Planning to Avoid Side Effects

Toryn Q. Klassen[1,2,3], **Sheila A. McIlraith**[1,2,3], **Christian Muise**[4], **Jarvis Xu**[4]

[1]Department of Computer Science, University of Toronto, Toronto, Canada
[2]Vector Institute for Artificial Intelligence, Toronto, Canada
[3]Schwartz Reisman Institute for Technology and Society, Toronto, Canada
[4]School of Computing, Queen's University, Kingston, Canada
toryn@cs.toronto.edu, sheila@cs.toronto.edu, christian.muise@queensu.ca, 15gx3@queensu.ca

Abstract

In sequential decision making, objective specifications are often underspecified or incomplete, neglecting to take into account potential (negative) side effects. Executing plans without consideration of their side effects can lead to catastrophic outcomes – a concern recently raised in relation to the safety of AI. In this paper we investigate how to avoid side effects in a symbolic planning setting. We study the notion of minimizing side effects in the context of a planning environment where multiple independent agents co-exist. We define (classes of) negative side effects in terms of their effect on the agency of those other agents. Finally, we show how plans which minimize side effects of different types can be computed via compilations to cost-optimizing symbolic planning, and investigate experimentally.

1 Introduction

Sequential decision making relies on the specification of an objective, such as a final-state goal condition in the case of symbolic planning, or a reward function in the case of reinforcement learning. Such objectives may be underspecified or incomplete, allowing an AI to cause additional (often discretionary) changes to the environment, which we refer to as *side effects*. In some cases these side effects are of little consequence, while in other cases they may change the environment in ways that are undesirable. Amodei et al. (2016) give the example of a robot breaking a vase that it wasn't explicitly told not to break, noting that

> [F]or an agent operating in a large, multifaceted environment, an objective function that focuses on only one aspect of the environment may implicitly express indifference over other aspects of the environment.

More reflective of the potential for catastrophic outcomes, Stuart Russell gave the example of a robot, tasked to get coffee, killing everyone on its path to getting it (Lebans 2020).

A number of approaches to (learn to) avoid negative side effects have recently been developed for Markov Decision Processes (MDPs) and related formalisms (Zhang, Durfee, and Singh 2018; Krakovna et al. 2019; Turner, Hadfield-Menell, and Tadepalli 2020; Krakovna et al. 2020; Saisubramanian, Kamar, and Zilberstein 2020). In this paper we explore how to avoid side effects in symbolic planning.

Objective underspecification seems to be an overlooked topic in the classical planning literature. This may be because symbolic planning domains are typically designed by hand for specific tasks, often without the necessary symbols to even *represent* many side effects. However, in the future more general-purpose symbolic domains – perhaps learned from data – may come into use, bringing the issue of side effects to the fore. Such domain descriptions may yield more expansive state representations and associated action descriptions. Those action descriptions are likely to characterize action effects beyond those germane to the pursuit of some preconceived tasks, and thereby expose a litany of side effects. Furthermore, domain descriptions learned from data may themselves be inaccurate or incomplete, which can also be a cause of (negative) side effects (e.g., Saisubramanian, Zilberstein, and Kamar 2020).

Here, we restrict our attention to side effects that result from the underspecification of symbolic planning objectives. The side effects we are largely concerned with are those that impact the agency of other independent agents that co-exist in the same environment. Consider, for example, a home environment with several humans and robots. In this context, whether a side effect of a robot plan is construed as negative or positive is determined by those affected by the change. Consider a robot's plan that has the side effect that a screwdriver is left in the robot's possession. This side effect may be positive for the robot, but negative for all other agents who require the screwdriver to realize their goals and plans.

How do we generate classical plans that avoid such negative side effects? In the absence of definitive information, a conservative stance is for an agent generating a plan in pursuit of an underspecified objective to minimize *all* side effects – to leave the world as close as possible to the way it was prior to the execution of its plan. In this paper, we consider how one agent's actions may prevent other agents from achieving goals or executing plans that they would have otherwise been able to pursue and achieve, and propose means to generate plans that minimize such negative side effects.

The main contributions of this paper are the following.

1. We formalize the notion of side effect in classical planning.
2. We characterize classes of negative side effects that relate to the impact of an acting agent's plan on other agents' ability to subsequently realize their goals and plans.

3. We propose and implement mechanisms for computing side-effect-minimizing plans for STRIPS planning problems by compiling the task of achieving that objective into a planning problem with action costs.

2 Preliminaries

We begin by reviewing some definitions and notation.

Mathematical notation. Given a set A, $|A|$ denotes the cardinality of A, and A^* denotes the set of all finite sequences of elements from A. We use $|\pi|$ for the length of a sequence π. We use $\mathbb{I}(x)$ as the function which has value 1 if the condition described by x is true, and 0 otherwise.

We define a planning problem in terms of a state transition system, following Ghallab, Nau, and Traverso (2004, 2016).

Definition 1 (State-transition system)**.** A *state-transition system* is a tuple $\langle S, A, \delta\rangle$ where S is a finite set of states, A is a finite set of actions, and $\delta : S \times A \to S$ is a partial function.

Notation. We extend the definition of δ to take a sequence of actions as an argument. If $\delta(s_0, a_1) = s_1, \delta(s_1, a_2) = s_2, \ldots, \delta(s_{k-1}, a_k) = s_k$, then $\delta(s_0, a_1, \ldots, a_k) = s_k$.

Definition 2 (Planning problem and plan)**.** A *planning problem* is a tuple $\mathcal{P} = \langle \Sigma, s_0, S_G\rangle$ where $\Sigma = \langle S, A, \delta\rangle$ is a state-transition system, $s_0 \in S$ is the initial state, and $S_G \subseteq S$ is the set of goal states. A sequence of actions $\pi \in A^*$ is a *plan* for $\mathcal{P}$ iff $\delta(s_0, \pi) \in S_G$.

A more general sort of control structure than a plan is a partial policy:

Definition 3 (Partial policy)**.** Given a state-transition system $\langle S, A, \delta\rangle$, a (partial) policy is a (partial) function $\rho : S \to A$.

We will exploit this more flexible control structure in Section 4. Furthermore, a partial policy can be constructed from a sequential plan (Fritz and McIlraith 2007), which we will make use of in Section 5.

In the above definitions we make no commitment to the nature of the states in our planning problem. They could be comprised of pixels or propositions. In symbolic planning, a state is typically defined in terms of a set of propositions that establish the truth or falsity of properties of the state. In so-called *classical planning* a state s is represented compactly in terms of the set of propositions (*fluents*) that are true in the state and all fluents not in the set are regarded as false, like a database. The transition function is similarly represented compactly in terms of a set of database operations that add or delete propositions from the database when an action is performed. The truth or falsity of all other propositions persists (the frame assumption). This class of planning problems is referred to as STRIPS. Following Geffner and Bonet (2013, p. 24), we have the following definition:

Definition 4 (STRIPS planning problem)**.** A *STRIPS planning problem* is a tuple $\langle F, I, A, G\rangle$ where F is a finite set of propositional symbols, $I \subseteq F$ represents the initial state, A is the finite set of actions, and $G \subseteq F$ represents the goal. Furthermore, an action $a \in A$ is represented by three sets of atoms: the "Add list" $\mathsf{Add}(a)$, the "Delete list" $\mathsf{Del}(a)$, and the "Precondition list" $\mathsf{Pre}(a)$.

