Overview of the tool - Graal Stratified Negation

Boixel Arthur

July 28, 2017

Abstract

This document presents the different functionalities of the Java tool and how to use them in the Graphical User Interface. This is a minimal description of the Graphical User Interface, only the most important functionalities are described.

Contents

1 Launch the GUI **3**

2 Ontology **3**
2.1 Load 3
2.2 Visualize 3
2.3 Save 3

3 Graph of Rule Dependencies **5**
3.1 Visualize 5
3.1.1 Text 5
3.1.2 Graphical 5
3.2 Save 6

4 Graph of Strongly Connected Components **7**
4.1 Visualize 7
4.1.1 Text 7
4.1.2 Graphical 7
4.2 Save 9

5 Forward Chaining **10**
5.1 Execute 10
5.1.1 From a file 10
5.2 Save 10

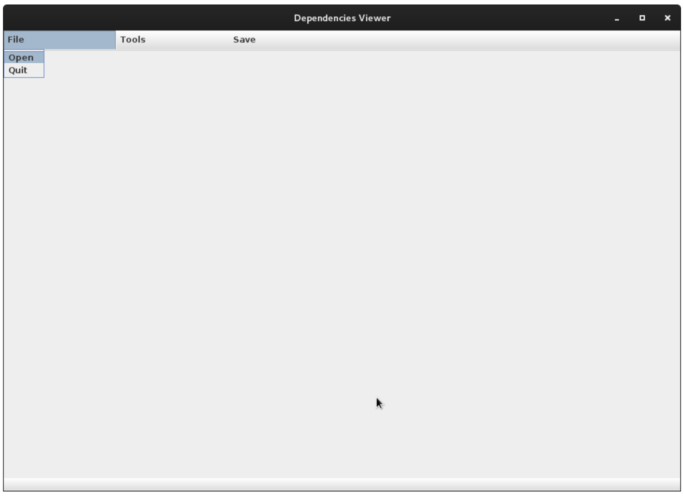


Figure 1: Open an ontology

1 Launch the GUI

The GUI will start if you launch the tool with the `-w` or `--window` option. No other options are needed.

```
java -jar graal-stratified-negation.jar -w
```

2 Ontology

The programm works with an ontology.

2.1 Load

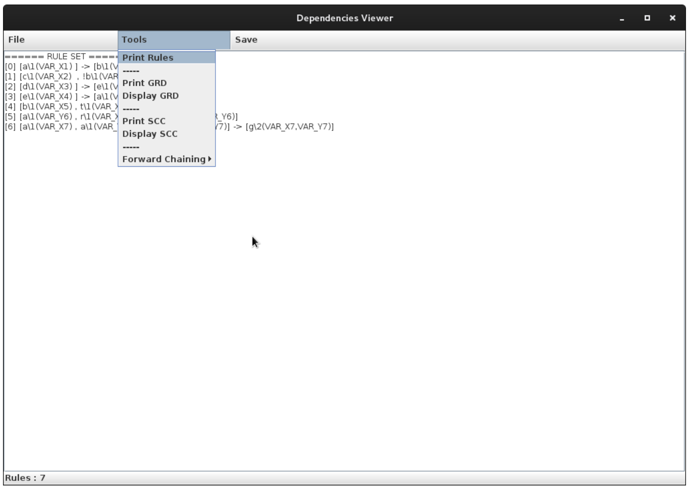
To load an ontology (in `DLGP` format) from a file, click `Open` on the top left `File` menu of the GUI as shown in Figure 2.1 and select the file that contains your rules.

2.2 Visualize

Once you opened an ontology you can visualize the rules by clicking on `Print Rules` in the `Tool` menu as shown in Figure 2.2.

