# Generating Explanations for Complex Biomedical Queries

**Umut Oztok** and **Esra Erdem**
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey

**Abstract**

We present a computational method to generate explanations to answers of complex queries over biomedical ontologies and databases, using the high-level representation and efficient automated reasoners of Answer Set Programming. We show the applicability of our approach with some queries related to drug discovery over PHARMGKB, DRUGBANK, BIOGRID, CTD and SIDER.

## Introduction

Representing biomedical knowledge in structured forms, like ontologies, in different formal languages, constructing them independently from each other, and storing them at different locations have brought about many challenges for answering queries over these ontologies. One challenge is, for the experts, to be able to represent a complex query in a natural language, and get its answers in an understandable form. Another challenge is to be able to answer complex queries that require appropriate integration of relevant knowledge stored in different places and in various forms. In addition, explaining why the complex query has such an answer is challenging.

The first two challenges are addressed in (Erdem and Yeniterzi 2009) and (Bodenreider et al. 2008), by developing a controlled natural language, called BIOQUERYCNL, to represent biomedical queries, and an algorithm to convert queries in BIOQUERYCNL to a program in Answer Set Programming (ASP) (Lifschitz 2008)—a declarative programming paradigm with a high-level representation language and efficient solvers. The idea is to compute answers to complex biomedical queries automatically using ASP solvers, considering relevant parts of ontologies and the "rule layer" that integrates these ontologies. In this paper, we address the third challenge, generating explanations for complex queries, also using computational methods of ASP.

## Explanation Generation

A *normal (ASP) program* is a finite set of *rules* of the form

$$A_0 \leftarrow A_1, \ldots, A_k, \text{not } A_{k+1}, \ldots, \text{not } A_m$$

where $m \geq k \geq 0$ and each $A_i$ is an atom. For a rule $r$, we denote the head of a rule $r$ by $H(r)$. Also, the set $\{A_1, ..., A_k\}$ of atoms is denoted by $B^+(r)$, and the set $\{A_{k+1}, ..., A_m\}$ of atoms is denoted by $B^-(r)$.

In following, let $\Pi$ be a normal ASP program, $X$ be an answer set for $\Pi$ that contains an atom $x$. The goal is to understand why $x$ is in $X$.

**Explanations:** Let us first define the positive dependency graph of a program: The *positive dependency graph* of a normal ASP program $\Pi$ is a directed graph, whose vertices denote the atoms in $\Pi$, and edges $\langle x, y\rangle$ denote the existence of a rule $r$ in $\Pi$ where $x \in H(r)$ and $y \in B^+(r)$.

An *explanation* for $x$ with respect to $\Pi$ and $X$ is a finite sequence $\langle R_1, \ldots, R_n\rangle$ of rules in $\Pi$ such that the following hold:

- $H(R_i) \cup B^+(R_i) \subseteq X$ and $B^-(R_i) \cap X = \emptyset$ for $1 \leq i \leq n$;
- $H(R_i) \neq H(R_j)$ for $1 \leq i < j \leq n$;
- $H(R_n) = x$;
- for every $R_i$, for every atom $a \in B^+(R_i)$, there exists a rule $R_k$ ($i \neq k$) such that $H(R_k) = a$;
- for every $R_i$ ($1 \leq i < n$), $H(R_i) \in B^+(R_j)$ for some $j \neq i$;
- The positive dependency graph of the ASP program $\{R_1, \ldots, R_n\}$ is acyclic.

Informally, an explanation for an atom is a sequence of rules which are the reasons of that atom being in an answer set. There may be different explanations for an atom. Among them, we are interested in shortest explanations, with the minimum number of rules.

**Finding shortest explanations:** To compute shortest explanations, we use "explanation trees " whose vertices correspond to atoms/rules and edges describe dependencies between atoms/rules. An explanation tree whose root corresponds to an atom x represents all possible explanations for x; the idea is then to extract a shortest explanation among all explanations.

Let us now summarize our algorithm to find a shortest explanation for an atom $x$ with respect to $\Pi$ and $X$.

First, it generates an explanation tree $T$. For every vertex $v$ in $T$ that denotes an atom $x$, it finds every rule $R$ in $\Pi$ that support $x$ (i.e., every rule $R \in \Pi$ such that $H(R) = x$, $B^+(R) \subseteq X$, and $B^- \cap X = \emptyset$) and assigns these rules as the children of $v$ in $T$. For every vertex $v$ in $T$ that denotes a rule $R$, it assigns the atoms in $B^+(R)$ as the children of $v$.

Next, our algorithm assigns weights to the vertices of $T$ so that we can find a shortest explanation. Since the length of an explanation depends on the number of rules it contains, we need to assign weights to the vertices of the tree in such a way that the weight of a vertex $v \in X$ in the tree corresponds to the number of rules in a shortest explanation for $v$ and the weight of a vertex $u \in \Pi$ in the tree corresponds to the sum of the weights of its children. As child of a vertex $u \in \Pi$ is a vertex $u' \in X$ and the weight of $u'$ is the number of rules in a shortest explanation of $u'$, the weight of $u$ becomes the least number of rules needed to explain atoms in its body. So, starting at a vertex $v \in X$ and traversing a path by moving towards the vertex which has the smallest weight until every branch reaches a leaf, we can find a shortest explanation of $v$ (vertices $v' \in \Pi$ on the path). Thus, the weight $W(v)$ of a vertex $v$ can be defined as follows:

- If $v \in \Pi$, $W(v) = \sum_{i=1}^{i=n} W(c_i) + 1$;
- If $v \in X$, $W(v) = min_{1 \leq i \leq n}(W(c_i))$

where $c_i$ is the $i$'th child of $v$.