A STRIPS problem $\langle F, I, A, G\rangle$ represents a planning problem $\langle \Sigma, s_0, S_G\rangle$ where $\Sigma = \langle S, A, \delta\rangle$ is such that $S = 2^F$, $s_0 = I$, $S_G = \{s \in S \mid G \subseteq s\}$, and δ is as follows:

$$\delta(s, a) = \begin{cases} (s \setminus \mathsf{Del}(a)) \cup \mathsf{Add}(a) & \text{if } \mathsf{Pre}(a) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases}$$

Given a set of fluents F, we will write $\mathcal{L}(F)$ to denote the set of Boolean formulas over those fluents, i.e., formulas constructed using negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), and so on as usual. A *literal* is either an atomic formula $f \in F$ or its negation $\neg f$. We define the set of literals $\mathsf{lits}(F) = F \cup \{\neg f \mid f \in F\}$. Given a literal ℓ, we may write $\overline{\ell}$ for the complementary literal, i.e., $\overline{f} = \neg f$ and $\overline{\neg f} = f$.

Given a STRIPS state $s \subseteq F$ and formula $\varphi \in \mathcal{L}(F)$, we will write $s \models \varphi$ if φ is true under the truth assignment which maps all the fluents in s to true and all the fluents in $F \setminus s$ to false. Sometimes we will treat a set $\varphi \subseteq F$ of atoms as their conjunction.

While the problems we examine are cast as classical planning problems, our computation of side-effect minimizing plans in Section 5 will cause us to go beyond classical planning, incorporating preconditions that involve negative literals, effects that are conditional (Pednault 1989; Geffner and Bonet 2013), and associating actions costs with plans in order to compute cost-minimizing plans.

3 Minimizing Side Effects

As discussed in Section 1, planning problems with underspecified objectives may lead to solutions that cause additional changes – side effects.

Definition 5 (Side effect (informal definition))**.** A side effect of a plan is any change to the state, resulting from the execution of the plan, that was not prescribed explicitly as part of the goal.

A simple way to minimize side effects could be for the acting agent to minimize change – to construct a plan that would leave the world as close as possible to the way it was before it acted. We define a plan that minimizes change by appealing to a distance function that measures the distance (the change) between two states – here the initial state s_0 and the terminating state of the plan π.

Definition 6 (Change-minimizing plan)**.** Given a planning problem $\mathcal{P} = \langle \Sigma, s_0, S_G\rangle$ where $\Sigma = \langle S, A, \delta\rangle$, and a *distance function* $d : S \times S \to [0, \infty)$, a plan π for $\mathcal{P}$ is change minimizing iff there is no plan π' for $\mathcal{P}$ such that $d(\delta(s_0, \pi'), s_0) < d(\delta(s_0, \pi), s_0)$.

Depending on how the distance function is defined, it can be used to minimize *all* change or change with additional qualifications. (Despite it being called "distance", the function does not have to be symmetric.)

The change-minimizing objective in Definition 6 could be viewed as a classical planning version of the "very naive approach" to side effects discussed by Amodei et al. (2016):

> A very naive approach would be to penalize state distance $d(s_i, s_0)$ between the present state s_i and some initial state s_0. Unfortunately, such an agent wouldn't just avoid changing the environment—it will resist

any other source of change, including the natural evolution of the environment and the actions of any other agents!

This "naivety" does not stand out so much in our context, since a standard assumption of classical planning "excludes the possibility of actions by other actors, or exogenous events that are not due to any actor" (Ghallab, Nau, and Traverso 2016, p. 20).

3.1 Minimizing Side Effects in STRIPS

When a planning problem is represented in STRIPS, having a vocabulary of fluents gives an obvious way to define side effects and measure distance between states. Below, we define a literal as being a *fluent side effect* of a plan if that plan changes the literal's truth value even though the goal didn't specify that the truth value should change.

Definition 7 (Fluent side effect (FSE))**.** Let $\langle F, I, A, G\rangle$ be a STRIPS planning problem, and π a plan that solves it. Then $f \in F$ is a *fluent side effect* of π if $f \in \delta(I, \pi)$ but $f \notin I$ and $f \notin G$. Furthermore, if $f \notin \delta(I, \pi)$ but $f \in I$, we will say the literal $\neg f$ is a fluent side effect of π.

One simple idea is to try to find a plan for a planning problem that minimizes the number of side effects.

Definition 8 (Fluent-preserving plan)**.** Given a STRIPS planning problem $\mathcal{P} = \langle F, I, A, G\rangle$ and associated plan π for $\mathcal{P}$, π is a *fluent-preserving* plan iff there is no plan π' for $\mathcal{P}$ such that π' has strictly fewer fluent side effects than π.

This can be viewed as an instantiation of Definition 6 with the distance function d defined so that $d(\delta(I, \pi), I) = |\{\ell \in \mathsf{lits}(F) \mid \ell \text{ is an FSE of } \pi\}|$. It makes no attempt to distinguish side effects that are in any sense negative.

The determination of whether a side effect is negative or positive is often domain specific. Changing some fluents (e.g., one representing whether a vase is broken) could be given greater weight than changing others. Such domain-specific details could be built into a domain-tailored distance function. In the next section, we propose and characterize classes of negative side effects that relate, in general terms, to the impact an acting agent has on other agents, and are domain independent in the sense that they rely on general properties of the planning problem specification.

4 Effects of Plans on Other Agents

We claim that important classes of negative side effects of a plan are those that prevent agents operating in a shared environment from realizing possible goals or plans. We argue that the acting agent should therefore minimize such negative side effects; i.e, they should conceive and prefer plans that minimize the effect that the plan's execution will have on other agents' (and possibly their own) ability to achieve their goals or plans in the future.

In particular, we consider environments shared by multiple agents – each with its own actions, goals, and plans that it may wish to be able to execute at some point. Figure 1 introduces the *Canadian wildlife* domain, in which a robot truck's actions may prevent other agents (the wildlife) from achieving their goals or from following particular paths. This

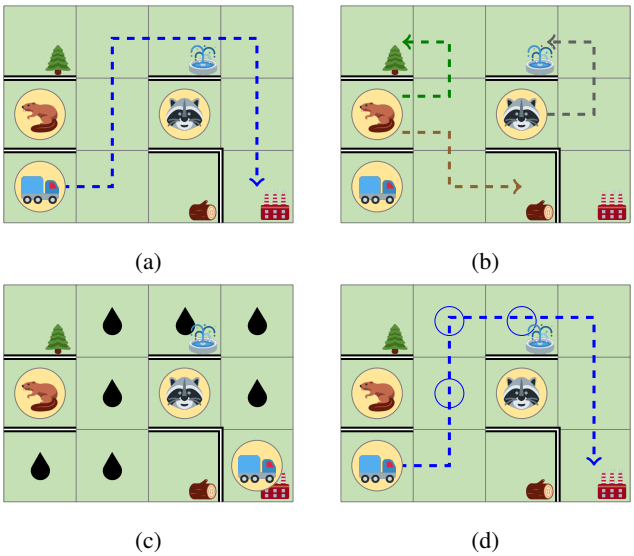

Figure 1: The Canadian wildlife domain. A robot truck (🚚), beaver (🦫), and raccoon (🦝) can move to adjacent cells, but not through walls (╗) or each other. The robot wants to get to the factory (🏭), but each cell it touches is contaminated with oil (🛢), after which it cannot be visited by animals. The beaver may want to reach the tree (🌲) or the wood (🪵), or the raccoon may want to wash its hands in the fountain (⛲); possible plans for these are shown in (b). If the robot just goes directly to the factory, following the path in (a), then the animals will be blocked from those goals by oil, as shown in (c). However, the robot also has the limited ability to clean up to three cells of oil. Following the plan in (d) which cleans (○) some cells after reaching them, the robot would allow the beaver to reach the tree, and the raccoon to reach the fountain – but not by the raccoon's plan in (b).

is an example of a *shared environment* with an *acting agent* (the robot).

