Abstract. We present a representation for reasoning and planning with an incomplete state description (open-world). The presented formalism has several properties critical for application domains with a large degree of incompleteness in the state description, particularly, in domains with a large or unknown set of all objects. The formalism offers (1) considerably expressive state and goal description language, that includes limited universal quantification, (2) representation of sensing actions and knowledge goals, (3) a correct and complete state update procedure, and (4) complete reasoning within a substantial subset of the language. The approach is illustrated by examples from a working system.

1 Introduction

Planning with correct but incomplete information and with or without sensing actions (a.k.a. information gathering) has been addressed by numerous researchers, offering several approaches for representing an agent’s knowledge and for reasoning with an incomplete state description (e.g., [17, 13, 8, 14, 5, 10, 7, 11, 18]).

One major challenge is the computational complexity of reasoning with incomplete information. Our work occupies a unique niche by offering: (1) a logic of knowledge that has sufficient expressive power for a large array of applications, (2) polynomial time algorithms to determine entailment and other key relationships, and (3) completeness of reasoning for a large subset of the language – and we conjecture that it is complete for the entire language. We report on the logical foundations of this representation and give examples of its important properties.

We assume that the agent’s knowledge is correct but incomplete and that it has a set of primitive actions available. Domain actions change the world, have deterministic effects and do not have conditional effects. Sensing actions have non-deterministic effects, i.e., there are several possible outcomes. Moreover, actions can have preconditions.

Our representation can be used with a variety of planning languages and planning algorithms. To date it has been used for a conformant (i.e., with open worlds but without sensing) partial order planner [2], a conformant Graphplan-based planner [6], and a planner that interleaves planning and execution [1].
At the core of any planning language and algorithm is the need to project or regress an agent’s state of knowledge over a sequence of primitive actions. For this paper, we focus only on this core and on progression, and therefore define a planning problem to be a 3-tuple \( < I, G, A > \) where \( I \) is an initial knowledge base of the agent, \( G \) is a goal to achieve, described as a partial knowledge state, and \( A \) is a set of primitive actions. A plan \( P \) is an executable sequence of ground instances of elements of \( A \). To be executable means that it can be proven that the agent will know each action’s preconditions are true just before executing the action. Finally, a plan solves a planning problem if and only if it can be proven that if \( I \) is true before the first action then \( G \) is true after the last.

Parts of PSIPLAN-S have been presented in [2] and [1], both of which addressed the use of PSIPLAN-S representation in planning algorithms. In this paper we focus on the logical foundations of reasoning and planning with sensing actions and knowledge goals and prove properties underlying its soundness and completeness that have not been published before.

1.1 Overview of PSIPLAN-S

To illustrate the representational needs of an open-world planner that are addressed in PSIPLAN-S we present the following scenario from a real-world application called Writer’s Aid [1]. An agent must locate and possibly download papers that contain certain keywords in their title. The agent knows nothing about any actual papers, their locations or contents. It knows about a few bibliographic web-sites and it has actions for querying these sites and downloading papers from a given location.

In order to complete its task the agent must (1) find all relevant papers from these bibliographies, (2) determine if these bibliographic collections have viewable versions of the papers, and (3) if a paper is located, download it; otherwise, try another bibliographic source.

First, it must be possible to distinguish between those bibliographies known to be preferred by the user, known not to be preferred, and those about which the agent does not know the user’s preference. The closed world assumption cannot be used here, so negative facts must be included in the state description. When the number of domain individuals is large or unknown, as it is in Writer’s Aid due to the number of potential bibliographies and citable papers, it is impossible to represent the known negative facts without some sort of universal quantification. Moreover, when the domain is either infinite or not entirely known to the agent, reasoning with quantified propositions cannot be done by expanding each to the of set of ground instances it represents.

PSIPLAN-S uses a combination of ground atoms and \( \psi \)-forms ([2, 1]) to represent such scenarios. For example, to represent that

The only preferred bibliographies are the digital library of the ACM
and maybe the ResearchIndex database

PSIPLAN-S uses the following two propositions
1. a \( Kn \)-atom, expressing \( ACM \)’s digital library is a preferred bibliography

\[ PN(Bib(ACM)) \] (1)

2. a \( \psi \)-form \( \psi_1 \), expressing Nothing is a preferred bibliography except for possibly \( ACM \) and the ResearchIndex

\[ \psi_1 = \{ Kn(\neg PN(Bib(b))) \text{ except } \{ b = ACM \}, \{ b = RI \} \} \] (2)

which denotes all ground instances of the formula \( KN(\neg PN(Bib(b))) \) minus two exceptions: \( KN(\neg PN(Bib(ACM))) \) and \( KN(\neg PN(Bib(RI))) \) and could also be expressed as a universally quantified formula

\[ \forall b. KN(\neg PN(Bib(b))) \lor (b = ACM) \lor (b = RI) \]

\( \psi_1 \) alone does not commit to the truth or falsity of its exception clauses, i.e. \( KN(\neg PN(Bib(ACM))) \) and \( KN(\neg PN(Bib(RI))) \), but from (1) and (2) we can conclude that the agent knows that ACM is a preferred bibliography, and nothing else is a preferred bibliography except for possibly the ResearchIndex.

Note that while PSIPLAN-S can represent infinite “negative” knowledge via \( \psi \)-forms, it can represent only finite “positive knowledge”, in other words a finite number of propositions of the form \( KN(a) \). This design reflects the assumption typically made in all closed world problems, that in a planning problem there is a finite number of relevant things that are true, and maybe an infinite number of things that are false. For instance, there is only a finite number of items in any briefcase, while there is an infinite number of objects that are not in it.

Second, it must be possible to represent knowledge goals, like find the relevant papers recommended by the digital library of the ACM, and identify that such goals can be achieved by using a sensing action, which queries a bibliography for a list of papers related to a keyword. PSIPLAN-S uses \( KW \)-propositions to represent knowing whether a statement is true or false. For example, the goal above, as well as the effect of the sensing action is expressed with the following \( KW \)-\( \psi \)-form

\[ \tilde{\psi}_2 = [KW(\neg InCollection(y, ACM)) \lor \neg Rel(y, Kwd)] \], (3)

which states that for every possible paper \( y \), the agent knows whether this paper is in the collection of ACM and is also recommended as related to keyword \( Kwd \). Note that while \( KW(p) \) is equivalent to \( KW(\neg p) \), the disjunction of negated literals inside of the \( KW() \) statement is used instead of a conjunction of atoms to simplify the formulation of the reasoning algorithms.

Third, it must be possible to download all relevant papers. Thus the agent must reason about domain goals such as having a paper stored on a hard drive. PSIPLAN-S’s domain goals include atoms and universally quantified propositions represented with \( \psi \)-forms.
Furthermore, for reasons of efficiency, the agent should avoid redundant actions. There is no need to download a paper that is known to be stored on the local hard drive, and no need to sense for information already implied by the agent’s knowledge base. Thus the agent needs to know what it does and does not know. PSIPLAN-S provides complete reasoning from the agent’s state of knowledge about domain and knowledge goals. The reasoning algorithms work directly with quantified propositions and never convert them to sets of ground instances.

Another aspect contributing to non-redundancy of information gathering is PSIPLAN-S’s ability to update its knowledge state upon executing an action, so that the resulting knowledge state correctly and completely describes the set of possible worlds resulting from taking an action. This is done without actually considering all possible worlds individually.

Although we do not report on it in this paper, the reasoning from an agent’s state of knowledge and the state update procedure have polynomial complexity in the number and size of propositions in the agent’s state of knowledge [3].

2 Representing an Agent’s Knowledge

Before defining the language of PSIPLAN-S formally, we define two languages that do not have quantification, $\mathcal{L}_D$ and $\mathcal{L}_K$. $\mathcal{L}_D$ is a ground domain language, used to define the state of the world. $\mathcal{L}_K$ is a ground knowledge language, expressing the agent’s knowledge of the world state in the form of ground knowledge propositions. PSIPLAN-S is an open world planning language, which includes

1. a compact representation of propositions of $\mathcal{L}_K$ used to denote agent’s knowledge of the world state and goals;
2. an action definition language.

$\mathcal{L}_D$ and $\mathcal{L}_K$ are not directly used in planning, because our goal is to provide a planning representation which could operate in domains where the number of objects is infinite or at least very large, and can never be fully acquired. Thus, it is impossible to enumerate all ground facts using $\mathcal{L}_D$ or $\mathcal{L}_K$.

However, we introduce $\mathcal{L}_D$ and $\mathcal{L}_K$ for the purpose of defining PSIPLAN-S’s semantics and proving some of its important logical properties.

2.1 The Propositional Domain Language $\mathcal{L}_D$

The domain language, denoted $\mathcal{L}_D$, consists of ground atoms (also called domain atoms) and clauses of the form $\neg a_1 \lor \neg a_2 \lor \ldots \lor \neg a_n$, where each $a_i$ is a ground atom. A domain proposition is either a domain atom or domain clause.

Domain propositions are interpreted over a set of worlds, which are standard first-order interpretations.¹ For a world $w$ and a domain proposition $p$, $\text{wt}(p)$ denotes that $p$ is true in the world $w$ according to the standard interpretation rules of propositional formulas.

¹ While we need only propositional interpretations here, we later need first-order interpretations when we introduce quantification via $\psi$-forms.
Assuming standard propositional semantics, it is easy to see that deduction in $L_D$ is complete under unit clause resolution and subsumption:

\[
\begin{align*}
(1) & \quad a; \neg a \vee \neg a_1 \vee \ldots \vee \neg a_k \quad \text{(unit clause resolution)} \\
(2) & \quad \neg a \vee \neg a_1 \vee \ldots \vee \neg a_k \quad \text{(subsumption)}
\end{align*}
\]

2.2 The Propositional SOK Language $L_K$

We define $L_K$, a language for representing the state of agent’s knowledge of the world and of its own knowledge. The sentences of $L_K$ are called knowledge propositions that are either Kn-literals or KW-literals as defined below. For any domain proposition $p \in L_D$,

1. a Kn-literal is either a Kn-atom, $Kn(p)$, or its negation, $\neg Kn(p)$. Kn-atoms represent the agent’s knowledge of the world, e.g. $Kn(On(A, B))$ represents that the agent knows that block $A$ is on block $B$.
2. a KW-literal is either a KW-atom $KW(p)$ or its negation $\neg KW(p)$.

