Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

Spectra: An Expressive STRIPS-Inspired AI Planner Based on Automated Reasoning

(System Description)

Brandon Rozek · Selmer Bringsjord

December 2023

Abstract Research in automated planning traditionally focuses on model-based approaches that often sacrifice expressivity for computational efficiency. For artificial agents that operate in complex environments, however, frequently the agent needs to reason about the beliefs of other agents and be capable of handling uncertainty. We present Spectra, a STRIPS-inspired AI planner built atop automated reasoning. Our system is expressive, in that we allow for state spaces to be defined as arbitrary formulae. Spectra is also designed to be logic-agnostic, as long as an automated reasoner exists that can perform entailment and question-answering over it. Spectra can handle environments of unbounded uncertainty; and with certain non-classical logics, our system can create plans under epistemic beliefs. We highlight all of these features using the cognitive calculus $\mathcal{DCC}$. Lastly, we discuss that under this framework, in order to fully plan under uncertainty, a defeasible (= non-monotonic) logic can be used in conjunction with our planner.

1 Introduction

Agents who interact with the physical world often encounter uncertainty in the environment. One strategy to deal with uncertainty is to capture it in declarative form. For example, consider a situation where in the coming days the weather will either be rain (R), snow (S), or sunny (S^*). One goal that a rational agent might have is to ensure it stays dry while outside (D). We can represent the initial state space (Γ_0), with its uncertainty w.r.t. weather conditions, as the formula $R \vee S \vee S^*$. The agent can then attempt to find a plan or sequence of actions that takes an agent from Γ_0 to a state space that satisfies its goal D.

Model-based automated planning has primarily focused on environments wherein complete state information is provided to the agent as a collection of facts. This is standardly enabled through the usage of the *closed world assumption* (CWA), which states that any fact not specified in the state description is assumed to be false. Model-based conformant planning extends this by replacing an individual state with a finite collection of states otherwise known as a *belief state*. Each state within the belief state is individually closed under CWA. We take this a step further in our work: we allow and handle uncertain situations that can be captured by a collection of arbitrary (potentially higher-order and modal) formulae. This allows us to not only plan over a finite set of states, but potentially over an infinite set in the quantified case. We allow for the usage of arbitrary theories, such as Peano Arithmetic, to consider problems that require complex actions to reach the goal. We achieve this by making use of automated reasoning *within the planning process itself*.

Automated planning through reasoning was first investigated by Green [20] in 1981. Work continued throughout the early 2000s, where the planning problem was represented using the event calculus [11, 30, 31, 43] and/or the situation calculus [13, 39, 42]. Model-based approaches, however, have gained more traction over the years due to efficient implementations such as Fast Downward [23] with strong heuristics like delete relaxation [4], lm-cut [24], cost partitioning [26], and potential heuristics [38]. Mikhail and Ryan recently developed a planner that makes use of both heuristic search and reasoning over the situation calculus [45]. We believe this direction deserves additional attention; however, we are primarily concerned with problems that re-

Rensselaer Polytechnic Institute (RPI) E-mail: {Rozek.Brandon, Selmer.Bringsjord}@gmail.com

quire non-classical, specifically intensional,[1] reasoning. Examples of such problems arise in modeling misconceptions of beliefs between multiple agents; such modeling must for instance require sufficient expressivity to formalize the propositions that Agent 1 believes that Agent 2 believes that Agent 1 believes there is a stack of blocks that are to be unstacked, even when no such belief on the part of Agent 2 is veridical.[2] In addition, we're interested in looking beyond deductive logics and toward inductive ones to allow for careful treatment of information gathered and its likelihood [8, 9]. For example, one might wish to have an artificial agent assign a belief derived from perception a higher likelihood than a belief derived from agent communication. These are examples of issues that the relevant non-classical-logic community aims to tackle.

We introduce Spectra[3], a planner founded on automated reasoning. We designed Spectra to be logic agnostic, allowing one to plug in any logic that has an associated automated reasoner capable of determining entailment and question-answering. This allows the user to model the planning problem in a logic they're accustomed to. We showcase Spectra with the non-classical, specifically intensional cognitive calculi $\mathcal{DCC}$, a fragment of $\mathcal{DCEC}$ (which has been used in numerous prior papers and simulations; e.g. [18]). We provide an example of how Spectra with $\mathcal{DCC}$ can solve epistemic planning problems. Finally, we discuss how defeasible (= non-monotonic) logics can be used in our framework to create plans over a large range of uncertain scenarios.

2 Background and Related Work

Model-based automated classical planning can be captured by a propositional STRIPS model [12]. This is a tuple $\langle P, O, I, G\rangle$ where P is a finite set of ground atomic formulae, O is the finite set of operators, $I \subseteq P$ is the initial state, and G are the goals. Operators consist of preconditions $\texttt{Pre} \subseteq P$, add effects $\texttt{Add} \subseteq P$, and delete effects $\texttt{Del} \subseteq P$. Given a state $s \subseteq P$, an action is applicable iff $\texttt{Pre} \subseteq s$. After performing an action o on state s, the next state will be $(s - \texttt{Del}) \cup \texttt{Add}$.

