Conditional Updates of Answer Set Programming and Its Application in Explainable Planning*

Extended Abstract

Van Nguyen and Tran Cao Son
New Mexico State University
Las Cruces, NM
{vnguyen,tson}@cs.nmsu.edu

Stylianos Loukas Vasileiou and William Yeoh
Washington University in St. Louis
St. Louis, MO
{v.stylianos,wyeoh}@wustl.edu

ABSTRACT

In explainable planning, the planning agent needs to explain its plan to a human user, especially when the plan appears infeasible or suboptimal for the user. A popular approach is called *model reconciliation*, where the agent reconciles the differences between its model and the model of the user such that its plan is also feasible and optimal to the user. This problem can be viewed as a more general problem as follows: Given two knowledge bases π_a and π_h and a query q such that π_a entails q and π_h does not entail q, where the notion of entailment is dependent on the logical theories underlying π_a and π_h, how to change π_h – given π_a and the support for q in π_a – so that π_h does entail q. In this paper, we study this problem under the context of answer set programming. To achieve this goal, we (1) define the notion of a *conditional update* between two logic programs π_a and π_h with respect to a query q; (2) define the notion of an explanation for a query q from a program π_a to a program π_h using conditional updates; (3) develop algorithms for computing explanations; and (4) show how the notion of explanation based on conditional updates can be used in explainable planning.

KEYWORDS

Explainable Planning; Answer Set Programming

ACM Reference Format:
Van Nguyen and Tran Cao Son and Stylianos Loukas Vasileiou and William Yeoh. 2020. Conditional Updates of Answer Set Programming and Its Application in Explainable Planning. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 3 pages.

1 LOGIC PROGRAMMING

Answer set programming (ASP) [10, 11] is a declarative programming paradigm based on logic programming under the answer set semantics. A logic program Π is a set of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_m,\ not\ a_{m+1}, \ldots,\ not\ a_n$$

where $0 \leq m \leq n$, each a_i is an atom of a propositional language, and *not* represents (default) negation.Intuitively, a rule states that if all positive literals a_i are believed to be true and no negative literal *not* a_i is believed to be true, then a_0 must be true. If a_0 is omitted, the rule is called a *constraint*. If $n = 0$, it is called a *fact*. For a rule r, $head(r)$ denotes a_0; $pos(r)$ and $neg(r)$ denote the set $\{a_1, \ldots, a_m\}$ and $\{a_{m+1}, \ldots, a_n\}$, respectively. $atoms(r)$ denotes the set of all atoms in r, viz. $\{head(r)\} \cup pos(r) \cup neg(r)$; and, $atoms(\Pi)$ denotes the set of all atoms of Π. $heads(\Pi)$ ($negs(\Pi)$) denotes the set of atoms occurring in the head of rules of Π (negative literals of Π).

Let Π be a program. $I \subseteq atoms(\Pi)$ is called an interpretation of Π. For an atom a, a (resp. *not* a) is satisfied by I, denoted by $I \models a$ (resp. $I \models not\ a$), if $a \in I$ (resp. $a \notin I$). A set of literals S is satisfied by I ($I \models S$) if I satisfies each literal in S. A rule r is satisfied by I if $I \not\models body(r)$ or $I \models head(r)$. I is a *model* of a program if it satisfies all its rules. An atom a is *supported* by I in Π if there exists $r \in P$ such that $head(r) = a$ and $I \models body(r)$. The *reduct* of Π w.r.t. I (denoted by Π^I) is the program obtained from Π by deleting *(i)* each rule r such that $neg(r) \cap I \neq \emptyset$, and *(ii)* all negative literals in the bodies of the remaining rules. I is an *answer set* [5] of Π if I is the least Herbrand model of Π^I [14], which is the least fixpoint of the operator T_Π defined by $T_\Pi(I) = \{a \mid \exists r \in \Pi, head(r) = a, I \models body(r)\}$ and is denoted by $lfp(T_\Pi)$.

Given an answer set I of Π and an atom q, a justification for q wrt. I is a set of rules $S \subseteq \Pi$ such that $I \models body(r)$ for $r \in S$ and $q \in lfp(T_{S^I})$. A justification S for q wrt. I is minimal if there exists no proper subset $S' \subset S$ such that S' is also a justification for q wrt. I. It is easy to see that if S is a minimal justification for q wrt. I then $negs(S) \cap heads(S) = \emptyset$ and $heads(S)$ is an answer set of S.

2 PLANNING USING ASP

Answer set planning refers to answer set programming in planning [9]. It has been shown by Gebser et al. [4] that answer set planning, combined with good heuristics, can perform at the highest level of state-of-the-art planning systems.

A planning problem – as described using PDDL [6] – is a triple (I, G, D), where I and G encode the initial state of the world and the goal, respectively; and D (the domain) specifies the actions and their preconditions and effects. Given a problem $P = (I, G, D)$, answer set planning translates it into a program $\pi(P, n)$ to compute solutions of P, where n is constant indicating the maximal length of solutions that we are interested in (i.e., horizon). Program $\pi(P, n)$ consists of different groups of rules:

- **Facts:** These atoms define object constants, types of objects, actions, the initial state, and the goal state.
- **Reasoning About Effects of Actions:** Rules in this group make sure that an action can only be executed if all of its conditions are true and all of the effects of the actions become true. We use $h(l, t)$ to denote that l is true at step t for $1 \leq t \leq n$.