Finally, our algorithm extracts a shortest explanation for $x$ with respect to $\Pi$ and $X$ from $T$. The explanation computed by our algorithm is indeed the shortest.

**Proposition 1.** *Let $\Pi$ be a normal ASP program, $X$ be an answer set for $\Pi$, and $x$ be an atom in $X$. Above algorithm finds a shortest explanation for $x$ with respect to $\Pi$ and $X$.*

The time complexity of generating a weighted explanation tree and extracting a shortest explanation from that tree, in the worst case, is $h \times b$, where $h$ is the maximum height of the tree and $b$ is the maximum branching factor of a vertex in the tree. Since the set of vertices of the explanation tree is a subset of $X \cup \Pi$, $h = |X| + |\Pi|$ and $b = max\{|X|, |\Pi|\}$. So, the time complexity of our algorithm is $O((|X| + |\Pi|) \times max\{|X|, |\Pi|\})$.

**Example:** After translating the BIOQUERYCNL query

> What are the genes that are targeted by the drug Epinephrine and that interact with the gene DLG4?

into an ASP program, and extracting the relevant parts of ontologies and the rule layer, we can find answers to this query using the ASP solver CLASP (Gebser et al. 2007). One of the answers is ADRB1. At this stage, to better understand the relationships between genes and drugs, the experts may ask for an explanation as to why ADRB1 is an answer. Our algorithm above can generate a shortest explanation:

$$
\begin{aligned}
&drug\_gene\_ctd(Epinephrine, ADRB1).\\
&drug\_gene(Epinephrine, ADRB1) \leftarrow\\
&\quad drug\_gene\_ctd(Epinephrine, ADRB1).\\
&gene\_gene\_biogrid(DLG4, ADRB1).\\
&gene\_gene(DLG4, ADRB1) \leftarrow\\
&\quad gene\_gene\_biogrid(DLG4, ADRB1).\\
&gene\_gene(ADRB1, DLG4) \leftarrow\\
&\quad gene\_gene(DLG4, ADRB1).\\
&answer(ADRB1) \leftarrow\\
&\quad drug\_gene(Epinephrine, ADRB1),\\
&\quad gene\_gene(ADRB1, DLG4).
\end{aligned}
$$

Moreover, we can translate this explanation into BIOQUERYCNL:

> the drug Epinephrine targets the gene ADRB1 according to CTD and the gene ADRB1 interacts with the gene DLG4 according to BIOGRID.

## Discussion

The most recent work related to explanation generation in ASP are (Pontelli, Son, and El-Khatib 2009) and (Brain and De Vos 2005), in the context of debugging normal ASP programs. (Brain and De Vos 2005) studies the question "why is an atom $x$ in an answer set $X$ for an ASP program $\Pi$." As an answer to this question, the authors find the rule in $\Pi$ that supports $x$ with respect to $X$; whereas we compute a shortest explanation (a sequence of rules) to answer this question. (Pontelli, Son, and El-Khatib 2009) studies also the same question, and, as an answer, finds a "justification", which is a labeled graph that provides an explanation for the truth values of atoms with respect to an answer set. There is a provable correspondence between a justification and an explanation as we defined above:

**Proposition 2.** *Let $\Pi$ be a normal ASP program, $X$ be an answer set for $\Pi$, and $x$ be an atom in $X$. An offline justification of $x$ with respect to $X$ and some $U \in$ Assumptions$(\Pi, X)$ can be translated into an explanation for $x$ with respect to $\Pi$ and $X$. An explanation for an atom $x$ with respect to $\Pi$ and $X$ can be transformed into a subgraph of an offline justification of $x$ with respect to $X$ and some $U \in$ Assumptions$(\Pi, X)$.*

The reason for not being able to transform an explanation into a complete justification is the lack of information about why an atom is not in the given answer set. Although there is such a correspondence between a justification and an explanation, (Pontelli, Son, and El-Khatib 2009) finds an explanation whereas we find a shortest explanation.

Extending our definitions and algorithms to ASP programs with aggregates, improving our algorithm with some search heuristics, and a detailed experimental evaluation are part of our ongoing work.

**Acknowledgments** This work has been supported by TUBITAK Grant 108E229.

## References

Bodenreider, O.; Coban, Z.; Doganay, M. C.; Erdem, E.; and Kosucu, H. 2008. A preliminary report on answering complex queries related to drug discovery using answer set programming. In *Proc. of ALPSWS'08*.

Brain, M., and De Vos, M. 2005. Debugging logic programs under the answer set semantics. In *Proc. of ASP'05*.

Erdem, E., and Yeniterzi, R. 2009. Transforming controlled natural language biomedical queries into answer set programs. In *Proc. of BioNLP'09*, 117–124.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. clasp: A Conflict-Driven Answer Set Solver. In *Proc. of LPNMR*, 260–265.

Lifschitz, V. 2008. What is answer set programming? In *Proc. of AAAI*.

Pontelli, E.; Son, T. C.; and El-Khatib, O. 2009. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming* 1–56.