Alternatives to the STRIPS model include but are not limited to Functional STRIPS [15], ADL [35], and SAS$^+$ [2]. However, often the more compact representation of PDDL [21] is used. This allows declarative information to be captured through predicate logic as opposed to only propositional logic, along with other features such as conditional effects. However due to the domain-closure assumption, this representation can be translated to an equivalent STRIPS problem with an exponential blowup of grounded operators compared to the lifted ones described with predicate logic. When using PDDL, a finite list of object labels Obj are provided. Formulae may include $\forall$ and $\exists$ statements; however, these get grounded to their propositional form. The domain-closure assumption takes all quantifiers and replaces them using the *truth-functional expansion* over the list of object labels Obj. That is, $\forall x, P(x)$ is equivalent to

$$\bigwedge_{o \in Obj} P(o),$$

and $\exists x, P(x)$ is equivalent to

$$\bigvee_{o \in Obj} P(o).$$

Conformant planning extends the STRIPS model by changing the initial state I to a finite set of possible initial states, often called a *belief state* in the literature [3]. Actions are then applicable at a given belief state b if it is applicable for all states $s \in b$. Similarly, a belief state b satisfies a goal G iff for all $s \in b$, $G \subseteq s$. In addition to CWA and domain-closure assumptions, classical model-based planning also carries the unique-name assumption and grounds all actions prior to search. This assumption states that for any two distinct object labels $o_1, o_2 \in Obj$, they do not refer to the same object; i.e., $obj(o_1) \neq obj(o_2)$. Before the search algorithm commences, a traditional classical model-based planner will ground all actions to their propositional form. Over the years, domains have been identified where the size of their grounded representations are too large to contain in device memory. [22, 28]. An alternate line of research that addresses this is *lifted planning*.

We use a fragment of the Deontic Cognitive Event Calculus ($\mathcal{DCEC}$) (see e.g. [7]) in order to showcase our planner solving epistemic tasks. $\mathcal{DCEC}$ is a quantified, multi-modal,[4] sorted cognitive-event calculus. Our frag-

[1] As a reminder, extensional logics are marked by the fact that semantic values of "inner" parts of formulae compositionally determine such values for these formulae. Hence if it's false that a is a block, $\neg Block(a)$, then standard compositionality in first-order logic dictates that $Block(a) \rightarrow Block(b)$ is true. In contrast, despite the fact that $\neg Block(a)$, it could be that Jones believes the opposite, i.e. $\mathbf{B}(j, Block(a))$. Efficient coverage of the fundamental distinction between extensional versus intensional logics is given in [14].

[2] A traditional "engine" of demands for such intensional expressivity from computational logics has been increasingly demanding versions of the *false-belief task*. See e.g. [6]; and for apparently the first formalization and simulation of the task see [1] .

[3] Code is available at `https://github.com/rairlab/spectra`

[4] For every substantive cognitive verb in the human case (e.g., *intends*, *desires*, *says/communicates*, *perceives*, the epistemic verbs (the class most relevant to the present paper), *attends to*, etc.), the approach of which this particular calculus is an example ultimately calls for a corresponding modal operator to be present.

$\mathcal{DCC}$ Signature

$$\phi ::= \begin{cases} \psi : Predicate \mid \forall x : \phi \mid \exists x : \phi \\ \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi \\ \mathbf{B}(a,\phi) \mid \mathbf{K}(a,\phi) \mid \mathbf{C}(\phi) \mid \mathbf{S}(a_1,a_2,\phi) \mid \mathbf{P}(a,\phi) \end{cases}$$

$\mathcal{DCC}$ Inference Schemata (Subset)

$$\frac{\mathbf{B}(a,\Gamma) \quad \Gamma \vdash \phi}{\mathbf{B}(a,\phi)}\ I_B \qquad \frac{\mathbf{S}(s,h,\phi)}{\mathbf{B}(h,\mathbf{B}(s,\phi))}\ I_{12}$$

$$\frac{}{\mathbf{C}(\mathbf{P}(a,\phi) \rightarrow \mathbf{K}(a,\phi))}\ I_1 \qquad \frac{}{\mathbf{C}(\mathbf{K}(a,\phi) \rightarrow \mathbf{B}(a,\phi))}\ I_2$$

ment $\mathcal{DCC}$ disregards time and treats all timepoints as equivalent. The box labeled $\mathcal{DCC}$ Signature shows the signature of our fragment. Cognitive operators here are only: **B**elieves, **C**ommon-knowledge, **S**ays, **P**erceives, and **K**nows. A sample reading for the formula

$$\mathbf{B}(a, \mathbf{S}(a, b, \mathbf{P}(c, \phi)))$$

is "agent a believes that a says to agent b that agent c perceives ϕ." The formula $\forall x : \phi$ reads that ϕ holds for all bound variables x ranging over the domain of discourse. A subset of our inference schemata are shown in the box labeled '$\mathcal{DCC}$ Inference Schemata (Subset).'

Regarding related work, Soutchanski & Young [45] designed an automated planner that also performs a heuristic search guided by automated reasoning. Their work is specific to the situation calculus and instead of states, they transition over *situations* which are formulae in the situation calculus. The Planning Techniques and Action Languages (PLATAS) project integrates planning and the GOLOG action language by embedding the planning description language PDDL into GOLOG, which uses an extended version of the situation calculus [10].[5] Our work differs in that our system is logic-agnostic, and our state spaces consist of a set of arbitrary formulae that may get added to and deleted in keeping with performed actions. Answer-set planning, recently surveyed in [46], translates planning problems to logic problems whose answer sets correspond to solutions or plans for the original planning problem. Instead of finding proofs that actions are applicable or that the goal is reached, answer-set planners find an answer set or logical model that corresponds to a solution. Approaches within answer-set planning typically require the grounding of predicates prior to search. For conformant problems, the number of states within a belief state may grow exponentially with the number of unknown predicates. In our work, it is often the case that the state space does not increase in size with the number of unknown formulae, since a tautology $\phi \vee \neg\phi$ does not need to be included within a state space. Multiple techniques have been introduced to compile conformant planning problems into classical planning problems via a sound yet incomplete method [33], a complete method for a bounded contingent width [34], and a linear translation for problems of contingent width 1 [5]. All these methods require knowing ahead of time the contingent width of the problem, or in other words the maximum number of uncertain state variables that interact through conditional effects.

