# Probabilistic Logical Networks

## Introduction

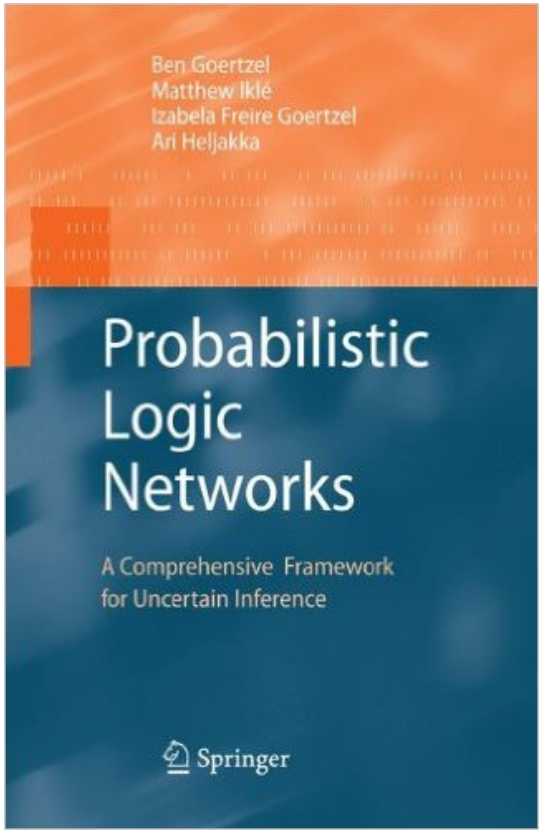
Nil Geisweiller

OpenCog Foundation

The Robotics Garage Project 2016

# Outline

# What is PLN?

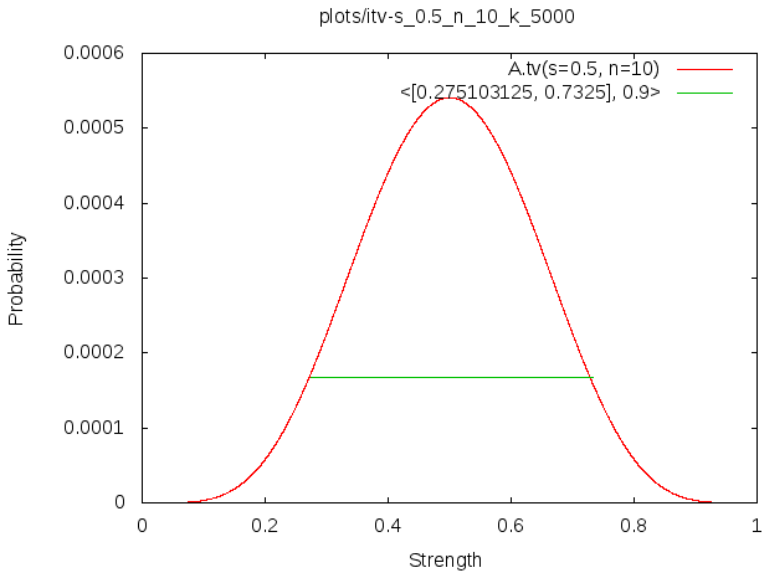


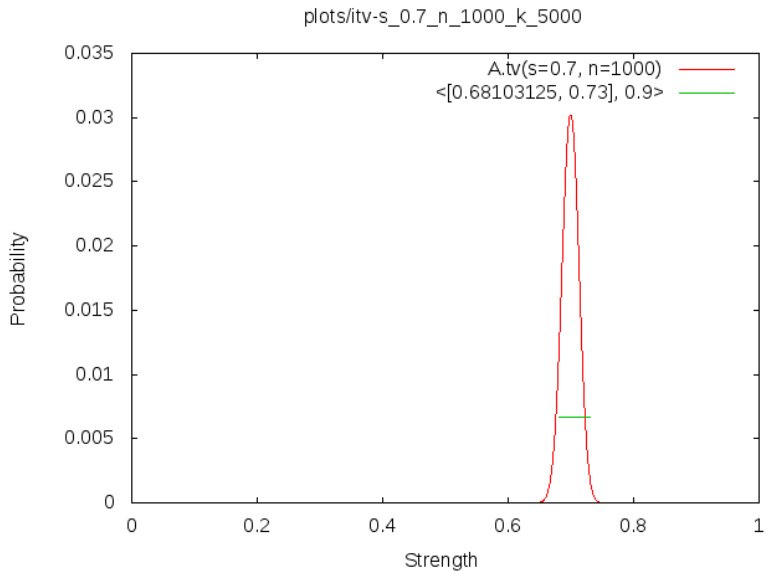

http://wiki.opencog.org/wikihome/index.php/OpenCogPrime:PLNBookErrata

http://wiki.opencog.org/wikihome/index.php/PLNBook

# What is PLN?

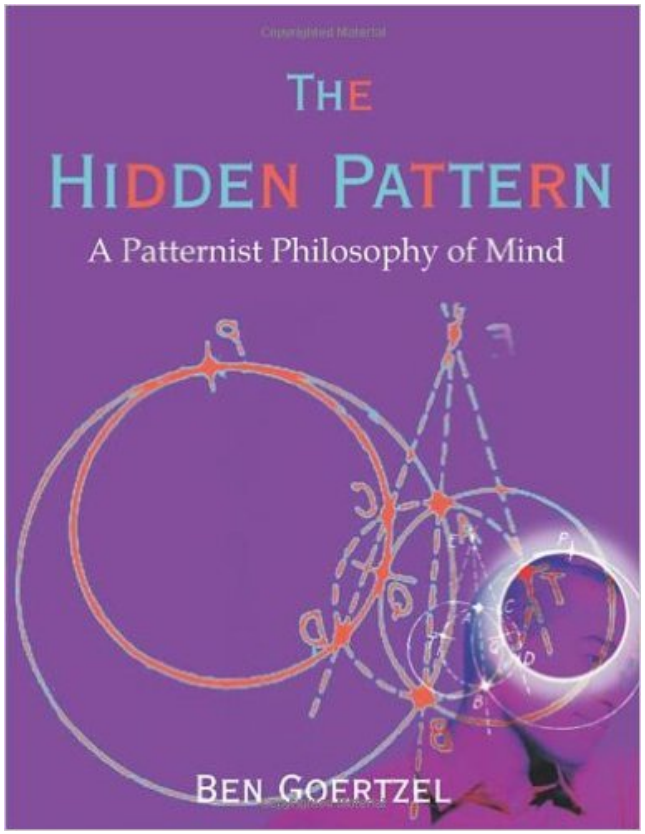
- Handling the kind of uncertain, messy reasoning we humans are so good at, but machines not so good.
- While maintaining full consistency with Probability Theory!



# What is PLN?

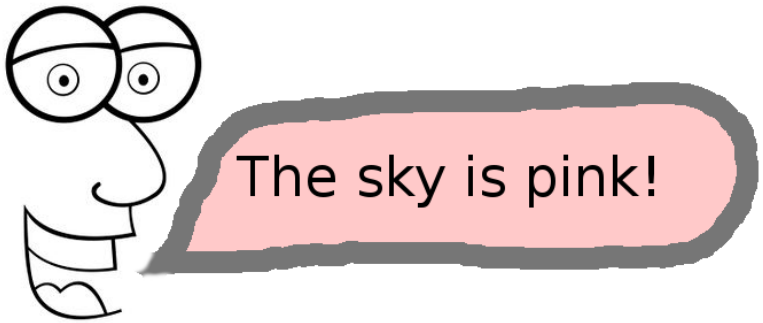
Follow the Patternist Philosophy. Mixing extensional and intensional reasoning.




Inheritance <0.4, 0.8>
  Dolphin
  Fish

# What is PLN?

Very good at mixing knowledge coming from different sources with different confidence levels.




