On Plans With Loops and Noise

Vaishak Belle
University of Edinburgh & Alan Turing Institute
vaishak@ed.ac.uk

ABSTRACT
In an influential paper, Levesque proposed a formal specification for analysing the correctness of program-like plans, such as conditional plans, iterative plans, and knowledge-based plans. He motivated a logical characterisation within the situation calculus that included binary sensing actions. While the characterisation does not immediately yield a practical algorithm, the specification serves as a general skeleton to explore the synthesis of program-like plans for reasonable, tractable fragments.

Increasingly, classical plan structures are being applied to stochastic environments such as robotics applications. This raises the question as to what the specification for correctness should look like, since Levesque’s account makes the assumption that sensing is exact and actions are deterministic. Building on a situation calculus theory for reasoning about degrees of belief and noise, we revisit the execution semantics of generalised plans. The specification is then used to analyse the correctness of example plans.

KEYWORDS
Generalised planning; program-like plans; nondeterminism; noisy acting and sensing; reasoning about knowledge and belief

1 INTRODUCTION
In an influential paper, Levesque [1] proposed a formal specification for analysing the correctness of program-like plans, such as conditional plans, iterative plans, and knowledge-based plans. The problem setting is this: in a world where the agent can affect and sensing; reasoning about knowledge and belief

In this paper, there is a natural evolution of theory and formulation when compared to Levesque’s account. The original account was based on the situation calculus extended for knowledge and sensing [9] derived from classical epistemic logic [10]. Our account is based on the situation calculus extended for probabilistic belief and noise [11] derived from probabilistic epistemic logic [12, 13]. In principle, of course, any logical language for reasoning about actions and probabilities could have been used for the formalisation,
e.g. [14]. But by using the situation calculus, we can clearly explicate the generalisation from Levesque’s account, but also benefit from its first-order expressiveness, at least for axiomatising the planning domain. Moreover, to logically characterise unbounded iteration and transitive closure, we use second-order logic, as would Levesque.

Its worth remarking that while this logical machinery makes the account more involved than (say) POMDP specifications [15], having a more general language is useful for contextualising involved extensions such as the handling of non-unique prior distributions. For example, in [16], it is argued that when planning in highly stochastic and unknown environments, it is useful to allow a margin of error in what the values of fluents will be. This means that the planning system has to reason about multiple distributions satisfying such constraints, and as in [16], the use of logical connectives allows us to express such scenarios effortlessly.

2 A THEORY OF KNOWLEDGE AND ACTION

We will not go over the language \( \mathcal{L} \) of the situation calculus in detail [17, 18], but simply note that it is a many-sorted dialect of predicate calculus, with sorts for actions, situations, denoting a (possibly) empty sequence of actions, and objects, for everything else. A special constant \( S_0 \) denotes the real world initially, and the term \( do(a, s) \) denotes the situation obtained on doing \( a \) in \( s \). Fluents, whose last argument is always a situation, can be used to capture changing properties. Following [18], application domains are axiomatised as basic action theories, which stipulate the conditions under which actions are executable, and their affects on fluents, while embodying a monotonic solution to the frame problem. For example, for the tree chop problem, using the functional fluent \( d \) to mean the thickness of the tree, we may have:

\[
\begin{align*}
\text{Poss}(\text{chop}(x), s) & \equiv d(s) \geq x. \\
\text{do}(a, s) & = \equiv (a = \text{chop}(x) \land d(s) = u + x) \lor (a \neq \text{chop}(x) \land d(s) = u).
\end{align*}
\]

We let \( \text{chop} \) be an abbreviation for \( \text{chop}(1) \). These axioms say that \( \text{chop} \) is only possible when the tree’s thickness is non-zero, and that the current value of \( d \) is \( u \) if and only if its previous value was also \( u \) and no chop action occurred, or its previous value was \( u + 1 \) and a single chop action was executed.

A special binary fluent \( K(s’, s) \) denotes that \( s’ \) is a possible world when the agent is at \( s \). As usual, knowledge is defined as truth at accessible worlds:

\[
\text{Know}(s, s) \equiv \forall s’. K(s’, s) \supset \phi[ s’ ].
\]

A modeler axiomatises the initial beliefs of the agent:

\[
K(i, S_0) \supset (1 \leq d(i) \leq 10). \tag{1}
\]

This is equivalently written using \( \text{Know} \) and a special term \( \text{now} \) to denote the current situation as:

\[
\text{Know}(d(\text{now}) = 1 \lor \ldots \lor d(\text{now}) = 10, S_0).
\]

The observations obtained by the agent are described using a special function \( SF \). In the tree chop problem, we may have:

\[
SF(a, s) = \begin{cases} 
\text{down} & a = \text{getd} \land d(s) = 0 \\
\text{up} & a = \text{getd} \land d(s) \neq 0 \\
0 & \text{otherwise}
\end{cases}
\tag{2}
\]

which says that on doing the sensing action \( \text{getd} \), if the tree is still standing, the sensor returns \( \text{up} \), else it would return \( \text{down} \).