Definition 9 (Shared environment and acting agent)**.** A *shared environment* is a planning problem $\mathcal{P} = \langle \Sigma, s_0, S_G\rangle$ with $\Sigma = \langle S, A, \delta\rangle$, as in Definition 2, but where the action set A is the union $A = \bigcup_{i=1}^{n} A_i$, such that each set A_i corresponds to the actions available to one of n distinct agents. We designate agent 1 as the *acting agent*.

The introduction of a shared environment with multiple agents might suggest that this problem be addressed using multi-agent planning, but our approach is deliberately single agent. Our focus is on the plan of the single acting agent, and how it can act to avoid causing negative side effects for agents who may act subsequently. The acting agent does not assume that other agents are cooperative, nor does it negotiate with them. Rather, it considers the achievement of its own objective, exercising discretion to minimize the impact of its plan on the subsequent agency of other agents, where objective underspecification allows for such discretion.

Functionally, this amounts to considering a simplified setting in which there are two phases: first, the acting agent executes a plan to accomplish its goal, and then afterwards

some agent (possibly the same agent again) has the opportunity to execute an additional action sequence. To illustrate, consider a set of roommates who share a kitchen. Each prepares their dinner alone, but with the understanding that any one of the roommates will subsequently use the kitchen to prepare a meal, and that the acting agent should minimize negative impact on whoever uses the kitchen next.

The actions of the first agent could affect others in a positive or negative way. A positive side effect might advance other agents' future goals, while a negative side effect would impede other agents' goal or plan realization. Positive side effects may be related to the notion of "helpfulness" (Freedman and Zilberstein 2017; Freedman et al. 2020), which considers how much an agent can reduce the cost of a team's plan. For this paper, we are concerned with negative effects.

The following definition will be useful.

Definition 10 (Achievable/Unachievable)**.** Given a shared environment $\mathcal{P}$, a goal $\hat{S}_G \subseteq S$ is achievable by agent i in state $s \in S$, written $\mathsf{achievable}(\hat{S}_G, i, s)$, if there exists a plan $\hat{\pi} \in A_i^*$ (i.e., using agent i's actions) such that $\delta(s, \hat{\pi}) \in \hat{S}_G$. If no such plan exists, then we say the goal is unachievable by agent i in s: $\mathsf{unachievable}(\hat{S}_G, i, s)$.

We can now define a class of negative side effects.

Definition 11 (Goal side effect (GSE))**.** Suppose $\mathcal{P} = \langle \Sigma, s_0, S_G \rangle$ is a shared environment. Suppose $\hat{S}_G \subseteq S$ is another goal and $\mathsf{achievable}(\hat{S}_G, i, s_0)$. Then a plan $\pi \in A_1^*$ for $\mathcal{P}$ has a *goal side effect (GSE)* on agent i w.r.t. goal $\hat{S}_G$ if $\mathsf{unachievable}(\hat{S}_G, i, \delta(s_0, \pi))$.

Definition 11 captures the case where the acting agent's plan precludes the subsequent achievement of another agent's goal. To illustrate, the robot truck's plan in Figure 1a will have GSEs on each animal w.r.t. any goal that requires that animal to move. There are many possible variants to Definition 11. If we were to add a quality measure (e.g., action costs or reward) then a negative side effect of a plan could be one that compromises the quality of another agent's subsequent achievement of their goal.

As with FSEs, we may wish to minimize the GSEs of a plan. To do so, we define a way of scoring how well a plan avoids GSEs. Given a set H of goal-agent pairs (each indicating a possible future goal that might be pursued by that agent), and associated weights, the score of a plan is just the sum of the weights of the goal-agent pairs from H such that that agent cannot reach that goal after the plan is executed:

Definition 12 (GSE score and goal-preserving)**.** Given a shared environment $\mathcal{P} = \langle \langle S, \bigcup_{i=1}^{n} A_i, \delta \rangle, s_0, S_G \rangle$, a plan $\pi \in A_1^*$ (i.e., consisting of the acting agent's actions) for $\mathcal{P}$, a finite set H of pairs $\langle \hat{S}_G, i \rangle$ where $\hat{S}_G$ is a goal and i an agent such that $\mathsf{achievable}(\hat{S}_G, i, s_0)$, and a weight function $w : H \to \mathbb{R}$, the *GSE score* of π is the sum

$$\sum_{\langle \hat{S}_G, i \rangle \in H} w(\hat{S}_G, i) \cdot \mathbb{I}(\mathsf{unachievable}(\hat{S}_G, i, \delta(s_0, \pi)))$$

We will say that π is *goal-preserving* if there is no other plan for $\mathcal{P}$ in A_1^* that has a lower GSE score.

If the weights are all 1, the GSE score is just how many goals are made unreachable for the corresponding agents. With weights of 1, the robot truck's plan in Figure 1a would have a GSE score of 3 (assuming H contains the animal goals described in Figure 1), while the plan in Figure 1d would have a superior GSE score of 1.

One reason to find a goal-preserving plan is to avoid causing a GSE under uncertainty about what future goal will be desired – i.e., uncertainty about what side effects are negative. Weights could be used to reflect that uncertainty: if we had a probability function $\Pr(\langle \hat{S}_G, i \rangle)$ giving the probability that the next agent to act would be i with goal $\hat{S}_G$, we could set the weight of $\langle \hat{S}_G, i \rangle$ to that probability. Then a goal-preserving plan would maximize the probability of not having a GSE on the next agent to act w.r.t. whatever its goal will be. (H could include all possible goals.)

An alternative reason to seek a goal-preserving plan might be to preserve the *freedom* of other agents to choose from a set of goals, even if some of those goals are not expected to be actually chosen. In that case the weights might be used to indicate how important different goals were.

Much as the acting agent may be uncertain about the goals of other agents, it might also be uncertain about what plans or policies (Definition 3) the other agents will follow. Avoiding interfering with other agents' plans or policies (even if their goals are still reachable) could be important to save other agents the effort of replanning. Various forms of interference could be of concern, but here we will consider just whether a goal is still *achievable* with a partial policy.

Definition 13 (Achievable (with a partial policy))**.** Given a planning problem $\mathcal{P}$, we'll say that a goal $\hat{S}_G \subseteq S$ is *achievable* with partial policy ρ from state s – written $\mathsf{achievable}(\hat{S}_G, \rho, s)$ – if there are states $s^0, \ldots, s^k$ (for some $k \geq 0$), such that $s^0 = s$, $s^{i+1} = \delta(s^i, \rho(s^i))$, and $s^k \in \hat{S}_G$. If there are not, then we may write $\mathsf{unachievable}(\hat{S}_G, \rho, s)$.

