Symbolic knowledge representation with artificial neural networks

Sebastian Bader[1] **and Pascal Hitzler**[2]

[1]**ICCL, TU Dresden, Germany**
[2]AIFB, **Universität Karlsruhe, Germany**

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.

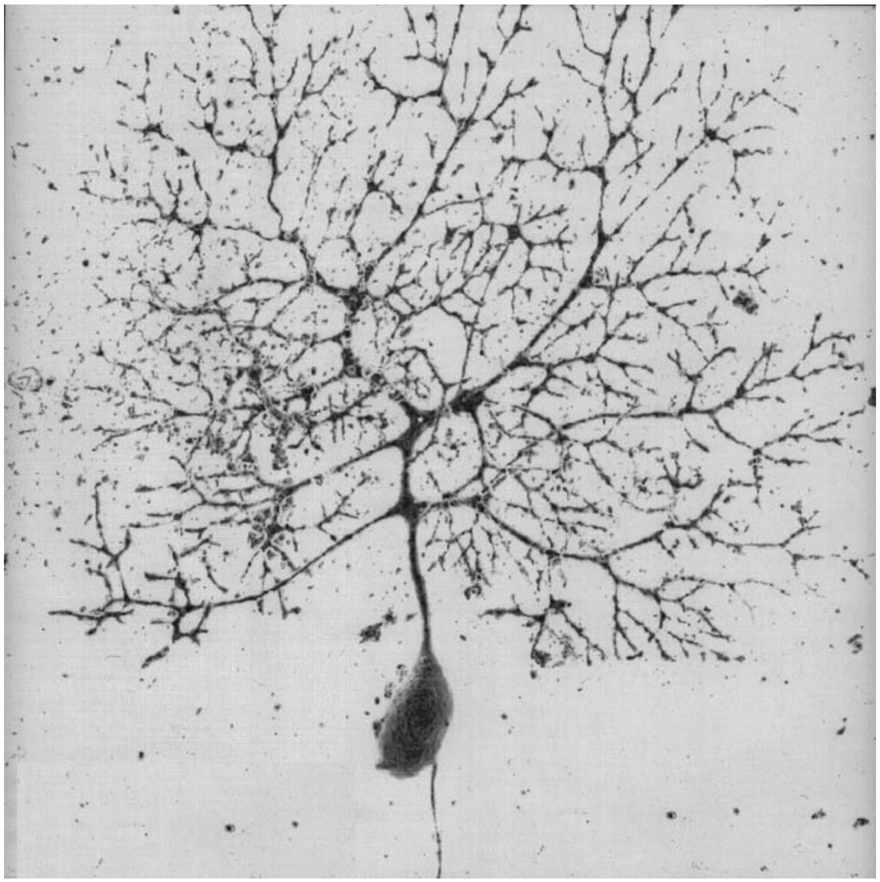
We seek intergrated paradigms retaining the best of both worlds!

Contents

- Artificial Neural Networks
- The challenge: Representing first-order logic programs
- Logic programs and iterated function systems
- Logic programs as recurrent neural networks

The second author acknowledges support by the Boole Center for Research in Informatics, BCRI, Cork, Ireland, for presenting this work.

Biological neural nets



Neuron,
with dendrites, soma, and axon.

(Purkinje cell from cerebellum)

Picture:
Spektrum der Wissenschaft 10,
October 2001

Artificial neural nets/Connectionist systems

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

In particular:

- Every unit computes a *simple* real 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*.

The first-order challenge

We need to represent something infinite using finitely many nodes/weights!

Variable bindings?

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

Term representation?

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

Infinite ground instantiations?

$$\forall x(\texttt{prime}(x) \land \neg\texttt{equalTo}(x,2) \rightarrow \texttt{odd}(x))$$

Approach taken

Idea:

Hölldobler, Kalinke, Störr 1999

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.

$T_P : I_P \to I_P$
where $I_P = 2^{B_P} \approx 2^{\mathbb{N}}$ is *space of all interpretations*.

T_P *computes one-step consequences along* $\leftarrow$.

$$\texttt{odd}(x) \leftarrow \texttt{prime}(x) \wedge \neg\texttt{equalTo}(x, 2)$$

Self-Similarity of T_P

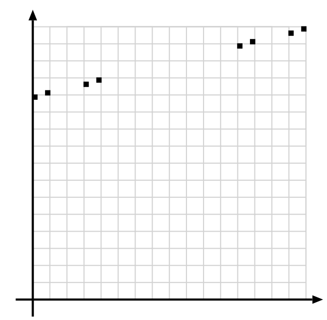
Graph of T_P visualized via embedding into $[0,1] \times [0,1]$.

