink
       (GeneNode "LY96" (stv 4.1666666e-05 0.89999998))))
   (PredicateNode "LongLived" (stv 0.15000001 0.80000001))
)
```

“Overexpression of LY96 implies longevity” (Our target conclusion)

Next big AI challenge here: Fully automated, scalable inference control (choice of which inference steps to take), via data-mining of inference history

Adaptive Inference Control — one missing link to AGI

Toward Grand Unified AGI



From Here To AGI

From Here to AGI via A Few Concrete Steps

To get from where we are now to AI systems with general intelligence at the human level, and beyond, advances on multiple levels will be needed; at the very least:

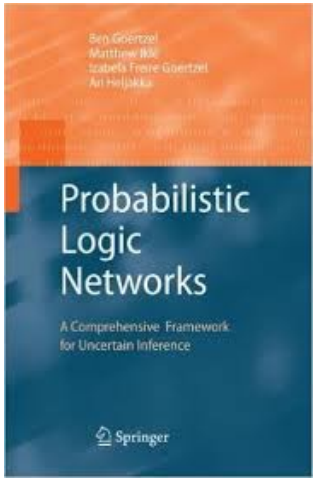
1. More effective and fluid interconnection of different AIs into processing networks (which is the core thing the SingularityNET technical design solves).
2. Incentivising of more developers to work on AGI rather than purely on AI that serves the business models of large tech companies and military/intel agencies (again something SingularityNET addresses from an economic/organizational perspective).
3. Rich interoperation of sensory, motoric and cognitive subsystems, so that AIs can learn better from the natural and human world (SingularityNET is working on this via its collaboration with Hanson Robotics).
4. Effective meta-learning—general-purpose learning algorithms that can learn how to learn better, learn how to best carry out learning in particular contexts, learn how to learn and also *learn how to* learn how to learn.

Singularity Net

HANSON ROBOTICS

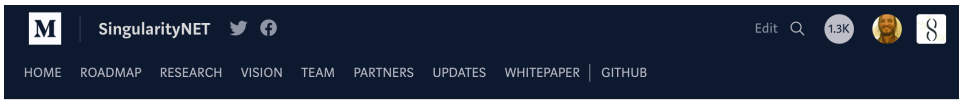
Probabilistic Logical Inference has potential to serve at the core of a highly powerful, impressively rational AGI system

