exp(ASP\textsuperscript{c}): Explaining ASP Programs with Choice Atoms and Constraint Rules

Ly Ly Trieu\textsuperscript{1}, Tran Cao Son\textsuperscript{1} and Marcello Balduccini\textsuperscript{2}

\textsuperscript{1}New Mexico State University, New Mexico, USA
\textsuperscript{2}Saint Joseph\textquotesingle s University, Pennsylvania, USA

Abstract
We present an enhancement of exp(ASP), a system that generates explanation graphs for a literal \(\ell\) – an atom \(a\) or its default negation \(\sim a\) – given an answer set \(A\) of a normal logic program \(P\), which explain why \(\ell\) is true (or false) given \(A\) and \(P\). The new system, exp(ASP\textsuperscript{c}), differs from exp(ASP) in that it supports choice rules and utilizes constraint rules to provide explanation graphs that include information about choices and constraints.

Keywords
explainable Artificial Intelligence, Answer Set Programming, Artificial Intelligence.

1. Introduction

Answer Set Programming (ASP) \cite{1, 2} is a popular paradigm for decision making and problem solving in Knowledge Representation and Reasoning. It has been successfully applied in a variety of applications such as robotics, planning, diagnosis, etc. ASP is an attractive programming paradigm as it is a declarative language, where programmers focus on the representation of a specific problem as a set of rules in a logical format, and then leave computational solutions of that problem to an answer set solver. However, this mechanism typically gives little insight into why something is a solution and why some proposed set of literals is not a solution. This type of reasoning falls within the scope of explainable Artificial Intelligence and is useful to enhance the understanding of the resulting solutions as well as for debugging programs. There have been a number of approaches proposed \cite{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}, but to the best of our knowledge, no system deals directly with ASP programs with choice atoms.

In this paper, we present an improvement over our previous system, exp(ASP) \cite{14}, called exp(ASP\textsuperscript{c}). Given an ASP program \(P\), an answer set \(A\), and an atom \(a\), exp(ASP\textsuperscript{c}) is aimed at answering the question “why is \(a\) true/false in \(A\)” by producing explanation graphs for...
atom $a$. The current system, $\text{exp(ASP)}$, does not consider programs with choice atoms and other constructs that extend the modeling capabilities of ASP. For instance, Fig. 1 shows an explanation graph for the atom $\text{colored}(1, \text{red})$ given the typical encoding of the graph coloring problem that does not use choice rules. This explanation graph does provide the reason for the color assigned to node 1 by indicating that the node is red because it is not blue and not green. It is not obvious that this information represents the requirement that each node is colored with exactly one color. The improvement described here extends our approach with the ability to handle ASP programs containing choice rules and include constraint information in the explanation graphs.

The rest of the paper is organised as follows. Section 2 briefly introduces our previous system, $\text{exp(ASP)}$. Section 3 describes the components of an explanation graph. Section 4 describes how our enhanced system, $\text{exp(ASP}^c)$, computes explanation graphs for an atom $a$, with an illustrative example. Finally, Section 5 concludes our paper.

2. Background: The $\text{exp(ASP)}$ System

$\text{exp(ASP)}$ deals with normal logic programs which are collection of rules of the form $\text{head}(r) \leftarrow \text{body}(r)$ where $\text{head}(r)$ is an atom and $\text{body}(r) = r^+ \setminus r^-$ are collections of atoms in a propositional language and $\text{not } r^-$ denotes the set $\{ \text{not } x \mid x \in r^- \}$ and $\text{not}$ is the default negation.

$\text{exp(ASP)}$ generates explanation graphs under the answer set semantics [15]. It implemented the algorithms proposed in [12] to generate explanation graphs of a literal $\ell$ (or $\sim a$ for some atom $a$ in the Herbrand base $H$ of $P$), given an answer set $A$ of a program $P$. Specifically, the system produces labeled directed graphs, called explanation graphs, for $\ell$, whose nodes belong to $H \cup \{ \sim x \mid x \in H \} \cup \{ \top, \bot, \text{assume} \}$ and whose links are labeled with $+, -, o$ (in Fig. 1, solid/dash/dot edges represent $+/−/o$ edges). Intuitively, for each node $x, x \notin \{ \top, \bot, \text{assume} \}$ in an explanation graph $(E, G)$, the set of neighbors of $x$ represents a support for $x$ being true given $A$ (see below).

The main components of $\text{exp(ASP)}$ are:

1. **Preprocessing:** This component produces an $\text{aspi}$ representation [16] of $P$ that will be used in the reconstruction of ground rules of $P$. It also computes supported sets for atoms (or its negations) in the Herbrand base of $P$ and stored in an associative array $E$.

2. **Computing minimal assumption set:** This calculates a minimal assumption set $U$ given the answer set $A$ and $P$ according to the definition in [12].

3. **Computing explanation graphs:** This component uses the supported sets in $E$ and constructs $e$-graphs for atoms in $H$ (or their negations) under the assumption that each element $u \in U$ is assumed to be false.

We note that $\text{exp(ASP)}$ does not deal with choice atoms [17]. The goal of this paper is to extend $\text{exp(ASP)}$ to deal with choice atoms and utilize constraint information.

