# A General Logic-based Approach for Explanation Generation


## Abstract

In an explanation generation problem, an agent needs to identify and explain the reasons for its decisions to another agent. Existing work in this area is mostly confined to planning-based systems that use automated planning approaches to solve the problem. In this paper, we approach this problem from a new perspective, where we propose a general logic-based framework for explanation generation. In particular, given a knowledge base $KB_1$ that entails a formula $\phi$ and a second knowledge base $KB_2$ that does not entail $\phi$, we seek to identify an explanation $\epsilon$ that is a subset of $KB_1$ such that the union of $KB_2$ and $\epsilon$ entails $\phi$. We define two types of explanations, model- and proof-theoretic explanations, and use cost functions to reflect preferences between explanations. Further, we present our algorithm implemented for propositional logic that compute such explanations and empirically evaluate it in random knowledge bases and a planning domain.


## Introduction

With increasing proliferation and integration of AI systems in our daily life, there is a surge of interest in *explainable AI*, which includes the development of AI systems whose actions can be easily understood by humans. Driven by this goal, *machine learning* (ML) researchers have begun to classify commonly used ML algorithms according to different dimensions of explainability (Guidotti *et al.* 2018); improved the explainability of existing ML algorithms (Vaughan *et al.* 2018; Alvarez Melis and Jaakkola 2018; Petkovic *et al.* 2018); as well as proposed new ML algorithms that trade off accuracy for increasing explainability (Dong *et al.* 2017; Gilpin *et al.* 2018).[1]


Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.


[1]While the term *interpretability* is more commonly used in the ML literature and can be used interchangeably with *explainability*, we use the latter term as it is more commonly used broadly across different subareas of AI.

In contrast, researchers in the *automated planning* community have mostly taken a complementary approach. While there is some work on adapting planning algorithms to find easily explainable plans[2] (i.e., plans that are easily understood and accepted by a human user) (Zhang *et al.* 2017), most work has focused on the *explanation generation problem* (i.e., the problem of identifying explanations of plans found by planning agents that when presented to users, will allow them to understand and accept the proposed plan) (Langley 2016; Kambhampati 1990). Within this context, researchers have tackled the problem where the model of the human user may be (1) inconsistent with the model of the planning agent (Chakraborti *et al.* 2017); (2) must be learned (Zhang *et al.* 2017); and (3) a different form or abstraction than that of the planning agent (Sreedharan *et al.* 2018; Tian *et al.* 2016). However, a common thread across most of these works is that they, not surprisingly, employ mostly automated planning approaches. For example, they often assume that the models of both the agent and human are encoded in PDDL format.

In this paper, we approach the explanation generation problem from a different perspective – one based on *knowledge representation and reasoning* (KR). We propose a general logic-based framework for explanation generation, where given a knowledge base $KB_1$ (of an agent) that entails a formula $\phi$ and a knowledge base $KB_2$ (of a human user) that does not entail $\phi$, the goal is to identify an explanation $\epsilon \subseteq KB_1$ such that $KB_2 \cup \epsilon$ entails $\phi$. We define two types of explanations, model- and proof-theoretic explanations, and use cost functions to reflect preferences between explanations. Further, we present an algorithm, implemented for propositional logic, that computes such explanations and evaluate its performance experimentally in random knowledge bases as well as in a planning domain.

In addition to providing an alternative approach to

[2]Also called *explicable* plans in the planning literature.

solve the same explanation generation problem tackled thus far by the automated planning community, our approach has the merit of being more generalizable to other problems beyond planning problems as long as they can be modeled using a logical KR language.

## Preliminaries

A *logic* $L$ is a tuple $(KB_L, BS_L, ACC_L)$ where $KB_L$ is the set of well-formed knowledge bases (or theories) of $L$ – each being a set of formulae. $BS_L$ is the set of possible belief sets; each element of $BS_L$ is a set of syntactic elements representing the beliefs $L$ may adopt. $ACC_L : KB_L \rightarrow 2^{BS_L}$ describes the *"semantics"* of $L$ by assigning to each element of $KB_L$ a set of acceptable sets of beliefs. For each $KB \in KB_L$ and $B \in ACC_L(KB)$, we say that $B$ is a *model* of $KB$. A logic is monotonic if $KB \subseteq KB'$ implies $ACC_L(KB') \subseteq ACC_L(KB)$.

**Example 1** *Assume that $L$ refers to the propositional logic over an alphabet $P$. Then, $KB_L$ is the set of propositional theories over $P$, $BS_L = 2^P$, and $ACC_L$ maps each theory $KB$ into the set of its models in the usual sense.*

We say that a $KB$ is *consistent* if $ACC_L(KB) \neq \emptyset$. A formula $\varphi$ in the logic $L$ is *entailed* by $KB$, denoted by $KB \models_L \varphi$, if $ACC_L(KB) \neq \emptyset$ and $\varphi \in B$ for every $B \in ACC_L(KB)$.

