BIOQUERY-ASP: Querying Biomedical Ontologies using Answer Set Programming

Esra Erdem, Halit Erdogan, and Umut Oztok

Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

Abstract. We describe a software system, called BIOQUERY-ASP, that finds answers and generates explanations to complex biomedical queries over the available knowledge resources, such as, PHARMGKB, DRUGBANK, CTD, SIDER, BIOGRID, using the computational methods/tools of Answer Set Programming.

1 Introduction

Recent advances in health and life sciences have led to generation of a large amount of biomedical data. To facilitate access to its desired parts, such a big mass of data has been represented in structured forms, like biomedical ontologies and databases. On the other hand, representing these ontologies and databases in different forms, constructing them independently from each other, and storing them at different locations have brought about many challenges for answering queries about the knowledge represented in these ontologies.

One of the challenges for the users is 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 from different knowledge resources and/or that require auxiliary definitions, such as, chains of drug-drug interactions, cliques of genes based on gene-gene relations, or similarity/diversity of genes/drugs. Furthermore, once an answer is found for a complex query, the experts may need further explanations about the answer.

Consider, for instance, the following queries:

Q1 What are the genes that are targeted by the drug Epinephrine and that interact with the gene DLG4?
Q2 What are the genes that are targeted by all the drugs that belong to the category Hmg-coa reductase inhibitors?
Q3 What are the genes related to the gene ADRB1 via a gene-gene relation chain of length at most 3?
Q4 What are the 3 most similar genes that are targeted by the drug Epinephrine?

Most of the existing biomedical querying systems (e.g, web services built over the available knowledge resources) support keyword search but not complex queries like Q1–Q4. Some of these complex queries, such as Q1 and Q2, can be represented in a formal query language (e.g., SQL/SPARQL) and then answered using Semantic Web technologies. However, queries, like Q3, that require auxiliary recursive definitions (such as transitive closure) cannot be directly represented in these languages; and

thus such queries cannot be answered directly using Semantic Web technologies. The experts usually compute auxiliary relations externally, for instance, by enumerating all drug-drug interaction chains or gene cliques, and then use these auxiliary relations to represent and answer a query like Q3. Similarity/diversity queries, like Q4, cannot be represented directly in these languages either, and require a sophisticated reasoning algorithm. Also, none of the existing systems can provide informative explanations about the answers, but point to related web pages of the knowledge resources available online.

We have built a software system, called BIOQUERY-ASP,[1] that handles all these challenges using Answer Set Programming (ASP) [9]. To address the first challenge, we have developed a controlled natural language for biomedical queries about drug discovery; this language is called BIOQUERY-CNL [5]. For instance, the queries Q1–Q4 are in BIOQUERY-CNL. Then we have built an intelligent user interface that allows users to enter biomedical queries in BIOQUERY-CNL and that presents the answers (possibly with explanations or related links, if requested) in BIOQUERY-CNL [6]. To address the second challenge, we have developed a rule layer over biomedical ontologies and databases, that not only integrates the concepts in these knowledge resources but also provides definitions of auxiliary concepts [1]. We have introduced an algorithm to identify the relevant parts of the rule layer and the knowledge resources with respect to the given query, and used automated reasoners of ASP to answer queries considering these relevant parts [4]. To address the third challenge, we have developed an algorithm to generate an explanation for a given answer, with respect to the query and the relevant parts of the rule layer and the knowledge resources [10,4]. The overall system architecture for BIOQUERY-ASP is presented in Figure 1.

We have shown the applicability of BIOQUERY-ASP to answer queries over large biomedical knowledge resources about genes, drugs and diseases, such as PHARMGKB,[2] DRUGBANK,[3] BIOGRID,[4] CTD,[5] and SIDER,[6] using efficient solvers of ASP [4]. For queries that are not concerned about similarity/diversity of genes/drugs, we have used the ASP solver CLASP [7]. For similarity/diversity queries, we have utilized the online methods of [2] for finding similar/diverse solutions, and thus used the ASP solver CLASP-NK, a variant of CLASP.