$KW(p)$ denotes that the agent knows the truth value of $p$, and is thus semantically equivalent to $Kn(p) \vee Kn(\neg p)$.

As mentioned earlier, KW-atoms are used to represent knowledge goals, effects of sensing actions, and projections of the agent’s state of knowledge over sensing actions.

2.3 Semantics and Entailment

We use $k$-states of Baral and Son [5] to define the semantics of the knowledge language $L_K$. Let $W$ denote the set of all worlds, which are first-order interpretations over a given domain of discourse. A k-state is a pair $(w, W)$, where $W$ is a subset of $W$ and $w$ is an element of $W$. A k-state $(w, W)$ represents the knowledge state of an agent who actually being in the world $w$ thinks it might be in any of the worlds of $W$. We are assuming that the agent’s knowledge is correct, hence we require that for any k-state $(w, W)$, $w \in W$.

The set of all possible k-states is denoted $K$.

The set of models of a knowledge proposition or a set of knowledge propositions from $L_K$ is denoted by $\alpha()$ and defined below. We note that $\neg(\neg a_1 \vee \ldots \vee \neg a_k)$ is equivalent to $\{a_1, \ldots, a_k\}$.

1. $\alpha(Kn(p)) = \{(w, W) \mid w \in W \land \forall w' \in W : w'(p)\}$
2. $\alpha(\neg Kn(p)) = \{(w, W) \mid w \in W \land \exists w' \in W : w'(-p)\}$
3. $\alpha(KW(p)) = \{(w, W) \mid w \in W \land [\forall w' \in W : w'(p)] \lor [\exists w' \in W : w'(-p)]\}$
4. $\alpha(\neg KW(p)) = \{(w, W) \mid w \in W \land [\exists w' \in W : w'(p)] \land [\exists w'' \in W : w''(-p)]\}$
5. $\alpha(\{k_1, \ldots, k_m\}) = \cap_{i=1}^{m} \alpha(k_i)$, where each $k_i$ is from $L_K$. 

5
Note that according to the definition above, for each \( k \in L_K \), we have 
\[ \alpha(\neg k) = \mathcal{K} - \alpha(k). \]

Given the definition of a model of a knowledge proposition, we define *entailment in \( L_K \) \( (\models_k) \) in the usual way. For two knowledge propositions or sets of knowledge propositions \( q \) and \( r \) we write \( q \models_k r \) if and only if every model of \( q \) is a model of \( r \), i.e.
\[ q \models_k r \text{ if and only if } \alpha(q) \subseteq \alpha(r). \] (4)

The following Proposition states an important property of \( L_K \) that allows the reduction of the entailment between \( Kn \)-propositions to the usual propositional entailment. It also proves that in our logic an agent knows all the propositional consequences of its knowledge.

**Proposition 1.** Let \( p_1, \ldots, p_k, q \) be domain propositions:
\[ (p_1, \ldots, p_k \models q) \iff (Kn(p_1) \ldots Kn(p_k) \models_k Kn(q)). \]

Some special cases of entailment between propositions of our logic are considered in the two propositions below. These properties are later used in establishing the soundness and completeness properties of PSIPLAN-S. Proposition 2 considers entailment between a set of \( Kn \)-atoms and a single \( KW \)-atom. Proposition 3 considers entailment between \( KW \)-atoms.

**Proposition 2.** Let \( c \) be a domain clause.

1. If \( a \) is a domain atom, \( Kn(a) \models_k KW(c) \) if and only if \( c = \neg a \).
2. If \( c' \) is a domain clause \( Kn(c') \models_k KW(c) \) if and only if \( c' \subseteq c \).
3. If \( a_1, \ldots, a_k \) are domain atoms, and \( c_1, \ldots, c_n \) are domain clauses, and the set \( \{a_1, \ldots, a_k, c_1, \ldots, c_n\} \) is consistent,
\[ s = \{Kn(a_1), \ldots, Kn(a_k), Kn(c_1), \ldots, Kn(c_n)\} \models_k KW(c) \]
if and only if
(a) there exists a domain atom \( a \), such that \( Kn(a) \in s \) and \( c = \neg a \), or
(b) there exists a domain clause \( c' \), such that \( s \models_k Kn(c') \) and \( c' \subseteq c \).

**Proposition 3.** \( KW(c_1), \ldots, KW(c_n) \models_k KW(c) \) if and only if there exist a set of indices \( i_1, \ldots, i_k \) where \( 1 \leq i_j \leq n \) for each \( i_j \), such that \( c = c_{i_1} \lor \ldots \lor c_{i_k} \).

In other worlds, for a set of \( KW \)-atoms to entail another \( KW \)-atom \( KW(c) \), there must be a subset whose domain clauses produce the domain clause \( c \) when combined via the union operator. Note that when the left side consists of just a single \( KW \)-atom \( KW(c') \), we have \( KW(c') \models_k KW(c) \) if and only if \( c' = c \).
2.4 Inference Rules

The soundness of inference rules depicted in Figure 1 can be easily verified using the definition of entailment and Propositions 1-3. We conjecture, but have not shown, that this set of rules is complete for $L_K$.

However, the two propositions presented later in this section establish that the rule system is complete with respect to reasoning about $Kn$ and $KW$-atoms from a set of $Kn$-atoms. We focus on this kind of reasoning since, as we define in the next section, PSIPLAN-S only uses $Kn$ and $KW$-atoms for the representations and reasoning involved in planning. We have presented all 13 rules here to provide a more complete description of the propositional language $L_K$, which can be used in formalisms that do not make some of the PSIPLAN-S’s assumptions and thus may need to rely on another subset of these rules.

**Proposition 4.** $Kn(p_1), \ldots, Kn(p_k) \models_k Kn(q)$ if and only if it can be derived using rules (R1) and (R2).

**Proposition 5.** $Kn(p_1), \ldots, Kn(p_k) \models_k KW(q)$ if and only if it can be derived using rules (R1), (R2), (R10) and (R11).

The significance of Propositions 4, 5 and the Conjecture below is that the number of rules used by a PSIPLAN-S reasoning is limited to only 6 out of 13, while the completeness of such reasoning, at least in case of reasoning without sensing actions, is retained. Indeed, the agent’s State Of Knowledge (SOK) in PSIPLAN-S is represented by a set of $Kn$-propositions, and Propositions 4, 5 establish completeness of reasoning about $Kn$ and $KW$-goals from SOK. This is possible because of a number of assumptions made in PSIPLAN-S, for instance, the requirement on the correctness of agent’s knowledge and complete

![Fig. 1. Inference rules for $L_K$](image-url)
specification of action effects allow PSIPLAN-S to operate without the negated \( Kn \)-propositions.

Conjecture 1 \[ Kn(p_1), \ldots , Kn(p_k), KW(q_1), \ldots , KW(q_l) \models_k Kn(r) \text{ if and only if it can be derived using rules (R1) and (R2).} \]

\[ Kn(p_1), \ldots , Kn(p_k), KW(q_1), \ldots , KW(q_l) \models_k KW(r) \text{ if and only if it can be derived using rules (R1), (R2), (R10), (R11), (R12) and (R13).} \]

2.5 The Agent’s State Of Knowledge (SOK)

In this section we introduce other definitions and properties required in the context of planning.

An agent’s state of knowledge, or SOK, is represented by a set of \( Kn \) atoms.

The set of possible worlds of an agent with SOK \( s \) is defined as the set of all worlds in all models of \( s \). Let \( B(s) \) denote all the worlds that are possible given \( s \).

\[ B(s) \overset{\text{def}}{=} \{ w \mid \exists (w, W) \in \alpha(s) \} \]

Proposition 6. \( B(s) = \{ w \mid \forall p . (s \models_k Kn(p)) \Rightarrow w(p) \} \), or in other words, the set of possible worlds is equivalent to the set of all worlds in which everything known to the agent is true.

We make the Closed Knows Whether Assumption, which means that for a given SOK \( s \) and for all domain propositions \( p \), if \( s \) does not entail \( KW(p) \), i.e., \( s \not\models_k KW(p) \), then the agent assumes that \( \neg KW(p) \). We define \( CKWA(s) \) for a SOK \( s \) with

\[ CKWA(s) = s \cup \{ \neg KW(p) \mid p \in \mathcal{L}_D \wedge s \not\models_k KW(p) \} \]

The set of possible worlds of \( s \) and of its closure \( CKWA(s) \) are the same, as shown next. This allows to eliminate the direct use of the negated \( KW \)-literals in PSIPLAN-S relying on the negation as failure instead: if \( KW(p) \) cannot be established via application of rules identified in Conjecture 1, we assume ignorance about \( p \), i.e. \( \neg KW(p) \).

Proposition 7. \( B(s) = B(CKWA(s)) \)

3 \( \psi \)-forms - A Compact Representation

We now introduce propositions called \( \psi \)-forms that compactly represent sets of \( L_K \) propositions. We define \( \psi \)-forms before introducing PSIPLAN-S fully in Section 4.

We assume an infinite number of domain constants, which denote distinct domain individuals, and no other function symbols. We note that in \( \psi \)-forms the \( KW \) operator is used only with a negated clause. Using \( KW \)-propositions with a conjunction of atoms inside would result in more cumbersome formulations of the theorems, while giving us no additional expressive power, since \( KW(\neg p) \), is equivalent to \( KW(p) \).