For our later use, we will assume that a negation operator $\neg$ over formulas exists; and $\varphi$ and $\neg\varphi$ are contradictory with each other in the sense that for any $KB$ and $B \in ACC_L(KB)$, if $\varphi \in B$ then $\neg\varphi \notin B$; and if $\neg\varphi \in B$ then $\varphi \notin B$. $\epsilon \subseteq KB$ is called a *sub-theory* of $KB$. A theory $KB$ *subsumes* a theory $KB'$, denoted by $KB \lhd KB'$ if $ACC_L(KB) \subset ACC_L(KB')$.

Conclusions of a knowledge base can also be derived using rules. A rule system $\Sigma_L$ of a logic $L$ is a set of rules of the form

$$\varphi_1, \ldots, \varphi_k \vdash_L \varphi_0 \tag{1}$$

where $\varphi_i$ are formulas. The left hand side could be empty. For a rule $r$ of the form (1), $body(r)$ (resp. $head(r)$) denotes the left (resp. right) side of $r$. Intuitively, a rule $r$ states that if the body is true then the head is also true.

Given a knowledge base $KB$ and a rule system $\Sigma_L$, we say $KB \vdash_{\Sigma_L} \varphi$ if either $\varphi \in KB$ or there exists a sequence of rules $r_1, \ldots, r_n$ in $\Sigma_L$ such that $body(r_1) \subseteq KB$, $head(r_n) = \varphi$, $head(r_i) \in KB$ for $i = 1, \ldots, n-1$, and $body(r_i) \subseteq KB$ or $body(r_i) \subseteq body(r_1) \cup \{head(r_j) \mid j = 1, \ldots, i-1\}$ for every $i = 2, \ldots, n$. We call the sequence $\epsilon = \langle r_1; \ldots; r_n \rangle$ as a *proof* from $KB$ for $\varphi$ w.r.t. $\Sigma_L$ and say that the proof $\epsilon$ has the length $n$.

$\Sigma_L$ is said to be *sound* if for every $\varphi$, $KB \vdash_{\Sigma_L} \varphi$ implies $KB \models_L \varphi$. It is *complete* if for every $\varphi$, $KB \models_L \varphi$ implies $KB \vdash_{\Sigma_L} \varphi$.

In this paper, we will mostly use examples from propositional logic. We make use of the fact that the resolution rule is sound and complete in first-order logic (Robinson 1965), and hence, in propositional logic. This allows us to utilize the well-known DPLL algorithm (Davis *et al.* 1962) in computing proofs for a formula given a knowledge base.

## Two Accounts of Explanations

In this section, we introduce the notion of an explanation in the following setting:

> **Explanation Generation Problem:** Given two knowledge bases $KB_1$ and $KB_2$ and a formula $\varphi$ in a logic $L$. Assume that $KB_1 \models_L \varphi$ and $KB_2 \not\models_L \varphi$. The goal is to identify an explanation (i.e., a set of formulas) $\epsilon \subseteq KB_1$ such that $KB_2 \cup \epsilon \models \varphi$.

We first define the notion of a support of a formula w.r.t. a knowledge base.

**Definition 1 (Support)** *Assume that $KB \models_L \varphi$. We say that $\epsilon \subseteq KB$ is a* support *of $\varphi$ w.r.t. $KB$ if $\epsilon \models_L \varphi$. Assume that $\epsilon$ is a support of $\varphi$ w.r.t. $KB$. We say that $\epsilon \subseteq KB$ is a* $\subseteq$-minimal support *of $\varphi$ if no proper subtheory of $\epsilon$ is a support of $\varphi$. Furthermore, $\epsilon$ is a* $\lhd$-general support *of $\varphi$ if there is no support $\epsilon'$ of $\varphi$ w.r.t. $KB$ such that $\epsilon$ subsumes $\epsilon'$.*

We now define below two types of explanations – model-theoretic and proof-theoretic explanations.

### Model-Theoretic Explanations

**Definition 2 ($m$-Explanation)** *Given two knowledge bases $KB_1$ and $KB_2$ in logic $L$ and a formula $\varphi$. Assume that $KB_1 \models_L \varphi$ and $KB_2 \not\models_L \varphi$.*

*A* model-theoretic explanation (or $m$-explanation) for $\varphi$ from $KB_1$ for $KB_2$ *is a support $\epsilon$ w.r.t. $KB_1$ for $\varphi$ such that $KB_2 \cup \epsilon \models_L \varphi$.*

**Example 2** *Consider proposition logic theories over the set of propositions $\{a, b, c\}$ with the usual definition of models, satisfaction, etc. Assume $KB_1 = \{a, b, a \rightarrow c, a \wedge b \rightarrow c\}$ and $KB_2 = \{a\}$. We have that $\epsilon_1 = \{a, a \rightarrow c\}$ and $\epsilon_2 = \{a, b, a \wedge b \rightarrow c\}$ are two $\subseteq$-minimal supports of $c$ w.r.t. $KB_1$. Only $\epsilon_1$ is a $\lhd$-general support of $c$ w.r.t. $KB_1$ since $\epsilon_2 \lhd \epsilon_1$.*

