Efficient Representations for the Modal Logic S5
Alexandre Niveau, Bruno Zamuttini

To cite this version:
Alexandre Niveau, Bruno Zamuttini. Efficient Representations for the Modal Logic S5. Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, Jul 2016, New York City, United States. hal-01311642

HAL Id: hal-01311642
https://hal.archives-ouvertes.fr/hal-01311642
Submitted on 23 Aug 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Efficient Representations for the Modal Logic S5  
(first revision from published version)

Alexandre Niveau and Bruno Zanuttini  
GREYC, UMR 6072, UNICAEN/CNRS/ENSICAEN, France  
{alexandre.niveau,bruno.zanuttini@unicaen.fr}

Abstract  
We investigate efficient representations of subjective formulas in the modal logic of knowledge, S5, and more generally of sets of sets of propositional assignments. One motivation for this study is contingent planning, for which many approaches use operations on such formulas, and can clearly take advantage of efficient representations. We study the language S5-DNF introduced by Bienvenu et al., and a natural variant of it that uses Binary Decision Diagrams at the propositional level. We also introduce an alternative language, called Epistemic Splitting Diagrams, which provides more compact representations. We compare all three languages from the complexity-theoretic viewpoint of knowledge compilation and also through experiments. Our work sheds light on the pros and cons of each representation in both theory and practice.

1 Introduction  
The epistemic modal logic S5 is the logic of monagent knowledge [Fagin et al., 1995], allowing for statements such as \((Kp \lor Kp) \land (\neg K(p \land q))\), which means that the agent knows that \(p\) is true or knows that \(p\) is false (i.e., it knows the value of \(p\)), but does not know that \(p \land q\) is true (it knows that \(p \land q\) is false, or does not know whether it is true or false).

A particular setting where the logic S5 arises naturally is that of contingent planning [Herzig et al., 2003; Petrick & Bacchus, 2004; Hoffmann & Brafman, 2005; Iocchi et al., 2004; Bonet & Geffner, 2014], which is the problem of computing a plan towards a given goal, using two kinds of actions. Ontic actions change the actual state of the world in a non-deterministic fashion, and epistemic actions give the agent feedback about the actual state. The plan sought for can be conditional on the feedbacks received from the epistemic actions.

It is not hard to see that at any moment, an agent executing a contingent plan has a unique set of states which are candidates for being the actual state of the environment. Such a set is called a belief state in the planning literature. For instance, if the agent knows that the initial state satisfies \(p \land q\), and executes an ontic action switch which non-deterministically switches either the value of \(p\) or that of \(q\), then the resulting belief state can be described by \((p \land q) \lor (p \land \neg q)\). If the agent then executes the epistemic action test, which indicates whether \(p \land q\) is true, and receives the feedback that it is the case, then the resulting set can be described by \((p \land q) \lor r\).

When planning, or verifying the validity of a plan, it is also useful to consider the evolution of several possible belief states at the same time, a process usually called offline progression. For instance, if the initial belief state is \(p \land q\), then the ontic action switch leads to \((p \land q) \lor (p \land \neg q)\) as above, but the epistemic action test leads to two possible belief states, depending on its feedback: either \((p \land q) \lor (p \land q \land r)\) if feedback \(p \lor r\) is received. So, in planning and in other applications, it is important to be able to handle general S5 formulas, which represent sets of belief states. In the example above, we would use the representation \(K((p \land q) \lor (p \land q \land r))\) (note that planning usually does not use only-knowing [Levesque, 1990]: goals being typically positive, knowing more is always better—in particular, one often needs only positive knowledge formulas).

Motivated by such uses in planning, we investigate several representations of (subjective) S5 formulas from the point of view of space and time efficiency. We consider the s-S5-DNF, s-S5-CNF representation proposed by Bienvenu et al. [2010], as well as its natural variant s-S5-DNF, s-S5-CNF. We moreover introduce a new representation (Sections 3 and 4), using structures that we call Epistemic Splitting Diagrams (ESDs), which use ideas similar to Binary Decision Diagrams. We investigate these three languages from the point of view of knowledge compilation [Darwiche & Marquis, 2002], comparing their ability to support queries and transformations efficiently (Section 5) and to represent S5 formulas succinctly (Section 6). Finally, we report on experiments, which confirm in practice the properties of the different languages (Section 7). Our results show that each language has its pros and cons; they also show that ESDs are more compact than previous representations for positive S5 formulas.

2 Preliminaries  
S5  The reader is supposed to be acquainted with the basic concepts of propositional logic. We consider the language of propositional S5 [Fagin et al., 1995], in which formulas are built on a set of propositional atoms \(X\) with the usual connectives \(\land, \lor, \land\) and the knowledge modality \(K\). For instance, \((Kx_1 \land \neg (Kx_2 \lor x_3)) \lor (\neg Kx_1)\) is an S5 formula. We denote by \(\text{Var}(\Phi)\) the set of propositional atoms mentioned in a formula.

Abstract  
We investigate efficient representations of subjective formulas in the modal logic of knowledge, S5, and more generally of sets of sets of propositional assignments. One motivation for this study is contingent planning, for which many approaches use operations on such formulas, and can clearly take advantage of efficient representations. We study the language S5-DNF introduced by Bienvenu et al., and a natural variant of it that uses Binary Decision Diagrams at the propositional level. We also introduce an alternative language, called Epistemic Splitting Diagrams, which provides more compact representations. We compare all three languages from the complexity-theoretic viewpoint of knowledge compilation and also through experiments. Our work sheds light on the pros and cons of each representation in both theory and practice.

