Explanation Generation in Applications of Answer Set Programming

Esra Erdem

Sabanci University, Istanbul, Turkiye

COST DigForASP, TU Wien, January 27, 2023

Answer Set Programming (ASP)

- Declarative programming paradigm.
- Theoretical basis: answer set semantics (Gelfond and Lifschitz, 1988)
- Expressive representation languages: Defaults, recursive definitions, aggregates, preferences, etc.
- ASP solvers: DLV (University of Calabria), CLINGO (University of Potsdam)
- Applications: planning, learning, multi-agent systems, natural language understanding, robotics, bioinformatics, systems biology, VLSI design, historical linguistics, game theory, e-tourism, etc.

Erdem, Gelfond, Leone: Applications of Answer Set Programming. AI Magazine 37(3): 53-68 (2016)

Traditional Programs

- Programs consist of rules of the form

$$Head \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n$$

  where each A_i is a propositional atom, and *Head* is a propositional atom or $\bot$.
- A rule is called a *fact* if $m = n = 0$, and a *constraint* if *Head* is $\bot$.

Answer Sets: Nonmonotonicity

Program	Answer sets
$p \leftarrow \; not\ q$	$\{p\}$
$p \leftarrow \; not\ q$ $q \leftarrow \; not\ p$	$\{p\}, \{q\}$
$p \leftarrow \; not\ q$ $q \leftarrow \; not\ p$ $r \leftarrow \; p$ $r \leftarrow \; q$	$\{p, r\}, \{q, r\}$

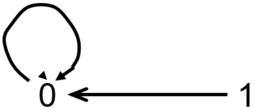
Answer Sets: Generate and Eliminate

Program	Answer sets
$p \leftarrow \ not\ q$ $q \leftarrow \ not\ p$	$\{p\}, \{q\}$
$p \leftarrow \ not\ q$ $q \leftarrow \ not\ p$ $\leftarrow q$	$\{p\}$
$p \leftarrow \ not\ q$ $q \leftarrow \ not\ p$ $\leftarrow not\ q$	$\{q\}$

Choice Rules and Cardinality Constraints

Program	Answer sets
$\{p, q\} \leftarrow$	$\{\}, \{p\}, \{q\}, \{p, q\}$
$1\{p, q\} \leftarrow$	$\{p\}, \{q\}, \{p, q\}$
$\{p, q\}1 \leftarrow$	$\{\}, \{p\}, \{q\}$
$1\{p, q\}1 \leftarrow$	$\{p\}, \{q\}$

Example: Checking Reachability



Program:

$$
\begin{array}{l}
edge(0,0) \leftarrow \\
edge(1,0) \leftarrow \\
reachable(x,y) \leftarrow edge(x,y) \\
reachable(x,y) \leftarrow edge(x,z), reachable(z,y)
\end{array}
$$

Answer set:

$$\{edge(0,0), edge(1,0), reachable(0,0), reachable(1,0)\}$$

Declarative Problem Solving using ASP

The basic idea is

- to represent the given problem by a **program**,
- to find **answer sets** for the program using an **ASP solver**, and
- to extract the solutions from the answer sets.

Example: Graph Coloring Problem

Given a set $C = \{C_1, \ldots, C_n\}$ of colors and a graph $G = \langle V, E \rangle$, decide whether there exists an assignment of colors in C to vertices in V such that:

- every vertex in V is assigned to exactly one color,
- no two adjacent vertices are assigned to the same color.

Generate: assign exactly one color to every vertex.

$$1\{assign(v, C_1), \ldots, assign(v, C_n)\}1 \leftarrow vertex(v) \qquad (v \in V)$$

Test: ensure that no two adjacent vertices have the same color.

$$\leftarrow assign(v, c), assign(v', c), edge(v, v') \qquad (v \neq v')$$

A solution can be computed using an ASP solver:

$$\{assign(1, C1), assign(2, C3), \ldots\}$$

Explanation Generation in Three Applications of ASP

- Generating explanations for complex biomedical queries
- Explanation generation for multi-agent path finding
- Explainable robotic plan execution monitoring

Explanation Generation in Three Applications of ASP

- Generating explanations for complex biomedical queries
- Explanation generation for multi-agent path finding
- Explainable robotic plan execution monitoring

Biomedical Query Answering and Explanation Generation

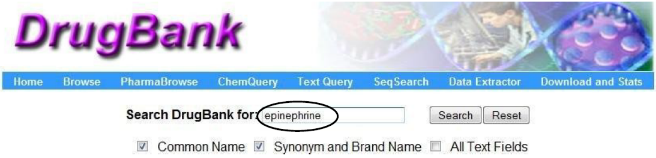
- Biomedical data is stored in various structured forms and at different locations.
- With the current Web technologies, reasoning over these data is limited to answering simple queries by keyword search and by some direction of humans.
- Vital research, like drug discovery, requires high-level reasoning.

A Simple Query

What are the genes that are targeted by the drug Epinephrine?

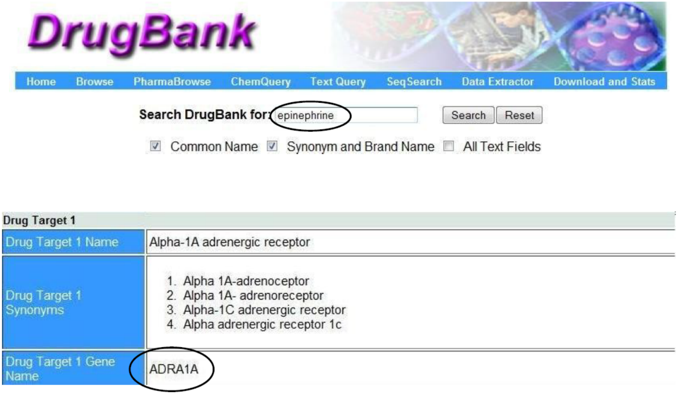
A Simple Query

What are the genes that are targeted by the drug Epinephrine?



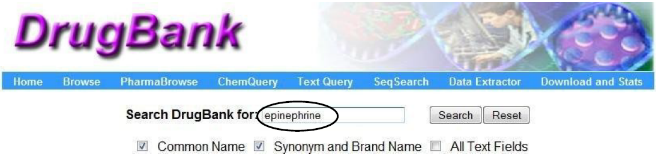
A Simple Query

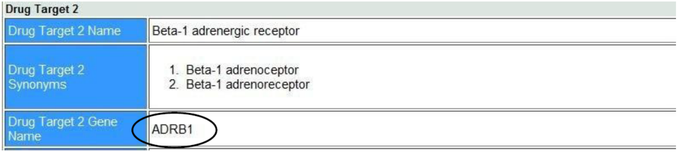
What are the genes that are targeted by the drug Epinephrine?



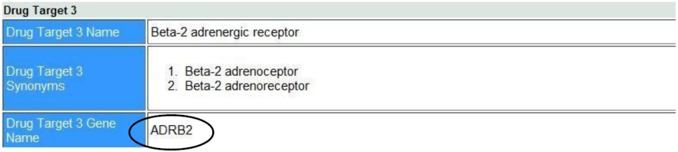
A Simple Query

What are the genes that are targeted by the drug Epinephrine?





What are the genes that are targeted by the drug Epinephrine?



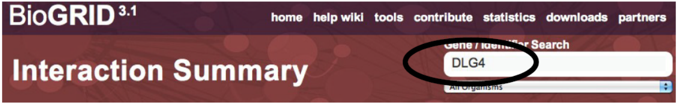
Drug Target 3	
Drug Target 3 Name	Beta-2 adrenergic receptor
Drug Target 3 Synonyms	1. Beta-2 adrenoceptor 2. Beta-2 adrenoreceptor
Drug Target 3 Gene Name	ADRB2

Another Simple Query

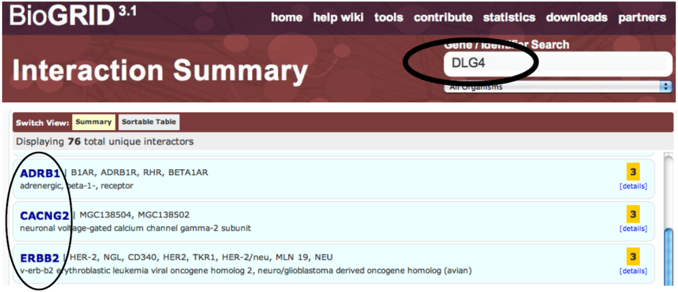
What are the genes that interact with the gene DLG4?

Another Simple Query

What are the genes that interact with the gene DLG4?

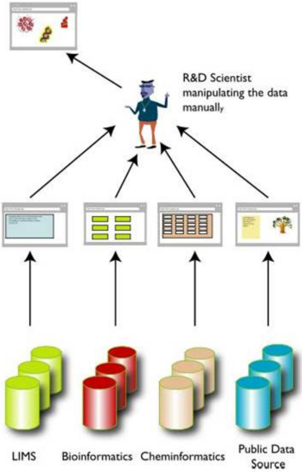


What are the genes that interact with the gene DLG4?



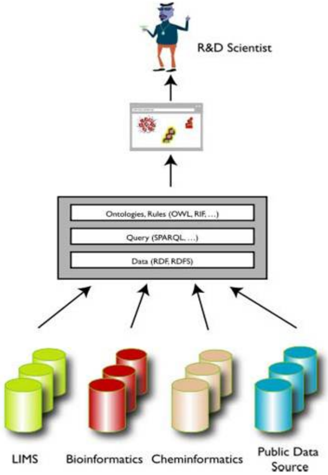
Complex Queries

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



Our goal is...

... to extract relevant parts of the knowledge resources, integrate them, answer the queries efficiently, and generate explanations.



Complex Queries

- What are the genes that are targeted by the drug Epinephrine and that interact with the gene DLG4?
- What are the genes that are targeted by all the drugs that belong to the category Hmg-coa reductase inhibitors?
- What are the drugs that treat the disease Depression and that do not target the gene ACYP1?
- What are the cliques of 5 genes, that contain the gene DLG4?
- What are the genes that are related to the gene ADRB1 via a gene-gene relation chain of length at most 3?
- What are the 3 most similar genes that are targeted by the drug Epinephrine?

Challenges

1. It is hard to represent a query in a formal language.
2. Databases/ontologies are in different formats/locations.
3. Complex queries require recursive definitions, aggregates, etc.
4. Databases/ontologies are large.
5. Experts may ask for further explanations.

Challenges

1. It is hard to represent a query in a formal language.
   - Represent queries in a controlled natural language.
2. Databases/ontologies are in different formats/locations.
3. Complex queries require recursive definitions, aggregates, etc.
4. Databases/ontologies are large.
5. Experts may ask for further explanations.

Challenges

1. It is hard to represent a query in a formal language.
   - Represent queries in a controlled natural language.
2. Databases/ontologies are in different formats/locations.
   - Integrate different sources via a rule layer in ASP.
3. Complex queries require recursive definitions, aggregates, etc.
4. Databases/ontologies are large.
5. Experts may ask for further explanations.