In other words, a goal is achievable with ρ if following the actions selected by ρ zero or more times will lead to the goal. We next define another class of negative side effects regarding when the acting agent's plan prevents another agent from achieving its goal using a particular (partial) policy.

Definition 14 (Policy side effect (PSE))**.** Suppose $\mathcal{P} = \langle \Sigma, s_0, S_G \rangle$ is a shared environment, $\rho : S \to A_i$ is a (partial) policy for some agent i, and $\hat{S}_G \subseteq S$ is a goal such that $\mathsf{achievable}(\hat{S}_G, \rho, s_0)$. Then a plan $\pi \in A_1^*$ for $\mathcal{P}$ has a *policy side effect (PSE)* on agent i w.r.t ρ and $\hat{S}_G$ if $\mathsf{unachievable}(\hat{S}_G, \rho, \delta(s_0, \pi))$.

Definition 15 (PSE score and policy-preserving)**.** Suppose we have a shared environment $\mathcal{P}$, plan $\pi \in A_1^*$, and weight function $w : H \to \mathbb{R}$ as in Definition 12, but the finite set H now contains pairs $\langle \hat{S}_G, \rho \rangle$ where $\hat{S}_G$ is a goal and $\rho : S \to A_i$ is a partial policy (for some i) such that $\mathsf{achievable}(\hat{S}_G, \rho, s_0)$. Then the *PSE score* of π is the sum

$$\sum_{\langle \hat{S}_G, \rho \rangle \in H} w(\hat{S}_G, \rho) \cdot \mathbb{I}(\mathsf{unachievable}(\hat{S}_G, \rho, \delta(s_0, \pi))).$$

We will say that π is *policy-preserving* if there is no other plan for $\mathcal{P}$ in A_1^* that has a lower PSE score.

To illustrate, consider again the robot truck's plan in Figure 1d, where we take the set H to be the partial policies determined by the animals' plans and goals from Figure 1b, and let the weights be 1. The robot's plan's PSE score is 2.

Observation 1. Given a shared environment $\langle \Sigma, s_0, S_G \rangle$, if a plan π has a GSE on agent i w.r.t. goal $\hat{S}_G$, then π also has a PSE on agent i w.r.t. ρ and $\hat{S}_G$, for any partial policy ρ such that $\mathsf{achievable}(\hat{S}_G, \rho, s_0)$. On the other hand, a plan may have PSEs without having any GSEs.

5 Computation

In this section we address how to compute side-effect minimizing plans. We represent the planning problems of our acting agent in STRIPS and, as such, use G instead of S_G to refer to the goal, and $\hat{G}$ instead of $\hat{S}_G$ to refer to a possible future goal whose achievability we wish to preserve. To find a *goal-preserving* plan for a STRIPS problem, for example, we will have a set of pairs $\langle \hat{G}, i \rangle$, where $\hat{G} \subseteq F$ and i is the agent that might wish to achieve $\hat{G}$.

We show how to find plans for the acting agent that are fluent-preserving, policy-preserving, or goal-preserving (Definitions 8, 15, and 12). We do so by compiling the original STRIPS problem into a planning problem with action costs. Plans for the compiled problem will involve extra bookkeeping actions, but we show the compiled problem to be equivalent in the sense defined shortly in Definition 17.

5.1 Preliminaries

We begin with two definitions – one old, one of our constructions – that are necessary to the exposition in this section.

Regression Regression is a rewriting operation, first introduced by Waldinger (1975), that given an action a and a state, s', resulting from performing a, returns a formula that characterizes the conditions that must have been true in the previous state to result in s'. (In this regard, the formula parsimoniously characterizes a family of states.) Following Muise (2014, Definition 2), we use the following definition:

Definition 16 (Regression in STRIPS)**.** Given a STRIPS problem $\langle F, I, A, G \rangle$, if $\varphi \subseteq F$ and $a \in A$, then the regression of φ through a, written $\mathcal{R}(\varphi, a)$, is defined as follows:

$$\mathcal{R}(\varphi, a) \stackrel{\text{def}}{=} \begin{cases} (\varphi \setminus \mathsf{Add}(a)) \cup \mathsf{Pre}(a) & \text{if } \mathsf{Del}(a) \cap \varphi = \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

Note that the result of regression is not interpreted as a state but as representing a set of states (the set of states that make it true). We use $\mathcal{R}^*(\varphi, \pi)$ to denote the iterated regression through all the actions in π in order.

The significance of regression for us comes from the following result, which is a specialization of Reiter's regression theorem (Reiter 2001).

Theorem 1 (Regression Theorem for STRIPS (Muise 2014, Theorem 2))**.** *An action sequence $\vec{a}$ is a plan for a STRIPS planning problem $\langle F, I, A, G \rangle$ if and only if $I \models \mathcal{R}^*(G, \vec{a})$.*

The following definition will be used to establish the correctness of our compilations.

Definition 17 (x-equivalent)**.** Let x be one of {fluent, policy, goal}. Suppose we have a STRIPS problem $\mathcal{P} = \langle F, I, \bigcup_{i=1}^{n} A_i, G \rangle$ and, if $x \in \{\text{policy}, \text{goal}\}$, an appropriate set H and weight function w for specifying x side effect scores. We will say that a planning problem $\mathcal{P}' = \langle F', I', A', G', c \rangle$ (with action costs c) is x-equivalent to $\mathcal{P}$ if there is a set $A'_1 = \{a' : a \in A_1\} \subseteq A'$ such that:

1. For any cost-optimal plan $\pi' \in A'^*$ for $\mathcal{P}'$, if the longest prefix of π' from A'^*_1 is $a'_1, \ldots, a'_k$, then $a_1, \ldots, a_k \in A^*_1$ is an x-preserving plan for $\mathcal{P}$.
2. If $a_1, \ldots, a_k \in A^*_1$ is an x-preserving plan for $\mathcal{P}$, then there is a cost-optimal plan for $\mathcal{P}'$ whose longest prefix of actions from A'^*_1 is $a'_1, \ldots, a'_k$.

5.2 Computing Fluent-Preserving Plans

To compute a fluent-preserving plan for a STRIPS problem $\mathcal{P} = \langle F, I, A_1, G \rangle$, we will introduce the *fluent-preserving compilation* $\mathcal{P}'$, which is fluent-equivalent to $\mathcal{P}$.

The full definition of the compilation is below (Definition 18), but first we give an overview. Essentially, we are taking as *soft goals* (see, e.g., Baier and McIlraith 2008) the fluents initially true, as well as the negations of the fluents initially false (that are not in the original goal), and applying the soft goal compilation from Keyder and Geffner (2009). For each fluent $f \in F \setminus G$ in $\mathcal{P}$, the fluent-preserving compilation $\mathcal{P}'$ introduces two new actions, $\checkmark_f$ and $\times_f$, and a new fluent $noted_f$. For $\checkmark_f$ to be executed requires that f's truth value not have changed from the initial state, while for $\times_f$ to be executed requires that f's truth value did change (i.e., there was an FSE). Either of $\checkmark_f$ or $\times_f$ makes $noted_f$ true, and the new goal G' requires that $noted_f$ be true for every f. For either $\checkmark_f$ or $\times_f$ to be executed requires the new end action has been executed, which can only occur after the original goal G has been achieved. The idea is that a plan for $\mathcal{P}'$ will consist of actions that make G true, followed by end, followed by $\checkmark_f$ or $\times_f$ actions to make $noted_f$ true for every f. The $\times_f$ actions have a cost, so a cost-optimal plan to $\mathcal{P}'$ will correspond to causing as few FSEs as possible.

