# Thinking About the AtomSpace

## Knowledge Representation with Graphs

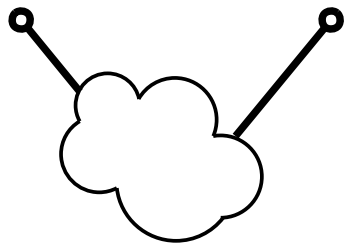
Linas Vepstas

15-16 July 2020

(Virtual) OpenCogCon 2020

# Topics

New Features and New Ideas



# Representing Extremely Sparse Data

A common task in data science

- A matrix: $M = M_{ij} = P(x, y) = P(A|B) =$ 
- For example: conditional probabilities, marginal probabilities...
- When $i, j, x, y, A, B$ are words, genes, protiens...
- Extremely sparse data
- Out of 100K $\times$100K there are maybe 10M pairs!

# OpenCog "Classic" Style

Symbolic Relational Algebra
Knowledge representation with Atoms

**Genomics**

EvaluationLink
    Predicate "up-regulates"
    List
        Gene "FLNC"
        Gene "MAP2K4"

- $M_{ij} = P(x, y) = R_{upregulates}\,(FLNC, MAP2K4)$
- But where are the numbers?

# Key-Value Store per Atom

Declarative expressions

## Setting Values by declaring them! ... with Atoms!

SetValueLink
    EvaluationLink
        Predicate “up-regulates”
        List
            Gene “FLNC”
            Gene “MAP2K4”
    *<some key>* *<some value>*

# Matrix Subsystem

Scheme: `(use-modules (opencog matrix))`

- Object-oriented API to matricies in the AtomSpace
- Generic programming: "parametric polymorphism"

```
(define (my-genomics-object)
    (define (get-left-type) 'Gene)
    (define (get-right-type) 'Gene)
    (define (get-pair-type) 'EvaluationLink)
    (define (get-count PAIR)
        (cog-value PAIR (Predicate "some-key"))))
```

# Matrix Toolkit

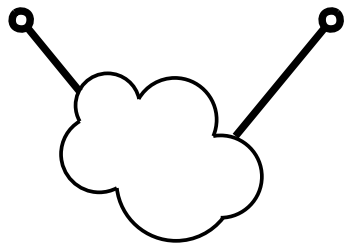
- Frequencies, marginal probabilities
- Mutual Information
- Similarity: e.g. cosine similarity
- $\ell_p$-norms ("manhattan distance", etc.)
- Data filters and data cuts!

Someone:

PLEASE DO THIS: Port to Gnu R or to SciPy(?)

# Topics

New Features and New Ideas



# Values are Mutable

- `(FloatValue 1 2 3)`
- `(SimpleTruthValue 0.99 0.6)`
- `(StringValue “a” “b” “c”)`
- `(LinkValue (StringValue “answer”) (FloatValue 42))`
- `StreamValue`
  - `RandomStream`
  - `QueueValue` ;;; aka FIFO, buffer
  - `FormulaStream` ;;; stream transducer

# Values can be manipulated...

... with Atoms!

**Copying Values**

```
(SetValue (Concept "foo") (Predicate "some key")
    (ValueOf (Concept "bar") (Predicate "other key")))
```

- Declarative Knowledge!
  - Texbook relations: dog is-a animal, dog has-a tail
- Declare the movement of values
  - Copying, arithmetic, formulas...

# Formulas

Values can be transformed

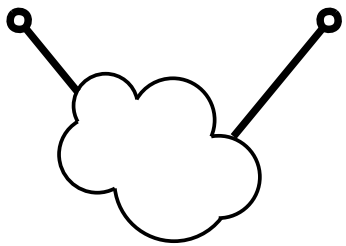
## Triangle Numbers

```
(Lambda
    (Variable "$X")
    (Divide
        (Times (Variable "$X")
            (Plus (Variable "$X") (Number 1)))
        (Number 2))))
```

- Verbose!
- But Declarative!
- Searchable!

# Topics

New Features and New Ideas

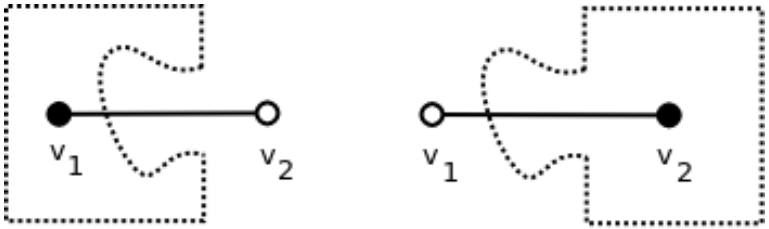


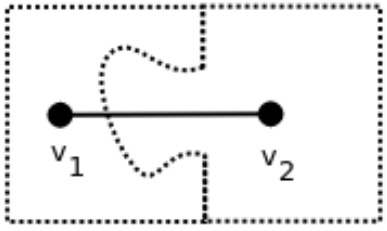
# Connectors and Bonds

Terms and variables

- A term: $f\left(x\right)$ or an $n$-ary function symbol: $f\left(x_1, x_2, \cdots, x_n\right)$
- A variable: $x$ or maybe more: $x, y, z, \cdots$
- A number: 42 .. or a string "foobar" ... or ...
- Plug it in: $f\left(x\right) : 42 \mapsto f\left(42\right)$
- "Call function $f$ with argument of 42"