1 Introduction  
The epistemic modal logic S5 is the logic of monagent knowledge [Fagin et al., 1995], allowing for statements such as \((Kp \lor Kp) \land (\neg K(p \land q))\), which means that the agent knows that \(p\) is true or knows that \(p\) is false (i.e., it knows the value of \(p\)), but does not know that \(p \land q\) is true (it knows that \(p \land q\) is false, or does not know whether it is true or false).

A particular setting where the logic S5 arises naturally is that of contingent planning [Herzig et al., 2003; Petrick & Bacchus, 2004; Hoffmann & Brafman, 2005; Iocchi et al., 2004; Bonet & Geffner, 2014], which is the problem of computing a plan towards a given goal, using two kinds of actions. Ontic actions change the actual state of the world in a non-deterministic fashion, and epistemic actions give the agent feedback about the actual state. The plan sought for can be conditional on the feedbacks received from the epistemic actions.

It is not hard to see that at any moment, an agent executing a contingent plan has a unique set of states which are candidates for being the actual state of the environment. Such a set is called a belief state in the planning literature. For instance, if the agent knows that the initial state satisfies \(p \land q\), and executes an ontic action switch which non-deterministically switches either the value of \(p\) or that of \(q\), then the resulting belief state can be described by \((p \land q) \lor (p \land \neg q)\). If the agent then executes the epistemic action test, which indicates whether \(p \land q\) is true, and receives the feedback that it is the case, then the resulting set can be described by \((p \land q) \lor r\).

When planning, or verifying the validity of a plan, it is also useful to consider the evolution of several possible belief states at the same time, a process usually called offline progression. For instance, if the initial belief state is \(p \land q\), then the ontic action switch leads to \((p \land q) \lor (p \land \neg q)\) as above, but the epistemic action test leads to two possible belief states, depending on its feedback: either \((p \land q) \lor (p \land q \land r)\) if feedback \(p \lor r\) is received. So, in planning and in other applications, it is important to be able to handle general S5 formulas, which represent sets of belief states. In the example above, we would use the representation \(K((p \land q) \lor (p \land q \land r))\) (note that planning usually does not use only-knowing [Levesque, 1990]: goals being typically positive, knowing more is always better—in particular, one often needs only positive knowledge formulas).

Motivated by such uses in planning, we investigate several representations of (subjective) S5 formulas from the point of view of space and time efficiency. We consider the s-S5-DNF, s-S5-CNF representation proposed by Bienvenu et al. [2010], as well as its natural variant s-S5-DNF, s-S5-CNF. We moreover introduce a new representation (Sections 3 and 4), using structures that we call Epistemic Splitting Diagrams (ESDs), which use ideas similar to Binary Decision Diagrams. We investigate these three languages from the point of view of knowledge compilation [Darwiche & Marquis, 2002], comparing their ability to support queries and transformations efficiently (Section 5) and to represent S5 formulas succinctly (Section 6). Finally, we report on experiments, which confirm in practice the properties of the different languages (Section 7). Our results show that each language has its pros and cons; they also show that ESDs are more compact than previous representations for positive S5 formulas.

2 Preliminaries  
S5  The reader is supposed to be acquainted with the basic concepts of propositional logic. We consider the language of propositional S5 [Fagin et al., 1995], in which formulas are built on a set of propositional atoms \(X\) with the usual connectives \(\land, \lor, \land\) and the knowledge modality \(K\). For instance, \((Kx_1 \land \neg (Kx_2 \lor x_3)) \lor (\neg Kx_1)\) is an S5 formula. We denote by \(\text{Var}(\Phi)\) the set of propositional atoms mentioned in a formula.
\( \Phi \) (i.e., \( \{x_1, x_2, x_3\} \) in the previous example). For space reasons, and motivated by the representations typically needed in planning, we restrict our study to subjective S5, in which one can express statements about the knowledge of agents, but not about the actual state of the world (even though our study could be rather directly extended to general S5 formulas). We thus only consider formulas in which all propositional atoms appear in the scope of K: e.g., \( x_1 \land Kx_2 \) is not a subjective formula. Since any S5 formula is equivalent to one without nested modalities, we use the following definition.

**Definition 1.** A subjective S5 formula over \( X \) is a Boolean combination, using \( \neg, \land, \lor \), of epistemic atoms of the form \( K\varphi \), where each \( \varphi \) is a propositional formula over \( X \).

We use uppercase (resp. lowercase) Greek letters \( \Phi, \Psi, \ldots \) (resp. \( \varphi, \psi, \ldots \) ) to denote S5 (resp. propositional) formulas. Due to the axiomatic of S5, subjective S5 formula are naturally interpreted over structures, which are simply nonempty subsets of \( \wp^X \), that is, nonempty sets of propositional assignments (we silently assume that structures are over the set of all propositional atoms under consideration). Intuitively, a structure represents a belief state, i.e., a set of assignments that the agent considers as candidates for being the actual state of the world; an S5 formula represents a set of such belief states.