[5] Readers unfamiliar with GOLOG can start with [27]. A number of interesting questions arise from comparison of PLATAS with our approach, for future analysis and engineering. We mention two very briefly: (1) As mentioned in [10], and discussed at some length lucidly in the complementary [41], in this line of work it is often desirable to consider restrictions in underlying declarative expressivity (e.g. to the PDDL fragment ADL). In contrast, we find such restrictions to be in considerable tension with human cognition, in planning. (2) An inferential backbone of resolution (GOLOG is written in Prolog) is in many ways incontestably advantageous, but we are guided by the fact that first-rate human deductive reasoning is almost invariably carried out in natural deduction, first invented for the extensional case in 1935 [16, 25].

3 STRIPS-inspired Planning over Automated Reasoning

Taking inspiration from STRIPS, we model our planning problem with actions having addition and deletion effects. The main difference is that we're operating over *state spaces* as opposed to a single state. A planning-with-formulae (PwF) problem Π is the tuple $\langle \mathcal{L}, \mathcal{A}, \Gamma_0, G \rangle$ where $\mathcal{L}$ is some logic, $\mathcal{A}$ is the set of lifted actions that take an agent from one state space to another, $\Gamma_0 \subset \mathcal{L}$ is the initial state space characterized by formulae from the logic, and $G \subset \mathcal{L}$ is a partial state space that represents a goal. Care must be taken to ensure that the initial state space is consistent; otherwise, for logics including the principle of explosion, the goal will be satisfied.

A lifted action $a \in \mathcal{A}$ is a tuple $\langle \chi, \texttt{Pre}, \texttt{Add}, \texttt{Del}, C \rangle$ where χ is a set of variables and `Pre` a partial state space, parameterized by χ and required to be be satisfied in order for an action to be taken. `Add` is the set of formulae, parameterized by χ that get added to the state space Γ when the action is taken. `Del` is the set of formulae, parameterized by χ, that gets removed from the state space Γ when the action is taken. Lastly, $C \in \mathbb{N}$ is the cost of the action, assumed to be 1 if left unspecified. A substitution is a mapping from a variable

to a term. Let $Dom(\sigma)$ return the domain of a substitution σ. Then for a given action a, a substitution σ is valid iff it has a mapping for all the variables in χ within the action. That is, $\forall v \in \chi(a), v \in Dom(\sigma)$. A ground action $ground(a, \sigma)$ from an action a and valid substitution σ is the tuple $\langle \emptyset, \texttt{Pre}', \texttt{Add}', \texttt{Del}', C \rangle$. In this tuple, $\texttt{Pre}' = \{f\sigma \mid f \in \texttt{Pre}\}$. A similar formulation is defined for $\texttt{Add}'$ and $\texttt{Del}'$. We denote a_g to be an arbitrary grounding of an action a. A state space Γ satisfies a partial state space γ iff $\Gamma \vdash \gamma$. A ground action a_g is applicable in state space Γ_t iff Γ_t satisfies $\texttt{Pre}(a_g)$. If an applicable ground action a_g is executed on state space Γ_t, then the next state space is $\Gamma_{t+1} = (\Gamma_t - \texttt{del}(a_g)) \cup \texttt{add}(a_g)$. A solution to the PwF problem Π is a plan $\pi = (a_1, \ldots, a_n)$ that takes an agent from an initial state space to a state space that satisfies the goal, where each $a_i \in \pi$ is an applicable action.

There are two key points in this formulation where an automated reasoner is crucial; the first is in deciding whether or not a given state space Γ_t satisfies the goal G. This is determined as an entailment check in an automated reasoner (e.g. $\Gamma_t \vdash G$). Secondly, we rely on the automated reasoner to find the set of applicable grounded actions A_g for a given action a. This is equivalent to finding the set of substitutions σ that when applied to our precondition, the grounded precondition holds under the current state space Γ_t. For an arbitrary $\sigma_i \in \sigma$, i.e. for all $f \in \texttt{Pre}(a)$, we have $\Gamma_t \vdash f\sigma_i$. It is possible that the size of σ is empty or infinite depending on the theory used. If σ is empty, then we are unable to apply any grounded actions from that particular lifted action. In the infinite case, we would need to place a bound on the length of σ returned by the automated reasoner.

In addition to changes in the planning structure, Spectra relaxes a few of the commonly made assumptions in model-based automated planning. Spectra does not assume CWA, domain closure, or the unique-name assumption. Nor does Spectra require that all grounded actions are found in the beginning portion of the algorithm. Instead, Spectra relies on the question-answering algorithm to iteratively find applicable actions.