Challenges

1. It is hard to represent a query in a formal language.
   - Represent queries in a controlled natural language.
2. Databases/ontologies are in different formats/locations.
   - Integrate different sources via a rule layer in ASP.
3. Complex queries require recursive definitions, aggregates, etc.
   - Represent queries as ASP programs.
4. Databases/ontologies are large.

5. Experts may ask for further explanations.

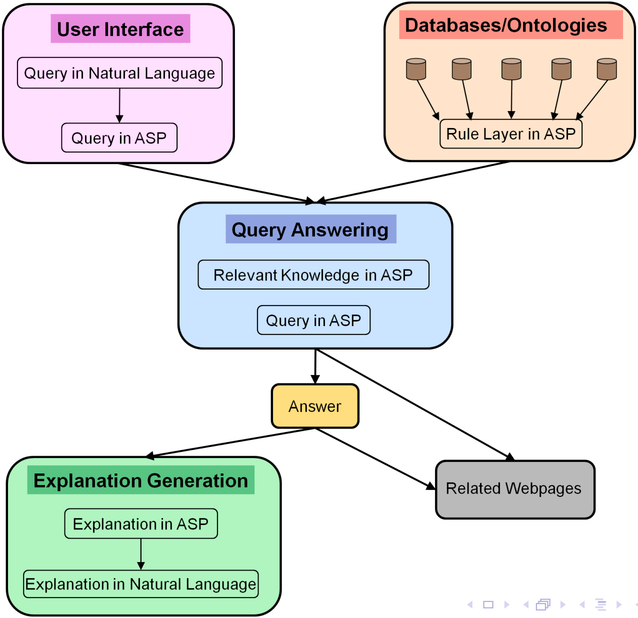
Challenges

1. It is hard to represent a query in a formal language.
   - Represent queries in a controlled natural language.
2. Databases/ontologies are in different formats/locations.
   - Integrate different sources via a rule layer in ASP.
3. Complex queries require recursive definitions, aggregates, etc.
   - Represent queries as ASP programs.
4. Databases/ontologies are large.
   - Extract the relevant part for faster reasoning.
5. Experts may ask for further explanations.

Challenges

1. It is hard to represent a query in a formal language.
   - Represent queries in a controlled natural language.
2. Databases/ontologies are in different formats/locations.
   - Integrate different sources via a rule layer in ASP.
3. Complex queries require recursive definitions, aggregates, etc.
   - Represent queries as ASP programs.
4. Databases/ontologies are large.
   - Extract the relevant part for faster reasoning.
5. Experts may ask for further explanations.
   - Generate shortest explanations.

BIOQUERY-ASP: System Overview



BIOQUERY-CNL*: A CNL for biomedical queries

- We consider queries in a specific domain, namely biomedicine, and over specific sources of information, namely biomedical ontologies.
- We design a CNL (BIOQUERY-CNL*) for representing biomedical queries, and develop an algorithm to transform it into ASP.
- This allows us to use ASP systems to find answers to queries expressed in BIOQUERY-CNL*.

Erdem, Yeniterzi: Transforming Controlled Natural Language Biomedical Queries into Answer Set Programs. BioNLP@HLT-NAACL 2009.

BioQuery-CNL* syntax

BioQuery-CNL* **Grammar:**

Query → WhatQuery QuestionMark
WhatQuery → What are OfRelation NestedPredicateRelation
OfRelation → *Noun*() of *Type*()
NestedPredicateRelation → (...)* that PredicateRelation
PredicateRelation → InstanceRelation (...)*
InstanceRelation → (Neg)? *Verb*() the *Type*() *Instance*()
QuestionMark → ?

Ontology functions:

Type() returns the type information, e.g., gene, disease, drug
Instance(T) returns instances of the type T, e.g., Asthma for type disease
Verb(T, T') returns the verbs where type T is the subject and type T' is the object, e.g., drug treat disease

Noun(T) returns the nouns that are related to the type T, e.g., side-effects of type drug

Example: What are the side-effects of the drugs that treat the disease Asthma?

Representing Queries in ASP

Query in BIOQUERY-CNL*: What are the genes that are targeted by all the drugs that belong to the category Hmg-coa reductase inhibitors?

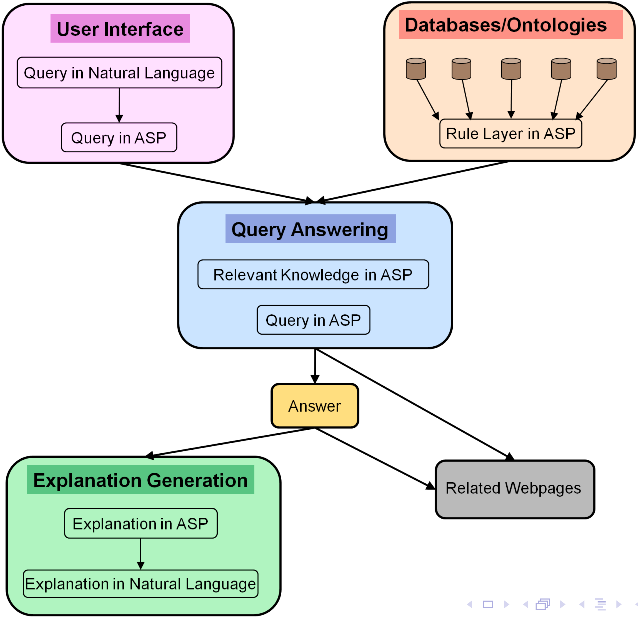
Query in ASP:

$$notcommon(gn_1) \leftarrow not\ drug\_gene(d_2, gn_1), condition_1(d_2)$$
$$condition_1(d) \leftarrow drug\_category(d, \text{``}Hmg - coa\ reductase\ inhibitors\text{''})$$

$$what\_be\_genes(gn_1) \leftarrow not\ notcommon(gn_1), notcommon\_exists$$
$$notcommon\_exists \leftarrow notcommon(x)$$

$$answer\_exists \leftarrow what\_be\_genes(gn)$$

BioQuery-ASP: System Overview



Extraction and Integration of Knowledge using ASP

Knowledge from RDF(S)/OWL ontologies can be extracted using "external predicates" supported by the ASP solver DLVHEX:

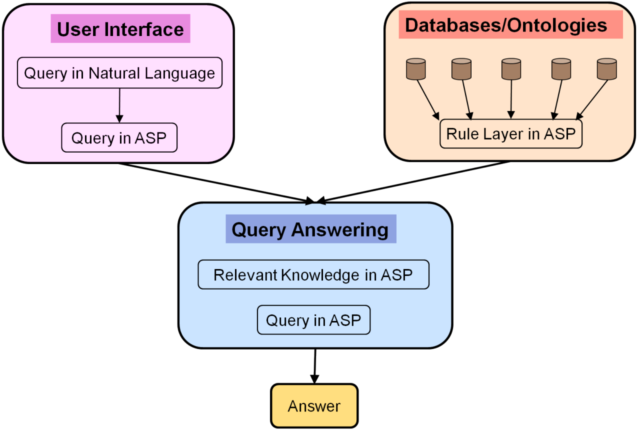
$$\begin{array}{l} triple\_gene(x,y,z) \leftarrow \&rdf[\text{``}URIforGeneOntology\text{''}](x,y,z) \\ gene\_gene(g_1,g_2) \leftarrow triple\_gene(x, \text{``}geneproperties:name\text{''}, g_1), \\ \qquad triple\_gene(x, \text{``}geneproperties:related\_genes\text{''}, b), \ldots \end{array}$$

ASP rules integrate the extracted knowledge, or define new concepts:

$$\begin{array}{l} gene\_reachable\_from(x,1) \leftarrow gene\_gene(x,y), start\_gene(y) \\ gene\_reachable\_from(x,n+1) \leftarrow gene\_gene(x,z), \\ \qquad gene\_reachable\_from(z,n), max\_chain\_length(l) \quad (0<n, n<l) \end{array}$$

Erdogan et al.: Querying Biomedical Ontologies in Natural Language using Answer Set Programming. SWAT4LS 2010.

Query Answering in ASP



- Generally, only a small part of the underlying databases and the rule layer is related to the given query.
- We introduce a method to identify the relevant part of the ASP program for more efficient query answering.

Relevant Part of a Program

Underlying databases as facts:

$gene\_gene(G1, G2) \leftarrow$ $\quad$ $gene\_gene(G2, G3) \leftarrow$

$drug\_drug(D1, D2) \leftarrow$ $\quad$ $drug\_drug(D2, D3) \leftarrow$

Rule layer:

$gene\_gene(g_1, g_2) \leftarrow gene\_gene(g_2, g_1)$

$gene\_related\_gene(g_1, g_2) \leftarrow gene\_gene(g_1, g_2)$

$gene\_related\_gene(g_1, g_3) \leftarrow gene\_related\_gene(g_1, g_2), gene\_gene(g_2, g_3)$

$drug\_drug(d_1, g_2) \leftarrow drug\_drug(d_2, d_1)$

$drug\_related\_drug(g_1, g_2) \leftarrow drug\_drug(d_1, d_2)$

$drug\_related\_drug(g_1, g_3) \leftarrow drug\_related\_drug(d_1, d_2), drug\_drug(d_2, d_3)$

Query: What are the genes that are related to gene $G1$?

$what\_be\_genes(g) \leftarrow gene\_related\_gene(g, G1)$

Relevant Part of a Program

Underlying databases as facts:

$gene\_gene(G1, G2) \leftarrow$ $\quad$ $gene\_gene(G2, G3) \leftarrow$

$drug\_drug(D1, D2) \leftarrow$ $\quad$ $drug\_drug(D2, D3) \leftarrow$

Rule layer:

$gene\_gene(g_1, g_2) \leftarrow gene\_gene(g_2, g_1)$

$gene\_related\_gene(g_1, g_2) \leftarrow gene\_gene(g_1, g_2)$

$gene\_related\_gene(g_1, g_3) \leftarrow gene\_related\_gene(g_1, g_2), gene\_gene(g_2, g_3)$

$drug\_drug(d_1, g_2) \leftarrow drug\_drug(d_2, d_1)$

$drug\_related\_drug(g_1, g_2) \leftarrow drug\_drug(d_1, d_2)$

$drug\_related\_drug(g_1, g_3) \leftarrow drug\_related\_drug(d_1, d_2), drug\_drug(d_2, d_3)$

Query: What are the genes that are related to gene $G1$?

$what\_be\_genes(g) \leftarrow gene\_related\_gene(g, G1)$

Identifying the relevant part improves the computational time up to 100 times.