But we need to solve the problem of adaptive inference control — making systems that learn how to choose what logic steps to do, in what contexts, based on their experience doing reasoning in various other contexts with varying levels and types of similar to the current context



This sort of work is at the boundary of AGI, probabilistic programming, hypergraph pattern mining, and automated theorem proving, etc. etc.

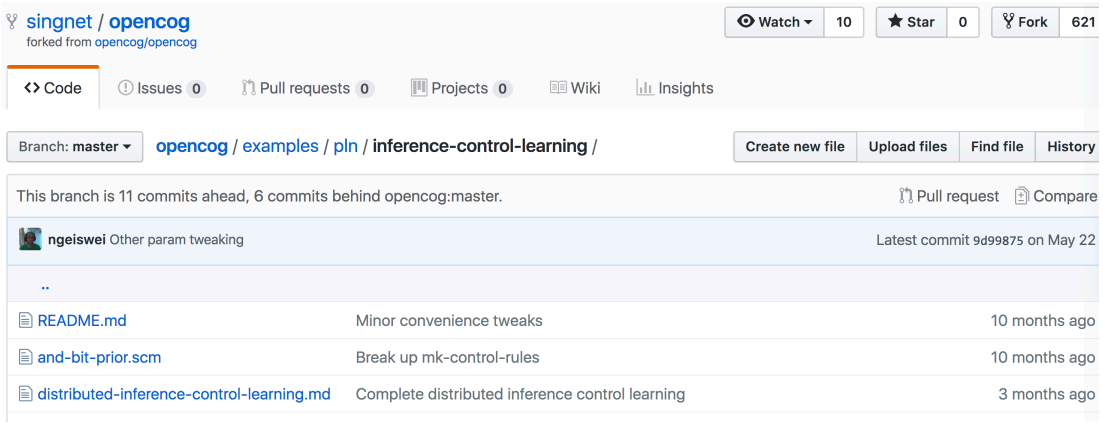
Inference meta-learning!



Nil Geisweiller Follow
OpenCog Foundation, SingularityNET Foundation developer and researcher.
Jun 15 · 8 min read

Introspective Reasoning within the OpenCog Framework

How SingularityNET leverages the OpenCog Framework to bridge both the human and machine mind, and addresses the challenges of effective reasoning.

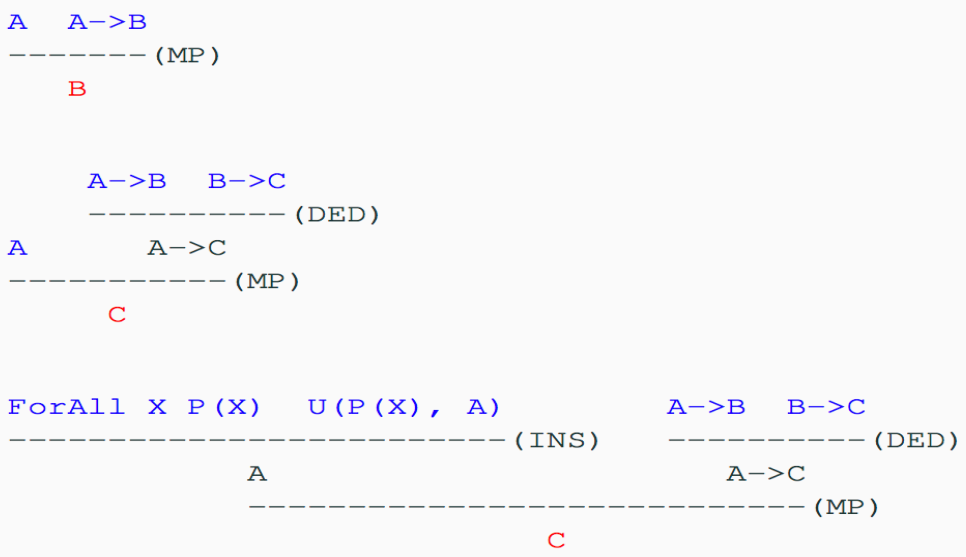


Path to Practical Metalearning in OpenCog

- Key learning algorithms have been, or are being, re-implemented in a common rule-engine framework (the URE: Unified Rule Engine):
  - PLN Probabilistic Logic Engine
  - MOSES Evolutionary Learning
  - Pattern Mining
  - OpenPsi Motivated Action Selection
  - Natural Language Comprehension/Generation
  - Simulation Modeling
- Meta-learning is being implemented for URE, with PLN as initial application case
  - This allows meta-learning to be implemented just once
  - This allows meta-learning to occur across multiple cognitive algorithms

URE: OpenCog Universal Rule Engine

The URE is a tool to evolve Inference Trees



- Leaves are premises
- Roots are conclusions

URE: OpenCog Universal Rule Engine

Inference Trees are constructed by composing Rules

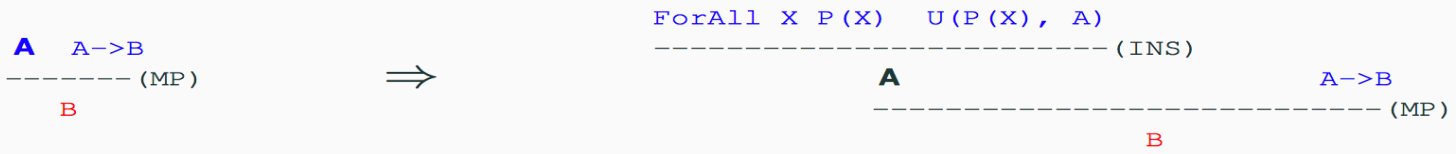
- forward: expand conclusion

URE: OpenCog Universal Rule Engine

Inference Trees are constructed by composing Rules

- forward: expand conclusion

- backward: expand premises



URE: OpenCog Universal Rule Engine

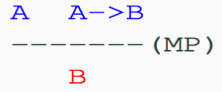
Inference Trees are Atomese Programs

