Computational Logic in Artificial Neural Networks

Ekaterina Komendantskaya

INRIA Sophia Antipolis, France

EPSRC Interview, 6th of March 2008

Outline

Outline

Outline

Outline

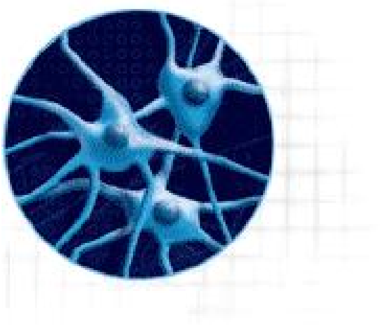
Introduction

Symbolic Logic as Deductive System

- Deduction in logic calculi;
- Logic programming;
- Higher-order proof assistants...

Sound symbolic methods we can trust

Neural Networks



- spontaneous behavior;
- learning and adaptation;
- computational power

Corner-stone Result, [Kalinke, Hölldobler, 94]

Theorem

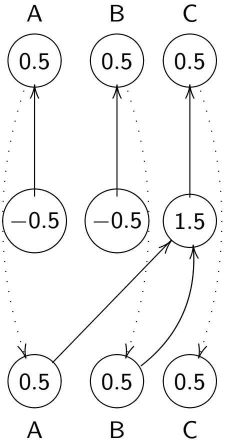
For each propositional program P, there exists a 3-layer feedforward neural network which computes T_P.

We will call such neural networks T_P-neural networks.

A simple T_P-neural network

$B \leftarrow$
$A \leftarrow$
$C \leftarrow A, B$

$T_P \uparrow 0 = \{B, A\}$
$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$



Characteristic Properties of T_P-Neural Networks

1. Require infinitely long layers in the first-order case.

Characteristic Properties of T_P-Neural Networks

1. Require infinitely long layers in the first-order case.
2. First-order atoms are not represented in the neural network directly, and only truth values 1 and 0 are propagated.

Characteristic Properties of T_P-Neural Networks

1. Require infinitely long layers in the first-order case.
2. First-order atoms are not represented in the neural network directly, and only truth values 1 and 0 are propagated.
3. No learning or adaptation.

Characteristic Properties of T_P-Neural Networks

1. Require infinitely long layers in the first-order case.
2. First-order atoms are not represented in the neural network directly, and only truth values 1 and 0 are propagated.
3. No learning or adaptation.

$\Longrightarrow$ Impractical for Computational Logic; not interesting for Neurocomputing audience.

Example 2

$$P(0) \leftarrow$$
$$P(s(x)) \leftarrow P(x)$$
$$T_P \uparrow 0 = \{P(0)\}$$
$$lfp(T_P) = T_P \uparrow \omega =$$
$$\{0, s(0), s(s(0)),$$
$$s(s(s(0))), \ldots\}$$

Example 2

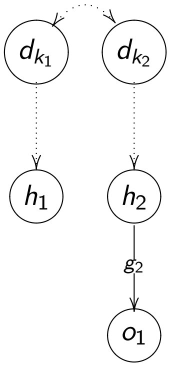
$P(0) \leftarrow$
$P(s(x)) \leftarrow P(x)$
$T_P \uparrow 0 = \{P(0)\}$
$lfp(T_P) = T_P \uparrow \omega =$
$\{0, s(0), s(s(0)),$
$s(s(s(0))), \ldots\}$

Paradox:
(computability,
complexity,
proof theory)

I propose SLD Neural networks

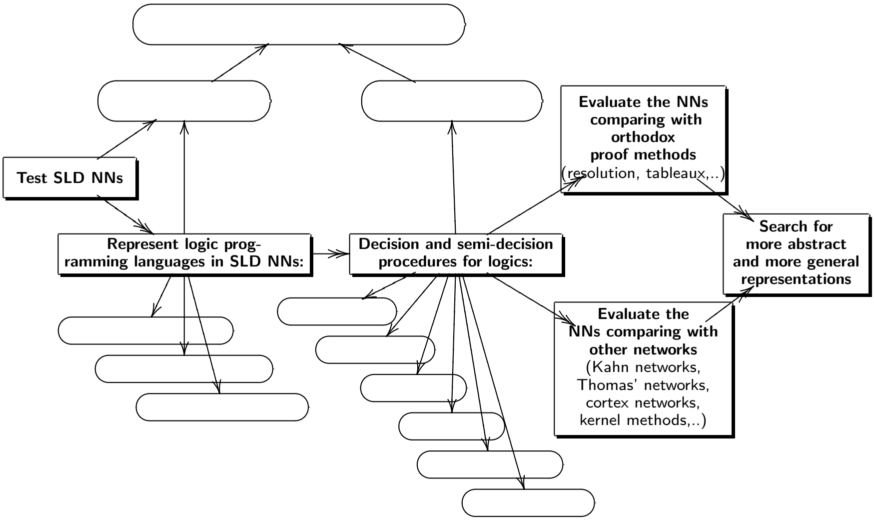
- They have finite architecture that does not depend on the size of the Herbrand base B_P.
- Their effectiveness is due to several learning functions.
- Allow easy implementation of computational logic.