4 Implementation

Spectra is implemented using the programming language Java and it's source code is available on GitHub (`https://github.com/rairlab/spectra`). Similar to model-based automated planners, we employ the A^* search algorithm as our core loop in order to find K plans that solve the given PwF problem P. An item in our search space ω is a tuple $\langle \Gamma_t, \pi \rangle$, where Γ_t is a state space and π the plan to get the initial state space Γ_0 to Γ_t. We make use of a priority queue Ω consisting of tuples ω. Each $\langle \Gamma_t, \pi \rangle \in \Omega$ is assigned a priority $h(\Gamma_t) + C(\pi)$, where h is the heuristic function over state spaces and C the cost function over plans. In our current implementation, we assign an uninformed heuristic over all state spaces, i.e. $\forall \Gamma, h(\Gamma) = 0$. Future work will include identifying domain-independent heuristics that can apply over a wide range of logics. Given an admissible heuristic, the first plan we find will be optimal. However, the other generated plans may not be optimal due to duplicate pruning. $C(\pi)$ is the cost of a plan π, which is the sum of the cost of all actions within π, i.e.

$$C(\pi) = \sum_{a \in \pi} C(a).$$

Algorithm 1 A^* Search Over PwF Problem

```
1:  procedure PLAN(L, A, Γ_0, G, K)
2:      Initialize priority queue Ω = [⟨Γ_0, ()⟩]
3:      Π = []
4:      while Ω.notEmpty() and |Π| < K do
5:          ⟨Γ_t, π_t⟩ = Ω.pop()
6:          if Γ_t satisfies G (*) then
7:              Π = Π ∪ {π_t}
8:          for lifted action a ∈ A do
9:              A_g = applicable grounded actions in Γ_t (*)
10:             for ground action a_g ∈ A_g do
11:                 Γ_{t+1} = (Γ_t − del(a_g)) ∪ add(a_g)
12:                 if Γ_{t+1} not already seen then
13:                     π_{t+1} = π_t ∪ {a_g}
14:                     Set priority P to h(Γ_{t+1}) + C(π_{t+1})
15:                     Add ⟨Γ_{t+1}, π_{t+1}⟩ to Ω with priority P.
16:     return Π
(*) denotes usage of the automated reasoner described in §3
```

The core search procedure is specified in Algorithm 1. We perform an A^* search over the transition space until either no more applicable actions can be performed, or the number of requested plans have been met. Viewing the core loop from the perspective of an arbitrary $\omega = \langle \Gamma_t, \pi_t \rangle \in \Omega$: We first check if the current state space Γ_t satisfies our goal. This is equivalent to making a call to an automated reasoner to see if $\Gamma_t \vdash G$. If so, we add π_t — the plan to get from Γ_0 to Γ_t — to the list of found plans. Then for each lifted action $a \in A$, we find the set of applicable ground actions for the state space Γ_t. As described in §3, we rely on the question-answering algorithm of the automated reasoner to find a set σ of valid substitutions for the given action. If σ is non-empty, then for each $\sigma_i \in \sigma$ we ground the lifted action. With each grounded action, we compute a new ω consisting of our new state space Γ_{t+1} and the plan to get from the initial state space to Γ_{t+1}; i.e.

$\omega = \langle \Gamma_{t+1}, \pi_t \cup \{a_g\} \rangle$. This new ω gets added to the priority queue if Γ_{t+1} has not already been visited.

5 Cognitive Planning with $\mathcal{DCC}$

ShadowProver [19] is an automated reasoner over cognitive calculi such as $\mathcal{DCEC}$. It is under active development with support for entailment and enhanced question-answering. Again, herein we use the fragment $\mathcal{DCC}$ of $\mathcal{DCEC}$ in which the number of cognitive operators are reduced and all timepoints are identified. Consider the Grapevine domain from [32]. In this problem, a group of agents have a secret they may wish to communicate with each other. Agents in this environment can move freely between rooms and broadcast to everyone in the room their secret. The initial state of this problem specifies that agents only know their own secret. One possible goal is for an agent to propagate their secrets to a subset of agents. Let's analyze an instance of this problem with $n = 3$ agents and $p = 2$ rooms. As before, each agent has their own secret, and no one except a believes a's secret. Ignoring type predicates and unique-name axioms, the initial state space can be characterized as

$$\begin{aligned}\Gamma_0 = \{&at(a,p1), \neg at(a,p2), at(b,p1), \neg at(b,p2), at(c,p1)\\ &\neg at(c,p2), \mathbf{B}(a, the(a)), \mathbf{B}(b, the(b)), \mathbf{B}(c, the(c)),\\ &\neg \mathbf{B}(b, the(a)) \neg \mathbf{B}(c, the(a))\}\end{aligned}$$

The goal for this problem includes three components: 1) agent b believes a's secret; 2) c does not believe a's secret; 3) agent a believes that agent b believes a's secret. The partial state space G is then specified as:

$$G = \{\mathbf{B}(b, the(a)), \neg \mathbf{B}(c, the(a)), \mathbf{B}(a, \mathbf{B}(b, the(a)))\}$$

The available actions are: `left`, `right`, `share-both`, `share-single`. We denote p_1 as the left room and p_2 as the right room. For an agent to move left, they must be in the right room; *vice versa* for the right action. For the `share-both` action, all agents must be in the same room. One agent then shares their secret, and both agents believe the secret, and the agent sharing believes that the other agents believe the secret. The action `share-single` is similar; however, the precondition is that one agent is not in the same room, and that agent does not gain a belief about the secret. We provide an example of how the share-single action would get encoded in Spectra in Figure 1. Note that in the example we delete the negation of the added formulae in order to stay consistent. The `Believes!` keyword denotes the model operator $\mathbf{B}$ within $\mathcal{DCC}$, and the predicate `the` represents the secret of the agent specified in its parameter. The figure does not provide a complete example, as one would need to add type restrictions as well as restrictions that all arguments are unique for that lifted action. Requesting $K = 2$ plans from Spectra provides the following two plans:

```
(define-action share-single [?a1 ?a2 ?a3 ?r] {
  :preconditions [
    (at ?a1 ?r)
    (at ?a2 ?r)
    (not (at ?a3 ?r))
  ]
  :additions [
    (Believes! ?a2 (the ?a1))
    (Believes! ?a1 (Believes! ?a2 (the ?a1)))
  ]
  :deletions [
    (not (Believes! ?a2 (the ?a1)))
    (not (Believes! ?a1 (Believes! ?a2 (the
        ?a1))))
  ]
})
```

