We don't have a clue how the mind is working.

Neural-Symbolic Integration

Pascal Hitzler

AIFB **Universität Karlsruhe (TH)**

Motivation

- *Biological* neural networks can easily do logical reasoning.
- Why is it so difficult with *artificial* ones?

Some Other Motivation

Artificial neural networks constitute a *robust and successful machine learning paradigm*.

But they are black boxes.

Symbolic logic provides *declaratively well understood knowledge representation and reasoning paradigms*.

Which lack robustness and powerful learning abilities.

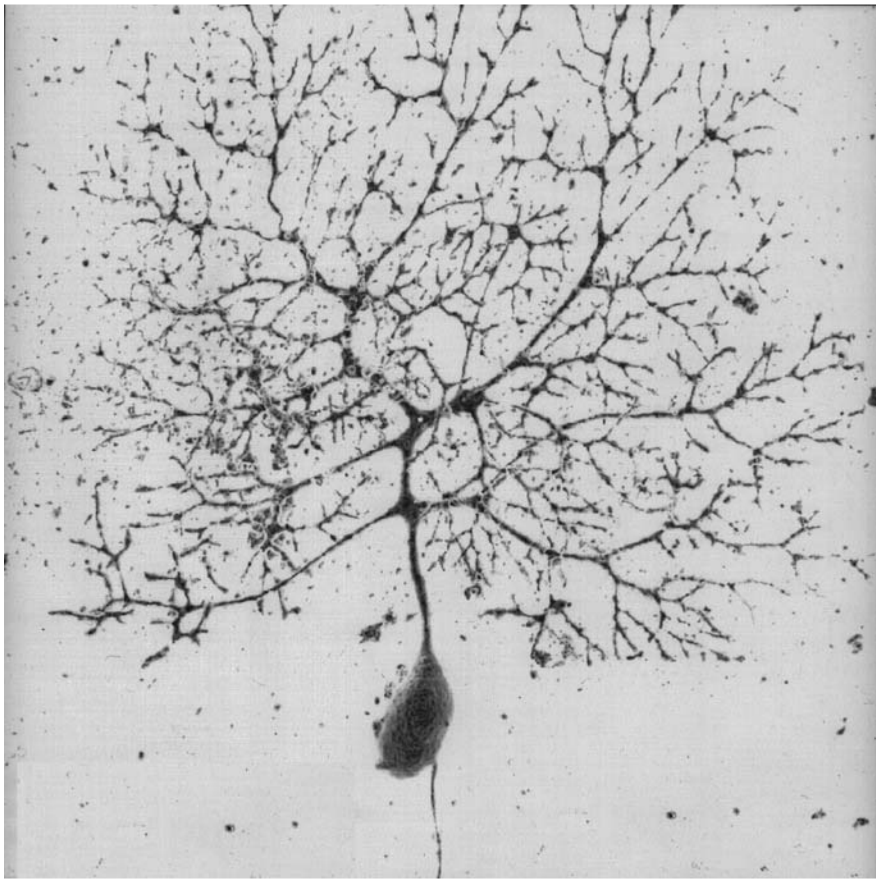
We seek intergrated paradigms retaining the best of both worlds!

Contents

- Artificial neural networks
- Propositional Logic and Connectionism
  - Problem description
  - McCulloch-Pitts Networks
  - CILP and KBANN
  - Hopfield networks
- First-Order Logic and Connectionism
  - Particular difficulties
  - SHRUTI
  - RAAM architectures
  - Approximation of logic programs
- Ontology learning as a use case

Research collaborators: Sebastian **Bader** (Dresden, Germany), Artur S. d'Avila **Garcez** (London, UK), Steffen **Hölldobler** (Dresden, Germany), Anthony K. **Seda** (Cork, Ireland).

Biological neural nets



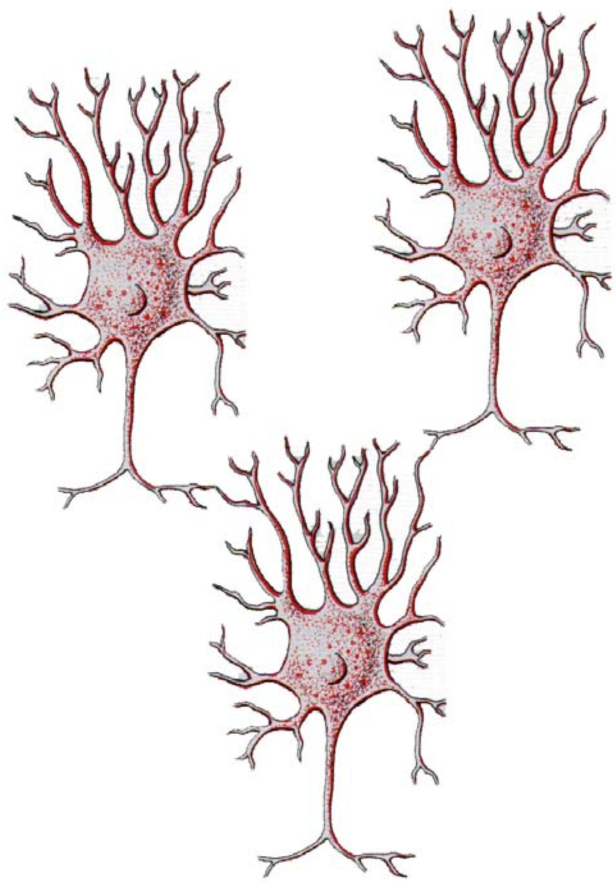
Neuron,

with dendrites, soma, and axon.