3. Explanation Graphs in Programs with Choice Atoms

$\text{exp(ASP)}$ employs the notion of a supported set of a literal in a program in its construction.
Given a program \( P \), an answer set \( A \) of \( P \), and an atom \( c \), if \( c \in A \) and \( r \) is a rule such that (i) \( \text{head}(r) = c \), (ii) \( r^+ \subseteq A \), and (iii) \( r^- \cap A = \emptyset \), \( \text{support}(c, r) = r^+ \cup \{ \sim n \mid n \in r^- \} \); and refer to this set as a supported set of \( c \) for rule \( r \). If \( c \notin A \), for every rule \( r \) such that \( \text{head}(r) = c \), then \( \text{support}(\sim c, r) = \emptyset \). We use choice atoms synonymous with weight constraints.

To account for choice atoms\(^1\) in \( P \), the notion of supported set needs to be extended. For simplicity of the presentation, we assume that any choice atom \( x \) is of the form \( \{p_1 : q_1, \ldots, p_n : q_n\} u \) where\(^2\) \( p_i \)’s and \( q_i \)’s are atoms. Let \( x_i \) and \( x_u \) denote \( l \) and \( u \), respectively. Furthermore, we write \( c \in x \) to refer to an element in \( \{p_1, \ldots, p_n\} \). For \( c \in x \), \( q_1 \equiv c \) indicates that \( c : q_i \) belongs to \( \{p_1 : q_1, \ldots, p_n : q_n\} \).

In the presence of choice atoms, an atom \( c \) can be true because \( c \) belongs to a choice atom that is a head of a rule \( r \) and \( \text{body}(r) \) is true in \( A \). In that case, we say that \( c \) is chosen to be true and extend \( \text{support}(c, r) \) with a special atom \(+\text{choice} \) to indicate that \( c \) is chosen to be true. Likewise, \( c \) can be false even if it belongs to a choice atom that is a head of a rule \( r \) and \( \text{body}(r) \) is true in \( A \). In that case, we say that \( c \) is chosen to be false and extend \( \text{support}(\sim c, r) \) with a special atom \(-\text{choice} \) to indicate that \( c \) is chosen to be false. Also, \( q \equiv c \) will belong to the supported set of \( \text{support}(c, r) \) and \( \text{support}(\sim c, r) \).

The above extension only considers the case \( c \) belongs to the head of a rule. \( \text{support}(c, r) \) also needs to be extended with atoms corresponding to choice atoms in the body of \( r \). Assume that \( x \) is a choice atom in \( r^+ \). By definition, if \( \text{body}(r) \) is true in \( A \) then \( x_l \leq |S| \leq x_u \) where \( S = \{ (c, q) \mid c \in x, q \cong c, A \models c \land q \} \). For this reason, we extend \( \text{support}(c, r) \) with \( x \). Because \( x \) is not a standard atom, we indicate the support of \( x \) given \( A \) by defining \( \text{support}(x, r) = \{ S \} \). Furthermore, for each \( s \in S \), \( \text{support}(s, r) = \{ \text{True} \} \). When \( S = \emptyset \), we write \( \text{support}(x, r) = \{ \text{Empty} \} \). Similar elements will be added to \( \text{support}(c, r) \) or \( \text{support}(\sim c, r) \) in other cases (e.g., the choice atom belongs to \( r^- \) or has different form (e.g., when \( l = 0 \) or \( u = \infty \)). We omit the precise definitions of the elements that need to be added to \( \text{support}(c, r) \) for brevity.

The introduction of different elements in supported sets of literals in a program necessitates the extension of the notion of explanation graph. Due to the space limitation, we introduce its key components and provide the intuition behind each component. The precise definition of an explanation graph is rather involved and is included in the appendix for review. First, we introduce additional types of nodes. Besides \(+\text{choice} \), \(-\text{choice} \), \text{True}, and \text{Empty}, we consider the following types of nodes:

- **Tuples** are of the form \( (x_1, \ldots, x_m, \text{not } y_1, \ldots, \text{not } y_n) \) to represent elements belonging to choice atoms (e.g., \( \text{colored}(1, \text{blue}), \text{color}(\text{blue}) \)) representing an element in \( 1\{\text{colored}(N, C) : \text{color}(C)\}1 \). \( T \) denotes all tuple nodes in program \( P \).
- **Choices** are of the form \( l \leq T \leq u \) or \( \sim (l \leq T \leq u) \) where \( T \subseteq T \). Intuitively, when \( l \leq T \leq u \) (resp. \( \sim (l \leq T \leq u) \)) occurs in an explanation graph, it indicates that \( l \leq T \leq u \) is satisfied (resp. not satisfied) in the given answer set \( A \). \( O \) denotes all choices.
- **Constraints** are of the form \( \text{triggered_constraint}(x) \) or \( \text{triggered_constraint}(\sim x) \). The former (resp. latter) indicates that \( x \) is (resp. is not) true in \( A \) and satisfies all

\(^1\)We use choice atoms synonymous with weight constraints.