Definition 18 (Fluent-preserving compilation)**.** Given a STRIPS planning problem $\mathcal{P} = \langle F, I, A_1, G \rangle$, its fluent-preserving compilation $\mathcal{P}'$ is a planning problem $\langle F', I, A', G', c \rangle$ (with action costs c) where

- $F' = F \cup \{ended\} \cup \{noted_f \mid f \in F \setminus G\}$
- $A' = A'_1 \cup \{end\} \cup A_\times \cup A_\checkmark$ where
  - $A'_1 = \{a' \mid a \in A_1\}$, where a' is like a except that $\mathsf{Pre}(a') = \mathsf{Pre}(a) \cup \{\neg ended\}$
  - $\mathsf{Pre}(end) = G$, $\mathsf{Add}(end) = \{ended\}$, $\mathsf{Del}(end) = \emptyset$
  - $A_\times = \{\times_f \mid f \in F \setminus G\}$, where
    $\mathsf{Pre}(\times_f) = \begin{cases} \{\neg f, ended\} & \text{if } f \in I \\ \{f, ended\} & \text{if } f \notin I \end{cases}$
    $\mathsf{Add}(\times_f) = \{noted_f\}$, $\mathsf{Del}(\times_f) = \emptyset$
  - $A_\checkmark = \{\checkmark_f \mid f \in F \setminus G\}$, where $\checkmark_f$ is like $\times_f$ except
    $\mathsf{Pre}(\checkmark_f) = \begin{cases} \{f, ended\} & \text{if } f \in I \\ \{\neg f, ended\} & \text{if } f \notin I \end{cases}$
- $G' = \{ended\} \cup \{noted_f : f \in F \setminus G\}$
- $c(a) = 1$ if $a \in A_\times$ and 0 otherwise

Theorem 2. *Given a STRIPS planning problem $\mathcal{P}$, the fluent-preserving compilation $\mathcal{P}'$ is fluent-equivalent to $\mathcal{P}$.*

Proof. See Appendix A.[1] □

[1] Appendices are in the technical report (Klassen et al. 2021).

Remark 1. The fluent-preserving compilation's search space may be quite large, since the actions from $A_{\checkmark}$ and $A_{\times}$ can be performed in any order. However, following Keyder and Geffner (2009), given an arbitrary ordering $f_1, \ldots, f_n$ on the fluents, it's easy to modify the compilation to allow $\checkmark_{f_k}$ or $\times_{f_k}$ to be executed only once $noted_{f_{k-1}}$ holds. Analogous optimizations can be applied to the policy- and goal-preserving compilations in the next two sections. We adopt these optimizations for evaluation.

5.3 Computing Policy-Preserving Plans

We next consider how to compute policy-preserving plans. We will suppose that the partial policies in H are represented using a plan (along with a goal). To make explicit how a plan can correspond to a policy, we will make use of the following definition that employs regression (from Definition 16).

Definition 19 ($\mathcal{R}^k(h)$)**.** Given a STRIPS problem $\mathcal{P} = \langle F, I, A, G\rangle$ and a pair $h = \langle \hat{G}, \hat{\pi}\rangle$ where $\hat{G} \subseteq F$ is a goal and $\hat{\pi} = a_1, \ldots, a_m \in A^*$, for $k \in \{1, \ldots, m\}$ we define $\mathcal{R}^k(h) = \mathcal{R}^*(\hat{G}, a_{m-k+1}, \ldots, a_m)$. Also, $\mathcal{R}^0(h) = \hat{G}$.

That is, for $h = \langle \hat{G}, \hat{\pi}\rangle$, we have defined $\mathcal{R}^k(h)$ to be the regression of $\hat{G}$ through the last k actions in $\hat{\pi}$. Following Fritz and McIlraith (2007), we can derive a (partial) policy from $\hat{\pi} = a_1, \ldots, a_m$, which outputs in a state the last action a_i such that executing $a_i, \ldots, a_m$ will reach the goal:

Definition 20 (Policy derived from a plan)**.** Given a STRIPS problem $\mathcal{P} = \langle F, I, A, G\rangle$ and a goal-plan pair $h = \langle \hat{G}, \hat{\pi}\rangle$ (such that $\delta(I, \hat{\pi}) \models \hat{G}$) where $\hat{\pi} = a_1, \ldots, a_m$, the *partial policy ρ derived from* $\hat{\pi}$ (and $\hat{G}$) is given by the following:

$$\rho(s) = \begin{cases} a_{m-k+1} & \text{if } s \models \mathcal{R}^k(h),\ k \neq 0, \text{ and} \\ & s \not\models \mathcal{R}^\ell(h) \text{ for } \ell < k \\ \text{undefined} & \text{otherwise} \end{cases}$$

It follows from Theorem 1 that this policy can achieve $\hat{G}$ from a state just in case $\mathcal{R}^k(h)$ is true for some k.

The policy-preserving compilation we define below is similar to the fluent-preserving one, except that rather than adding actions $\times_f$ and $\checkmark_f$ (and the fluent $noted_f$) for $f \in F \setminus G$ to keep track of whether f's truth value has changed, it does something similar for $h \in H$. For each goal-plan pair $h = \langle \hat{G}, \hat{\pi}\rangle \in H$, it adds the actions $\times_h$ and $\checkmark_h^k$ (for $k \in \{0, \ldots |\hat{\pi}|\}$) – and the fluent $noted_h$ – to keep track of whether the policy derived from $\hat{\pi}$ can still reach $\hat{G}$. $\checkmark_h^k$ can only be executed in a state where $\mathcal{R}^k(h)$ holds, i.e., a state in which the policy can reach its goal. $\times_h$ does not have that precondition, but has a cost. Under the condition that each given pair $\langle \hat{G}, \hat{\pi}\rangle$ is such that the plan $\hat{\pi}$ can actually reach the goal $\hat{G}$ from the initial state, a cost-optimal plan for the compilation will correspond to causing as few PSEs as possible, as will be formalized in Theorem 3. (If that condition is not guaranteed, a preprocessing step could be used to filter out the input pairs in which the plan doesn't reach the goal.)

Definition 21 (Policy-preserving compilation)**.** Given a STRIPS planning problem $\mathcal{P} = \langle F, I, \bigcup_{i=1}^n A_i, G\rangle$, a finite set H of pairs $\langle \hat{G}, \hat{\pi}\rangle$ where $\hat{G} \subseteq F$ is a goal and $\hat{\pi} \in A_i^*$ for some i, and a weight function $w : H \to \mathbb{R}$, the *policy-preserving compilation* $\mathcal{P}'$ is a planning problem $\langle F', I, A', G', c\rangle$ defined below.