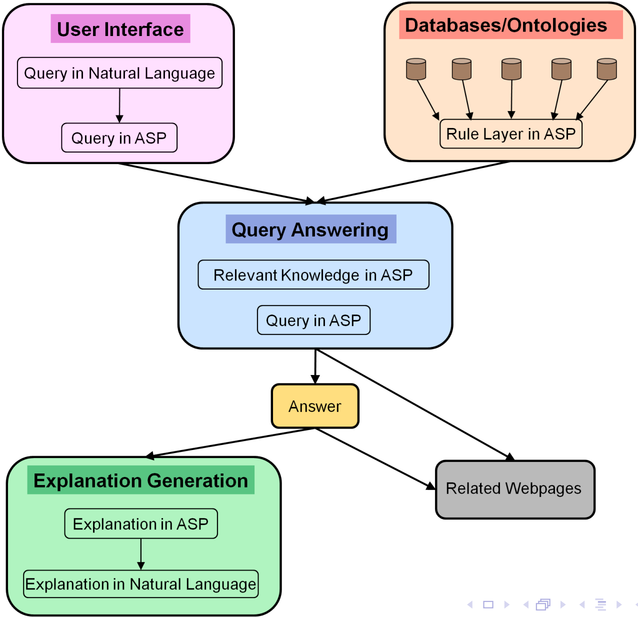
Identifying the Relevant Part of a Program

Theorem 1

Let Π be a stratified normal program, Q be a general program. Then $Rel_{\Pi,Q}$ is the relevant part of Π with respect to Q.

Erdem et al.: Finding Answers and Generating Explanations for Complex Biomedical Queries. AAAI 2011.

BioQuery-ASP: System Overview



Explanations

ASP program Π:

$$
\begin{aligned}
a &\leftarrow b, c \\
a &\leftarrow d \\
d &\leftarrow \\
b &\leftarrow c \\
c &\leftarrow
\end{aligned}
$$

An answer set X for Π: $\{a, b, c, d\}$

An explanation for a wrt Π and X:

$$
\begin{tikzcd}
 & a \leftarrow b, c \arrow[dl] \arrow[dr] & \\
b \leftarrow c \arrow[d] & & c \leftarrow \\
c \leftarrow & &
\end{tikzcd}
$$

Explanations

ASP program Π:

$$
\begin{aligned}
&a \leftarrow b, c \\
&a \leftarrow d \\
&d \leftarrow \\
&b \leftarrow c \\
&c \leftarrow
\end{aligned}
$$

An answer set X for Π: $\{a, b, c, d\}$

Another explanation for a wrt Π and X:

$$
\begin{array}{c}
a \leftarrow d \\
\downarrow \\
d \leftarrow
\end{array}
$$

Explanations

ASP program Π:

$$
\begin{aligned}
a &\leftarrow b, c \\
a &\leftarrow d \\
d &\leftarrow \\
b &\leftarrow c \\
c &\leftarrow
\end{aligned}
$$

An answer set X for Π: $\{a, b, c, d\}$

Support of an Atom by a Rule

Supports

A rule r <u>supports</u> an atom p using atoms in Y but not in Z, if the following hold:

$$\begin{array}{l} H(r) = p, \\ B^{+}(r) \subseteq Y \setminus Z, \\ B^{-}(r) \cap Y = \emptyset, \\ Y \models B_{card}(r). \end{array}$$

Example:

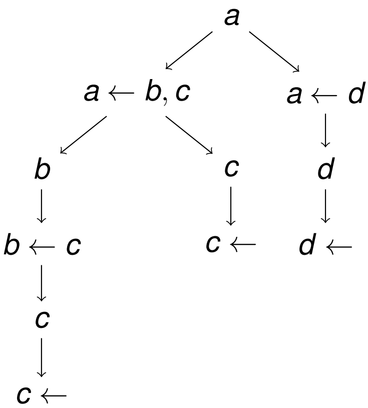
The rule

$$a \leftarrow d, 1 \leq \{b, c\} \leq 2, not\ e$$

supports the atom a with respect to $Y = \{a, b, c, d, f\}$ but $Z = \{f\}$.

Finding Explanations

The and-or explanation tree for atom *a* with respect to Π and *X*:

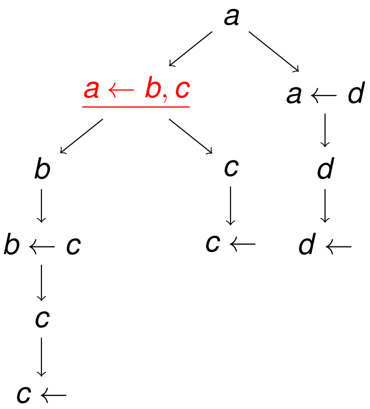


Π :

$$
\begin{array}{l}
a \leftarrow b, c \\
a \leftarrow d \\
d \leftarrow \\
b \leftarrow c \\
c \leftarrow
\end{array}
$$

$$X = \{a, b, c, d\}$$

The and-or explanation tree for atom a with respect to Π and X:



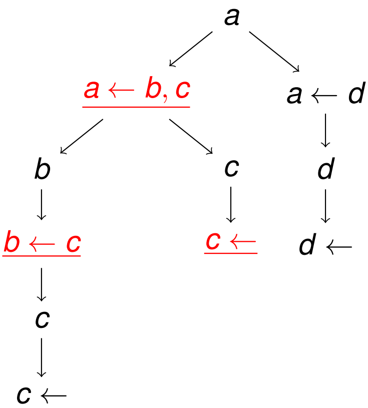
Π :

$a \leftarrow b, c$
$a \leftarrow d$
$d \leftarrow$
$b \leftarrow c$
$c \leftarrow$

$X = \{a, b, c, d\}$

Finding Explanations

The and-or explanation tree for atom a with respect to Π and X:



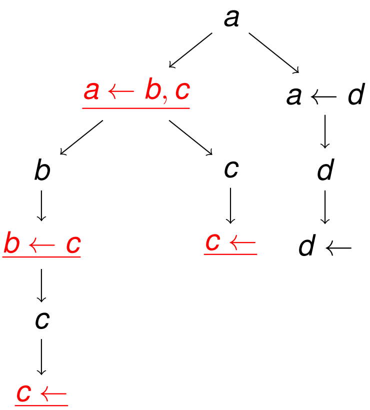
Π :

$a \leftarrow b, c$
$a \leftarrow d$
$d \leftarrow$
$b \leftarrow c$
$c \leftarrow$

$X = \{a, b, c, d\}$

Finding Explanations

The and-or explanation tree for atom a with respect to Π and X:



Π :

$a \leftarrow b, c$
$a \leftarrow d$
$d \leftarrow$
$b \leftarrow c$
$c \leftarrow$

$X = \{a, b, c, d\}$

Finding Explanations

An explanation for atom a with respect to Π and X:

$$
\begin{tikzcd}
 & a \leftarrow b, c \arrow[dl] \arrow[dr] & \\
b \leftarrow c \arrow[d] & & c \leftarrow \\
c \leftarrow & &
\end{tikzcd}
$$

Π :

$$
\begin{aligned}
& a \leftarrow b, c \\
& a \leftarrow d \\
& d \leftarrow \\
& b \leftarrow c \\
& c \leftarrow
\end{aligned}
$$

$$X = \{a, b, c, d\}$$

Another Example

An explanation for atom a with respect to Π and X:

$$a \leftarrow d, 1 \leq \{b, c\} \leq 2$$
$$\downarrow$$
$$d \leftarrow$$

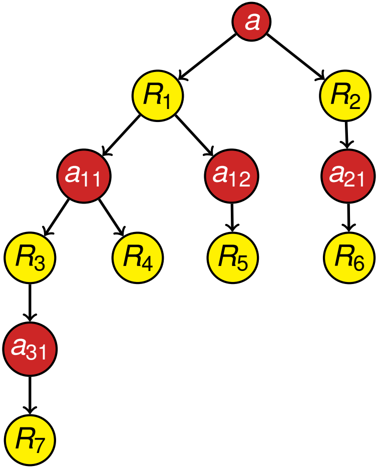
Π :

$$a \leftarrow d, not\ b$$
$$a \leftarrow d, 1 \leq \{b, c\} \leq 2$$
$$d \leftarrow$$
$$b \leftarrow c$$
$$c \leftarrow$$

$$X = \{a, b, c, d\}$$

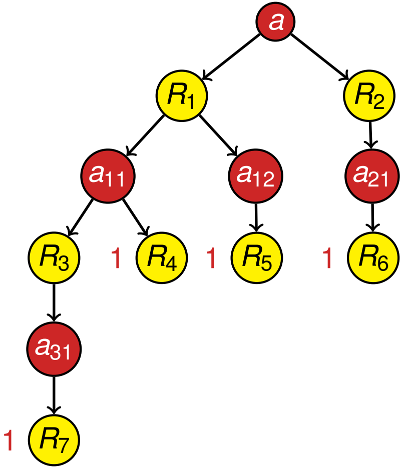
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



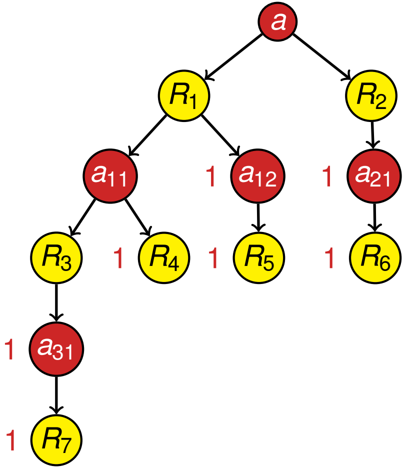
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



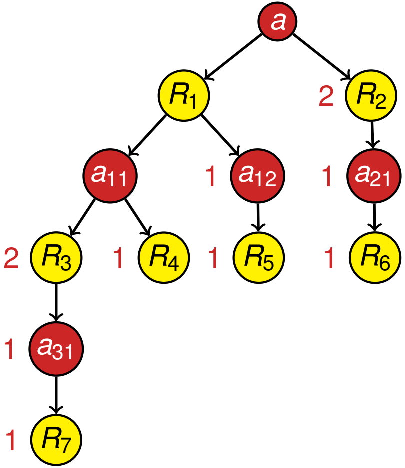
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



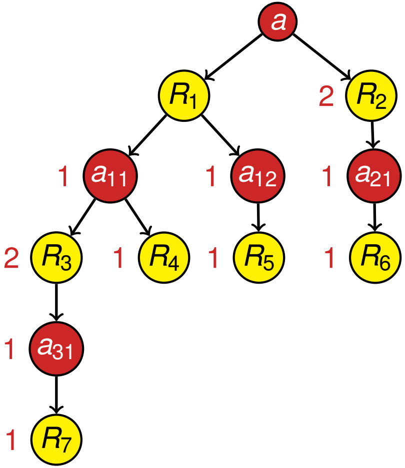
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



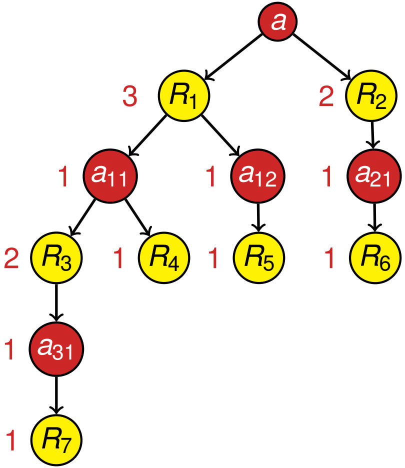
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