\(^2\)As we employ the aspif representation, this is a reasonable assumption.
the constraints }\mathit{r}\text{ such that } x \in r^+ \text{ (resp. } x \in r^-). The set of all constraints is denoted with } \mathcal{C}.

Having defined the nodes of the graph, we next introduce the new types of links in explanation graphs as follows:

- • is used to connect literals }c\text{ and } \sim c\text{ to +choice and } \text{−choice, respectively, where } c \in x \text{ and } x \text{ is a choice atom in the head of a rule.}
- ◦ is used to connect literals }c\text{ and } \sim c\text{ to } \text{triggered_constraint}(c)\text{ and } \text{trigger_ed_constraint}(\sim c)\text{, respectively.}
- ⨀ is used to connect a tuple } t \in \mathcal{T}\text{ to } \ast \text{True.}
- ⨂ is used to connect a choice } n \in \mathcal{O}\text{ to } \ast \text{Empty.}

We present here an updated definition of explanation graph by adding the necessary nodes and links, which is defined as follows:

**Definition 1.** [Explanation Graph] Let us consider a program } P\text{, an answer set } A\text{, a set of assumptions } \mathcal{U}\text{ with respect to } A\text{, Herbrand base } \mathcal{H}\text{ and a set of choice head atoms } \mathcal{G} = \{ g \mid g \in c, \text{choice head } c \text{ of rule } r, r \in P \}. \text{ Let } \mathcal{T} = \{ (x_1, \ldots , x_m, \text{not } y_1, \ldots , \text{not } y_n) \mid x_i, y_i \in \mathcal{H} \}, \mathcal{O} = \{ l \leq \{ t_1, \ldots , t_m \} \leq u \mid t_i \in \mathcal{T}, l \in \mathbb{N}, u \in \mathbb{N} \cup \emptyset \}, \mathcal{C} = \{ \text{triggered_constraint}(x) \mid x \in A \} \cup \{ \text{triggered_constraint}(\sim x) \mid x \notin A \}, \mathcal{N} = \mathcal{N} \cup \mathcal{C} \cup \{ \sim o \mid o \in \mathcal{O} \} \cup \mathcal{T} \cup \{ \perp, \top, \text{assume, +choice, −choice, } \ast \text{True, } \ast \text{Empty} \} \text{, where } \top \text{ and } \perp \text{ represent true and false, respectively. An explanation graph of an atom } a \text{ occurring in } P \text{ is a finite labeled and directed graph } DG_a = (\mathcal{N}_a, \mathcal{E}_a) \text{ with } \mathcal{N}_a \subseteq \mathcal{N} \text{ and } \mathcal{E}_a \subseteq \mathcal{N}_a \times \mathcal{N}_a \times \{ +, −, ◦, •, ⨀, ⨂, ⨂ \}, \text{ where } (x, y, z) \in \mathcal{E}_a \text{ represents a link from } x \text{ to } y \text{ with the label } z, \text{ and satisfies the first five conditions in Definition 2.1 [14] and the following additional conditions:}

- if } (x, +\text{choice, •}) \in \mathcal{E}_a \text{ then } x \in A \cap \mathcal{G};
- if } (\sim x, −\text{choice, •}) \in \mathcal{E}_a \text{ then } x \notin A \text{ and } x \in \mathcal{G};
- if } (x, y, ◦) \in \mathcal{E}_a \text{ then } y \in \mathcal{C} \text{ and } y \text{ is of the form } \text{triggered_constraint}(x);
- if } (x, \ast \text{True, ◦}) \in \mathcal{E}_a \text{ then } x \in \mathcal{T};
- if } (x, \ast \text{Empty, ◦}) \in \mathcal{E}_a \text{ then } x \in \mathcal{O} \cup \{ \sim o \mid o \in \mathcal{O} \};
- if } (x, y, +) \in \mathcal{E}_a \text{ such that } x \in \mathcal{O} \text{ and } t \in \mathcal{T} \text{ then } t \in \{ t_1, \ldots , t_m \};
- if } (x, y, −) \in \mathcal{E}_a \text{ such that } x \in \{ \sim o \mid o \in \mathcal{O} \} \text{ and } t \in \mathcal{T} \text{ then } t \in \{ t_1, \ldots , t_m \};
- if } \text{triggered_constraint}(x), (y, +) \in \mathcal{E}_a \text{ and } \text{triggered_constraint}(x), (y, −) \in \mathcal{E}_a \text{ then for all triggered constraints containing } x \in r^+ \text{ in } P \text{ satisfied by } A.
- if } \text{triggered_constraint}(\sim x), (y, +) \in \mathcal{E}_a \text{ and } \text{triggered_constraint}(\sim x), (y, −) \in \mathcal{E}_a \text{ then for all triggered constraints containing } x \in r^- \text{ in } P \text{ satisfied by } A.
- there exists no } x, y \text{ such that } (\perp, x, y) \in \mathcal{E}_a \text{, } (\top, x, y) \in \mathcal{E}_a \text{, } (\text{assume}, x, y) \in \mathcal{E}_a \text{, } (+\text{choice}, x, y) \in \mathcal{E}_a \text{, } (−\text{choice}, x, y) \in \mathcal{E}_a \text{, } (\ast \text{True}, x, y) \text{ or } (\ast \text{Empty}, x, y) \in \mathcal{E}_a.
- for every } x \in \mathcal{N}_a \cap A \text{ and } x \text{ is not a fact in } P, \text{ or } x \in G_{na} = \{ \sim g \mid g \notin A \land g \in \mathcal{G} \}
  - there exists no } y \in \mathcal{N}_a \cap A \text{ such that } (x, y, −) \text{ or } (x, y, ◦) \text{ belong to } \mathcal{E}_a;
  - there exists no } \sim y \in \mathcal{N}_a \cap \{ \sim u \mid u \notin A \} \text{ such that } (x, \sim y, +) \text{ or } (x, \sim y, ◦) \text{ belong to } \mathcal{E}_a.
Ly Ly Trieu et al.