(Purkinje cell from cerebellum)

Picture:
Spektrum der Wissenschaft 10,
October 2001

Biological neural nets



Potentials are being propagated from the dendrites to the soma.

If the accumulated potetial is above a certain threshold, the neuron fires.

The resulting potential is being propagated to other neurons via the axon.

Pictures: Birbaumer & Schmidt, Biologische Psychologie, ^{2}1991

Artificial neural nets

(Finite) set of *units* (nodes, neurons) with *connections*.

▶ graph

- Potentials are real numbers.
- Propagation doesn't need time.
- Potentials accumulate as weighted sums.
  (Weights stand for synaptic activity and can be *learned*.)
- Units become active in discrete time steps.
- Threshold function is the same everywhere in the network.

There exist many competing architectures.

Artificial neural nets

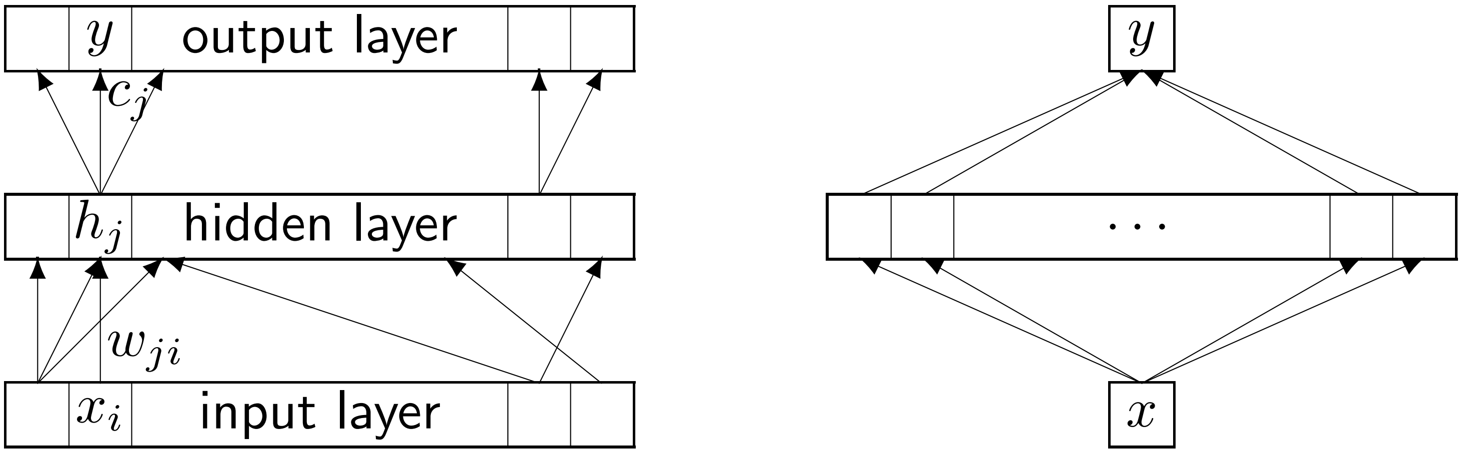
In particular:

- Every unit computes a *simple* input-output function.
- The units are *blind* concerning the sources of their input and the targets of their output.

Information (knowledge)
is being represented by the
(weighted) connections
in the network!

▶ *Connectionist systems.*

3-layer feedforward nets/perceptrons



x_i inputs; y output; w_{ji}, c connection weights

I/O-function:

$$y = f(x_1, \ldots, x_r) = \sum_j c_j \sigma \left(\sum_i w_{ji} x_i - \theta_j \right)$$

θ_j thresholds

σ threshold function (e.g. sigmoidal or gaussian bell)

Example: TD-Gammon

Tesauro 1995

Backgammon program based on standard neural network architecture.
Learns by playing against itself.
Reaches almost professional player's strength by itself.
After some manual adjustements, it is now far beyond human level.

We play Backgammon *symbolically*. The learning of some known strategic insights could be observed when the program was trained.

Can we learn from the program how to play better?

Propositional Logic and Connectionism

Idea:

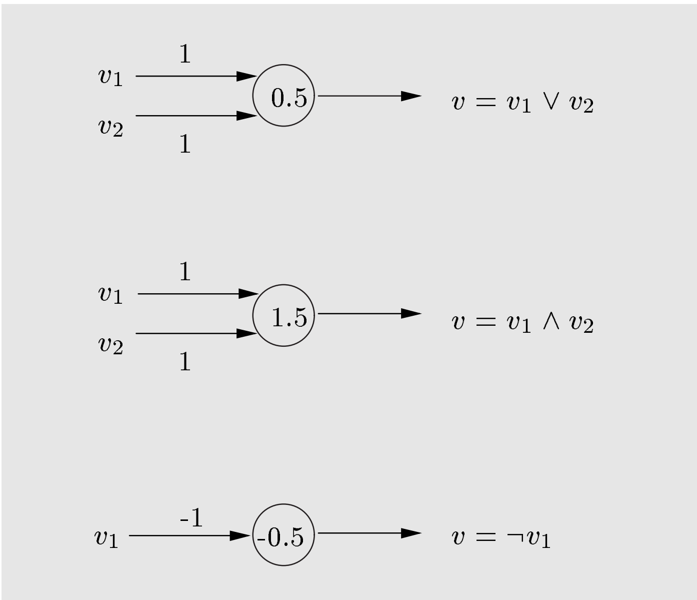
- ▶ Representation of knowledge via network.
  - ▶ Training of the network.
    - ▶ Extraction of learned knowledge.