```
A   A->B
--------(MP)
    B
```

```
(BindLink
  (VariableList                                                    \
    (TypedVariableLink                                             |
      (VariableNode "$A")                                          |
      (TypeChoice                                                  |
        (TypeNode "LambdaLink")                                    |
        (TypeNode "PredicateNode")                                 |
      )                                                            |
    )                                                              | Variables
    (TypedVariableLink                                             |
      (VariableNode "$B")                                          |
      (TypeChoice                                                  |
        (TypeNode "LambdaLink")                                    |
        (TypeNode "PredicateNode")                                 |
      )                                                            |
    )                                                              |
  )                                                                /
  (AndLink                                                         \
    (ImplicationLink                                               |
      (VariableNode "$A")                                          | Pattern
      (VariableNode "$B")                                          |
    )                                                              /
    (EvaluationLink                                                \
      (GroundedPredicateNode "scm: true-enough")                   |
      (ImplicationLink                                             |
        (VariableNode "$A")                                        |
        (VariableNode "$B")                                        |
      )                                                            |
    )                                                              | Preconditions
    (EvaluationLink                                                |
      (GroundedPredicateNode "scm: true-enough")                   |
      (VariableNode "$A")                                          |
    )                                                              |
  )                                                                /
  (ExecutionOutputLink                                             \
    (GroundedSchemaNode "scm: modus-ponens-formula")               |
    (ListLink                                                      |
      (VariableNode "$B")                                          |
      (VariableNode "$A")                                          |
      (ImplicationLink                                             | Production
        (VariableNode "$A")                                        |
        (VariableNode "$B")                                        |
      )                                                            |
    )                                                              |
```

URE: OpenCog Universal Rule Engine

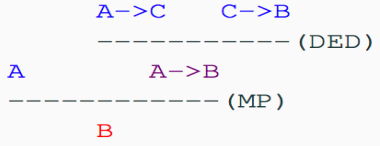
Inference Trees are Atomese Programs




```
(ExecutionOutputLink
  (GroundedSchemaNode "scm: modus-ponens-formula")
  (ListLink
    (VariableNode "$B")
    (VariableNode "$A")
    (ImplicationLink
      (VariableNode "$A")
      (VariableNode "$B")
    )
  )
)
```

URE: OpenCog Universal Rule Engine

Inference Trees are Atomese Programs




```
(ExecutionOutputLink
  (GroundedSchemaNode "scm: modus-ponens-formula")
  (ListLink
    (VariableNode "$B")
    (VariableNode "$A")
    (ExecutionOutputLink
      (GroundedSchemaNode "scm: deduction-formula")
      (ListLink
        (ImplicationLink
          (VariableNode "$A")
          (VariableNode "$B")
        )
        (ImplicationLink
          (VariableNode "$A")
          (VariableNode "$C")
        )
        (ImplicationLink
          (VariableNode "$C")
          (VariableNode "$B")
        )
      )
    )
  )
)
```

URE: OpenCog Universal Rule Engine

Inference Trees are Atomese Programs

```
A   A->B
-------(MP)
   B
```

```
(ExecutionOutputLink
  (GroundedSchemaNode "scm: modus-ponens-formula")
  (ListLink
    (VariableNode "$B")
    (VariableNode "$A")
    (ImplicationLink
      (VariableNode "$A")
      (VariableNode "$B")
    )
  )
)
```

URE: OpenCog Unified Rule Engine

Algorithm

1. Select an inference tree to expand
2. Select a node from that tree to expand
3. Select a rule to expand with
4. Expand the inference tree and place it back to the pool of inference trees. Repeat till termination.

!!! Combinatorial Explosion !!!

Inference Control

Delegate the hard decisions to a cognitive process

Cognitive Schematics:

Context & Action ⇒ Goal

