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).

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 (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 (Sreedsaran 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 compute such explanations and empirically evaluate it in random knowledge bases as well as in a planning domain.

In addition to providing an alternative approach to

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

\footnote{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.}

\footnote{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 formulæ. $BS_L$ 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 $KB \models_\varphi$, 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 formulæ 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 $\neg \varphi \not\in B$; and if $\neg \varphi \in B$ then $\varphi \not\in B$. $\varphi$ is called a sub-theory of $KB$. A theory $KB$ subsumes a theory $KB'$, denoted by $KB \subseteq KB'$ if $ACC_L(KB) \subseteq ACC_L(KB')$.

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

$$\varphi_1, \ldots, \varphi_k \vdash_L \varphi_0 \quad (1)$$

where $\varphi_i$ are formulæ. 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 $KB$ exists a sequence of rules $r_1, \ldots, r_n$ in $\Sigma_L$ such that $body(r_1) \subseteq KB$, $head(r_1) = \varphi$, $head(r_i) \subseteq KB$ for $i = 1, \ldots, n - 1$, and $body(r_1) \subseteq body(r_i) \cup \{head(r_j) \mid j = 1, \ldots, i - 1\}$ 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 $KB |=_L \varphi$. It is complete if for every $\varphi$, $KB |\!|= L \varphi$ 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 formulæ 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 $KB_1 \models_\varphi$ and $KB_2 \not\models_\varphi$. The goal is to identify an explanation (i.e., a set of formulæ) $\epsilon \subseteq KB_1$ such that $KB_2 \cup \epsilon \models \varphi$.

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

Definition 1 (Support) Assume that $KB \models_\varphi$. We say that $\epsilon \subseteq KB$ is a support of $\varphi$ w.r.t. $KB$ if $KB \models_\epsilon \varphi$. Assume that $\epsilon$ is a support of $\varphi$ w.r.t. $KB$. We say that $\epsilon \subseteq KB$ is a $\epsilon$-minimal support of $\varphi$ if no proper sub-theory of $\epsilon$ is a support of $\varphi$. Furthermore, $\epsilon$ is a $\epsilon$-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 $(\epsilon$-Explanation) Given two knowledge bases $KB_1$ and $KB_2$ in logic $L$ and a formula $\varphi$. Assume that $KB_1 \models_\varphi$ and $KB_2 \not\models_\varphi$. A model-theoretic explanation (or $\epsilon$-explanation) for $\varphi$ from $KB_1$ to $KB_2$ is a support $\epsilon$ w.r.t. $KB_1$ for $\varphi$ such that $KB_2 \cup \epsilon \models_\varphi$.

Example 2 Consider proposition logic theories over the set of propositions $\{a, b, c\}$ with the usual definition of models, satisfaction, etc. Assume $KB_1 = \{a, b \rightarrow c, c \land a \rightarrow c\}$ and $KB_2 = \{a\}$. We have that $\epsilon_1 = \{a, a \rightarrow c\}$ and $\epsilon_2 = \{a, b, a \land b \rightarrow c\}$ are two $\epsilon$-minimal supports of $c$ w.r.t. $KB_1$. Only $\epsilon_1$ is a $\epsilon$-general support of $c$ w.r.t. $KB_1$, since $\epsilon_2 \not\prec \epsilon_1$.

Both $\epsilon_1$ and $\epsilon_2$ can serve as $\epsilon$-explanations for $c$ from $KB_1$ for $KB_2$. Of course, $KB_1$ is itself an $\epsilon$-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 $\epsilon$-explanation for $c$ from $KB_1$ for $KB_3$.

Now, consider $KB_4 = \{\neg a\}$. In this case, we have no $\epsilon$-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 $KB_1 \models_\varphi$ and $KB_2 \models_\varphi \neg \varphi$, then there exists no $\epsilon$-explanation for $\varphi$ from $KB_1$ for $KB_2$. 

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 formulæ given a knowledge base.
The $KB_1$ 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 $KB_2 \not\models_L \varphi$.