A fixed successor state axiom for \( K \) then determines how the agent’s knowledge changes over actions:

\[
K(s’, do(a, s)) \equiv \exists s’’[ K(s’’) \land s’ = do(a, s’’) \land \text{Poss}(a, s’’’) \land SF(a, s’’’) = SF(a, s)]
\]

which has the effect of eliminating worlds that disagree with truth at the real world, leading to knowledge expansion.

Plans with loops

We will be interested in computable program-like plans. We consider finite state controllers, which are fairly common in the literature [3, 4, 19].

Definition 2.1. Suppose \( \mathcal{A} \) is a finite set of (parameterless) action terms, and \( \mathcal{O} \) a finite set of objects, denoting observations. A finite memoryless plan \( X \) is a tuple \( \langle Q, Q_0, Q_F, \gamma, \delta \rangle \):

- \( Q \) is a finite set of control states;
- \( Q_0 \in Q \) is the initial state, and \( Q_F \in Q \) the final one;
- \( \gamma \in [Q^- \times O] \) is a labelling function for \( Q^- = Q \setminus \{Q_F\} \);
- \( \delta \in [Q^- \times Q] \) is a transition function.

Example 2.2. For Figure 1, we might have \( Q = \{Q_0, Q, Q_F\} \), \( \gamma(Q_0) = \text{chop}, \gamma(Q) = \text{getd}, \delta(Q, \text{up}) = Q_0, \delta(Q, \text{down}) = Q_F, \) and \( \delta(Q_F, 1) = Q_F \).

Informally, we may think of applying these plans in an environment as follows: starting from \( Q_0 \) that advises the action \( \gamma(Q_0) \), the environment executes that action and changes externally to return \( o \in O \). Then, internally, we reach the control state \( \delta(Q_0, o) \) and so on, until \( Q_F \). To reason about these plans in an environment enabled in the situation calculus, we need to encode the plan structure as a \( \mathcal{L} \)-sentence, and axiomatise the execution semantics over situations, for which we follow [19] and define:

Definition 2.3. Let \( \Sigma \) be the union of the following axioms for:

1. domain closure for the control states: \( \forall q \in Q \) \( \{ q = Q_0 \lor q = Q_1 \lor \ldots \lor q = Q_N \lor q = Q_F \} \);
2. unique names axiom for the control states: \( Q_i \neq Q_j \) for \( i \neq j \);
3. action association: \( \forall Q \in Q^- \) of the form \( \gamma(Q) = a \);
4. transitions: \( \forall Q \in Q^- \) of the form \( \delta(Q, o) = Q’ \).

Assume terms such as \( \text{chop} \in \mathcal{A} \) and \( (\text{up}, \text{down}) \subseteq O \).
Definition 2.4. We use $T^*(q, s, q', s')$ as abbreviation for $\forall T[\ldots \supset T(q, s, q', s')]$, where the ellipsis is the conjunction of the universal closure of:

- $T(q, s, q, s)$
- $T(q, s, q', s') \land T(q', s', q', s') \supset T(q, s, q', s')$
- $\gamma(T) = a \land \text{Poss}(a, s) \land SF(a, s) = 0$
- $\delta(T, a) = q' \supset T(q, s, q', do(a, s))$

In English: $T^*$ is the reflexive transitive closure of the one-step transitions in the plan.

We are now prepared to reason about plan correctness:

Definition 2.5. For any goal formula $\phi \in \mathcal{L}$, basic action theory $D$, plan $X$, and its encoding $\Sigma$, we say $X$ is correct for $\phi$ iff

$D \cup \Sigma \models \forall s. K(s, S_0) \supset \exists s'[T^*(Q_0, s, Q_F, s') \land \phi(s')]$.

Example 2.6. Let $D_{dyn}$ denote the tree chop axioms, and then it is easy to see that $D_{dyn} \cup \{(1, 2)\} \cup \Sigma$, where $\Sigma$ represents the encoding of Figure 1, is correct for the goal $d = 0$.

