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 monoagent knowledge [Fagin et al., 1995], allowing for statements such as $(\mathrm{K}p \vee \mathrm{K}\bar{p}) \wedge (\neg\mathrm{K}(p \wedge q))$, 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 $p \wedge q$ is true (it knows that $p \wedge q$ 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 nondeterministic 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 $p \wedge q$, and executes an ontic action *switch* which nondeterministically switches either the value of p or that of q, then the resulting belief state can be described by $(\bar{p} \wedge q) \vee (p \wedge \bar{q})$. If the agent then executes the epistemic action *test*, which indicates whether $p \wedge r$ is true, and receives the feedback that it is the case, then the resulting set can be described by $(p \wedge \bar{q} \wedge r)$.

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 $p \wedge q$, then the ontic action *switch* leads to $(\bar{p} \wedge q) \vee (p \wedge \bar{q})$ as above, but the epistemic action *test* leads to two possible belief states, depending on its feedback: either $(p \wedge \bar{q} \wedge r)$ as above, or $(\bar{p} \wedge q) \vee (p \wedge \bar{q} \wedge \bar{r})$ if feedback $\bar{p} \vee \bar{r}$ 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 $\mathrm{K}(p \wedge \bar{q} \wedge r) \vee \mathrm{K}\big((\bar{p} \wedge q) \vee (p \wedge \bar{q} \wedge \bar{r})\big)$ (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 $\texttt{s-S5-DNF}_{\texttt{DNF,CNF}}$ representation proposed by Bienvenu et al. [2010], as well as its natural variant $\texttt{s-S5-DNF}_{\texttt{OBDD,OBDD}}$. 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 $\neg, \vee, \wedge$ and the knowledge modality K. For instance, $(\mathrm{K}x_1 \wedge \neg\mathrm{K}(x_2 \vee \bar{x}_3)) \vee \neg\mathrm{K}(\bar{x}_1)$ is an S5 formula. We denote by $\mathrm{Var}(\Phi)$ the set of propositional atoms mentioned in a formula

Φ (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., $x_1 \wedge \mathrm{K}x_2$ 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 $\neg$, $\vee$ and $\wedge$, of *epistemic atoms* of the form $\mathrm{K}\varphi$, where each φ 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 formulas are naturally interpreted over *structures*, which are simply nonempty subsets of 2^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 $\mathrm{K}\varphi$ if all propositional assignments $m \in M$ satisfy φ under the standard propositional semantics; M satisfies $\Phi \wedge \Psi$ (resp. $\Phi \vee \Psi$) if it satisfies Φ and Ψ (resp. Φ or Ψ), and $\neg\Phi$ if it does not satisfy Φ. Note that M satisfies $\neg\mathrm{K}\phi$ if it contains at least one propositional countermodel of φ (intuitively, the agent does not know ϕ if ϕ is false in at least one state which may be the actual one), and that this is different from satisfying $\mathrm{K}\neg\varphi$.

We write $M \models \Phi$ if the structure M satisfies the subjective S5 formula Φ; it is a *model* of Φ, and we denote by $\mathrm{Mod}(\Phi)$ the set of models of Φ. When $\mathrm{Mod}(\Phi) = \mathrm{Mod}(\Psi)$, Φ and Ψ represent the same set of belief states; we call them *logically equivalent*, written $\Phi \equiv \Psi$. When $\mathrm{Mod}(\Phi) \subseteq \mathrm{Mod}(\Psi)$, we say that Φ *entails* Ψ, written $\Phi \models \Psi$. A formula is *tautological* if all structures satisfy it. Notably useful in planning are *positive S5 formulas*, i.e., formulas equivalent to some $\bigvee_i \mathrm{K}\varphi_i$; it can be shown that those 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 $\vee$ and $\wedge$ of *propositional literals* of the form x or $\bar{x}$ ($x \in X$). Identical subformulas are shared in NNF formulas: they are not trees but rather directed acyclic graphs (DAGs), and the *size* $|\varphi|$ of a formula φ 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 φ by a literal ℓ is, intuitively, deciding on the value of the corresponding atom; it can be done syntactically by replacing every instance of ℓ (resp. $\bar{\ell}$) 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 ℓ, then removing ℓ from them.