```
Implication <TV>
        And
        <Context>
              <Action>
              <Goal>
```

Inference Control

Which rule to choose?

1.Modus Ponens

2.Universal Instantiation

```
A   A->B
-------(MP)
   B
```

Look for:

```
Implication <TV>
        And
    <inference-tree-pattern>
        <node-pattern>
        <rule-pattern>
    <produce-good-inference>
```

Inference Control

Examples of actual URE Cognitive Schematics

```
(ImplicationScopeLink (stv 0.45945946 0.04625)
   (VariableList
      (VariableNode "$T")
      (TypedVariableLink
         (VariableNode "$A")
         (TypeNode "DontExecLink")
      )
      (VariableNode "$L")
      (TypedVariableLink
         (VariableNode "$B")
         (TypeNode "DontExecLink")
      )
   )
   (AndLink
      (EvaluationLink
         (PredicateNode "URE:BC:preproof-of")
         (ListLink
            (VariableNode "$A")
            (VariableNode "$T")
         )
      )
      (ExecutionLink
         (SchemaNode "URE:BC:expand-and-BIT")
         (ListLink
            (VariableNode "$A")
            (VariableNode "$L")
            (DontExecLink
               (DefinedSchemaNode "deduction-inheritance-rule")
            )
         )
         (VariableNode "$B")
      )
   )
   (EvaluationLink
      (PredicateNode "URE:BC:preproof-of")
      (ListLink
         (VariableNode "$B")
         (VariableNode "$T")
      )
   )
)
```

```
(ImplicationScopeLink (stv 1 0.00625)
   (VariableList
      (VariableNode "$T")
      (TypedVariableLink
         (VariableNode "$A")
         (TypeNode "DontExecLink")
      )
      (VariableNode "$X")
      (TypedVariableLink
         (VariableNode "$B")
         (TypeNode "DontExecLink")
      )
   )
   (AndLink
      (EvaluationLink
         (PredicateNode "URE:BC:preproof-of")
         (ListLink
            (VariableNode "$A")
            (VariableNode "$T")
         )
      )
      (ExecutionLink
         (SchemaNode "URE:BC:expand-and-BIT")
         (ListLink
            (VariableNode "$A")
            (InheritanceLink
               (ConceptNode "a")
               (VariableNode "$X")
            )
            (DontExecLink
               (DefinedSchemaNode "conditional-full-instantiation-implication-scope-meta-rule"
            )
         )
         (VariableNode "$B")
      )
   )
   (EvaluationLink
      (PredicateNode "URE:BC:preproof-of")
      (ListLink
         (VariableNode "$B")
         (VariableNode "$T")
      )
   )
```

Inference Meta-Learning in OpenCog

How to get Cognitive Schematics?

Answer: Learning

Record a trace of all decisions the URE takes and learn from it

Inference Meta-Learning in OpenCog

How to learn Cognitive Schematics in OpenCog?

1. PLN Logic
2. Pattern Mining *(greedy learning driven by informational surprisingness)*
3. MOSES *(probabilistic modeling driven evolutionary learning)*
4. ??

All these algorithms are implemented using URE, enabling repeatedly nested recursive meta-learning

Inference Meta-Learning in OpenCog: Example URE Trace

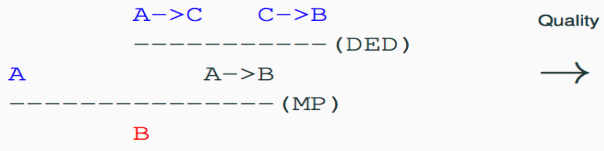
Atomese:

```
(ExecutionLink (stv 1 1)
   (SchemaNode "URE:BC:expand-and-BIT")
   (ListLink
      (DontExecLink
         (BindLink
            (VariableNode "$A-591b9032")
            (AndLink
               (VariableNode "$A-591b9032")
               (InheritanceLink
                  (VariableNode "$A-591b9032")
                  (InheritanceLink (stv 1 1)
                     (ConceptNode "v" (stv 0.84615385 1))
                     (ConceptNode "w")
                  )
               )
               (EvaluationLink
                  (GroundedPredicateNode "scm: gt-zero-confidence")
                  (VariableNode "$A-591b9032")
               )
               (EvaluationLink
                  (GroundedPredicateNode "scm: gt-zero-confidence")
                  (InheritanceLink
                     (VariableNode "$A-591b9032")
                     (InheritanceLink (stv 1 1)
                        (ConceptNode "v" (stv 0.84615385 1))
                        (ConceptNode "w")
                     )
                  )
               )
               (ExecutionOutputLink
                  (GroundedSchemaNode "scm: modus-ponens-formula")
                  (ListLink
                     (InheritanceLink (stv 1 1)
                        (ConceptNode "v" (stv 0.84615385 1))
                        (ConceptNode "w")
                     )
                     (InheritanceLink
                        (VariableNode "$A-591b9032")
                        (InheritanceLink (stv 1 1)
                           (ConceptNode "v" (stv 0.84615385 1))
                           (ConceptNode "w")
                        )
                     )
                     (VariableNode "$A-591b9032")
                  )
               )
            )
         )
         (VariableNode "$A-1860e3b6")
         (DontExecLink
            (DefinedSchemaNode "deduction-inheritance-rule")
         )
      )
   )
```

```
(DontExecLink
   (BindLink
      (VariableList
         (TypedVariableLink
            (VariableNode "$A-2fbf17d5")
            (TypeChoice
               (TypeNode "ConceptNode")
               (TypeNode "AndLink")
               (TypeNode "OrLink")
               (TypeNode "NotLink")
            )
         )
         (TypedVariableLink
            (VariableNode "$C-2bdc1299")
            (TypeChoice
               (TypeNode "ConceptNode")
               (TypeNode "AndLink")
               (TypeNode "OrLink")
               (TypeNode "NotLink")
            )
         )
         (TypedVariableLink
            (VariableNode "$B-4a714efb")
            (TypeChoice
               (TypeNode "ConceptNode")
               (TypeNode "AndLink")
               (TypeNode "OrLink")
               (TypeNode "NotLink")
            )
         )
      )
      (AndLink
         (EvaluationLink
            (GroundedPredicateNode "scm: gt-zero-confidence")
            (InheritanceLink
               (InheritanceLink
                  (VariableNode "$A-2fbf17d5")
                  (VariableNode "$C-2bdc1299")
               )
               (InheritanceLink (stv 1 1)
                  (ConceptNode "v" (stv 0.84615385 1))
                  (ConceptNode "w")
               )
            )
         )
         (InheritanceLink
            (VariableNode "$B-4a714efb")
            (VariableNode "$C-2bdc1299")
         )
         (InheritanceLink
            (InheritanceLink
               (VariableNode "$A-2fbf17d5")
               (VariableNode "$C-2bdc1299")
            )
            (InheritanceLink (stv 1 1)
               (ConceptNode "v" (stv 0.84615385 1))
               (ConceptNode "w")
            )
         )
         (NotLink
            (IdenticalLink
               (VariableNode "$C-2bdc1299")
               (VariableNode "$A-2fbf17d5")
            )
         )
         (InheritanceLink
            (VariableNode "$A-2fbf17d5")
            (VariableNode "$B-4a714efb")
         )
      )
   )
```