- If we have
  * \( X^+ = \{ a \mid (x,a,+) \in E_a, a \in H \} \) then \( X^+ \subseteq A \),
  * \( X^- = \{ a \mid (x,\sim a,+) \in E_a, a \in H \} \) then \( X^- \cap A = \emptyset \),
  * \( C_1 = (x,y,+) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_1 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_1, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( l \leq |S_1| \leq u \),
  * \( C_2 = (x,\sim (y),-) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_2 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_2, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( |S_2| < l \) or \( |S_2| > u \),
  * \( C_3 = (x,y,+) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_3 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_3, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( |S_3| \geq l \),
  * \( C_4 = (x,\sim (y),-) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_4 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_4, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( |S_4| < l \),
  * there is a rule \( r \in \mathcal{P} \) whose head is \( x \) or \( x \in \mathcal{C} \) where \( c = l \{ p_1 : q_1, \ldots, p_n : q_n \} \) u such that \( r^+ = X^+ \setminus \{ \text{choice atom } c_1 \} \), \( r^- = X^- \setminus \{ \text{choice atom } c_1 \} \), and whose body contains choice atoms
    - \( c_1 = l \{ p_1 : q_1, \ldots, p_n : q_n \} u \) in \( r^+ \) if we have \( C_1 \);
    - \( c_2 = l \{ p_1 : q_1, \ldots, p_n : q_n \} u \) in \( r^- \) if we have \( C_2 \);
    - \( c_3 = l \{ p_1 : q_1, \ldots, p_n : q_n \} \subseteq r^+ \) or \( c_3' = \{ p_1 : q_1, \ldots, p_n : q_n \} l-1 \in r^- \) if we have \( C_3 \). Note that in \( c_3' \), the upper bound \( c_{3u} = l-1 \);
    - \( c_4 = l \{ p_1 : q_1, \ldots, p_n : q_n \} \subseteq r^- \) or \( c_4' = \{ p_1 : q_1, \ldots, p_n : q_n \} l-1 \in r^+ \) if we have \( C_4 \). Note that in \( c_4' \), the upper bound \( c_{4u} = l-1 \).
- \( DG_a \) contains no cycle containing \( x \).

• for every \( \sim x \in N_a \cap \{ u \mid u \notin A \} \) and \( x \notin U \),
  * there exists no \( y \in N_a \cap A \) such that \( (\sim x, y, +) \) or \( (\sim x, y, -) \) belong to \( E_a \);
  * there exists no \( y \in N_a \cap \{ u \mid u \notin A \} \) such that \( (\sim x, \sim y, +) \) or \( (\sim x, \sim y, -) \) belong to \( E_a \);
  * if we have
    * \( X^+ = \{ a \mid (\sim x, a, -) \in E_a, a \in H \} \) then \( X^+ \subseteq A \)
    * \( X^- = \{ a \mid (\sim x, \sim a, +) \in E_a, a \in H \} \) then \( X^- \cap A = \emptyset \)
    * \( C_1 = (\sim x, \sim (y), -) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_1 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_1, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( |S_1| < l \) or \( |S_1| > u \),
    * \( C_2 = (\sim x, y, +) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_2 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_2, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( |S_2| < l \) or \( |S_2| > u \),
    * \( C_3 = (\sim x, \sim (y), +) \in E_a \) where \( y = l \leq \{ t_1, \ldots, t_m \} \leq u \in \mathcal{O} \) and a set of atoms \( S_3 \subseteq \{ t_1, \ldots, t_m \} \) such that \( \forall t = (x_{t_1}, \ldots, x_{t_m}, \not y_{t_1}, \ldots, \not y_{t_n}) \in S_3, A \models x_{t_1} \land \cdots \land x_{t_m} \land \not y_{t_1} \land \cdots \land \not y_{t_n} \) and \( |S| < l \)
• $C_4 = (\sim x, y, -) \in E_a$ where $y = l \leq \{t_1, \ldots, t_m\} \in O$ and a set of atoms $S_4 \subseteq \{t_1, \ldots, t_m\}$ such that $\forall t = (x_{t_1}, \ldots, x_{t_m}, \ not \ y_{t_1}, \ldots, \ not \ y_{t_n}) \in S_4, A \models x_{t_1} \wedge \ldots \wedge x_{t_m} \wedge \ not \ y_{t_1} \wedge \ldots \wedge \ not \ y_{t_n}$ and $|S_4| \geq l$, and