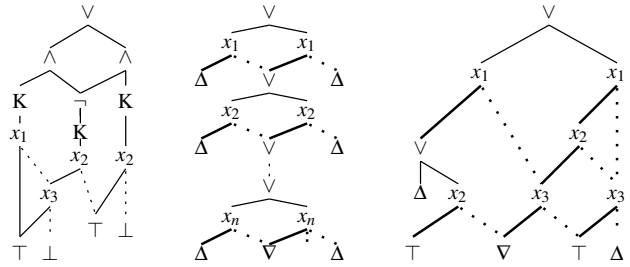

Figure 1: An EBDD (left) and two ESDs. Dots are to right children of ite's and spl's, and leaves are duplicated only for clarity.

A *Binary Decision Diagram* (BDD) is a constant or a formula of the form $(x \wedge N) \vee (\bar{x} \wedge N')$, where N, N' are BDDs; the latter is written $\mathsf{ite}(x, N, N')$. A BDD can be seen as a DAG over nodes labeled with atoms (e.g., on Figure 1, left, the DAG rooted at x_1 is a BDD). A BDD is an *ordered* BDD (OBDD) if atoms are encountered at most once and in the same order along all paths from the root to a leaf. It is well-known that any NNF φ can be represented as an OBDD (over any atom ordering), using the *Shannon expansion*.

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

They study the properties of $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ in a general setting, in terms of the properties of L and $\mathtt{L}'$; they also focus on a specific instantiation, $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{DNF},\mathtt{CNF}}$ (we denote it by EDNF for short), which turns out to have good properties for the operations involved in planning. Another natural instantiation is $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{OBDD},\mathtt{OBDD}}$ (denoted here by EBDD); Bienvenu et al. [2010] do not explicitly consider it, but most of their general results directly apply to it. Figure 1 (left) gives an EBDD for $(\mathrm{K}(x_1 \vee x_3) \wedge \neg\mathrm{K}(\bar{x}_2 \vee x_3)) \vee (\neg\mathrm{K}(\bar{x}_2 \vee x_3) \wedge \mathrm{K}(x_2))$.

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 $M|_x$ and $M|_{\bar{x}}$. For instance, with $M_1 = \{x_2x_3, \bar{x}_2\bar{x}_3\}$ and $M_2 = \{x_2x_3\}$, $\mathsf{spl}(x_1, M_1, M_2)$ represents the structure $\{x_1x_2x_3, x_1\bar{x}_2\bar{x}_3, \bar{x}_1x_2x_3\}$. More generally, we allow splits to represent *sets* of structures of a specific form: $\mathsf{spl}(x, \mathcal{M}, \mathcal{M}')$ represents the set $\{\mathsf{spl}(x, M, M') \mid M \in \mathcal{M}, M' \in \mathcal{M}'\}$.

As splits are not enough for obtaining a complete language, ESD also uses the $\vee$ 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 $\top$, which are satisfied by no structure and by all structures, respectively, and two new ones, Δ and ∇, respectively satisfied exactly by the empty structure, and by all nonempty structures. The two latter constants are notably used as children of splitting nodes; for instance, on Figure 1 (right), the ESD rooted at the bottom left x_2 node is satisfied by all structures which contain an assignment satisfying $\bar{x}_2$. Formally:

Definition 3. *Epistemic splitting diagrams* (ESDs) are defined inductively as follows:

- $\top$, ∇, $\bot$, and Δ are ESDs;
- if Φ_1 and Φ_2 are ESDs, then $\mathsf{spl}(x, \Phi_1, \Phi_2)$ 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 Φ, denoted by $M \models \Phi$, if either (i) Φ is ∇ (resp. Δ) and M is not $\varnothing$ (resp. is $\varnothing$); or (ii) Φ is $\mathsf{spl}(x, \Phi_1, \Phi_2)$ and $M|_x \models \Phi_1$ and $M|_{\bar{x}} \models \Phi_2$ hold; or (iii) Φ is $\bigvee_{i=1}^{n} \Phi_i$ and M satisfies Φ_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 $M_\varnothing$ as a model of some ESDs. This is harmless since $\Phi \wedge \nabla$ has the same models as Φ except for $M_\varnothing$, 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) $M|_{x_1}$ is empty or contains an assignment satisfying $\bar{x}_2$, and (ii) $M|_{\bar{x}_1}$ contains an assignment satisfying x_3. Remark that, like OBDDs, we view ESDs as DAGs, and we assume that identical subgraphs are systematically shared.