A PSIPLAN-S proposition is either a \( Kn \)-atom, or a \( \psi \)-form (as defined below) that represents a set of \( Kn \) or \( KW \)-atoms.
3.1 $\psi$-forms

The general form of a $\psi$-form is $[Q(x) \text{ except } \{\sigma_1, \ldots, \sigma_n]\]$. The formula $Q(x)$ is called the main form and has the form $Kn(p(x))$ or $KW(p(x))$ where $p(x)$ is a disjunction of negated literals. The main form of $\psi$ is denoted $M(\psi)$. Each of $\sigma_1, \ldots, \sigma_n$ defines $\psi$’s exceptions, and each exception is a substitution on a subset of variables in $x$.

A $\psi$-form may have an empty set of exceptions. In that case it has a form $[Q(x)]$ and is called simple.

$\psi$-forms represent the set of all ground instantiations of the main form, except for those that can be obtained by instantiating the exceptions, as illustrated by (2) earlier.

Formally, we define the set of ground propositions of $L_K$ represented by a $\psi$-form as follows

1. $\phi([Q(x)]) = \{Q(x)\sigma \mid Q(x)\sigma \text{ is ground}\}$
2. $\phi([Q(x) \text{ except } \{\sigma_1, \ldots, \sigma_n\}]) = \phi([Q(x)]) - \phi([Q(x)\sigma_1]) - \ldots - \phi([Q(x)\sigma_n])$

A $\psi$-form with a main form that does not contain any variables and thus represents a single ground negated clause is called a singleton. When the main form contains at least one variable, we call it non-ground, or, sometimes, quantified.

4 PSIPLAN-S

PSIPLAN-S assumes infinite number of domain objects. A PSIPLAN-S proposition is either

1. a $Kn$-atom of the form $Kn(a)$, where $a$ is a ground atom, or
2. a $Kn$-\psi-form or $KW$-\psi-form.

PSIPLAN-S States Of Knowledge. PSIPLAN-S’s State Of Knowledge is a conjunction of PSIPLAN-S $Kn$-propositions that represent agent’s knowledge about the world.

PSIPLAN-S Goals. A PSIPLAN-S goal is any PSIPLAN-S proposition.

$KW$-\psi-forms in PSIPLAN-S are used to represent information goals, results of sensing actions and reason about knowledge and ignorance. Consider $\psi = [KW(\neg PrefBib(x))]$, for example. Posted as a goal, $\psi$ requires knowing the value of each ground instance of $PrefBib(x)$, or in other words, knowing the set of preferred bibliographies. On the other hand, $\psi$ describes the effect of a sensing action that identifies the set of all preferred bibliographies.

A negated $KW$-proposition $\neg KW(p)$ represents ignorance about $p$. Although negated $KW$-propositions are not part of PSIPLAN-S, the Closed Knows Whether Assumption allows us to conclude $p$ is unknown, if $KW(p)$ cannot be deduced.

Note also that reasoning in PSIPLAN-S is done without expanding the $\psi$-forms into universal base.
4.1 PSIPLAN-S Action Language

PSIPLAN-S distinguishes two types of actions: domain actions that change the world (e.g., an action of downloading a paper from a url), and sensing actions that do not change the world but only return information about it (e.g., querying a bibliography).

Each domain action has a list of preconditions, $P$, and an encoding of the effects of the action as a set of literals, called the assert list, $A$. Action preconditions identify the propositions necessary for executing the action. The propositions in $P$ can include literals and quantified $\psi$-forms, where the term quantified is used informally to denote a $\psi$-form that uses at least one variable, and thus represents infinite number of ground instances. We assume that an action is deterministic and can change the truth-value of only a finite number of atoms, thus assert list contains literals only, and no quantified $\psi$-forms.

Action $RmDir(?d)$ of deleting an empty directory $?d$, depicted in Figure 2 is an example of a PSIPLAN-S domain action. Its preconditions are a $Kn$-atom and a non-ground $Kn$-$\psi$-form. Its assert list consists of a $Kn$-$\psi$-form denoting a single $Kn$-atom.

<table>
<thead>
<tr>
<th>Domain action $RmDir(?d)$</th>
<th>Sensing action $QueryBib(?b, ?kwd)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P : Kn(Dir(?d)), [Kn(\neg In(x, ?d))]$</td>
<td>$P : Kn(PrefBib(?b))$</td>
</tr>
<tr>
<td>$A : [Kn(\neg Dir(?d))]$</td>
<td>$K : [KW(\neg Rel(y, ?kwd) \lor \neg InCollection(y, ?b))]$</td>
</tr>
</tbody>
</table>

Fig. 2. Example of PSIPLAN-S’s domain and sensing actions. Variables $x$ and $y$ are implicitly universally quantified. Other variables are action schema parameters.

Sensing actions also have preconditions. Effects of the sensing are given by its knowledge list, denoted $K$. The propositions in $K$ are $KW$-$\psi$-forms. After a sensing action is executed, it returns an observation list of $Kn$-propositions corresponding to the information that was learned, denoted $\Delta$. When the knowledge list $K$ of a sensing action contains a non-ground $KW$-$\psi$-form, one of the propositions of $\Delta$ will also necessarily be a non-ground $\psi$-form, since our assumption is that there is always a finite number of "positive" facts.

For example, $QueryBib(?b, ?kwd)$, depicted in Figure 2 is a sensing action that identifies all papers that according to bibliography $?b$ are related to keyword $?kwd$. The precondition requires that $?b$ be a preferred bibliography. The effect of this action is encoded in the knowledge list that contains a non-ground $\psi$-form, and states that as a result of this action the set of all papers in collection of bibliography $?b$ that are related to keyword $?kwd$ in the title will be known.

Suppose after executing sensing action $a = QueryBib(RI, K)$ with effect $[KW(\neg Rel(p, K) \lor \neg InCollection(p, RI))]$ papers $Paper_1$ and $Paper_2$ were found as the only ones containing keyword K in their title, i.e. $\Delta(a)$ consists of the fol-
following propositions:

\[
[Kn(\neg Rel(p, K) \lor \neg InCollection(p, RI)) \text{ except } \{\{p = Paper_1\}, \{p = Paper_2\}\}]
\]
\[
Kn(\neg Rel(Paper_1, K)), Kn(InCollection(Paper_1, RI))
\]
\[
Kn(\neg Rel(Paper_2, K)), Kn(InCollection(Paper_2, RI))
\]

(5)

4.2 Reasoning in PSIPLAN-S

In this section we consider entailment in PSIPLAN-S. We first address entailment of \(Kn\) and \(KW\) propositions (representing domain and knowledge goals) from a set of PSIPLAN-S \(Kn\)-propositions. This kind of reasoning is applied when sensing actions are not included in planning, as sensing actions have \(KW\)-propositions as their effects. We show that in PSIPLAN-S reasoning about domain and knowledge goals is sound and complete; we also provide a rationale for why its complexity is polynomial. This combination of soundness, completeness and tractability is unique among all implemented open world planning formalisms.

Furthermore, we consider entailment from a set of \(Kn\) and \(KW\)-propositions, which is used when sensing actions are applied by the planner to satisfy knowledge goals. We present the sufficient conditions for entailment in such situations and illustrate its use in planning with examples.

4.3 Reasoning without sensing actions

First, consider entailment between two simple \(\psi\)-forms, i.e. \(\psi\)-forms without exceptions.

**Lemma 1.** Let \(\psi\) and \(\psi'\) denote simple \(Kn\)-\(\psi\)-forms, and \(\tilde{\psi'}\) denote a simple \(KW\)-\(\psi\)-form.

- \(\psi \models_k \psi'\) if and only if there exists a substitution \(\sigma\) such that \(M(\psi)\sigma \subseteq M(\psi')\).
- \(\psi \models_k \tilde{\psi'}\) if and only if there exists a substitution \(\sigma\) such that \(M(\psi)\sigma \subseteq M(\psi')\), where \(\psi'\) is obtained by replacing \(KW\) in \(\tilde{\psi'}\) by \(Kn\).

When \(\psi\) is a singleton, the entailment is thus achieved in case of subsumption between the main forms of \(\psi\) and \(\psi'\). In a general case, this Lemma reduces entailment to the existence of a substitution that matches the main form of the entailing \(Kn\)-\(\psi\)-form \(\psi\) onto a subset of clauses of the entailed \(\psi\)-form. In case such substitution exists, for each ground proposition \(c\) of the entailed \(\psi\)-form there exists a ground proposition in the entailing \(\psi\)-form \(\psi\) which alone entails \(c\). Thus, the ground propositions of \(\psi\) should never be combined together

---

\(\text{Here and below, whenever the } \subseteq \text{ sign is used to relate } Kn \text{ or } KW \text{ propositions, we actually mean the subset relationship between the domain literals within the parentheses that follow. For example, we would write } Kn(\neg P \lor \neg Q) \subseteq Kn(\neg R \lor \neg Q \lor \neg P), \text{ since } \{\neg P, \neg Q\} \subseteq \{\neg R, \neg Q, \neg P\}.\)
to establish entailment. This property plays a critical role in the efficiency of \( \psi \)-form reasoning.

The property critical for the efficiency of reasoning is formulated in Theorem 1 below: given a set of \( Kn\psi \)-forms \( \Psi = \{ \psi_1, \ldots, \psi_n \} \), \( \Psi \models_k \psi \) only if there is a \( \psi \)-form \( \psi_i \in \Psi \) that nearly entails \( \psi \), i.e. main part of \( \psi_i \) entails the main part of \( \psi \), or \([M(\psi_i)] \models_k [M(\psi)]\). Note that this is a necessary condition only when the domain of objects is assumed to be infinite, which is the assumption that PSIPLAN-S makes.

**Theorem 1**

Given a set of \( Kn\psi \)-forms \( \Psi = \{ \psi_1, \ldots, \psi_n \} \) and a \( \psi \)-form \( \psi \), \( \Psi \models_k \psi \) only if there is a \( \psi \)-form \( \psi_i \) in \( \Psi \) such that \([M(\psi_i)] \models_k [M(\psi)]\).