Issues:

- How to represent the knowledge?
- Standard network architectures preferred.
- How to extract the knowledge?

- How to advance beyond propositional logic?

McCulloch-Pitts networks



McCulloch & Pitts 1943

Neurons with binary threshold functions for $\vee$, $\wedge$, $\neg$.

Updates are being computed for all units at the same time.

McCulloch-Pitts networks are exactly the finite automata.

Picture: Hölldobler, Lecture notes *Introduction to Computational Logic*, 2001

CILP, KBANN

Hölldobler & Kalinke 1994: Representation of propositional logic programs by 3-layer feedforward networks, extending on McCulloch and Pitts.

D'Avila Garcez, Broda, Gabbay, Zaverucha 1999/2001:
Extension to sigmoidal (differentiable) threshold functions.
Learning possible via backpropagation (gradient descent).

CILP system

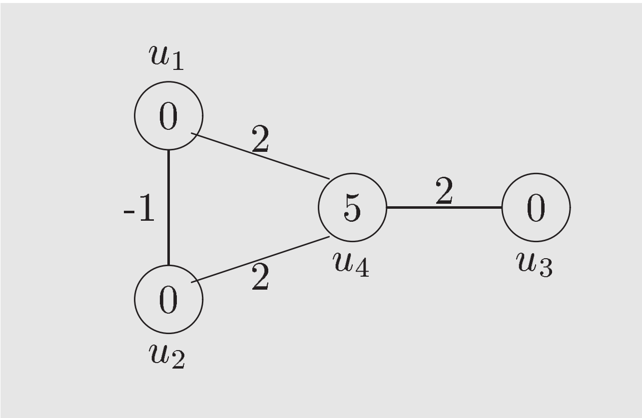
similar: KBANN (Towell and Shavlik 1994)

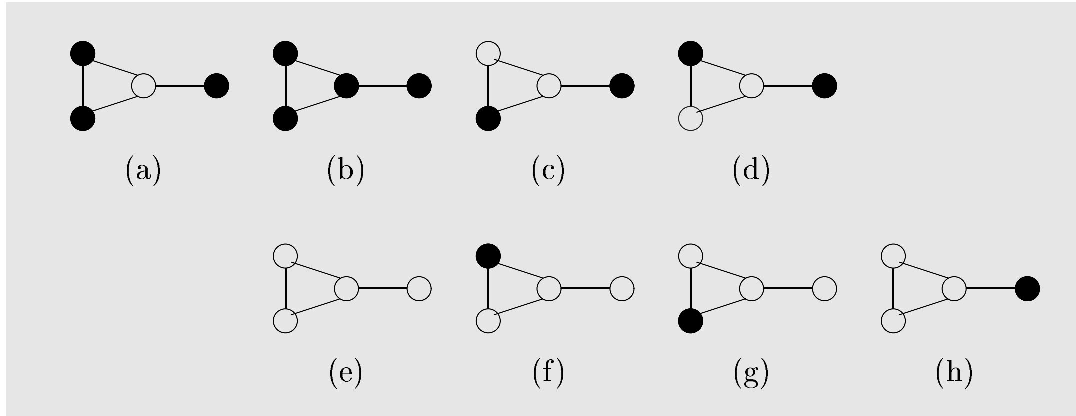
CILP evaluation:

Initiating network using background knowledge increases learning performance.

Extraction of knowledge after learning very difficult.

Symmetric nets and propositional logic





Pinkas 199x: Hopfield networks with symmetric connections.

Update by probabilistic choice of unit.

There exsists relation between stable states in network
and models of propositional formulae (via energy minimization).

▶ Treatment of some non-monotonic propositional logic.

Pictures: Hölldobler, *Introduction to Computational Logic*, 2001

Beyond propositional logic

We need to represent something infinite using finitely many nodes!

Variable bindings?