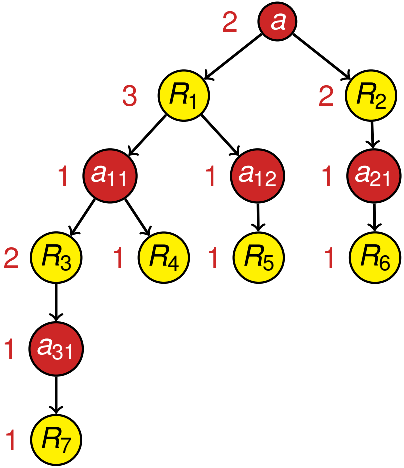
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



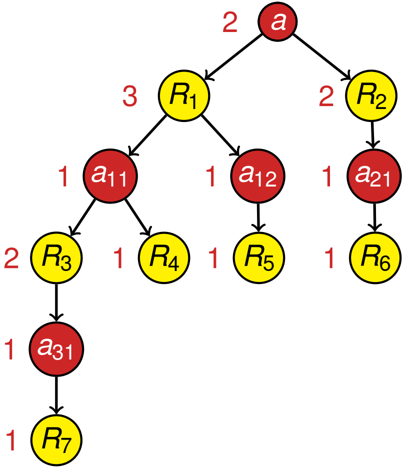
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



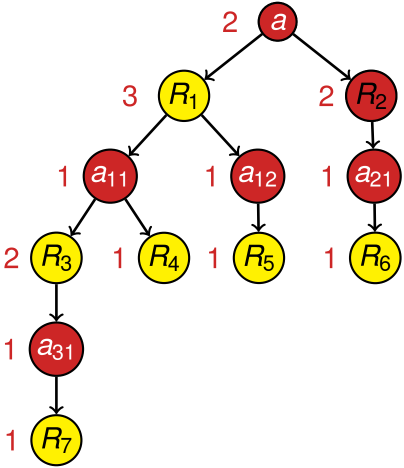
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



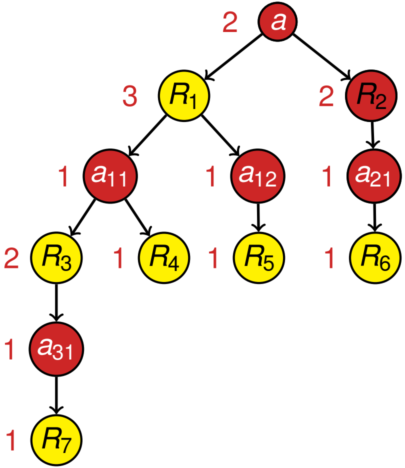
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



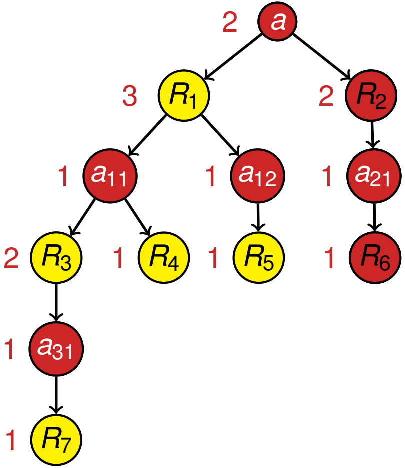
Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



Shortest Explanations

- $W(a) = min_{c \in child(a)}(W(c))$
- $W(r) = \sum_{c \in child(r)} W(c) + 1$



Explanation Generation

Theorem 2

Let Π be a normal ASP program, X be an answer set for Π and p be an atom in X. Our algorithm generates a shortest explanation for p with respect to Π and X.

Erdem, Oztok: Generating explanations for biomedical queries. TPLP 2015.

Example: Explanation Generation

Query in BIOQUERY-CNL*: What are the genes that are targeted by the drug Epinephrine and that interact with the gene DLG4?

An Answer: `ADRB1`

Shortest Explanation in ASP:

$$\mathit{what\_be\_genes}(\mathit{ADRB}1) \leftarrow \mathit{drug\_gene}(\mathit{Epinephrine}, \mathit{ADRB}1), \mathit{gene\_gene}(\mathit{ADRB}1, \mathit{DLG}4)$$

$$\mathit{drug\_gene}(\mathit{Epinephrine}, \mathit{ADRB}1) \leftarrow \mathit{drug\_gene\_ctd}(\mathit{Epinephrine}, \mathit{ADRB}1)$$

$$\downarrow$$

$$\mathit{drug\_gene\_ctd}(\mathit{Epinephrine}, \mathit{ADRB}1) \leftarrow$$

$$\mathit{gene\_gene}(\mathit{ADRB}1, \mathit{DLG}4) \leftarrow \mathit{gene\_gene}(\mathit{DLG}4, \mathit{ADRB}1)$$

$$\downarrow$$

$$\mathit{gene\_gene}(\mathit{DLG}4, \mathit{ADRB}1) \leftarrow \mathit{gene\_gene\_biogrid}(\mathit{DLG}4, \mathit{ADRB}1)$$

$$\downarrow$$

$$\mathit{gene\_gene\_biogrid}(\mathit{DLG}4, \mathit{ADRB}1) \leftarrow$$

Explanation in Natural Language:

```
The drug Epinephrine targets the gene ADRB1 according to CTD.
The gene DLG4 interacts with the gene ADRB1 according to BioGrid.
```

Example: Different Explanations

Query in BIOQUERY-CNL*: What are the drugs that treat the disease Asthma or that react with the drug Epinephrine?

Answer: `Doxepin`

Explanation:

```
what_be_drugs("Doxepin") :- drug_drug("Doxepin","Epinephrine").
   drug_drug("Doxepin","Epinephrine") :- drug_drug("Epinephrine","Doxepin").
       drug_drug("Epinephrine","Doxepin") :- drug_drug_drugbank("Epinephrine","Doxepin").
           drug_drug_drugbank("Epinephrine","Doxepin").
```

Example: Different Explanations

Query in BIOQUERY-CNL*: What are the drugs that treat the disease Asthma or that react with the drug Epinephrine?

Answer: `Doxepin`

Another Explanation:

```
what_be_drugs("Doxepin") :- drug_disease("Doxepin","Asthma").
   drug_disease("Doxepin","Asthma") :- drug_disease_ctd("Doxepin","Asthma").
       drug_disease_ctd("Doxepin","Asthma").
```

Distance Measure for Explanations

Let Z be a set of (previously computed) explanations and S be a (to be computed) explanation.

Let R_Z and R_S be the rule vertices contained in Z and S.

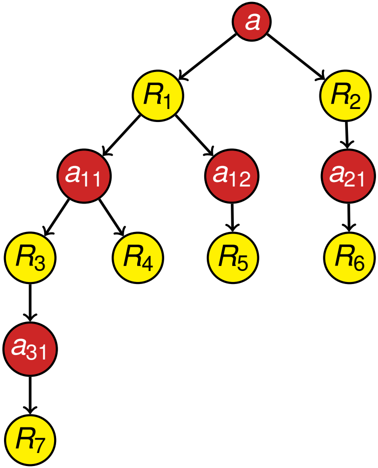
The distance function Δ_D between Z and S is defined as follows.

$$\Delta_D(Z, S) = |R_S \backslash R_Z|$$

Generating k Different Explanations

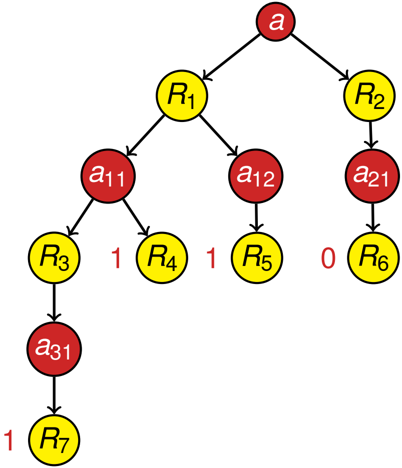
- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

$R = \{R_2, R_6\}$



- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

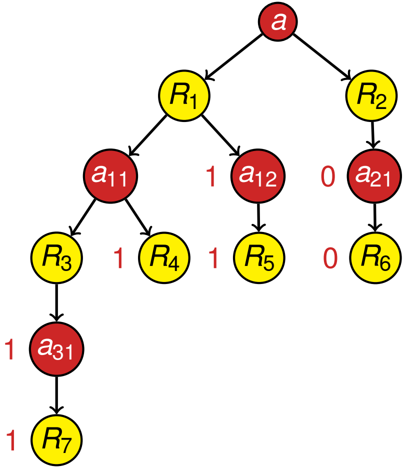
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

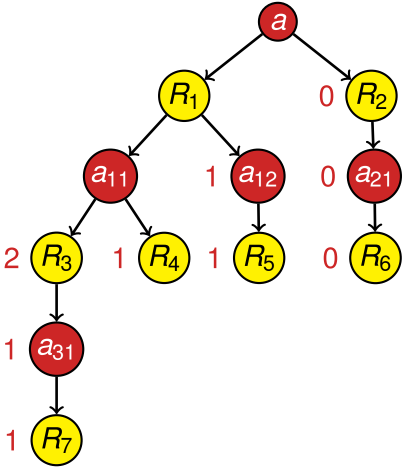
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

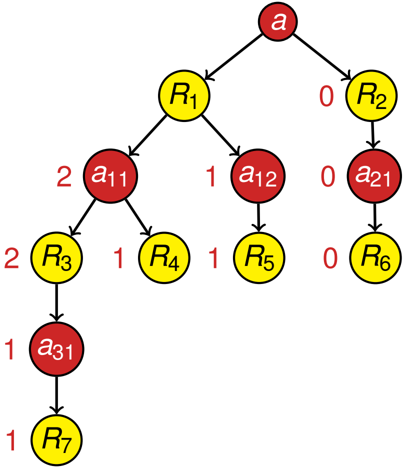
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

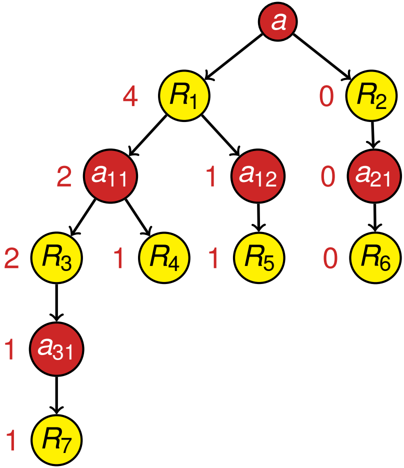
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