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

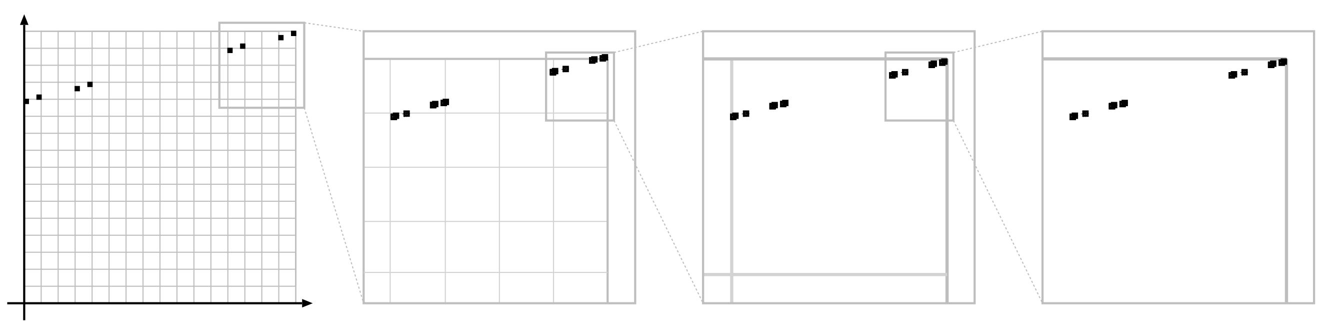
Representation of T_P-operator in the reals:

```
n(0).
n(s(X)) ← n(X).
```

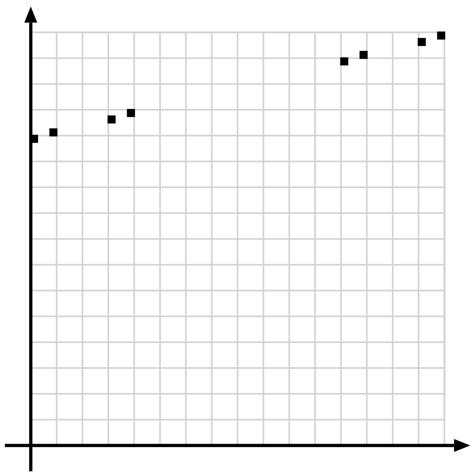
$\Rightarrow$



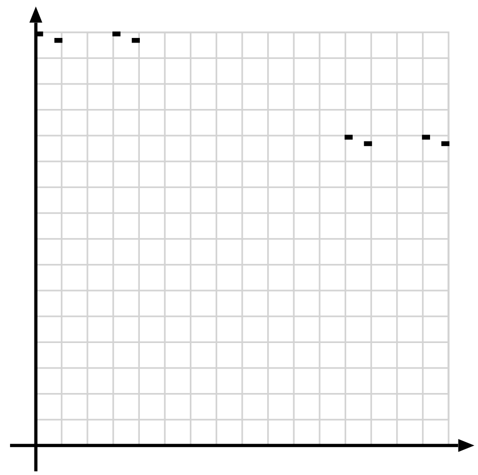
Shows self-similarity by zooming in:



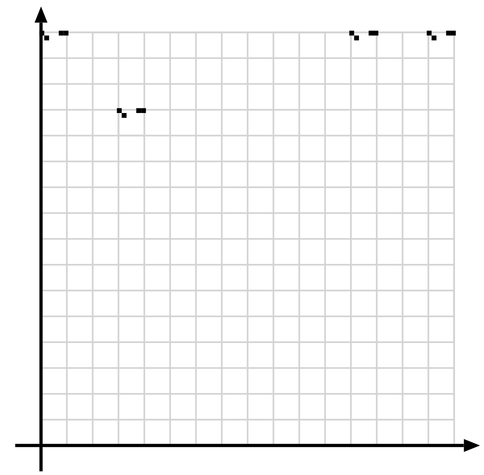
Examples of graphs of logic programs



```
n(0).
n(s(X))←n(X).
```



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

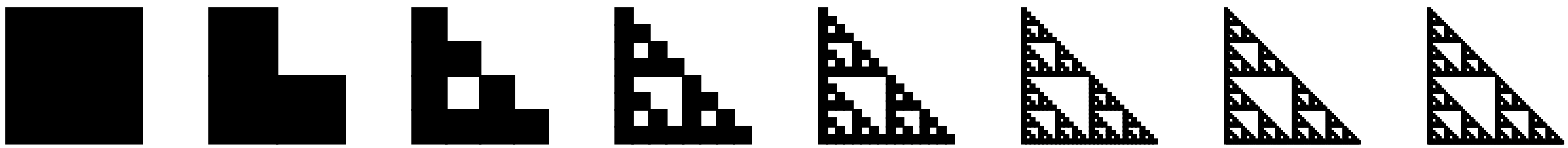


```
p(0).
p(s(X))←p(X).
p(X)←not p(X).
```

