

Integrating Logic Programs and Connectionist Systems

A Constructive Approach

Sebastian Bader^{1*}, Pascal Hitzler^{2†}, **Andreas Witzel**^{3‡}

¹International Center for Computational Logic, Technische Universität Dresden, Germany

²AIFB, Universität Karlsruhe, Germany

³Department of Computer Science, Technische Universität Dresden, Germany

*Sebastian Bader is supported by the GK334 of the German Research Foundation (DFG).

†Pascal Hitzler is supported by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project, and by the European Union under the KnowledgeWeb Network of Excellence.

‡Andreas Witzel is supported by Freunde und Förderer der Informatik an der TU Dresden e.V.

- 1 Motivation
- 2 Approximating Logic Programs
- 3 Multi-Layer Feed-Forward Networks
- 4 Radial Basis Function (RBF) Networks
- 5 Conclusions

Logic Programs (LP)

- well-defined semantics
- human-readable
- human-writable

Connectionist Systems (CS)

- robust
- adaptive
- trainable

Goal:

- Integrate both paradigms in order to exploit all advantages

One step towards achieving this goal:

- Transform LP into CS

What we have so far:

- Constructions for Propositional LP
- Non-constructive proofs for First-Order LP

In this work:

- Constructions for First-Order LP

A Simple Example

- A **Logic Program** P

```
even(0).                % 0 is an even number
even(s(X)) ← not even(X). % the successor of a
                        % non-even X is even
```

- The **Herbrand Base** \mathcal{B}_P and some **Interpretations**

$$\begin{aligned}\mathcal{B}_P &= \{ \text{even}(0), \text{even}(s(0)), \text{even}(s^2(0)), \dots \} \\ I_1 &= \{ \text{even}(0), \text{even}(s(0)) \} \\ I_2 &= \{ \text{even}(0), \text{even}(s^3(0)), \text{even}(s^4(0)), \text{even}(s^5(0)), \dots \}\end{aligned}$$

- The **Single-Step Operator** or **Meaning Function** T_P

$$\begin{aligned}I_1 &\xrightarrow{T_P} I_2 \xrightarrow{T_P} \{ \text{even}(0), \text{even}(s^2(0)), \text{even}(s^3(0)) \} \\ &\xrightarrow{T_P} \dots \xrightarrow{T_P} \{ \text{even}(0), \text{even}(s^2(0)), \text{even}(s^4(0)), \\ &\quad \text{even}(s^6(0)), \text{even}(s^8(0)), \text{even}(s^{10}(0)), \dots \}\end{aligned}$$

Embedding T_P in \mathbb{R}

- Enumerate \mathcal{B}_P using $\|\cdot\| : \mathcal{B}_P \rightarrow \mathbb{N} \setminus \{0\}$

$$\|\text{even}(s^n(0))\| := n + 1$$

- Embed $I \in \mathcal{J}_P$ into \mathbb{R} using $R(I) := \sum_{A \in I} 3^{-\|A\|}$

$$R(\{\text{even}(0), \text{even}(s^2(0))\}) = 0.10100\dots_3$$

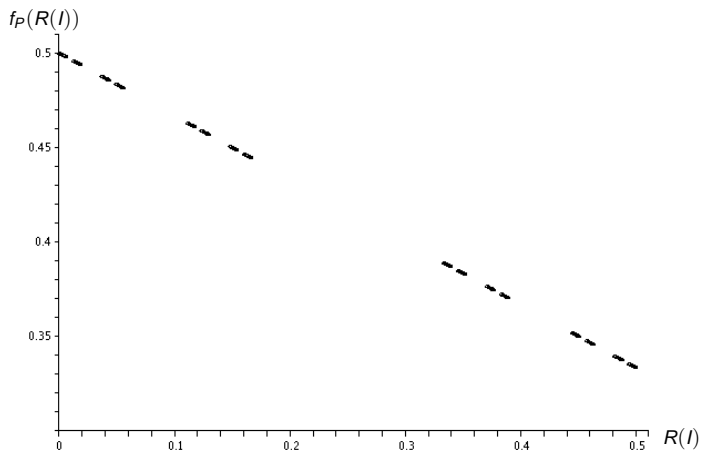
$$\begin{array}{c} \text{even}(s^4(0)) \\ \text{even}(s^3(0)) \\ \text{even}(s^2(0)) \\ \text{even}(s(0)) \\ \text{even}(0) \end{array}$$

- Embed T_P into \mathbb{R} :

$$\begin{array}{ccc} I \in \mathcal{J}_P & \xrightarrow{T_P} & I' \in \mathcal{J}_P \\ \uparrow R^{-1} & & \downarrow R \\ x \in D_f & \xrightarrow{f_P} & x' \in D_f \end{array}$$

where $D_f := \{R(I) \mid I \in \mathcal{J}_P\}$

Embedding of the Example Program



In general, the graph is more complicated and not on a straight line!