# Plug it in!



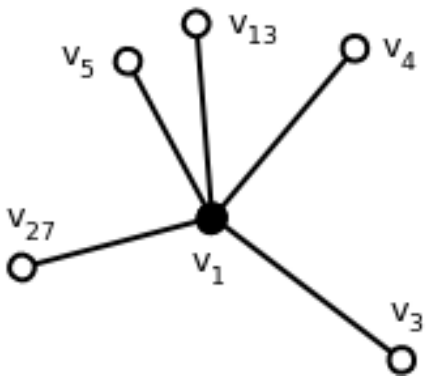




Agnostic connections

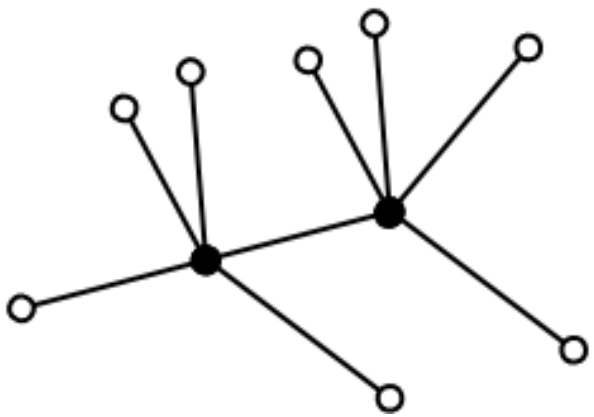
- Which one is the function?
- Which one is the argument?
- Who called who?

# Generic Connectors

Generic connectors (aka "tensors": $M_{ijk\ldots}$)




Generic bonds (aka "tensor contraction": $v^{\mu} g_{\mu\nu} dx^{\nu}$)



# Connectors and Bonds in Atomese

An $n$-ary function symbol: $f\left(x_1, x_2, \cdots, x_n\right)$

```
(Section
    (Concept "function f")
    (ConnectorSeq
        (Connector (Type "num var") (Label "x1"))
        (Connector (Type "num var") (Label "x2"))
        ...
        (Connector (Type "num var") (Label "xn"))))
```

# Natural Language

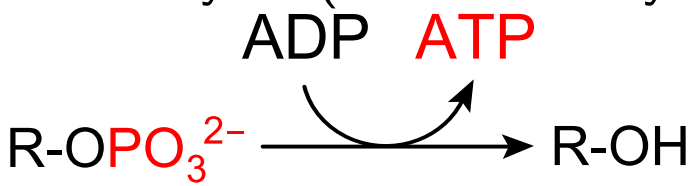
Linguistics: SUBJECT threw an OBJECT

```
(Section
    (Word "throw")
    (ConnectorSeq
        (Connector (Type "SUBJECT") (ConnectorDir "left"))
        (Connector (Type "OBJECT") (ConnectorDir "right"))))
```

This is what Link Grammar is!

# Chemistry

Krebs cycle (Citric acid cycle)




```
(Section
    (Concept “Phosphorylation”)
    (ConnectorSeq
        (Connector (Type “R-OPO3”) (ConnDir “input”))
        (Connector (Concept “ADP”) (ConnDir “input”))
        ...
        (Connector (Type “R-OH”) (ConnDir “output”))
```

# Theorem Proving

Natural Deduction - Judgements and Propositions

Rule of inference: $\dfrac{A \text{ prop} \qquad B \text{ prop}}{(A \wedge B) \text{ prop}} \wedge_F$

```
(Section
    (Label “Rule of Introduction A and B”)
    (ConnectorSeq
        (Connector (Type “Prop”) (ConnDir “input”))
        (Connector (Type “Prop”) (ConnDir “input”))
        (Connector (Type “Prop”) (ConnDir “output”))
```

# Connectors and Bonds: Why?

Because Computer Science!

- Parsing and Grammar and Syntax and Language
- Generation of graphs (sentences, languages ...)
- ...with weighted probabilities (Bayesian, PLN, ...)
- ...with constraints (constraint satisfaction)
- Satisfiability Modulo Theories
- Logical Inference and Deduction (...probabilistic....)
- Tensor algebras and deformations

# Take-aways

- Sparse data is rampant in real life.
- Graphical representations are natural.
- Jigsaw puzzle pieces are actually ... tensors!
- Parsing == assembling jigsaw puzzle pieces!
- Values flow along graph edges

- Projects
  - Nascant generation of graphs: https://github.com/opencog/generate
  - Learning graph components: https://github.com/opencog/learn