

# SHERLOCK — Neural Network Software for Automated Problem Solving

## 1 Introduction

### a) Intended Readership

People who want a logic programming application to perform deduction tasks and those who want to study neural-symbolic systems generate neural networks automatically.

### b) Purpose

The application is designed to provide an interface to use neural-symbolic systems to do deduction. It also provide a function to generate a Tp- or CILP- neural network for a general logic program automatically, which gives an shortcut to make a knowledge refining system.

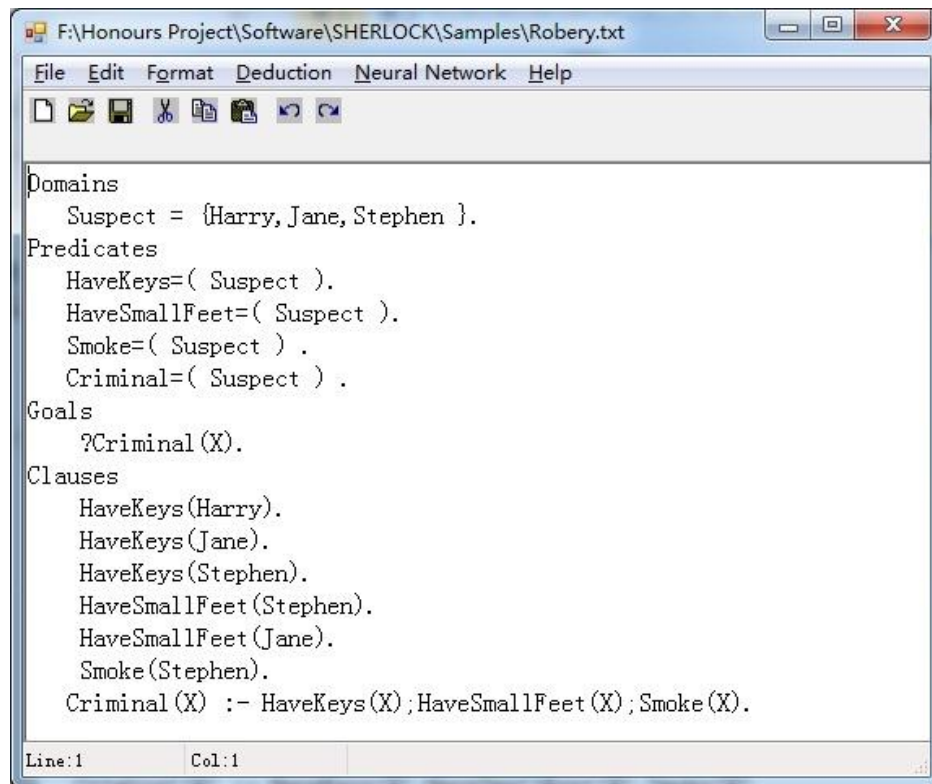
---

## 2 Overview

### a) Functional Description

The application has a logic programming editor, a function – *Neural Network* which translates a logic program to a Tp- or CILP- neural network and a function *Deduction* which *not only translates a logic* program to a Tp- or CILP- neural network, but also perform massively parallel computing and interpret the result to symbolic knowledge.

#### i. *Logic Programming Editor*



```

F:\Honours Project\Software\SHERLOCK\Samples\Robbery.txt
File Edit Format Deduction Neural Network Help
[Icons]

Domains
    Suspect = {Harry, Jane, Stephen }.
Predicates
    HaveKeys=( Suspect ).
    HaveSmallFeet=( Suspect ).
    Smoke=( Suspect ) .
    Criminal=( Suspect ) .
Goals
    ?Criminal(X).
Clauses
    HaveKeys(Harry).
    HaveKeys(Jane).
    HaveKeys(Stephen).
    HaveSmallFeet(Stephen).
    HaveSmallFeet(Jane).
    Smoke(Stephen).
    Criminal(X) :- HaveKeys(X);HaveSmallFeet(X);Smoke(X).

Line:1 Col:1

```

The syntax in a regular form is as follows:

- Key words: [a-zA-Z] [a-zA-Z]\*
- Domains: [a-zA-Z] [a-zA-Z]\*={ [a-zA-Z] [a-zA-Z]\* [, [a-zA-Z] [a-zA-Z]\* ] }.
- Predicates: [a-zA-Z] [a-zA-Z]\*=( [a-zA-Z] [a-zA-Z]\* [, [a-zA-Z] [a-zA-Z]\* ] ).
- Goals: ?[a-zA-Z] [a-zA-Z]\*=( ([a-zA-Z] [a-zA-Z]\* |[A-Z] ) [, [a-zA-Z] [a-zA-Z]\* |[A-Z] ) ).
- Clauses are divided into Facts and Rules
  - Rules:
    - [a-zA-Z] [a-zA-Z]\*=( ([a-zA-Z] [a-zA-Z]\* |[A-Z] ) [, [a-zA-Z] [a-zA-Z]\* |[A-Z] ) ).
  - Facts: [a-zA-Z] [a-zA-Z]\*([a-zA-Z] [a-zA-Z]\* [, [a-zA-Z] [a-zA-Z]\* ] ).

ii. Tp- neural network

```

Result
input neurons 12
hidden neurons 9
output neurons 12

input-hidden weights:
(0, 0, 0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

hidden thresholds:
(-0.5, -0.5, -0.5, -0.5, -0.5, -0.5, 2.5, 2.5, 2.5)

hidden-output weights :
(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)

output thresholds:
(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

```

iii. CILP- neural network

```

Result
input neurons 12
hidden neurons 9
output neurons 12

input-hidden weights:
(0, 0, 0, 0, 0, 0, 0, 55.5995, 0, 0
0, 0, 0, 0, 0, 0, 0, 55.5995, 0
0, 0, 0, 0, 0, 0, 0, 0, 55.5995
0, 0, 0, 0, 0, 0, 55.5995, 0, 0
0, 0, 0, 0, 0, 0, 0, 55.5995, 0
0, 0, 0, 0, 0, 0, 0, 0, 55.5995
0, 0, 0, 0, 0, 0, 55.5995, 0, 0
0, 0, 0, 0, 0, 0, 0, 55.5995, 0
0, 0, 0, 0, 0, 0, 0, 0, 55.5995
0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0)

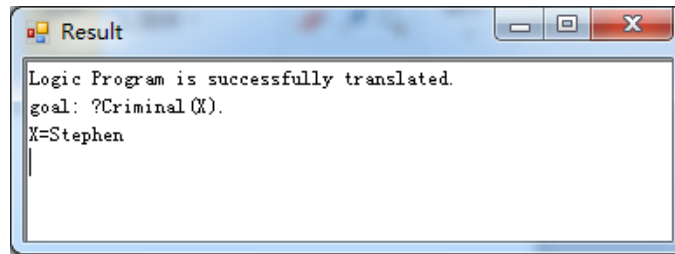
hidden thresholds:
(-41.8386, -41.8386, -41.8386, -41.8386, -41.8386, -41.8386, 83.6773, 83.6773, 83.6773)

hidden-output weights :
(55.5995, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 55.5995, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 55.5995, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 55.5995, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 55.5995, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 55.5995, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 55.5995, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 55.5995)

output thresholds:
(0, 0, 0, 41.8386, 0, 0, 41.8386, 41.8386, 0, 0, 0, 0)

```

iv. Deduction result



## b) Cautions and Warnings

- a) This program supports basic logic deduction. It does not support list.
- b) “;” means AND, which is different from Prolog.