```
(ExecutionOutputLink
   (GroundedSchemaNode "scm: modus-ponens-formula")
   (ListLink
      (InheritanceLink (stv 1 1)
         (ConceptNode "v" (stv 0.84615385 1))
         (ConceptNode "w")
      )
      (InheritanceLink
         (InheritanceLink
            (VariableNode "$A-2fbf17d5")
            (VariableNode "$C-2bdc1299")
         )
         (InheritanceLink (stv 1 1)
            (ConceptNode "v" (stv 0.84615385 1))
            (ConceptNode "w")
         )
      )
      (ExecutionOutputLink
         (GroundedSchemaNode "scm: deduction-formula")
         (ListLink
            (InheritanceLink
               (VariableNode "$A-2fbf17d5")
               (VariableNode "$C-2bdc1299")
            )
            (InheritanceLink
               (VariableNode "$A-2fbf17d5")
               (VariableNode "$B-4a714efb")
            )
            (InheritanceLink
               (VariableNode "$B-4a714efb")
               (VariableNode "$C-2bdc1299")
            )
         )
      )
   )
)
```

Inference Meta-Learning in OpenCog



Quality

(stv 0.0001, 0.00075)

Atomese:

```
(EvaluationLink (stv 0.0001 0.00075)
   (PredicateNode "URE:BC:preproof-of")
   (ListLink
      (DontExecLink
         (BindLink
            (VariableList
               (TypedVariableLink
                  (VariableNode "$A-2fbf17d5")
                  (TypeChoice
                     (TypeNode "ConceptNode")
                     (TypeNode "AndLink")
                     (TypeNode "OrLink")
                     (TypeNode "NotLink")
                  )
               )
               (TypedVariableLink
                  (VariableNode "$C-2bdc1299")
                  (TypeChoice
                     (TypeNode "ConceptNode")
                     (TypeNode "AndLink")
                     (TypeNode "OrLink")
                     (TypeNode "NotLink")
                  )
               )
               (TypedVariableLink
                  (VariableNode "$B-4a714efb")
                  (TypeChoice
                     (TypeNode "ConceptNode")
                     (TypeNode "AndLink")
                     (TypeNode "OrLink")
                     (TypeNode "NotLink")
                  )
               )
            )
            (AndLink
               (EvaluationLink
                  (GroundedPredicateNode "scm: gt-zero-confidence")
                  (InheritanceLink
                     (InheritanceLink
                        (VariableNode "$A-2fbf17d5")
                        (VariableNode "$C-2bdc1299")
                     )
                     (InheritanceLink (stv 1 1)
                        (ConceptNode "v" (stv 0.84615385 1))
                        (ConceptNode "w")
                     )
                  )
               )
               (InheritanceLink
                  (VariableNode "$B-4a714efb")
                  (VariableNode "$C-2bdc1299")
               )
               (InheritanceLink
                  (InheritanceLink
                     (VariableNode "$A-2fbf17d5")
                     (VariableNode "$C-2bdc1299")
                  )
                  (InheritanceLink (stv 1 1)
                     (ConceptNode "v" (stv 0.84615385 1))
                     (ConceptNode "w")
                  )
               )
               (NotLink
                  (IdenticalLink
                     (VariableNode "$C-2bdc1299")
                     (VariableNode "$A-2fbf17d5")
                  )
               )
               (InheritanceLink
                  (VariableNode "$A-2fbf17d5")
                  (VariableNode "$B-4a714efb")
               )
            )
         )
      )
      (ExecutionOutputLink
         (GroundedSchemaNode "scm: modus-ponens-formula")
         (ListLink
            (InheritanceLink (stv 1 1)
               (ConceptNode "v" (stv 0.84615385 1))
               (ConceptNode "w")
            )
            (InheritanceLink
               (InheritanceLink
                  (VariableNode "$A-2fbf17d5")
                  (VariableNode "$C-2bdc1299")
               )
               (InheritanceLink (stv 1 1)
                  (ConceptNode "v" (stv 0.84615385 1))
                  (ConceptNode "w")
               )
            )
            (ExecutionOutputLink
               (GroundedSchemaNode "scm: deduction-formula")
               (ListLink
                  (InheritanceLink
                     (VariableNode "$A-2fbf17d5")
                     (VariableNode "$C-2bdc1299")
                  )
                  (InheritanceLink
                     (VariableNode "$A-2fbf17d5")
                     (VariableNode "$B-4a714efb")
                  )
                  (InheritanceLink
                     (VariableNode "$B-4a714efb")
                     (VariableNode "$C-2bdc1299")
                  )
               )
            )
         )
      )
      (InheritanceLink (stv 1 1)
         (ConceptNode "v" (stv 0.84615385 1))
         (ConceptNode "w")
      )
   )
)
```

Inference Meta-Learning in OpenCog

Call the URE with a Control Rule Template as target