*Both $\epsilon_1$ and $\epsilon_2$ can serve as $m$-explanations for $c$ from $KB_1$ for $KB_2$. Of course, $KB_1$ is itself an $m$-explanation for $c$ from $KB_1$ for $KB_2$.*

*Consider $KB_3 = \{a, \neg b\}$. In this case, we have that only $\epsilon_1$ is an $m$-explanation for $c$ from $KB_1$ for $KB_3$.*

*Now, consider $KB_4 = \{\neg a\}$. In this case, we have no $m$-explanation for $c$ from $KB_1$ for $KB_4$.*

**Proposition 1** *For two knowledge bases $KB_1$ and $KB_2$ in a monotonic logic $L$, if $KB_1 \models_L \varphi$ and $KB_2 \models_L \neg\varphi$, then there exists no $m$-explanation for $\varphi$ from $KB_1$ for $KB_2$.*

The $KB_4$ in Example 2 and Proposition 1 show that $m$-explanations alone might be insufficient. Sometimes, we also need to persuade the other agent that its knowledge base is not correct. We leave this for the future. In this paper, we assume that $KB_2 \not\models_L \neg\varphi$ and $KB_2 \not\models_L \varphi$ and, thus, an $m$-explanation always exists.

### Proof-Theoretic Explanations

**Definition 3 ($p$-explanation)** *Given a logic $K$ with a sound and complete rule system $\Sigma_L$ and two knowledge bases $KB_1$ and $KB_2$ in logic $L$ and a formula $\varphi$. Assume that $KB_1 \vdash_L \varphi$ and $KB_2 \not\vdash_L \varphi$.*

*A* proof-theoretic explanation (or $p$-explanation) for $\varphi$ from $KB_1$ for $KB_2$ *is a proof $\langle r_1; \ldots; r_n\rangle$ from $KB_1$ for $\varphi$ such that $KB_2 \cup (\bigcup_{i=1}^{n} body(r_i) \cap KB_1) \vdash_{\Sigma_L} \varphi$ and $KB_2 \cup (\bigcup_{i=1}^{n} body(r_i) \cap KB_1)$ is consistent.*

**Example 3** *Consider the theories $KB_1 = \{a, b, a \rightarrow c, a \wedge b \rightarrow c\}$ and $KB_2 = \{a\}$ from Example 2. Let us assume that $\Sigma_L$ is the set of rules of the form $l \vdash_L l$ and $l, \neg l \vee p \vdash p$ for any literals $l, p$ in the language of $KB_1$ and $KB_2$. Then, $\langle a, \neg a \vee c \vdash_L c\rangle$ is a proof from $KB_1$ for c, which is also a p-explanation for $\varphi$ from $KB_1$ for $KB_2$.*

*Likewise, $\langle a \vdash_L a; b \vdash_L b; a, \neg a \vee \neg b \vee c \vdash_L \neg b \vee c; b, \neg b \vee c \vdash_L c\rangle$ is a p-explanation for c from $KB_1$ for $KB_2$.*

**Proposition 2** *Assume that $\Sigma_L$ is a sound and complete rule system of a logic L, $KB_1$ is a knowledge base, and $\varphi$ is a formula in L. For each proof $\langle r_1; \ldots; r_n\rangle$ from $KB_1$ for $\varphi$ w.r.t. $\Sigma_L$, $\bigcup_{i=1}^{n} body(r_i) \cap KB_1$ is a support of $\varphi$ w.r.t. $KB_1$.*

Proposition 2 implies that each proof from $KB_1$ for $\varphi$ could be identified as a $p$-explanation for $\varphi$ from $KB_1$ if $\Sigma_L$ is sound and complete. This provides the following relationship between $m$-explanations and $p$-explanations.

**Proposition 3** *Assume that $\Sigma_L$ is a sound and complete rule system of a logic L, $KB_1$ and $KB_2$ are two knowledge bases in L, and $\varphi$ is a formula in L. Then,*

- *for each $m$-explanation $\epsilon$ for $\varphi$ from $KB_1$ for $KB_2$, there exists a p-explanation $\langle r_1; \ldots; r_n\rangle$ for $\varphi$ from $KB_1$ for $KB_2$ such that $\bigcup_{i=1}^{n} body(r_i) \cap KB_1 \subseteq \epsilon$; and*
- *for each p-explanation $\langle r_1; \ldots; r_n\rangle$ for $\varphi$ from $KB_1$ for $KB_2$, $\bigcup_{i=1}^{n} body(r_i) \cap KB_1$ is an $m$-explanation for $\varphi$ from $KB_1$ for $KB_2$.*

## Preferred Explanations

Given $KB_1$ and $KB_2$ and a formula $\varphi$, there might be several ($m$- or $p$-) explanations for $\varphi$ from $KB_1$ for $KB_2$. For brevity, we will now use the term $x$-explanation for $x \in \{m, p\}$ to refer to an $x$-explanation for $\varphi$ from $KB_1$ for $KB_2$. Obviously, not all explanations are equal. One might preferred a subset minimal $m$-explanation or a shortest length $p$-explanation over others. We will next define a general preferred relation among explanations.

