Speculative Scientific Inference via Synergetic Combination of Probabilistic Logic and Evolutionary Pattern Recognition

Ben Goertzel[1,2], Nil Geisweiller[1], Eddie Monroe[2], Mike Duncan[2], Selamawit Yilma[3], Meseret Dastaw[3], Misgana Bayetta[4], Amen Belayneh[4], Matthew Ikle'[4,5], and Gino Yu[4]

[1]OpenCog Foundation, [2] SciCog Systems, [3] iCog Labs, [4] School of Design, Hong Kong Poly U, [5] Adams State College

Abstract. The OpenCogPrime cognitive architecture is founded on a principle of "cognitive synergy" – judicious combination of different cognitive algorithms, acting on different types of memory, in a way that helps overcome the combinatorial explosions each of the algorithms would suffer if used on its own. Here one manifestation of the cognitive synergy principle is explored – the use of probabilistic logical reasoning (based on declarative knowledge) to generalize procedural knowledge gained by evolutionary program learning. The use of this synergy is illustrated via an example drawn from a practical application of the OpenCog system to the analysis of gene expression data, wherein the MOSES program learning algorithm is used to recognize data patterns and the PLN inference engine is used to generalize these patterns via cross-referencing them with a biological ontology. This is a case study of both automated scientific inference, and synergetic cognitive processing.

1 Introduction

Conceptually founded on the "patternist" systems theory of intelligence outlined in [4] and implemented in the OpenCog open-source software platform, the OpenCogPrime (OCP) cognitive architecture combines multiple AI paradigms such as uncertain logic, computational linguistics, evolutionary program learning and connectionist attention allocation in a unified architecture [7] [8]. Cognitive processes embodying these different paradigms, and generating different kinds of knowledge (e.g. declarative, procedural, episodic, sensory) interoperate together on a common neural-symbolic knowledge store called the Atomspace. The interaction of these processes is designed to encourage the self-organizing emergence of high-level network structures in the Atomspace, including superposed hierarchical and heterarchical knowledge networks, and a self-model network enabling meta-knowledge and meta-learning.

This overall architecture can be used as a tool within practical applications in areas such as data analysis or natural language processing. For instance, the OpenCog system, leveraging elements of the OpenCogPrime design, has been

used for commercial applications in the area of natural language processing and data mining; e.g. see [9] where OpenCog's PLN reasoning and RelEx language processing are combined to do automated biological hypothesis generation based on information gathered from PubMed abstracts. The same system can also be used to control an intelligent embodied agent (e.g. a game character [6] or robot [11]). In this case the focus of the system's cognition is to find and execute the procedures that it believes have the best probability of working toward its goals in its current context.

Memory Types in OpenOCP OCP's main memory types are the declarative, procedural, sensory, and episodic memory types that are widely discussed in cognitive neuroscience [14], plus attentional memory for allocating system resources generically, and intentional memory for allocating system resources in a goal-directed way. Table 1 overviews these memory types, giving key references and indicating the corresponding cognitive processes, and which of the generic patternist cognitive dynamics each cognitive process corresponds to (pattern creation, association, etc.).

The essence of the OCP design lies in the way the structures and processes associated with each type of memory are designed to work together in a closely coupled way, the operative hypothesis being that this will yield cooperative intelligence going beyond what could be achieved by an architecture merely containing the same structures and processes in separate "black boxes." This sort of cooperative emergence has been labeled "cognitive synergy." In this spirit, the inter-cognitive-process interactions in OpenCog are designed so that conversion between different types of memory is possible, though sometimes computationally costly (e.g. an item of declarative knowledge may with some effort be interpreted procedurally or episodically, etc.)

A Practical Example of Procedural/Declarative Synergy . We describe here some currently ongoing work using OpenCog, and elements of the OCP design, to analyze genomic data using a combination of two different OpenCog cognitive processes: the MOSES procedure learning algorithm, and the PLN probabilistic logic engine. This work is a practical illustration of the cognitive synergy principle: PLN helps MOSES overcome its difficulty with generalization, and MOSES helps PLN overcome its difficulty scanning large datasets for patterns. The two together can find abstract patterns in datasets, via MOSES first finding concrete patterns and PLN then abstracting them. The result is a novel form of automated speculative scientific inference that is potentially quite powerful.