• for every rule $r \in P$ whose head is $x$ we have that $r^+ \cap X^- \neq \emptyset$ or $r^- \cap X^+ \neq \emptyset$ or body($r$) contains choice atoms
  1. $c_1 = l \{p_1 : q_1, \ldots, p_n : q_n\} u \in r^+$ if we have $C_1$;
  2. $c_2 = l \{p_1 : q_1, \ldots, p_n : q_n\} u \in r^-$ if we have $C_2$;
  3. $c_3 = l \{p_1 : q_1, \ldots, p_n : q_n\} \in r^+$ or $c_3' = \{p_1 : q_1, \ldots, p_n : q_n\} l - 1 \in r^-$ if we have $C_3$. Note that in $c_3'$, the upper bound $c_3'_{\text{ub}} = l - 1$;
  4. $c_4 = l \{p_1 : q_1, \ldots, p_n : q_n\} \in r^-$ or $c_4' = \{p_1 : q_1, \ldots, p_n : q_n\} l - 1 \in r^+$ if we have $C_4$. Note that in $c_4'$, the upper bound $c_4'_{\text{ub}} = l - 1$.

– Any cycle containing $\sim x$ in $DG_a$ contains only nodes in $N_a \cap \{u \mid u \not\in A\}$.

4. The exp(ASP⁺) System

In this section, we will focus on describing how the three main tasks in Sec. 2 are implemented. exp(ASP⁺) uses a data structure, associative array, whose keys can be choices, tuples, constraints, or literals. For an associative array $D$, we use $D.\text{keys}()$ to denote the set of keys in $D$ and $k \mapsto D[k]$ to denote that $k$ is associated to $D[k]$. To illustrate the different concepts, we will use the program $P_1$ that contains a choice atom and a constraint rule as follows:

(r1) a :- not b, not c. (r2) b :- c, a.
(r3) c :- not a. (r4) b :- m(1).
(r5) 1 {m(X) : n(X)} 1 :- c. (r6) n(1..2).

4.1. Preprocessing

Similar to exp(ASP), a program is preprocessed to maintain facts and as many ground rules as possible by using the -text and -keep-facts options and replacing facts with the external statements. The aspif representation [16] of the program is then obtained and processed, together with the given answer set, for generating explanation graphs. The aspif statements of $P_1$ is given in Listing 1. Let us briefly discuss the aspif representation before continuing with the description of other components.

Listing 1: aspif Representation of $P_1$

```plaintext
1  asp 1 0 0
2  5 1 2
3  5 2 2
4  1 0 1 3 0 1 -4
5  1 0 1 4 0 2 -5 -3
6  1 0 1 5 0 2 4 3
7  1 0 1 6 0 1 3
8  1 0 0 0 2 7 5
9  1 1 1 7 0 2 6 1
10 1 1 1 8 0 2 6 2
```
Each line encodes a statement in aspif. Lines starting with 4, 5, and 1 are output, external, and rule statements, respectively. Atoms are associated with integers and encoded in output statements (e.g., Line 17: 1 is the identifier of n(1)). External statements help us to recognize the facts in \( P \), e.g. atom \( n(1) \). A rule statement \( r \) is of the form: \( 1 \ H \ B \), where \( H \) and \( B \) are the encoding of the head and body of \( r \), respectively. Because of page limitation, we focus on describing the rule statement whose head is a choice atom or whose body is a weight body. If the head is a choice, its encoding \( H \) has the form: \( 1 \ n \ i_{c_1} \ldots i_{c_n} \), where \( n \) is the number of head atoms and \( i_{c} \) is an integer identifying the atom \( c \). E.g. \( m(1) \) (ID = 7) and \( m(2) \) (ID = 8) are the head choices in Lines 9 and 10, respectively, which represents rule \( r_5 \).

If the body of a rule is a weight body, its encoding \( B \) has the form: \( 1 \ l \ n \ i_{a_1} \ w_{a_1} \ldots i_{a_n} \ w_{a_n} \), where \( l > 0, l \in \mathbb{N} \) is the lower bound, \( n > 0 \) is the number of literal \( a_i \)'s with \( ID = i_{a_i} \) and weight \( w_{a_i} \). E.g. Lines 13-14 contain weight bodies. Given an ID \( i \) that does not occur in any output statement [16, 14], we use \( l(i) \) to denote the corresponding literal. Constraint \( r_4 \) is shown in Line 8. It is interesting to observe that there is one additional constraint in Line 16. By tracking integer identifiers, one can notice that Line 16 states that it can not be the case that \( c \) is true (via Line 7) and \( l(13) \) can not be proven to be true. Lines 13-15 ensure that \( l(13) \) is true if \( 1\{l(9); l(10)\} \) is true and \( 2\{l(9); l(10)\} \) cannot be proven to be true. Note that \( l(9) \) and \( l(10) \) are non-empty, so we ignore their weight. Line 11 states that \( l(9) \) is true if \( m(1) \) and \( n(1) \) are true. Line 12 states that \( l(10) \) is true if \( m(2) \) and \( n(2) \) are true. Thus, the new constraint is generated from the semantics of choice rule \( r_5 \), which is added to aspif representation. Note that the grounding of rule \( r_5 \) includes the new constraint.