# Outline

# Example: Query

*What is the color of the sky?*

Inheritance <?>
  Sky
  Variable "$Color"

# Example: Background Knowledge

*John says that the sky is Pink* (KB.1)

```
Evaluation <1 0.99>
  Predicate "Say"
  List
    Concept "John"
    Inheritance
      Concept "Sky"
      Concept "Pink"
```

*John's credential is low* (KB.2)

```
Implication <0.6 0.1>
  Variable "$X"
  Evaluation
    Predicate "Say"
    List
      John
      Variable "$X"
  Variable "$X"
```

*According to my limited observations the sky is blue* (KB.3)

```
Inheritance <0.9 0.6>
  Concept "Sky"
  Concept "Blue"
```

*Though sometimes pink as well* (KB.4)

```
Inheritance <0.1 0.6>
  Concept "Sky"
  Concept "Pink"
```

# Example: Rules

*Deduction* (R.1)

Inheritance <TV1>
  A
  B
Inheritance <TV2>
  B
  C
⊢
Inheritance <$f_1$(TV1, TV2, ...)>
  A
  C

*Disjunction Composition* (R.2)

P <TV1>
Q <TV2>
⊢
And <$f_4$(TV1, TV2)>
  P
  Q

*Inversion* (R.3)

Inheritance <TV>
  A
  B
⊢
Inheritance <$f_3$(TV, ...)>
  B
  A

*Conditional Instantiation* (R.4)

Implication <TV>
  V
  P
  Q
T
⊢
Q[V->T] <$f_2$(TV, ...)>

# Example: Inference

Only one inference step: Apply (R.4) on (KB.2)

Inheritance <0.6 0.099>
  Concept "Sky"
  Concept "Blue"

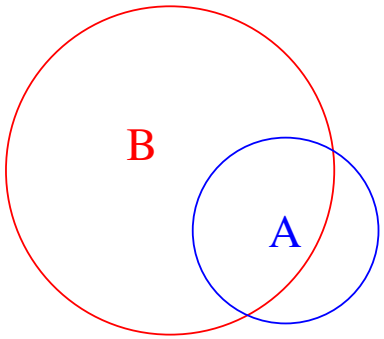
When producing that conclusion PLN will merge it to the existing one which has a much higher confidence, so the result will be

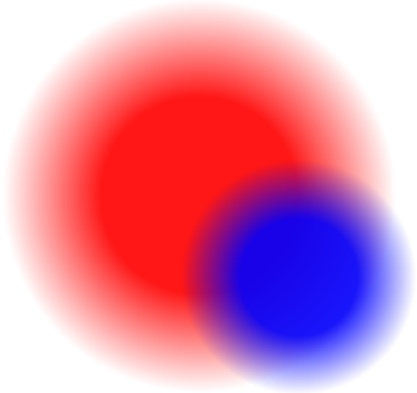
Inheritance <0.91 0.62>
  Concept "Sky"
  Concept "Blue"

# Outline

# Outline

# SubSet



SubSet A B <s c> $\equiv$ $P(B|A) = s$

- Fuzzy/Multi sets

$$P(B|A) = \frac{\sum_x \min(A(x), B(x))}{\sum_x A(x)}$$

# And, Or, Not

- And $<P(A, B)\ c>$
  - A
  - B
- Or $<P(A \cup B)\ c>$
  - A
  - B
- Not $<P(\neg A)\ c>$
  - A
  - B

# Level 0: Subset, And, Or, Not

Level 0 is all about extensional constructs.

Rules dealing with these constructs

- modus ponens
- deduction, inversion
- conjunction, disjunction, negation compositions
- universal and conditional instantiations

form the basis of PLN.[1]

[1] ForAll let aside

# Level 0: Subset, And, Or, Not

Level 0 is all about extensional constructs.

Rules dealing with these constructs

- modus ponens
- deduction, inversion
- conjunction, disjunction, negation compositions
- universal and conditional instantiations

form the basis of PLN.[1]

**Take away**

All the remaining of PLN is built on top of that.