Fig. 1: Share-Single Action from Grapevine

$$\pi_1 = ((\texttt{right}\ a)\ (\texttt{right}\ b)\ (\texttt{sharesingle}\ a\ b\ c\ p_2))$$

and

$$\pi_2 = ((\texttt{right}\ c)\ (\texttt{sharesingle}\ a\ b\ c\ p_1)).$$

6 Planning Under Uncertainty

In the last example, we looked at a problem whose state space is equivalent to a single state. In this section, we discuss how to move beyond this restriction and consider state spaces that represent multiple possible states. This is equivalent to planning under uncertainty. To begin, let us put aside $\mathcal{DCC}$ and consider Spectra using a classical first-order-logic reasoner. What we discuss here will easily extend to the epistemic uncertainty case. So, consider the "safe problem" from the conformant planning literature [36]. In this problem, there is a closed safe and the agent has only one correct combination out of a collection thereof that can open that safe. At the outset of the problem, the agent does not know which one of these combinations is correct. Narrowing to an instance of this problem, consider two possible combinations c_1 and c_2. The initial state space of this problem is that one of the two combinations is correct. We can represent that as follows:

$$\Gamma_0 = \{(correct(c_1) \wedge \neg correct(c_2)) \vee (\neg correct(c_1) \wedge correct(c_2))\}$$

There is one available action, `try`, which takes a combination and, if it's correct, opens the safe. The

```
(define-action try [?x] {
    :preconditions [  ]
    :additions [
        (if (correct ?x) (safe-open))
    ]
})
```

Fig. 2: Action with Conditional Effect in Spectra

`try` action is shown in Figure 2. Note that the condition that the combination is correct is only specified in the addition effect. This is because for the action `try` to be applicable for a given state space Γ_t, all states $s \in \Gamma_t$ must satisfy the preconditions of `try`. Now imagine that the agent tried both combination c_1 and c_2 sequentially from the initial state. The updated state space is then:

$$\Gamma_2 = \Gamma_0 \cup \{correct(c_1) \rightarrow open, correct(c_2) \rightarrow open\}$$

The alert reader might notice that Γ_2 satisfies the goal of opening the safe. This can be shown by proof-by-cases on the disjunctive formula and applying the appropriate *modus ponens* for each case.

The Need for Defeasible Reasoning

A challenge arises for our STRIPS-inspired model when we want to swap the valuation of an unknown formula ϕ. More formally, suppose at a state space Γ_t that ϕ is unknown; i.e. $\Gamma_t \nvdash \phi$ and $\Gamma_t \nvdash \neg\phi$. Let's consider that ϕ holds for some states s within Γ_t and denote this set as Γ_t^+. The parallel, assume, holds for the negated case; the set here is Γ_t^-. Now assume that after we execute some applicable action, we want $\neg\phi$ to hold in the next state space for all states in Γ_t^+, and ϕ to hold in the next state space for all states in Γ_t^-. How exactly can we model this? To make the formal challenge more concrete, consider an alternative to the `safe` problem from before. Three lockers A, B, C appear before the agent with a button to toggle whether locker A is locked or not. At the initial configuration there are two possibilities, either $\Gamma_{0,0} = locked(A) \wedge locked(B) \wedge \neg locked(C)$ or $\Gamma_{0,1} = \neg locked(A) \wedge locked(B) \wedge locked(C)$. Therefore, the initial state space $\Gamma_0 = \{\Gamma_{0,0} \vee \Gamma_{0,1}\}$. Note from the initial state that the agent does not know whether the locker A is locked.

Note that in "real life" such epistemic indeterminacy with regard to binary conditions of devices and systems is far from uncommon. That is, humans are often forced to be rationally agnostic belief-wise when faced with wanting to know whether or not some system s has property R (i.e., a human sometimes must rationally believe neither $R(s)$ nor $\neg R(s)$), while at the same time having the ability to perform an action that will cause either $R(s)$ or $\neg R(s)$ to obtain. For example, when leaving his home in a rush for a crucial engagement, Smith activates the alarm system for it at an interior keypad. When he returns home he does not remember whether he activated the alarm or not. As in our safe scenario, his combination applied to an external keypad toggles the system on if off, and off if on, and he doesn't recall what the indicator lights indicate on this external keypad. He must remain noncommittal after tapping in the combination, and resort to other approaches to relieve his agnosticism.

One approach to tackling the toggle action in the simpler safe challenge, which is better for technical exposition, is to use FOL to have the following effects added to the state space: (1) $locked(A) \rightarrow \neg locked(A)$ and (2) $\neg locked(A) \rightarrow locked(A)$. However, both of these effects together yield a contradiction by simple deduction.