We assume a cost function $\mathcal{C}_L^x$ that maps pairs of knowledge bases and sets of explanations to non-negative real values, i.e.,

$$\mathcal{C}_L^x : KB_L \times \Omega \rightarrow \mathcal{R}^{\geq 0} \tag{2}$$

where $\Omega$ is the set of $x$-explanations and $\mathcal{R}^{\geq 0}$ denotes the set of non-negative real numbers. Intuitively, this function can be used to characterize different complexity measurements of an explanation.

A cost function $\mathcal{C}_L^m$ is *monotonic* if for any two $m$-explanations $\epsilon_1 \subseteq \epsilon_2$, $\mathcal{C}_L^m(KB, \epsilon_1) \leq \mathcal{C}_L^m(KB, \epsilon_2)$. A cost function $\mathcal{C}_L^p$ is *monotonic* if for any two $p$-explanations $\epsilon_1$ and $\epsilon_2$ such that $\epsilon_1$ is a subsequence of $\epsilon_2$, $\mathcal{C}_L^p(KB, \epsilon_1) \leq \mathcal{C}_L^m(KB, \epsilon_2)$.

$\mathcal{C}_L^x$ induces a preference relation $\prec_{KB}$ over explanations as follows.

**Definition 4 (Preferred Explanation)** *Given a cost function $\mathcal{C}_L^x$, a knowledge base $KB_2$, and two $x$-explanations $\epsilon_1$ and $\epsilon_2$ for $KB_2$, we say $\epsilon_1$ is* preferred *over $\epsilon_2$ w.r.t. $KB_2$ (denoted by $\epsilon_1 \preceq^x_{KB_2} \epsilon_2$) iff*

$$\mathcal{C}_L^x(KB_2, \epsilon_1) \leq \mathcal{C}_L^x(KB_2, \epsilon_2) \tag{3}$$

*and $\epsilon_1$ is* strictly preferred *over $\epsilon_2$ w.r.t. $KB_2$ (denoted by $\epsilon_1 \prec^x_{KB_2} \epsilon_2$) if*

$$\mathcal{C}_L^x(KB_2, \epsilon_1) < \mathcal{C}_L^x(KB_2, \epsilon_2) \tag{4}$$

This allows us to compare explanations as follows.

**Definition 5 (Most Preferred Explanation)** *Given a cost function $\mathcal{C}_L^x$ and a knowledge base $KB_2$, an explanation $\epsilon$ is a* most preferred $x$-explanation *w.r.t. $KB_2$ if there exists no other explanation $\epsilon'$ such that $\epsilon' \prec^x_{KB_2} \epsilon$.*

**Proposition 4** *If $\mathcal{C}_L^x$ is monotonic then the relation $\preceq^x_{KB_2}$ over $x$-explanations is transitive, anti-symmetric, and reflexive; and the relation $\prec^x_{KB_2}$ over $x$-explanations is transitive and anti-symmetric.*

There are several natural monotonic cost functions. Examples for cost functions for $m$-explanations include:

- $c_L^1(KB_2, \epsilon) = |\epsilon|$, the cardinality of $\epsilon$, indicates the number of formulas that need to be explained;
- $c_L^2(KB_2, \epsilon) = |\epsilon \setminus KB_2|$, the cardinality of $\epsilon \setminus KB_2$, indicates the number of *new* formulas that need to be explained;
- $c_L^3(KB_2, \epsilon) = |new_vars(KB_2, \epsilon)|$ indicates the number of *new* symbols occurring in $\epsilon$ that are not in $KB_2$ and need to be explained;
- $c_L^4(KB_2, \epsilon) = length(\epsilon)$ indicates the number of literals in $\epsilon$ that need to be explained.

**Algorithm 1:** genExp$(KB_1, KB_2, \varphi)$

**Input:** Logic $L$, formula $\varphi$, KBs $KB_1$ and $KB_2$, cost function $\mathcal{C}_L^x$
**Output:** A most preferred $x$-explanation w.r.t. $\mathcal{C}_L^x$ from $KB_1$ to $KB_2$ for $\varphi$; or $nil$

1 **if** $KB_1 \not\models_L \varphi$ *or* $KB_2 \models_L \varphi$ **then**
2  **return** $nil$
3 **if** $KB_1 \models_L \varphi$ *and* $KB_2 \not\models_L \neg\varphi$ **then**
4  $\epsilon = most_preferred(KB_1, KB_2, \varphi)$
5  **return** $\epsilon$

Naturally, some of these cost functions can also be combined (e.g., $c_L^2 + c_L^3$ will measure the number of new formulas and new symbols that must be explained).