While this particular genomics application is "narrow AI", the fact that it is being carried out in a software framework and cognitive architecture oriented toward general intelligence means that development and conceptual refinement done in the context of this application can be used for any OpenCog application. Further, many of the lessons learned in the context of this work are quite generally applicable, e.g. the highlighting of the "rule choice" problem as the key issue in PLN inference control (as will be discussed at the end).

Memory Type	Specific Cognitive Processes	General Cognitive Functions
Declarative	Probabilistic Logic Networks (PLN) [3]; concept blending [2]	pattern creation
Procedural	MOSES (a novel probabilistic evolutionary program learning algorithm) [12]	pattern creation
Episodic	internal simulation engine [6]	association, pattern creation
Attentional	Economic Attention Networks (ECAN) [10]	association, credit assignment
Intentional	probabilistic goal hierarchy refined by PLN and ECAN, structured according to MicroPsi [1]	credit assignment, pattern creation
Sensory	In OpenCogBot, this will be supplied by the DeSTIN component	association, attention allocation, pattern creation, credit assignment

Table 1. Memory Types and Cognitive Processes in OpenCog Prime. The third column indicates the general cognitive function that each specific cognitive process carries out, according to the patternist theory of cognition.

2 Cognitive Synergy for Procedural and Declarative Learning

The specific work to be discussed here involves combined use of OpenCog's MOSES and PLN cognitive algorithms; we now briefly indicate what each of these does, pointing to prior references for details.

MOSES for Automated Program Learning . MOSES, OCP's primary algorithm for learning procedural knowledge, has been tested on a variety of application problems including standard GP test problems, virtual agent control, biological data analysis and text classification [12]. It represents procedures internally as program trees. Each node in a MOSES program tree is supplied with a "knob," comprising a set of values that may potentially be chosen to replace the data item or operator at that node. So e.g. a node containing the number 7 may be supplied with a knob that can take on any integer value. A node containing a while loop may be supplied with a knob that can take on various possible control flow operators including conditionals or the identity. A node containing a procedure representing a particular robot movement, may be supplied with a knob that can take on values corresponding to multiple possible movements. The metaphor is that MOSES learning covers both "knob twiddling" (setting the values of knobs) and "knob creation."

One common application of MOSES is to the supervised or unsupervised analysis of datasets. In this case MOSES is learning procedures that take in a dataset, and output a prediction of what category that dataset belongs to, or

what properties that dataset has. For example, consider the following MOSES model learned in the context of supervised-classification analysis of a gene expression dataset comprising 50 human nonagenarians and 50 middle-aged controls [13]:

```
or(and(or(!$TTC3 !$ZNF542P)
or(!$LOC285484 !$RAI2 $CCNA1)
or($SERPING1 !$NLRC3))
and(!$SEMA7A !$LOC285484 !$LOC100996246)
and($SEMA7A $TJP2 !$ARMC10)
and($LOC100996246 $PSRC1 $SLC7A5P1))
==> nonagenarian
```

The semantics here is that:

- The variable containing the name of a gene, e.g. "$RAI2", denotes the predicate "Gene $RAI2 was overexpressed, i.e. expressed greater than the median across all genes, in the gene expression dataset corresponding to a particular person."
- if this Boolean combination of variables is true, then the odds are higher than average that the person is a nonagenarian rather than a control

This particular model has moderate but not outstanding statistics on the dataset in question (precision = .6, recall = .92, accuracy = .77), and was chosen for discussion here because of its relatively simple form.

PLN for Probabilistic Logical Inference . OCP's primary tool for handling declarative knowledge is an uncertain inference framework called Probabilistic Logic Networks (PLN). The complexities of PLN are the topic of two lengthy technical monographs [3] [5], and here we will eschew most details and focus mainly on pointing out how PLN seeks to achieve efficient inference control via integration with other cognitive processes.

As a logic, PLN is broadly integrative: it combines certain term logic rules with more standard predicate logic rules, and utilizes both fuzzy truth values and a variant of imprecise probabilities called *indefinite probabilities*. PLN mathematics tells how these uncertain truth values propagate through its logic rules, so that uncertain premises give rise to conclusions with reasonably accurately estimated uncertainty values.