*This research is partially supported by NSF grants 1345232, 1619273, 1757207, 1829859, 1812619, and 1812628.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

- **Goal Enforcement and Action Generation:** The rules in this group generates action occurrences and ensure that only valid plans are generated.

3 EXPLAINABLE PLANNING

In *explainable planning* (XAIP) problems [7], the planning agent needs to find ways to ensure that its plans are understood and accepted by human users. As the model or knowledge base of the robot differs from that of the human users, a plan that may be optimal in the model of the robot may be suboptimal or, worse, infeasible in the model of the human user. Researchers have approached this problem from two perspectives. The first is by enforcing that the robot finds *explicable* plans (i.e., plans that are optimal or feasible in the model of the human user) [8, 15]. The second is for the robot to provide *explanations* to the human user and *reconciling* their two models such that the plan of the robot is also optimal in the reconciled model of the human user [3, 12, 13]. There is also recent work in balancing both approaches [1, 2].

In an XAIP problem, a planning problem $P = (I, G, D)$ is given, which is identical to the robot model $P_a = (I_a, G_a, D_a)$. The human model of the planning problem $P_h = (I_h, G_h, D_h)$ might be different from the model of the robot. The focus of this paper is in the *model reconciliation process*, i.e., to bring the human's model closer to the robot's model by means of explanations in the form of model updates. Given P_a and P_h, a *model reconciliation problem* (MRP) is defined by a tuple $\langle \pi^*, P_a, P_h \rangle$, where π^* is a cost-minimal solution for P_a. A solution for an MRP is a multi-model explanation ϵ, which creates a model P_h^* from P_a and P_h such that π^* is also a cost-minimal solution of P_h^* by inserting to P_h (or removing from P_h) some initial conditions, action preconditions, action effects, or goals. It is required that the changes in the model of the human must be consistent with the robot's model.

4 EXPLANATIONS USING ASP

Let π_a be the program of the robot, π_h be the program of the human, and q be an atom of π_a such that $\pi_a \mathrel{|\!\sim} q$ and $\pi_h \not\mathrel{|\!\sim} q$. Assume that the robot wishes to explain to the human that q, representing a plan, is true. The robot could do so by identifying an answer set I supporting q and explaining to the human by presenting a set of rules $\lambda \subseteq \pi_a$, which might be a justification for q wrt. I, such that an update of π_h by λ given I will allow the human to accept that q is entailed. In other words, the process of updating π_h by λ given I should result in a new program, denoted by $\pi_h \otimes_I \lambda$ such that $\pi_h \otimes_I \lambda \mathrel{|\!\sim} q$. Therefore, we define the operator $\otimes$ before we discuss the explanation process.

Definition 4.1 (Conditional Update). Let π_a and π_h be two programs. Further, let I be an answer set of π_a and $\lambda \subseteq \pi_a$. The *conditional update* of π_h with respect to λ and I is the program $\pi_h' \cup \lambda$, denoted by $\pi_h \otimes_I \lambda$, where π_h' is the collection of rules from $\pi_h \setminus \lambda$ such that (*i*) $head(r) \in I$ and $neg(r) \cap I = \emptyset$ or (*ii*) $neg(r) \cap heads(\lambda) \neq \emptyset$.

Let π_a and π_h to denote two arbitrary but fixed programs and $q \in atoms(\pi_a)$ such that $\pi_a \mathrel{|\!\sim} q$ and $\pi_h \not\mathrel{|\!\sim} q$.

Definition 4.2 (Explanation). A subprogram $\epsilon \subseteq \pi_a$ is a *lp-explanation* for q from π_a to π_h wrt. an answer set I of π_a (or an *lp-explanation* for q wrt. I) if $\pi_h \otimes_I \epsilon \mathrel{|\!\sim} q$. ϵ is a *minimal lp-explanation* for q wrt. I if there exists no proper subset ϵ' of ϵ s.t. ϵ' is an lp-explanation for q wrt. I. ϵ is a *lp-explanation with justification* if ϵ^I contains a justification for q wrt. I. Finally, if $\{q \leftarrow\}$ is an lp-explanation for q, we call it a *trivial lp-explanation.*

Algorithm 1: $\mathtt{LP-Explanation}(\pi_a, \pi_h, q)$

Input: Programs π_a, π_h, atom q
Output: An explanation ϵ for q

1 **if** $\pi_a \cup \{\leftarrow\ not\ q\}$ *has no answer set* **then return** nil
2 Let I be an answer set of $\pi_a \cup \{\leftarrow\ not\ q\}$
3 Compute $\Pi(\pi_a, I)$
4 Compute an answer set J of $\Pi(\pi_a, I)$
5 Compute $\epsilon = \{head(r) \leftarrow pos(r), neg(r) \mid head(r) \leftarrow pos(r), neg(r), ok(r) \in \Pi(\pi_a, I), ok(r) \in J\}$.
6 **return** $\epsilon \setminus \pi_h$ (or $(\epsilon \setminus \pi_h, \pi_h \setminus \epsilon)$)

