SvL Software

Quick guide

The SvL software allows to compute the two-valued Tp operator or the three-valued SvL operator, given a logic program. Additionally to computing the above semantic operators fix-point, the software is capable of performing abduction and constructing a neural network corresponding to the logic program, although these features are still under development. In the last version of the software, and additional feature was added to obtain a neural network, corresponding to the logic program.

The software is developed in Java and delivered as a .jar file, along with some prolog routines in separate files. This makes it possible to use it under any platform: Windows, Linux or Mac.

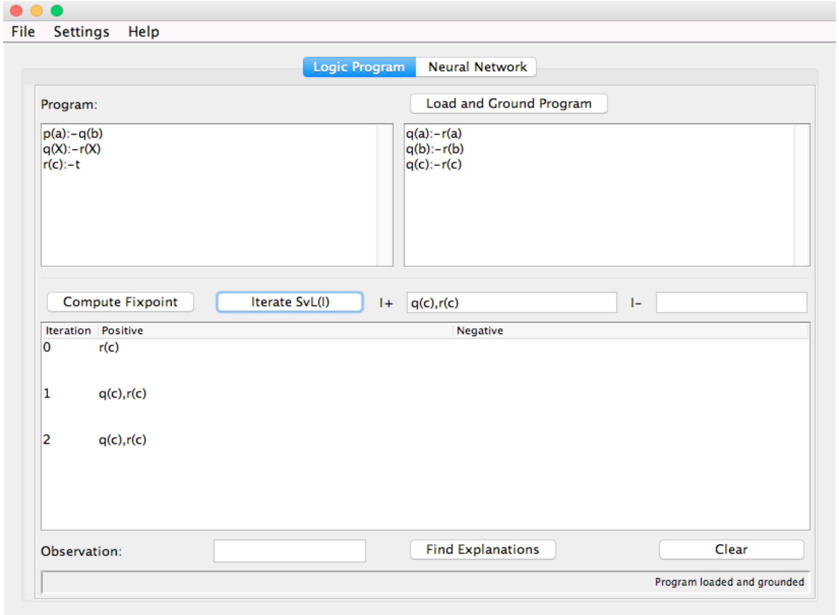
Requirements

1. Java Runtime 6+
2. Prolog

How to run it

1. Make sure all files provided are located in the same directory (SvL.jar and the *.pl files)
2. Execute SvL.jar from the command line: java -jar SvL.jar

This will prompt the application interface. As soon as the application is started, it will test if it can connect to Prolog. If it fails, settings must be changed in **Menu>Settings>Prolog Version**.



Program:

```
p(a):-q(b)
q(X):-r(X)
r(c):-t
```

```
q(a):-r(a)
q(b):-r(b)
q(c):-r(c)
```

I+ q(c),r(c)

I-

Iteration	Positive	Negative
0	r(c)	
1	q(c),r(c)	
2	q(c),r(c)	

Observation:

Program loaded and grounded

The following is a description of the main functions of the program:

1. **Load and Ground Program:** The input program is loaded into the software's engine and the clauses with variables are grounded. Grounded clauses are shown in the top-right area.

2. **Compute Fixpoint:** Depending on the selected operator (Menu>Settings>Semantic Operator) the fixpoint is computed and presented as a two or three-valued interpretation, in the lower area.
3. **Iterate SvL(I):** The program allows to provide an input interpretation for iterating the operator. Pressing the **Iterate SvL(I)** button will compute one iteration of the selected operator, and the output will be provided in both: the lower area, and as the new input for the next iteration.
4. **Find Explanations:** Allows to compute abductive explanations for the observation given in the **Observation** text field. An observation consists of a single grounded literal.
5. **Clear:** Erases the computed iterations, and the input for the initial interpretation.

Menu

The top menu has three sections:

1. **File:** Allows to save the current program, and to load previously saved ones. A program file can contain comments. These are lines starting with the character "**%**".
2. **Settings:** Allows to change the Prolog version to be used, as well as choosing the semantic operator to be computed. It also allows to show all iterations until getting to the fixpoint.
3. **Help:** Provides a link to this guide, and the credits of the software.