PLN can be used in either forward or backward chaining mode. In backward chaining mode, for example,

1. Given an implication $L \equiv A \rightarrow B$ whose truth value must be estimated, create a list $(A_1, ..., A_n)$ of *(inference rule, stored knowledge)* pairs that might be used to produce L
2. Using analogical reasoning to prior inferences, assign each A_i a probability of success
   - If some of the A_i are estimated to have reasonable probability of success at generating reasonably confident estimates of L's truth value, then

invoke Step 1 with A_i in place of L (at this point the inference process becomes recursive)

- If none of the A_i looks sufficiently likely to succeed, then inference has "gotten stuck" and may be abandoned; or, another cognitive process may optionally be invoked, e.g. various options (not all currently implemented and tested) include:
  - **Concept creation** may be used to infer new concepts related to A and B, and then Step 1 may be revisited, in the hope of finding a new, more promising A_i involving one of the new concepts
  - **MOSES** may be invoked with one of several special goals, e.g. the goal of finding a procedure P so that $P(X)$ predicts whether $X \to B$. If MOSES finds such a procedure P then this can be converted to declarative knowledge understandable by PLN and Step 1 may be revisited....
  - **Simulations** may be run in OCP's internal simulation engine, so as to observe the truth value of $A \to B$ in the simulations; and then Step 1 may be revisited....

3 Example of PLN Inference on MOSES Output

Now we give a specific example of how PLN and MOSES can be used together, via applying PLN to generalize program trees learned by MOSES. We will use the MOSES model given above, learned via analysis of nonagenarian gene expression data, as an example. Further details on the specific inferences described here can be found in online supplementary material at `http://goertzel.org/BioInference.pdf`.

As the MOSES model in question is at the top level a disjunction, it's easy to see that, if we express the left hand side in OpenCog's Atomese language [1] using ANDLinks, ORLinks and NOTLinks, a single application of the PLN rule

```
Implication
     AND
        OR
           ListLink: $L
        MemberLink
           $X
           $L
     $X
```

will yield corresponding implications for each clause, such as

```
and($SEMA7A $TJP2 !$ARMC10) ==> nonagenarian
```

for the third second-level clause. For the rest of our discussion here we will focus on this clause due to its relatively small size. Of course, similar inferences to the ones we describe here can be carried out for larger clauses and for Boolean

[1] see e.g. [7] for a review of this notation

combinations with different structures. The PLN software deals roughly equally well with Boolean structures of different shapes and size.

This latter implication, in the OpenCog Atomspace, actually takes the form

```
ImplicationLink
    ANDLink
        ExecutionOutputLink
            SchemaNode "makeOverexpressionPredicate"
            GeneNode "SEMA7A"
        ExecutionOutputLink
            SchemaNode "makeOverexpressionPredicate"
            GeneNode "TJP2"
        NotLink
            ExecutionOutputLink
                SchemaNode "makeOverexpressionPredicate"
                GeneNode "ARMC10"
    PredicateNode "Nonagenarian"
```

where

```
EquivalenceLink
    EvaluationLink
        ExecutionOutputLink
            SchemaNode "makeOverexpressionPredicate"
            GeneNode $G
        ConceptNode $H
    EvaluationLink
        GroundedPredicateNode "scm:above-median"
        ListLink
            ExecutionOutputLink
                SchemaNode "makeExpressionLevelPredicate"
                GeneNode $G
            ConceptNode $H
            ConceptNode $P

EquivalenceLink
    EvaluationLink
        ExecutionOutputLink
            SchemaNode "makeExpressionLevelPredicate"
            GeneNode $G
        ConceptNode $H
    EvaluationLink
        PredicateNode "Expression level"
        ListLink
            GeneNode $G
            ConceptNode $H
```

where

– "scm:above-median" is a helper function that evaluates if a certain predicate (arg1) evaluated at arg2 is above the median of the set of values obtained by applying arg1 to every member of the category arg3.