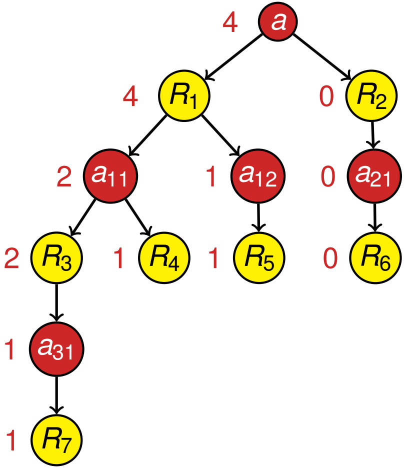
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

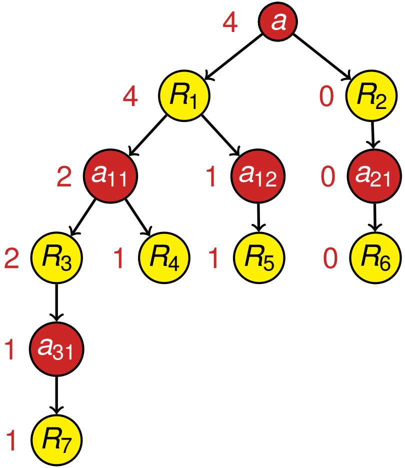
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

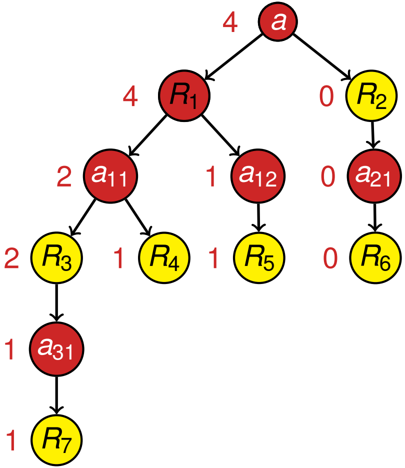
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

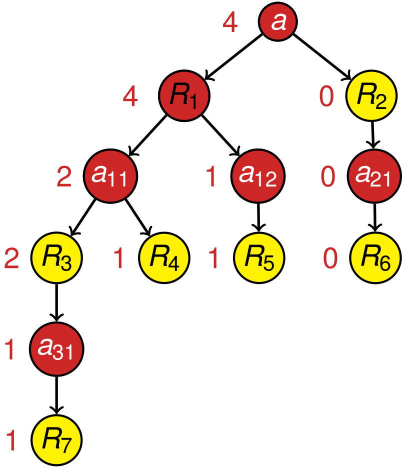
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

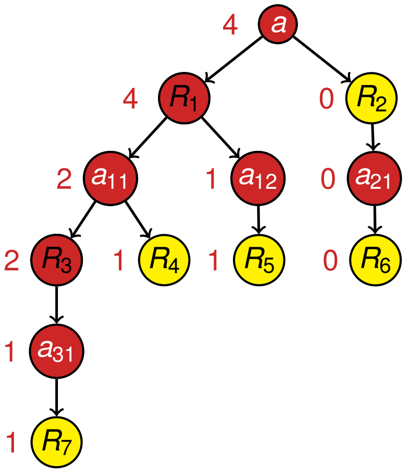
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

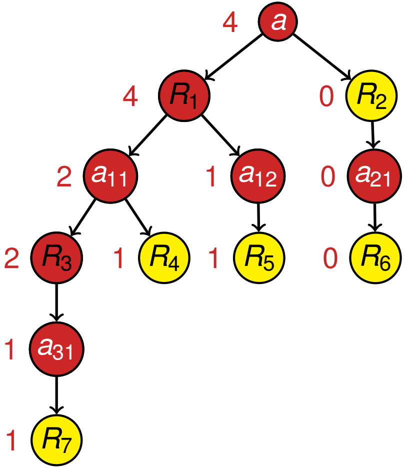
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

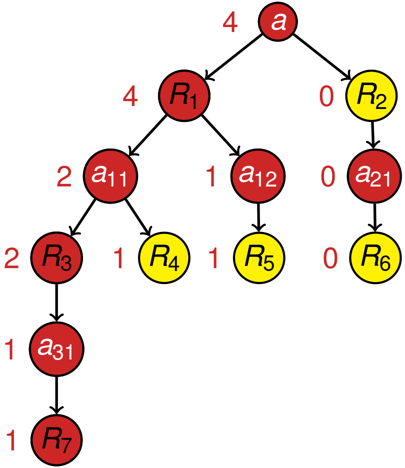
$R = \{R_2, R_6\}$



Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

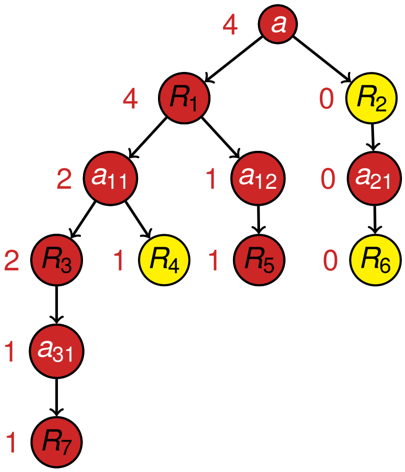
$R = \{R_2, R_6\}$



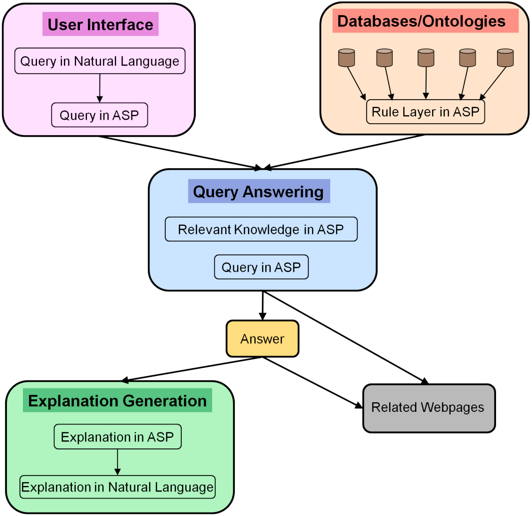
Generating k Different Explanations

- $W_{T,R}(a) = max_{c \in child(a)}(W_{T,R}(c))$
- $W_{T,R}(r) = \sum_{c \in child(r)} W_{T,R}(c)$ if $r \in R$
- $W_{T,R}(r) = 1 + \sum_{c \in child(r)} W_{T,R}(c)$ if $r \notin R$

$R = \{R_2, R_6\}$



BioQuery-ASP



Representing Explanations in Natural Language

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

Answer: `ADRB1`

Shortest Explanation:

what_be_genes("ADRB1") ← drug_gene("Epinephrine","ADRB1"), gene_gene("ADRB1","DLG4")

drug_gene("Epinephrine","ADRB1") ←
drug_gene_ctd("Epinephrine","ADRB1")

drug_gene_ctd("Epinephrine","ADRB1") ←

gene_gene("ADRB1","DLG4") ←
gene_gene("DLG4,ADRB1")

gene_gene("DLG4,ADRB1") ← gene_gene_biogrid("DLG4","ADRB1")

gene_gene_biogrid("DLG4","ADRB1") ←

Explanation in Natural Language:

```
The drug Epinephrine targets the gene ADRB1 according to CTD.
The gene DLG4 interacts with the gene ADRB1 according to BioGrid.
```

Related Work

- Debugging in ASP: (Brain and DeVos 2005, Syrjanen 2006, Gebser et al. 2008, Oetsch et al. 2010)
- Generating justifications: (Pontelli et al. 2009, Schulz and Toni 2016, Cabalar et al. 2014).

Theorem 3

For every offline justification of an atom p, there is an explanation of p. For every explanation of an atom p, there is an offline justification of p in the reduct of the program with respect to given answer set.

Erdem, Oztok: Generating explanations for biomedical queries. TPLP 2015.

Explanation Generation in Three Applications of ASP

- Generating explanations for complex biomedical queries
- Explanation generation for multi-agent path finding
- Explainable robotic plan execution monitoring

Multi-Agent Path Finding Problems

- Multi-Agent Path Finding (MAPF) Problem
  - Finding a plan for each agent in an environment, without collisions
  - Constraints on plan length
- Optimization Variants
  - Minimizing maximum makespan, total plan length
- MAPF and its optimization variants are intractable (Ratner and Warmuth, 1986).
- Robotics, video games, autonomous aircraft towing vehicles, traffic control, autonomous warehouse systems.

Flexible Frameworks for MAPF

- A flexible AI method developed to solve a problem can accommodate variations of the problem.
- Some of our studies focus on finding a general framework that is flexible for variations of MAPF:
  - Multi-Agent Path Finding (MAPF)
  - Dynamic Multi-Agent Path Finding (D-MAPF)
  - Multi-Modal Multi-Agent Path Finding with Optimal Resource Utilization (mMAPF)

Erdem et al.: A general formal framework for pathfinding problems with multiple agents. AAAI 2013.
Bogatarkan et al.: A declarative method for dynamic MAPF. GCAI 2019.

Bogatarkan et al.: Multi-modal MAPF with optimal resource utilization. AMP 2020.

Explainable Frameworks for MAPF

- An explainable AI method can provide answers to queries about the (in)feasibility and the optimality of solutions.
- We introduce a method for generating explanations for queries regarding
  - feasibility and optimality of solutions
  - nonexistence of solutions
  - observations about solutions

  for a general variation of MAPF.

Multi-Modal Multi-Agent Path Finding with Optimal Resource Utilization (mMAPF)

mMAPF is a variant of MAPF that takes into account realistic conditions in the warehouses that are not addressed by MAPF:

- *Resources:* Battery levels of the robots decrease while moving and may need charging.
- *Multi-Modality:* Robots may need to move slowly due to tight passages or humans.
- *Waypoints:* Robots may need to visit several locations along their paths.
- *Optimizations:* Makespan, sum of costs, number of charging batteries

Bogatarkan et al.: Multi-modal MAPF with optimal resource utilization. AMP 2020.

Query-Based Explanation Generation Using Counterfactuals

- We consider queries about optimality of solutions, as well as observations about these solutions.
- Explanations generated by means of queries...
  - *Why does an agent wait too long at a location?*
  - *Why does an agent take a longer path?*
  - *Why does an agent charge many times?*

  ...using counterfactuals:
  - *What would happen, if the agent waited shorter?*
  - *What would happen, if the agent took a shorter path?*
  - *What would happen, if the agent charged less?*
- Together with the explanations, some recommendations are also given.

  "There is a better solution with shorter plan length."

Query-Based Explanation Generation Using Violations

- We also consider the infeasibility of solutions.

  “Why does not the instance have a solution?”

- Explanations are generated by identifying violations of constraints.
- If a solution is infeasible, an explanation regarding infeasibility or nonoptimality is provided.

  “There is no solution because the robot collides with an obstacle.”

- Explanations may include suggestions.

  “We suggest removing the obstacle, if it is possible to change the infrastructure.”

Sorts of Queries for MMAPF

- Queries about waiting (QW1-QW4)

  Why does not Agent `a` wait at location `x` at time `s` for less than `n` steps?