Definition 3 places no specific syntactic restriction on ESDs; it is however useful to consider *reduced* ESDs.

Definition 4. An ESD Φ is said to be *reduced* if none of the following rules applies to it:

- simplify using $(\bot \vee \Phi) \equiv \Phi$, $(\top \vee \Phi) \equiv \top$, $\mathsf{spl}(x, \Delta, \Delta) \equiv \Delta$, $\mathsf{spl}(x, \top, \top) \equiv \top$, $\mathsf{spl}(x, \Phi, \bot) \equiv \mathsf{spl}(x, \bot, \Phi) \equiv \bot$;
- replace $\mathsf{spl}(x, \Phi, \Psi_1) \vee \mathsf{spl}(x, \Phi, \Psi_2)$ by $\mathsf{spl}(x, \Phi, \Psi_1 \vee \Psi_2)$, and dually when it is right children that match;
- remove duplicates among children of $\vee$-nodes, and flatten $(\Phi_1 \vee \cdots \vee (\Psi_1 \vee \cdots \vee \Psi_k) \vee \cdots \vee \Phi_m)$ into $(\Phi_1 \vee \cdots \vee \Psi_1 \vee \cdots \vee \Psi_k \vee \cdots \vee \Phi_m)$, $\bigvee\{\Phi\}$ into Φ, and $\bigvee \varnothing$ into $\bot$;
- replace $(\Phi_1 \vee \cdots \vee \Delta \vee \cdots \vee \Phi_k)$ by $(\Phi_1 \vee \cdots \vee \Phi_k)$ if some Φ_i is satisfied by $M_\varnothing$;
- replace $(\Phi_1 \vee \cdots \vee \nabla \vee \cdots \vee \Phi_k)$ by $\top$ if some Φ_i is satisfied by $M_\varnothing$ and by ∇ otherwise.

It is easily seen that all these rules preserve logical equivalence, and can be enforced in linear time. Another important property (obtained by a simple structural induction), is that the only reduced ESD equivalent to $\bot$ (resp. to Δ) is $\bot$ itself (resp. Δ itself). Contrastingly, as is the case for EDNF and EBDD, there are several reduced ESDs equivalent to $\top$ or ∇. For instance (abusing notation), $\mathsf{spl}(x, \mathrm{K}y, \Delta) \vee \mathsf{spl}(x, \top, \nabla) \vee \mathsf{spl}(x, \neg \mathrm{K}y, \top)$ is reduced but logically equivalent to $\mathrm{K}(x \wedge y) \vee \neg \mathrm{K}x \vee \neg \mathrm{K}(\bar{x} \vee y)$, which is tautological.

As for OBDD, we can impose ESDs to be *ordered*. Given a total ordering $<$ on X, an ESD is said to be $<$-ordered if the propositional atoms appear in (strict) increasing order wrt $<$ along each path from the root to a leaf.

In this paper, we only consider reduced ordered ESDs. We write ESD for the language consisting of all ESDs which are reduced and ordered (leaving $<$ implicit). For instance, the ESDs on Figure 1 are reduced, and ordered wrt $x_1 < \cdots < x_n$.

ESDs share many features with OBDDs. An important difference is in the status of *stuttering* nodes. A node $\mathsf{ite}(x, \varphi, \varphi)$ can be eliminated from an OBDD (and replaced by φ), but the same is not true for ESDs: in general, we have $\mathsf{spl}(x, \Phi, \Phi) \not\models \Phi$. For instance, with $\Phi \equiv \mathrm{K}y \vee \mathrm{K}\bar{y}$, we have that $M = \{xy, \bar{x}\bar{y}\}$ satisfies $\mathsf{spl}(x, \Phi, \Phi)$ (since $M|_x = \{y\}$ satisfies $\mathrm{K}y$ and hence Φ, and $M|_{\bar{x}} = \{\bar{y}\}$ satisfies $\mathrm{K}\bar{y}$ and hence Φ), but M does not satisfy Φ (since in M both y and $\bar{y}$ are possible). Due to this, in order to be efficient, some transformations require that (reduced, ordered) ESDs have a specific form.