Given the aspif representation \( P' \) of a program \( P \), an associative array \( D_P \) is created where \( D_P = \{(t, h) \rightarrow B \mid t \in \{0, 1\}, h \in H, B = \{body(r) \mid r \in P', head(r) = h\}\} \). Here, for an element \((t, h) \rightarrow B \) in \( D_P \), \( t \) is the type of head \( h \), either disjunction \( (t = 0) \) or choice \( (t = 1) \). Furthermore, for an answer set \( A \) of \( P \), \( E_{r(P)} = \{k \mapsto V \mid k \in \{a \mid a \in A\} \cup \{\sim a \mid a \notin A\}, V = \{support(k, r) \mid r \in P\}\} \) [14].

Algorithm 1 shows how constraints are processed given the program \( P \) and its answer set \( A \). The outcome of this algorithm is an associative array \( E_c \). First, \( V_c \)—the set of constraints (the bodies of constraints)—is computed. Afterwards, for each body \( B \) of a constraint \( r \) in \( V_c \), \( violation \) and \( support \) are computed. Each element in \( violation \) requires some trigger constraint to falsify the body \( B \), which are those in \( support \). Each \( choice\_support \) encodes a support for a choice.
Algorithm 1: constraint_preprocessing($D, A$)

Input: $D$ - associative array of rules (this is $D_P$), $A$ - an answer set
1 $V_c = \{B \mid D((0, h)) = B \land h = \emptyset\}$
2 $E_c \leftarrow \{\emptyset \mapsto \emptyset\}$ // Initialize an empty associative array $E_c$: $E_c.key() = \emptyset$
3 for $B \in V_c$ do
4   violation $\leftarrow \{a \mid a \in r^+ \land a \in A\} \cup \{\neg a \mid a \in r^- \land a \notin A\}$
5   support $\leftarrow \{\neg a \mid a \in r^+ \land a \notin A\} \cup \{a \mid a \in r^- \land a \in A\}$
6   if choice atom $x$ in $B$ and $S = \{(c, q) \mid c \in x, q \equiv c, A \models c \land q\}$ then
7       $\mathcal{X} = \{(c, q) \mid c \in x, q \equiv c\}$
8       if $x = l \{p_1 : q_1, \ldots, p_n : q_n\} u \in r^+ \land |S| < l \text{ or } |S| > u$ then
9           \[ \text{choice support} \leftarrow \{"\neg (l \leq \mathcal{X} \leq u)"\} \]
10      \[ \text{choice support} \leftarrow \{"l \leq \mathcal{X} \leq u"\} \]
11      if $x = l \{p_1 : q_1, \ldots, p_n : q_n\} u \in r^- \land l \leq |S| \leq u$ then
12         \[ \text{choice support} \leftarrow \{"\neg (l \leq \mathcal{X})"\} \]
13         \[ \text{choice support} \leftarrow \{"l \leq \mathcal{X}"\} \]
14      if $x = l \{p_1 : q_1, \ldots, p_n : q_n\} \in r^+ \text{ or } x = \{p_1 : q_1, \ldots, p_n : q_n\} l - 1 \in r^-$
15         and $|S| < l$ then
16            \[ \text{choice support} \leftarrow \{"\neg (l \leq \mathcal{X})"\} \]
17            \[ \text{choice support} \leftarrow \{"l \leq \mathcal{X}"\} \]
18      if $x = l \{p_1 : q_1, \ldots, p_n : q_n\} \in r^- \text{ or } x = \{p_1 : q_1, \ldots, p_n : q_n\} l - 1 \in r^+$
19         and $|S| \geq l$ then
20            \[ \text{choice support} \leftarrow \{"\neg (l \leq \mathcal{X})"\} \]
21            \[ \text{choice support} \leftarrow \{"l \leq \mathcal{X}"\} \]
22   support $\leftarrow$ support $\cup$ choice_support
23 if $S \neq \emptyset$ then
24     $E_c[\text{choice support}] \leftarrow [S]$
25     $E_c[p_i] \leftarrow \{\"* \text{True}\"\}$ such that $p_i \in S$
26 else
27     $E_c[\text{choice support}] \leftarrow \{\"* \text{Empty}\"\}$
28 for $v \in$ violation do
29     Append \{(triggered_constraint($v$))\} to a list $E_c[v]$
30     $E_c[\text{triggered_constraint($v$)}] \leftarrow [c \cup \{s\} \mid s \in \text{support}, c \in E_c[\text{triggered_constraint($v$)}]]$
31 return $E_c$

atom. triggered_constraint($v$), where $v \in$ violation, is assigned to support the explanation of $v$ (Line 23), and support is used for the explanation of triggered_constraint($v$) to justify the satisfaction of constraints containing $v$ (Line 24). For $\{p_1 : q_1, \ldots, p_n : q_n\}$ in a choice atom $x$, we write $\mathcal{X} = \{(c, q) \mid c \in x, q \equiv c\}$ (e.g., Line 7).