A structure \( M \) is said to satisfy an epistemic atom \( K\varphi \) if all propositional assignments \( m \in M \) satisfy \( \varphi \) under the standard propositional semantics; \( M \) satisfies \( \Phi \land \psi \) (resp. \( \Phi \lor \psi \) ) if it satisfies \( \Phi \) and \( \psi \) (resp. \( \Phi \) or \( \psi \) ), and \( \neg \Phi \) if it does not satisfy \( \Phi \). Note that \( M \) satisfies \( \neg K\varphi \) if it contains at least one propositional countermodel of \( \varphi \) (intuitively, the agent does not know \( \varphi \) if \( \varphi \) is false in at least one state which may be the actual one), and that this is different from satisfying \( \neg \Phi \).

We write \( M \models \Phi \) if the structure \( M \) satisfies the subjective S5 formula \( \Phi \); it is a model of \( \Phi \), and we denote by \( \text{Mod}(\Phi) \) the set of models of \( \Phi \). When \( \text{Mod}(\Phi) = \text{Mod}(\psi) \), \( \Phi \) and \( \psi \) represent the same set of belief states; we call them logically equivalent, written \( \Phi \equiv \psi \). When \( \text{Mod}(\Phi) \subseteq \text{Mod}(\psi) \), we say that \( \Phi \) entails \( \psi \), written \( \Phi \models \psi \). An formula is tautological if all structures satisfy it. Notably useful in planning are positive S5 formulas, i.e., formulas equivalent to some \( \lor_i K\varphi_i \); it can be shown that these are exactly the formulas whose model set is closed by taking (nonempty) subsets.

**Propositional languages** A propositional formula (over \( X \)) is in the NNF (Negation Normal Form) language if it is a combination by \( \lor \) and \( \land \) of propositional literals of the form \( x \lor x' \) (\( x \in X \)). Identical subformulas are shared in NNF formulas: they are not trees but rather directed acyclic graphs (DAGs), and the size \( |\varphi| \) of a formula \( \varphi \) is thus its number of nodes. A term (resp. clause) is a conjunction (resp. disjunction) of literals. A formula is in disjunctive (resp. conjunctive) normal form if it is a disjunction of terms (resp. a conjunction of clauses); the corresponding language is called DNF (resp. CNF). Conditioning a formula \( \varphi \) by a literal \( \ell \) is, intuitively, deciding on the value of the corresponding atom; it can be done syntactically by replacing every instance of \( \ell \) (resp. \( \ell \)) by \( \top \) (resp. \( \bot \)). The result is denoted by \( \varphi|_{\ell} \). We also write \( M|_{\ell} \) for the structure obtained from a structure \( M \) by keeping only the assignments satisfying \( \ell \), then removing \( \ell \) from them.

**S5-DNF** Bienvenu et al. [2010] give the first study of effective representations for S5 formulas, notably introducing the following parameterized language:

**Definition 2.** Let \( L, L' \) be two (propositional) sublanguages of NNF; an S5 formula is in \( s\text{-S5-DNF}_{L,L'} \) if it is a disjunction of terms, each consisting of at most one positive epistemic literal \( K\varphi \) and of an arbitrary number of negative epistemic literals \( \neg K\psi \), where \( \varphi \) is in \( L \) and each \( \psi \) is in \( L' \).

They study the properties of \( s\text{-S5-DNF}_{L,L'} \) in a general setting, in terms of the properties of \( L \) and \( L' \); they also focus on a specific instantiation, \( s\text{-S5-DNF}_{\text{DAG,CNF}} \) (we denote it by EDNF for short), which turns out to have good properties for the operations involved in planning. Another natural instantiation is \( s\text{-S5-DNF}_{\text{OBDD,OBDD}} \) (denoted here by EBDDB); Bienvenu et al. [2010] do not explicitly consider it, but most of their general results directly apply to it. Figure 1 (left) gives an EBDDB for \( (K(x_1 \lor x_3) \land \neg K(x_2 \lor x_3)) \lor (\neg K(x_2 \lor x_3) \land K(x_2)) \).

**3 Epistemic Splitting Diagrams**

We now introduce a new language, written ESD, for representing subjective S5 formulas. As pointed out by Bienvenu et al. [2010, Example 15], the Shannon expansion cannot be used for S5 formulas at the epistemic level, but we introduce the related notion of splitting. Intuitively, a split over a propositional atom \( x \) divides a structure \( M \) into \( M'_1 \) and \( M'_2 \). For instance, with \( M_1 = \{x_2, x_3\} \) and \( M_2 = \{x_2, x_3\} \), \( \text{split}(x_1, M_1, M_2) \) represents the structure \( \{x_1, x_2, x_3, x_1, x_2, x_3, x_1, x_2, x_3\} \). More generally, we allow splits to represent sets of structures of a specific form: \( \text{split}(x, M, M') \) represents the set \( \{\text{split}(x, M, M') \mid M \in \mathcal{M}, M' \in \mathcal{M}'\} \).