2 Demonstration of BIOQUERY-ASP by Examples

Let us demonstrate the use of BIOQUERY-ASP with four examples: the queries Q1, Q2, Q3 and Q4 presented in the introduction.

2.1 Representing a Query in BIOQUERY-CNL

BIOQUERY-ASP allows users to construct queries in the grammar of BIOQUERY-CNL (an extended version of the grammar introduced in our earlier work [5]), by providing

[1] http://krr.sabanciuniv.edu/projects/BioQuery-ASP/
[2] http://www.pharmgkb.org/
[3] http://www.drugbank.ca/
[4] http://thebiogrid.org/
[5] http://ctd.mdibl.org/
[6] http://sideeffects.embl.de/

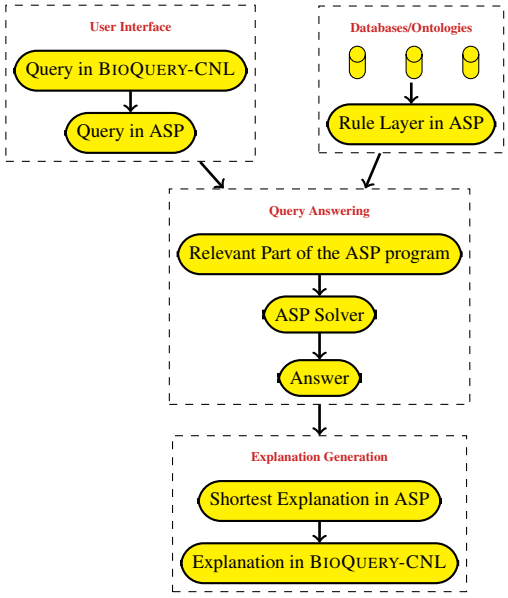

Fig. 1. System overview of BIOQUERY-ASP.

them two options: by showing them some templates so that they can choose one from among them (as shown in Figure 2 for the query Q1); or by guiding them to construct the query by showing the possibilities with an auto-completion feature (as shown in Figure 3 for the query Q1).

2.2 Transforming the Query in BIOQUERY-CNL to ASP

Once a query is constructed in BIOQUERY-CNL, we transform the query into ASP by an extension of the algorithm introduced in our earlier work [5]. For instance, the query Q1 is translated into the following ASP program:

```
what_be_genes(GN1) :- condition1(GN1), condition2(GN1).
condition1(GN) :- drug_gene("Epinephrine",GN).
condition2(GN) :- gene_gene(GN,"DLG4").
answer_exists :- what_be_genes(GN1).
```

where `condition1` and `condition2` are invented relations, and `gene_gene` and `drug_gene` are defined in the rule layer as follows:

```
drug_gene(D,G) :- drug_gene_pharmgkb(D,G).
drug_gene(D,G) :- drug_gene_ctd(D,G).

gene_gene(GN1,GN2) :- gene_gene_biogrid(GN1,GN2).
gene_gene(GN2,GN1) :- gene_gene(GN1,GN2).
```

Similarly, the query Q2 is translated into the following ASP program:

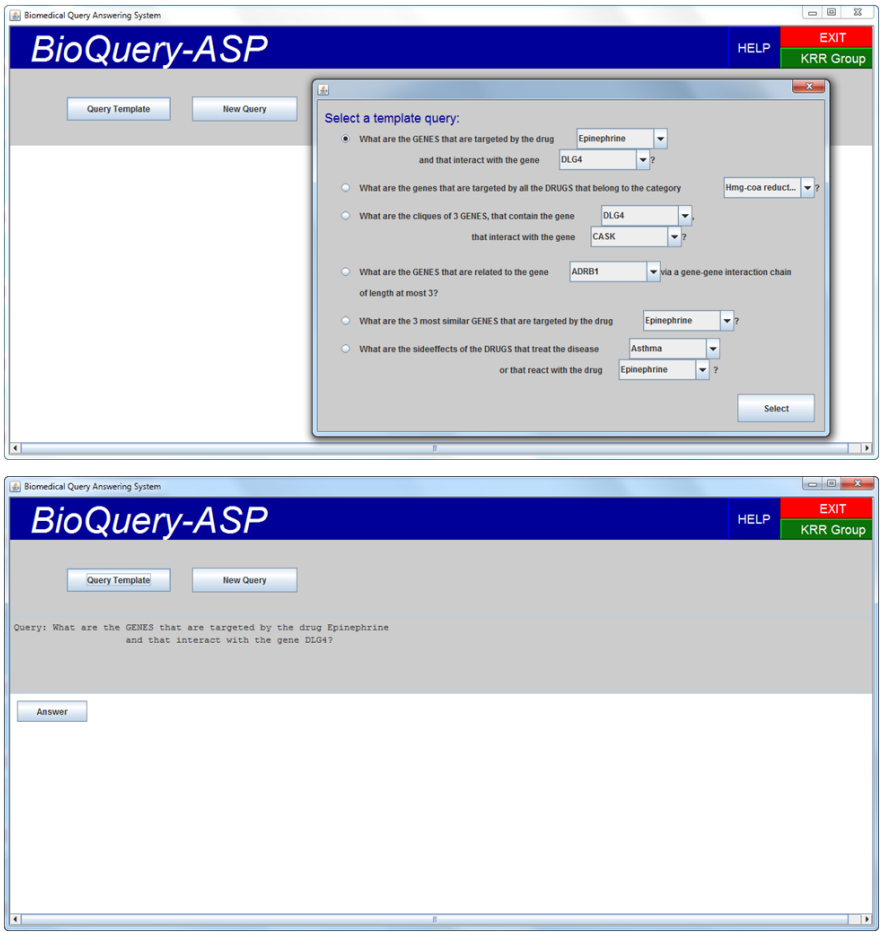

Fig. 2. In BIOQUERY-ASP a query can be constructed by making use of a template.

```
what_be_genes(GN1) :- not notcommon(GN1),
   gene_name(GN1), notcommon_exists.
notcommon(GN1) :- not drug_gene(D2,GN1),
   condition1(D2), gene_name(GN1).
notcommon_exists :- notcommon(X).
condition1(D) :- drug_category(D,"Hmg-coa reductase inhibitors").
answer_exists :- what_be_genes(GN).
```

The query Q3 is translated into the following ASP program:

```
what_be_genes(GN) :- condition1(GN).
condition1(GN) :- gene_reachable_from(GN,L).
start_gene("ADRB1").
max_chain_length(3).
answer_exists :- what_be_genes(GN).
```

where `gene_reachable_from` is defined in the rule layer as follows:

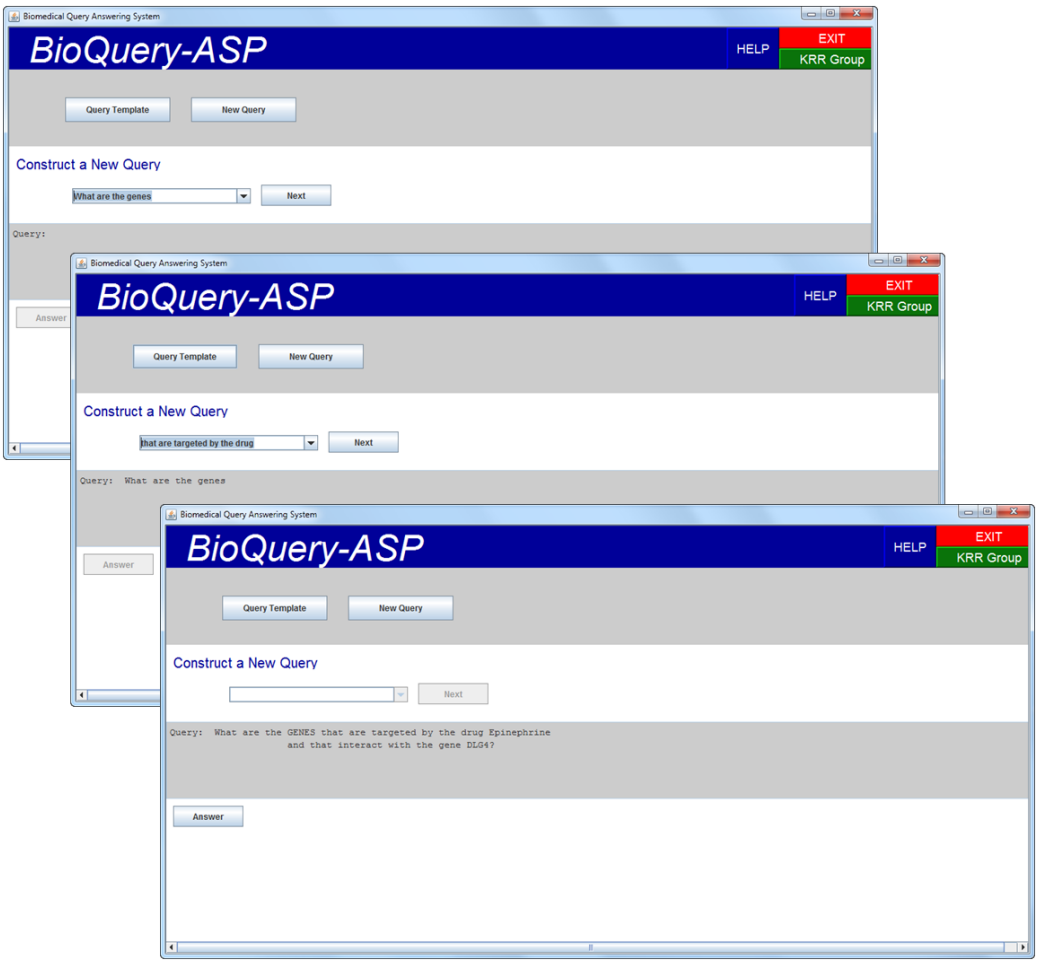

Fig. 3. In BIOQUERY-ASP a query can be constructed by making use of the auto-completion feature.

```
gene_reachable_from(X,1) :- gene_gene(X,Y), start_gene(Y).
gene_reachable_from(X,N+1) :- gene_gene(X,Z),
   gene_reachable_from(Z,N), 0 < N, N < L,
   max_chain_length(L).
```

Note that unlike the rules for `gene_gene` and `drug_gene` that integrate knowledge resources, the rules for `gene_reachable_from` define an auxiliary concept to be used for deep reasoning.

The query Q4 is translated into the following ASP program:

```
1{what_be_genes(GN) : condition1(GN)}1.
condition1(GN) :- drug_gene("Epinephrine", GN).
answer_exists :- what_be_genes(GN).
```

2.3 Extracting Information from the Knowledge Resources using ASP

Some of the ASP solvers, such as DLVHEX [3], provide constructs to import external theories that may be in different formats (e.g., ontologies in RDF(S)/OWL). For instance, consider as an external theory a Drug Ontology described in RDF. All triples from this theory can be exported using the external predicate `&rdf`:

```
triple_drug(X,Y,Z) :- &rdf["URI for Drug Ontology"](X,Y,Z).
```

Then the names of drugs can be extracted by DLVHEX using the rule:

```
drug_name(A) :- triple_drug(_,"drugproperties:name",A).
```

Similarly, gene-gene interactions could be extracted from a Gene Ontology by DLVHEX using the rules

```
gene_gene(G1,G2) :- triple_gene(X,"geneproperties:name",G1),
   triple_gene(X,"geneproperties:related_genes",B),
   triple_gene(B,Z,Y), Z!="rdf:type",
   triple_gene(Y,"geneproperties:name",G2).
```

Some knowledge resources are provided as relational databases, or more often as a set of triples (probably extracted from ontologies in RDF). In such cases, we introduce special algorithms to transform the relations into ASP.

2.4 Answering the Queries using ASP

At this stage, the given biomedical query Q, the rule layer, and the information extracted from the knowledge resources are all in ASP. Let us denote by Π the union of the rule layer and the information extracted from the knowledge resources. To be able to answer the given query Q efficiently, BIOQUERY-ASP extracts the relevant part of Π with respect to Q [4], and then computes a model (called "an answer set" [8]) for the relevant part (if exists) using a state-of-the-art ASP system, such as CLASP. After that, BIOQUERY-ASP extracts the answers to Q from the answer set, and presents them as a list. For instance, the answers to the query Q3 is shown in Figure 4.