Idea for Approximating f_P

- **Goal:** approximate f_P (the embedded T_P) up to ε
- Consider $x, x' \in D_f$:

$$x = 0.\underbrace{001010111010000000}_{l \text{ digits are equal}} \dots_3$$

$$x' = 0.\overbrace{001010111010111111} \dots_3$$

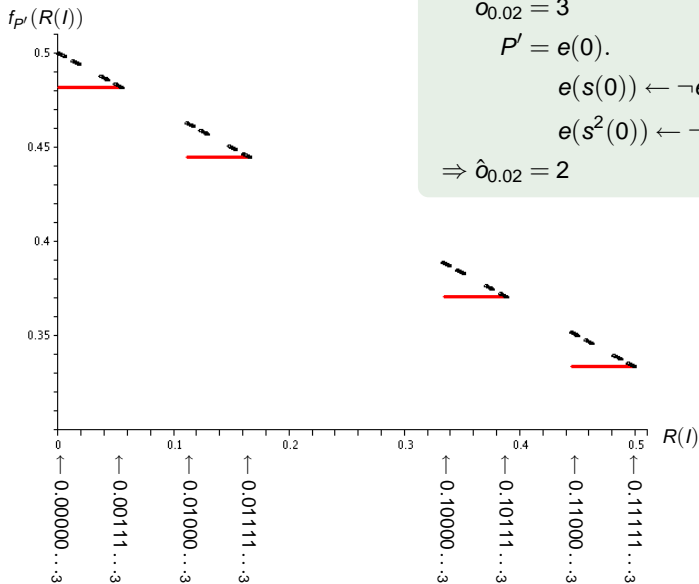
Maximum difference $\delta_l := \sum_{i>l} 3^{-i} = \frac{1}{3^{l+1}}$

- **Greatest relevant output level** $o_\varepsilon := \min \{n \in \mathbb{N} \mid \delta_n < \varepsilon\}$
- Assume $T_{P'}$ and T_P agree on all atoms of level $\leq o_\varepsilon$
 - $\Rightarrow f_{P'}$ and f_P agree on the first o_ε digits
 - $\Rightarrow f_{P'}$ **approximates** f_P up to ε

The Instance of P up to o_ϵ

- **Goal:** find P' such that $T_{P'}$ and T_P agree on atoms of level $\leq o_\epsilon$
- Inclusion of A in $T_P(I)$ depends only on clauses with **head A**
- $P' := \{A \leftarrow B \in \mathcal{G}(P) \mid \|A\| \leq o_\epsilon\}$
where $\mathcal{G}(P) :=$ set of all ground instances of clauses from P
- P' is **finite** if P is covered, i.e. if there are no local variables
- **Greatest relevant input level**
 $\hat{o}_\epsilon := \max \{ \|L\| \mid L \text{ is body literal of some clause in } P' \}$
- $T_{P'}$ depends only on atoms of level $\leq \hat{o}_\epsilon$
 - $\Rightarrow f_{P'}$ depends only on the first \hat{o}_ϵ digits
 - $\Rightarrow f_{P'}$ is constant for all inputs which agree on first \hat{o}_ϵ digits
 - $\Rightarrow f_{P'}$ consists of **finitely many constant pieces**