As splits are not enough for obtaining a complete language, ESD also uses the \( \land \) connective. Moreover, since the model set of an S5-formula contains structures, which are themselves sets, we have four constant formulas: the usual constants \( \bot \) and \( \top \), the empty set and the whole set of structures.
Definition 3. Epistemic splitting diagrams (ESDs) are defined inductively as follows:

- \( \top, \bot, \land, \lor, \Delta \) are ESDs;
- if \( \Phi_1 \) and \( \Phi_2 \) are ESDs, then \( \text{spl}(x, \Phi_1, \Phi_2) \) is an ESD;
- if \( \Phi_1, \ldots, \Phi_n \) are ESDs, then \( \bigvee_{i=1}^n \Phi_i \) is an ESD.

A structure \( M \) satisfies an ESD \( \Phi \), denoted by \( M \models \Phi \), if either (i) \( \Phi \) is \( \bot \) (resp. \( \land \)) and \( M \) is not \( \emptyset \) (resp. is \( \emptyset \)); or (ii) \( \Phi \) is \( \text{spl}(x, \Phi_1, \Phi_2) \) and \( M \models \Phi_1 \) and \( M \models \Phi_2 \) hold; or (iii) \( \Phi \) is \( \bigvee_{i=1}^n \Phi_i \) and \( M \) satisfies \( \Phi_i \) for at least one \( i \in \{1, \ldots, n\} \). Additionally, \( M \) always satisfies \( \top \) and never satisfies \( \bot \).

Recall that S5 formulas are interpreted over nonempty structures. However, for ease of exposition we allow the empty structure, \( M_0 \), as a model of some ESDs. This is harmless since \( \Phi \land \lor \) has the same models as \( \Phi \) except for \( M_0 \), and can be computed efficiently (Proposition 14).

Figure 1 (right) gives an example of an ESD: the left child of its root is satisfied exactly by the structures \( M \) such that (i) \( M \models x_1 \) is empty or contains an assignment satisfying \( x_2 \), and (ii) \( M \models \top \) contains an assignment satisfying \( x_3 \). Let us build an ESD \( \Phi \) that is explicitly ordered. First, the third path from the left, \( (\lor, x_1, \lor, x_2, \lor) \), is explicitly ordered with respect to \( x_1 < x_2 < x_3 \), but the fourth one, \( (\lor, x_1, \lor, x_3, \lor) \), is not \( (x_2 \neq \text{missing}) \). Hence the ESD is not explicitly ordered.

For instance, on the ESD of Figure 1 (right), each node is satisfied by all structures \( M \) that contain \( x_i \). An ESD \( \Phi \) is said to be explicitly \( \prec \)-ordered if it is \( \prec \)-ordered and for each path \( P \) from the root to a leaf, the set of atoms appearing along \( P \) is \( \{x_{i_1}, x_{i_2}, \ldots, x_{i_n}\} \) for some \( i \in \{1, \ldots, n\} \).

Definition 5. Given a formula \( \phi \) and an ESD \( \Phi \), one can build in linear \( \prec \)-time an ordered ESD (resp. an explicitly ordered ESD) logically equivalent to \( K \Phi \lor \Delta \) or to \( \neg K \Phi \).

Proof. Let us build an ESD \( \Phi \) from \( \phi \) by replacing the \( \bot \) leaf by \( \Delta \) and each node \( \text{ite}(x, \Phi_1, \Phi_2) \) by \( \text{spl}(x, \Phi_1, \Phi_2) \), with \( \Phi_1, \Phi_2 \) obtained recursively from \( \phi \). Then we can show by induction that \( \Phi \) is equivalent to \( K \Phi \lor \Delta \) because (i) by replacing \( \bot \) by \( \Delta \) we prevent any countermodel of \( \phi \) to be in a satisfying structure, and (ii) by keeping the \( \top \) leaf we allow any model of \( \phi \) to be or not to be in a satisfying structure. The result follows from \( M \models K \Phi \lor \Delta \iff M \subseteq \text{Mod}(\phi) \).

For \( \neg K \Phi \), we build an ESD \( \Psi \) from \( \phi \) by replacing \( \bot \) by \( \lor \), \( \top \) by \( \bot \), and each node \( \text{ite}(x, \Phi_1, \Phi_2) \) by \( \text{spl}(x, \Phi_1, \Phi_2) \), with \( \Phi_1, \Phi_2 \) obtained recursively. An easy induction shows that a structure \( M \) satisfies \( \Psi \) exactly if it contains at least one countermodel of \( \phi \), hence \( \Psi \equiv \neg K \Phi \).
Table 1: Complexity of operations; names come from Bienvenu et al. [2010] and the KC literature. Symbols $\lor$, $\land$, $\neg$ resp. mean “polytime”, “not polytime”, and “not polytime if $P \neq NP$”; $\lor_E$ means “polytime if the formula is explicitly ordered, otherwise unknown”. Brackets refer to propositions here or in Bienvenu et al. [2010].