2.3 Save

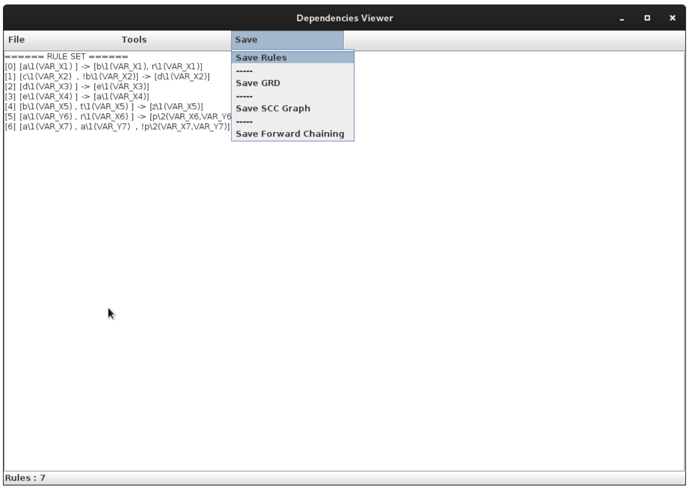
You can save your labeled ontology into a file by clicking on `Save Rules` in the `Save` menu as shown in Figure 2.3.



```
====== RULE SET =====
[0] [a\1(VAR_X1) ] -> [b\1(V
[1] [c\1(VAR_X2) , !b\1(VAR
[2] [d\1(VAR_X3) ] -> [e\1(V
[3] [e\1(VAR_X4) ] -> [a\1(V
[4] [b\1(VAR_X5) , t\1(VAR_X
[5] [a\1(VAR_Y6) , r\1(VAR_X                R_Y6)]
[6] [a\1(VAR_X7) , a\1(VAR_                 Y7)] -> [g\2(VAR_X7,VAR_Y7)]
```

Rules : 7

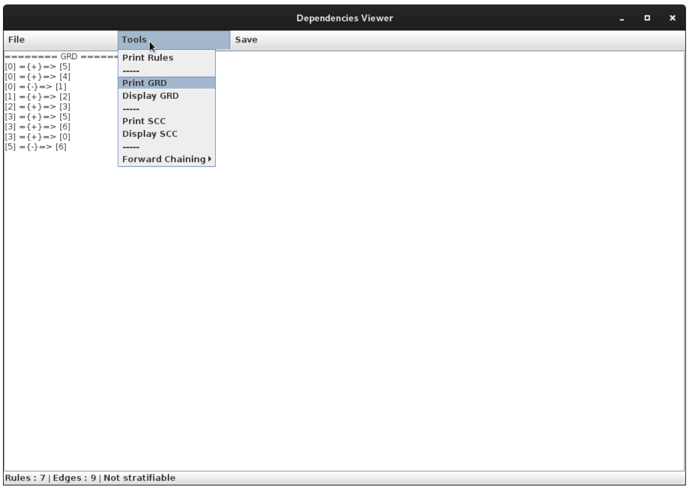
Figure 2: Print rules set



```
====== RULE SET ======
[0] [a\1(VAR_X1) ] -> [b\1(VAR_X1), r\1(VAR_X1)]
[1] [c\1(VAR_X2) , !b\1(VAR_X2)] -> [d\1(VAR_X2)]
[2] [d\1(VAR_X3) ] -> [e\1(VAR_X3)]
[3] [e\1(VAR_X4) ] -> [a\1(VAR_X4)]
[4] [b\1(VAR_X5) , t\1(VAR_X5) ] -> [z\1(VAR_X5)]
[5] [a\1(VAR_Y6) , r\1(VAR_X6) ] -> [p\2(VAR_X6,VAR_Y6
[6] [a\1(VAR_X7) , a\1(VAR_Y7) , !p\2(VAR_X7,VAR_Y7)]
```

Rules : 7

Figure 3: Save rules set



```
========= GRD ======
[0] ={+}=> [5]
[0] ={+}=> [4]
[0] ={-}=> [1]
[1] ={+}=> [2]
[2] ={+}=> [3]
[3] ={+}=> [5]
[3] ={+}=> [6]
[3] ={+}=> [0]
[5] ={-}=> [6]
```

Rules : 7 | Edges : 9 | Not stratifiable

Figure 4: Print GRD

3 Graph of Rule Dependencies

Once you opened an ontology the tool will compute its graph of rules dependencies. Please note that this may take some time with big ontologies.

3.1 Visualize

There are two ways to visualize the Graph of Rules Dependencies.

