Explaining ethical planning using ASP

Martin JEDWABNY [a,1], Pierre BISQUERT [b] and Madalina CROITORU [b]
[a] *GraphIK, University of Montpellier, France*
[b] *GraphIK, IATE, INRA, Montpellier, France*

Abstract. Ethical planning requires explanation capabilities. We demonstrate in this paper how to formalise consequentialist ethical planning with answer set programming (ASP) in order to benefit from ASP explanation approaches.

Keywords. explanation, answer set programming, ethical planning

1. Introduction

In this paper we place ourselves in the context of single agent planning and investigate the problem of how to take into account an ethical dimension to this problem. Our research hypothesis is that a logic based implementation of the problem could help foster potential for explain-ability in ethical planning. To this end this paper will explain how we can capture planning for single agents using an answer set programming (ASP) implementation.

2. Planning background

State transition systems [1] are a simple yet powerful model to represent planning domains. Formally, it is a structure $\langle S, A, R \rangle$ in which: S is a (non-empty) set of "states", A is a (non-empty) set of transition labels (also called "actions"), and R is a set of transitions i.e. $R \subseteq S \times A \times S$. A transition system can be depicted as a labelled directed graph. Every state s in S is a node of the graph. Labelled directed edges of the graph are the tuples (s, e, s') of R, which represents that the execution of action e in state s leads to the state s'.

Typically, action languages [2,3,4,5] such as STRIPS, ADL, PDDL, A, B, C, C+ and such, define the states as combinations of value assignments for a finite set of propositional (or ground first-order) symbols called "fluents". They specify a language of fluent symbols and action symbols, and their respective domains. Then, states become combinations (or conjunctions) of these symbols assigned with a member of their domains. Actions are essentially a set of assignments of action symbols to their domains.

A planning task is a 4-tuple $T = \langle V, s_0, s_*, O \rangle$ that describes all the relevant information that characterizes the states of the world, the changes actions bring in the form of transitions between states, the starting state and the conditions of a world in which the task is satisfied.

[1]Corresponding Author; E-mail: martin.jedwabny@lirmm.fr

More precisely:

- A fluent (or state variable) v is a grounded first-order atom e.g location(r_1). V is a finite set of fluents where each of them has an associated finite domain $Dom(v)$ of possible values. We call a partial state a function from members of a subset of V to their respective domains. A state is a (total) function from each fluent v in V to a member of its domain.
- s_0 is the initial state, expressed as a (total) state,
- s_* is the goal condition, expressed as a partial state, and
- O is a finite set of operators, also called actions, of the form of a tuple $a = \langle a_{pre}, a_{eff} \rangle$, which are respectively, the partial states denoting the preconditions and the effects of the action (fluents to modify in the next state).

We denote $Pre(a) = a_{pre}$ the preconditions and $Eff(a) = a_{eff}$ the effects of the action a. We will not consider derived fluents or conditional operators as in (Helmert, 2006). Given a state s and an action a, the successor state $succ(s,a)$ obtained applying $a = \langle a_{pre}, a_{eff} \rangle$ is defined iff $a_{pre} \subseteq s$. We call these the "possible" actions in situation s, and denote it $Poss(s)$. If an action is indeed possible, for every fluent $v \in V$, if there is some $d \in Dom(v)$ such that $v = d \in a_{eff}$, then $v = d \in succ(s,a)$, otherwise there must exist some $d' \in Dom(v)$ such that $v = d \in a_{pre} \cap s$.

Then, a plan $\pi = [a_1, a_2, ..., a_n]$ with $n \geq 0$ is a sequence of actions $a_i \in O$, such that $s_* \subseteq succ(a_n, ...succ(a_2, succ(a_1, s_0)))$. The embedded state transition system by such a planning task is a directed graph $S(T)$ where T is a task defined as above where:

- The nodes correspond to the set of states V, and
- There is an edge (s,a,s') with if and only if there exists an action $a = \langle a_{pre}, a_{eff} \rangle$ in T such that $s' = succ(s,a)$.

3. Ethical planning

In the context of AI decision making and planning, ethics captures the moral elements to be considered when taking an action [6,7]. As such, an ethical framework extends the process of taking action with considerations coming from one or more ethical theories to proscribe behaviours deemed as unethical and favor those that exhibit adequate moral characteristics. This is strongly related to the concept of normative ethics [8], the branch of philosophical ethics that investigates the questions that arise when considering how one ought to act, morally speaking. The three main branches of normative ethics are consequentialist, deontological and virtue ethics. All of them have been studied to some degree in the context of AI planning and decision making. Here we will consider consequentialist ethics mainly.

In consequentialist ethics, actions are evaluated upon their consequences. The precise method to determine which action is right varies between branches of consequentialist ethics. Some of the most prominent contrast points are the way in which consequences are determined, the perspective from which consequences are evaluated, and how consequences are compared.