Theorem 1 thus requires that to satisfy a \( \psi \)-form goal, the planner must find an effect \( \psi_i \) that nearly entails \( \psi \). But, as exceptions of \( \psi_i \) weaken it, \( \psi_i \) alone may not entail the entire \( \psi \). To represent the propositions of \( \psi \) which are not entailed, we introduce the e-difference operator.

For any two sets of ground propositions \( A \) and \( B \), e-difference is defined as follows.

\[
B \models_k^- A = \{ b \mid b \in B \land A \not\models_k b \}
\]

As \( \psi \)-forms are compact representations of sets of ground propositions, we extend the e-difference operation to \( \psi \)-forms. The algorithms for computing the e-difference between \( \psi \)-forms are not presented in this paper (see [3]), however, we note that the computation is carried out by manipulations on the main form and exceptions of the \( \psi \)-forms without expanding the \( \psi \)-form into the corresponding set of ground propositions. The following example illustrates the e-difference operation.

**Example 1**

Assume \( \psi = [Kn(\neg PrefBib(z)) \text{ except } \{(z = ACM), (z = RI)\}] \), which represents that there are no preferred bibliographies except possibly ACM and RI.

Let \( \bar{\psi} = [KW(\neg PrefBib(x) \lor \neg InCollection(y, x) \lor \neg Rel(y, Kwrd))] \), which can represent a goal of knowing for all papers \( y \) if they are related to keyword Kwrd and are in collection of any preferred bibliography \( x \).

\( \psi \) entails most of \( \bar{\psi} \), indeed, since \( \neg PrefBib(x) \) is true for all values of \( b \) except possibly RI and ACM, then so is the disjunction inside the \( \bar{\psi} \)'s KW clause. Thus, the only parts of \( \bar{\psi} \) that are not entailed by \( \psi \) are

\[
\bar{\psi}_1 = [KW(\neg PrefBib(RI) \lor \neg InCollection(y, RI) \lor \neg Rel(y, Kwrd))] \\
\bar{\psi}_2 = [KW(\neg PrefBib(ACM) \lor \neg InCollection(y, ACM) \lor \neg Rel(y, Kwrd))]
\]

and therefore \( \bar{\psi} \models_k^- \psi = \{ \psi_1, \psi_2 \} \).

This example demonstrates how e-difference is used in planning to identify the part of a non-ground goal \( \bar{\psi} \) that is not implied by the existing knowledge \( \psi \). Indeed, knowing that there are no preferred bibliographies except possibly RI and ACM means no bibliographies except for possibly these two need to be considered, thus reducing the knowledge goal \( \bar{\psi} \) to two simpler goals \( \psi_1 \) and \( \psi_2 \).
The e-difference operator also plays a key role in computing entailment. The next Theorem describes the necessary and sufficient conditions for entailment of a Kn, or KW ψ-form by a set of atoms and Kn-ψ-forms.

We call a set s of Kn and KW-propositions saturated, when the application of rule (R1) does not generate any new propositions not already in s. In other words, s is saturated when for every Kn(a) and Kn(¬a ∨ ¬a_1 ∨ ... ∨ ¬a_n), the resolvent Kn(¬a_1 ∨ ... ∨ ¬a_n) is in s. A saturated equivalent of such a set can always be computed by performing all possible such resolutions.

**Theorem 2** Let s = A ∪ Ψ be a consistent saturated set of Kn-atoms (A) and Kn-ψ-forms (Ψ), and ψ is any ψ-form (either Kn, or KW). s ⊨ k ψ if and only if

1. there exist Kn(a_1),...,Kn(a_n) ∈ A, such that ψ = [KW(¬a_1 ∨ ... ∨ ¬a_n)],

or

2. there exists ψ ∈ Ψ, such that [M(ψ_k)] ⊨ k [M(ψ)], and, furthermore,

   s − ψ ⊨ k (ψ − ψ_k)

Thus, one way of establishing entailment of a ground KW-ψ-form (i.e. a ψ-form without any variables, i.e. denoting a single ground proposition) is to find a set of corresponding Kn-atoms as described in clause 1 of Theorem 2.

Clause 2 applies to any (singleton or non-ground) ψ-form, requires finding a near entailing ψ-form ψ_k, and then checking entailment of the part of ψ that is not entailed by ψ_k. Thus, the Theorem describes a recursive procedure for checking entailment. Importantly, each ψ-form in the result of e-difference ψ − ψ_k will have fewer variables than there are in ψ. This factor limits the depth of the recursion tree to V, which is the maximum number of variables in a ψ-form.

Overall, the complexity of the entire procedure of checking entailment s ⊨ k ψ is $O(β^V n)$, where n is the number of Kn-atoms and ψ-forms in s, V is the maximum number of variables in a ψ-form, and β is the maximum number of ψ-forms in the e-difference. We assume that the cardinality of predicate symbols is constant bounded, and unification takes constant bounded time. We also assume the number of variables, literals and exceptions are all constant bounded. In that case β is also constant bounded, and checking entailment has a linear complexity bound. When the maximum number of exceptions is proportional to n, the procedure’s complexity is bound by a polynomial of the degree $V + 1$.

### 4.4 Adding sensing actions

When sensing actions are added, the planner must reason from a set of propositions which include both Kn and KW ψ-forms. Since the KW-propositions do not contribute to the entailment of Kn-propositions, we consider entailment of only knowledge goals, i.e. KW-ψ-forms.

---

3 Planning with PSPLAN-S requires that it is performed only once in the initial state. The only other time resolutions must be computed is after execution of a sensing action, which adds Kn-propositions, though in this case the saturation computation is limited to the newly added propositions and their consequences.
First, we present an analog of Lemma 1, establishing the necessary and sufficient conditions for entailment between two simple \(KW\)-\(\psi\)-forms. However, to achieve an important property of \(Kn\)-\(\psi\)-form entailment which follows from Lemma 1 in the case of \(KW\)-\(\psi\)-form entailment, namely, the fact that each ground proposition of the entailed \(\psi\)-form is always entailed by some ground proposition of the entailing \(\psi\)-form alone, we must require that the main form of each \(KW\)-\(\psi\)-form does not contain duplicate predicate symbols. Consider the following example:

Example 2 Let

\[
\bar{\psi}_1 = [KW(\neg P(x, y))]
\]
\[
\bar{\psi}_2 = [KW(\neg P(A, B) \lor \neg P(C, D))].
\]

It is impossible to conclude that the value of the disjunction \((\neg P(A, B) \lor \neg P(C, D))\) is known from any one of propositions of \(\bar{\psi}_1\). However, from \(\bar{\psi}_1\) it follows that the values of \(\neg P(A, B)\) and \(\neg P(C, D)\) are known, and the value of the disjunction can be determined. \(KW(\neg P(A, B))\) and \(KW(\neg P(C, D))\) are in \(\bar{\psi}_1\) and together they entail \(KW(\neg P(A, B) \lor \neg P(C, D))\).

Lemma 2. Let \(\bar{\psi}, \bar{\psi}'\) denote simple \(KW\)-\(\psi\)-forms without repeated predicate symbols in the main form. \(\bar{\psi} \models_k \bar{\psi}'\) if and only if there exists a substitution \(\sigma\) such that \(M(\bar{\psi})\sigma = M(\bar{\psi}')\).

From now on, when referring to a \(KW\)-\(\psi\)-form we mean only those that do not contain duplicate occurrences of the same predicate symbol in their main form.

The added ways of proving a \(KW\)-\(\psi\)-form goal are based on inference rules (R12) and (R13). We precede the formalulation of the entailment Theorem for the full language with two examples that illustrate it.

Example 3 Rule (R12) essentially states that the value of a disjunction (or conjunction) can be determined from the values of its subclauses. To illustrate the idea, consider the \(\psi\)-form goal from the previous example:

\[
\bar{\psi}_1 = [KW(\neg PrefBib(RI) \lor \neg InCollection(y, RI) \lor \neg Rel(y, Kwrd))].
\]

The above goal can be achieved by a combination of sensing operators with effects of knowing if the ResearchIndex that are related to keyword \(Kwrd\), which is expressed with a \(\psi\)-form \(KW(\neg InCollection(y, RI) \lor \neg Rel(y, Kwrd))\).

An example from a different domain illustrates application of rule R(13). This rule is based on the fact that the value of a clause \(c\) can be determined by resolution between an atom \(a\) and the clause \(\neg a \lor c\).
Imagine a goal of finding out if vehicle $V$ is in garage $G$, i.e. 
$[KW(\neg In(V,G))]$. Suppose there was a sensing action $AllTrucks(G)$ with the effect $[KW(\neg In(v,G) \lor \neg Truck(v))]$ identifying all trucks in garage $G$. If it is known that $V$ is indeed a truck, i.e. $Kn(Truck(V))$, then after executing the $AllTrucks(G)$ action it will be known if $V$ is in the garage $G$, in other words, the goal $[KW(\neg In(V,G))]$ will be achieved.

Differently from the reasoning without sensing, we do not make the completeness case for reasoning about $KW$-propositions from a set of $Kn$ and $KW$-propositions, because we have not shown completeness of the ground language $\mathcal{L}_K$ with respect to this kind of reasoning. However, we believe it to be true, based on Conjecture 1.

**Theorem 3** Let $A$ denote a set of $Kn$-atoms $\{Kn(a_1),\ldots,Kn(a_n)\}$, $\Psi$ denote a set of $Kn$-$\psi$-forms $\{\psi_1,\ldots,\psi_t\}$, and $\hat{\Psi}$ denote a set of $KW$-$\psi$-forms $\{\hat{\psi}_1,\ldots,\hat{\psi}_t\}$.