<table>
<thead>
<tr>
<th>Query</th>
<th>ESD</th>
<th>BBDD</th>
<th>EDNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>$\lor [9]$</td>
<td>$\lor [19]$</td>
<td>$\lor [19]$</td>
</tr>
<tr>
<td>NOT</td>
<td>$\neg$</td>
<td>$\neg$</td>
<td>$\neg$</td>
</tr>
<tr>
<td>IMPL</td>
<td>$\land [12]$</td>
<td>$\land [12]$</td>
<td>$\land [12]$</td>
</tr>
<tr>
<td>EQUI</td>
<td>$\equiv [10]$</td>
<td>$\equiv [10]$</td>
<td>$\equiv [10]$</td>
</tr>
</tbody>
</table>

Observe that both constructions do not require $\phi$ to be a reduced OBDD. Hence, the resulting ESD is explicitly ordered if $\phi$ is first made explicitly ordered, by recursively replacing all nodes $\text{ite}(x_i, \psi, \psi)$, where $\psi = \text{ite}(x_k, \phi_1, \phi_2)$, by $\text{ite}(x_i, \text{ite}(x_k, \psi, \psi), \psi)$, as long as there is a $j$ such that $x_i < x_j < x_k$ holds (and dually for the other child). This clearly increases the size of $\phi$ by a factor $|\text{Var}(\phi)|$ at most.

Moreover, an interesting feature of ESDs is that they can efficiently represent “only-know” atoms. Recall that $\text{Op}$ is satisfied by a single structure, namely $\text{Mod}(\phi)$: the agent knows $\phi$, but does not know more [Levesque, 1990]. Mixing $K$ and O modalities requires specific inference rules, while representing $\text{Op}$ using $K$ modalities requires the conjunction $K\phi \land \bigwedge_{i=1}^m \neg K\psi_i$ (in general, exponentially long). So the ability to represent $\text{Op}$ naturally is unique to ESD.

**Proposition 7.** Given an OBDD $\phi$, one can build an explicitly ordered ESD logically equivalent to $\text{Op} \phi$ in quadratic time.

**Proof.** We first make $\phi$ fully explicit: for each path in $\phi$, we proceed as in Proposition 6, but also until the leaves are reached (e.g., we recursively replace $\text{ite}(x_i, \psi, \psi)$ by $\text{ite}(x_i, \text{ite}(x_k, \psi, \psi), \psi)$). Then we replace $\bot$ by $\Delta$, $\top$ by $\forall$, and each $\text{ite}(x, \phi_1, \phi_2)$ by $\text{spl}(x, \phi_1, \phi_2)$, with $\phi_1, \phi_2$ obtained recursively. The resulting ESD is satisfied exactly by $\text{Mod}(\phi)$ because $\Delta$ prevents the countermodels of $\phi$ to be in a satisfying structure, and $\forall$ forces its models to be in.

Observing that any formula $\phi$ is equivalent to $\forall_{M, \varphi} \text{Op}_M \varphi$, where $\varphi_M$ is an OBDD with $\text{Mod}(\varphi) = M$, we easily deduce:

**Proposition 8.** ESD is complete for subjective S5: for any subjective S5 formula $\phi$, there exists an ESD $\psi$ with $\psi \equiv \phi$.

### 5 Queries and Transformations

With EDNF, BBDD, and ESD, we have three available languages for representing subjective S5 formulas. Before comparing their ability to represent formulas compactly (Section 6), we study how efficiently they support queries (i.e., reasoning tasks) and transformations; this is summarized in Table 1 to allow for easy comparison. Most results for EDNF and BBDD come from Bienvenu et al. [2010], hence we focus on ESD. The first result is easy, so we omit the proof (simply recall from Section 3 that $\bot$ and $\Delta$ have unique reduced ESDs).

**Proposition 9.** Given a formula $\phi$ in either EDNF, BBDD or ESD, it can be decided in polynomial time whether $\phi$ is satisfiable, and whether it is satisfied by a structure $M$ given in extension (i.e., as a set of propositional assignments).

**Proposition 10.** Given a satisfiable formula in either EDNF, BBDD or ESD, a structure satisfying it can be computed in polynomial (in particular, a poly-size model always exists).

**Proof.** For EDNF (resp. BBDD), we choose a satisfiable term $K\phi \land \bigwedge_{i=1}^m \neg K\psi_i$, and we build $M$ by taking one model of $\phi \land \neg \psi_i$ for each $i$, which is polytime since $\phi$ is in DNF and $\psi_i$ in CNF (resp. since both are OBDDs). For ESD, at each $\lor$-node we select a satisfiable child, and at each node $\text{spl}(x, \phi_1, \phi_2)$, we build $M$ from $M_{\phi_1}$ and $M_{\phi_2}$, obtained recursively.

We now turn to validity and entailment checking:

**Proposition 11.** Given a formula $\phi$ in ESD, it is coNP-hard to decide whether $\phi$ is tautological, and to decide whether $K\psi \models \phi$ holds for $\psi$ in CNF or OBDD.

**Proof.** Let $\phi = \bigwedge_{i\in I} c_i$ be a propositional CNF, where each $c_i$ is a clause, and let $\Phi$ be the ESD $\Delta \lor \bigvee_{i\in I} \Phi_i$, where $\Phi_i$ is equivalent to $\neg Kc_i$. Since $\Phi$ can be built in polytime (Prop. 6), and it can be shown that $\Phi$ is tautological if and only if $\phi$ is unsatisfiable, we get the first statement. For the second one, remark that $\Phi$ is tautological if and only if $K\top$ entails $\phi$.

**Proposition 12.** For any fixed $k$, given an explicitly ordered ESD $\Phi$ and a disjunction $\Psi$ of at most $k$ atoms $K\phi_i$ or $\neg K\phi_i$, where all $\phi_i$’s are OBDDs, it is polytime to decide $\Phi \models \Psi$.