Algorithm 2: Computing Non-Trivial LP-Explanation

1 **if** $\Pi(\pi_a, I) \setminus \{q \leftarrow\}$ *has no model* **then**
2 **return** $\{q \leftarrow\}$—% only trivial lp-explanation exists
3 Compute an answer set of J of $\Pi(\pi_a, I) \setminus \{q \leftarrow\}$

Given a program π_a and an answer set I supporting q of π_a, we define $\Pi(\pi_a, I)$ be the program such that:

- $\Pi(\pi_a, I)$ contains the constraint $\leftarrow\ not\ q$;
- for each $x \in \pi_a$ s.t. $head(x) \in I$ and $neg(x) \cap I = \emptyset$:
  - $head(x) \leftarrow pos(x), neg(x), ok(x)$ is a rule in $\Pi(\pi_a, I)$;
  - $\{ok(x)\} \leftarrow$ is a rule of $\Pi(\pi_a, I)$.
  - $\texttt{\#mimimize}\{1, X : ok(X)\}$ is a rule of $\Pi(\pi_a, I)$.
- No other rule is in $\Pi(\pi_a, I)$.

Algorithm 1 can be used for computing an lp-explanation. To compute a non-trivial lp-explanation, Line 4 is replaced by the three lines (Lines 1-3) in Algorithm 2.

The proposed notion of an lp-explanation can be used in explainable planning as follows. Let $\pi(P_a, t)$ and $\pi(P_h, t)$ be the two programs encoding the planning model of the robot and the human, respectively. Assume that $\alpha = [a_1, \ldots, a_{t-1}]$ is a plan in $\pi(P_a, t)$ and is not a plan in $\pi(P_h, t)$. This implies that $\pi_a = \pi(P_a, t) \cup occurs^*(\alpha) \mathrel{|\!\sim} goal$ and $\pi_h = \pi(P_h, t) \cup occurs^*(\alpha) \not\mathrel{|\!\sim} goal$ where $occurs^*(\alpha) = \{occurs(a_i, i) \mid i = 1, \ldots, t-1\}$. As such, an lp-explanation for the atom *goal* from π_a to π_h could explain why α is not a solution in the model of P_h. Indeed, Algorithm 1 can be used to compute an lp-explanation for the atom *goal* from π_a to π_h, i.e., an explanation for the MRP between the robot and the human. This can be used as a seed for computing complete explanations for the MRP.

5 CONCLUSIONS AND FUTURE WORK

In this abstract, we consider a general problem of updating a theory π_h so that the resulting theory $\hat{\pi}_h$ credulously entails an atom q given that q is entailed by a theory π_a using ASP by proposing the notion of conditional updates in logic programming and use it to define the notion of an explanation. We then show how it can be used to compute explanations for MRP problems. Future work includes experimentally evaluating this approach against the state of the art.

REFERENCES

[1] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2018. Explicability versus explanations in human-aware planning. In *AAMAS*. 2180–2182.

[2] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2019. Balancing Explicability and Explanations in Human-Aware Planning. In *IJCAI*. 1335–1343.

[3] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*. 156–163. https://doi.org/10.24963/ijcai.2017/23

[4] Martin Gebser, Benjamin Kaufmann, Javier Romero, Ramón Otero, Torsten Schaub, and Philipp Wanko. 2013. Domain-Specific Heuristics in Answer Set Programming. In *AAAI*. 350–356.

[5] M. Gelfond and V. Lifschitz. 1990. Logic programs with classical negation. In *LP*. 579–597.

[6] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. PDDL – the planning domain definition language.

[7] Subbarao Kambhampati. 2019. Synthesizing Explainable Behavior for Human-AI Collaboration. In *AAMAS*. 1–2.

[8] Anagha Kulkarni, Yantian Zha, Tathagata Chakraborti, Satya Gautam Vadlamudi, Yu Zhang, and Subbarao Kambhampati. 2019. Explicable Planning as Minimizing Distance from Expected Behavior. In *AAMAS*. 2075–2077.

[9] V. Lifschitz. 2002. Answer set programming and plan generation. *Artificial Intelligence* 138, 1–2 (2002), 39–54. https://doi.org/10.1016/S0004-3702(02)00186-8

[10] V. Marek and M. Truszczyński. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-year Perspective*. 375–398. https://doi.org/10.1007/978-3-642-60085-2_17

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

[12] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. 2018. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *ICAPS*. 518–526.

[13] Sarath Sreedharan, Alberto Olmo Hernandez, Aditya Prasad Mishra, and Subbarao Kambhampati. 2019. Model-Free Model Reconciliation. In *IJCAI*. 587–594.

[14] M. van Emden and R. Kowalski. 1976. The semantics of predicate logic as a programming language. *J. ACM* 23, 4 (1976), 733–742. https://doi.org/10.1145/321978.321991

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