Explainable Problem in *clingo-dl* Programs

Van Nguyen,[1] **Tran Cao Son,** [1] **William Yeoh** [2]
[1] New Mexico State University, Las Cruces NM 88003, USA
[2] Washington University in St. Louis, St. Louis MS, USA
vnguyen,tson@cs.nmsu.edu, wyeoh@wustl.edu

Abstract

Research in explainable planning is becoming increasingly important as human-AI collaborations become more pervasive. An explanation is needed when the planning system's solution does not match the human's expectation. In this paper, we introduce the explainability problem in *clingo-dl* programs (XASP-D) because *clingo-dl* can effectively work with numerical scheduling, a problem similar to the explainable planning.

Introduction

In *Explainable AI Planning* (XAIP) (Chakraborti, Sreedharan, and Kambhampati 2020) , the planning agent needs to find ways to ensure that its plan is understood and accepted by human users. Inspired by the rapid development of XAIP in the last couple of years (Chakraborti, Sreedharan, and Kambhampati 2020) and the need for dealing with different optimal criteria relating to numerical constraints (e.g., action costs), we propose the *explainable problem in clingo-dl programs* (or XASP-D). *clingo-dl*, first introduced by Janhunen et al. (2017), is a syntactic extension of the well-known *answer set programming* (ASP) paradigm (Marek and Truszczyński 1999; Niemelä 1999) with difference logic. Intuitively, in a XASP-D problem, the agent and the human knowledge bases are *clingo-dl* programs Π_a and Π_h, respectively, and the agent's goal is to explain to the human a set of atoms $\mathcal{I}$ which belong to one of its answer sets I^*. To do so, the agent needs to identify an explanation $\epsilon = (\epsilon^+, \epsilon^-)$ such that $\epsilon^+ \subseteq \Pi_a$, $\epsilon^- \subseteq \Pi_h$, and $\Pi_h \setminus \epsilon^- \cup \epsilon^+$ has an answer set containing $\mathcal{I}$. We make the same assumption as in XAIP that the agent is aware of the knowledge base of the human.

Problem Formulation

Given a *clingo-dl* program Π and its answer set I, we use I_r and I_{dl} to denote, respectively, the set of regular atoms and difference logic assignments (dl/2) in I. Assume that the human's and the agent's perspectives of a problem associated with their reasoning engine are respectively encoded in *clingo-dl* program Π_h and Π_a.

An *explainable problem* in *clingo-dl* programs (XASP-D) is defined by a tuple $\langle \mathcal{I}, I^*, \Pi_a, \Pi_h \rangle$, where I^* is an answer set of Π_a and $\mathcal{I} \subseteq I^*_{dl}$ is the set of dl-assignments that the agent wants to explain to the human. We observe that the focus is on explaining a set of dl-assignments. The reason for this choice is twofold. First, *clingo-dl* has been successfully used in solving scheduling problem and we would like to focus on explainable scheduling. Second, the paper by Nguyen et al. (2020) already works with regular atoms.

A *solution* for a XASP-D problem $\langle \mathcal{I}, I^*, \Pi_a, \Pi_h \rangle$ is a pair (ϵ^+, ϵ^-) such that $\epsilon^+ \subseteq \Pi_a$, $\epsilon^- \subseteq \Pi_h$, and $\hat{\Pi}_h = \Pi_h \setminus \epsilon^- \cup \epsilon^+$ has an answer set containing $\mathcal{I}$. Intuitively, ϵ^+ denotes the set of rules that is going to be added to Π_h and ϵ^- denotes the set of rules that is going to be removed from Π_h. The fact that $\mathcal{I}$ is contained in an answer set of Π_h indicates that the human can accept the agent's explanation. The goal of this paper is to compute solutions for XASP-D problems. We assume that *clingo-dl* programs for solving scheduling problems can be developed in such a way that dl-atoms only occur in the heads of rules. For a dl-programs Π, we use $\mathcal{B}_r(\Pi)$ to denote the set of regular atoms in the language of Π.

