# AGI-20 Tutorial
# OpenCog, PLN and Pattern Miner

Nil Geisweiller, Matt Ikle

SingularityNET & OpenCog Foundations

1. Install docker
   - Debian/Ubuntu

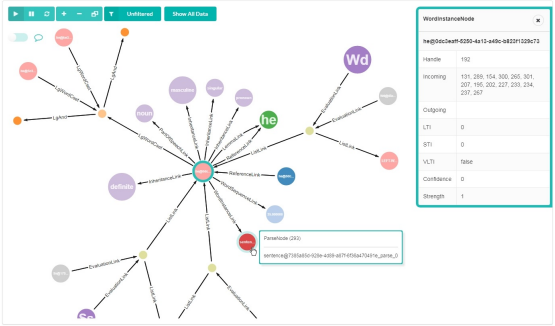
     ```
     sudo apt install docker.io
     ```
   - Arch/Manjaro

     ```
     sudo pacman -S docker
     ```
2. Download docker image (1.6GB)

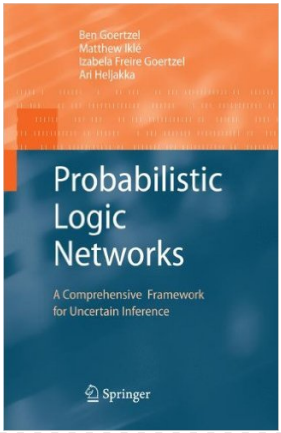
   ```
   sudo docker pull ngeiswei/opencog:agi20
   ```

# Framework for AGI

1. Hypergraph Database:
   - AtomSpace
   - Atomese: query, rewrite and more
2. Mind Agents (cognitive processes):
   - Reasoning: PLN, Miner
   - Learning: MOSES, Miner
   - Decision: OpenPsi (Bach's MicroPsi)
   - Language Processing
   - Attention Allocation




Download docker image: `sudo docker pull ngeiswei/opencog:agi20`

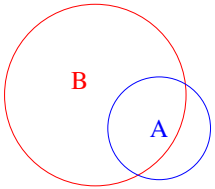



- Probability Theory
- Uncertainty management
- Common sense reasoning
- Mathematical reasoning
- Resource management

---

Download docker image: `sudo docker pull ngeiswei/opencog:agi20`

```
(Subset (stv s c)
    A
    B)
```

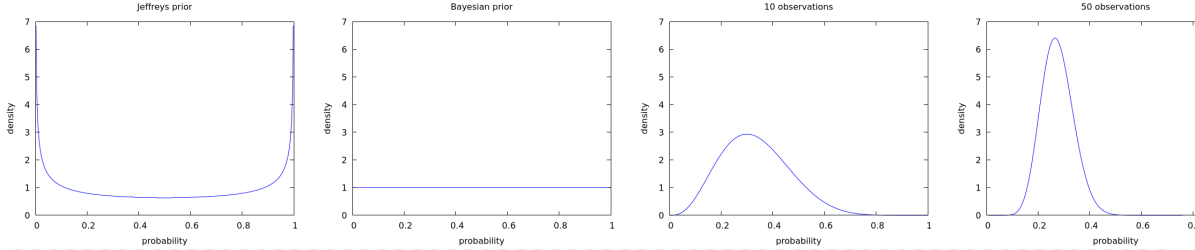



**Definitions:**

- `stv` = *Simple Truth Value*
- `Subset A B` = $P(B|A)$
- `s` = strength = $P(B|A) = \frac{|A \cap B|}{|A|}$
- `c` = confidence = $\frac{|A|}{|A| + K}$

Download docker image: `sudo docker pull ngeiswei/opencog:agi20`

*Truth Value = Second Order Distribution*




Download docker image: `sudo docker pull ngeiswei/opencog:agi20`

- Knowledge base:

```
(Subset (stv 0.4 0.1) A B)
(Subset (stv 0.6 0.2) B C)
A (stv 0.1 0.6)
```

- Rule base:

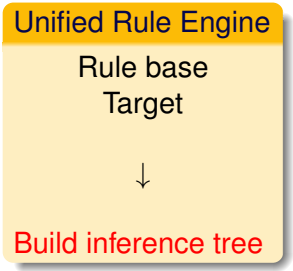
$$\frac{\texttt{(Subset <tv1> X Y)} \qquad \texttt{(Subset <tv2> Y Z)}}{\texttt{(Subset <tv3> X Z)}} \text{ (Deduction)}$$

$$\frac{\texttt{X <tv1>} \qquad \texttt{(Subset <tv2> X Y)}}{\texttt{Y <tv3>}} \text{ (Modus Ponens)}$$

- Target: `C <?>`

---

Download docker image: `sudo docker pull ngeiswei/opencog:agi20`




$$\dfrac{\dfrac{\texttt{(Subset <?> A B)} \qquad \texttt{(Subset <?> A B)}}{\texttt{A <?>} \qquad \texttt{(Subset <?> A C)}}\ \text{(Deduction)}}{\texttt{C <?>}}\ \text{(Modus Ponens)}$$

---

Download docker image: `sudo docker pull ngeiswei/opencog:agi20`

Basic:

- Deduction
- Modus Ponens
- Conjunction/Disjunction/Negation Introduction
- Universal Instantiation
- ...

Advanced:

- Intensional
- Contextual
- Temporal
- Spatial
- ...

I also mentioned that reasoning can also used for learning. In OpenCog we at least have one component that's already built like that, this is the pattern miner. The pattern miner allows to discover rather simple frequent patterns hiding in an atomspace. By simple I don't mean that they are small BTW, I just mean that their size is not gonna be a very good measure of their Kolmogorov complexity.
So here's a simple example, if
A->B, A->C, A->D |- A->X
we have a pattern which is that A implies something.

So how to map that pattern miner as a form of reasoning, we just formalize the following property, which is that if a pattern is frequent enough over some dataset, then an abstract of that pattern is frequent enough. In practice, the implementation is more trickier, and we also need to account for the surprisingness of the pattern, but that's the idea, or at least one of the ideas.