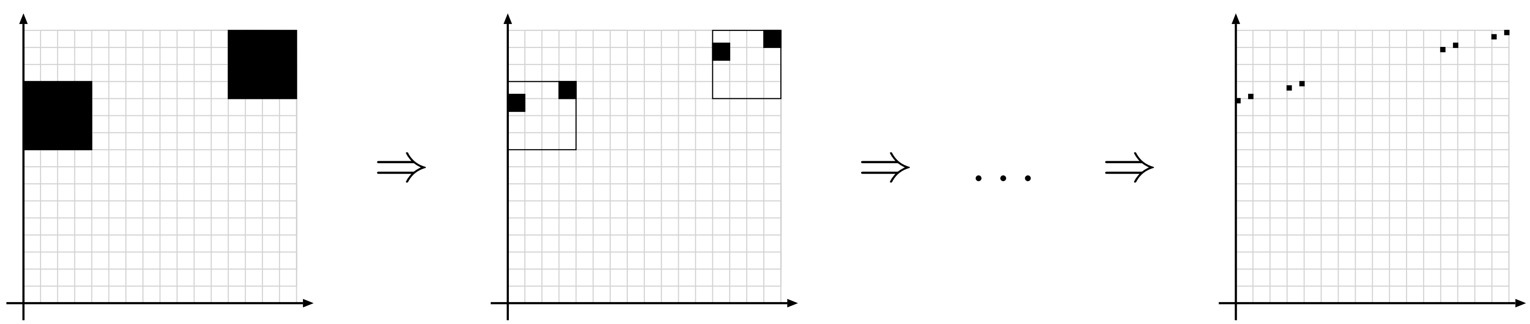
Self-similarity observed for *all* programs.

(Hyperbolic) Iterated function systems (IFSs)

Can be used for generating self-similar images, e.g. the *Sierpinski Triangle*:



Idea: Use it for generating graph of T_P.

IFS: 

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

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

iff

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

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

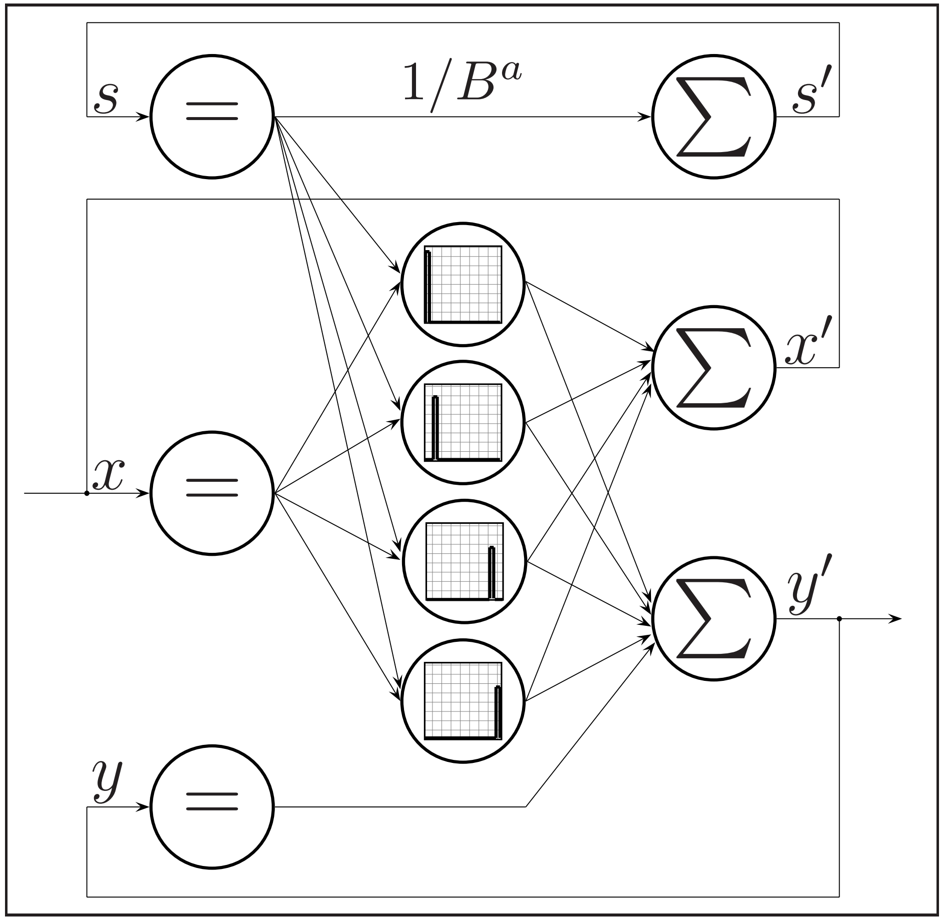
$$\omega_i^1(x) = \frac{1}{B^a}x + d_i^1$$

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

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



Neural-symbolic integration research

We need to find constructive representations using standard architectures.

We need to study learning and information extraction.

We need to develop use cases to guide our research.

Recently: learning of ontologies for the Semantic Web.