Example 2 in SLD neural networks

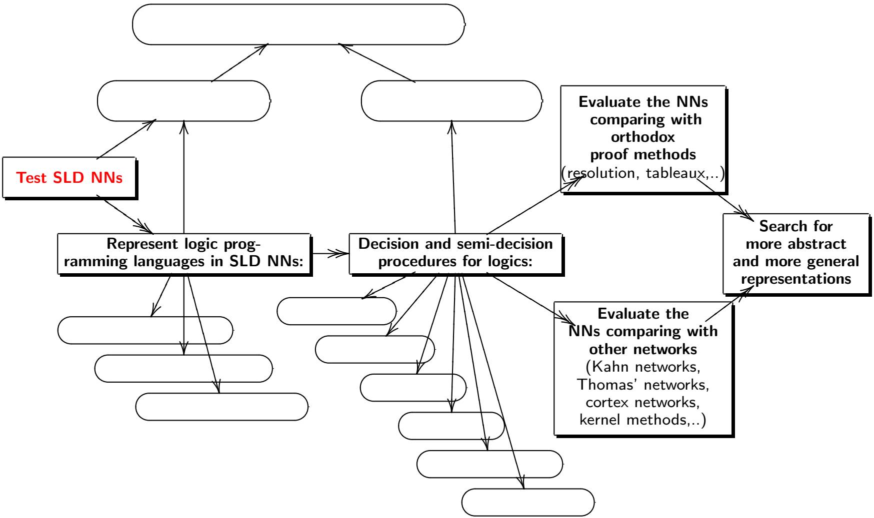


$$P(0) \leftarrow;$$
$$P(s(x)) \leftarrow P(x).$$

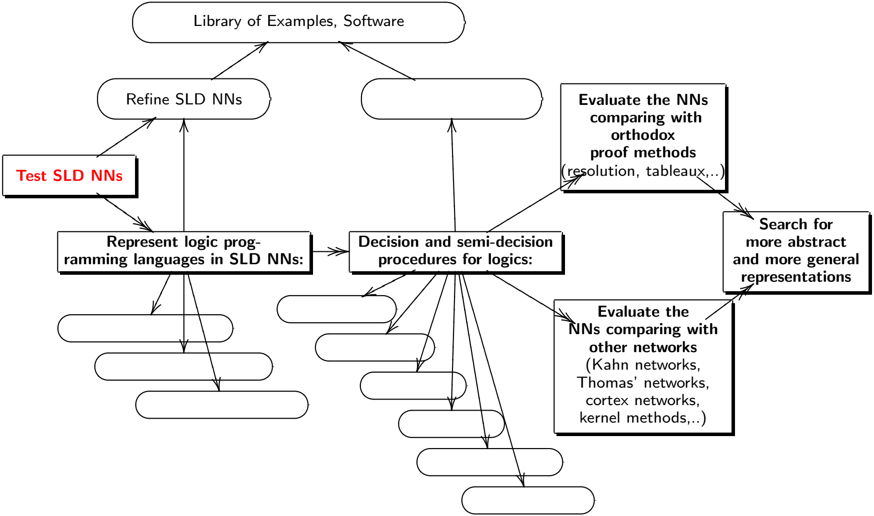
Workplan


Evaluate the NNs comparing with orthodox proof methods (resolution, tableaux,..)
Test SLD NNs
Search for more abstract and more general representations
Represent logic programming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Evaluate the NNs comparing with other networks (Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