Observe that the three functions $c_L^1$ and $c_L^4$ are independent from $KB_2$ while $c_L^2$ and $c_L^3$ depend on $KB_2$. A potential advantage of a cost function that is independent from $KB_2$ is that it helps simplify the computation of most preferred explanations.

**Example 4** *Continuing with Example 2, if we use $c_L^1$ as the cost function, then we have that $\epsilon_1 \prec_{KB_2}^m \epsilon_2 \prec_{KB_2}^m KB_1$. Furthermore, $\epsilon_1$ is the most preferred $m$-explanation from $KB_1$ to $KB_2$.*

## Computing Preferred Explanations

At a high level, Algorithms 1 and 2 can be used for computing most-preferred explanations given a formula $\varphi$ and two knowledge bases $KB_1$ and $KB_2$ of a logic $L$ with the cost function $\mathcal{C}_L^x$. We assume that when computing for $p$-explanations, a sound and complete rule system is available. Our algorithms rely on the existence of an algorithm for checking entailment between knowledge bases and formulas (Lines 1 and 3 in Algorithm 1 and Line 4 in Algorithm 2) and an algorithm for computing a potential explanation that is minimal with respect to a cost function and a knowledge base (Lines 2-3 in Algorithm 2). These two algorithms depend on the logic $L$ and the cost function $\mathcal{C}_L^x$ and need to be implemented for specific logic $L$ and function $\mathcal{C}_L^x$.

In the rest of this section, we discuss the implementation of our algorithms for propositional logic and different cost functions. With propositional logic, it is easy to see that checking for entailment can be done by a SAT solver (e.g., `MiniSat` (Eén and Sörensson 2003)). We next discuss two algorithm implementations, one for $m$-explanations and one for $p$-explanations, that find an explanation that is minimal with respect to a cost function and a knowledge base.

### Most-Preferred $m$-Explanations

Given a cost function $\mathcal{C}_L^m$ such as $c_L^1$, $c_L^2$, $c_L^3$, or $c_L^4$ as defined in Section , Algorithm 3 computes a most preferred $m$-explanations w.r.t. $\mathcal{C}_L^m$ from $KB_1$ to $KB_2$ for $\varphi$ or returns $nil$ if none exists.

**Algorithm 2:** most_preferred$(KB_1, KB_2, \varphi)$

**Input:** Logic $L$, formula $\varphi$, KBs $KB_1$ and $KB_2$, cost function $\mathcal{C}_L^x$
**Output:** A most-preferred explanation w.r.t. $\mathcal{C}_L^x$ from $KB_1$ to $KB_2$ for $\varphi$; or $nil$

1 **repeat**
2  non-deterministically select a potential $x$-explanation $\epsilon$, a minimal element w.r.t. $\mathcal{C}_L^x$ and $KB_2$
3  **if** $\epsilon \models \varphi$ *and* $KB_2 \cup \epsilon \models \varphi$ **then**
4  **return** $\epsilon$
5 **until** *all possible explanations are considered*
6 **return** $nil$

The key data structures in the algorithm is a priority queue $q$, initialized to only include the empty set, of potential explanations ordered by their costs (Line 1) and a set $checked$ of invalid explanations that have been considered thus far (line 2). The algorithm repeatedly loops the following steps: (*i*) move the explanation with the smallest cost from the priority queue $q$ to $checked$ (Lines 4-5); (*ii*) check if it is a valid $m$-explanation and return if it is (Lines 6-7); (*iii*) if not, extend the explanation by 1 (with each clause from $KB_1$) and insert the extended explanations into the priority queue $q$ (Lines 8-12). If all potential explanations are exhausted, which means that there are no valid $m$-explanations, then the algorithm returns $nil$ (Line 14). It is straightforward to see that the following proposition holds.

**Proposition 5** *For two propositional theories $KB_1$ and $KB_2$ and a formula $\varphi$, Algorithm 3 returns a most preferred $m$-explanation w.r.t. $\mathcal{C}_L^m$ for $\varphi$ from $KB_1$ to $KB_2$ if one exists.*

### Most-Preferred $p$-Explanations

Given a cost function $\mathcal{C}_L^p$ on $p$-explanations, Algorithm 4 computes a most-preferred $p$-explanation w.r.t. $\mathcal{C}_L^p$ from $KB_1$ to $KB_2$ for $\varphi$ or returns $nil$ if none exists.

We use the following notations in the pseudocode: For a proof $\langle\epsilon\rangle$, where $\epsilon$ is the sequence $\langle r_1; \ldots; r_n\rangle$, we write $c(\epsilon) = head(r_n)$ and $b(\epsilon) = \bigcup_{i=1}^n body(r_i)$. We also write $\frac{\varphi_1, \varphi_2}{\varphi}$ to indicate that $\varphi$ is the result of applying the resolution rule on $\varphi_1$ and $\varphi_2$. And we use $\circ$ to denote the concatenation of two sequences.

