Towards Generating Explanations for ASP–Based Link Analysis using Declarative Program Transformations

Martin Atzmüller¹, Cicek Güven¹, Dietmar Seipel²

¹ Tilburg University, Cognitive Science and AI, Netherlands
² University of Würzburg, Department of CS, Germany

Cottbus, September 12, 2019
<table>
<thead>
<tr>
<th></th>
<th>Artificial Intelligence</th>
<th>Declarativity and ASP</th>
<th>Link Analysis</th>
<th>Rule Instrumentation</th>
<th>Explanations</th>
</tr>
</thead>
</table>
| 1 | Artificial Intelligence
| 2 | Declarativity and ASP
| 3 | Link Analysis
| 4 | Rule Instrumentation
| 5 | Explanations
Artificial Intelligence (AI)

The history of mankind brought many dramatic changes, e.g. the agricultural, the industrial, and the technological revolution.

According to Prof. Yuval Noah Harari from Isreal, the 21st century is characterized by bio–technology and artificial intelligence.

The smart city of the future is facing a growing wave of digitization. CS is used in many areas of everyday life, including, e.g., business, industry, internet, health, and entertainment (Smart Phone / Home).
Symbolic and Subsymbolic AI

*Symbolic or knowledge / rule–based* AI models central cognitive abilities of humans like *logic*, deduction and planning in computers. Mathematically exact operations can be defined. Decision making can be supported.

*Subsymbolic* or statistical AI tries to learn a model of a process (e.g., an action of a robot or the classification of sensor data), from training data.

Recently, there has been great progress in the fields of Deep / Machine Learning and Neural Networks.
Current Status of AI

Since the formulation of the Turing Test in 1950, there have been many success stories of AI:

- Computers can compete in games (Chess 1997, Go 2015).
- In health care, computers can support sick, handicapped or old-aged people.
- Hotlines of call centers are supported by computers based on NLP, image recognition, and intelligent decision support.
- Algorithms from deep learning are increasingly used by banks and insurance companies for decision making.
- Companies are developing autonomously driving cars (cf. Google) and devising autonomous space missions.
Future of AI

- AI reports about popular developments can be found in many newspapers: https://www.bbc.com/news/business-48994128

- AI is considered to be a *key technology* for society. As for other new technologies, the questions arises, how AI will influence our future and how we can guide its development.

- Key questions about the *chances and risks* have to be answered in the fields of, e.g., ethics, law, biology, and computer science.

- Robots with intelligent behaviour might become possible in the future. The Turing Test could be passed before 2040: https://www.youtube.com/watch?v=WT5K3aUagm4

- Until 2090, super-intelligent, self-optimizing beeings might become possible.
The first Android that could pass the Turing Test:
https://www.youtube.com/watch?v=WT5K3aUagm4
Symbolic AI and Declarativity

Techniques from *symbolic AI* can help to solve problems of the digitization wave. Modern *intelligent* information systems need to integrate (incl. inference) hybrid *knowledge bases* with *declarative* expert knowledge.

*Declarative programming (DP)* is another advanced paradigm from symbolic AI for modeling and solving complex problems.

- Traditional, *imperative* programming languages tell the computer exactly how to accomplish a goal.
- Modern DP languages only specify the goal to the computer, e.g. Rule, ASP, Database and Ontology Languages.

The toolkit Declare offers a mediator technology for intelligent knowledge based systems, data mining tools, non–monotonic reasoning (NMR, ASP), etc. using PROLOG and domain–specific languages (DSL).
Declarative Knowledge / Rule Bases

Declarativity: forward reasoning, treatment of negation and aggregation with ASP–tools such as Clingo or dlv.

Deduction with complex rules is possible (generic, heuristic, defeasible, with disjunction and default negation, . . .).

Logical rules of the form \( A \land B \rightarrow C \lor D \) can be written in DSL notation as if \( A \) and \( B \) then \( C \) or \( D \).

Applications

- Diagnosis in Industry, Medicine, etc.
- Handling of Business Rules in E–Commerce
- Root Cause Analysis in Computer Networks
- Link Analysis in Social Networks, Association Rules
Symbolic AI and Explanations

With the availability of powerful AI methods, more and more are applied to decision support. *Explicative methods*

- are interpretable, transparent, and explainable,
- make models and methods understandable and
- enable computational sensemaking.

Thus, the trust of the user applying these methods is increased.