To address the problem, we can use defeasible reasoning.[6] Consider a defeasible logic such as $\mathcal{IDCEC}$ [9, 17]. The schema in $\mathcal{IDCEC}$ Belief Propagation box shows that beliefs are propagated forward in time as long as doing so doesn't contradict any newer beliefs. At the initial state ($t = 0$), we can represent the state space as $\Gamma_0 = \{B(a, 0, \Gamma_{0,0} \vee \Gamma_{0,1})\}$. Then for the toggle action we can set the effects to: (1) $B(a, t, locked(A)) \rightarrow B(a, t+1, \neg locked(A))$ and (2) $B(a, t, \neg locked(A)) \rightarrow B(a, t+1, locked(A))$. These formulae are no longer contradictory and suitably capture the toggle action as the following noncommittal belief is entailed at the next time step:

$$B(a, 1, (\neg locked(A) \wedge locked(B) \wedge locked(C)) \vee (locked(A) \wedge locked(B) \wedge locked(C)))$$

7 Discussion and Conclusion

The extension from states with predicates to state spaces with arbitrary formulae catalyzes several challenges. The first is that the problem modeler must take care not

[6] We have already indicated that such logics are known also as *non-monotonic* logics. As a reminder given for fuller context, deduction is monotonic; i.e., if $\Phi \vdash \phi$ (which is to say that ϕ can be proved deductively from Φ), then for any formula ψ, $\Phi \cup \{\psi\} \vdash \phi$ holds. In stark contrast, non-monotonic logics, long created and implemented in AI (e.g. originally by Reiter and McCarthy [29, 40], with more expressive such logics more recently presented in [9, 37]) are such that new declarative information can invalidate what was earlier a valid inference. It may e.g. be rational to infer from Mr. Smith's telling you that it's raining that you should believe that it is, and you thus may believe it is; but if you then find out that Smith is a pathological liar under treatment for his condition, you will now *ceteris paribus* not believe that's raining.

$\mathcal{IDCEC}$ Belief Propagation

$$\frac{\mathbf{B}^{\ell}(a,t_1,\phi) \quad \Gamma \nvdash \neg\mathbf{B}^{\ell}(a,t_2,\phi) \quad t_1 < t_2}{\mathbf{B}^{\ell}(a,t_2,\phi)}\ I^{l}_{PROP}$$

to introduce any contradictions in the PwF problem. This is an issue conveniently absent from the STRIPS model, as the user does not specify any negated predicates (as those are assumed via CWA). If contradictions do exist in the PwF model, then for many logics, the goal would be satisfied for any arbitrary contradictory state space. This is due to *explosion* which is from falsum, anything follows. The second challenge is that the computational properties of Spectra are heavily reliant on the underlying automated reasoner. For example, if the underlying automated reasoner is undecidable for entailment and question-answering for a given logic, then Spectra will be incomplete. Such undecidability, in human-level planning, which presumably is a level that must ultimately be reached (or exceeded), is common and irrepressible. Even undergraduate students are routinely required to find proofs of formulae in not only first-order logic and other more expressive extensional logics (e.g. second-order logic), but in quantified modal logics as well. When they are tasked with performing actions that reach these goals, they are embodying the daunting challenge that, in our planning paradigm, Spectra faces.

Of course, this situation gives rise to the question as to how, with undecidability having to be part of what our engineering must factor in, performance is assessed/measured. In our current implementation, we rely on pragmatic time bounds for calls to the automated reasoner. A result returned under the bound for a timer is clear success, because that bound is set for applications at hand. No result before the expiration of the timer is failure, and something else must be tried. This approach is none other than what AI founder and nobelist Herbert Simon famously introduced under the banner "satisficing" as an approach to both human and AI planning and deciding (e.g. see [44]).[7]

Importantly, undecidability does not in any way stop engineering designed to mitigate it. Accordingly, when using Spectra with a monotonic logic, we for instance include optimizations to reduce the number of calls to the automated reasoner. For example, if we have cached that $\Gamma_1 \vdash \phi$, then we assume for an arbitrary Γ_2 that $\Gamma_1 \cup \Gamma_2 \vdash \phi$. Also, consider we have cached that $\Gamma_1 \nvdash \phi$. Then for $\Gamma_2 \subseteq \Gamma_1$, we assume that $\Gamma_2 \nvdash \phi$. Of course, in addition, predictably, our engineering in service of reasoner-based planning in the face of undecidability includes making use, whenever possible, of automated reasoners for decidable fragments of both first-order logic and propositional modal logics; details here are beyond present scope, but see [19].

Formulae in Spectra are treated strictly syntactically when updating state spaces. This is because in general consistency and redundancy checks are (again) undecidable for an arbitrary logic. Future work includes adding an additional layer which is able to perform quick (potentially incomplete) consistency and redundancy checks for a wide range of logics. For example, when deleting $A \wedge B$, this layer can additionally check for $B \wedge A$ and delete this. While this may not hold for logics in general, a wide class of logics share first-order semantics, which we can provide a default layering over. Additionally, in work described herein, we presented an uninformed heuristic $h(\Gamma) = 0$ for all state spaces Γ. Future work includes investigating heuristics that can hold for a wide range of logics.

To briefly recap, we presented Spectra, a logic-agnostic AI planner based on automated reasoning. We discussed how the extension from states with predicates to state spaces with arbitrary formulae enables high-expressivity processing and captures uncertainty in the PwF problem. We additionally discussed how using non-classical, intensional reasoners such as ShadowProver over $\mathcal{DCC}$ allows Spectra to create complex cognitive plans, such as ones with epistemic goals. Lastly, we discussed how defeasible logics in conjunction with Spectra can be used to solve a larger class of conformant planning problems. Future work also includes incorporating a perception model into Spectra. And the authors are particularly interested in using inductive reasoning to capture the adjudication of competing arguments an agent might need to face, as a way to carry out sophisticated defeasible reasoning in service of planning.