```
(cog-bc icr-rb (QuoteLink
                   (ImplicationScopeLink
                      (UnquoteLink
                         (VariableNode "$impl-vardecl")
                      )
                      (AndLink
                         (EvaluationLink
                            (PredicateNode "URE:BC:preproof-of")
                            (UnquoteLink
                               (VariableNode "$preproof-A-args")
                            )
                         )
                         (ExecutionLink
                            (SchemaNode "URE:BC:expand-and-BIT")
                            (UnquoteLink
                               (VariableNode "$expand-inputs")
                            )
                            (UnquoteLink
                               (VariableNode "$expand-output")
                            )
                         )
                      )
                      (EvaluationLink
                         (PredicateNode "URE:BC:preproof-of")
                         (UnquoteLink
                            (VariableNode "$preproof-B-args")
                         )
                      )
                   )
                )
             )
          )
```

Inference Meta-Learning in OpenCog

Call the URE with a Control Rule Template as target

```
(cog-bc icr-rb (QuoteLink
                   (ImplicationScopeLink
                      (UnquoteLink
                         (VariableNode "$impl-vardecl")
                      )
                      (AndLink
                         (EvaluationLink
                            (PredicateNode "URE:BC:preproof-of")
                            (UnquoteLink
                               (VariableNode "$preproof-A-args")
                            )
                         )
                         (ExecutionLink
                            (SchemaNode "URE:BC:expand-and-BIT")
                            (UnquoteLink
                               (VariableNode "$expand-inputs")
                            )
                            (UnquoteLink
                               (VariableNode "$expand-output")
                            )
                         )
                      )
                      (EvaluationLink
                         (PredicateNode "URE:BC:preproof-of")
                         (UnquoteLink
                            (VariableNode "$preproof-B-args")
                         )
                      )
                   )
                )
             )
          )
```

```
(ImplicationScopeLink (stv 0.45945946 0.04625)
   (VariableList
      (VariableNode "$T")
      (TypedVariableLink
         (VariableNode "$A")
         (TypeNode "DontExecLink")
      )
      (VariableNode "$L")
      (TypedVariableLink
         (VariableNode "$B")
         (TypeNode "DontExecLink")
      )
   )
   (AndLink
      (EvaluationLink
         (PredicateNode "URE:BC:preproof-of")
         (ListLink
            (VariableNode "$A")
            (VariableNode "$T")
         )
      )
      (ExecutionLink
         (SchemaNode "URE:BC:expand-and-BIT")
         (ListLink
            (VariableNode "$A")
            (VariableNode "$L")
            (DontExecLink
               (DefinedSchemaNode "deduction-inheritance-rule")
            )
         )
         (VariableNode "$B")
      )
   )
   (EvaluationLink
      (PredicateNode "URE:BC:preproof-of")
      (ListLink
         (VariableNode "$B")
         (VariableNode "$T")
      )
   )
)
```

Inference Meta-Learning in OpenCog

Call the URE with a Control Rule Template as target + Pattern Miner

```
(cog-bc icr-rb (QuoteLink
                   (ImplicationScopeLink
                      (UnquoteLink
                         (VariableNode "$impl-vardecl")
                      )
                      (AndLink
                         (EvaluationLink
                            (PredicateNode "URE:BC:preproof-of")
                            (UnquoteLink
                               (VariableNode "$preproof-A-args")
                            )
                         )
                         (ExecutionLink
                            (SchemaNode "URE:BC:expand-and-BIT")
                            (UnquoteLink
                               (VariableNode "$expand-inputs")
                            )
                            (UnquoteLink
                               (VariableNode "$expand-output")
                            )
                         )
                      )
                      (EvaluationLink
                         (PredicateNode "URE:BC:preproof-of")
                         (UnquoteLink
                            (VariableNode "$preproof-B-args")
                         )
                      )
                   )
                )
             )
          )
```