The algorithm uses the same two data structures – priority queue $q$ and set $checked$ – as in Algorithm 3. The algorithm first populates the queue $q$ with single-rule proofs consist of single clauses in $KB_1$ (Lines 2-4). Then, it repeatedly loops the following steps: (*i*) move the proof with the smallest cost from the priority queue $q$ to $checked$ (Lines 8-9); (*ii*) check if it is a valid $p$-explanation and return if it is (Lines 10-11); (*iii*) if not, extend the proof by 1 and insert the extended proofs into

**Algorithm 3:** `most_preferred_m`$(KB_1, KB_2, \varphi)$

**Input:** Formula $\varphi$, KBs $KB_1$ and $KB_2$, cost function $\mathcal{C}_L^m$
**Output:** A most-preferred $m$-explanation w.r.t. $\mathcal{C}_L^m$ from $KB_1$ to $KB_2$ for $\varphi$; or $nil$

```
1  q = [∅]          % a priority queue of potential explanations
2  checked = ∅      % a set of sets of elements in KB_1 that have been considered
3  repeat
4      ϵ = dequeue(q)
5      insert ϵ into checked
6      if ϵ ⊨ φ and KB_2 ∪ ϵ ⊨ φ then
7          return ϵ
8      else
9          for a ∈ KB_1 do
10             if ϵ ∪ {a} ∉ checked then
11                 v = C_L^m(KB_2, ϵ ∪ {a})
12                 q = enqueue(ϵ ∪ {a}) % use v as key
13 until q is empty
14 return nil
```

**Algorithm 4:** `most_preferred_p`$(KB_1, KB_2, \varphi)$

**Input:** Formula $\varphi$, KBs $KB_1$ and $KB_2$, cost function $\mathcal{C}_L^p$
**Output:** A most-preferred $p$-explanation w.r.t. $\mathcal{C}_L^p$ from $KB_1$ to $KB_2$ for $\varphi$; or $nil$

```
1  q = [∅]    % priority queue of potential explanations
2  for ϵ in KB_1 do
3      v = C_L^p(KB_2, ϵ)
4      q = enqueue(⟨ϵ⟩)        % use v as key
5  Ω = {⟨ϵ⟩ | ϵ ∈ KB_1}
6  checked = ∅
7  repeat
8      ⟨ϵ⟩ = dequeue(q)
9      insert (b(ϵ), c(ϵ)) into checked
10     if c(ϵ) = φ and KB_2 ∪ (b(ϵ) ∩ KB_1) ⊨ φ then
11         return ϵ
12     for ϵ' in Ω do
13         if c(ϵ) and c(ϵ') contain complementary literals and (c(ϵ), c(ϵ'))/ϕ holds then
14             ϵ̂ = ⟨ϵ ∘ ϵ'; (c(ϵ), c(ϵ'))/ϕ⟩
15             if (b(ϵ̂), ϕ) ∉ checked then
16                 v = C_L^p(KB_2, ϵ̂)
17                 q = enqueue(⟨ϵ̂⟩)        % use v as key
18 until q is empty
19 return nil
```

the priority queue $q$ (Lines 12-17). If all potential proofs are exhausted, which means that there are no valid $p$-explanations, then the algorithm returns $nil$ (Line 19). It is straightforward to see that the following proposition holds.

**Proposition 6** *For two propositional theories $KB_1$ and $KB_2$ and a formula $\varphi$, Algorithm 4 returns a most preferred $p$-explanation w.r.t. $\mathcal{C}_L^p$ for $\varphi$ from $KB_1$ to $KB_2$ if one exists.*

## Experimental Results

We empirically evaluate our implementation of Algorithm 3 to find $m$-explanations on two synthetically generated benchmarks – random knowledge bases and a planning domain called BLOCKSWORLD – both encoded in propositional logic.[3] We evaluated our algorithm using the four cost functions described in Section . Our algorithm was implemented in Python and experiments were performed on a machine with an Intel i7 2.6GHz processor and 16GB of RAM. We report both the cost of the optimal $m$-explanation found as well as the runtime of the algorithm.

[3]For random knowledge bases, we used an optimized version that uses a version of backward chaining that finds the set of all possible explanations. This approach works only when the clauses in the knowledge base are in Horn form and is sound and complete for such a case (Russell and Norvig 2009).

### Random Knowledge Bases

We first evaluated our algorithm on random knowledge bases with clauses in Horn form, where we varied the cardinality of $KB_1$ (the KB of the agent providing the explanation) from 20 to 1000. To construct $KB_2$ (the KB of the agent receiving the explanation), we randomly chose 25% of the clauses from $KB_1$.