3.1.1 Text

By clicking on `Print GRD` in the `Tool` menu you will see the text version of the graph as shown in Figure 3.1.1. The text version can be used to do some post treatment. In the bottom of the GUI you can see different information about the graph :

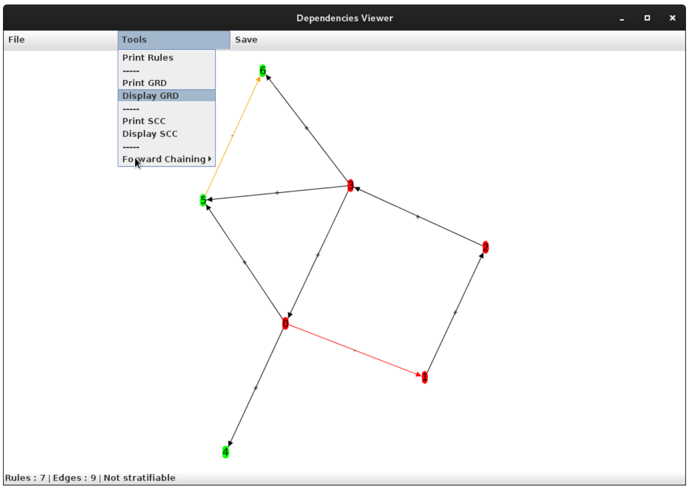
- Number of rules (nodes)
- Number of dependencies (edges)
- Wether the ontology can be stratified or not

3.1.2 Graphical

By clicking on `Display GRD` in the `Tool` menu as shown in Figure 3.1.2, you can see a graphical and dynamic representation of the graph.

Nodes can be green or red. A red node belongs to a circuit which contains a negative reliance. That means that the set of rules is not stratifiable. If there are only green nodes, the set of rules is stratifiable.

Egdes can be black, orange or red. A red edge represents a negative reliance which is part of a circuit : the rule set is not stratifiable. An orange edge



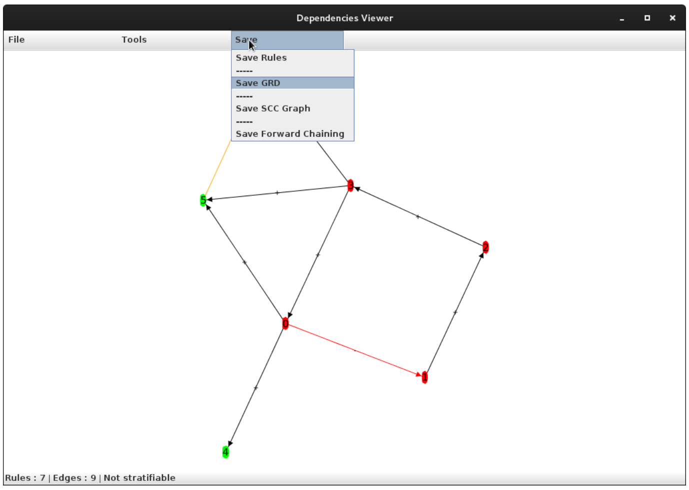
Rules : 7 | Edges : 9 | Not stratifiable

Figure 5: Display GRD

represents a negative reliance which does not cause any problem and a black edge represents a positive reliance between two rules.

3.2 Save

To make some post treatment of the Graph of Rule Dependencies you can save it to a file by clicking on `Save GRD` in the `Save` menu as shown in Figure 3.2. The graph will be saved as you can see it when you print it.



Rules : 7 | Edges : 9 | Not stratifiable

Figure 6: Save GRD

4 Graph of Strongly Connected Components

When computing the Graph of Rule Dependencies the tool will automatically compute its Graph of Strongly Connected ComponentS. This graph helps you detect where are the bad circuits in the Graph of Rule Dependencies.

4.1 Visualize

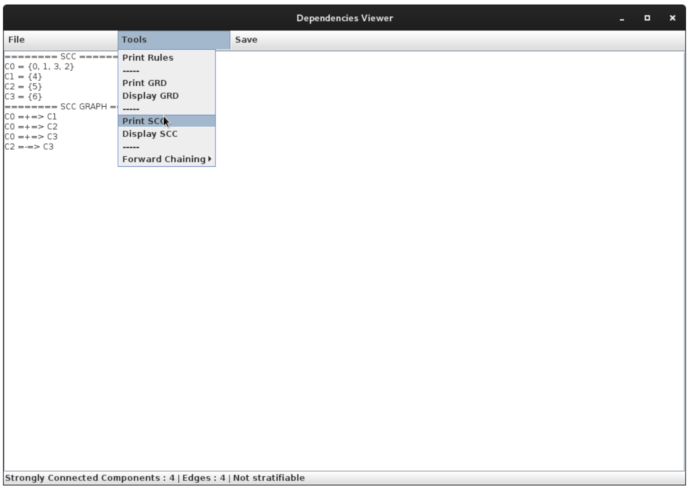
There are two ways to visualize it.