- Queries about charging (QC1-QC4)

  Why does Agent `a` charge at location `x` (at any time)?

- Queries about traversals (QP1-QP5)

  Why does not Agent `a` have a plan whose length is less than `l`?

- Query about nonexistence of a solution (QU)

  Why does not the instance have a solution?

Our Query-Based Explanation Generation Algorithm

Input : a **mMAPF** instance, a plan for this instance, and a query q of type QW1–QU
Output : An explanation
// Suppose that Π denotes the **mMAPF** program, possibly augmented with some hard constraints due to previous queries

queries

if *query q is of type QW1–QP5* **then**
  $\Pi_h \leftarrow$ Add the relevant hard constraint for q to the **mMAPF** program Π
  if Π_h *has an answer set* X **then**
    Display an explanation, presenting an alternative (better/worse) solution
  end
  else
    $\Pi_w \leftarrow$ Replace the **mMAPF** constraints in Π_h relevant for q, with the corresponding rules and weak constraints
    $Y \leftarrow$ Compute an answer set for Π_w
    Display an explanation based on violations
  end
end
else
  // query q is of type QU
  $\Pi_w \leftarrow$ Replace the **mMAPF** constraints in Π with the corresponding rules and weak constraints
  $Y \leftarrow$ Compute an answer set for Π_w
  Display an explanation based on violations
end

Our Query-Based Explanation Generation Algorithm

```
Input   : a mMAPF instance, a plan for this instance, and a query q
          of type QW1–QU
Output : An explanation
// Suppose that Π denotes the mMAPF program, possibly augmented
   with some hard constraints due to previous queries
if query q is of type QW1–QP5 then
    Π_h ← Add the relevant hard constraint for q to the mMAPF
      program Π
    if Π_h has an answer set X then
        Display an explanation, presenting an alternative
          (better/worse) solution
    end
    else
        Π_w ← Replace the mMAPF constraints in Π_h relevant for q,
          with the corresponding rules and weak constraints
        Y ← Compute an answer set for Π_w
        Display an explanation based on violations
    end
end
else
    // query q is of type QU
    Π_w ← Replace the mMAPF constraints in Π with the
      corresponding rules and weak constraints
    Y ← Compute an answer set for Π_w
    Display an explanation based on violations
end
```

counterfactuals

Queries and Relevant Hard Constraints: Counterfactuals

QC1 Why does Agent `a` charge at location `x` (at any time)?

What would happen if Agent `a` does not charge at location `x`?

```
:- batteryLevel(a,T+1,b), plan(a,T,x), charging(x).
```

QP1 Why does not Agent `a` have a plan whose length is less than `l`?

What would happen if Agent `a` has a plan with length at least `l`?

```
:- planLength(a,L), L>=l.
```

Our Query-Based Explanation Generation Algorithm

Input : a **mMAPF** instance, a plan for this instance, and a query q of type QW1–QU
Output : An explanation
// Suppose that Π denotes the **mMAPF** program, possibly augmented with some hard constraints due to previous queries
if *query q is of type QW1–QP5* **then**
  $\Pi_h \leftarrow$ Add the relevant hard constraint for q to the **mMAPF** program Π
  if Π_h *has an answer set* X **then**
    Display an explanation, presenting an alternative (better/worse) solution
  end
  else
    $\Pi_w \leftarrow$ Replace the **mMAPF** constraints in Π_h relevant for q, with the corresponding rules and weak constraints
    $Y \leftarrow$ Compute an answer set for Π_w
    Display an explanation based on violations
  end
end
else
  // query q is of type QU
  $\Pi_w \leftarrow$ Replace the **mMAPF** constraints in Π with the corresponding rules and weak constraints
  $Y \leftarrow$ Compute an answer set for Π_w
  Display an explanation based on violations
end

suggestions

Queries and Alternative Solutions: Suggestions

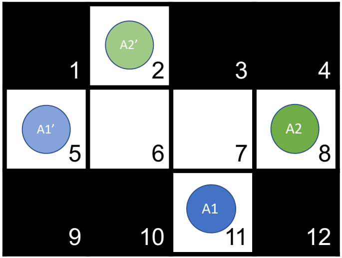
QC1 Why does Agent `a` charge at location `x` (at any time)?

Actually, Agent `a` *does not have to charge at location* `x` *(at any time). Here is an alternative plan: ...*

QP1 Why does not Agent `a` have a plan whose length is less than `l`?

Actually, Agent `a` *can follow a shorter path whose length is smaller than* `l`*. Here is an alternative plan: ...*

An Example Scenario for QW1



Time	A1 Location	A2 Location
0	11	8
1	7	8
2	6	7
3	5	6
4	-	2

1. User asks the query of type QW1:
   "Why does Robot 2 wait at Cell 8 (at any time)?"
2. The algorithm gives the following explanation:
   "Actually, Robot 2 does not have to wait at Cell 8 from time step 0 to 2. Here is an alternative optimal plan: ..."

Our Query-Based Explanation Generation Algorithm

Input : a **mMAPF** instance, a plan for this instance, and a query q of type QW1–QU
Output : An explanation

// Suppose that Π denotes the **mMAPF** program, possibly augmented with some hard constraints due to previous queries

if *query q is of type QW1–QP5* **then**
    $\Pi_h \leftarrow$ Add the relevant hard constraint for q to the **mMAPF** program Π
    if Π_h *has an answer set* X **then**
        Display an explanation, presenting an alternative (better/worse) solution
    end
    else
        $\Pi_w \leftarrow$ Replace the **mMAPF** constraints in Π_h relevant for q, with the corresponding rules and weak constraints
        $Y \leftarrow$ Compute an answer set for Π_w
        Display an explanation based on violations
    end
end
else
    // query q is of type QU
    $\Pi_w \leftarrow$ Replace the **mMAPF** constraints in Π with the corresponding rules and weak constraints
    $Y \leftarrow$ Compute an answer set for Π_w
    Display an explanation based on violations
end

violations

violations

Nonexistence of Solutions

QU Why does not the instance have a solution?

- We replace the relevant MMAPF constraints by violation definitions and weak constraints.
- We identify which constraints are violated.

Violation of Collision Constraint

The collision constraint:

```
:- plan(A1,T,X), plan(A2,T,X), agent(A1;A2), A1<A2,
X!=intransit.
```

Violation of Collision Constraint

The collision constraint:

```
:- plan(A1,T,X), plan(A2,T,X), agent(A1;A2), A1<A2,
X!=intransit.
```

is replaced by the definition:

```
violate_collision(A1,A2,T,X) :- plan(A1,T,X), plan(A2,T,X),
agent(A1;A2), A1<A2, X!=intransit.
```

Violation of Collision Constraint

The collision constraint:

```
:- plan(A1,T,X), plan(A2,T,X), agent(A1;A2), A1<A2,
X!=intransit.
```

is replaced by the definition:

```
violate_collision(A1,A2,T,X) :- plan(A1,T,X), plan(A2,T,X),
agent(A1;A2), A1<A2, X!=intransit.
```

and the weak constraint:

```
:~ violate_collision(A1,A2,T,X). [1@7, A1,A2,T,X,vc]
```

Violation of Collision Constraint

The collision constraint:

```
:- plan(A1,T,X), plan(A2,T,X), agent(A1;A2), A1<A2,
X!=intransit.
```

is replaced by the definition:

```
violate_collision(A1,A2,T,X) :- plan(A1,T,X), plan(A2,T,X),
agent(A1;A2), A1<A2, X!=intransit.
```

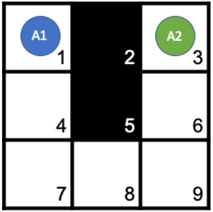
and the weak constraint:

```
:~ violate_collision(A1,A2,T,X). [1@7, A1,A2,T,X,vc]
```

An example explanation generated about violation of this constraint:

"Robot 1 has to wait at Cell 11; otherwise, Robot 1 and Robot 2 will collide with each other at Cell 7."

An Example Scenario for QU



1. User asks the query QU:

   "Why does not the instance have a solution?"

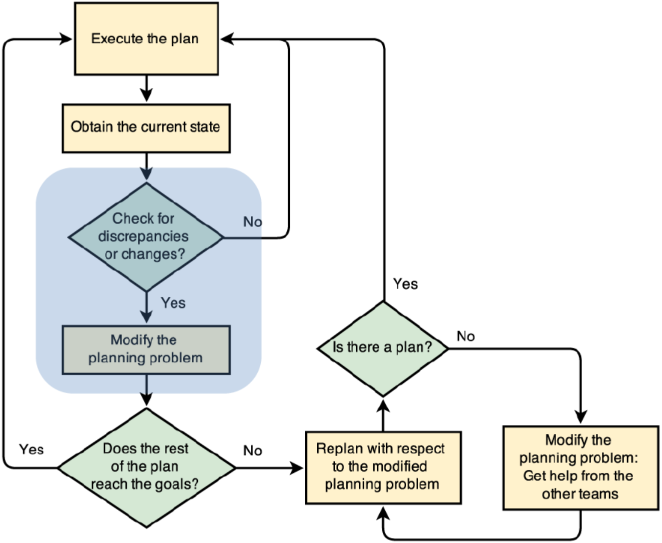
2. The algorithm displays the following explanation:

   "There is no solution because Robot 2 collides with the obstacle at Cell 2 at time step 1. We suggest removing this obstacle, if infrastructure change is allowed."

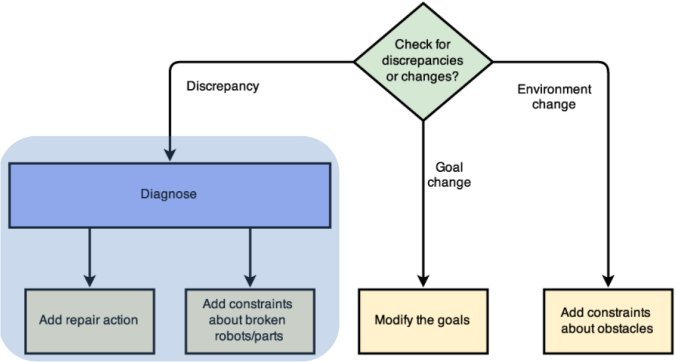
Explanation Generation in Three Applications of ASP

- Generating explanations for complex biomedical queries
- Explanation generation for multi-agent path finding
- Explainable robotic plan execution monitoring

Plan Execution Monitoring



Causal Replanning



Hybrid Diagnostic Reasoning

Description of robotic actions and change

Plan execution so far from the initial state

Observations about current state

Hypothesis about broken robots or components

$$\underbrace{D_{diag} \cup S_0^M \cup P_t \cup O_t^M \cup H}$$

Is the logical theory consistent?

If it is consistent then which actions in P_t could not be executed and why not?

Erdem et al.: Integrating hybrid diagnostic reasoning in plan execution monitoring for cognitive factories with multiple robots. IEEE ICRA 2015. (Best Paper Award Finalist)

Explanation Generation based on Hybrid Diagnosis

While diagnoses may explain the reasons of relevant discrepancies in terms of broken robotic components, further explanations can be generated to include the actions whose effects have not been observed as expected due to these diagnoses.