**Proof.** Since it is polytime to compute the negation of the atoms in $\Psi$ (Prop. 6), and since bounded conjunction and satisfiability are polytime on ESD (Prop. 14 and 9), we can decide efficiently whether $\Phi \models \neg \Psi$ is unsatisfiable.

However, we conjecture that unbounded clausal entailment (CE) is hard on ESD. Note that BBDD supports bounded CE (similar proof as Prop. 12); EDNF supports unbounded CE for another representation of atoms [Bienvenu et al., 2010].

We finally consider combinations and transformations.

**Proposition 13.** Given $k$ ESDs $\Phi_1, \ldots, \Phi_k$, an ESD equivalent to $\bigwedge_{i=1}^k \Phi_i$ can be computed in linear time, but an ESD of size polynomial in $\sum_{i=1}^k |\Phi_i|$ and equivalent to $\bigwedge_{i=1}^k \Phi_i$ does not always exist. Also, given an ESD $\Phi$, an ESD equivalent to $\neg \Phi$ and of size polynomial in $|\Phi|$ does not always exist.

**Proof sketch.** This is clear for disjunction: ESD allows the $\lor$ connective. For conjunction, let $\Phi$ be an ESD equivalent to $K(x_i = y_i) \lor K(x_i \neq y_i)$. Clearly, $\bigwedge_{i=1}^k \Phi_i$ is linear in $k$; yet it can be shown that the smallest ESD equivalent to $\bigwedge_{i=1}^k \Phi_i$ has size exponential in $k$. Finally, since $\bigwedge_{i=1}^k \Phi_i \equiv \neg \bigwedge_{i=1}^k \neg \Phi_i$, if negation were poly-size, conjunction also would.

However, bounded conjunction can be computed efficiently on ESDs that are explicitly ordered:

**Proposition 14.** Given two explicitly ordered ESDs $\Phi_1, \Phi_2$, an explicitly ordered ESD $\Phi$ for $\Phi_1 \land \Phi_2$ can be computed in quadratic time.

**Proof sketch.** One can design an algorithm akin to “apply” on OBDDs [Bryant, 1986], relying on rules (i) $|\Phi_1 \lor \Phi_2| \leq |\Phi_1 \land \Phi_2| \lor |\Phi_2 \land \Phi_2|$ and (ii) $\text{spl}(x, \Phi_1, \Phi_2) \land \text{spl}(x, \Phi_3, \Phi_4) \leq \text{spl}(x, \Phi_1 \land \Phi_3, \Phi_2 \land \Phi_4)$ (explicit ordering guarantees that the current split atom is the same).
The last transformation we consider is forgetting. Given an atom \( x \) and a structure \( M, \text{Fo}(x,M) \) is defined to be the structure \( M[x]_1 \cup M[x]_2 \). For a subjective formula \( \Phi, \text{Fo}(x, \Phi) \) is any formula \( \Psi \) satisfying \( \text{Mod}(\Psi) = \{ \text{Fo}(x,M) \mid M \in \text{Mod}(\Phi) \} \); this is naturally extended to forgetting sets of variables. Forgetting turns out to be polytime for explicitly ordered ESDs, but only when restricted to a bounded number of atoms.

**Proposition 15.** Given an explicitly ordered ESD \( \Phi \) and a propositional atom \( x \), an ESD for \( \text{Fo}(x, \Phi) \) can be computed in polytime. However, given a set \( Y \), it is not guaranteed that there is a polysize ESD for \( \text{Fo}(Y, \Phi) \).

**Proof.** Clearly, forgetting distributes over \( \lor \). Now let \( \Phi = \text{spl}(y, \Phi_1, \Phi_2) \). If \( x > y \), then \( \Phi = \text{spl}(x, \Phi_1, \Phi_2) \) is appropriate, and \( x < y \) cannot occur because \( \Phi \) is explicitly ordered, so let \( \Phi = \text{spl}(x, \Phi_1, \Phi_2) \). It follows from the definitions that \( \text{Fo}(x, \Phi) \) is equivalent to \( \Psi = \Phi_1 \lor \Phi_2 \) defined by \( \text{Mod}(\Psi) = \{ M_1 \cup M_2 \mid M_1 \models \Phi_1, M_2 \models \Phi_2 \} \). Finally, it can be shown that binary \( \lor \) can be applied efficiently on explicitly ordered ESDs, using an algorithm similar to the one sketched above for binary \( \land \). Now the negative result can be lifted from OBDD, by considering the case \( \text{Fo}(Y, K \phi) \equiv K(\text{Fo}(Y, \phi)) \).

## 6 Succinctness

We now turn to the comparison of the three languages with respect to their ability to represent S5 formulas compactly.

**Definition 16.** A language \( L_1 \) is at least as succinct as another language \( L_2 \), denoted by \( L_1 \leq_{s} L_2 \), if and only if there exists a polynomial \( P \) verifying that for any formula \( \Phi_2 \) in \( L_2 \), there exists an equivalent formula in \( L_1 \) such that \( |\Phi_1| \leq P(|\Phi_2|) \).

The succinctness relation is a preorder; we write \( L_1 \nleq_{s} L_2 \) if both \( L_1 \nleq_{s} L_2 \) and \( L_2 \nleq_{s} L_1 \) hold, that is, if the two languages are incomparable with respect to succinctness. The following proposition shows that it is the case for two of our three languages when restricted to positive epistemic formulas (we denote by \( L^+ \) the language \( L \) restricted to positive formulas).