- $F' = F \cup \{ended\} \cup \{noted_h \mid h \in H\}$
- $A' = A'_1 \cup \{end\} \cup A_\times \cup A_\checkmark$ where
  - $A'_1 = \{a' \mid a \in A_1\}$, where a' is like a except that $\mathsf{Pre}(a') = \mathsf{Pre}(a) \cup \{\neg ended\}$
  - $\mathsf{Pre}(end) = G$, $\mathsf{Add}(end) = \{ended\}$, $\mathsf{Del}(end) = \emptyset$
  - $A_\times = \{\times_h \mid h \in H\}$, where $\mathsf{Pre}(\times_h) = \{\neg noted_h, ended\}$, $\mathsf{Add}(\times_h) = \{noted_h\}$, and $\mathsf{Del}(\times_h) = \emptyset$
  - $A_\checkmark = \{\checkmark_h^k \mid h = \langle \hat{G}, \hat{\pi}\rangle \in H \text{ and } 0 \leq k \leq |\hat{\pi}|\}$, where $\checkmark_h^k$ is like $\times_h$ except $\mathsf{Pre}(\checkmark_h^k) = \mathcal{R}^k(h) \cup \mathsf{Pre}(\times_h)$, where regression is defined w.r.t. the original problem $\mathcal{P}$.
- $G' = \{ended\} \cup \{noted_h : h \in H\}$
- $c(a) = w(h)$ if $a = \times_h$ for an $h \in H$, and 0 otherwise

Theorem 3. *Given a STRIPS problem $\mathcal{P} = \langle F, I, \bigcup_{i=1}^n A_i, G\rangle$, a finite set H of pairs $\langle \hat{G}, \hat{\pi}\rangle$ where $\hat{G} \subseteq F$ is a goal and $\hat{\pi} \in A_i^*$ for some i (s.t. $\delta(I, \hat{\pi}) \models \hat{G}$), and a weight function $w : H \to \mathbb{R}$, the policy-preserving compilation $\mathcal{P}'$ is policy-equivalent to $\mathcal{P}$.*

Proof. See Appendix A. □

5.4 Computing Goal-Preserving Plans

Finally, we show how to compute goal-preserving plans, using another compilation. Instead of describing an action's effects with Add and Delete lists, we will sometimes use a set of *conditional effects*. A conditional effect is a pair $\langle C, E\rangle$, where C and E are sets of literals, denoting that if C holds in the current state, then E will hold after executing the action.

Definition 22 (Goal-preserving compilation)**.** Given a STRIPS planning problem $\mathcal{P} = \langle F, I, \bigcup_{i=1}^n A_i, G\rangle$; a finite set H of pairs $\langle \hat{G}, i\rangle$ where $\hat{G} \subseteq F$ is a goal and i is an agent; and a weight function $w : H \to \mathbb{R}$, the *goal-preserving compilation* $\mathcal{P}'$ is a planning problem $\langle F', I', A', G', c\rangle$ where

- $F' = F \cup \{ended\} \cup \{noted_h \mid h \in H\} \cup \{f_{cloned} : f \in F\} \cup \{resetNeeded\} \cup \{acting_i \mid 1 \leq i \leq n\}$
- $I' = I \cup \{acting_1\}$
- $A' = (\bigcup_{i=1}^n A'_i) \cup \{clone\} \cup \{reset_i \mid 1 \leq i \leq n\} \cup A_\times \cup A_\checkmark$ where
  - $A'_i = \{a' \mid a \in A_i\}$ where a' is like a except that $\mathsf{Pre}(a') = \mathsf{Pre}(a) \cup \{acting_i\}$
  - $\mathsf{Pre}(clone) = G \cup \{\neg ended\}$, and $clone$'s effects are given by this set of conditional effects: $\{\langle f, f_{cloned}\rangle : f \in F\} \cup \{\langle \neg f, \neg f_{cloned}\rangle : f \in F\} \cup \{\langle \top, \neg acting_1\rangle, \langle \top, ended\rangle, \langle \top, resetNeeded\rangle\}$
  - $\mathsf{Pre}(reset_i) = \{resetNeeded\}$, and $reset_i$'s effects are given by this set of conditional effects: $\{\langle f_{cloned}, f\rangle : f \in F\} \cup \{\langle \neg f_{cloned}, \neg f\rangle : f \in F\} \cup \{\langle \top, acting_i\rangle, \langle \top, \neg resetNeeded\rangle\}$
  - $A_\times = \{\times_h \mid h \in H\}$, where for each $h = \langle \hat{G}, i\rangle \in H$, we have $\mathsf{Pre}(\times_h) = \{acting_i, \neg noted_h, ended, \neg resetNeeded\}$, $\mathsf{Add}(\times_h) = \{noted_h, resetNeeded\}$, $\mathsf{Del}(\times_h) = \{acting_i\}$.
  - $A_\checkmark = \{\checkmark_h \mid h \in H\}$, where for $h = \langle \hat{G}, i\rangle \in H$, $\checkmark_h$ is the same as $\times_h$ except that $\mathsf{Pre}(\checkmark_h) = \hat{G} \cup \mathsf{Pre}(\times_h)$
- $G' = \{ended\} \cup \{noted_h : h \in H\}$

- $c(a) = w(h)$ if $a = \times_h$ for an $h \in H$, and 0 otherwise

This is the most complicated compilation. The idea is that in a plan for $\mathcal{P}'$, the original goal G will be reached and then the special $clone$ action is executed, which captures a copy of the truth values of all the original fluents. Then the other agents can try to achieve each of their goals (the $acting_i$ fluents are used to make sure only one agent acts at a time). Each time some agent's goal $\hat{G}$ is completed by that agent, the state can then be reset to what it was when G was reached (because we're interested in what goals are reachable from that point). For each goal-agent pair $h = \langle \hat{G}, i \rangle \in H$, there are two new actions, $\checkmark_h$ and $\times_h$. For $\checkmark_h$ to be executed requires that $\hat{G}$ be true while i is acting, while $\times_h$ does not require that (but has a cost). The result is that a cost-optimal plan for $\mathcal{P}'$ will be one in which the first actions (before $clone$) set things up to maximize the weighted sum of goal-agent pairs $\langle \hat{G}, i \rangle \in H$ such that $\hat{G}$ can be reached by i.

Note that the compilation does not check if the goals in H are all initially reachable by their agents. If they are not, a preprocessing step to filter out unreachable goals would be needed for the condition in the following theorem to apply.

Theorem 4. *Given a STRIPS problem $\mathcal{P} = \langle F, I, \bigcup_{i=1}^{n} A_i, G \rangle$, a finite set H of pairs $\langle \hat{G}, i \rangle$ where $\hat{G} \subseteq F$ is a goal and i an agent (such that* $\mathsf{achievable}(\hat{G}, i, s_0)$*), and a weight function $w : H \to \mathbb{R}$, the goal-preserving compilation $\mathcal{P}'$ is goal-equivalent to $\mathcal{P}$.*

Proof. See Appendix A. □

6 Experiments

We ran experiments to demonstrate, as a proof of concept, how our compilations can minimize different types of side effects, and also to assess the effect on run time of the different minimizations.[2]

We tested on a formalization of our Canadian wildlife domain (from Figure 1), and adaptations of the standard IPC planning domains *zenotravel*, *floortile*, and *storage* (see Appendix B). These domains were chosen because they allowed for agents to significantly interfere with each other. For each domain and problem, we found plans for the acting agent by standard planning (with no consideration of side effects), and by finding fluent-preserving, policy-preserving, and goal-preserving plans (via planning on the respective compilations[3]). The results are in Table 1.

The experiments were run on a Linux workstation with a Core i9-9900K CPU (3.60 GHz) and 32GB of RAM. We used the LM-Cut planner (Helmert and Domshlak 2011) for all problems except the goal-preserving plans whose compilations had conditional effects, necessitating LAMA (Richter and Westphal 2010), which we ran to completion instead. Both LM-Cut and LAMA are configurations of Fast Downward (Helmert 2006) (version 20.06 was used).