4.1.1 Text

By clicking on `Print SCC` in the `Tool` menu you will see the text version of the graph as shown in Figure 4.1.1. The text version can be used to do some post treatment.

4.1.2 Graphical

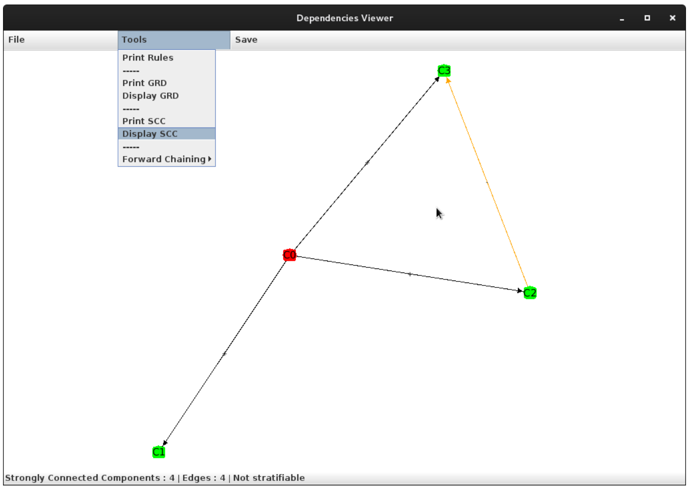
By clicking on `Display SCC` in the `Tool` menu as shown in Figure 4.1.2, you can see a graphical and dynamic representation of the graph.
Nodes can be green or red. A red node is a strongly connected components which contains a circuit with a negative reliance. That means that the set of rules is not stratifiable. By clicking on a node, a new window will display the Graph of Rule Dependencies of the associated strongly connected component. If there are only green nodes, the set of rules is stratifiable.
Egdes can be black or orange. An orange edge represents a negative reliance and a black edge represents a positive reliance between to set of rules.



```
======== SCC ======
C0 = {0, 1, 3, 2}
C1 = {4}
C2 = {5}
C3 = {6}
======== SCC GRAPH ==
C0 =+=> C1
C0 =+=> C2
C0 =+=> C3
C2 =-=> C3
```

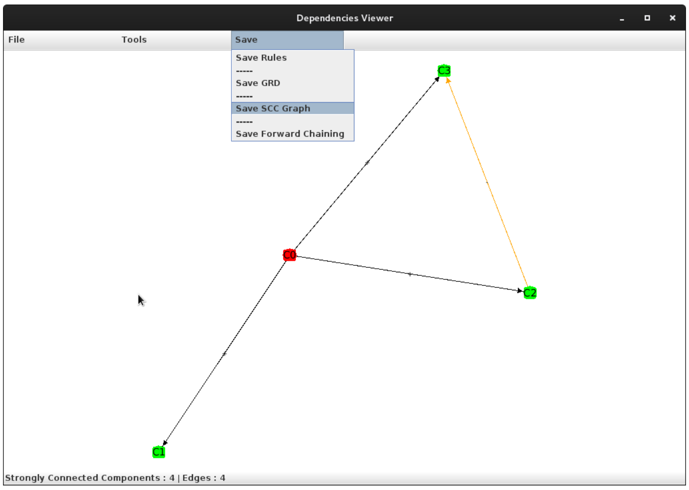
Strongly Connected Components : 4 | Edges : 4 | Not stratifiable