## 3 A THEORY OF PROBABILISTIC BELIEFS

Our objective now is to generalise the above well-understood framework on knowledge and loopy plans to a stochastic setting. The account of knowledge, deterministic acting and exact sensing was extended by Bacchus, Halpern, and Levesque [11] – BHL henceforth – to deal with degrees of belief in formulas, and in particular, with how degrees of belief should evolve in the presence of noisy sensing and acting, in accordance with Bayesian conditioning. The main advantage of a logical account like BHL is that it allows a specification of belief that can be partial or incomplete, in keeping with whatever information is available about the application domain. The account is based on 3 distinguished fluents: $p, l$ and $alt$. The $p$ fluent here is a numeric analogue to $K$ in that $p(s', s)$ denotes the weight (or density) accorded to $s'$ when the agent is at $s$. Initial constraints about what is known can be provided as usual. For example:

$$p(t, S_0) = \begin{cases} .1 & \text{if } 1 \leq d(t) \leq 10 \\ 0 & \text{otherwise} \end{cases}$$

(3)

is saying that the tree’s thickness $d$ takes a value that is uniformly drawn from $\{1, \ldots, 10\}$. We provide a definition for a belief modality $\text{Bel}$, but the above constraint can be equivalently written as:

$$\text{Bel}(d(\text{now})) = 1, S_0) = .1$$

and so on for the other values. As argued earlier, the logical account also allows for uncertainty about the prior distribution – as needed, for example, in [16] – by means of expressions such as:

$$\text{Bel}(d(\text{now})) = 1, S_0) \geq .05$$

which says that any distribution where $d$ takes a value of 1 with a probability greater than or equal to .05 is a permissible one.

The $l$ fluent is used to express the likelihoods of outcomes. For example, suppose we had a sensor that would inform the agent about the numeric value of the $d$ fluent in a situation. Then an axiom of the form:

$$l(\text{get}(d)(z), s) = N(z; d(s), .25)$$

(4)

says that the observed value on the sensor is normally distributed around the true value, with a variance of .25. In robotics terminology [20], the sensor is said to have a Gaussian error profile.

To handle noisy actions, the idea is that if $\text{chop}(x)$ represents a chop that decrement’s the tree’s thickness by $x$ units, assume a new action type $\text{chop}(x, y)$ in that $x$ is the intended argument and $y$ is taken to be what actually happens. These are chosen by nature, and as such, out of the agent’s control. These action types are retrofitted in successor state axioms:

$$d(\text{do}(a, s)) = u \equiv (a = \text{chop}(x, y) \land u = d(s) - y) \lor (a \neq \text{chop}(x, y) \land u = d(s))$$

(5)

says that $d$ is actually affected by the second argument.

Since the agent is assumed to not control $y$, we use $alt$ to model the possible alternatives to an intended action:

$$\text{alt}(\text{chop}(x, y), a', z) \equiv a' = \text{chop}(x, z)$$

(6)

says that, for example, $\text{chop}(1, 2)$ is indistinguishable from $\text{chop}(1, 3)$. To suggest that some alternatives are more likely than others, an axiom like

$$l(\text{chop}(x, y), s) = N(y; x, .25)$$

(7)

then says that the actual value is normally distributed around the intended value, with a variance of .25. In robotics terminology, this is the equivalent to an action having additive Gaussian noise.

Likelihoods and $alt$-axioms determine the probability of successors, enabled by the following successor state axiom:

$$p(s', d(a, s)) = u \equiv$$

$$\exists a', z, s'' \left[ \text{alt}(a, a', z) \land s' = d(a', s'') \land$$

$$u = p(s'', s) \times l(a', s'') \lor$$

$$\neg \exists a', z, s'' \left[ \text{alt}(a, a', z) \land s' = d(a', s'') \land Poss(a', s'') \land u = 0 \right]$$

which essentially says that if two situations $s$ and $s'$ are considered epistemically possible, on doing $a$ at $s$, $d(a, s)$ and $d(b, s')$ will also be considered epistemically possible, where $b$ is any $alt$-related action to $a$. Moreover, the $p$-value of $d(b, s')$ is that of $s'$ multiplied by the likelihood of the outcome $b$.

Putting all this together, the degree of belief in $\phi$ at $s$ is defined as the weight of worlds where $\phi$ is true:

$$\text{Bel}(\phi, s) = \sum_{s': \phi(s')} p(s', s)$$

We write $K(s', s)$ to mean $p(s', s) > 0$, and $\text{Know}(\phi, s) \equiv \text{Bel}(\phi, s) = 1$. Finally, note that when the likelihood models are trivial, that is: $\forall a, s. l(a, s) = 1$, we are in the setting of nondeterministic but non-probabilistic acting and sensing.

## 4 CONTROLLERS WITH NOISY ACTING

Our first objective will be to motivate a definition for analysing the correctness of plan structures when actions are noisy (that is, they are non-deterministic, and the actual outcome is not directly observable). We assume, for now, that sensing is exact. This then also handles the case where actions are non-deterministic but observable immediately after, by way of a sensing action to inform the

---

* A more involved version would introduce $alt$ as a fluent, allowing possible outcomes to be determined by a situation.
planner about the outcome. As far as the syntax of the plan structure goes, we will not want it to be any different than Definition 2.1; however, we will need to revisit Definition 2.4 to internalise the noisy aspects of acting by using \textit{alt}.