In this work, actions and consequences are predetermined by the planning formalism. The perspective from which consequences are determined will be the welfare of society, also called utilitarianism. Utilitarianism specifies that the right action is the one that creates the most good for society, another branch being egoism, which tries to create the most good for oneself. Lastly, the way in which consequences are compared will be along the lines of plain consequentialism. Plain consequentialism advocates that the only action that is right is the one with the best consequences. Other branches are: expected consequentialism (similar to plain consequentialism, but takes uncertainty and beliefs into account) and rule consequentialism (instead of comparing outcomes of an action, it compares the action's adherence to a set of ethical rules).

In our planning task model $T = \langle V, s_0, s_*, O \rangle$, given an action $a \in O$, the consequences of a is the set $Eff(a)$ of effects (defined above) of the action.

Due to the fact that consequentialist ethics focuses on the consequences of actions only, the way in which actions are compared to one another depends solely on their consequences. This can be captured simply by a preference relation $\prec$ as in (Bonnemains et al., 2016) which we will call the consequentialist base, that compares sets of fluent assignments i.e. action consequences. A consequentialist base $\prec_c$ is a total order (binary, antisymmetric, transitive and connex relation) on sets of fluent assignments $(v = d)$ with $d \in Dom(v)$.

One practical way to define this preference relation is by introducing utilities. Utilities measure in a concrete manner the level of desireness of a consequence and assigns it a numerical value. This idea has been previously applied in the context of AI planning in (Berreby et al., 2017; Lindner et al., 2019, 2017): Given $v \in V$ and $d \in Dom(v)$, an utility function $u(v = d) \in \mathbb{R}$ for V, maps an effect (assignment of a fluent) to a real number.

An utilitarian base $\prec_u$ is defined as a consequentialist base that takes into account an utility function $u(v = d)$ and extends it to compare sets of consequences. An action is right, or permitted if and only if its consequences are the best with respect to all other possible actions. We model this as follows. Given a planning domain T, a consequentialist base $\prec_c$ of T, a situation s of T, and an action $a \in Poss(s)$, we say that a is permitted iff there is no action $a' \in Poss(s)$ such that $Eff(a) \prec_c Eff(a')$.

4. Answer Set Programming

Answer set programming (ASP) [9] is a well-known form of logic programming oriented towards hard search problems that relies on what is called answer set semantics to solve normal logic programs.

A normal logic program is one that is composed f rules (clauses) of the form: $r = A_0 \leftarrow A_1, \cdots, A_n, not\ A_{n+1}, \cdots, not\ A_{n+m} (n, m \geq 0)$, where each A_i $(i = 0, \cdots, n+m)$ is an atom from a first-order language with function symbols, and "not" is interpreted as negation by failure. We denote $head(r) = A_0$, $body^+(r) = \{A_1, \cdots, A_n\}$ and $body^-(r) = \{A_{n+1}, \cdots, A_{n+m}\}$.

Let P be a ground normal logic program and X be a set of grounded atoms. The reduct P^X is the set of clauses obtained from P as $P^X = \{head(r) \leftarrow body^+(r) | r \in P, body^-(r) \cap X = \emptyset\}$.

In other words, it deletes any clause in P that has a condition '$not\ A_i$' in its body where $A_i \in X$ and deletes every condition of the form '$not\ A_i$' in the bodies of the remaining clauses.

If we denote the minimal Herbrand model (i.e. a minimal set of grounded atoms coming from the program that satisfies all clauses) of a program as $M(P)$, we define the stable models $SM(P)$ of a normal program P as the sets of grounded atoms X such that $M(P^X) = X$.

A program P is locally stratified if there exists a partition $B_1, \dots, B_m$ of its Herbrand base such that for each grounded rule $r \in P$: if $head(r) = H$ and A in $body^+(r)$, then A in B_i, H in B_j and $i \leq j$, and if $head(r) = H$ and A in $body^-(r)$, then A in B_i, H in B_j and $i < j$. It has been shown that if a program is locally stratified, then a stable model exists and is unique.

An ASP solver finds the stable models of a normal logic program, typically with numerous extensions such as disjunctions (called choice) and cardinal constraints, amongst others. It has been shown all STRIPS problems can be translated into ASP [10], see: `https://github.com/potassco/plasp`.

In this case fluents are modeled by the “fluent/1” predicate, action by “action/1”, and operators by “pre/2”, “add/2” and “del/2”. The initial situation is captured by “init/1” and the different sub-goals by “query/1”. Then, planning in this case is implemented using inference in a predefined number of steps that make the sub-goals hold at a future state (successive states being modeled by timepoints). This definition is inspired by the event calculus.