Figure 7: Print SCC



Strongly Connected Components : 4 | Edges : 4 | Not stratifiable

Figure 8: Display SCC

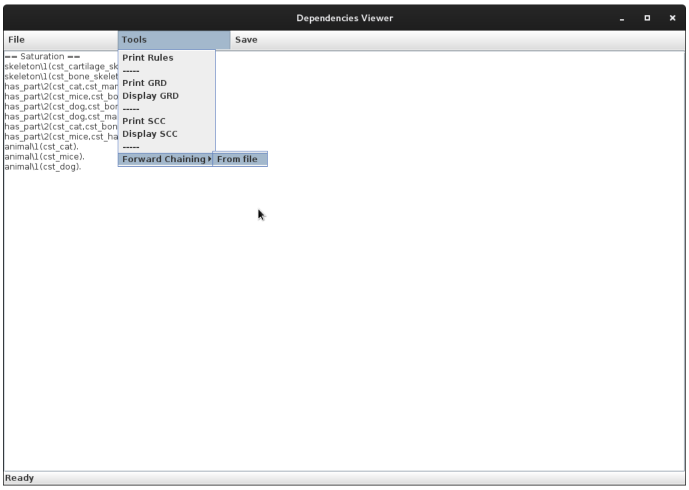


Strongly Connected Components : 4 | Edges : 4

Figure 9: Save SCC Graph

4.2 Save

To make some post treatment on the Graph of Strongly connected Components you can save it to a file by clicking on `Save SCC Graph` in the `Save` menu as shown in Figure 4.2. The graph will be saved as you can see it when you print it.



```
== Saturation ==
skeleton\1(cst_cartilage_sk
skeleton\1(cst_bone_skelet
has_part\2(cst_cat,cst_mar
has_part\2(cst_mice,cst_bo
has_part\2(cst_dog,cst_bor
has_part\2(cst_dog,cst_ma
has_part\2(cst_cat,cst_bon
has_part\2(cst_mice,cst_ha
animal\1(cst_cat).
animal\1(cst_mice).
animal\1(cst_dog).
```

Figure 10: Forward Chaining Application

5 Forward Chaining

If your set of rules is stratifiable you can apply the forward chaining mechanism on a set of facts (the forward chaining will use the computed stratification).

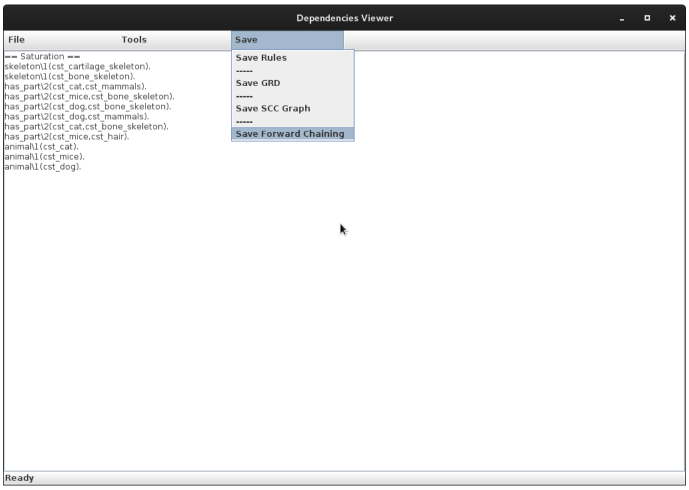
5.1 Execute

5.1.1 From a file

By clicking on `From file` in the `Forward Chaining` submenu of the `Tool` menu, you can select a file which contain your set of facts. The tool will apply the forward chaining mechanism and print the result as shown for example in Figure 5.1.1.

5.2 Save

By clicking on `Save Forward Chaining` in the `Save` menu as shown in Figure 5.2 you can save a saturated fact base. You can select a file in which the tool will save the result of the forward chaining execution on a specified file of facts.



```
== Saturation ==
skeleton\1(cst_cartilage_skeleton).
skeleton\1(cst_bone_skeleton).
has_part\2(cst_cat,cst_mammals).
has_part\2(cst_mice,cst_bone_skeleton).
has_part\2(cst_dog,cst_bone_skeleton).
has_part\2(cst_dog,cst_mammals).
has_part\2(cst_cat,cst_bone_skeleton).
has_part\2(cst_mice,cst_hair).
animal\1(cst_cat).
animal\1(cst_mice).
animal\1(cst_dog).
```

Figure 11: Save Forward Chaining