Our Example with $\varepsilon = 0.02$



$$\alpha_{0.02} = 3$$

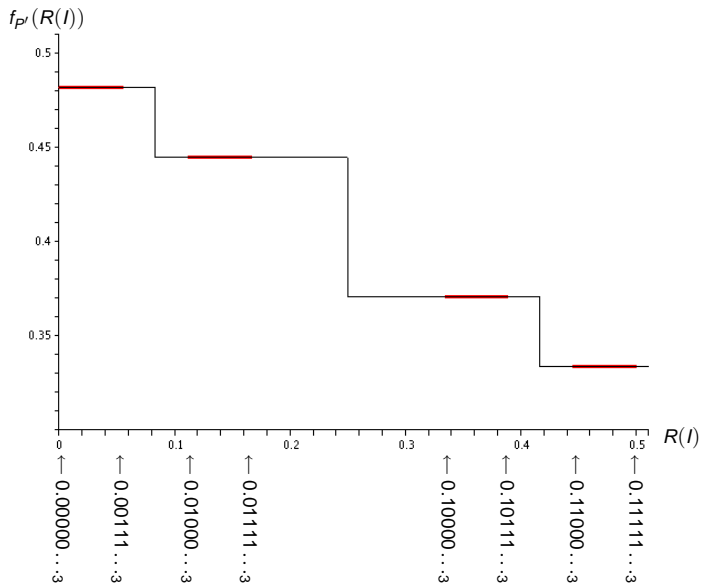
$$P' = e(0).$$

$$e(s(0)) \leftarrow \neg e(0).$$

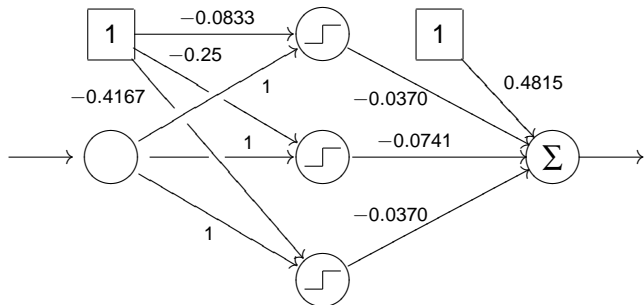
$$e(s^2(0)) \leftarrow \neg e(s(0)).$$

$$\Rightarrow \hat{\alpha}_{0.02} = 2$$

Building a CS with Step Activation Functions

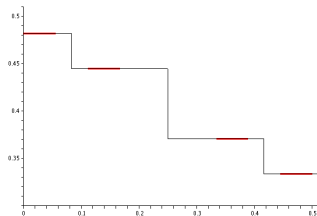


The Resulting CS



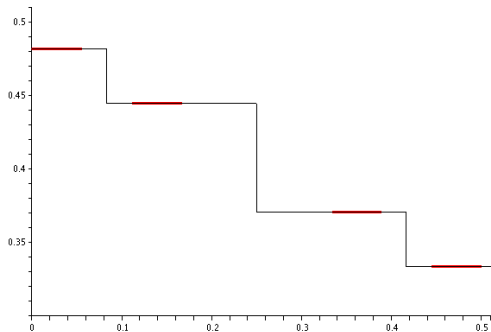
Each  computes

- 0, if weighted sum of inputs ≤ 0
- 1, otherwise



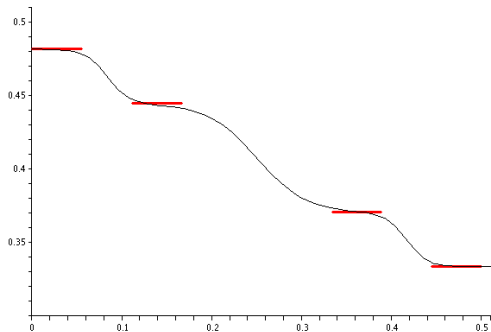
Building a CS with Sigmoidal Activation Functions

Approximate the step functions



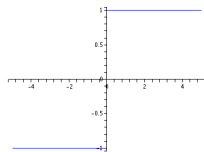
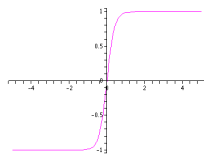
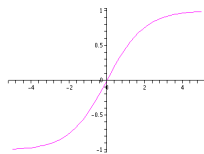
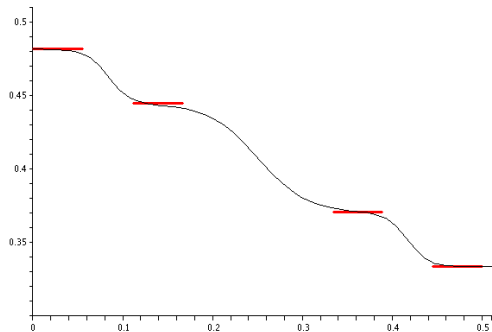
Building a CS with Sigmoidal Activation Functions