Explication can be implemented using declarative programming, especially ASP.
Answer Set Programming (ASP)

In ASP systems, a rule can have a positive body and a test body (earlier: negative body) containing meta/test–predicates, e.g., default negation not or aggregation such as count:

$$\begin{align*}
A & :- B_1, \ldots, B_n, C_1, \ldots, C_m. \\
\text{head} & \quad \text{positive body} \quad \text{test body}
\end{align*}$$

A set of atoms $I$ is an answer set (stable model) of an ASP $\mathcal{P}$, if $I$ is a minimal model of the Gelfond/Lifschitz transformation $\mathcal{P}^I$.

$\mathcal{P}^I$ is the reduct of an instance of $\mathcal{P}$, where

- all atoms of the head and the positive body are ground and
- all atoms of the test body are evaluated w.r.t. $I$.

$\mathcal{P}^I$ contains $A\theta:- (B_1, \ldots, B_n)\theta$, if $I((C_1, \ldots, C_m)\theta) = true.$
Example: ASP in Clingo

The following ASP program is stratified, i.e., there is no recursion through test predicates:

\[
c(X, Y, Z) :- \text{edge}(X, Y), \text{edge}(X, Z), Y \neq Z.
\]
\[
similiar(Y, Z) :- \text{node}(Y), \text{node}(Z), Y \neq Z,\]
\[
    n=\#\text{count}\{X: c(X, Y, Z)\}.
\]
\[
lp(Y, Z) :- \text{similiar}(Y,Z), \neg \text{edge}(Y, Z).
\]

The positive body predicates \textit{edge}, \textit{node}, \textit{similiar}, and \textit{lp} are used in forward reasoning for computing the minimal model, while \textit{not} and \textit{count} are meta/test–predicates.

Technically, the atom \(n=\#\text{count}\{X: c(X, Y, Z)\}\) can be considered as \(\text{count}(X, c(X, Y, Z), n)\).
Link prediction and (anomalous) link discovery are prominent methods in social network analysis:

- Link prediction aims to estimate the future link structure.
- Anomalous link discovery focusses on the identification of links that deviate from a given model of normality, e.g. from expectations.

Using declarative programming approaches, we can utilize background knowledge, e.g. formalized using ASP:

- We apply ASP since it allows to specify interesting structures and patterns in an abstract, compact way.
- Link prediction approaches can be easily implemented and complemented.
We focus on link prediction and the analysis of anomalous links.

- A very simple case predicts a link between two unconnected nodes, if they share a given number of common neighbours.
- We consider an explanation method tracing the inference steps.
- This is implemented using syntactic program transformations with rule instrumentation.

Our implementation using ASP is similar to offline justification approaches, but more general, since we apply syntactic program transformations, that are not necessarily specific to ASP programs.
Use Case 1: Link Prediction by ASP

We predict links between similar nodes to close triangles:

\[
\text{const } n=2.
\]

\[
c(X, Y, Z) : \text{edge}(X, Y), \text{edge}(X, Z), Y\neq Z.
\]

\[
similar(Y, Z) : \text{node}(Y), \text{node}(Z), Y\neq Z,
\]
\[
n=\#\text{count}\{X:c(X, Y, Z)\}.
\]

\[
lp(Y, Z) : similar(Y, Z), \text{not edge}(Y, Z).
\]

\(*similiar(Y, Z)*: the nodes \(Y\) and \(Z\) have \(n\) common neighbours \(X\),
\(*lp(Y, Z)*: we predict a link from node \(Y\) to the similar node \(Z\),
\quad if there is no edge from \(Y\) to \(Z\)

Later, we will use a simplified ASP that collapses the last two rules into a single rule to eliminate similar:

\[
lp(Y, Z) : \text{node}(Y), \text{node}(Z), Y\neq Z,
\]
\[
\quad \text{not edge}(Y, Z), n=\#\text{count}\{X:c(X, Y, Z)\}.
\]
Use Case 2: Anomalous Link Discovery using ASP

Another method predicts links to similar nodes:

\[
\begin{align*}
\text{const } n=2. \\
c(X, Y, Z) &\ :- \ edge(X, Y), edge(X, Z), Y\neq Z. \\
similar(Y, Z) &\ :- \ node(Y), node(Z), Y\neq Z, \\
&\quad n=\#\text{\{count}\{X:c(X, Y, Z)\}.} \\
lp(X, Y) &\ :- \ edge(X,Z), similar(Y,Z), \neg edge(X,Y)
\end{align*}
\]

\(lp(X, Y)\): if there is an edge from \(X\) to \(Z\) but no edge from \(X\) to the similar node \(Y\), then we predict a link from \(X\) to \(Y\)
Anomalous Link Discovery using ASP

For two graphs, we obtain two methods $lp_1(X, Y)$ and $lp_2(X, Y)$ for predicting links.