Assume $A \cup \Psi \cup \hat{\Psi}$ is consistent and saturated. Let $\hat{\psi}$ be a $KW$-$\psi$-form. Then, $A \cup \Psi \cup \hat{\Psi} \models_k \hat{\psi}$ if at least one of the following conditions holds:

1. There are $Kn(a_1),\ldots,Kn(a_n) \in A$, such that $\hat{\psi} = [KW(\neg a_1 \lor \ldots \lor \neg a_n)]$.
2. There is a $\psi_k \in \Psi \cup \hat{\Psi}$ such that $[M(\psi_k)] = [M(\hat{\psi})]$, and $A \cup \Psi \cup \hat{\Psi} - \psi_k \models_k \hat{\psi} - \psi_k$.
3. There is a decomposition of the main part of $\hat{\psi}$ into two subclauses $Q'$ and $Q''$ such that $M(\hat{\psi}) = Q' \lor Q''$ and

$$A \cup \hat{\Psi} \models_k [KW(Q') | \Sigma'],$$

$$A \cup \hat{\Psi} \models_k [KW(Q'') | \Sigma''].$$

where $\Sigma'$, $\Sigma''$ are subsets of the set of exceptions of $\hat{\psi}$, which only use variables from $Q'$ and $Q''$, respectively.

4. Assume $\hat{\psi} = [KW(Q) \text{ except } \Sigma]$. There is an atom $Kn(a_i) \in A$ and $\psi_i \in \hat{\Psi}$ such that $\hat{\psi} \models_k [KW(\neg a_i \lor Q) \text{ except } \Sigma]$.

Cases 1 and 2 of this Theorem repeat the conditions of Theorem 2. Cases 3 and 4 describe entailment that involves $KW$-premises. These entailment methods were demonstrated in Examples 3 and 4.

Unlike the case of only $Kn$-premises, the completeness of these methods is not guaranteed, and its complexity is no longer polynomial in the number of propositions, since different combinations of $KW$-$\psi$-forms and $Kn$-atoms must be considered to cover cases 3 and 4. In our experience with using the language for planning in the Writer’s Aid application, however, this theoretically exponential upper bound did not present a practical problem, due to a small number of cases in which we had to apply cases 3 and 4 of the Theorem.

### 4.5 SOK Update After an Action and Projection

**SOK update.** As actions cause transitions between the world states, the agent’s SOKs must evolve in parallel with the world, and must adequately reflect the
changes in the world that occur due to an action. Correctness of an SOK update guarantees that the SOK is always consistent with the world model, given a consistent initial SOK. The other desirable property of the SOK update is completeness: we would like the agent to take advantage of all information that becomes available and not to discard what was previously known and has not changed. Clearly, the correctness and completeness properties of the SOK update, as well as soundness and completeness of entailment within the state language are prerequisites for a sound and complete planning algorithm.

The correctness and completeness criteria are best formulated in the context of possible worlds. Let do(a, W) denote the set of world states obtained from performing action a in any of the world states in W, and update(s, a) denote the SOK that results if the agent performs action a from SOK s. We say that the update procedure is correct iff

\[ \mathcal{B}(update(s, a)) \subseteq do(a, \mathcal{B}(s)) \]  

i.e. every possible world after performing the action a has to have a possible predecessor.

The update procedure is complete iff

\[ do(a, \mathcal{B}(s)) \subseteq \mathcal{B}(update(s, a)) \]  

i.e. every world obtained from a previously possible world is accessible from the new SOK. This implies that all changes to the world must be reflected in the new SOK.

To achieve correctness of SOK updates, the agent must remove from the SOK all propositions whose truth value might have changed as the result of the performed action. In order to be complete, the agent must also add to the SOK all facts that become known.

To update SOK s after performing a domain action ad all propositions whose truth value could have been changed must be removed from s – these are all propositions entailed by the negation of some effect of ad. The propositions entailed by effects of ad are also removed, and then the effects of ad are added to the new SOK. The agent’s SOK after executing a domain action ad in the SOK s is computed as shown below.

\[ update(s, a_d) = ((s \models A^{-}(a_d)) \models A(a_d)) \cup A(a_d), \]  

where \( A^{-}(a_d) \) denotes the set of propositions obtained by negating each domain proposition inside the Kn() in ad’s assert list \( A(a_d) \).

**Example 5** For an example, consider action \( a = Download(P, U, D) \) of downloading file P from url U into directory D. Let precondition \( \mathcal{P}(a) = \{Kn(Location(P, U))\} \), denote that P is located at url U. The effect of this action is having file P stored in directory D, described by \( A(a) = \{Kn(In(P, D))\} \).

We begin with an SOK s, which states that location of paper P is U, and that A is the only Postscript file in the system,. T(x, PS) below denotes that
the type of file $x$ is Postscript.

$$s = \left\{ \begin{array}{l} Kn(Location(P, U)), Kn(T(A, PS)), \\ [Kn(\neg In(x, y) \lor \neg T(x, PS)) \text{ except } \{x = A\}] \end{array} \right\}$$

$a = Download(P, U, D)$ is executable in $s$, and $A^{-}(a) = \{[Kn(\neg In(P, /img)])\}$. The e-difference $s \searrow A^{-}(a)$ results in the exception $\{x = P, y = D\}$ being added to the $\psi$-form. Further e-difference with $A(a)$ and union with $A(a)$ yields the following SOK

$$s' = \left\{ \begin{array}{l} Kn(Location(P, U)), Kn(T(A, PS)), \\ [Kn(\neg In(x, y) \lor \neg T(x, PS)) \text{ except } \{x = A, x = P, y = D\}] \\ Kn(In(P, D)) \end{array} \right\}$$

Note that $s$ entailed $Kn(\neg In(P, D) \lor \neg T(P, PS))$ and that we added the effect $Kn(In(P, D))$ when determining $s'$. If our update rule retained $Kn(\neg In(P, D) \lor \neg T(P, PS))$ as a part of a $\psi$-form in $s'$, then in $s'$ we could perform resolution and conclude that $\neg T(P, PS)$. However, this would be wrong because we have no information on whether or not $P$ is a Postscript file. Instead, our update rule removes any clause that is entailed by $\neg In(P, D)$, and so $s'$ does not entail $\neg In(P, D) \lor \neg T(P, PS)$.

The update (8) produces the same result as Winslett’s update operator [20] in the special case where actions are deterministic. Moreover, our rule accomplishes this without considering all possible worlds corresponding to SOK $s$ explicitly, and thus is efficient. After the execution of a sensing action $a_s$, the set of observed propositions, denoted below by $\Delta(a_s)$ is added to the SOK, i.e.

$$update(s, a_s) = s \cup \Delta(a_s) \quad (9)$$

After propositions from $\Delta(a)$ are added to the SOK, all possible resolutions from SOK propositions must be computed and added to the new SOK to keep it saturated.

**Theorem 4** Assume $s$ is consistent and saturated. The state of knowledge update procedure (8),(9) is correct and complete, i.e. $do(a, B(s)) = B(update(s, a)).$

**SOK Projection.** In order to plan, it is necessary to predict the state of knowledge after an action is performed. This is realized using the projection operation, $project(s, a)$. For a domain action, assuming no execution failure, $project(s, a_d) = update(s, a_d)$.

For a sensing action projection is different from updating, since it occurs prior to executing the sensing action and no results of sensing are available:

$$project(a_s, s) = s \cup K(a_s). \quad (10)$$

Projection of a sensing action, thus, simply adds the knowledge list to the agent’s SOK, establishing that in the future SOK the value of some propositions will become known, without committing to the actual truth value of those propositions.
Note that the result of projection \( \text{project}(a_s, s) \) is not a SOK, since PSIPLAN-S SOKs do not contain \( KW \)-propositions. We call the result of projection a \textit{projected} SOK, or \( PSOK \), denoted \( \tilde{s} \), and apply the Closed Knows Whether Assumption to PSOKs as we do with SOKs: \( CKWA(\tilde{s}) = \tilde{s} \cup \{ \neg KW(p) \mid p \in L_D \land \tilde{s} \not\models KW(p) \} \).

Note that the result of the projection does not change the set of possible worlds corresponding to the SOK, i.e.
\[
B(\text{project}(a_s, s)) = B(s),
\]
but it has fewer models, since those models of \( s \) that are uncertain about the value of propositions in \( K(a_s) \) are not part of the PSOK \( \text{project}(a_s, s) \).

**Example 6** Using the projection operator and Proposition 3, we can show that the goal \( \tilde{\psi}_1 = [KW(\neg PrefBib(RI) \lor \neg InCollection(y, RI) \lor \neg Rel(y, Kwrd)) ] \) can be established by a combination of two sensing actions: \( a_1 \) that finds out if \( RI \) is a preferred bibliography, i.e. has effect \( [KW(\neg PrefBib(RI))] \), and \( a_2 = \text{QueryBib}(RI, Kwrd) \) with effect \( [KW(\neg InCollection(y, RI) \lor \neg Rel(y, Kwrd)) ] \), since \( (\text{project}(\{a_1, a_2\}, s)) \models_h \tilde{\psi}_1 \) (the entailment follows from case 3 of Theorem 3).

Note further that if \( a_1 \) is executed first and returns with observation \( [Kn(\neg PrefBib(RI))] \), this proposition would be added to the SOK by update (9), and the updated SOK will entail \( \psi_1 \) (from case 2 of Theorem 3), rendering this goal satisfied.

5 Related Work

PSIPLAN-S’s use of the modal operators \( Kn \) and \( KW \) is inspired by the theoretical foundation for representing knowledge and sensing and the solution to the frame problem developed by Scherl and Levesque [17]. Their solution is developed within the framework of the \textit{situation calculus} [15] in which general reasoning is not decidable. The goal of PSIPLAN-S is to provide a representation for open world planning with sensing that uses the modal operators \( Kn \) and \( KW \) [17] in a less expressive language for the sake of decidability and tractability.