Example: A most-probable min-cardinality diagnosis for a relevant discrepancy detected at time step $t{=}3$:

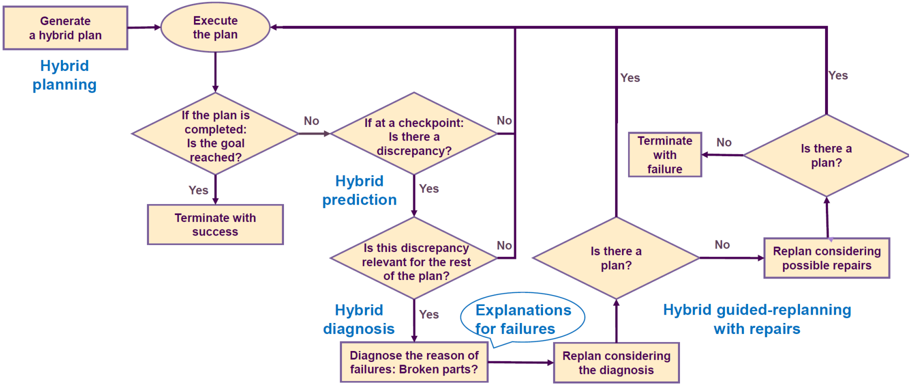
$$X_3 = \{(R_1, Base, 1)\}$$

with further explanations:

> *Base of R_1 got broken at time step $t{=}1$; so R_1 could not move to the left side of the table at time step $t{=}1$ as expected.*

Coruhlu et al.: Explainable Robotic Plan Execution Monitoring Under Partial Observability. IEEE Transactions on Robotics 2021.

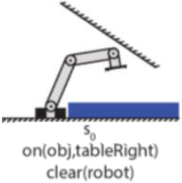
Explainable Robotic Plan Execution Monitoring under Partial Observability



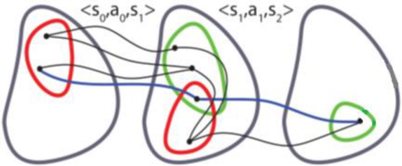
Hybrid Planning: Problem

How can a robot reach a goal state from its initial state in at most k steps without any collisions?

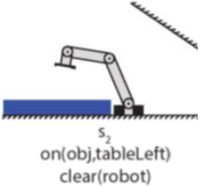
Initial state



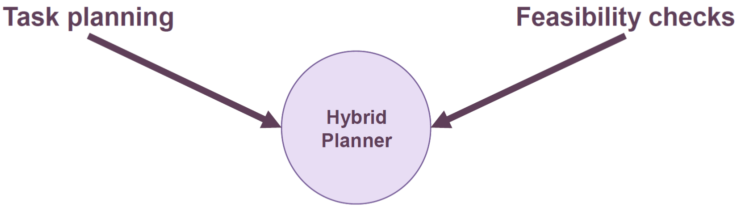
<pick(robot,obj), place(robot,tableLeft)>



Goal state



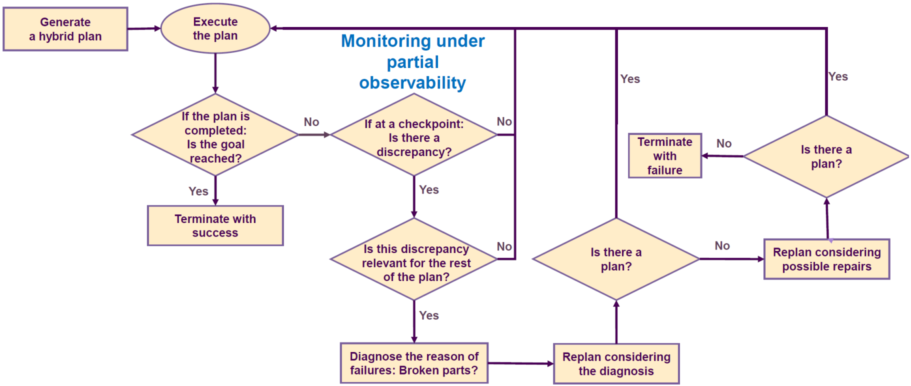
Hybrid Planning: Method



Hybrid plan

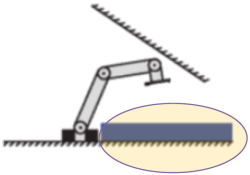
A task plan whose execution is feasible with respect to the feasibility checks.

Explainable Robotic Plan Execution Monitoring under Partial Observability

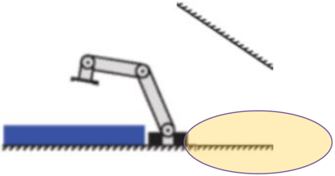


Partial Observability

Only the right side of the table can be observed during plan execution.

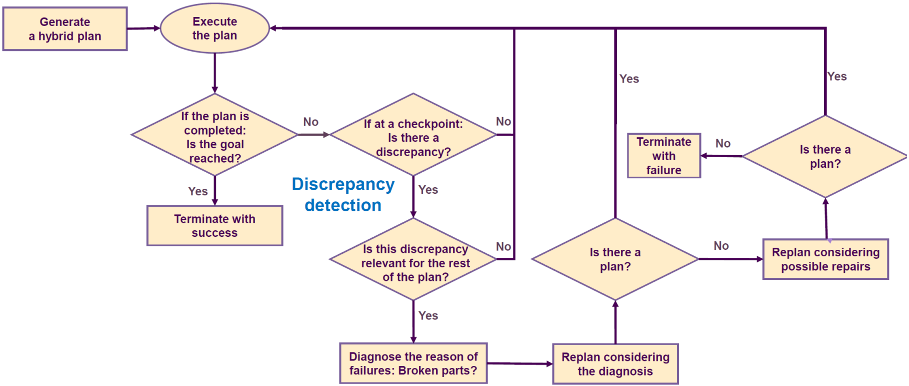


Observed: on(obj,tableRight)



Observed:

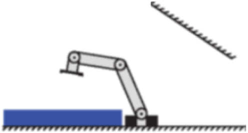
Explainable Robotic Plan Execution Monitoring under Partial Observability



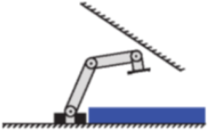
Discrepancy Detection

Is there a discrepancy between the partially observed state and the expected state?

YES

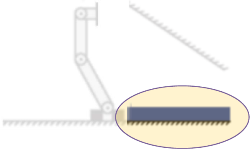


Expected state:
on(obj,tableLeft),
clear(robot)



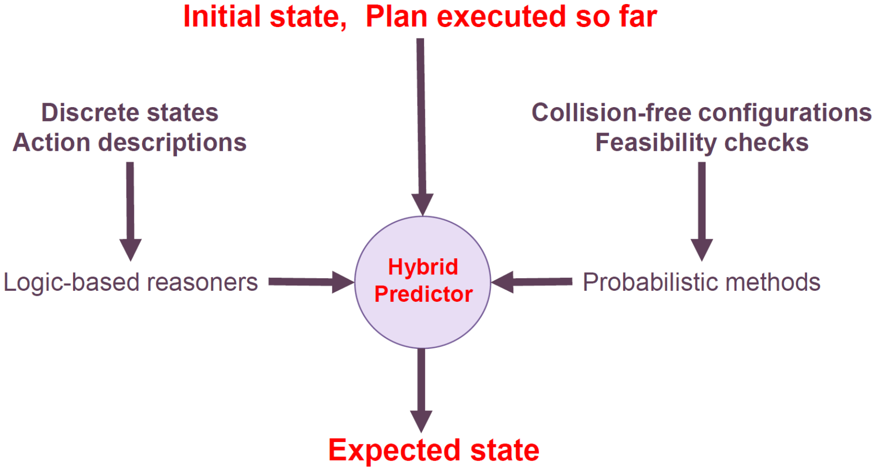
Initial state:
on(obj,tableRight),
clear(robot)

<pick(robot,obj), place(robot,tableLeft>

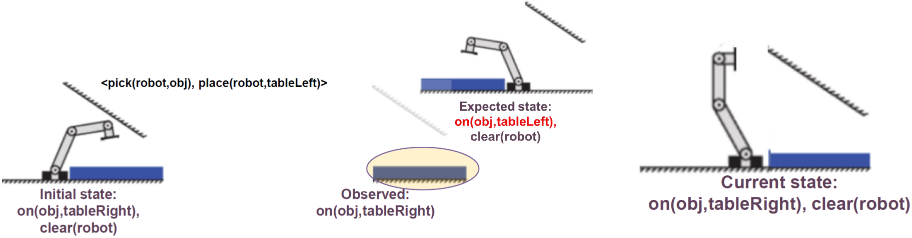


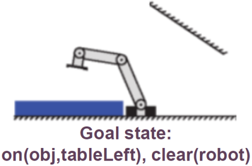
Observed:
on(obj,tableRight)

Hybrid Prediction – Expected State



Discrepancy Relevancy

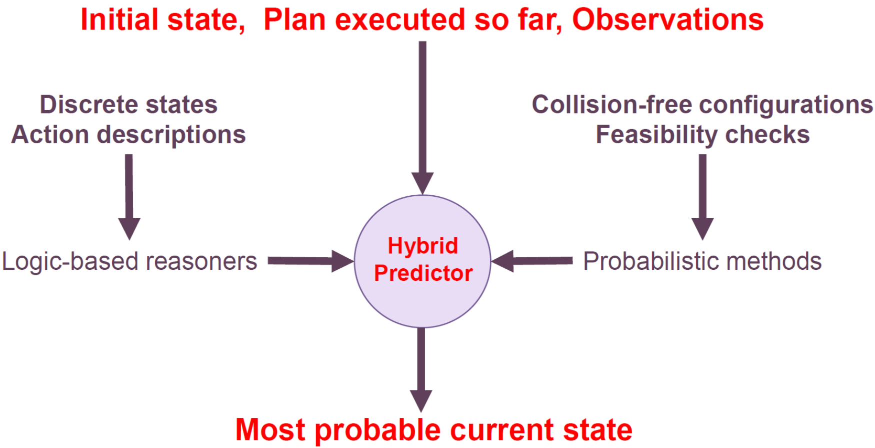




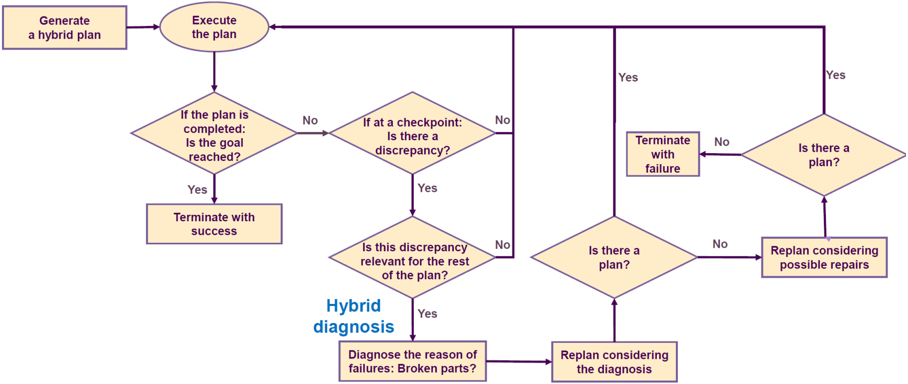
Is this discrepancy relevant for the rest of the plan to reach the goal from the current state?