During the preprocessing, the set of all negation atoms in $P$, $\text{NANT}(P) = \{a \mid a \in r^- \land r \in P\}$ [12, 14], is computed. For $P_1$ and the answer set $A = \{n(1), n(2), c, m(1)\}$, we have $\text{NANT}(P_1) = \{a, b, c\}$.

Example 1. For program $P_1$ and its answer set $A = \{n(1), n(2), c, m(1)\}$, the output of preprocessing, $E_r(P_1)$ (left) and $E_c(P_1)$ (right), are as follows:
\[ E_{r(P_1)} = \{
\begin{align*}
c & : [\sim\{a\}], \\
\sim a & : [{c}], \\
\sim b & : [\sim\{a\}], \\
m(1) & : [\text{triggered\_constraint}(m(1))], \\
\text{triggered\_constraint}(m(1)) & : [\sim\{b\}], \\
c & : [\text{triggered\_constraint}(c)], \\
\text{triggered\_constraint}(c) & : [1\leq (m(1),n(1)), (n(2),m(2))\leq 1 : [(m(1),n(1))]], \\
(m(1),n(1)) & : [{\text{True}}]
\end{align*}
\]  
\[ E_{c(P_1)} = \{
\begin{align*}
m(1) & : [\text{triggered\_constraint}(m(1))], \\
\text{triggered\_constraint}(m(1)) & : [\sim\{a\}], \\
c & : [\text{triggered\_constraint}(c)], \\
\text{triggered\_constraint}(c) & : [1\leq (m(1),n(1)), (n(2),m(2))\leq 1 : [(m(1),n(1))]], \\
(m(1),n(1)) & : [{\text{True}}]
\end{align*}
\]  

\( E_{r(P_1)} \) shows that the supported set of two choice heads \( m(1) \) and \( m(2) \) contains \( \text{choice} \) and \( \sim\text{choice} \), respectively, which depends on their truth values and the value of their bodies. \( E_{c(P_1)} \) shows that atom \( b \) in \( r_4 \) makes the constraint satisfied while \( m(1) \) does not support the constraint. Thus, \( \sim b \) is the support set of \( \text{triggered\_constraint}(m(1)) \) and \( \text{triggered\_constraint}(m(1)) \) is the support set of \( m(1) \). For the additional constraint of \( P_1 \), \( l(9) \) is true (encoded in \( (m(1),n(1)) \)) w.r.t \( A \), resulting the constraint is satisfied. The truth value of \( c \) does not contribute to making the constraint satisfied. Thus, \( \text{triggered\_constraint}(t(c)) \) is added to the explanation of \( c \).

### 4.2. Minimal assumption set

The pseudocode of computing minimal assumption sets is shown in Algorithm 2. A tentative assumption set \( TA \) [12, 14] is computed (Line 1), which is a superset of minimal assumption sets. The atoms in \( TA \) are false in \( A \) and do not belong to the set of cautious consequences, denoted by \( C(P) \), of the program \( P \). The minimal assumption set \( U \) is computed in Line 6, which is the union of outputs from functions \( \text{derivation} \) and \( \text{dependency} \). Note that several minimal assumption sets w.r.t an answer set \( A \) of \( P \) may exist.

Function \( \text{derivation} \), \( E_{r(P)} \) in Sec. 4.1 is utilized to compute all derivation paths \( M \) of \( a \in TA \) (Line 12). Then, the derivation paths in \( M \) are examined to see whether the cycle condition in the definition of the explanation graph is satisfied (Lines 13-16). During this process, other tentative assumption atoms, that are derived from \( a \), are stored in a set \( D \), which is appended to \( DA[a] \) (\( DA \) is an associative array). If \( a \) is derivable from other atoms in \( TA \), then the relation of \( a \) will be checked in function \( \text{dependency} \) and \( a \) is stored in a set \( T' \). A set \( T = TA \setminus T' \) contains atoms that must be assumed to be false (Lines 17-19).

We calculate sets of minimal atoms that break all cycles among tentative assumption atoms via \( DA \). This is done by the function \( \text{dependency} \).

**Example 2.** Let us reconsider the program \( P_1 \) and Example 1.

- For the program \( P_1 \), we have: \( TA = \{a, b\} \)
- From \( E_{r(P_1)} \) in Example 1, atom \( a \) is not derivable from other atoms in \( TA \) while atom \( b \) is derivable from an atom in \( \{a\} \). Thus, we have \( T' = \{b\}, T = \{a\} \) and \( DA = b : [\{a\}] \). Also, there is no cycle between \( a \) and \( b \), so \( \min(B) = \emptyset \). As a result, the minimal assumption set is \( U = \{a\} \).
Algorithm 2: assumption_func(C(P), NANT(P), E_r)

**Input:** C(P) - A cautious consequence of a program P, NANT(P) - A set of negative atoms in P, E_r - A associative array computed in Sec. 4.1