Liu and Levesque [11] present a subset of situation calculus [12, 16] with equality with complete and tractable reasoning. The completeness of determining whether a formula \( \phi \) is true after executing an action is achieved by restricting \( \phi \) to be in a certain normal form, the knowledge base to a certain \textit{proper} form and restricting the actions to those with quantifier-free conditional effects. Furthermore, the preconditions of each conditional effect must be known in the knowledge base which describes the situation in which the action is applied. The language of Liu and Levesque does not subsume PSIPLAN-S, however there is some overlap between the two.

Thielscher [18] presents FLUX - a logical programming framework for agent program design in the presence of incomplete information and sensing. FLUX is based on fluent calculus and is implemented as a set of constraints, defining
the domain, action update, agent’s knowledge and action execution. The syntax of the language is carefully restricted to provide linear time evaluation of the constraints. The constraint language includes universally quantified negated clauses, similar to the simple $\psi$-forms of PSIPLAN-S. However, the constraint solver assumes a finite domain, and does not represent exceptions to the universally quantified clauses. FLUX has nice computational properties, but it is not complete. Differently from PSIPLAN-S the presented FLUX framework is designed for programming the intended behavior of the agent via a designer-specified strategy, which defines the set of agent control rules. PSIPLAN-S has so far been used for automated planning, i.e. the problem of automatically constructing a sequence of actions that will result in the achievement of the goal.

The work most closely related to PSIPLAN-S’s representation is Bacchus and Petrick [14]. Their knowledge state representation uses propositions similar to PSIPLAN-S’s $Kn$ and $KW$-propositions, and their sensing actions have $KW$-effects. While they use quantified propositions like our $[KW(Q(x))]$, but without exceptions, to represent the fact that the agent knows the truth values of all ground instances of $[KW(Q(x))]$, their goals are limited to ground propositions. Their representation is based on the LCW-representation ([8]), which is tractable but incomplete. The LCW representation has also been used in PUCCINI planner [9]. By contrast, in PSIPLAN-S reasoning about domain and knowledge goals from an agent’s SOK is complete and tractable, and reasoning from a projected SOK is tractable and, we conjecture, complete.

The action languages in both PUCCINI and planner of Bacchus and Petrick are different from PSIPLAN-S’s: there are no universally quantified preconditions, but there are conditional effects of actions. Conditional effects specify propositions that become true if a certain condition holds prior to taking the action. While conditional effects can be easily incorporated in PSIPLAN-S, by excluding conditional effects from PSIPLAN-S actions, we are able to specify exactly the set of propositions whose truth value is changed when executing an action, thus contributing to the tractability of reasoning. Planning with conditional effects and without sensing has been shown (e.g. [19, 4]) to be outside of the NP class of problems. Such limitations on action specification does not seem to preclude PSIPLAN-S’s applicability to some real-life domains, as demonstrated by its use in Writer’s Aid ([1]). We are currently working on incorporating conditional effects in a Graphplan-based conformant PSIPLAN planner.

6 Conclusions and Future Work

We have presented PSIPLAN-S, a representation language for reasoning and planning in open world applications with sensing. Several distinguishing features are critical for PSIPLAN-S’s suitability in practice.

- PSIPLAN-S is based on a language of knowledge propositions $L_k$, which was carefully designed to support sound and complete reasoning about domain and knowledge goals from the agent’s state of knowledge.
- **PSIPLAN-S** introduces a novel Closed Knows Whether Assumption to represent the agent’s ignorance. Verifying an agent’s ignorance about a proposition is a necessary component of ensuring non-redundant sensing.

- to represent statements that are quantified over the entire domain of discourse, **PSIPLAN-S** uses $\psi$-forms, which represent infinite sets of propositions of $L_{KC}$. Completeness of reasoning from an agent’s SOK in **PSIPLAN-S** follows from the same properties of reasoning in $L_{KC}$ and $\psi$-forms.

- **PSIPLAN-S**’s SOK update after an action completely and correctly reflects the transition due to an action without expanding the SOK into the set of possible worlds. Although we have not shown it here, the update procedure has polynomial complexity in the number of propositions in SOK [3].

- **PSIPLAN-S**’s projection operator predicts the future SOK and serves as a basis for planning with sensing actions.

Thus, **PSIPLAN-S** efficiently handles domains with an incomplete specification of the initial state without considering the set of all possible worlds, and does not require that the agent know the set of all objects in the domain. In the future, we will extend **PSIPLAN-S** to allow function symbols. Another interesting and important task is proving the completeness of the rule system for $L_{KC}$.

## References


7 Proofs

We restate all propositions here for convenience.

**Proposition 1** Let $p_1, \ldots, p_k, q$ be domain propositions:

$$(p_1, \ldots, p_k \models q) \iff (Kn(p_1) \ldots Kn(p_k) \models_k Kn(q)).$$

**Proof.** Follows from the definition of $\models_k$ (on page 6).

**Proposition 2** Let $c$ be a domain clause.

1. If $a$ is a domain atom, $Kn(a) \models_k KW(c)$ if and only if $c = \neg a$.
2. If $c'$ is a domain clause $Kn(c') \models_k KW(c)$ if and only if $c' \subseteq c$.
3. If $a_1, \ldots, a_k$ are domain atoms, and $c_1, \ldots, c_n$ are domain clauses, and the set \{a_1, \ldots, a_k, c_1, \ldots, c_n\} is consistent,

$$s = \{Kn(a_1), \ldots, Kn(a_k), Kn(c_1), \ldots, Kn(c_n)\} \models_k KW(c)$$

if and only if
(a) there exists a domain atom \( a \), such that \( Kn(a) \in S \) and \( c = \neg a \), or
(b) there exists a domain clause \( c' \), such that \( s \models_k Kn(c') \) and \( c' \subseteq c \).

**Proof.** The if parts of claims 1 and 2 of this proposition are easily verified using the definitions of the set of models \( \alpha() \) and entailment in Section 2.3. The if parts of claim 3 follows from the claims 1 and 2. It is only left to prove the only if part.

1. The only restriction that the proposition \( Kn(a) \) places on its set of models \( (w, W) \in \alpha(Kn(a)) \) is that \( w \in W \) and for all world states \( w' \in W \), \( a \) must be true in \( w' \). On the other hand, to be a model of \( KW(c) \), all worlds in \( W \) should have the same value of \( c \). Since \( c \) is a domain clause, a model of \( Kn(a) \) is also a model of \( KW(c) \) if and only if \( c = \neg a \).

2. Similar to the previous case, the only restriction that the proposition \( Kn(c') \) places on its set of models \( (w, W) \in \alpha(Kn(c')) \) is that \( w \in W \) and for all world states \( w' \in W \), \( c' \) must be true in \( w' \).

For all such models to also be a model of \( KW(c) \), \( c' \) must either entail \( c \) or it must entail \( \neg c \). Since both \( c' \) and \( c \) are disjunctions of negated atoms, it is only possible that \( c' \models c \), i.e. \( c' \subseteq c \).

3. Similar to the previous cases, the key observation that we use is that for a model of \( S \) to also be a model of \( KW(c) \), \( a_1, \ldots, a_k, c_1, \ldots, c_n \) must together either entail \( c \) or entail \( \neg c \).

The set of clauses that the set \( D = \{a_1, \ldots, a_k, c_1, \ldots, c_n\} \) entails is limited to the set of clauses obtained from unit clause resolution rule and the subclause rule. The set of entailed atoms is exactly \( a_1, \ldots, a_k \), because we cannot derive any new atoms by application of either rule.

Since \( c \) is a disjunction of negated atoms, it is only possible that \( D \models c \), if \( D \models c' \), such that \( c' \subseteq c \). On the other hand, if \( c \) is a single negated literal, then \( D \models \neg c \) iff \( c = \neg a \) and \( a \in D \).

**Proposition 3** \( KW(c_1), \ldots, KW(c_n) \models_k KW(c) \) if and only if there exist a set of indices \( i_1, \ldots, i_k \) where \( 1 \leq i_j \leq n \) for each \( i_j \), such that \( c = c_{i_1} \lor \cdots \lor c_{i_k} \).

**Proof.** The if part of the claims follows from the soundness of rule (R13).

So it is left to prove that \( KW(c_1), \ldots, KW(c_n) \models_k KW(c) \) only if there is such a subset of clauses \( c_1, \ldots, c_n \). Suppose there’s no such subset, i.e., for all such subsets \( \{c_{i_1}, \ldots, c_{i_k}\} \), \( c_{i_1} \lor \cdots \lor c_{i_k} \neq c \). We show that in that case there exists a model of \( KW(c_1), \ldots, KW(c_n) \) that is not a model of \( KW(c) \), thus contradicting the entailment.

Let \( q \) denote the clause \( c_1 \lor \cdots \lor c_n \). Here we consider each clause as a set of negated literals. We define sets of domain literals \( A, B \) and \( C \) as follows.

\[
A = q - c, B = q \cap c, C = c - q
\]

Thus, \( A \) is the set of literals of \( q \) that are not in \( c \), \( B \) is the set of common literals of \( q \) and \( c \), and \( C \) is the set of literals of \( c \) that are not in \( q \), and

\[
q = A \cup B, c = B \cup C
\]

22
Note that $B \neq \emptyset$. When $B = \emptyset$, then $q$ and $c$ are disjoint, and we can build a model of $KW(c_1), \ldots, KW(c_n)$, in which there are two worlds that have different values of $c$.

Three cases ensue:

1. Suppose $A = \emptyset$. If $C$ were equal to $\emptyset$, we would have that $c = B = q$, violating our assumption that there is no subset of clauses of the set $\{c_1, \ldots, c_n\}$ that equals $c$. Thus, $B = q$ and $C$ is non-empty. Let $k$-state $(w, W)$ be a model of $KW(c_1), \ldots, KW(c_n)$ such that $W$ contains two world states $w', w''$ such that

$$w'(B) = \text{false} \quad w''(B) = \text{false}$$
$$w'(C) = \text{true} \quad w''(C) = \text{false}$$

This $k$-state is not a model of $KW(c)$, because $w'(c) \neq w''(c)$.