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 $KB_2 \cup (\bigcup_{i=1}^n \text{body}(r_i) \cap KB_1) \models_L \varphi$ and $KB_2 \cup (\bigcup_{i=1}^n \text{body}(r_i) \cap KB_1)$ is consistent.

**Example 3** Consider the theories $KB_1 = \{a, b, a \rightarrow c, a \land b \rightarrow c\}$ 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 $\neg l \rightarrow l \vdash$ for any literals $l, p$ in the language of $KB_1$ and $KB_2$. Then, $\langle a \land b \rightarrow c, a \rightarrow \neg b \land c \vdash_L c \rangle$ is a proof for $c$ from $KB_1$ for $KB_2$.

Likewise, $\langle a \vdash_L a; b \vdash_L b; a, a \land b \rightarrow c \vdash_L \neg b \land c \rangle$ 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. $\Sigma_L$, $\bigcup_{i=1}^n \text{body}(r_i) \cap KB_1$ 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 $L$.

- 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 $\bigcup_{i=1}^n \text{body}(r_i) \cap KB_1 \subseteq \epsilon$; and
- for each $p$-explanation $\langle r_1; \ldots; r_n \rangle$ for $\varphi$ from $KB_1$ for $KB_2$, $\bigcup_{i=1}^n \text{body}(r_i) \cap KB_1$ 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 $C^p_L$ that maps pairs of knowledge bases and sets of explanations to non-negative real values, i.e.,

$$C^p_L : KB_L \times \Omega \rightarrow \mathbb{R}^{\geq 0}$$

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

**Definition 5 (Most Preferred Explanation)** Given a cost function $C^p_L$, 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 $\epsilon_1 \preceq^{KB_2} \epsilon_2$) if

$$C^p_L(KB_2, \epsilon_1) \leq C^p_L(KB_2, \epsilon_2)$$

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

$$C^p_L(KB_2, \epsilon_1) < C^p_L(KB_2, \epsilon_2)$$

This allows us to compare explanations as follows.

**Definition 4 (Preferred Explanation)** Given a cost function $C^p_L$, 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 $\epsilon_1 \preceq^{KB_2} \epsilon_2$) if

$$C^p_L(KB_2, \epsilon_1) \leq C^p_L(KB_2, \epsilon_2)$$

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

$$C^p_L(KB_2, \epsilon_1) < C^p_L(KB_2, \epsilon_2)$$

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

- $c^m_L(KB_2, \epsilon) = |\epsilon|$, the cardinality of $\epsilon$, indicates the number of formulas that need to be explained;
- $c^m_L(KB_2, \epsilon) = |\epsilon \setminus KB_2|$, the cardinality of $\epsilon \setminus KB_2$, indicates the number of new formulas that need to be explained;
- $c^m_L(KB_2, \epsilon) = |\text{new-vars}(KB_2, \epsilon)|$ indicates the number of new symbols occurring in $\epsilon$ that are not in $KB_2$ and need to be explained;
- $c^m_L(KB_2, \epsilon) = \text{length}(\epsilon)$ 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 $C_L$
Output: A most preferred $x$-explanation w.r.t. $C_L^x$ 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. $\epsilon = \text{most_preferred}(KB_1, KB_2, \varphi)$
5. return $\epsilon$

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

Input: Logic $L$, formula $\varphi$, KBs $KB_1$ and $KB_2$, cost function $C_L$
Output: A most-preferred $p$-explanation w.r.t. $C_L^p$ 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. $C_L^x$ 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 $\text{checked}$ 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 $\text{checked}$ (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. $C_L^m$ for $\varphi$ from $KB_1$ to $KB_2$ if one exists.