Definition 5. Let $<$ be a total order on X, with $x_1 < \cdots < x_n$. An ESD Φ is said to be *explicitly* $<$*-ordered* if it is $<$-ordered and for each path P from the root to a leaf, the set of atoms appearing along P is $\{x_1, x_2, \ldots, x_i\}$ for some $i \in \{1, \ldots, n\}$.

For instance, on the ESD of Figure 1 (right), the third path from the left, $(\vee, x_1, \vee, x_2, \nabla)$, is explicitly ordered with respect to $x_1 < x_2 < x_3$, but the fourth one, $(\vee, x_1, x_3, \nabla)$, is not (x_2 is missing). Hence the ESD is *not* explicitly ordered.

In the rest of the paper, we always consider OBDDs and ESDs ordered over the same (implicit) atom ordering. Unless specified, we do *not* require the ESDs to be *explicitly* ordered.

4 Compiling Epistemic Atoms into ESDs

Arguably, it is natural to specify formulas such as goals, initial states, etc., in the form of S5 logical formulas. Manipulating ESDs thus requires to first *compile* [Marquis, 2015] standard epistemic representations into the ESD language. We show in this section how to compile epistemic *atoms* as ESDs; Section 5 will show how to combine them using connectives.

Since compiling propositional formulas into OBDD is a well-studied problem [Bryant, 1992; Meinel & Theobald, 1998; Huang & Darwiche, 2005], we assume that epistemic atoms are in the form $\mathrm{K}\varphi$, where φ is an OBDD over the atom ordering that we want for the ESD. Building φ is hard, but once it is done, $\mathrm{K}\varphi$ can be obtained very efficiently:

Proposition 6. *Given a formula φ in* OBDD*, one can build in linear (resp. quadratic) time an ordered ESD (resp. an explicitly ordered ESD) logically equivalent to $\mathrm{K}\varphi \vee \Delta$ or to $\neg \mathrm{K}\varphi$.*

Proof. Let us build an ESD Φ from φ by replacing the $\bot$ leaf by Δ and each node $\mathsf{ite}(x, \varphi_1, \varphi_2)$ by $\mathsf{spl}(x, \Phi_1, \Phi_2)$, with Φ_1, Φ_2 obtained recursively from φ_1, φ_2. Then we can show by induction that Φ is equivalent to $\mathrm{K}\varphi \vee \Delta$ because (i) by replacing $\bot$ by Δ we prevent any countermodel of φ to be in a satisfying structure, and (ii) by keeping the $\top$ leaf we allow any model of φ to be or not to be in a satisfying structure. The result follows from $M \models \mathrm{K}\varphi \vee \Delta \iff M \subseteq \mathrm{Mod}(\varphi)$.

For $\neg \mathrm{K}\varphi$, we build an ESD Ψ from φ by replacing $\bot$ by ∇, $\top$ by $\bot$, and each node $\mathsf{ite}(x, \varphi_1, \varphi_2)$ by $\mathsf{spl}(x, \Psi_1, \top) \vee \mathsf{spl}(x, \top, \Psi_2)$, with Ψ_1, Ψ_2 obtained recursively. An easy induction shows that a structure M satisfies Ψ exactly if it contains at least one countermodel of φ, hence $\Psi \equiv \neg \mathrm{K}\varphi$.

Query	ESD	EBDD	EDNF
CO	$\surd$ [9]	$\surd$ [B19]	$\surd$ [B19]
VA, **IM**	$\circ$ [11]	$\circ$ [B18]	$\circ$ [B18]
EQ, **SE**	$\circ$ [11]	$\circ$ [B18]	$\circ$ [B18]
$\mathbf{MC}_e$	$\surd$ [9]	$\surd$ [9]	$\surd$ [9]
$\mathbf{BCE}_{\mathrm{BDD}}$	$\surd_E$ [12]	$\surd$ [12]	?
$\mathbf{MX}_e$	$\surd$ [10]	$\surd$ [10]	$\surd$ [10]