2. Suppose $A \neq \emptyset$ and suppose $C \neq \emptyset$. Then let $k$-state $(w, W)$ be a model of $KW(c_1), \ldots, KW(c_n)$ such that $W$ contains two world states $w', w''$ where

$$w'(A) = \text{true} \quad w''(A) = \text{true}$$
$$w'(B) = \text{false} \quad w''(B) = \text{false}$$
$$w'(C) = \text{true} \quad w''(C) = \text{false}$$

This $k$-state is not a model of $KW(c)$, because then $w'(c) = \text{true}$, but $w''(c) = \text{false}$, since $c = B \cup C$.

3. Suppose $A \neq \emptyset$ and suppose $C = \emptyset$. Then $c = B$ There must be a clause $c_i$ such that $c_i$ has literals in both $A$ and $B$. Otherwise, every $c_i$ would either be a subset of $A$, or a subset of $B$, in which case selecting all those $c_i$ that are subsets of $B$ would yield a subset of $q$ whose union equals $c$, which would violate our assumption.

For simplicity, we assume that there is a unique such $c_i$. When there are several clauses that are partly in $A$ and partly in $B$, the construction is very similar.

Let $k$-state $(w, W)$ be a model of $KW(c_1), \ldots, KW(c_n)$ such that $W$ contains two world states $w', w''$ where

$$w'(A - c_i) = \text{true} \quad w''(A - c_i) = \text{true}$$
$$w'(A \cap c_i) = \text{true} \quad w''(A \cap c_i) = \text{false}$$
$$w'(B - c_i) = \text{false} \quad w''(B - c_i) = \text{false}$$
$$w'(B \cap c_i) = \text{false} \quad w''(B \cap c_i) = \text{true}$$

Here, $\cap$ and $-$ denote the clause that results from intersection and set difference of literals, respectively, between two sets. Since $c = B$, then $w'(c) = \text{false}$, but $w''(c) = \text{true}$, i.e. $(w, W)$ is not a model of $KW(c)$. 

23
**Proposition 4** $Kn(p_1), \ldots, Kn(p_k) \models_k Kn(c)$ if and only if it can be derived using rules (R1) and (R2).

*Proof.* Rules (R1) and (R2) are analogous to unit clause resolution and subsumption rules of propositional logic. Given Proposition 1 and the fact that the deduction in the domain language $\mathcal{L}_D$ is closed under unit clause resolution and subsumption, we conclude that deduction using rules (R1) and (R2) is complete for $Kn$-atoms.

**Proposition 5** $Kn(p_1), \ldots, Kn(p_k) \models_k KW(c)$ if and only if it can be derived using rules (R1), (R2), (R10) and (R11).

*Proof.* The if-part can be easily verified by verifying the soundness of each rule.

To prove the only if-part of this proposition we use case 3 of Proposition 2, which identifies the only possible cases of entailment between a set of $Kn$-propositions and a $KW$-proposition. Let’s denote the set $\{Kn(p_1), \ldots, Kn(p_k)\}$ with $s$, for convenience.

Case 3(a) of entailment requires that there is $Kn(a) \in s$, such that $a$ is an atom and $c = \neg a$. If that’s the case, $KW(\neg a)$ can be derived using rule (10).

When case 3(b) applies, i.e. there is a proposition $Kn(c')$, such that $s \models_k Kn(c')$ and $c' \subseteq c$ we note that

1. By Proposition 4, if such $c'$ exists, then $Kn(c')$ is derivable from $s$ using rules (R1) and (R2).
2. An application of (R2) followed by application of (R11) yields $KW(c)$.

**Proposition 6** $B(s) = \{w \mid \forall p. (s \models_k Kn(p)) \Rightarrow w(p)\}$, or in other words, the set of possible worlds is equivalent to the set of all worlds in which everything known to the agent is true.

*Proof.* This proposition is easily verified by checking that (a) for every domain proposition $p$ and SOK $s$, $s \models_k Kn(p)$ if and only if $p$ is true in every possible world of $s$, $B(s)$, and (b) every world $w$ in which every proposition $p$ such that $s \models_k Kn(p)$ is true, $w \in B(s)$.

**Proposition 7** $B(s) = B(CKWA(s))$. (Note that $s$ is a set of ground $Kn$-atoms of $\mathcal{L}_K$).

*Proof.* From the definition of $CKWA(s)$ it follows that $B(CKWA(s)) \subseteq B(s)$.

To prove that $B(s) \subseteq B(CKWA(s))$ consider an arbitrary k-state model $\gamma = (w, W)$ in $\alpha(s)$. If we show that any such $w \in B(s)$ appears in some model $\gamma'$ of $B(CKWA(s))$, that would prove that $B(s) \subseteq B(CKWA(s))$. Given such $\gamma$, we construct $\gamma'$ as follows. Let $w_p$ denote a word state that is obtained from $w$ by negating the value of the atom $p$, in other words, $w_p$ differs from $w$ only in the value of $p$. Let $\gamma'$ be a k-state $(w, W')$, where $W' = \{w\} \cup \{w_p \mid p \text{ is an atom and } s \not\models_k KW(\neg p)\}$. $\gamma'$ is a model of $CKWA(s) = s \cup \{\neg KW(p) \mid p \in \mathcal{L}_D \land s \not\models_k KW(p)\}$, since $w$ satisfies all domain propositions of $s$ and, furthermore, for every ground proposition $\neg KW(\neg p) \in CKWA(s)$, there exist world states in $W'$, namely $w$ and $w_p$ in which the truth values of $p$ are different. Thus, $\gamma'$ is a model of $CKWA(s)$, and therefore $w \in B(CKWA(s))$. 

24
Lemma 1 Let $\psi$ and $\psi'$ denote simple $Kn$-$\psi$-forms, and $\tilde{\psi}'$ denote a simple $KW$-psiform.

- $\psi \models_k \psi'$ if and only if there exists a substitution $\sigma$ such that $\mathcal{M}(\psi)\sigma \subseteq \mathcal{M}(\psi')$.
- $\psi \models_k \tilde{\psi}'$ if and only if there exists a substitution $\sigma$ such that $\mathcal{M}(\psi)\sigma \subseteq \mathcal{M}(\tilde{\psi}')$, where $\tilde{\psi}'$ is obtained by replacing $KW$ in $\tilde{\psi}'$ by $Kn$.

Proof. 1. if part. The existence of $\sigma$ such that $\mathcal{M}(\psi_1)\sigma \subseteq \mathcal{M}(\psi_2)$ means that for every ground clause $c_2$ of $\psi_2$, assuming $c_2 = \mathcal{M}(\psi_2)\sigma'$, there is a clause $c_1$ in $\psi_1$, $c_1 = \mathcal{M}(\psi_1)\sigma'\sigma$, which is a subset of $c_2$, and therefore $\psi_1 \models \psi_2$.

only if part. Suppose $\psi_1 \models \psi_2$, yet there is no substitution $\sigma$ as described in the condition of the Lemma.

Observe, that for $\psi_1$ to entail $\psi_2$ for each a ground clause $c_2$ from $\psi_2$ there must be a ground clause from $\psi_1$ $c_1$ such that $c_1 \subseteq c_2$. Let $p$ be an instance of $\mathcal{M}(\psi_2)$, which is obtained by assigning to each variable of $\psi_2$ a constant value which does not occur anywhere in $\mathcal{M}(\psi_1)$. Since the domain is infinite, this can always be done. Since for any substitution $\sigma$ on variables of $\psi_1$, there exists a literal in $\mathcal{M}(\psi_1)\sigma$, which does not match any of $p$’s disjuncts, there is no subclause of $p$ which is a clause of $\psi_1$, and thus $p$ is not entailed by $\psi_1$.

2. This proof follows the same argument as in part 1.

Theorem 1 Given a set of $Kn$-$\psi$-forms $\Psi = \{\psi_1, \ldots, \psi_n\}$ and a $\psi$-form $\psi$, $\Psi \models \psi$ only if there is a $\psi$-form $\psi_1$ in $\Psi$ such that $[[\mathcal{M}(\psi_1)] = [[\mathcal{M}(\psi)]]$.

Proof. We construct a clause of $[[\mathcal{M}(\psi)]]$ and show that if none of $[[\mathcal{M}(\psi_1)], \ldots, [[\mathcal{M}(\psi_n)]]$ entail it, then $\psi_1, \ldots, \psi_n$ does not entail $\psi$.

Suppose none of $[[\mathcal{M}(\psi_1)], \ldots, [[\mathcal{M}(\psi_n)]]$ entail $[[\mathcal{M}(\psi)]]$. Therefore according to Lemma 1 none of the main parts of these $\psi$-forms subset match onto $\mathcal{M}(\psi)$. Let $\sigma$ be a substitution on variables of $\psi$ that assigns to each variable a constant value that does not occur in any of $\psi_1, \ldots, \psi_n$, nor in the exceptions of $\psi$. This is always possible due to infinite number of constants in the language. None of the clauses in $\mathcal{M}(\psi_1), \ldots, \mathcal{M}(\psi_n)$ subset match onto $c = \mathcal{M}(\psi)\sigma$, because none of the main clauses of these $\psi$-forms subsume $\mathcal{M}(\psi)$ and the constants of $\sigma$ do not appear in any of $\mathcal{M}(\psi_1), \ldots, \mathcal{M}(\psi_n)$. Thus, the clause $c = \mathcal{M}(\psi)\sigma$ of $\psi$ is not entailed by any clause in $\{[[\mathcal{M}(\psi_1)]], \ldots, [[\mathcal{M}(\psi_n)]\}$. Since $\Phi \subseteq \{[[\mathcal{M}(\psi_1)]], \ldots, [[\mathcal{M}(\psi_n)]\}$, we conclude that $\Psi \not\models_k \psi$. We arrive at a contradiction.