**Proposition 17.** \( \text{EDNF}^+ \nleq_{s} \text{EBDD}^+ \) and \( \text{EDNF}^+ \nleq_{s} \text{EBDD}^+ \) hold.

**Proof sketch.** We first show \( \text{EBDD}^+ \nleq_{s} \text{EDNF}^+ \). Let \( (\phi_n)_n \) be a family of DNFs such that for no polynomial \( P \) does there exist a family of OBDDs \( (\psi_n)_n \) with \( \forall n, \psi_n \equiv \phi_n \) and \( |\psi_n| \leq P(|\phi_n|) \) (such a family exists since OBDD \( \nleq_{s} \text{DNF} \), see Darwiche & Marquis [2002]). Consider the family \( (K\phi_n)_n \) of formulas in \( \text{EDNF}^+ \) (actually, of epistemic atoms). It can be shown that the smallest representations of \( K\phi_n \) in \( \text{EBDD} \) are of the form \( K\psi_n \) for some OBDD \( \psi_n \equiv \phi_n \). Since by assumption \( \psi_n \) is exponentially larger than \( \phi_n \), we indeed get \( \text{EBDD}^+ \nleq_{s} \text{EDNF}^+ \).

We can show \( \text{EDNF}^+ \nleq_{s} \text{EBDD}^+ \) similarly, using OBDDs which have no equivalent polytime DNFs. Now, for EDNF, using the construction of Prop. 6 we can show that a smallest ESD for \( K \phi \) (\( \phi \in \text{OBDD} \)) has essentially the same size as \( \phi \), hence the proof of \( \text{EDNF}^+ \nleq_{s} \text{EBDD}^+ \) works for \( \text{EDNF}^+ \nleq_{s} \text{EBDD}^+ \).

As a corollary, we get \( \text{ESD} \nleq_{s} \text{EDNF} \) (since if any ESD could be turned into a polysize EDNF, it would be the case in particular for positive formulas, and conversely), and we recover \( \text{EBDD} \nleq_{s} \text{EDNF} \) [Bienvenu et al., 2010, Prop. 17].

We are thus left with comparing ESD and EDNN. We show that they are incomparable in general, but that ESD is strictly more succinct on positive formulas.

**Proposition 18.** It holds that \( \text{ESD} \nleq_{s} \text{EDNN} \).

**Proof sketch.** Let \( \Phi \) be the ESD \( \bigwedge_{i=1}^{n} \neg K(x_i), \) with \( x_i \) an OBDD for \( x \leftrightarrow x_i \), and assume \( \forall i, x < x_i \). The smallest ESD equivalent to \( \Phi \) is \( \bigvee_{i \in \{1, \ldots, n\}} \text{spl}(x; x_i \wedge x < x_i) \) (not proven here for space reasons), which is exponentially larger than \( \Phi \).

**Proposition 19.** \( \text{ESD}^+ \) is strictly more succinct than \( \text{EDNN}^+ \) (and, as a corollary, it holds that \( \text{ESD} \nleq_{s} \text{EDNN} \)).

**Proof.** A formula in \( \text{EDNN}^+ \) is simply a disjunction of positive atoms \( K\phi_i \), with each \( \phi_i \) an OBDD. From Proposition 6 and the fact that \( \lor \) is a connective in ESD, we get \( \text{ESD}^+ \nleq_{s} \text{EDNN}^+ \).

Now consider the family of formulas \( (\Phi_n)_{n}, \) with \( \Phi_n = \bigwedge_{i=1}^{n} (K(x_i) \lor K(x'_i)) \). It can be seen that the only ESD equivalent to \( \Phi_n \) is \( \bigvee_{i} (K(x_i)) \), where \( r \) ranges over all \( 2^n \) terms on \( x_1, \ldots, x_n \) (represented as OBDDs). Now, it can also be seen that the ESD of Figure 1 (middle) is equivalent to \( \Phi_n \) and has size linear in \( n \); hence \( \text{ESD}^+ \nleq_{s} \text{EDNN}^+ \). This in turn entails \( \text{ESD} \nleq_{s} \text{EBDD} \) and \( \text{EBDD} \nleq_{s} \text{ESD} \) with Proposition 18.

Interestingly, note that representing in \( s \text{-S5-\text{DINF}_{L^+}} \) the family \( (\phi_n)_n \) used in the previous proof requires exponential space for any choice of \( L, L' \), since the proof works at the epistemic level. This shows that even the language built as the union of all \( s \text{-S5-\text{DINF}_{L^+}} \) languages (over all languages \( L, L' \)) is not at least as succinct as ESD for representing positive epistemic formulas. This spatial efficiency of ESD does not hold only for positive epistemic formulas: as seen in Section 4, ESD can succinctly represent atoms of the form \( \text{O}\phi \), while \( s \text{-S5-\text{DINF}_{L^+}} \) cannot (whatever \( L, L' \)). This gives an example of formulas which are not positive and on which ESD is more succinct than \( s \text{-S5-\text{DINF}_{L^+}} \) as a whole.

Finally, the following result intuitively shows that while transforming a positive ESD into an equivalent ESD can require exponential space, it is not actually difficult.