\textbf{Definition 4.1.} We use $U'(q, s, q', s')$ as abbreviation for $\forall U[\ldots \supset U(q, s, q', s')]$, where the ellipsis is the conjunction of the universal closure of:

- $U(q, s, q, s)$
- $U(q, s, q', s') \land U(q'', s'', s', s') \supset U(q, s, q', s')$
- $\gamma(q) = a \land \exists b, z \ (alt(a, b, z) \land Pos(b, s) \land SF(b, s) = a \land \delta(q, o) = q') \supset U(q, s, q', do(b, s))$.

The main new ingredient here over $T^*$, of course, is how control states transition from a situation to a successor. The reflexive transitive closure of $U$ basically says that if the controller advises $a$ and $b$ is any action that is \textit{alt}-related to $a$, we consider the transition wrt the executability and the sensing outcome of the action $b$. The idea, then, is allow $U'$ to capture the least set that accounts for all the successors of a situation where a noisy action is performed. We define:

\textbf{Definition 4.2.} For any goal formula $\phi \in \mathcal{L}$, basic action theory $D$, plan $X$ and its encoding $\Sigma$, we say $X$ is correct for $\phi$ iff

$$D \cup \Sigma \models \forall s. K(s, S_0) \supset \exists s'[U'(q_0, s, Q_F, s') \land \phi(s')]$$

It can be shown that the new semantics coincides with Levesque’s account when the action theory is noise-free: that is, \textit{alt}-axioms are trivial $\forall x (alt(a, a', x) \equiv a = a')$, and $I$ mimics the behavior of $SF$ in assigning 1 to situations that agree with the sensing outcome and 0 to those that disagree. Then:

\textbf{Theorem 4.3.} Suppose $D$ is a noise-free action theory, $X$ and $\Sigma$ are as above, and $\phi$ is any situation-suppressed formula not mentioning the fluent $K$. Then, $X$ is correct for $\phi$ in the sense of Definition 2.5 iff $X$ is correct in the sense of Definition 4.2.

\textbf{Proof.} $U'$ differs from $T^*$ is only one aspect, that of \textit{alt}-related actions governing the transition to a successor situation. By assumption, $\forall x (alt(a, a', x) \equiv a = a')$, so Definition 2.5’s constraint on $T^*$ coincides with Definition 4.2’s constraint on $U'$.

Definition 4.2 only tests for a single goal-satisfying path, which is often referred to as a weak plan [2]. A property like \textit{termination} could be formalised using:

$$D \cup \Sigma \models \forall s. K(s, S_0) \supset \forall s'[U'(q_0, s, q, s') \supset \exists s'' [U'(q, s', Q_F, s'')]$$.

It is well-known [2] that in the absence of nondeterminism, termination is implied by the existence of a goal-satisfying path.\footnote{There are, of course, a number of other criteria in terms of which one characterises plan execution [2], a discussion of which is orthogonal to the issues of interest here and are hence omitted. Major criteria include \textit{fairness}, where if a nondeterministic action is executed infinitely many times then every outcome is assumed to occur infinitely often, and \textit{acyclicity}, where the same state is not allowed to be visited twice in plan execution paths.}

\textbf{Proposition 4.4.} Suppose $D, X$ and $\phi$ are as above. If $X$ is correct for $\phi$ in the sense of Definition 2.5 then (8) holds.

\textbf{Example 4.5.} Imagine a tree chop problem with noise-free sensing (i.e., let $SF$ work as in (2)), but with noisy actions. Let $chop \in A$ correspond to the $L$-action $chop(1, 1)$, with

$$l(chop(x, y), s) = \begin{cases} 0.9 & \text{if } x = y \\ 1 & \text{if } y = 0 \\ 0 & \text{otherwise} \end{cases}$$.

That is, the chop action does nothing with a small probability. Letting the initial theory be a $p$-based axiom for (1), we see that the plan from Figure 1 is correct in the sense of Definition 4.2, and it is also a terminating plan, because regardless of how many times the action fails, there is clearly one execution path, that of the appropriate number of chops always succeeding, which stops after enabling the goal.

While Definition 4.2 looks at every non-zero initial world, weaker specifications are possible still. For example:

$$D \cup \Sigma \models \forall s. p(s, S_0) > \kappa \supset \exists s'[U'(q_0, s, Q_F, s') \land \phi(s')]$$

looks at worlds with weights $> \kappa$ whereas

$$D \cup \Sigma \models Bel(\exists s'[U'(q_0, now, Q_F, s') \land \phi(s')], S_0) \geq \kappa$$

says that the sum (or integral) of initial worlds where there is a weak plan is $\geq \kappa$. Of course, since $K(s', s)$ is an abbreviation for $p(s', s) > 0$, and $Know(\phi(s)) \equiv Bel(\phi, s) = 1$, we have:

\textbf{Proposition 4.6.} Definition 4.2 is equivalent to (9) for $k = 0$, and it is equivalent to (10) for $k = 1$.