$$\texttt{male}(x) \wedge \texttt{hasSon}(x, y) \rightarrow \texttt{father}(x)$$

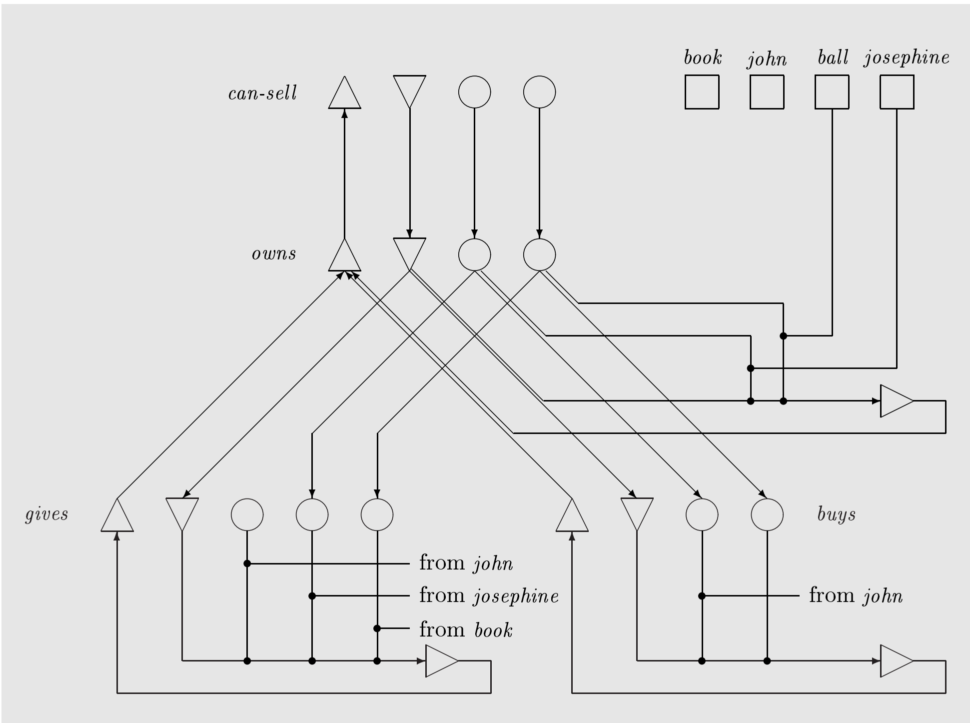
Term representation?

$$\texttt{member}(X, [a, b, c|[d, e]])$$

Infinite ground instantiations?

$$\forall x \forall y (\texttt{male}(x) \wedge \texttt{hasSon}(x, y) \rightarrow \texttt{father}(x))$$

SHRUTI



Shastri & Ajjanagadde 1993

Variable binding
via time synchronization.

Reflexive (i.e. fast)
reasoning possible.

Picture: Hölldobler,
Introduction to
Computational Logic, 2001

gives(X,Y,Z) $\rightarrow$ owns(Y,Z)
buys(X,Y) $\rightarrow$ owns(X,Y)
owns(X,Y) $\rightarrow$ can-sell(X,Y)

gives(john,josephine,book)
($\exists$X) buys(john,X)
owns(josephine,ball)

RAAM architectures

Recursive AutoAssociative Memory.
Pollack 1990 and others.

Holographic memory.
Learning of first order terms, represented as trees.

Performs badly for deeper nestings.

Inferencing/reasoning not studied and appears to be difficult.

Approximation of logic programs

Idea:

Hölldobler, Kalinke, Störr 1999

extended by Hitzler, Hölldobler, Seda JAL 2004

Given (first-order) logic program P.

Represent semantic operator T_P by I/O-function of a neural network.

T_P can be understood to represent the (declarative) meaning of P.

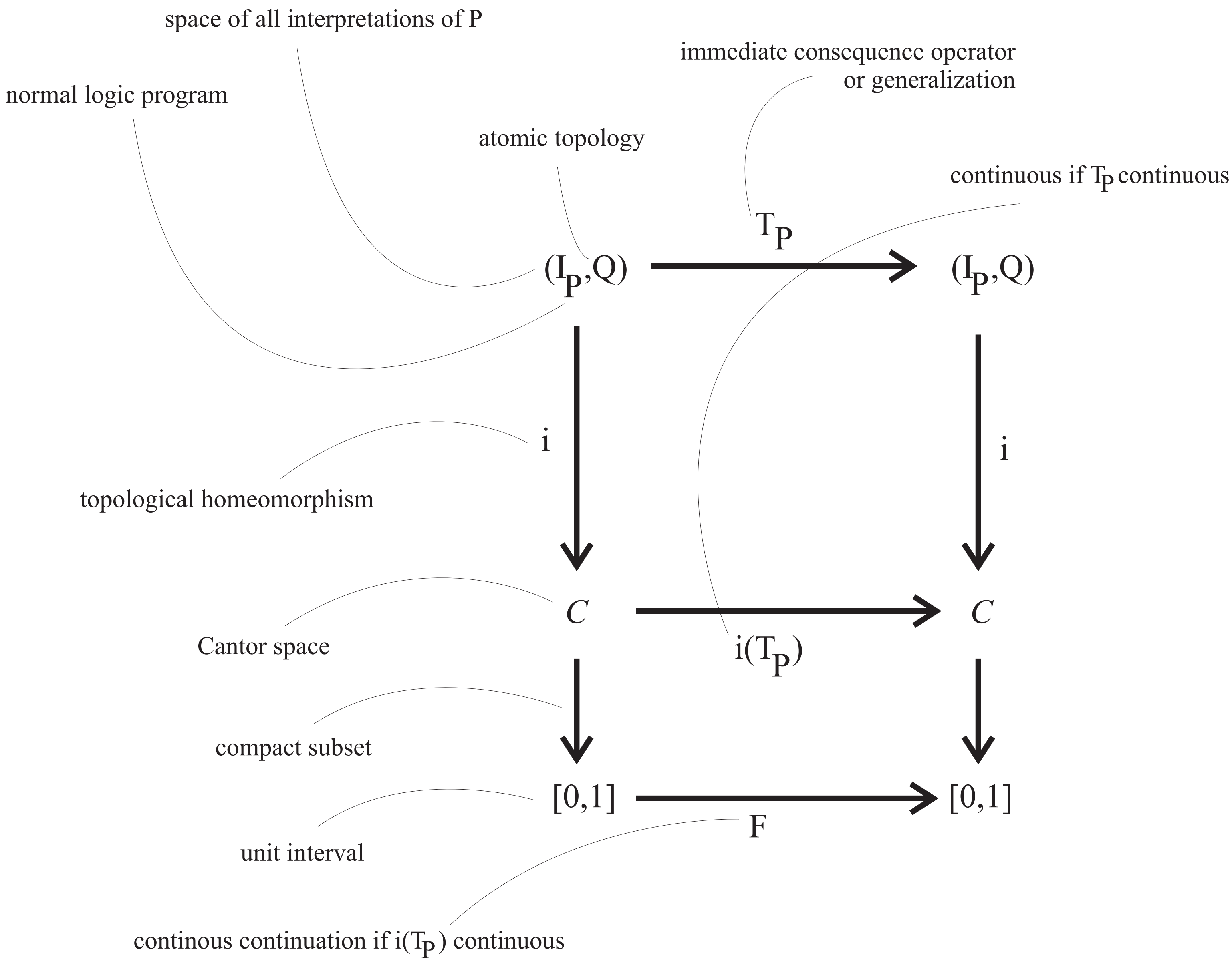
Issues:

T_P needs to be embedded in the reals.

Representation may not be possible, but approximation.

Existence results are already difficult.

Constructing approximating networks is more difficult.


space of all interpretations of P
immediate consequence operator
or generalization
normal logic program
atomic topology
continuous if T_P continuous
T_P
(I_P,Q)
(I_P,Q)
i
i
topological homeomorphism
C
C
Cantor space
i(T_P)
compact subset
[0,1]
[0,1]
F
unit interval
continous continuation if i(T_P) continuous

Continuity

Theorem (Funahashi 1989, simplified version):

σ sigmoidal
$K \subseteq \mathbb{R}$ compact,
$f : K \to \mathbb{R}$ continuous,
$\varepsilon > 0$.

Then there exists perceptron with sigmoidal σ and I/O-function $\bar{f} : K \to \mathbb{R}$ with

$$\max_{x \in K} \left\{ d\left(f(x), \bar{f}(x) \right) \right\} < \varepsilon;$$

d metric which induces natural topology on $\mathbb{R}$.

I.e. every continuous function $f : K \to \mathbb{R}$ can be uniformly approximated by I/O-functions of perceptrons.

Alternative Approach: Self-Similarity

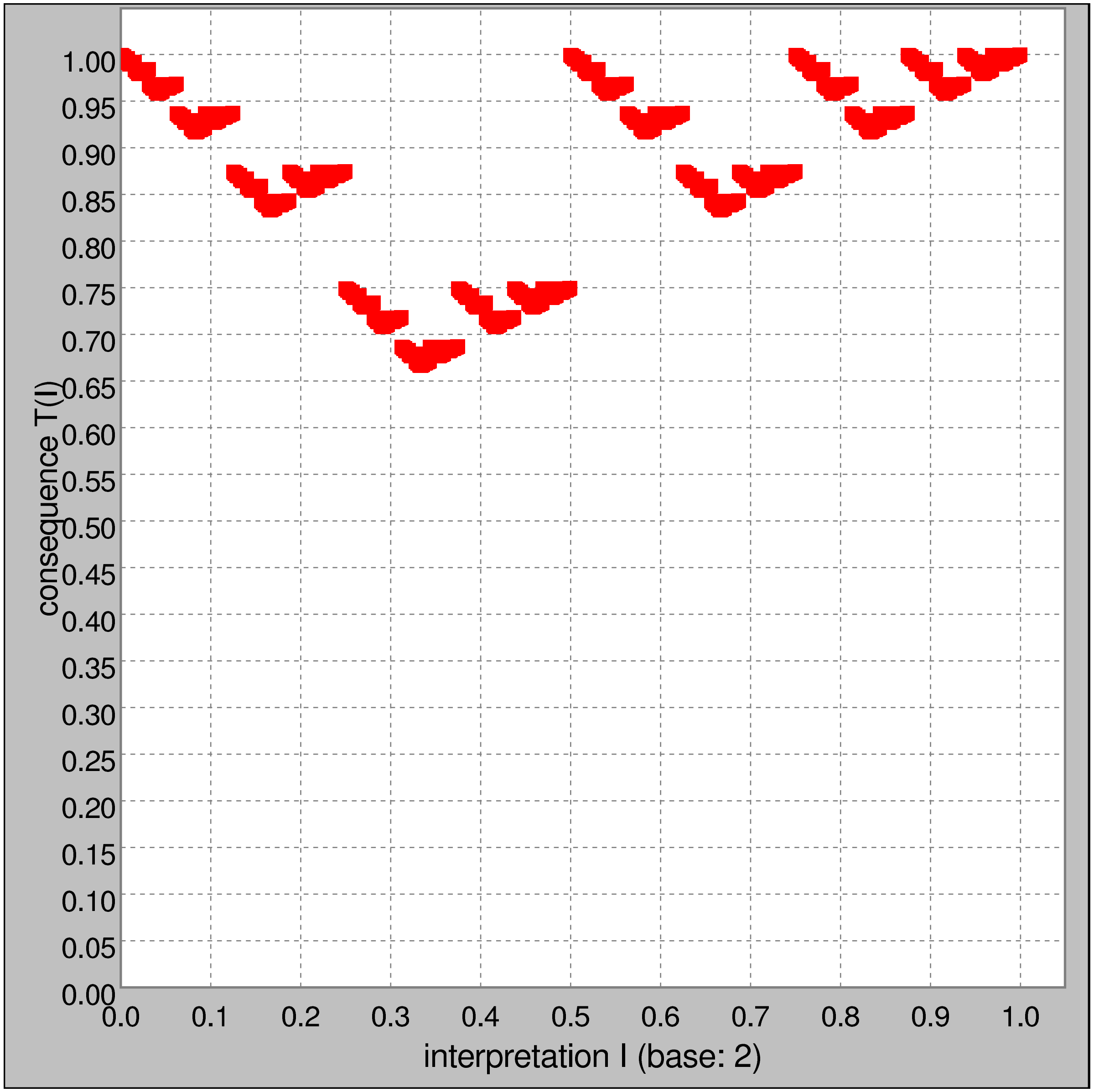
An observation by Sebastian Bader.
Approached worked out by Bader and Hitzler JAL 2004.