Approximate the step functions by sigmoids



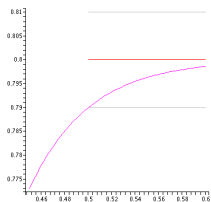
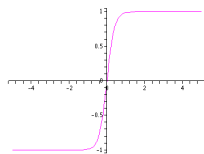
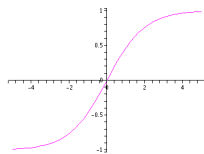
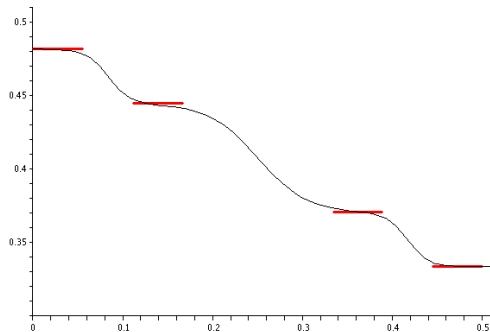
Building a CS with Sigmoidal Activation Functions

Approximate the step functions by sigmoids



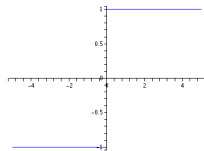
Building a CS with Sigmoidal Activation Functions

Approximate the step functions by sigmoids



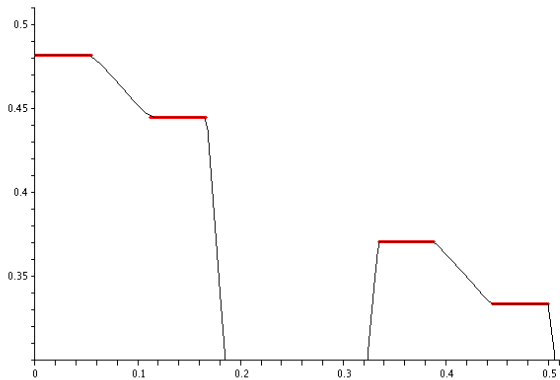
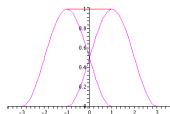
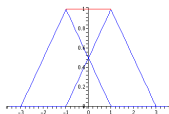
Divide ε into ε' for P' and ε'' for the sigmoids

The closest constant piece yields the zoom-out factor



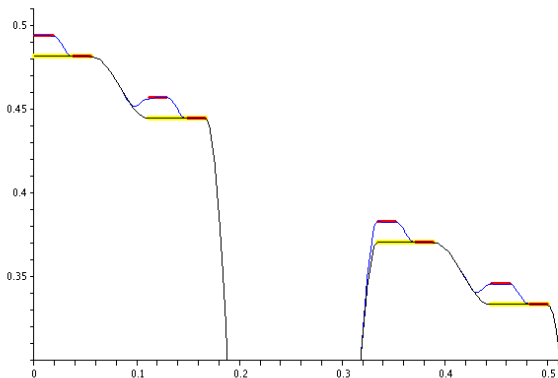
A CS with Triangle or Raised-Cosine Activation Functions

Describe each constant piece by two triangles or raised cosines:



Refining an Existing Network

- Decreasing ϵ will only add clauses to P'
- Consequence:
 - Constant pieces may be divided into smaller pieces
 - Some parts may be raised
- For $\epsilon = 0.007$, we get:



Conclusions and Problems

What we had before:

- Methods to construct CS for propositional LP
- Non-constructive proofs for the existence of CS approximating first-order LP

New results:

- Methods for constructing CS approximating first-order LP
- Method for iterative refinement

Problem:

- Floating point precision in real computers is very limited, so we can represent only few atoms

Possible remedy:

- Distribute representation on several input/output nodes

Conclusions and Problems

What we had before:

- Methods to construct CS for propositional LP
- Non-constructive proofs for the existence of CS approximating first-order LP

New results:

- Methods for constructing CS approximating first-order LP
- Method for iterative refinement

Problem:

- Floating point precision in real computers is very limited, so we can represent only few atoms

Possible remedy:

- Distribute representation on several input/output nodes

Thank you for your attention.