The compile and planning times reflect the extra effort required to avoid side effects, with all such strategies requiring at least an order of magnitude more time than standard planning. Planning on the goal-preserving compilations took the most time – not surprisingly since the planner has to, in effect, try to find a plan not just for the original problem's goal, but for *all* of the possible goals from H.

Compilation time was significant for all compilation types, especially on the zenotravel problems, where the number of (ground) actions is very large. The present implementation naively grounds planning problems before compiling them, which does not scale well with large numbers of objects. This is illustrated by the greatly increased times for the problems storage-c2 and storage-c3, which are identical to storage-c except for having extra irrelevant objects: for each (non-agent) object in storage-c, storage-c2 has two objects (the original object and a duplicate) and storage-c3 has three objects (the original object and two duplicates). Future work might be able to improve on that by performing the compilations directly on the lifted (PDDL) representation, or by using a smarter form of grounding.

With respect to the quality of the plans – the number of side effects – it's difficult to draw conclusions since so much depends on the specifics of the domains including the number of deadend-able goals, the number of irreversible actions, the diversity in the plans of future-acting agents, and their relationship to the goal of the acting agent – whether they're tightly coupled, or more independent. What we do see is that on the domains tested, the fluent-preserving plans sometimes showed some modest reduction in PSEs and GSEs, compared to standard planning, though this may be anecdotal. What was more notable was the reduction in GSEs and PSEs shown by the goal- and policy-preserving plans, relative to standard and fluent-preserving plans.

7 Related Work

As discussed in the introduction, the problem of avoiding side effects has been studied in MDPs in a number of papers. These works can be divided into two sorts, those which involve interacting with a human to get further information about what side effects are negative (e.g., Zhang, Durfee, and Singh 2018; Saisubramanian, Kamar, and Zilberstein 2020), and those not making use of human feedback. The latter are more related to our work.

In particular, our work was inspired at a high level by approaches to avoiding side effects without human feedback where the agent in an MDP is encouraged to preserve its own ability to reach states (Krakovna et al. 2019), gain reward from other reward functions (Turner, Hadfield-Menell, and Tadepalli 2020), or complete tasks from a given distribution (Krakovna et al. 2020). Beyond the difference of formalism, a conceptual difference of our work is that we allow for considering more than just the preservation of the agent's own abilities. To see why that's important, consider how a tall robot putting an object on a high shelf won't interfere with its own ability to use the object later, but may interfere with shorter agents. Note that Turner (2019) did informally discuss considering other agents' attainable utilities.

In the rest of this section, we review planning papers that have some relation to side effects on other agents.

[2] See https://github.com/tqk/side-effects-planner for the code for the experiments, which was written in Python and heavily relies on the Tarski library (https://github.com/aig-upf/tarski).

[3] We apply the optimization from Remark 1 to all compilations.

Domain & Problem	$\|H\|$	Standard planning				Fluent-preserving					Policy-preserving			Goal-preserving		
		FSE	PSE	GSE	PT	FSE	PSE	GSE	CT	PT	PSE	CT	PT	GSE	CT	PT
wildlife	$3, 3$	17	3	3	0.5	13	3	3	0.8	20.2	1	0.6	6.5	1	0.6	38.0
zeno-a	$5, 2$	7	4	0	0.5	5	4	0	17.6	10.6	3	17.6	9.5	0	17.3	23.3
zeno-b	$4, 2$	5	2	0	0.4	5	2	0	17.6	7.2	0	17.4	10.4	0	17.0	24.6
zeno-c	$7, 4$	5	3	0	0.4	3	3	0	18.2	12.3	3	17.9	7.9	0	17.2	26.3
floortile-a	$4, 2$	6	4	0	0.5	2	3	1	2.8	16.9	0	2.5	9.2	0	2.5	56.4
floortile-b	$4, 2$	5	4	0	0.4	1	3	0	2.8	11.6	0	2.4	7.3	0	2.5	54.6
floortile-c	$8, 4$	5	8	1	0.5	1	5	0	2.8	18.5	1	2.5	4.9	0	2.5	97.2
storage-a	$6, 2$	5	5	0	0.4	5	5	0	0.9	7.4	0	0.9	10.4	0	0.9	14.1
storage-b	$4, 2$	8	4	0	0.4	5	2	0	0.9	6.2	0	0.9	5.2	0	0.9	15.5
storage-c	$7, 4$	14	3	2	0.4	10	3	0	0.9	7.0	3	0.9	5.7	0	0.9	16.2
storage-c2	$7, 4$	14	3	2	0.4	10	3	0	10.2	44.0	3	10.0	48.8	0	10.1	21.0
storage-c3	$7, 4$	14	3	2	0.4	10	3	0	49.8	163.5	3	50.3	159.3	0	48.5	53.7

Table 1: The number of fluent side effects (FSE), policy side effects (PSE), goal side effects (GSE), compilation time (CT) and planning time (PT) (in seconds) caused by plans found by standard planning, and by fluent-, policy-, and goal-preserving plans. The $|H|$ column captures (i) the number of goal-policy pairs in the set H used in finding policy-preserving plans (and in calculating the PSE scores), and (ii) the number of goal-agent pairs in the other set H used in finding goal-preserving plans (and in calculating the GSE scores). The weights were all 1.

Freedman and Zilberstein (2017) defined "independent interaction" in which one agent should achieve its own goal while not stopping another agent from accomplishing its task. However, the formalization appears to sometimes require the first agent to actively help the second. Meanwhile, Karpas, Shleyfman, and Tennenholtz (2017) and following work (e.g., Nir, Shleyfman, and Karpas 2020) have considered how "social laws" – restrictions on what actions can be performed when – in multiagent STRIPS can prevent agents from blocking each other's plans.

In the context of *active goal recognition*, in which an observing agent performs actions to try to determine the goal G of another agent from a set $\mathcal{G}$ of hypothesized possible goals, Shvo and McIlraith (2020) defined an observer's plan τ to be *non-intervening* "if for every hypothesis $G \in \mathcal{G}$ the set of plans π, whose execution achieves G are preserved under the execution of τ." This could be thought of as the observer not having side effects on the observed agent.

Pozanco et al. (2019) introduced the problem of identifying a state that would, given a set of possible goals, minimize the average distance to the goals, or the maximum distance to any goal, which could be thought of as having positive effects on future goals. Another concept, related to having negative effects on others, is *Stackelberg planning* (Speicher et al. 2018), "where a *leader* player in a classical planning task chooses a minimum-cost action sequence aimed at maximizing the plan cost of a *follower* player in the same task."

8 Concluding Remarks

As AI systems proliferate within society, there is a growing concern related to unintended and potentially harmful behaviour. The much-cited 2016 paper, "Concrete Problems in AI Safety," presents five practical research problems in this vein, including the problem of "avoiding side effects" that emerge from having an underspecified or incomplete objective function (Amodei et al. 2016). While discussion of AI Safety has largely focused on data-driven machine learning systems, in this paper we argue that it is also an important problem for symbolic planning systems, particularly those whose domain theories may eventually be learned.

This paper introduces and frames this important problem, in a principled way, to the symbolic planning community. Here, we have considered the problem of avoiding (negative) side effects in classical planning. We observed that whether a side effect was negative or positive was often determined by those affected by the change. As such we argued for examining negative side effects in terms of their impact on the agency of other agents. We presented several versions of side-effect minimization, and showed how, in the case of STRIPS planning problems, to compute them through compilations to cost-optimizing planning problems.