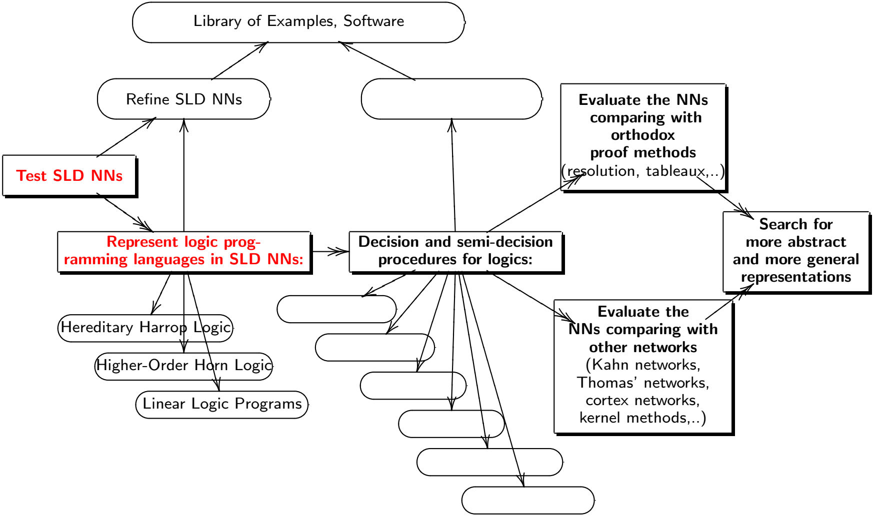
Workplan


Evaluate the NNs comparing with orthodox proof methods (resolution, tableaux,..)
Test SLD NNs
Search for more abstract and more general representations
Represent logic programming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Evaluate the NNs comparing with other networks (Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

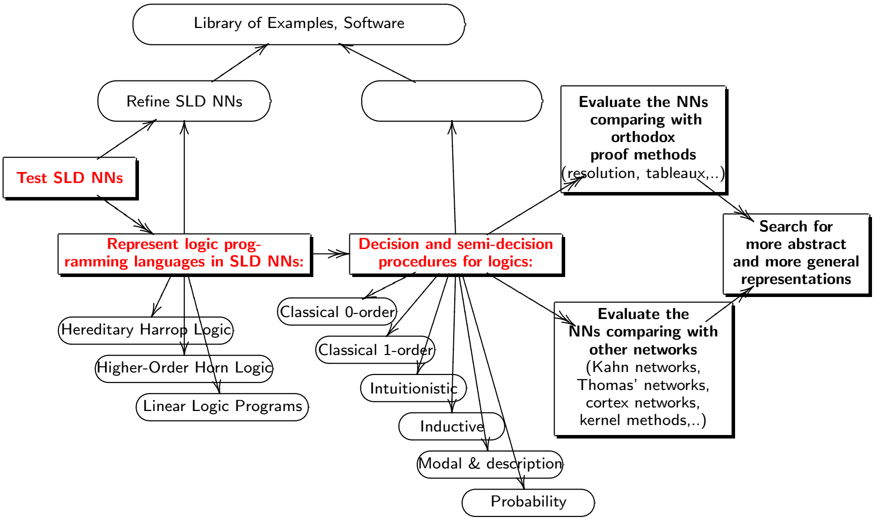
Workplan


Library of Examples, Software
Refine SLD NNs
Test SLD NNs
Represent logic programming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Evaluate the NNs comparing with orthodox proof methods (resolution, tableaux,..)
Search for more abstract and more general representations
Evaluate the NNs comparing with other networks (Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

Workplan

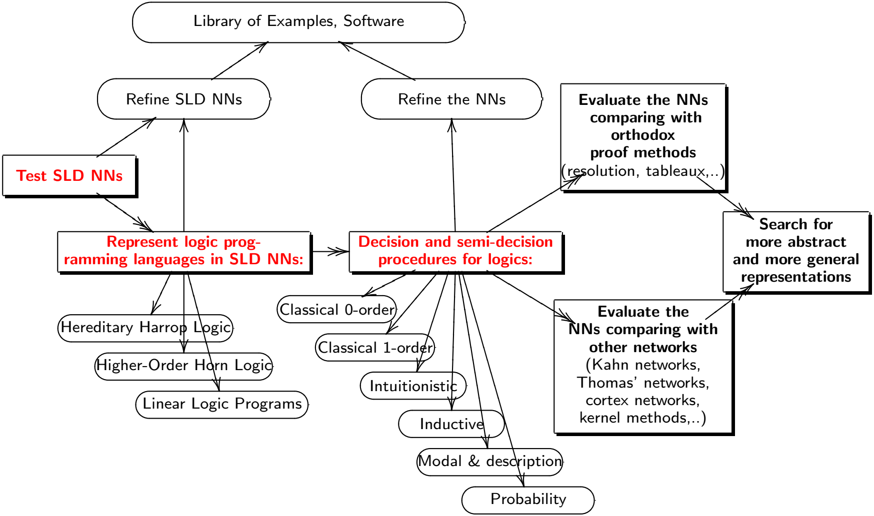

Library of Examples, Software
Refine SLD NNs
Test SLD NNs
Evaluate the NNs comparing with orthodox proof methods
(resolution, tableaux,..)
Search for more abstract and more general representations
Represent logic prog-ramming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Hereditary Harrop Logic
Higher-Order Horn Logic
Linear Logic Programs
Evaluate the NNs comparing with other networks
(Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