Graph of T_P visualized via embedding into $[0,1] \times [0,1]$
using p-adic numbers.

$R : I_P \to \mathbb{R} : I \mapsto \sum_{A \in I} B^{-l(A)}$, where $l : B_P \to \mathbb{N}$ injective, $B > 2$.

Graph shows self-similarity.

(The following pictures were provided by Sebastian Bader.)



p(0).

p(s(X)) :- p(X).

p(X) :- not p(X).

p(s(s(s(s(s(s(s(s(s(s(0))))))))))) => 11

p(s(s(s(s(s(s(s(s(s(0)))))))))) => 10

p(s(s(s(s(s(s(s(s(0))))))))) => 9

p(s(s(s(s(s(s(s(0)))))))) => 8

p(s(s(s(s(s(s(0))))))) => 7

p(s(s(s(s(s(0)))))) => 6

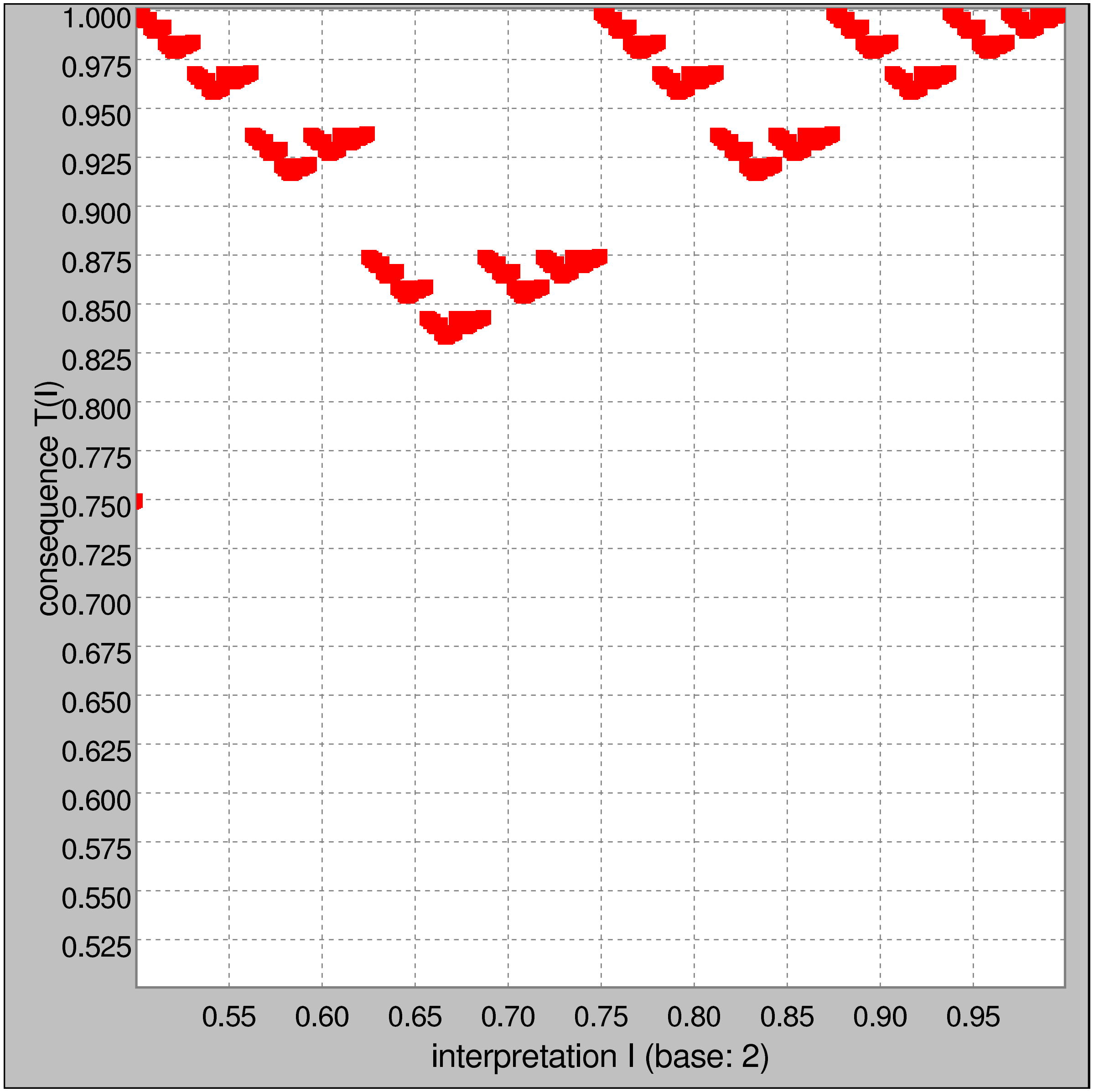
p(s(s(s(s(0))))) => 5

p(s(s(s(0)))) => 4

p(s(s(0))) => 3

p(s(0)) => 2

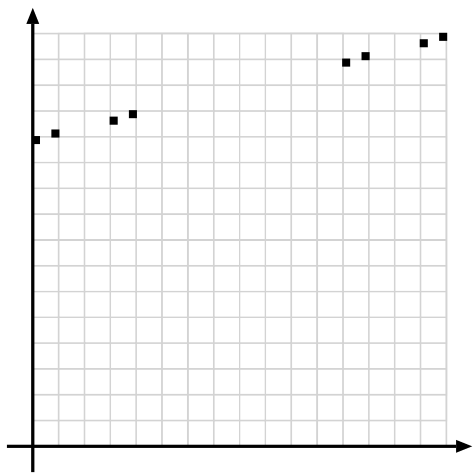
p(0) => 1



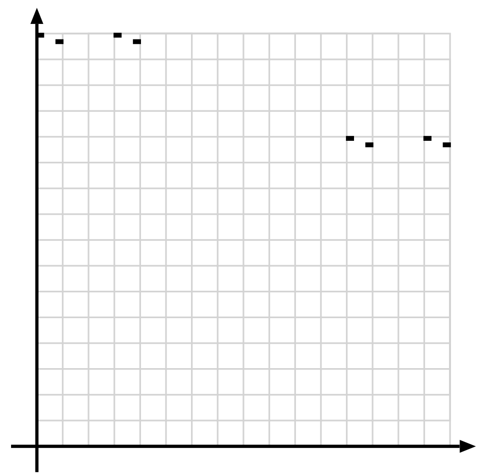

```
p(0).
p(s(X)) :- p(X).
p(X) :- not p(X).
```

```
p(s(s(s(s(s(s(s(s(s(s(0))))))))))) => 11
p(s(s(s(s(s(s(s(s(s(0)))))))))) => 10
p(s(s(s(s(s(s(s(s(0))))))))) => 9
p(s(s(s(s(s(s(s(0)))))))) => 8
p(s(s(s(s(s(s(0))))))) => 7
p(s(s(s(s(s(0)))))) => 6
p(s(s(s(s(0))))) => 5
p(s(s(s(0)))) => 4
p(s(s(0))) => 3
p(s(0)) => 2
p(0) => 1
```