For queries about similarity/diversity of genes, BIOQUERY-ASP uses the answer set solver CLASP-NK [2], an extension of CLASP that can compute similar/diverse answer sets for an ASP program with respect to a given distance measure. The idea behind CLASP-NK is to define the distance measure (as a C++ program) and modify the search algorithm of CLASP accordingly in the style of a branch-and-bound algorithm. BIOQUERY-ASP considers the semantic and functional similarity of genes defined over the gene ontology [11]; this measure can be computed by the software GOSEMSIM.

2.5 Generating Explanations for the Answers

Once an answer to a query (that does not involve aggregates as in Q3 and Q4) is computed, BIOQUERY-ASP can generate an explanation for it [4,10]. For instance, an explanation for the answer "ADRB1" to the query Q1 is shown in Figure 5. If an explanation cannot be found then related links to the knowledge resources are provided.

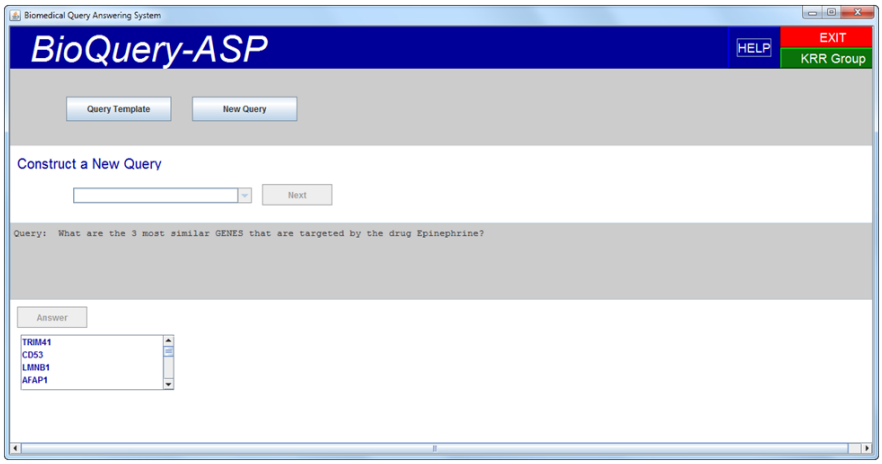

Fig. 4. BIOQUERY-ASP presents answers to a query as a list.

3 Conclusion

We have described the software system BIOQUERY-ASP that finds answers and generates explanations to complex biomedical queries over the available knowledge resources, such as, PHARMGKB, DRUGBANK, CTD, SIDER, BIOGRID, using the computational methods/tools of Answer Set Programming. These complex biomedical queries require appropriate integration of relevant knowledge from different knowledge resources; auxiliary definitions, such as, chains of drug-drug interactions, cliques of genes based on gene-gene relations, or similarity/diversity of genes/drugs; and further deep reasoning, like finding similar/diverse genes/drugs. No existing biomedical query answering systems (e.g., web services built over the available knowledge resources, that answer queries by means of keyword search) can directly answer such queries, or can generate explanations for answers. In that sense, BIOQUERY-ASP is a novel biomedical query answering system that can be useful for experts.

Acknowledgments.

We are grateful to Yelda Erdem (Sanovel Pharmaceuticals) for her suggestions about the biomedical queries related to drug discovery. This work has been supported by TUBITAK Grant 108E229.