\textbf{Example 4.7.} Imagine a tree chop problem with two types of trees, wooden ones and metal ones; the chop action has no effect on a metal tree [21]. Suppose we have 3 worlds: the first with a wooden tree of thickness 1 and weight $A$, the second with a wooden tree of thickness 2 and weight $A$, and the third with a metal tree of arbitrary non-zero thickness with weight $B$. If the goal is $d = 0$, and the plan is the one from Figure 1, then (10) holds for (say) $\kappa = .3$ and (10) holds for (say) $\kappa = .7$.

We do not think that there is one preferred definition for correctness. While Definition 4.2 attempts correctness in the sense of [2], planning for likely states, as in (10), is very common in robotics when exploring large biased search spaces [22, 23].

5 INCORPORATING NOISY SENSORS

Accounts like Definition 4.2 and (9) capture nondeterministic acting when the outcome of the action is immediately observable. In applications such as robotics, sensing is often noisy. Similar to contingent planning [7, 24, 25], we will now motivate a semantics of plan execution over belief states.

To review the setting informally, consider a noise-free tree chop problem instantiated by (1). Although the agent believes that $d \in \{1, \ldots, 10\}$, in the real world $S_0$ the tree has a fixed thickness, say 2. In this case, the controller discussed previously will advise two chop actions, which will be executed in each of the possible worlds, including $S_0$. At this point, a noise-free sensor returns down, and so, the agent will come to believe that the tree is down. Implicitly, all the worlds other than $S_0$ considered possible initially will be discarded, as they are no longer compatible with the sensing results. (For example, the world in which $d = 1$ will be discarded after the
sensor says that the tree is still standing on doing a chop action, because that is clearly not possible in such a world.) However, a noisy sensor might return a down despite the tree still standing. The point, then, is that the sensor’s error profile will inform the agent how likely it is that a down is observed when the tree still stands, and based on that, subsequent actions can be taken until it is believed that the tree is no longer standing.

To formalise this intuition, we will need to address two technical issues. First, observe that the account of Bel from BHL [11] makes no mention of sensing functions, and in this sense, the language is only geared for projection. That is, we can infer the value of Bel(\(d \leq 4\), do(getd(3), S0)) where we explicitly provide the sensor reading, but it is ill-formed in the language to reason about beliefs after getd sans argument that is determined only at run time. Moreover, as discussed above, the external feedback (e.g. number reported on a sensor) will only be an estimate of the true property when we turn to noisy sensors.

The solution to this issue by BHL was to define programs for sensing actions: for example, a sensor was defined as \(\Pi y. \text{getd}(y)\), the latter being a GOLOG program [18] that non-deterministically chooses the argument for the sensing action. In [26], a slightly simpler technical device was introduced where getd also stood for a program, but the semantics of program execution, which is defined over action sequences, incorporates run-time sensor readings. Neither of these solutions is appropriate for us, because: (a) we would like to avoid the complexity of defining GOLOG programs; and (b) we would like the new definition to be compatible with our previous accounts of plan execution, enabled via the SF function. We achieve this by considering a runtime sensing outcome function \(\Pi : A^* \rightarrow O\). Recall that in the situation calculus, situations are not states, and the initial situation corresponds to the setting where the agent has not executed any action. In a way, \(\Pi\) is an analogue to usual definitions of observation functions that map states to observations in that it responds to an action history. Then, we use SF to refer to runtime readings as follows:

\[
SF(a, do(a_1, do(\ldots, do(a_t, \ldots))) = \Pi(a_1, \ldots, a_t, a).
\]

With this machinery, we can use SF as usual in the plan execution semantics.\(^6\) We can then introduce parameterless sensing actions like getd whose likelihood is now defined to mimic (9) as follows:

\[
I(\text{getd}, s) = \mathcal{N}(SF(\text{getd}, s); d(s), 1).
\]

The second technical issue is to interpret execution paths over belief states, but while referencing the real world to test for sensing outcomes. That is, starting from a control state (from the plan structure) and the agent’s beliefs, we will need to define how a new control state is reached with an updated set of beliefs. So, we will need to “reify” beliefs in formulas: for any ground situation term \(s\), we introduce a new term \(\exists s.\) to be used with formulas in that we write \(\phi(\exists)\) to mean \(\forall s.\.\). We will often use two such terms in a predicate for one-step transitions: \(V(t, s, \exists t', s)\) can be first expanded to \(\exists s'. K(s', s) \supset V(t, s', \exists t', s')\), which then expands to \(\forall s', s'. [K(s', s) \land K(s', s')] \supset V(t, s', \exists t', s')\). Recall that the \(K\) fluent was assumed to be an equivalence relation, and so, roughly speaking, \(V(t, \exists t', \ldots)\) can be seen as saying that starting from \(t\) and the belief state given by the situation \(s\) (that is, all \(K\)-related situations from \(s\)), we perform a transition to \(t'\) and the belief state given by do(a, s).