YES

Hybrid Prediction – Current State

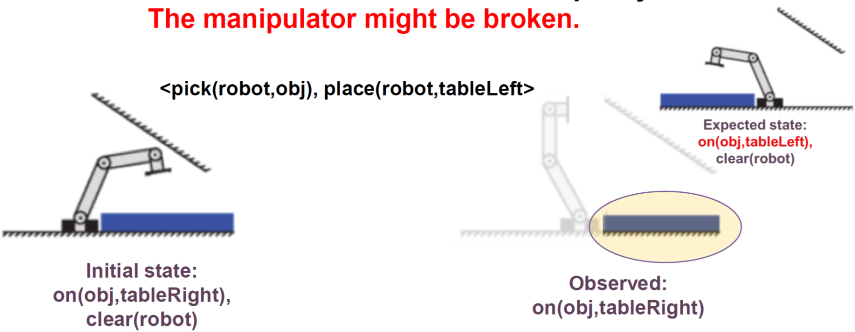


Explainable Robotic Plan Execution Monitoring under Partial Observability



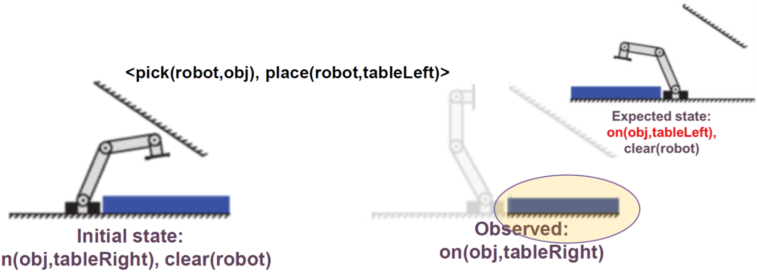
Diagnosis

What is the cause of this relevant discrepancy?
The manipulator might be broken.

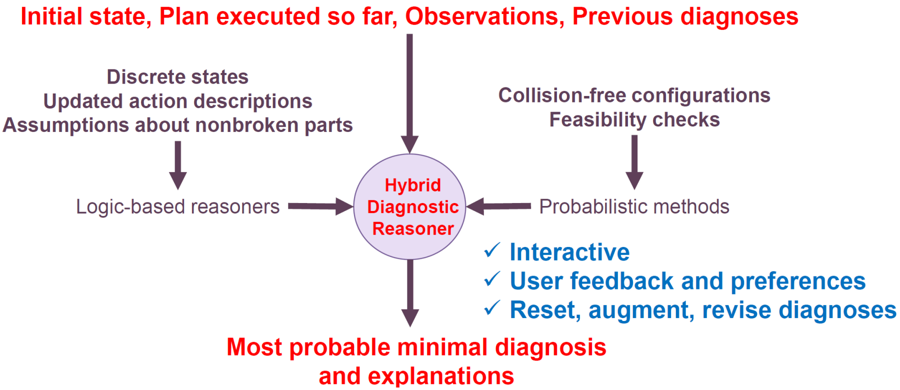


Explanations for Failures

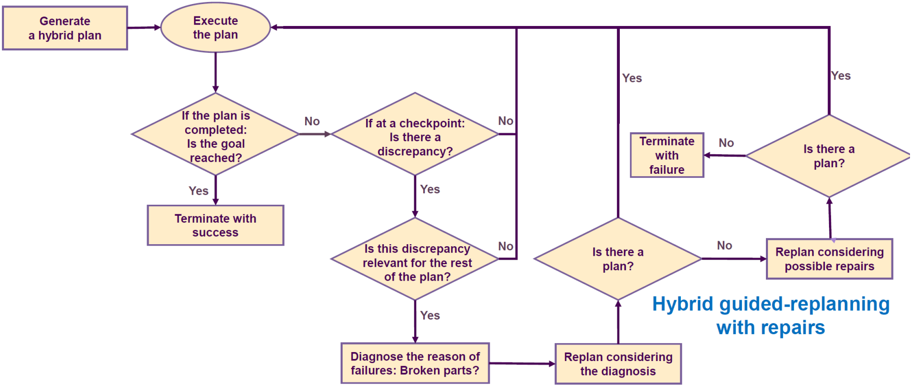
Please explain further.
What failed during plan execution to cause this relevant discrepancy?
As the manipulator is broken,
the robot could not pick the object at time step 0.



Hybrid Diagnosis and Explanations



Explainable Robotic Plan Execution Monitoring under Partial Observability

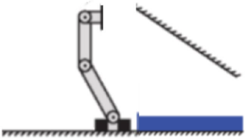


Hybrid guided-replanning with repairs

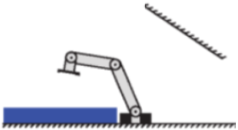
Guided Re-Planning with Repairs

Given that the manipulator is broken,
how can the robot reach the goal from the current state in at most 3 steps?

<repair(robot),pick(robot,obj),place(robot,tableLeft)>

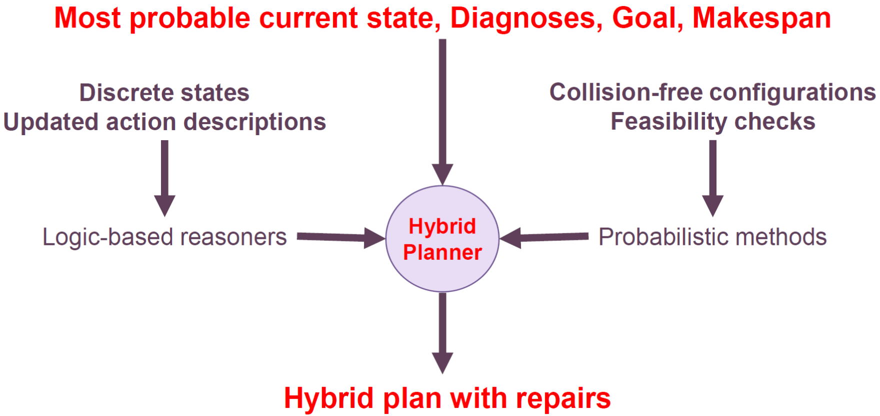


Current state:
on(obj,tableRight), clear(robot)



Goal state:
on(obj,tableLeft), clear(robot)

Hybrid Guided Re-Planning with Repairs



Does diagnostic reasoning improve execution monitoring?

Experiments 25 instances with 2 bimanual robots and 10 objects in kitchen domain.

	without diagnosis			with diagnosis		
# Broken parts	**# Replannings**	Total plan length	Success rate	**# Replannings**	Total plan length	Success rate
2	**8.48**	26.44	%80	**3.52**	17.76	%92
3	**14.12**	35.96	%72	**4.04**	22.48	%80

✓ **Less number of replannings**

Does diagnostic reasoning improve execution monitoring?

Experiments 25 instances with 2 bimanual robots and 10 objects in kitchen domain.

	without diagnosis			with diagnosis		
# Broken parts	# Replannings	**Total plan length**	Success rate	# Replannings	**Total plan length**	Success rate
2	8.48	**26.44**	%80	3.52	**17.76**	%92
3	14.12	**35.96**	%72	4.04	**22.48**	%80

✓ **Less number of replannings**
✓ **Shorter plans**

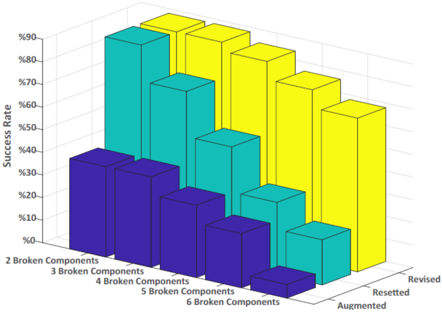
Does diagnostic reasoning improve execution monitoring?

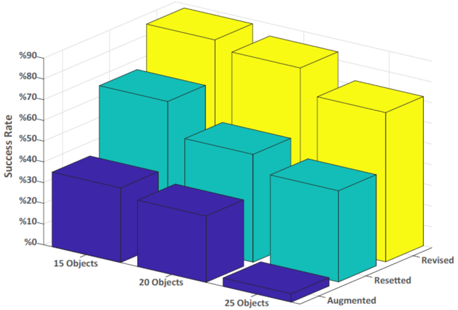
Experiments 25 instances with 2 bimanual robots and 10 objects in kitchen domain.

	without diagnosis			with diagnosis		
# Broken parts	# Replannings	Total plan length	**Success rate**	# Replannings	Total plan length	**Success rate**
2	8.48	26.44	**%80**	3.52	17.76	**%92**
3	14.12	35.96	**%72**	4.04	22.48	**%80**

✓ **Less number of replannings**
✓ **Shorter plans**
✓ **Higher success rate**

How does resetting, augmenting, or revising diagnosis improve execution monitoring?





✓ **Revising diagnoses improves the success rate.**

Conclusions

- Generating explanations for complex biomedical queries
  - Knowledge intensive application, utilizing different resources
  - Query guided explanation generation, based on justifications
  - Explanation trees, shortest explanations, diverse explanations
  - Explanations, augmented with provenance information

Conclusions

- Generating explanations for complex biomedical queries
  - Knowledge intensive application, utilizing different resources
  - Query guided explanation generation, based on justifications
  - Explanation trees, shortest explanations, diverse explanations
  - Explanations, augmented with provenance information
- Explanation generation for multi-agent path finding
  - Combinatorial search problem with many variants
  - Query guided explanation generation, based on hypothetical reasoning, counterfactuals, hard/weak constraints
  - Explanations about properties of solutions (e.g., optimality), augmented with alternative solutions
  - Explanations about nonexistence solutions, augmented with suggestions

Conclusions

- Generating explanations for complex biomedical queries
  - Knowledge intensive application, utilizing different resources
  - Query guided explanation generation, based on justifications
  - Explanation trees, shortest explanations, diverse explanations
  - Explanations, augmented with provenance information
- Explanation generation for multi-agent path finding
  - Combinatorial search problem with many variants
  - Query guided explanation generation, based on hypothetical reasoning, counterfactuals, hard/weak constraints
  - Explanations about properties of solutions (e.g., optimality), augmented with alternative solutions
  - Explanations about nonexistence solutions, augmented with suggestions
- Explainable robotic plan execution monitoring
  - Hybrid reasoning about actions and change
  - Interactive explanations based on diagnostic reasoning
  - Most probable explanations about broken components, augmented with actions