Solving XASP-D Problems for DL-Splittable Π_a and Π_h

In this section, we propose an approach to solving XASP-D problems of the form $\langle \mathcal{I}, I^*, \Pi_a, \Pi_h \rangle$ when Π_h and Π_a are dl-splittable programs.

Using splitting set theorem (Lifschitz and Turner 1994), it is easy to see that for a dl-splittable program Π, $U = \mathcal{B}_r(\Pi)$ is a splitting set of Π. Furthermore, the *bottom* of Π wrt. U is $b_U(\Pi) = \{r \mid r \in \Pi \text{ and } atoms(r) \subset U\}$ and the *top* of Π relative to U is $t_U(\Pi) = \Pi \setminus b_U(\Pi)$. By splitting the program using the set of regular atoms, we have that the *bottom* program contains the set of rules that have no dl-atom, and the *top* program contains the set of rules whose heads are dl-atoms. For convenience, we will denote with U_a and U_h the set $\mathcal{B}_r(\Pi_a)$ and $\mathcal{B}_r(\Pi_h)$, respectively.

Algorithm 1 computes a solution for XASP-D problems with dl-splittable programs. The algorithm takes as an input a XASP-D problem with dl-splittable programs and returns one of its solutions. Observe that we use $+=$ between sets and elements with the similar meaning in C-programming.

In Algorithm 1, lines 3-4 compute top_h and top_a. Given that we split the programs using their regular atoms, top programs of the human and the agent programs contain collections of rules that have difference atoms in their heads, i.e., top_h and top_a contain only difference logic constraints from the human and the agent, respectively. As generating a dl-atom in an answer set requires that the corresponding dl-term appears in the heads of some rules, it is clear that a part of the solution for the XASP-D problem is in the set of rules with dl-atoms in the heads. Line 6 computes the set $potential$ which contains possible additions or deletions of dl-constraints in top_h. Lines 8 computes $select$ set which is a subset of $potential$, such that there exists an answer set of the new top program $\widehat{top_h}$ that contains all assignments in $\mathcal{I}$. For any set x of rules (e.g., $potential$, $select$, or any set in later algorithms), we use x^+ and x^- to denote the set of rules that should be added to or removed from Π_h, respectively. Lines 9-18 find changes in the bottom program of the human to support the top program.

Observe that different strategies can be employed in the selection of the set $select$ (Line 8). We have implemented two strategies:

- *random*: a **while-loop** is implemented, which starts with $\emptyset$ and, at each iteration, adds one element from $potential$ to $select$ ($^+$ and $^-$ accordingly) until $\widehat{top_h}$ has an answer set containing $\mathcal{I}$. Observe that this process will eventually terminate since $select = potential$ is of course a potential answer.
- *trivial*: $select = potential$. This is a trivial choice. It avoids the computation requires to identify a subset of $potential$.

We prove some properties of Algorithm 1.

Proposition 1. *$\widehat{top_h}$ has some answer set I such that $\mathcal{I} \subseteq I$.*

Proof. Trivial due to the selection in Line 8, described above. □

Proposition 2. *Given $\hat{\Pi}_h$ which is the new human's program produced by Algorithm 1, $\widehat{top_h} \subseteq \{head(r) \mid r \in t_{U_{\hat{h}}}(\hat{\Pi}_h)\}$ where $U_{\hat{h}} = \mathcal{B}_r(\hat{\Pi}_h)$.*

Proof. Since γ is the set of regular rules, the set of rules with dl-atoms in the head of $\Pi_h \setminus \gamma$ equals the the set of rules with dl-atoms in the head of Π_h. This, together with the fact that $\widehat{top_h} = top_h \setminus select^- \cup select^+$ and ϵ^+ (ϵ^-) is the set of rules whose head are in $select^+$ ($select^-$), proves the conclusion of the proposition. □