– "makeExpressionLevelPredicate level" is a schema that outputs, for an argument $G, a predicate that is evaluated for an argument that represents an organism, and outputs the expression of $G in that organism.
– "Expression level" is a predicate that outputs, for arguments $G and $H, the level of expression of $G in organism $H.

Being a nonagenarian in itself is not that interesting, but if you know the entity in question is a human (instead of, say, a bristlecone pine tree), then it becomes interesting indeed. This knowledge is represented via

```
ImplicationLink
   AND
      PredicateNode "Human"
      PredicateNode "Nonagenarian"
   PredicateNode "LongLived"
```

from which PLN can conclude

```
ImplicationLink
   ANDLink
      ExecutionOutputLink
         PredicateNode "makeOverexpressionPredicate"
         GeneNode "SEMA7A"
      ExecutionOutputLink
         PredicateNode "makeOverexpressionPredicate"
         GeneNode "TJP2"
      NotLink
         ExecutionOutputLink
            PredicateNode "makeOverexpressionPredicate"
            GeneNode "ARMC10"
    PredicateNode "LongLived"
```

Next, how can PLN generalize this MOSES model? One route is to recognize patterns spanning this model and other MOSES models in the Atomspace. Another route, the one to be elaborated here, is cross-reference it with external knowledge resources, such as the Gene Ontology (GO). The GO is one of several bio knowledge resources we have imported into a bio-oriented OpenCog Atomspace that we call the Biospace.

Each of these three genes in our example belongs to multiple GO categories, so there are many GO-related inferences to be done regarding these genes. But for sake of tractable exemplification, let's just look at a few of the many GO categories involved:

– SEMA7A is_a GO:0045773 (positive regulation of axon extension)
– TJP2 is_a GO:0006915 (apoptotic process)
– ARMC10 is_a GO:0040008 (regulation of growth)

Let us also note a relationship between the first and third of these GO categories, drawn from the GO hierarchy:

– GO:0045773 is_a GO:0048639 (positive regulation of developmental growth)

- GO:0048639 is_a GO:0045927 (positive regulation of growth)
- GO:0045927 is_a GO:0040008 (regulation of growth)

As well as these relationships between genes and GO categories, the Biospace also contains knowledge

```
AssociativeLink
    ConceptNode "GO:0006915"
    PredicateNode "LongLived"
```

which is derived from the known association of multiple genes in the category GO:0006915 with longevity. From this, PLN can derive that

```
ImplicationLink
    MemberLink
        $G
        ConceptNode "GO:0006915"
    ImplicationLink
        ExecutionOutputLink
            PredicateNode "makeOverexpressionPredicate"
            $G
        PredicateNode "LongLived"
```

, i.e. that overexpression of genes in this GO category is likely to imply longevity. Since this GO categories contains one of the genes (TJP2) in the MOSES model under study, after a few PLN steps, this background knowledge, combined with the MOSES model, increases the estimated odds that the other two genes in the MOSES model are related to longevity, e.g. that

```
ImplicationLink
    MemberLink
        $G
        ConceptNode "ARMC10"
    ImplicationLink
        ExecutionOutputLink
            PredicateNode "makeOverexpressionPredicate"
            $G
        PredicateNode "LongLived"
```

Further, the membership of these other two genes in the GO category "GO:0040008" allows PLN to derive the abstraction

```
ImplicationLink
   AssociativeLink
       ConceptNode "GO:0040008"
       $L
   ImplicationLink
      ANDLink
         AppendLink
            ListLink: $L
            ExecutionOutputLink
               PredicateNode "makeOverexpressionPredicate"
```

```
            GeneNode "TJP2"
    PredicateNode "LongLived"
```

What this means, intuitively, is that combinations of TJP2 with growth-regulation genes tends to promote longevity. This is interesting, among other reasons, because it's exactly the kind of abstraction a human mind might form when looking at this kind of data.

In the above examples we have omitted quantitative truth values, which are attached to each link, and depend on the specific parameters associated with the PLN inference formulas. The probability associated with the final ImplicationLink above is going to be quite low, below 0.1 for any sensible parameter values. However, this is still significantly above what one would expect for a linkage of the same form with a random GO and gene inside it. We are not aiming to derive definite conclusions here, only educated speculative hypotheses, able to meaningfully guide further biological experimentation.