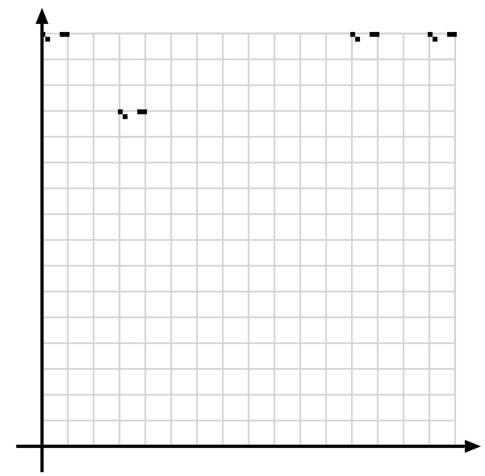
Examples of graphs of logic programs



```
n(0).
n(s(X))←n(X).
```

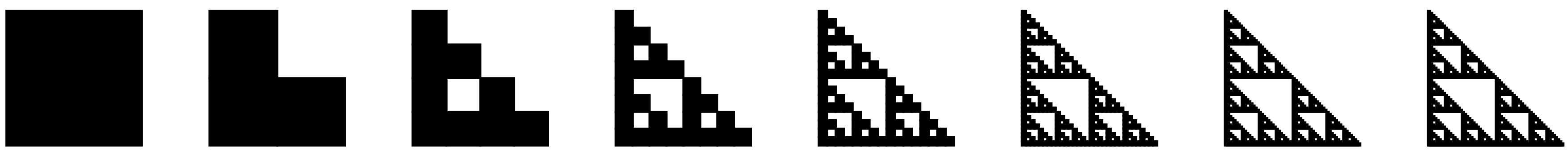


```
e(0).
e(s(X))←not e(X).
o(X)←not e(X).
```



```
p(0).
p(s(X))←p(X).
p(X)←not p(X).
```

(Hyperbolic) Iterated function systems (IFSs)



Space $\mathcal{H}$: Compact subsets of $\mathbb{R}^2$ with *Hausdorff metric*.
Set $\Omega = \{\omega_i\}$ of *contraction mappings* on $\mathbb{R}^2$.
$\bigcup \Omega(A) = \bigcup_i \omega_i(A)$ contraction on $\mathcal{H}$ with unique fixed point (*attractor*).

First representation theorem

P logic program. $R : I_P \to \mathbb{R}$ p-adic embedding.

$\left(\mathbb{R}^2, d, \Omega = \left\{\left(\omega_i^1, \omega_i^2\right)\right\}\right)$ hyperbolic IFS, attractor A.

Then

$$\mathrm{graph}(R(T_P)) = A$$

iff

$$\pi_1(A) = \mathrm{range}(R) \text{ and}$$
$$R(T_P)\left(\omega_i^1(a)\right) = \omega_i^2(a) \text{ for all } a \in \mathrm{graph}(R(T_P)) \text{ and all } i.$$

Second representation theorem

P logic program with Lipschitz-continuous $R(T_P)$.
Then there exists IFS with attractor graph$(R(T_P))$.