Workplan



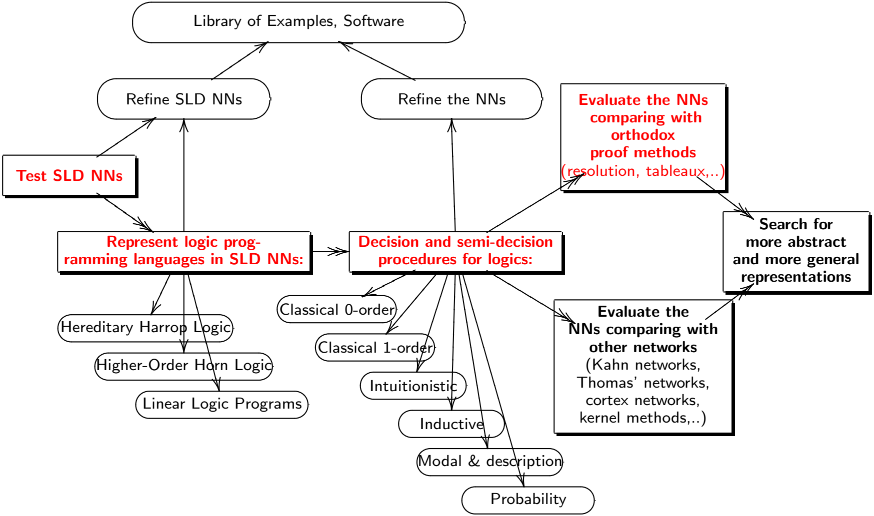
Library of Examples, Software
Refine SLD NNs
Test SLD NNs
Represent logic programming languages in SLD NNs:
Hereditary Harrop Logic
Higher-Order Horn Logic
Linear Logic Programs
Decision and semi-decision procedures for logics:
Classical 0-order
Classical 1-order
Intuitionistic
Inductive
Modal & description
Probability
Evaluate the NNs comparing with orthodox proof methods (resolution, tableaux,..)
Search for more abstract and more general representations
Evaluate the NNs comparing with other networks (Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

Workplan

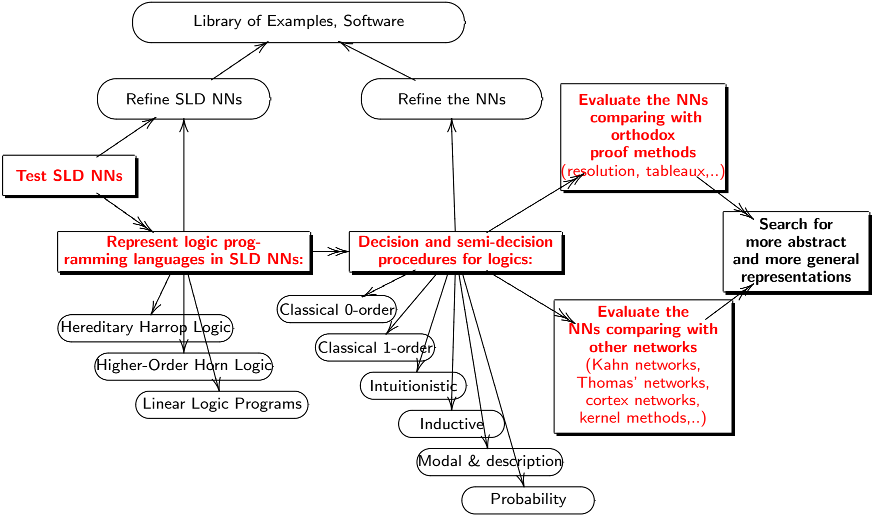


Library of Examples, Software
Refine SLD NNs
Refine the NNs
Evaluate the NNs comparing with orthodox proof methods (resolution, tableaux,..)
Test SLD NNs
Search for more abstract and more general representations
Represent logic prog- ramming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Hereditary Harrop Logic
Higher-Order Horn Logic
Linear Logic Programs
Classical 0-order
Classical 1-order
Intuitionistic
Inductive
Modal & description
Probability
Evaluate the NNs comparing with other networks (Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