Proposition 3. *I^*_{dl} is an answer set of top_a.*

Proof. Trivial due to the splitting set theorem. □

Proposition 4. *$\hat{\Pi}_h$ has an answer set containing $\mathcal{I}$.*

Proof. Observe that due to Lines 9, 10, and 14 of Algorithm 1, $b_{U_{\hat{h}}}(\hat{\Pi}_h)$ has I^*_r as an answer set. $e_{U_{\hat{h}}}(t_{U_{\hat{h}}}(\hat{\Pi}_h), I^*_r)$ contains rules whose head belong to $\widehat{top_h}$. From Proposition 1, it follows that $e_{U_{\hat{h}}}(t_{U_{\hat{h}}}(\hat{\Pi}_h), I^*_r)$ has some answer set Y containing $\mathcal{I}$. By the splitting theorem, $I^*_r \cup Y$ is an answer set of $\hat{\Pi}_h$, which is one answer set satisfying the conclusion of the proposition. □

Algorithm 1: `explain`$(\mathcal{I}, I^*, \Pi_a, \Pi_h)$

1 Input: Π_a and Π_h: dl-splittable programs, I^*: answer set of Π_a, and $\mathcal{I} \subseteq I^*_{dl}$
2 Output: $\hat{\Pi}_h$, $\epsilon = (\epsilon^+, \epsilon^-)$: solution of $\langle \mathcal{I}, I^*, \Pi_a, \Pi_h \rangle$
3 $top_h = \{head(r) \mid r \in t_{U_h}(\Pi_h)\}$
4 $top_a = \{head(r) \mid r \in t_{U_a}(\Pi_a) \wedge head(r)$ and $body(r)$ are True wrt $I^*)\}$
5 Compute the set AS of all answer sets of top_h, if top_h is unsatisfiable, $AS = \emptyset$
6 $potential^+ = top_a \setminus top_h$; $potential^- = top_h \setminus top_a$
7 $select^+ = select^- = \emptyset$
8 Find $select^+ \subseteq potential^+$ and $select^- \subseteq potential^-$ such that $\exists$ an answer set I of $\widehat{top_h} = top_h \setminus select^- \cup select^+, \mathcal{I} \setminus I = \emptyset$
9 $\gamma = \{r \mid r \in \Pi_a \wedge head(r)$ and $body(r)$ are True wrt $I^*_r\}$
10 $\epsilon^+ = \gamma \setminus \Pi_h$
11 **for** $p \in select^+$ **do**
12 $\quad \epsilon^+ \mathrel{+}= \{r \in \Pi_a \mid head(r) = p \wedge I^*_r \models body(r)\}$
13 **end**
14 $\epsilon^- = \Pi_h \setminus \gamma$
15 **for** $p \in select^-$ **do**
16 $\quad \epsilon^- \mathrel{+}= \{r \in \Pi_h \mid head(r) = p\}$
17 **end**
18 **return** $\hat{\Pi}_h = \Pi_h \setminus \epsilon^- \cup \epsilon^+$ and (ϵ^+, ϵ^-)

References

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2020. The emerging landscape of explainable ai planning and decision making. *arXiv preprint arXiv:2002.11697* .

Janhunen, T.; Kaminski, R.; Ostrowski, M.; Schellhorn, S.; Wanko, P.; and Schaub, T. 2017. Clingo goes linear constraints over reals and integers. *Theory and Practice of Logic Programming* 17(5-6): 872–888.

Lifschitz, V.; and Turner, H. 1994. Splitting a Logic Program. In Hentenryck, V.; and Pascal, eds., *Proceedings of International Conference on Logic Programming (ICLP)*, 23–37. URL http://www.cs.utexas.edu/users/ai-lab?lif94e.

Marek, V.; and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-year Perspective*, 375–398.

Nguyen, V.; Son, T. C.; Stylianos, V. L.; and Yeoh, W. 2020. Conditional Updates of Answer Set Programming and Its Application in Explainable Planning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1954–1956.

Niemelä, I. 1999. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25(3,4): 241–273.