Our work is not without its limitations. Goal- and policy-preserving plans minimize their impact on what goals or policies can be pursued immediately following the execution of the acting agent's plan. This treatment of side effects is somewhat myopic in that it doesn't look ahead to consider all possible sequences of goals that may arise in the future. The approach presented here also does not consider side effects in a true multi-agent environment where other agents are operating simultaneously alongside the acting agent. A final limitation of the presented approach is that it relies on having representations rich enough to make relevant distinctions, e.g., to tell whether a vase is broken.

There are many avenues for future work. Further side-effect-minimizing definitions could be made, taking into account aspects like action costs. There is also both need and opportunity for further work on efficient ways to compute plans that minimize side effects. Finally, the idea of considering the effect of actions on other agents' abilities is one that would be useful to bring from planning to reinforcement learning, and we have started to explore that (Alizadeh Alamdari et al. 2021, 2022).

Ethical Statement

Our work is concerned with making planning systems safer. However, depending on how our techniques are used, this is not guaranteed. For example, a robot acting to preserve the ability of other agents to achieve goals from some set could potentially further disrupt the environment, preventing the achievement of other goals that were not taken into consideration. Furthermore, as with many AI systems, techniques that are used to purposefully *avoid* harm can also be used to *wield* harm towards others, in this case, by supplying the acting agent with a plan that achieves its goals while purposefully *restricting* the future agency of others. This is not unlike the Stackelberg planning problem discussed at the end of Section 7.

Acknowledgements

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and Microsoft Research. Finally, we thank the Schwartz Reisman Institute for Technology and Society for providing a rich multi-disciplinary research environment.

A preliminary version of this paper appeared at an IJCAI 2021 workshop, "Robust and Reliable Autonomy in the Wild" (Klassen and McIlraith 2021).

The emojis in this paper are from the Twitter Emoji library (https://github.com/twitter/twemoji), which is copyrighted by Twitter, Inc and other contributors, and licensed under CC-BY 4.0 (https://creativecommons.org/licenses/by/4.0/). The truck and droplet emojis were modified.

References

Alizadeh Alamdari, P.; Klassen, T. Q.; Toro Icarte, R.; and McIlraith, S. A. 2021. Avoiding Negative Side Effects by Considering Others. In *NeurIPS 2021 Workshop on Safe and Robust Control of Uncertain Systems*.

Alizadeh Alamdari, P.; Klassen, T. Q.; Toro Icarte, R.; and McIlraith, S. A. 2022. Be Considerate: Avoiding Negative Side Effects in Reinforcement Learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022)*. To appear.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P. F.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*.

Baier, J. A.; and McIlraith, S. A. 2008. Planning with Preferences. *AI Magazine*, 29(4): 25–36.

Freedman, R. G.; Levine, S. J.; Williams, B. C.; and Zilberstein, S. 2020. Helpfulness as a Key Metric of Human-Robot Collaboration. *arXiv preprint arXiv:2010.04914*. Presented at *Artificial Intelligence for Human-Robot Interaction*, part of the AAAI 2020 Fall Symposium Series.

Freedman, R. G.; and Zilberstein, S. 2017. Integration of Planning with Recognition for Responsive Interaction Using Classical Planners. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 4581–4588.

Fritz, C.; and McIlraith, S. A. 2007. Monitoring Plan Optimality During Execution. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007*, 144–151.

Geffner, H.; and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning Theory and Practice*. Elsevier.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2011. LM-Cut: Optimal Planning with the Landmark-Cut Heuristic. In *Seventh International Planning Competition (IPC 2011), Deterministic Part*, 103–105.

Karpas, E.; Shleyfman, A.; and Tennenholtz, M. 2017. Automated Verification of Social Law Robustness in STRIPS. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017*, 163–171.

Keyder, E.; and Geffner, H. 2009. Soft Goals Can Be Compiled Away. *Journal of Artificial Intelligence Research*, 36: 547–556.

Klassen, T. Q.; and McIlraith, S. A. 2021. Planning to Avoid Side Effects (Preliminary Report). In *IJCAI Workshop on Robust and Reliable Autonomy in the Wild (R2AW)*.

Klassen, T. Q.; McIlraith, S. A.; Muise, C.; and Xu, J. 2021. Planning to Avoid Side Effects (Technical Appendix). Technical Report CSRG-641, Department of Computer Science, University of Toronto. https://ftp.cs.toronto.edu/csrg-technical-reports/641/.

Krakovna, V.; Orseau, L.; Martic, M.; and Legg, S. 2019. Penalizing Side Effects using Stepwise Relative Reachability. In *Proceedings of the Workshop on Artificial Intelligence Safety 2019 co-located with the 28th International Joint Conference on Artificial Intelligence, AISafety@IJCAI 2019*, volume 2419 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Krakovna, V.; Orseau, L.; Ngo, R.; Martic, M.; and Legg, S. 2020. Avoiding Side Effects By Considering Future Tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.

Lebans, J. 2020. The threat from AI is not that it will revolt, it's that it'll do exactly as it's told. CBC Radio. URL https://www.cbc.ca/radio/quirks/apr-25-deepwater-horizon-10-years-later-covid-19-and-understanding-immunity-and-more-1.5541299/the-threat-from-ai-is-not-that-it-will-revolt-it-s-that-it-ll-do-exactly-as-it-s-told-1.5541304.

Muise, C. 2014. *Exploiting Relevance to Improve Robustness and Flexibility in Plan Generation and Execution*. Ph.D. thesis, University of Toronto.

Nir, R.; Shleyfman, A.; and Karpas, E. 2020. Automated Synthesis of Social Laws in STRIPS. In *Proceedings of the*

Thirty-Fourth AAAI Conference on Artificial Intelligence, 9941–9948.

Pednault, E. P. D. 1989. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, 324–332.

Pozanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2019. Finding Centroids and Minimum Covering States in Planning. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2019*, 348–352.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Saisubramanian, S.; Kamar, E.; and Zilberstein, S. 2020. A Multi-Objective Approach to Mitigate Negative Side Effects. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 354–361.

Saisubramanian, S.; Zilberstein, S.; and Kamar, E. 2020. Avoiding Negative Side Effects due to Incomplete Knowledge of AI Systems. *arXiv preprint arXiv:2008.12146*.

Shvo, M.; and McIlraith, S. A. 2020. Active Goal Recognition. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 9957–9966.

Speicher, P.; Steinmetz, M.; Backes, M.; Hoffmann, J.; and Künnemann, R. 2018. Stackelberg Planning: Towards Effective Leader-Follower State Space Search. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, 6286–6293.

Turner, A. 2019. Reframing Impact. Blog post, https://www.lesswrong.com/s/7CdoznhJaLEKHwvJW.

Turner, A. M.; Hadfield-Menell, D.; and Tadepalli, P. 2020. Conservative Agency via Attainable Utility Preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, 385–391.

Waldinger, R. 1975. Achieving several goals simultaneously. Technical Note 107, SRI Project 2245.

Zhang, S.; Durfee, E. H.; and Singh, S. P. 2018. Minimax-Regret Querying on Side Effects for Safe Optimality in Factored Markov Decision Processes. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, 4867–4873.