To construct each $KB_1$, we first generated $\frac{|KB_1|}{2}$ random symbols, which will be used in the KB. Then, we iteratively generated clauses of increasing length $l$ from 2 to 7. For each length $l$, we generated $\frac{|KB_1|}{2 \cdot l}$ clauses using the symbols we previously generated such that each symbol is used at most once in these clauses of length $l$. Each clause is a conjunction of $l-1$ elements as the premise and the final $l^{\text{th}}$ element as the conclusion. For example, a KB with a cardinality of 20, 10 symbols are first generated. Then, 5 clauses of length 2, 3 clauses of length 3, 2 clauses of lengths 4 and 5, and 1 clause of lengths 6 and 7 are generated. Finally, to complete the KB, we add all the symbols that are exclusively in the premise of the clauses generated as facts in the KB. The formula $\varphi$ that we seek to explain is one of the randomly chosen conclusions in the clauses generated, which we ensure is entailed by $KB_1$.

Table 1(a) tabulates our results. We make the following observations:

(a) Experimental Results on Random Knowledge Bases

| $\|KB_1\|$ | $c_L^1$ | | $c_L^2$ | | $c_L^3$ | | $c_L^4$ | |
|---|---|---|---|---|---|---|---|---|
| | cost | time | cost | time | cost | time | cost | time |
| 20 | 7 | 23ms | 5 | 25ms | 2.4 | 26ms | 14 | 24ms |
| 100 | 15 | 2.5s | 10 | 3.0s | 3.8 | 3.1s | 30 | 2.9s |
| 1000 | 117 | 27m | 97 | 30m | 38 | 32m | 347 | 27m |

(b) Experimental Results on BLOCKSWORLD Domain

| $\|KB_1\|$ | $c_L^1$ | | $c_L^2$ | | $c_L^3$ | | $c_L^4$ | |
|---|---|---|---|---|---|---|---|---|
| | cost | time | cost | time | cost | time | cost | time |
| 225 | 4 | 15.0s | 1 | 16.0s | 0.5 | 15.5s | 7 | 15.0s |
| 387 | 16 | 2.0m | 12 | 2.2m | 0.5 | 2.2m | 35 | 2.0m |

Table 1: Experimental Results

- As expected, the runtimes increase as $|KB_1|$ increases since the algorithm will need to search over a larger search space.
- As expected, the costs of explanations also increase as $|KB_1|$ increases since the explanations are presumably longer and more complex.
- Finally, the runtimes for cost functions $c_L^1$ and $c_L^4$ are smaller than that of $c_L^2$ and $c_L^3$. The reason is the computation of the costs of possible explanations is faster with the former two cost functions since they are not dependent on $KB_2$ while the computation for the latter two cost functions are dependent on $KB_2$.

### Planning Domain

As we were motivated by the explanation generation problem studied in the automated planning community, we also conducted experiments on BLOCKSWORLD, a planning domain where multiple blocks must be stacked in a particular order on a table.[4]

For these planning problems, we first used FASTDOWNWARD (Helmert 2006) to find optimal solutions to the planning problem. Then, we translate the planning problem into a SAT problem with horizon $h$ (Kautz *et al.* 1992), where $h$ is the length of the optimal plan. These CNF clauses then form our $KB_1$ (the KB of the agent providing the explanation). Similar to random knowledge bases, we construct $KB_2$ (the KB of the agent receiving the explanation) by randomly choosing 25% of the clauses from $KB_1$. The formula $\varphi$ that we seek to explain is then that no plan of lengths 1 to $h-1$ exists, and that a plan of length $h$ (i.e., the plan found by FASTDOWNWARD) exists. Therefore, combined, that plan must be an optimal plan.

Table 1(b) tabulates our results, where we observe similar trends as in the experiment on random knowledge bases. The key difference is that the runtimes for all four cost functions here are a lot closer to each other, and the reason is because there was only one valid explanation in each problem instance. Thus, regardless of the choice of cost function, that explanation had to be found. Our experiments for larger problems are omitted as they timed out after 6 hours.

[4]It is one of the domains in the International Planning Competition. See *http://www.plg.inf.uc3m.es/ipc2011-learning/Domains.html*.

## Related Work and Discussions

There is a very large body of work related to the very broad area of explainable AI. We have briefly discussed some of them from the ML and planning literature in Section . We refer readers to surveys by (**?**) and (**?**) for more in-depth discussions of this area. Due to space limitations, we focus only on related work from the KR literature below.

We note that the notion of an explanation proposed in this paper might appear similar to the notion of a diagnosis that has been studied extensively in the last several decades (e.g., (Reiter 1987)) as both aim at explaining something to an agent. Diagnosis focuses on identifying the reason for the inconsistency of a theory whereas an $m$- or $p$-explanation aims at identifying the support for a formula. The difference lies in that a diagnosis is made with respect to the same theory and $m$- or $p$-explanation is sought for the second theory.

Another earlier research direction that is closely related to the proposed notion of explanation is that of developing explanation capabilities of knowledge-based systems and decision support systems, which resulted in different notions of explanation such as trace, strategic, deep, or reasoning explanations (see review by (**?**) for a discussion of these notions). All of these types of explanations focus on answering why certain rules in a knowledge base are used and how a conclusion is derived. This is not our focus in this paper. The present development differs from earlier proposals in that $m$- or $p$-explanations are identified with the aim of explaining a given formula to a second theory. Furthermore, the notion of an optimal explanation with respect to the second theory is proposed.