Formally, we define:

\[
\text{Definition 5.1.} \text{ We use } V^*(q, \exists q', s) \text{ as abbreviation for } \forall V[q, \exists q', s].
\]

\[
\text{Example 5.3.} \text{ Let } D \text{ be an action theory built from (3), (6), and the likelihood axiom (9). Suppose the sensor works as follows:}
\]

\[
I(\text{getd}, s) = \begin{cases} 
1 & \text{SF(\text{getd}, s) = down \land d = 0} \\
1 & \text{SF(\text{getd}, s) = up \land d > 0} \\
0 & \text{otherwise}
\end{cases}
\]

It is noise-free. Finally, suppose our goal is Bel(\(d < 10, \text{now} > .9\)).

A plan that is epistemically correct for this goal is given in Figure 2. We only argue for the initial world \(S_0\). It also depicts a possible execution path of the plan using \(V^*\). Let us suppose \(d(S_0) = 1\).

The first action advised by the controller is chop, and suppose the

\(\Pi\) would also take into account the values of fluents at situations, which we omit here for simplicity.

\(\Pi\).

\[\Pi(\ldots, do(a_t, \ldots)) = \Pi(a_t, \ldots, a_t, a).
\]

\[I(\text{getd}, s) = \mathcal{N}(SF(\text{getd}, s); d(s), 1).
\]

\[I(\text{getd}, s) = \begin{cases} 
1 & \text{SF(\text{getd}, s) = down \land d = 0} \\
1 & \text{SF(\text{getd}, s) = up \land d > 0} \\
0 & \text{otherwise}
\end{cases}
\]

\(^6\)This is a key point as far as practical planning frameworks are concerned: \(V^*\) does not look so different from the semantics of noise-free belief-based planning — see the account in [21], for example; however, what is then needed is a set of belief states that correctly accounts for the unobservability of nondeterministic outcomes and how that changes with noisy sensing.
action instantiates as \( \text{chop}(1, 0) \). Then \( d(\text{do}(\text{chop}(1, 0), S_0)) = 1 \); so its still standing, and also, the agent accords a belief of .9 to \([0, 9]\) that corresponds to a successful move, and a belief of .1 to its failure. The noise-free sensor naturally returns \( \text{up} \). The controller advises \( \text{chop} \) again, and suppose this time, the action instantiates as \( \text{chop}(1, 1) \).

Incidentally, even without a sensor reading, the degree of belief in \( d < 10 \) is .9 because the only branch that still entertains \( d \) retaining a value of 10 is that of both chop actions failing, with a likelihood of .1 \( \times .1 \). In any case, the controller advises a sensing action, which would return \( \text{down} \), and so the plan terminates for \( a = [\text{chop}(1,0) \cdot \text{getd} \cdot \text{chop}(1,1) \cdot \text{getd}] \) with KNow.(Bel(d < 10, now) > .9, do(α, S₀)).

Example 5.4. Consider the tree chop problem with noisy sensors and effectors. Suppose the initial theory and alt-axioms are (3) and (6) as before, but the likelihoods for the effector is given by (7) and that for the sensor is given by (4). Finally, suppose our goal is Bel(d ≤ 5, now) > .8.

A plan that is epistemically correct for the goal is given in Figure 3 wrt observed values of 5.5, 4.5 and 3.9. We only argue for S₀. Assume also that \( \{< 6, > 6\} \in O \) and that the numeric values obtained from the sensor map to these binary outcomes.

Here, the controller advises \( \text{chop} \), after which the belief in \( ψ = d \leq 5 \text{ is } .5 \). This is because the likelihood of the action succeeding is more than it failing, and given the prior in \( d \leq 5 \text{ is } .5 \), the posterior should be clearly greater than .5. Suppose now the sensed value is 5.5. The belief in \( ψ \) drops to < .5. The controller advises another \( \text{chop} \), at which point the belief in \( ψ \) increases to > .5. Next, we observe a reading of 4.5 followed by 3.9. The controller terminates. On termination, it can be verified that the robot knows that the degree of belief in \( ψ \) is > .8.

In a noise-free setting, our definition of an epistemically correct plan is downward compatible with Definition 2.5 (and thus, Definition 4.2):

**Theorem 5.5.** Suppose \( D \) is a noise-free action theory, \( X, Σ \) as above, and \( ϕ \) is any formula not mentioning \( K \). If \( X \) is epistemically correct for \( ϕ \), then it is correct for \( ϕ \) in the sense of Definition 2.5.