The "cognitive synergy" in the above may not be glaringly obvious but is critical nonetheless. MOSES is good at learning specific data patterns, but not so good at learning abstractions. To get MOSES to learn abstractions of this nature would be possible but lead to significant scalability problems. On the other hand, PLN is good at abstracting from particular data patterns, but doesn't have control mechanisms scalable enough to enable it to scan large datasets and pick out the weak but meaningful patterns among the noise. MOSES is good at this. The two algorithms working together can, empirically speaking, create generalizations from large, complex datasets, significantly better than either algorithm can alone.

4 Conclusions and Next Steps

The work described here has its specialized aspects, but also leads to various general ideas and lessons. Conceptual interplay between practical applications to complex real-world data and more abstract AGI R&D, helps to push both pursuits forward.

The workflow described above uses MOSES to analyze data and produce classification models, and PLN to draw conclusions from these models via cross-referencing them with external knowledge or (not elaborated above) one another. The loop may be closed by taking the genes highlighted as most relevant by PLN (in the above case, the genes found to imply longevity most strongly via combination of PLN) and using them as a restricted input feature set for MOSES. MOSES can then learn more models based on this feature set, which can then be exported to the Atomspace and used by PLN, etc. In this way PLN is being used to enable a kind of MOSES recursive feature selection.

One of the main lessons learned in experimenting with inferences like the ones mentioned above, is that the primary AI difficulty involved is telling PLN which rules to choose in what order. Choosing which nodes (e.g. GeneNodes) to include is challenging as well but is addressed via OpenCog's activation-spreading-like ECAN component. Choosing which rules to apply when is not currently handled

effectively; but in [7] it is proposed to do this via assigning probabilities to sequences of rule-choices (conditional on the context), thus allowing "rule macros" (i.e. sequences of rules) to be applied in a fairly habitual way in a given domain of inference. But of course that is a high-level description and there will be some devils in the details. It has been previously proposed to use pattern mining to learn macros of this nature, and it's clear this will be a good approach and necessary in the medium term. However, a simpler approach might be to simply run a bunch of inferences and store Markov probabilities indicating which chains of rule-applications tended to be useful and which did not; this might provide sufficient rule-choice guidance for "relatively simple" inferences like the ones given here.

References

1. Bach, J.: Principles of Synthetic Intelligence. Oxford University Press (2009)
2. Fauconnier, G., Turner, M.: The Way We Think: Conceptual Blending and the Mind's Hidden Complexities. Basic (2002)
3. Goertzel, B., Ikle, M., Goertzel, I., Heljakka, A.: Probabilistic Logic Networks. Springer (2008)
4. Goertzel, B.: The Hidden Pattern. Brown Walker (2006)
5. Goertzel, B., Coelho, L., Geisweiller, N., Janicic, P., Pennachin, C.: Real World Reasoning. Atlantis Press (2011)
6. Goertzel, B., Et Al, C.P.: An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In: Proc.of the First Conf. on AGI. IOS Press (2008)
7. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy. Springer: Atlantis Thinking Machines (2013)
8. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 2: The CogPrime Architecture for Integrative, Embodied AGI. Springer: Atlantis Thinking Machines (2013)
9. Goertzel, B., Pinto, H., Pennachin, C., Goertzel, I.F.: Using dependency parsing and probabilistic inference to extract relationships between genes, proteins and malignancies implicit among multiple biomedical research abstracts. In: Proc. of Bio-NLP 2006 (2006)
10. Goertzel, B., Pitt, J., Ikle, M., Pennachin, C., Liu, R.: Glocal memory: a design principle for artificial brains and minds. Neurocomputing (Apr 2010)
11. Goertzel, B.e.a.: Opencogbot: An integrative architecture for embodied agi. Proc. of ICAI-10, Beijing (2010)
12. Looks, M.: Competent Program Evolution. PhD Thesis, Computer Science Department, Washington University (2006)
13. Passtoors, W., JM, B., Goeman, J., Akker, E.: Transcriptional profiling of human familial longevity indicates a role for asf1a and il7r. PLoS One (2012)
14. Tulving, E., Craik, R.: The Oxford Handbook of Memory. Oxford U. Press (2005)