References

1. Bodenreider, O., Coban, Z.H., Doganay, M.C., Erdem, E.: A preliminary report on answering complex queries related to drug discovery using answer set programming. In: Proc. of ALPSWS (2008)
2. Eiter, T., Erdem, E., Erdogan, H., Fink, M.: Finding similar or diverse solutions in answer set programming. In: Proc. of ICLP. pp. 342–356 (2009)
3. Eiter, T., G.Ianni, R.Schindlauer, H.Tompits: Effective integration of declarative rules with external evaluations for Semantic-Web reasoning. In: Proc. of ESWC (2006)

4. Erdem, E., Erdem, Y., Erdogan, H., Oztok, U.: Finding answers and generating explanations for complex biomedical queries. In: Proc. of AAAI (2011)
5. Erdem, E., Yeniterzi, R.: Transforming controlled natural language biomedical queries into answer set programs. In: Proc. of the Workshop on BioNLP. pp. 117–124 (2009)
6. Erdogan, H., Oztok, U., Erdem, Y., Erdem, E.: Querying biomedical ontologies in natural language using answer set programming. In: Proc. of SWAT4LS (2010)
7. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: clasp: A Conflict-Driven Answer Set Solver. In: Proc. of LPNMR. pp. 260–265 (2007)
8. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
9. Lifschitz, V.: What is answer set programming? In: Proc. of AAAI (2008)
10. Oztok, U., Erdem, E.: Generating explanations for complex biomedical queries. In: Proc. of AAAI (2011)
11. Wang, J.Z., Du, Z., Payattakool, R., Yu, S.P., Chen, C.F.: A new method to measure the semantic similarity of go terms. Bioinformatics 23, 1274–1281 (2007)

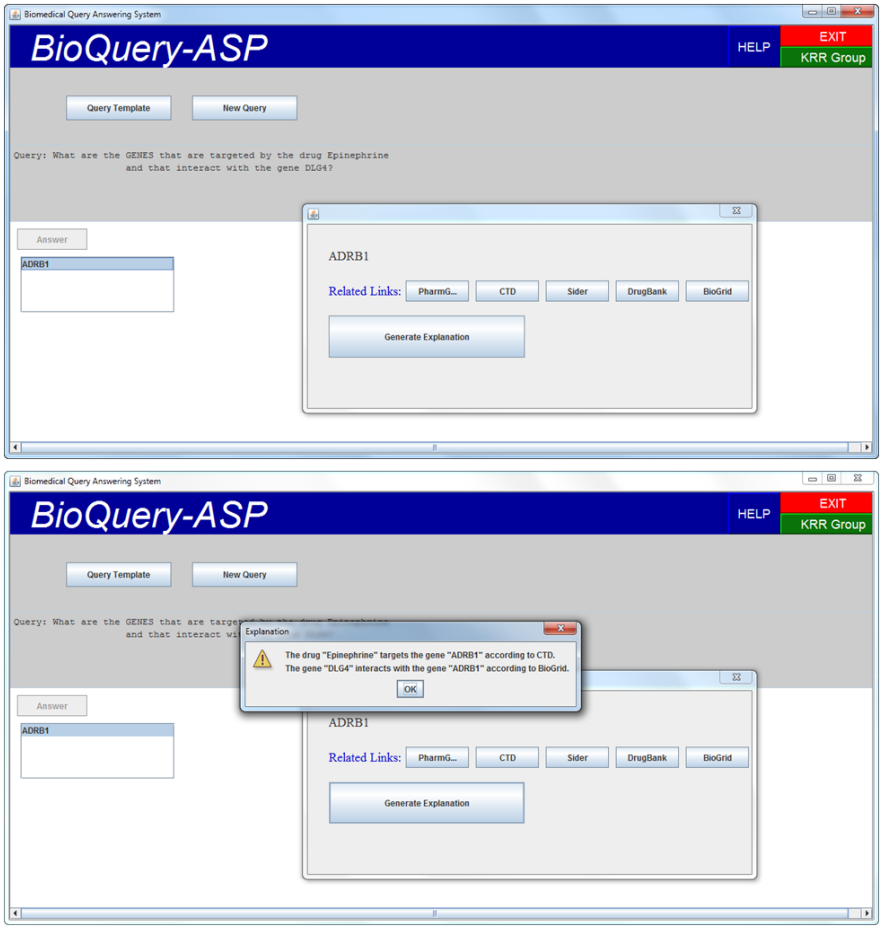

Fig. 5. BIOQUERY-ASP can generate an explanation for an answer to the given query.