There have been attempts to using argumentation for explanation (Cyras *et al.* 2017; Cyras *et al.* 2019) because of the close relation between argumentation and explanation. For example, argumentation was used by (Cyras *et al.* 2019) to answer questions such as why a schedule does (does not) satisfy a criteria (e.g., feasibility, efficiency, etc.); the approach was to develop for each type of inquiry, an abstract argumentation framework (AF) that helps explain the situation by extracting the attacks (non-attacks) from the corresponding AF. Our work differs from these works in that it is more general and does not focus on a specific question.

It is worth to pointing out that the problem of computing a most preferred explanation for $\varphi$ from $KB_1$ to $KB_2$ might look similar to the problem of computing a weakest sufficient condition of $\varphi$ on $KB_1$ under $KB_2$ as described by (Lin 2001). As it turns out, the two no-

tions are quite different. Given that $KB_1 = \{p, q\}$ and $KB_2 = \{p\}$. It is easy to see that $q$ is the unique explanation for $q$ from $KB_1$ to $KB_2$. On the other hand, the weakest sufficient condition of $q$ on $KB_1$ under $KB_2$ is $\perp$ (Proposition 8, (Lin 2001)).

## Conclusions and Future Work

Explanation generation is an important problem within the larger explainable AI thrust. Existing work on this problem has been done in the context of automated planning domains, where researchers have primarily employed, unsurprisingly, automated planning approaches. In this paper, we approach the problem from the perspective of KR, where we propose a general logic-based framework for explanation generation. We further define two types of explanations, model- and proof-theoretic explanations, and use cost functions to reflect preferences between explanations. Our empirical results with algorithms implemented for propositional logic on both random knowledge bases as well as a planning domain demonstrate the generality of our approach beyond planning problems. Future work includes investigating more complex scenarios, such as one where an agent needs to persuade another that its knowledge base is incorrect.

## References


[Alvarez Melis and Jaakkola 2018] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. pages 7775–7784, 2018.

[Chakraborti *et al.* 2017] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of IJCAI*, pages 156–163, 2017.

[Cyras *et al.* 2017] Kristijonas Cyras, Xiuyi Fan, Claudia Schulz, and Francesca Toni. Assumption-based argumentation: Disputes, explanations, preferences. *Journal of Logics and their Applications*, 4(8), 2017.

[Cyras *et al.* 2019] Kristijonas Cyras, Dimitrios Letsios, Ruth Misener, and Francesca Toni. Argumentation for explainable scheduling. In *Proceedings of AAAI*, 2019.

[Davis *et al.* 1962] Martin Davis, George Logemann, Donald, and Loveland. A Machine Program for Theorem Proving. *Communications of the ACM*, 5(7):394–397, 1962.

[Dong *et al.* 2017] Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. Improving interpretability of deep neural networks with semantic information. In *Proceedings of CVPR*, pages 4306–4314, 2017.

[Eén and Sörensson 2003] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *International conference on theory and applications of satisfiability testing*, pages 502–518. Springer, 2003.

[Gilpin *et al.* 2018] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings of DSAA*, pages 80–89, 2018.

[Guidotti *et al.* 2018] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Survey*, 51(5):93:1–93:42, 2018.

[Helmert 2006] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Kambhampati 1990] Subbarao Kambhampati. A classification of plan modification strategies based on coverage and information requirements. In *AAAI Spring Symposium Series*, 1990.

[Kautz *et al.* 1992] Henry A Kautz, Bart Selman, et al. Planning as satisfiability. In *Proceedings of ECAI*, volume 92, pages 359–363, 1992.

[Langley 2016] Pat Langley. Explainable agency in human-robot interaction. In *AAAI Fall Symposium Series*, 2016.

[Lin 2001] Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128(1-2):143–159, 2001.

[Petkovic *et al.* 2018] Dragutin Petkovic, Russ Altman, Mike Wong, and Arthur Vigil. Improving the explainability of random forest classifier–user centered approach. In *Pacific Symposium on Biocomputing*, volume 23, pages 204–215, 2018.

[Reiter 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Robinson 1965] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Communications of the ACM*, 5:23–41, 1965.

[Russell and Norvig 2009] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson, 2009.

[Sreedharan *et al.* 2018] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *Proceedings of IJCAI*, pages 4829–4836, 2018.

[Tian *et al.* 2016] Xin Tian, Hankz Hankui Zhuo, and Subbarao Kambhampati. Discovering underlying plans based on distributed representations of actions. In *Proceedings of AAMAS*, pages 1135–1143, 2016.

[Vaughan *et al.* 2018] Joel Vaughan, Agus Sudjianto, Erind Brahimi, Jie Chen, and Vijayan N. Nair. Explain-

able neural networks based on additive index models. *CoRR*, abs/1806.01933, 2018.

[Zhang *et al.* 2017] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan explicability and predictability for robot task planning. In *Proceedings of ICRA*, pages 1313–1320, 2017.