**Proposition 20.** There exist a polynomial \( P \) and an algorithm transforming any formula \( \Phi \) in \( \text{ESD}^+ \) into the (unique) equivalent formula \( \Psi \) in \( \text{EDNN}^+ \) in time bounded by \( P(|\Phi|, |\Psi|) \).

**Proof.** The algorithm simply consists in “pushing the disjunctions upwards” in the ESD, i.e., replacing bottom-up each node \( \text{spl}(x; \Phi_1 \lor \Phi_2, \Phi_3) \) by \( \text{spl}(x; \Phi_1, \Phi_3) \lor \text{spl}(x; \Phi_2, \Phi_3) \) (and symmetrically for disjunctions in the other child). Clearly, this process converges to an equivalent ESD which is structurally equal to \( \Psi \). Since the size of the ESD only increases at each step, the process is output-polynomial. 

This gives an interesting perspective about the relation between the two languages: while \( \text{EDNN}^+ \) can support queries that \( \text{ESD}^+ \) does not support, this proposition guarantees that \( \text{ESD}^+ \) will only perform polynomially worse than \( \text{EDNN}^+ \) on these queries. All in all, positive formulas are generally more compact in ESD, and in the worst case we can always “uncompress” the formula and fall back on EBDD.
7 Experiments

To investigate the languages in practice, we ran experiments on randomly drawn scenarios inspired from planning. Our first set of experiments focused on positive formulas, by running offline progressions of belief states by actions test(ϕi). For each run, we drew m actions of the form test(ϕi) (i = 1,...,m), with ϕi a random (uniform, satisfiable) term of a given size t. Then, starting from Φ0 = ⊤, we iteratively computed the current set of belief states Φi, i = 1,...,m, by progressing Φi−1 through test(ϕi), that is, by computing Φi−1 ∨ (Kϕi ∨ ¬ϕi). For instance, with t = 3, a possible term was x1 ∧ x2 ∧ x3, yielding progression by K(x1 ∧ ¬x2 ∧ x3) ∨ K(x4 ∨ ¬x1 ∧ x2).

We ran experiments with a moderate and a larger number of variables (n = 15 and n = 30; recall that there are 2^{2^n} structures over n atoms!) with term sizes t = 1, 3, 7, and numbers of actions m = 1,...,18. For each tuple (n,t,m), we averaged the results over 100 runs. Figure 2 plots the size of the final set of belief states Φm, and the time taken for computing it iteratively from Φ0. It can be seen that ESD provides the most compact representations, especially for small terms: as terms get larger (e.g., t = 7), feedbacks Kϕi are more constrained and the set of belief states shrinks, masking the differences between ESDs and EBDDs. On the other hand, it can be seen that in practice, EDNF does not provide compact representations. For running time, the advantage of ESD over EBDD and EDNF is not so clear; the gain in compactness in ESD comes with some computational overhead in practice (notably, reduction operations).

We also experimented on entailment: at the end of each run, we decided Φm |= (Kϕi ∨ ¬ϕi) for all atoms xi. The results (not reported here for lack of space) show that all three languages are very efficient at this, even when Φm is large.

We performed a second set of experiments, with the same setup except that for each feedback Kϕi and K¬ϕi, a polarity was drawn uniformly: for instance, the i-th action could yield an epistemic progression by Kϕi ∨ K¬ϕi. Results (again not reported in detail) show that for this setting the most interesting language is EBDD, both in succinctness and computation time. Both EDNF and ESD are clearly worse, and EDNFs tend to be more compact but not more efficient than ESDs.

Finally, we experimented interleaving progression by test(ϕi)’s (with positive feedbacks) and progression by ontic actions similar to conditional STRIPS actions, such as ψ = (x1 ∧ x2 ∧ x3) ∨ (x1 ∧ x2 ∧ x4), which sets x2,x3 to ⊤ in states satisfying x1, and x1 to ⊤ and x3 to ⊥ in other states. Progressing a belief state Φi−1 by an ontic action ψ essentially consists in computing Φi−1 ∨ Kψ and forgetting all nonprimed atoms in the result. We thus aimed at measuring the efficiency of forgetting in all three languages. The results show that forgetting is cheap for all languages, and we observed the same trends as in the first set of experiments.

8 Conclusion

We introduced the language ESD of epistemic splitting diagrams for representing subjective S5 formulas. We investigated ESD and the known languages s-S5-DFN and s-S5-DFN and studied the ESD of EDD here, both from the viewpoint of knowledge compilation and with experiments on random scenarios inspired from contingent planning. This is to our knowledge the first empirical study on effective S5 representations, although work in planning addressed specific issues of representation [e.g. Hoffmann & Braffman, 2005].

Our theoretical and empirical results complement each other. On positive formulas, ESD is more succinct than EBDD, while supporting mostly the same queries and transformations; this was confirmed by the experiments. On the other hand, both are incomparable to EDNF for succinctness, yet in practice EDNFs are clearly less compact. Experiments also show that computations are heavier on ESD, partly balancing succinctness. The picture is different on general formulas, for which EBDD turns out to be very succinct and efficient.

A short-term perspective of this work is to revisit with efficient representations, and notably EBDD and ESD, the standard planning approaches that (explicitly or not) use S5 reasoning.
References