Syntax

The **syntax** of the logic program has the following restrictions: an example, the following is a logic program with 3 rules:

1. Each rule is of the form:

   `head:-body1,body2,body3`

   where head is an atom and body1, body2, body3 are literals.
2. Each Atom is of the form: `predicate(arg1,...,arg3)`.
3. In principle each predicate can be of any arity, although an excessive number of arguments may cause memory fails.
4. Only one rule per line.
5. Literals in the body are separated by a coma ",".
6. The program must not contain white spaces.
7. Variables always begin with a capital letter.
8. The constants **t** and **f** are reserved to denote true and false, respectively. And should not be used as predicate symbols nor arguments. They are used to represent negative and positive facts.
9. Negation is represented as a predicate of the form: `n(Literal)`.
10. Avoid using numbers as arguments.

Example

As an example, the following is a logic program with 3 rules:

- A positive fact: `p(a,b):-t`
- A negative fact: `q(a,a):-f`
- A non-ground rule: `r(X,Y):-n(q(X,Y)),p(X,X)`

  Where the non-ground clause is replaced by four ground clauses in the ground program, and whose least SvL fix-point is `I=<{p(a,b)},{q(a,a)}>` .

Abduction

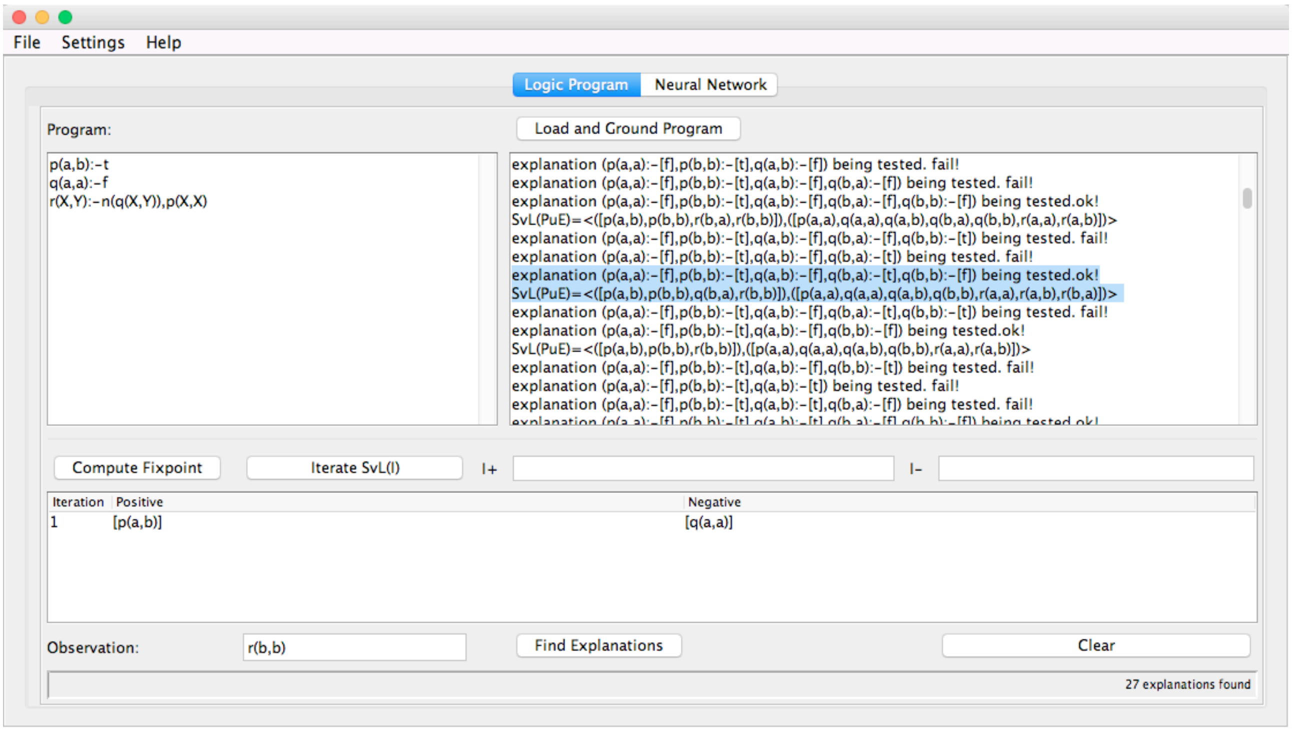
Abduction (finding explanations given an observation) is still on development and can take excessive time to compute if the number of abducibles is big, leading to a memory overflow error.

To obtain explanations for a given observation, with respect to a logic program:

1. First, obtain the SvL fix-point as described above.
2. Provide the observation on the corresponding text box, in the same format as

   described above. (As a ground literal).
3. Use the (Find explanations) button to look for explanations for the observation. All candidate explanations are shown, and when a valid one is found, it is highlighted and the corresponding least fix-point is given as evidence.