**Proof.** Suppose \( X \) is epistemically correct but not correct. Then there is some \( s \) such that \( K(s, S₀) \) and \( \neg G \) of \( T^∗(Q₀, s, Q_F, s′) \land ϕ(s′) \). By assumption, \( V^∗(Q₀, 3, Q_F, s′) \land ϕ(s′) \) for some \( s′ \). Thinking of \( (c, s, s′) \) as “nodes” in an execution path, the definition of \( V^∗ \) is the least set of pairs of nodes containing \( (Q₀, t) \) for all situation terms \( t \) such that \( K(t, s) \), which includes \( s \) because \( K \) is assumed to be an equivalence relation; \( (q, do(α, t)) \) for all situation terms \( t \) such that \( K(t, s) \) provided \( Q₀ \) advices \( α \), it is executable at every \( t \) and the sensing function returns \( o \) for \( a₁ \).

\[ \text{at } s \text{ and } δ(Q₀, o) = q; \text{ and so on. By assumption, } V^∗ \text{ contains as node } (Q_F, s′) \text{ for } s′ = do(α₁ \cdot αₖ, s). \text{ But, by the definition of } T^∗, \text{ it follows that } T^∗(Q₀, s, Q_F, s′). \text{ Moreover, since } ϕ(\overline{s′}) \text{ and } K \text{ is reflexive, } ϕ(s′). \text{ Contradiction.} \]

In general, in the presence of noise, as one would expect, epistemic correctness diverges from correctness criteria based on plan evaluation at initial worlds. For example, we have:

**Theorem 5.6.** Suppose \( D \) is any action theory, \( X, Σ \) as above, and \( ϕ \) is any formula not mentioning \( K \). If \( X \) is epistemically correct for \( ϕ \), then it does not follow that \( X \) satisfies (8).

**Proof.** To prove this result, it suffices to provide a (possibly unwise) controller that is epistemically correct, but does not satisfy (8). Consider the tree chop problem for a tree of unit thickness, but with an additional action for picking up a saw. Suppose there is only a single initial world \( S₀ \), and \( \forall a, s (\text{Poss}(a, s) \equiv true) \). Suppose the pick-up can fail, that is, suppose alt(pickup, b, z) \equiv b = noop, where noop is the action of doing nothing. Suppose neither of these actions provide the agent with any meaningful sensing result: that is, \( \forall s \text{ SF}(\text{pickup}, s) = \text{SF}(\text{noop}, s) = \text{nil} \). Imagine a controller like in Figure 4, where the control states are designated by numbers. (That is, \( q₁ \) is the control state advising pickup and \( q₄ \) is terminating state.) Initially, the controller advises pickup, leading to two successor situations: \( do(\text{pickup}, S₀) \) and \( do(\text{noop}, S₀) \). By way of the definition of \( SF \) and Poss for these actions, we have \( V^∗(q₁, S₀, q₂, do(\text{pickup}, S₀)) \) and \( U^∗(q₁, S₀, q₂, do(\text{noop}, S₀)) \). Suppose now \( SF(\text{chop}, do(\text{pickup}, S₀)) = \text{down} \) and so we have \( V^∗(q₁, S₀, q₄, do(\text{pickup} \cdot \text{chop}, S₀)) \), by construction, \( [d(\text{now}) = 0]((do(\text{pickup} \cdot \text{chop}, S₀)) \). However, suppose \( SF(\text{chop}, do(\text{noop}, S₀)) = \text{up} \). Then we have \( U^∗(q₁, S₀, q₃, do(\text{noop} \cdot \text{chop}, S₀)) \), which will not terminate.

The intuitive reason is that the termination conditions defined over \( U^∗ \) respond to sensing results along every execution path, whereas \( V^∗ \) only responds to the sensing outcomes for the path of advised actions from an initial world and how belief changes with it. This can be seen to be not surprising: among other things, the much stronger (8) is not needed because incompatible worlds will get discarded after sensing.

Let us conclude the section with two remarks. First, as mentioned earlier, the definition of \( V^∗ \) does not look very different from the semantics of noise-free belief-based planning, and this is good news: if the space of belief states is designed carefully to account for noise, algorithms for noise-free generalised planning may carry over to the stochastic case. Second, note that \( V^∗ \) was defined to include...
an explicit reference to sensing outcomes from the environment, which is external to the agent. A result in [21] shows that it is not possible to realise that we are making progress towards the goal without such a construction in belief-based planning.