Acknowledgements The authors would like to thank Naveen Sundar Govindarajulu for developing the initial version of Spectra that used the first-order reasoner Snark (as well as, for some applications, ShadowProver; this was enabled in no small part by AFOSR and ONR, support to Bringsjord and Govindarajulu for which we are most grateful). This paper was supported in part by a fellowship award to Rozek under contract FA9550-21-F-0003 through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program, sponsored by the United States of America's Air Force Research Laboratory (AFRL), the Office of Naval Research (ONR), and the Army Research Office (ARO). Additionally, the authors are greatly indebted to ONR and AFOSR for the initial development of Spectra, for the continued support of the automated reasoner ShadowProver, and to both organizations for longstanding support of r&d in automated reason-

[7] It was none other than Simon, with Newell, who introduced to the world in 1956 the very first automated prover: LogicTheorist.

ing, planning, and logic-based learning. Finally, perspicacious feedback from three anonymous reviewers, and from the editors, was extremely helpful, and we give our thanks to them.

References

1. Arkoudas, K., Bringsjord, S.: Propositional Attitudes and Causation. International Journal of Software and Informatics **3**(1), 47–65 (2009). URL http://kryten.mm.rpi.edu/PRICAI_w_sequentcalc_041709.pdf
2. Bäckström, C., Nebel, B.: Complexity results for SAS+ planning. In: R. Bajcsy (ed.) Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993, pp. 1430–1435. Morgan Kaufmann (1993)
3. Bonet, B., Geffner, H.: Planning with incomplete information as heuristic search in belief space. In: Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, pp. 52–61 (2000)
4. Bonet, B., Geffner, H.: Planning as heuristic search. Artificial Intelligence **129**(1-2), 5–33 (2001). DOI 10.1016/S0004-3702(01)00108-4
5. Bonet, B., Geffner, H.: Flexible and scalable partially observable planning with linear translations. In: C.E. Brodley, P. Stone (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 - 31, 2014, Québec City, Québec, Canada, pp. 2235–2241. AAAI Press (2014). DOI 10.1609/AAAI.V28I1.9047. URL https://doi.org/10.1609/aaai.v28i1.9047
6. Braüner, T.: Hybrid-Logical Reasoning in the Smarties and Sally-Anne Tasks. Journal of Logic, Language and Information **23**, 415–439 (2014)
7. Bringjsord, S., Govindarajulu, N.S.: Deontic cognitive event calculus (formal specification) (2023). URL https://www.cs.rpi.edu/~govinn/dcec.pdf
8. Bringsjord, S., Giancola, M., Govindarajulu, N.S., Slowik, J., Oswald, J., Bello, P., Clark, M.: Argument-Based Inductive Logics, With Coverage of Compromised Perception. Frontiers in Artificial Intelligence **6** (2024). DOI https://doi.org/10.3389/frai.2023.1144569. URL https://www.frontiersin.org/articles/10.3389/frai.2023.1144569/
9. Bringsjord, S., Govindarajulu, N., Giancola, M.: Automated Argument Adjudication to Solve Ethical Problems in Multi-Agent Environments. Paladyn, Journal of Behavioral Robotics **12**, 310–335 (2021). URL http://kryten.mm.rpi.edu/AutomatedArgumentAdjudicationPaladyn071421.pdf
10. Claßen, J., Röger, G., Lakemeyer, G., Nebel, B.: PLATAS - integrating planning and the action language GOLOG. Künstliche Intell. **26**(1), 61–67 (2012). DOI 10.1007/S13218-011-0155-2. URL https://doi.org/10.1007/s13218-011-0155-2
11. Eppe, M., Dylla, F.: An epistemic planning system based on the event calculus. Tech. rep., Technical Report 033-11/2012, University of Bremen, Bremen (2012)
12. Fikes, R.E., Nilsson, N.J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial intelligence **2**(3-4), 189–208 (1971). DOI 10.1016/0004-3702(71)90010-5
13. Finzi, A., Pirri, F., Reiter, R., et al.: Open world planning in the situation calculus. In: AAAI/IAAI, pp. 754–760 (2000)
14. Fitting, M.: Intensional Logic. In: E. Zalta (ed.) The Stanford Encyclopedia of Philosophy (2015). URL https://plato.stanford.edu/entries/logic-intensional
15. Geffner, H.: Functional STRIPS: a more flexible language for planning and problem solving. Logic-based artificial intelligence pp. 187–209 (2000)
16. Gentzen, G.: Investigations into Logical Deduction. In: M.E. Szabo (ed.) The Collected Papers of Gerhard Gentzen, pp. 68–131. North-Holland, Amsterdam, The Netherlands (1935). This is an English version of the well-known 1935 German version.
17. Giancola, M.: Reasoning with cognitive likelihood for artificially-intelligent agents: Formalization & implementation. Ph.D. thesis, Rensselaer Polytechnic Institute (2023)
18. Govindarajulu, N., Bringsjord, S.: On Automating the Doctrine of Double Effect. In: C. Sierra (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), pp. 4722–4730. International Joint Conferences on Artificial Intelligence (2017). DOI 10.24963/ijcai.2017/658. URL https://doi.org/10.24963/ijcai.2017/658
19. Govindarajulu, N., Bringsjord, S., Peveler, M.: On Quantified Modal Theorem Proving for Modeling Ethics. In: M. Suda, S. Winkler (eds.) Proceedings of the Second International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements (ARCADE 2019), *Electronic Proceedings in Theoretical Computer Science*, vol. 311, pp. 43–49. Open Publishing Association, Waterloo, Australia (2019). URL http://eptcs.web.cse.unsw.edu.au/paper.cgi?ARCADE2019.7.pdf. The ShadowProver system can be obtained here: https://naveensundarg.github.io/prover/.
20. Green, C.: Application of theorem proving to problem solving. In: Readings in Artificial Intelligence, pp. 202–222. Elsevier (1981). DOI 10.1016/B978-0-934613-03-3.50019-2
21. Haslum, P., Lipovetzky, N., Magazzeni, D., Muise, C., Brachman, R., Rossi, F., Stone, P.: An introduction to the planning domain definition language, vol. 13. Springer (2019). DOI 10.1007/978-3-031-01584-7
22. Haslum, P., et al.: Computing genome edit distances using domain-independent planning pp. 45–51 (2011)
23. Helmert, M.: The fast downward planning system. Journal of Artificial Intelligence Research **26**, 191–246 (2006). DOI 10.1613/jair.1705
24. Helmert, M., Domshlak, C.: Landmarks, critical paths and abstractions: What's the difference anyway? In: A. Gerevini, A.E. Howe, A. Cesta, I. Refanidis (eds.) Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009. AAAI (2009). URL http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/735
25. Jáśkowski, S.: On the Rules of Suppositions in Formal Logic. Studia Logica **1**(1), 5–32 (1934)
26. Katz, M., Domshlak, C.: Optimal additive composition of abstraction-based admissible heuristics. In: J. Rintanen, B. Nebel, J.C. Beck, E.A. Hansen (eds.) Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008, pp. 174–181. AAAI (2008). URL http://www.aaai.org/Library/ICAPS/2008/icaps08-022.php
27. Levesque, H., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.: GOLOG: A Logic Programming Language for Dy-