As explained earlier, an action is permissible for a utilitarian base if and only if it’s consequences are as least as preferred as those of any other action that is possible to take in the given situation. In order to implement this idea, we added the following predicates:

- actionOverallUtility(Action, Utility): specifies the Utility of an Action.
- actionPositiveUtility(Action, Utility): same but only positive utilities.
- actionNegativeUtility(Action, Utility): same but only negative utilities.
- permitted(Action, State, EthicBase): denotes that Action is permitted at State according to EthicBase.
- forbidden(Action, State, EthicBase): denotes that Action is forbidden at State according to EthicBase.

The full implementation of consequentialist ethics for the trolley dilemma problem can be found at:
`https://github.com/martinjedwabny/asp-consequentialist-ethics`

Once the ASP approach has been formalised we can use explanation methods designed for explaining the derivation of a literal (as overviewed in [11]), or for explaining derivation for the consistency of assuming the literal or why a literal is not included in any answer set (as proposed in [12]).

5. Conclusion

Ethical decision making and planning (EDMP) has recently received much attention from the Artificial Intelligence community [7,13,14,15,16,17,18]. The trolley problem has been widely investigated (despite many complaints of it not being relevant for practical autonomous vehicle applications). Most of these approaches boil down to translating ethical constraints (as also seen in this paper) into utilities or preferences. The question arises of what is the real technical difficulty of EDMP problems with respect to preference based decision making and planning [19]. We hypothesise that the EDMP problems distinguish themselves by a critical need for explanation capabilities [20]. In this short paper we demonstrated how an consequentialist ethical planning problem can be expressed using ASP. While dedicated explanation techniques for ASP have been designed [11,12], this work can be taken forward by employing an argumentation based approach to explanation such as [21] (that decomposes each derivation of an atom into a part whether only assumptions to derive an atom are considered and define attack tree justification to give an overall explanation). An interactive dialogue can be constructed over this process to further improve the user experience [22].

References

[1] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[2] Tom Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[3] Edwin PD Pednault. Adl: Exploring the middle ground between strips and the situation calculus. *Kr*, 89:324–332, 1989.

[4] Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[5] Michael Gelfond and Vladimir Lifschitz. Action languages. 1998.

[6] Colin Allen, Wendell Wallach, and Iva Smit. Why machine ethics? *IEEE Intelligent Systems*, 21(4):12–17, 2006.

[7] Michael Anderson and Susan Leigh Anderson. *Machine ethics*. Cambridge University Press, 2011.

[8] Shelly Kagan. *Normative ethics*. Routledge, 2018.

[9] Vladimir Lifschitz. *Answer set programming*. Springer International Publishing, 2019.

[10] Yannis Dimopoulos, Martin Gebser, Patrick Lühne, Javier Romero, and Torsten Schaub. plasp 3: Towards effective asp planning. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 286–300. Springer, 2017.

[11] Jorge Fandinno and Claudia Schulz. Answering the" why" in answer set programming-a survey of explanation approaches. *arXiv preprint arXiv:1809.08034*, 2018.

[12] Jérémie Dauphin and Ken Satoh. Explainable asp. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 610–617. Springer, 2019.

[13] Sofia Panagiotidi, Juan Carlos Nieves, and Javier Vázquez-Salceda. A Framework to Model Norm Dynamics in Answer Set Programming. In *MALLOW*, 2009.

[14] Naveen Sundar Govindarajulu and Selmer Bringsjord. On automating the doctrine of double effect. *arXiv preprint arXiv:1703.08922*, 2017.

[15] Fiona Berreby, Gauvain Bourgne, and Jean-Gabriel Ganascia. A declarative modular framework for representing and applying ethical principles. 2017.

[16] Vincent Bonnemains, Claire Saurel, and Catherine Tessier. How Ethical Frameworks Answer to Ethical Dilemmas: Towards a Formal Model. In *EDIA@ ECAI*, pages 44–51, 2016.

[17] Felix Lindner, Robert Mattmüller, and Bernhard Nebel. Moral permissibility of action plans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7635–7642, 2019.

[18] Miles Brundage. Limitations and risks of machine ethics. *Journal of Experimental & Theoretical Artificial Intelligence*, 26(3):355–372, 2014. Publisher: Taylor & Francis.

[19] Meghyn Bienvenu, Christian Fritz, and Sheila A McIlraith. Specifying and computing preferred plans. *Artificial Intelligence*, 175(7-8):1308–1345, 2011.

[20] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[21] Claudia Schulz and Francesca Toni. Justifying answer sets using argumentation. *Theory and Practice of Logic Programming*, 16(1):59–110, 2016.

[22] Abdallah Arioua and Madalina Croitoru. Formalizing explanatory dialogues. In *International Conference on Scalable Uncertainty Management*, pages 282–297. Springer, 2015.