Most-Preferred $p$-Explanations

Given a cost function $C_L^p$ on $p$-explanations, Algorithm 4 computes a most-preferred $p$-explanation w.r.t. $C_L^p$ 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 $\epsilon(\epsilon) = \text{head}(r_n)$ and $b(\epsilon) = \bigcup_{i=1}^{n} \text{body}(r_i)$. We also write $\epsilon_1 \epsilon_2$ 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 $\text{checked}$ – 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 $\text{checked}$ (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

Naturally, some of these cost functions can also be combined (e.g., $c_1^L + c_2^L$ will measure the number of new formulas and new symbols that must be explained).

Observe that the three functions $c_1^L$ and $c_2^L$ are independent from $KB_2$ while $c_1^L$ and $c_2^L$ 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 $c_1^L$ as the cost function, then we have that $x \models_{KB_2} c_1^L x \models_{KB_2} KB_1$. Furthermore, $x \models_{KB_2} c_1^L$ 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 $C_L^x$. 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 $C_L^x$ and need to be implemented for specific logic $L$ and function $C_L^x$.

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 $C_L^m$ as defined in Section 2, Algorithm 3 computes a most preferred $m$-explanation w.r.t. $C_L^m$ from $KB_1$ to $KB_2$ for $\varphi$ or returns nil if none exists.

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 $\text{checked}$ 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 $\text{checked}$ (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. $C_L^m$ for $\varphi$ from $KB_1$ to $KB_2$ if one exists.

Most-Preferred $p$-Explanations

Given a cost function $C_L^p$ on $p$-explanations, Algorithm 4 computes a most-preferred $p$-explanation w.r.t. $C_L^p$ 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 $\epsilon(\epsilon) = \text{head}(r_n)$ and $b(\epsilon) = \bigcup_{i=1}^{n} \text{body}(r_i)$. We also write $\epsilon_1 \epsilon_2$ 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 $\text{checked}$ – 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 $\text{checked}$ (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(KB₁, KB₂, ϕ)

Input: Formula ϕ, KBs KB₁ and KB₂, cost function C𝐾𝐵₁

Output: A most-preferred m-explanation w.r.t. C𝐾𝐵₁

1 q = [0] % a priority queue of potential explanations
2 checked = ∅ % a set of sets of elements in KB₁ that have been considered
3 repeat
4 ε = dequeue(q)
5 if ε ⊨ ϕ and KB₂ ∪ ε ⊨ ϕ then
6 return ε
7 else
8 for a ∈ KB₁ do
9 if ε ∪ {a} ⊄ checked then
10 if ε ∪ {a} ⊄ checked then
11 v = C𝐾𝐵₂(KB₂, ε ∪ {a})
12 q = enqueue(ε ∪ {a}) % use v as key
13 until q is empty
14 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 KB₁ and KB₂ and a formula ϕ, Algorithm 4 returns a most preferred p-explanation w.r.t. C𝐾𝐵₂ for ϕ from KB₁ to KB₂ 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. We evaluated our algorithm using the four cost functions described in Section 5. 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.

Algorithm 4: most_preferred_p(KB₁, KB₂, ϕ)

Input: Formula ϕ, KBs KB₁ and KB₂, cost function C𝐾𝐵₂

Output: A most-preferred p-explanation w.r.t. C𝐾𝐵₂ from KB₁ to KB₂ for ϕ; or nil

Random Knowledge Bases

We first evaluated our algorithm on random knowledge bases with clauses in Horn form, where we varied the cardinality of KB₁ (the KB of the agent providing the explanation) from 20 to 1000. To construct KB₂ (the KB of the agent receiving the explanation), we randomly chose 25% of the clauses from KB₁.

To construct each KB₁, we first generated |KB₁| 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 |KB₁| 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 lth 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.

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

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).
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 $c_1$ and $c_4$ are smaller than that of $c_2$ and $c_3$. 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.

### 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-

---

**Table 1:** Experimental Results

| $|KB_1|$ | $c_1$ time | $c_2$ time | $c_3$ time | $c_4$ time |
|-------|-----------|-----------|-----------|-----------|
| 20    | 7 2.5ms   | 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   |

| $|KB_1|$ | $c_1$ time | $c_2$ time | $c_3$ time | $c_4$ 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   |

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