Theorem 2 Let $s = A \cup \Psi$ be a consistent saturated set of PSIPLAN $kn$-atoms ($A$) and $kn$-$\psi$-forms ($\Psi$), and $\psi$ is any $\psi$-form (either $Kn$, or $KW$). $s \models_k \psi$ if and only if

1. there exist $Kn(a_1), \ldots, Kn(a_n) \in A$, such that $\psi = [KW(\neg a_1 \lor \ldots \lor \neg a_n)]$,

or

2. there exists $\psi' \in \Psi$, such that $[[\mathcal{M}(\psi')] \models_k [[\mathcal{M}(\psi)]]$, and, furthermore,

$s - \psi \models_k (\psi - \psi_k)$
Proof. The if-part trivially follows from Proposition 2 and definition of e-difference (\(\sim\)). The only if part. First of all, observe that since \(s\) is saturated, i.e. no resolutions are possible, \(s \models_k \psi\) implies \(A \models_k \psi\), or \(\Psi \models_k \psi\), or both.

Suppose \(A \models_k \psi\). This is only possible when \(\psi\) is a \(KW\)-\(\psi\)-form. Hence, the only way \(A \models_k \psi\), is if there exist \(Kn(a_1), \ldots, Kn(a_n) \in A\), such that \(\psi = [KW(\neg a_1 \lor \ldots \lor \neg a_n)]\).

Next, consider the case when \(\Psi \models_k \psi\). This is possible when \(\psi\) is either a \(Kn\) or \(KW\)-\(\psi\)-form. Then, by Theorem 1 there must be a \(\psi\)-form \(\psi_k\) in \(\Psi\) whose main part \([M(\psi_k)]\) entails the main part of \(\psi\), i.e. \([M(\psi)]\). Furthermore, since the exceptions of \(\psi_k\) weaken it, the e-difference \(\psi \sim \psi_k\) must be also entailed by \(s\).

Lemma 2 Let \(\tilde{\psi}, \tilde{\psi}'\) denote simple \(KW\)-\(\psi\)-forms without repeated predicate symbols in the main form. \(\tilde{\psi} \models_k \tilde{\psi}'\) if and only if there exists a substitution \(\sigma\) such that \(M(\psi)\sigma = M(\psi')\).

Proof. This proof follows the same argument as the proof of Lemma 1, except that for a ground \(KW\) proposition to be entailed by another, they must be equal.

Theorem 3 Let \(A\) denote a set of \(Kn\)-atoms \(\{Kn(a_1), \ldots, Kn(a_n)\}\), \(\Psi\) denote a set of \(Kn\)-\(\psi\)-forms \(\{\psi_1, \ldots, \psi_t\}\) and \(\Psi'\) denote a set of \(KW\)-\(\psi\)-forms \(\{\tilde{\psi}_1, \ldots, \tilde{\psi}_t\}\).

Assume \(A \cup \Psi \cup \Psi' \models_k \psi\). Let \(\tilde{\psi}\) be a \(KW\)-\(\psi\)-form. Then, \(A \cup \Psi \cup \Psi' \models_k \psi\) if at least one of the following conditions holds:

1. There are \(Kn(a_1), \ldots, Kn(a_n) \in A\), such that \(\tilde{\psi} = [KW(\neg a_1 \lor \ldots \lor \neg a_n)]\).
2. \([M(\psi_k)] \models_k [M(\tilde{\psi})]\), and \(A \cup \Psi \models_k \psi_k\).
3. \(\psi\) into two subclauses \(Q'\) and \(Q''\) such that \(M(\tilde{\psi}) = Q' \lor Q''\) and
   \[A \cup \Psi \models_k [KW(Q') | \Sigma'],\]
   \[A \cup \Psi \models_k [KW(Q'') | \Sigma'],\]
   where \(\Sigma', \Sigma''\) are subsets of the set of exceptions of \(\tilde{\psi}\), which only use variables from \(Q'\) and \(Q''\), respectively.
4. There is an atom \(Kn(a_i) \in A\) and \(\tilde{\psi}' \in \Psi'\) such that \(\tilde{\psi}' \models_k [KW(\neg a_i \lor Q) \setminus \Sigma]\).

Proof. Cases 1 and 2 follow directly from Theorem 2 and Lemma 2.

Case 3. If there is such a decomposition \(M(\tilde{\psi}) = Q' \lor Q''\), then for any ground \(KW\)-proposition \(c \in \tilde{\psi}\), obtained by instantiating the main form \(M(\tilde{\psi})\) with ground substitution \(\sigma\), we have that \(c = Q'\sigma \lor Q''\sigma\), and since \(A \cup \Psi\) entails \(KW(Q\sigma)\) and \(KW(Q''\sigma)\), by rule (R13) we conclude that \(A \cup \Psi \models_k c\), and thus \(A \cup \Psi \models_k \tilde{\psi}\).

Case 4. The proof is similar to the one for case 3. For every ground instance \(c\) of \(\psi\), there is a ground instance of \(\tilde{\psi}'\), which together with \(Kn(a_i)\) implies \(c\) by application of rule (R13).

Theorem 4 Assume \(s\) is saturated. The state of knowledge update procedure (8),(9) is correct and complete, i.e. \(do(a, B(s)) = B(update(s, a))\).
Proof. We restate the update procedure here.

\[
\text{update}(s,a) = \begin{cases} 
(s - \mathcal{A}^{-}(a)) \sim \mathcal{A}(a)) \cup \mathcal{A}(a), & \text{when } a \text{ is a domain action} \\
s \cup \Delta(a), & \text{when } a \text{ is a sensing action}
\end{cases}
\]

where \(\mathcal{A}^{-}(a)\) denotes the set of propositions obtained by negating each domain proposition inside the \(Kn()\) in \(a\)'s, assert list \(\mathcal{A}(a)\).

The case of update after a sensing action is straightforward, given that the possible world transition model for a sensing action \(a_s\) is defined as \(do(a_s, W) = \{w \mid w \in W \land w(\Delta(a_s))\}\). The rest of the proof considers the case of a domain action \(a\).

For brevity, in this proof we will refer to \textsc{psiplan} \(Kn\)-propositions without the \(Kn\) prefix, i.e. instead of writing \(Kn(p)\), we use \(p\).

Completeness proof. We first show that \(do(a, B(s)) \subseteq B(s')\), i.e. for every world \(w\) in \(B(s)\), its successor, \(w' = do(a, w)\), is in \(B(s')\).

The set of possible worlds consists of all and only worlds that model the agent’s knowledge of the world, i.e. for any \(s \in B(s) = \{w \mid (p \in s) \Rightarrow w(p) = true\}\). Thus, to show that \(w' \in B(s')\) we need to prove, that for every proposition \(p\) in \(s'\), \(w'(p) = true\).

Note that according to the world transition model, \(w' = (w - \mathcal{A}^{-}(a)) \cup \mathcal{A}(a)\).

We partition \(s'\) into \(s_1' = ((s - \mathcal{A}^{-}(a)) \sim \mathcal{A}(a))\) and \(s_2' = \mathcal{A}(a)\). We partition \(w'\) into \(w'_1 = w - \mathcal{A}^{-}(a) - \mathcal{A}(a)\) and \(w'_2 = \mathcal{A}(a)\). Since \(w \in B(s)\), for every \(p_1\) such that \(p_1 \in s_1'\), \(w'_1(p_1)\). Also, for each \(p_2 \in s_2'\), we have \(w'_2(p_2)\). Therefore for every \(p\) such that \(p \in s, w(p)\).

Correctness proof. Now we need to show that \(B(s') \subseteq do(a, B(s))\), i.e. every possible world \(w'\) of \(s'\) has a predecessor \(w\) that is a possible world of \(s\). We need to show that for every \(w' \in B(s')\) there is a world \(w\) such that \(w' = (w - \mathcal{A}^{-}(a)) \cup \mathcal{A}(a)\) where \(w\) is in \(B(s)\) and \(\mathcal{A}^{-}(a)\) denotes the union \(\mathcal{A}(a) \cup \mathcal{A}^{-}(a)\). A possible world is a model of all propositions \(p\) such that \(p \in s\), i.e. \(w\) is in \(B(s)\) if and only if \(w\) models every such domain proposition \(p\).

The proof is by construction. Since \(s\) is saturated, for every proposition \(p\) that is implied by \(s\), there’s a single proposition \(q \in s\) such that \(q \models p\).

STEP 1. Since \(w' = do(a, w)\) we need to include in \(w\) all literals of \(w'\) that are not in \(\mathcal{A}(a)\), because those would not have changed as a result of the action. Let \(w_0 = w' - \mathcal{A}(a)\) and \(w\) will include \(w_0\).

STEP 2. We also include in \(w\) those literals from \(\mathcal{A}^0(a)\) that are known in \(s\), i.e. the literals \(l \in \mathcal{A}(a)^c\) such that \(l \in s\).

STEP 3. At this point every literal or its complement is included in \(w\) except for \(l \in \mathcal{A}^0(a)\) where neither \(l\) nor \(\neg l\) in \(s\). We now describe a procedure for choosing either the literal or its complement for inclusion in \(w\) from these “leftover” literals. Suppose \(\neg l\) is an arbitrary negated literal from this set. Further, let \(C = \{c \mid c \in s \land \neg l \models c\}\), i.e. the set of propositions in \(s\) that are entailed by \(\neg l\). If there is a proposition in \(C\) that is not already implied by some \(p\), where \(p \in w\), then we must include \(\neg l\) in \(w\) in order to keep it accessible from \(s\). Otherwise we may include in \(w\) either \(l\) or \(\neg l\).
The world $w$ is now completely specified and is easy to verify that $w \in B(s)$, as well as $w' = do(a, w)$. 

28