```
(ImplicationScopeLink (stv 1 0.00625)
   (VariableList
      (VariableNode "$T")
      (TypedVariableLink
         (VariableNode "$A")
         (TypeNode "DontExecLink")
      )
      (VariableNode "$X")
      (TypedVariableLink
         (VariableNode "$B")
         (TypeNode "DontExecLink")
      )
   )
   (AndLink
      (EvaluationLink
         (PredicateNode "URE:BC:preproof-of")
         (ListLink
            (VariableNode "$A")
            (VariableNode "$T")
         )
      )
      (ExecutionLink
         (SchemaNode "URE:BC:expand-and-BIT")
         (ListLink
            (VariableNode "$A")
            (InheritanceLink
               (ConceptNode "a")
               (VariableNode "$X")
            )
            (DontExecLink
               (DefinedSchemaNode "conditional-full-instantiation-implication-scope-meta-rule"
               )
            )
         )
         (VariableNode "$B")
      )
   )
   (EvaluationLink
      (PredicateNode "URE:BC:preproof-of")
      (ListLink
         (VariableNode "$B")
         (VariableNode "$T")
```

Supercharging Cognitive Synergy with Meta-Learning

Cognitive synergy, in the simplest sense, involves solving cognitive problems via doing a few steps of one AI algorithm to get an intermediate result, then (concurrently or in parallel) a few steps of another AI algorithm acting on that intermediate result and producing another intermediate result, then (concurrently or in parallel) a few steps of another AI algorithm acting on that new intermediate result and producing one more intermediate result, etc.

Now, suppose all of these AI algorithms are operating using the URE. It then becomes possible to look for sequences of rule-applications that span multiple AI algorithms. One can look for patterns of the form "Apply these two types of mutation operations on a program tree, then apply these sorts of inference rules to combine the program tree with relatively crisp knowledge from an ontology, then apply pattern mining with these sorts of logical transformation rules to look for patterns in the results."

What Makes a Theorem Interesting?

If we knew this…

1. We could set "interestingness" as a fitness function for theorem evolution/learning
2. To the extent that interesting lemmas lead to interesting theorems, this would help with the inference control problem

Information-theoretic "surprisingness" seems to capture significant aspects of theorem-interestingness

… but surprisingness is a subtle beast

$$X + Y = Y + X$$

Is interesting because…

1. It is used in short proofs of a lot of interesting theorems
2. It provides high compressibility of “addition tables” of numbers

Irrationality of sqrt(2)

Is interesting because…

It (very simply) contradicts the probabilistic/heuristic extrapolation (made from the list of familiar numbers) that all numbers are rational

Infinitude of prime numbers

Is interesting because…

1. It contradicts the result of heuristic/probabilistic conjecturing, which says that as N increases, the number of primes greater than N eventually becomes zero
2. ??

The fundamental theorem of calculus

Is interesting because…

It contradicts the heuristic idea that differentiation and integration are very different concepts

I.e. in some sense, heuristic estimation suggests

ConceptualSimilarity (Differentiation , Integration)

has a low truth value, but then the Fundamental Theorem tells us in fact it has a high truth value.

The Generalized Frobenius Theorem

Is interesting because…

It contradicts the heuristic/probabilistic impression that there should be a lot of different algebras fulfilling the axioms of a division algebra over the reals

The consistency of hypersets under the Anti-Foundation Axiom

Is interesting because…

It contradicts the heuristic/probabilistic impression, obtained from working with ZFC (including the Axiom of Reducibility), that sets with circular membership structure are inconsistent

What Makes a Theorem Interesting?

If we knew this…

1. We could set "interestingness" as a fitness function for theorem evolution/learning
2. To the extent that interesting lemmas lead to interesting theorems, this would help with the inference control problem

Information-theoretic "surprisingness" seems to capture significant aspects of theorem-interestingness

… but surprisingness is a subtle beast