6 DISCUSSION AND CONCLUSIONS

Generalising plans has been of interest since the early days of planning [27]. Algorithmic proposals to synthesise plans that generalise varied widely in methodology, ranging from interactive theorem proving [28] to learning from examples [29]. The convergence of these approaches to synthesise plans that solve multiple problem instances is a recent effort [4, 19, 30–32]. We refer interested readers to [31] for a comprehensive list of references, and [5, 33] for recent advances on handling nondeterminism. Outside of Levesque’s account on the correctness of program-like plans as an epistemic formulation, there are numerous variants [2, 5, 33–35]. The semantics $U^*$ extended Levesque’s $T^*$ to handle noisy acting, and $V^*$ further extends that to noisy sensing, thereby obtaining a full generalisation of the formulation to handle nondeterminism.

Belief-based planning, which we touch upon, is widely studied, e.g., [24], and the usual approach is to formulate a nondeterministic (conformant) planning problem that treats belief states as first-class citizens. Our definition of $V^*$ can be seen as a formalisation of this semantics against a logic of probabilistic belief and action. In that regard, the motivation behind this work is close in spirit to knowledge-based programs [36, 37] and its stochastic extension [26]. These are formulated using the situation calculus and the high-level programming language GOLOG, but, of course, variant languages are also popular for developing such planning accounts [38]. At the outset, there are significant reasons to develop a semantics customised to memoryless plans, as we argue below. Moreover, since there are a number of generalised planning algorithms that synthesise loopy plans [4], an execution semantics tailored to that representation is useful to understand how those algorithms can be applied to domains with noise.

Let us begin by observing that memoryless plans can be easily encoded as GOLOG programs consisting of atomic physical actions and branches based on sensing outcomes. In general, given a program $\delta$, one is interested in showing that

$$\mathcal{D} \cup \Theta \models Do(\delta, S_0, do(\sigma, S_0))$$

where $\Theta$ encodes the single-step transition semantics of $\delta$, and $\sigma$ is a ground sequence of actions such that $\delta$ terminates in $do(\sigma, S_0)$. The key feature of knowledge-based programs is that $\delta$ can mention $\text{Know}$, and the probabilistic belief operator $\text{Bel}$ in [26]. Nonetheless, note that if $\delta$ does not mention $\text{Bel}$, the entailment criteria above is weaker than Definition 2.5 as it only looks at $S_0$. But if it mentions the $\text{Bel}$ operator, then it seems closer to Definition 5.2, but at the cost of a more cumbersome plan structure: $\delta$ can have unbounded memory (via while loops), can refer to complicated state properties, and is subjective, whereas Definition 5.2 is defined for memoryless plans built purely from a finite set of atomic actions.

So, in the current paper, the end result is an account of correctness with widely-studied loopy plan structures, which eschews the complications of GOLOG but is able to achieve almost as much. Naturally, then, relating knowledge-based programs and loopy plans (in belief-based settings) is likely to be of considerable theoretical interest, as would a closer study of the two execution semantics. (Cf. also [34] on memoryless structures being effective, and [37] on knowing how to execute GOLOG programs.)

Despite focusing on probabilities and nondeterminism, this paper has established no connection to the large body of work on Markov decision processes [39]. Mostly, decision-theoretic planning frameworks are characterised in terms of optimality criteria against expected rewards, often enabled via dynamic programming, while we have treated goals as arbitrary formulas that are to be satisfied, as would symbolic planning frameworks such as [2]. Nonetheless, one can imagine ways of recasting expected rewards in terms of goal satisfaction or vice versa [40], and that is arguably worth doing in the context of this paper so as to relate to efforts such as [41]. Interestingly, recent robotics planners such as [16] eschews a planning paradigm that advises actions for every belief state, as one would in partially observable Markov decision processes, and instead resorts to a scheme that computes plans for a designated initial belief state, as in belief-based planning. Moreover, as mentioned before, ultimately the goal here was to generalise Levesque’s account and to provide a rigorous foundation for extensions such as the handling of non-unique prior distributions.

As a final remark, like in Levesque’s original formulation, one can motivate a generic planning procedure as follows:

\[
\text{input: } \phi, E^* \in \{T^*, U^*, V^*\}, \Delta = \text{a correctness criteria} \\
\text{repeat with } X \in \text{FINITE STATE CONTROLLERS} \\
\text{if } \mathcal{D} \cup \Sigma \models Vs, K(s, S_0) \supset \Delta(E^*, \phi, s) \text{ then return } X
\]

Naturally, we do not expect to use a full-blown logical framework for planning, nor do we expect planners to actually use such a procedure in practise. Languages like the ones in [16, 42] seem entirely reasonable. It is also conceivable that existing algorithms for generalised planning, such as bounded AND/OR searches, can be adapted for stochastic settings, as argued earlier, possibly by leveraging abstraction techniques [5, 33]. We hope this paper is also useful for approaching and resolving that line of inquiry.

REFERENCES