[1] ForAll let aside

# Outline

# SubSet ≡ Extensional Inheritance

ExtensionalInheritance <TV>
  A
  B

≡

SubSet <TV>
  A
  B

# Extensional Inheritance ≡ Extensional Implication

ExtensionalInheritance <TV>
  SatisfyingSet P
  SatisfyingSet Q

≡

ExtensionalImplication <TV>
  P
  Q

# Extensional Inheritance ≡ Extensional Implication

ExtensionalInheritance <TV>
  SatisfyingSet P
  SatisfyingSet Q

≡

ExtensionalImplication <TV>
  P
  Q

Member <TV>
  T
  SatisfyingSet P

≡

Evaluation <TV>
  P
  T

# Outline

# Intensional Inheritance $\equiv$ Extensional Inheritance

Intensional Inheritance $\equiv$ Extensional Inheritance[2]!!!

[2] in pattern space

# Intensional Inheritance $\equiv$ Extensional Inheritance

Intensional Inheritance $\equiv$ Extensional Inheritance[2]!!!

| IntentionalInheritance <TV> | | ExtensionalInheritance <TV> |
|---|---|---|
| A | $\equiv$ | Patterns A |
| B | | Patterns B |

## Patterns A

All properties of A (super-sets) weighted by a prior (Solomonoff Universal Distribution).

[2] in pattern space

# Outline

# Context is almost kinda equivalent to SubSet!

ContextLink <TV>
  Concept C
  Concept A

$\equiv$

SubSet <TV>
  Concept C
  Concept A

# Context is almost kinda equivalent to SubSet!

```
ContextLink <TV>
  Concept C
  Concept A
```

≡

```
SubSet <TV>
  Concept C
  Concept A
```

More generally [or not]

```
ContextLink <TV>
  C
  R
    A1
    ...
    An
```

≡

```
R <TV>
  And A1 C
  ...
  And An C
```

# Outline

# Lambda

**Remark**

All atoms taking place in a PLN inference chain have only scoped variables.

# Implication Sugar Syntax

Basic format

Implication
  <implicant-predicate>
  <implicand-predicate>

Example

Implication
  Predicate "is-primate"
  Predicate "is-mammal"

# Implication Sugar Syntax

Using Lambda

```
Implication
  Lambda
    <variables>
    <implicant-predicate-body>
  Lambda
    <variables>
    <implicand-predicate-body>
```

Example

```
Implication
  TypedVariable
    Variable X
    Type "ConceptNode"
  Lambda
    TypedVariable
      Variable X
      Type "ConceptNode"
    Evaluation
      Predicate "is-primate"
      X
  Lambda
    TypedVariable
      Variable X
      Type "ConceptNode"
    Evaluation
      Predicate "is-mammal"
      X
```

# Implication Sugar Syntax

Sugar form

Implication
  <variables>
  <implicant-predicate-body>
  <implicand-predicate-body>

Example

Implication
  TypedVariable
    Variable X
    Type "ConceptNode"
  Evaluation
    Predicate "is-primate"
    X
  Evaluation
    Predicate "is-mammal"
    X

# Outline

# Instantiation done right

- How reconcile instantiations from different predicates or implications?
  Suppose we have predicate Q <0.01 0.99999>

  And implication P->Q <0.2 0.9>

  One universally instantiate with A based on Q, leading to Q(A) <0.01 0.99999>

  or conditionally instantiate with A based on P->Q (assuming that P(A) holds), leading to

  Q(A) <0.2 0.9>

# Priors

- How to represent (let alone decide) a prior in a Atom? For instance if we have

  Implication
  <very complex P>
  <very complex Q>

  we may want to use some prior to for instance decrease the confidence of such an atom, or perhaps increase its confidence that Q doesn't depend on P.

  Perhaps we can represent these priors in the AtomSpace itself! using higher order facts, etc, and just let PLN chew on them.

# Objective Probabilistic Atom Semantics

What is Ω?

```
Implication <TV>
  Evaluation
    Predicate "sensor-A"
  Evaluation
    Predicate "sensor-B"
```

```
Implication <TV>
  Lambda
    Variable "$T"
    AtTime "$T"
      Evaluation
        Predicate "sensor-A"
  Lambda
    Variable "$T"
    AtTime "$T"
      Evaluation
        Predicate "sensor-B"
```

So Evaluation a can be a Predicate as well. Then you don't want to use systematically Fuzzy TV semantics for And, Or, etc.