Idea: Set $\omega_i^2(x) = R(T_P)(\omega_i^1(x))$.
Choose $\omega_i^1(x)$ such that it generates range(R). This is possible with arbitrarily small contraction, the necessary size of which can be determined by the Lipschitz constant of $R(T_P)$.

Concrete approximation by interpolation

$a \in \mathbb{N}$ *accuracy*.
l injective level mapping (enumeration of B_P).
Interpolation points: $(R(I), R(T_P(I)))$, where $I \in D = \{A \mid l(A) < a\}$.

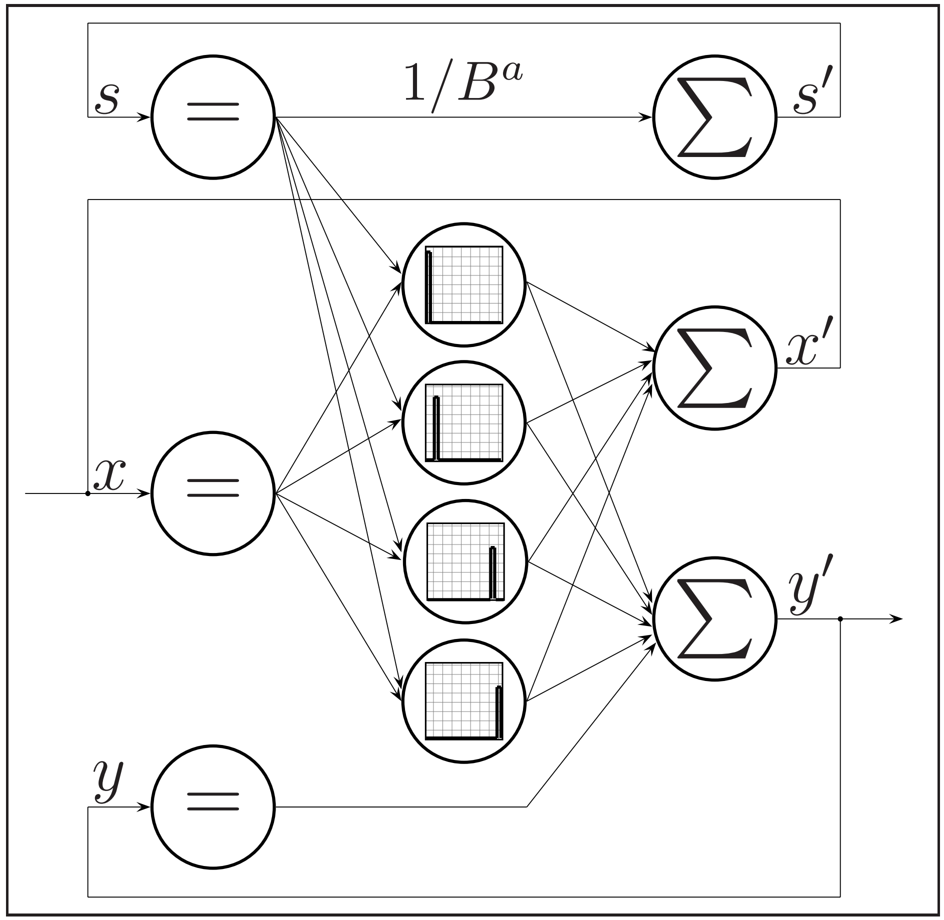
IFS with $\Omega_a = \{(\omega_i^1, \omega_i^2)\}$, where

$$\omega_i^1(x) = \frac{1}{B^a}x + d_i^1$$
$$\omega_i^2(x) = \frac{1}{B^a} + R(T_P)\left(d_i^1\right) - \frac{R(T_P)(0)}{B^a}$$

Attractors A_a are graphs of continuous functions.

$(A_a)_a$ converges in function space (with $\sup$-metric) to $R(T_P)$
if $R(T_P)$ Lipschitz-continuous.

Encoding as radial basis function network



Use case ontology learning

Why should it work?

- Languages (e.g. OWL-DL,DLP) are of finitary nature.
  Dealing with non-decidable fragments of FOL probably not necessary.
- Propositional case already interesting
  e.g. for learning hierarchies.
- Neural-symbolic approaches lend themselves naturally to dealing with noisy and/or probabilistic data.

Use case ontology learning

Issues

- Need symbolic representations to start with.
  Text can be preprocessed using e.g. TextTo/2Onto.
  Other domains (e.g. bioinformatics) may be accessible more directly.
- Extensive research concerning connectionist representations of
    logic programs exist already.
  DLP appears to be a good starting point.
  Suitable connectionist paradigm has yet to be developed.
- Other ideas? Neural Gas?

Current Actions

- Investigate the use of CILP for ontology learning.
  (with S. Bader)
- Investigate using ILP for knowledge extraction.
  (with S. Bader, S. Hölldobler and J. Lehmann)
- Advance first-order representations with perceptrons.
  (with S. Bader, S. Hölldobler and A. Witzel)

- Workshop NeSy'05 at IJCAI-05.
  (with A. Garcez and J. Elman)
- Course at ESSLLI 2005 summer school.
  (with S. Bader and S. Hölldobler)
- International research proposal in the making.
  (coordinated by A. Garcez)