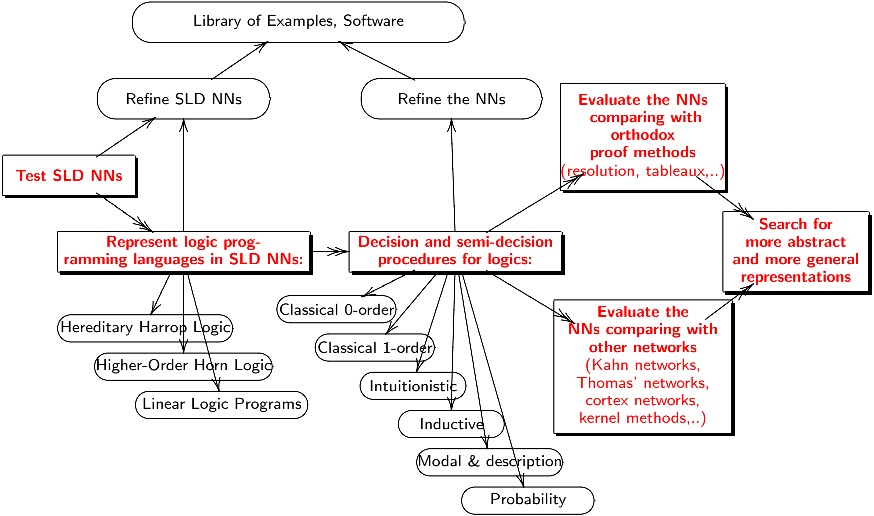
Workplan


Library of Examples, Software
Refine SLD NNs
Refine the NNs
Evaluate the NNs comparing with orthodox proof methods
(resolution, tableaux,..)
Test SLD NNs
Search for more abstract and more general representations
Represent logic prog- ramming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Classical 0-order
Hereditary Harrop Logic
Classical 1-order
Higher-Order Horn Logic
Intuitionistic
Linear Logic Programs
Inductive
Modal & description
Probability
Evaluate the NNs comparing with other networks
(Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

Workplan


Library of Examples, Software
Refine SLD NNs
Refine the NNs
Evaluate the NNs comparing with orthodox proof methods (resolution, tableaux,..)
Test SLD NNs
Search for more abstract and more general representations
Represent logic programming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Classical 0-order
Hereditary Harrop Logic
Classical 1-order
Higher-Order Horn Logic
Intuitionistic
Linear Logic Programs
Inductive
Modal & description
Probability
Evaluate the NNs comparing with other networks (Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

Workplan


Library of Examples, Software
Refine SLD NNs
Refine the NNs
Evaluate the NNs comparing with orthodox proof methods
(resolution, tableaux,..)
Test SLD NNs
Search for more abstract and more general representations
Represent logic programming languages in SLD NNs:
Decision and semi-decision procedures for logics:
Hereditary Harrop Logic
Higher-Order Horn Logic
Linear Logic Programs
Classical 0-order
Classical 1-order
Intuitionistic
Inductive
Modal & description
Probability
Evaluate the NNs comparing with other networks
(Kahn networks, Thomas' networks, cortex networks, kernel methods,..)

Timeliness

- Computability Characterisation of Neural Networks;
- A rich body of material accumulated in Connectionism but research vacuum in the field;
- Wide range of NN simulators; numerous research centres working on learning techniques.

Novelty

The project is novel in the aspects of

- **Theory** (Finite representation of Logic Deduction in Neural Networks);
- **Methodology** (SLD resolution rather than semantic operators; finite construction; the use of learning in deduction);
- **Practice** (Evaluation and implementation of neural networks in computational logic).

Potential Beneficiaries

1. Researchers in neuro-symbolic integration, AI;
2. The Computational Logic and Automated Reasoning communities;
3. Individuals and organisations using automated theorem provers;
4. Researchers in different areas of computer science investigating and applying learning techniques.

Potential Beneficiaries

1. Researchers in neuro-symbolic integration, AI;
2. The Computational Logic and Automated Reasoning communities;
3. Individuals and organisations using automated theorem provers;
4. Researchers in different areas of computer science investigating and applying learning techniques.

Items 2 - 3 $\Longrightarrow$ St Andrews.

St Andrews: Available Expertise

- Computational Logic Group (Roy Dyckhoff): development of proof assistants and theorem provers.
- Search algorithms; experimental methods in CS; constraint satisfaction problems (Ian Gent, Kevin Hammond, Ian Miguel, Tom Kelsey).
- Neural Network Research institutes in Edinburgh (Institute for Adaptive and Neural Computation; the Institute for Perception, Action and Behavior; the Neuroinformatics Doctoral Training Centre).

Thank you!