The links are considered anomalous, if they are not predicted by both methods in common.

\[
\begin{align*}
\text{common}(X, Y) & \leftarrow lp_1(X, Y), \; lp_2(X, Y). \\
\text{anomalous\_12}(X, Y) & \leftarrow lp_1(X, Y), \; \text{not common}(X, Y). \\
\text{anomalous\_21}(X, Y) & \leftarrow lp_2(X, Y), \; \text{not common}(X, Y).
\end{align*}
\]

The program is also stratified. But general, non–stratified ASPs are useful for more advanced applications.
Transformation using Rule Instrumentation

Program transformation for Clingo rules $r_j$:

\[
\begin{array}{c}
\text{head} \\
A \leftarrow B_1, \ldots, B_n, C_1, \ldots, C_m.
\end{array}
\]

The test body atoms include all atoms which occur under meta–predicates such as default negation not or count.

- $j$ is the number of the rule $r_j$ within the Clingo program,
- A Clingo rule can contain variable symbols under certain conditions. Let $X_1, \ldots, X_k$ be the variable symbols occuring in $A \leftarrow B_1, \ldots, B_n$.

A further rule is constructed for recording that the rule has fired:

\[
\text{rule\_fired}(j, X_1, \ldots, X_k) \leftarrow A, B_1, \ldots, B_n, C_1, \ldots, C_m.
\]

We assume that the predicate rule\_fired does not occur in $\mathcal{P}$. 
Example: Transformation

The Clingo rule $r_3$ of the first, simplified use case for predicting links

\[ lp(Y, Z) :- \]
\[ \text{node}(Y), \text{node}(Z), Y \neq Z, \]
\[ \text{not edge}(Y, Z), n = \#\text{count}\{X : c(X, Y, Z)\}. \]

is transformed to

\[ \text{rule}_\text{fired}(3, Y, Z) :- \]
\[ lp(Y, Z), \text{node}(Y), \text{node}(Z), Y \neq Z, \]
\[ \text{not edge}(Y, Z), n = \#\text{count}\{X : c(X, Y, Z)\}. \]

The variable $X$ does not occur in the head or the positive body of rule $r_3$ and thus not in the head of the transformed rule.
If $\mathcal{P}$ was the original Clingo program, and $\mathcal{P}_{fired}$ consists of the transformed Clingo rules, then we later evaluate the extended Clingo program $\mathcal{P}_{ext} = \mathcal{P} \cup \mathcal{P}_{fired}$.

**Theorem**: Every answer set $I_{ext}$ of $\mathcal{P}_{ext}$ corresponds to an answer set $I$ of $\mathcal{P}$ extended by atoms for rule_fired.

$I_{ext}$ indicates the ground instances of the rules that have fired.

In addition to offline justifications, we can handle non–ground rules. We can present textual explanations and select suitable subsets for graphical visualization.
Graphical Explanations

We can explain the computed answer set $I$ by a tree structure. E.g., for the simplified Example 1 we could get the following tree ($cn\_lp$ should be $lp$):

All body atoms are relevant; the positive body atoms lead to subtrees, whereas the test body atoms have only been tested in $I$. 
Textual Explanations

For every derived atom

\[ rule\_fired(j, x_1, \ldots, x_k) \in P_{ext}, \]

we know that the original Clingo rule \( r_j \) fired with the substitution

\[ \theta = \{ X_i \mapsto x_i \mid 1 \leq i \leq k \}. \]

Then \( A\theta \) is in the answer set \( I \) of \( P \) with the textual explanation

\[ (A\theta) - is\_supported\_by - r_j\theta \]

E.g., \( rule\_fired(3, 1, 3) \) leads to

\[ lp(1, 3) - is\_supported\_by - \]
\[ \quad ( lp(1, 3) :- node(1), node(3), \]
\[ \quad \quad not\_edge(1, 3), 1!=3, n=\#\text{count}\{X:c(X, 1, 3)\} ). \]
Workflow of Rule Instrumentation

- Clingo Program
- Declare Program
- Extended Declare Program
- Extended Clingo Program
- Extended Answer Set
- Answer Set
- Explanation
- Presentation
- Transformation
- Direct Evaluation
- Evaluation
- Filtering
- Extraction
Final Remarks

- We have exemplified the application and efficacy of the proposed explanation approach with rule instrumentation in the context of link analysis for social networks, i.e. link prediction and anomalous link discovery.

- Our experiments indicate that the method performs well for obtaining explanations and according presentations, which can also be incrementally refined.

- For future work, we aim to extend the approach towards knowledge–based automatic refinement methods, also taking into account more complex and richer data representations, e.g. in complex interaction networks.
Thanks for your attention!

Do you have questions?