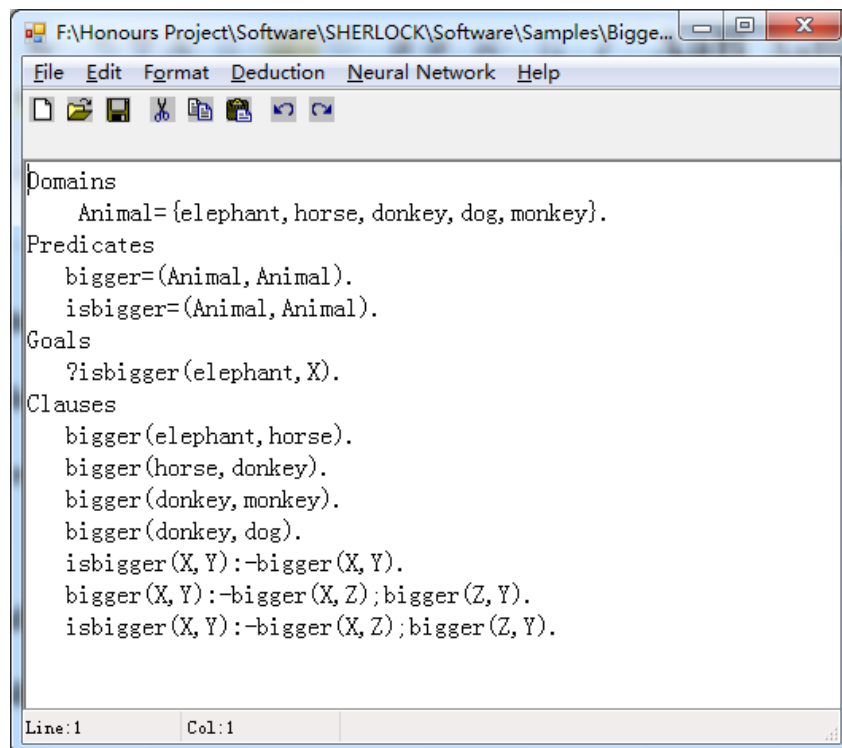
---

# 3 Instruction Sections

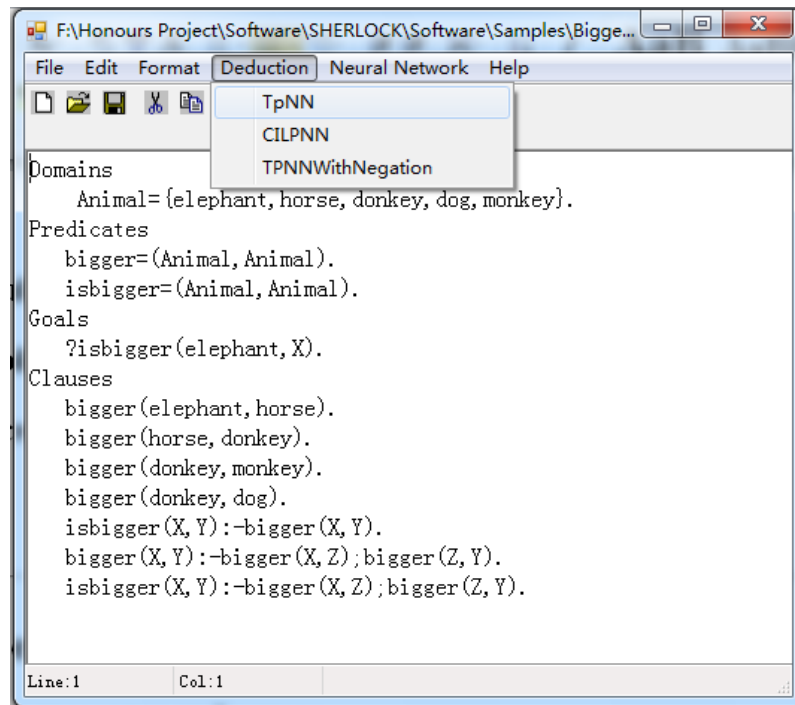
## Logic Deduction

### a) Procedures

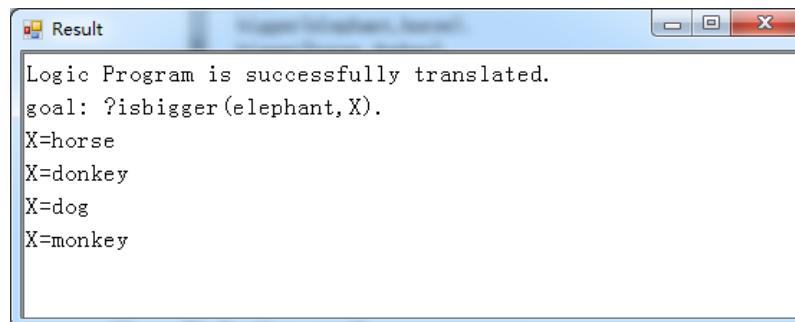
- i. Type in a logic program in the editor



- ii. Choose one type neural network to perform massively parallel deduction



iii. Deduction result



**b) Troubleshooting: Probable errors and possible causes**

**Neural-symbolic Network Generation**

**a) Procedures**

- i. Present coarse knowledge in a logic programming way, which does not contain a section – *Goals*.



- iii. Copy the CILP neural network to Matlab to build a patternnet and train the CILP network and the coarse knowledge will be refined.
-