Getting back to the example, if we would have that our observation is `r(b,b)`, and we want to have explanations for it, the result would be:



Program:

```
p(a,b):-t
q(a,a):-f
r(X,Y):-n(q(X,Y)),p(X,X)
```

```
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[f]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[f],q(b,b):-[f]) being tested.ok!
SvL(PuE)=<([p(a,b),p(b,b),r(b,a),r(b,b)]),([p(a,a),q(a,a),q(a,b),q(b,a),q(b,b),r(a,a),r(a,b)])>
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[f],q(b,b):-[t]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[t]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[t],q(b,b):-[f]) being tested.ok!
SvL(PuE)=<([p(a,b),p(b,b),q(b,a),r(b,b)]),([p(a,a),q(a,a),q(a,b),q(b,b),r(a,a),r(a,b),r(b,a)])>
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[t],q(b,b):-[t]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,b):-[f]) being tested.ok!
SvL(PuE)=<([p(a,b),p(b,b),r(b,b)]),([p(a,a),q(a,a),q(a,b),q(b,b),r(a,a),r(a,b)])>
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,b):-[t]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[t]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[t],q(b,a):-[f]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[t],q(b,a):-[f],q(b,b):-[f]) being tested.ok!
```

Iteration	Positive	Negative
1	[p(a,b)]	[q(a,a)]

Observation: r(b,b)

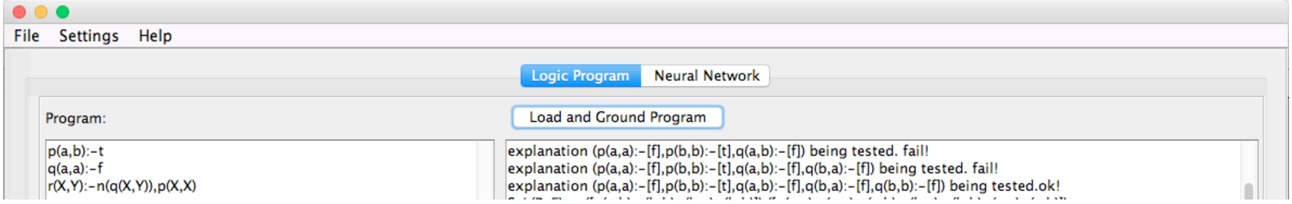
27 explanations found

Meaning there exists 27 explanations for r(b,b), i.e. all of those that contain p(b,b) and n(q(b,b)). The program, in the case an explanation is found. It also provides the least fixed point of the operator.

Neural Network Representation

An algorithm (Core Method) is used for translating a Logic Program into a Neural Network. This feature is only developed for three-valued logic and the SvL operator.

On the middle-top of the program interface, there are two tabs: One for making computation on the **logic program** and another to obtain its **neural network** representation.



```
p(a,b):-t
q(a,a):-f
r(X,Y):-n(q(X,Y)),p(X,X)
```

```
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[f]) being tested. fail!
explanation (p(a,a):-[f],p(b,b):-[t],q(a,b):-[f],q(b,a):-[f],q(b,b):-[f]) being tested.ok!
```

Once in this section, two options are given:

1. **Create Neural Network** : Takes the logic program specified in the Logic Program section, and obtains the corresponding neural network using the Core Method. In the grill where the network is shown each column has the followin meaning:
    - The first column represents the units of the network. There are three types. Each atom has two copies, one for the input layer labeled with *atom*_i , and another copy for the output layer labeled with *atom*_o. For each rule, the hidden unit is labelled with body_*headOfTheRule* .
    - Columns 2 to 4 represent the input, potential, threshold and output of each unit.
    - The fifth column "Connected to" shows all incoming connections to the current unit.
    - The sixth column shows the state of the unit. In this case we use *true* to denote the unit is **active**, and *false* to denote it is **inactive**.
2. **Update :** Update all units and propagates the truth values through the network. This allows to compute the SvL operator of the corresponding program, as well as its least fixed point.

Installing Prolog

The following links provide information on how to install prolog in the different platforms:

1. Linux: http://www.swi-prolog.org/build/Debian.html
2. MAC: http://www.swi-prolog.org/build/macos.html
3. Wndows: http://www.swi-prolog.org/build/windows.html

Luis Palacios Medinacelli

palacios.medinacelli@gmail.com