namic Domains. The Journal of Logic Programming **31**, 59–83 (1997)
28. Matloob, R., Soutchanski, M.: Exploring organic synthesis with state-of-the-art planning techniques. In: Proc. SPARK Workshop, pp. 52–61 (2016)
29. McCarthy, J.: Circumscription—A Form of Non-Monotonic Reasoning. Artificial Intelligence **13**, 27–39 (1980)
30. Mueller, E.T.: Automating commonsense reasoning using the event calculus. Communications of the ACM **52**(1), 113–117 (2009). DOI 10.1145/1435417.1435443
31. Mueller, E.T., Sutcliffe, G.: Reasoning in the event calculus using first-order automated theorem proving. In: FLAIRS Conference, pp. 840–841 (2005)
32. Muise, C., Belle, V., Felli, P., McIlraith, S., Miller, T., Pearce, A., Sonenberg, L.: Planning over multi-agent epistemic states: A classical planning approach. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29 (2015)
33. Palacios, H., Geffner, H.: Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA, pp. 900–905. AAAI Press (2006). URL `http://www.aaai.org/Library/AAAI/2006/aaai06-142.php`
34. Palacios, H., Geffner, H.: Compiling uncertainty away in conformant planning problems with bounded width. J. Artif. Intell. Res. **35**, 623–675 (2009). DOI 10.1613/JAIR.2708. URL `https://doi.org/10.1613/jair.2708`
35. Pednault, E.P.D.: ADL: exploring the middle ground between STRIPS and the situation calculus. In: R.J. Brachman, H.J. Levesque, R. Reiter (eds.) Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89). Toronto, Canada, May 15-18 1989, pp. 324–332. Morgan Kaufmann (1989)
36. Petrick, R.P., Bacchus, F.: A knowledge-based approach to planning with incomplete information and sensing. In: AIPS, vol. 2, pp. 212–222 (2002)
37. Pollock, J.: Defasible Reasoning with Variable Degrees of Justification. Artificial Intelligence **133**, 233–282 (2001)
38. Pommerening, F., Helmert, M., Röger, G., Seipp, J.: From non-negative to general operator cost partitioning. In: B. Bonet, S. Koenig (eds.) Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, pp. 3335–3341. AAAI Press (2015). DOI 10.1609/AAAI.V29I1.9668. URL `https://doi.org/10.1609/aaai.v29i1.9668`
39. Poole, D.: A framework for decision-theoretic planning I: Combining the situation calculus, conditional plans, probability and utility pp. 436–445 (1996)
40. Reiter, R.: A Logic for Default Reasoning. Artificial Intelligence **13**, 81–132 (1980)
41. Röger, G., Helmert, M., Nebel, B.: AAAI Press (2008)
42. Schiffel, S., Thielscher, M.: Reconciling situation calculus and fluent calculus. In: AAAI, vol. 6, pp. 287–292 (2006)
43. Shanahan, M.: An abductive event calculus planner. The Journal of Logic Programming **44**(1-3), 207–240 (2000). DOI 10.1016/S0743-1066(99)00077-1
44. Simon, H.: Rational Choice and the Structure of the Environment. Psychological Review **63**(2), 129–138 (1956)
45. Soutchanski, M., Young, R.: Planning as theorem proving with heuristics. arXiv preprint arXiv:2303.13638 (2023). DOI 10.48550/arXiv.2303.13638
46. Tran, S.C., Pontelli, E., Balduccini, M., Schaub, T.: Answer set planning: A survey. Theory and Practice of Logic Programming **23**(1), 226–298 (2023). DOI 10.1017/S1471068422000072