1. \( TA = \{ a | a \in NANT(P) \land a \notin A \land a \notin C(P) \} \)
2. \( DA \leftarrow \{ \emptyset \mapsto \emptyset \} \)
3. \((T, DA) = derivation(TA, E_r)\)
4. \( D = dependency(DA) \)

for \( M \in D \) do

6. \( U \leftarrow M \cup T \)
7. \( TU \leftarrow TU \cup \{U\} \)
8. return \( TU \)

function derivation(TA, E)

for \( a \in TA \) do

for \( N \in M \) do

Find \( D = \{ b | b \in TA \land b \neq a \} \) such that \( b \) is derived from \( a \)

if there is no negative cycles in derivation path \( N \) then

Append \( D \) to a list \( DA[a] \)

if \( |DA[a]| \neq 0 \) then

\( T' \leftarrow T' \cup \{a\} \)

end if

end if

end for

end for

return \((T, DA)\)

function dependency(DA)

\( D \leftarrow \{ DA_i | DA_i = k \mapsto V \land V \in DA[k] \land \forall k \in DA.keys(), k \in DA_i.keys() \land (k \mapsto V_1 \in DA_i \land k \mapsto V_2 \in DA_i) \Rightarrow V_1 = V_2 \} \)

\( B \leftarrow \emptyset \)

for \( DA_i \in D \) do

Find all dependency cycles DC among tentative assumption atoms in \( DA_i \)

\( B \leftarrow B \cup \{\{j_1, ..., j_n\} | (j_1, ..., j_n) \in J_1 \times .. \times J_n \land n = |DC| \land J_i \in DC\} \)

end for

min(B) \leftarrow \{ M | \forall C \in B, C \neq M \Rightarrow |M| \leq |C| \}

return \( \min(B) \)

4.3. ASP-based explanation system

In this section, we describe how the explanation graph is generated by utilizing \( E_r, E_c \) from Sec. 4.1 and the minimal assumption set \( U \) from Sec. 4.2. In order to leverage the algorithm from the previous work, \( E_r \) and \( E_c \) are combined into a dictionary \( E \) as follows: \( E = \{ k \mapsto V | k \in E_r.keys() \cup E_c.keys(), V = r \cup c \text{ such that } r \in E_r[k] \text{ and } c \in E_c[k] \} \). Note that \( r = \emptyset \) if \( \exists k \in E_r.keys() \) and \( c = \emptyset \) if \( \exists k \in E_c.keys() \). Given \( E \), the algorithm from [14] will find the explanation graph of literal in \( P \), taking into consideration the additional types of nodes and
Example 3. For program \( P_1 \), the explanation graph of \( m(1) \) is shown in Fig. 2.

As can be seen from Fig. 2, a justification for \( m(1) \) depends positively on \( c \) and \( n(1) \). A choice head \( m(1) \) is chosen to be true. The constraint containing \( m(1) \) is satisfied by \( A \) because of the truth value of \( b \). The constraint containing \( c \) is satisfied by \( A \) because the conjunction of \( n(1) \) and \( m(1) \) is true. The additional constraint comes from the semantics of choice rule \( r_5 \) as we mentioned in Sec. 4.1.

![Figure 2: Explanation of \( m(1) \)](image)

4.4. Illustration

We illustrate the application of our updated system, \( \text{exp}(\text{ASP}^c) \), to the graph coloring problem. We use a solution of the problem where each node is assigned a unique color by the choice rule:

\[
1\{\text{colored}(X, C) : \text{color}(C)\} \leftarrow \text{node}(X).
\]

Fig. 3 shows the explanation graph of \( \text{colored}(1, \text{red}) \). Unlike Fig. 1, Fig. 3 shows that a choice head \( \text{colored}(1, \text{red}) \) is chosen to be true while two choice heads, \( \text{colored}(3, \text{red}) \) and \( \text{colored}(2, \text{red}) \), are chosen to be false, which are represented via orange dotted links (link •). Fig. 3 displays the constraint that \( \text{node}(1) \) must assign a different color with \( \text{node}(3) \) and \( \text{node}(2) \). This shows via the links from \( \text{colored}(1, \text{red}) \) to \( \sim \text{colored}(2, \text{red}) \) and \( \sim \text{colored}(3, \text{red}) \) connected through \( \text{triggered}\_\text{constraint}(\text{colored}(1, \text{red})) \) (green dotted link ◇). Also, the triggered constraints of each \( \text{node}(1), \text{node}(2) \) and \( \text{node}(3) \) such that each node is assigned exactly one color are shown via the aggregate functions in the node labels (blue solid link ⊕).
5. Conclusion

In this paper, we proposed an extension of our explanation generation system for ASP programs, exp(ASPc), which supports choice rules and includes constraint information. Our future goal is to extend exp(ASPc) so that it can deal with other clingo constructs like the aggregates #sum, #min, #max, etc.

Acknowledgments

The second author would like to acknowledge the partial support of the NSF 1812628 grant. Portions of this publication and research effort are made possible through the help and support of NIST via cooperative agreement 70NANB19H102.

References