Transf.	ESD	EBDD	EDNF
$\vee\mathbf{C}$	$\surd$ [13]	$\surd$ [B20]	$\surd$ [B20]
$\wedge\mathbf{BC}$	$\surd_E$ [14]	$\surd$ [B20]	$\surd$ [B20]
$\wedge\mathbf{C}$	$\bullet$ [13]	$\bullet$ [B20]	$\bullet$ [B20]
$\neg\mathbf{C}$	$\bullet$ [13]	$\bullet$ [B20]	$\bullet$ [B20]
FO	$\bullet$ [15]	$\bullet$ [B21]	$\surd$ [B21]
SFO	$\surd_E$ [15]	$\surd$ [B21]	$\surd$ [B21]

Table 1: Complexity of operations; names come from Bienvenu et al. [2010] and the KC literature. Symbols $\surd, \bullet, \circ$ resp. mean "polytime", "not polytime", and "not polytime if $\mathsf{P} \neq \mathsf{NP}$"; $\surd_E$ means "polytime if the formula is explicitly ordered, otherwise unknown". Brackets refer to propositions here or in Bienvenu et al. [2010].

Observe that both constructions do not require φ to be a reduced OBDD. Hence, the resulting ESD is explicitly ordered if φ is first made explicitly ordered, by recursively replacing all nodes $\mathsf{ite}(x_i, \psi, \cdot)$, where $\psi = \mathsf{ite}(x_k, \varphi_1, \varphi_2)$, by $\mathsf{ite}(x_i, \mathsf{ite}(x_j, \psi, \psi), \cdot)$, 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 φ by a factor $|\mathrm{Var}(\varphi)|$ at most. □

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

Proposition 7. *Given an OBDD φ, one can build an explicitly ordered ESD logically equivalent to $\mathrm{O}\varphi$ in quadratic time.*

Proof. We first make φ fully explicit: for each path in φ, we proceed as in Proposition 6, but also until the leaves are reached (e.g., we recursively replace $\mathsf{ite}(x_i, \top, \cdot)$ by $\mathsf{ite}(x_i, \mathsf{ite}(x_j, \top, \top), \cdot)$). Then we replace $\bot$ by Δ, $\top$ by ∇, and each $\mathsf{ite}(x, \phi_1, \phi_2)$ by $\mathsf{spl}(x, \Phi_1, \Phi_2)$, with Φ_1, Φ_2 obtained recursively. The resulting ESD is satisfied exactly by $\mathrm{Mod}(\varphi)$ because Δ prevents the countermodels of φ to be in a satisfying structure, and ∇ forces its models to be in. □

Observing that any formula Φ is equivalent to $\bigvee_{M \models \Phi} \mathrm{O}\varphi_M$, where φ_M is an OBDD with $\mathrm{Mod}(\varphi) = M$, we easily deduce:

Proposition 8. ESD *is complete for subjective S5: for any subjective S5 formula Φ, there exists an ESD Ψ with $\Psi \equiv \Phi$.*

5 Queries and Transformations

With EDNF, EBDD, 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 EBDD 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 Δ have unique reduced ESDs).

Proposition 9. *Given a formula Φ in either* EDNF, EBDD *or* ESD*, it can be decided in polynomial time whether Φ 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, EBDD *or* ESD*, a structure satisfying it can be computed in polytime (in particular, a polysize model always exists).*

Proof. For EDNF (resp. EBDD), we choose a satisfiable term $\mathrm{K}\varphi \wedge \bigwedge_i \neg\mathrm{K}\psi_i$, and we build M by taking one model of $\varphi \wedge \neg\psi_i$ for each i, which is polytime since φ is in DNF and ψ_i in CNF (resp. since both are OBDDs). For ESD, at each $\vee$-node we select a satisfiable child, and at each node $\mathsf{spl}(x, \Phi_1, \Phi_2)$, we build M from $M|_x$ and $M|_{\bar{x}}$, obtained recursively. □

We now turn to validity and entailment checking:

Proposition 11. *Given a formula Φ in* ESD*, it is* coNP*-hard to decide whether Φ is tautological, and to decide whether $\mathrm{K}\psi \models \Phi$ holds for ψ in* NNF *or* OBDD*.*

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

Proposition 12. *For any fixed k, given an explicitly ordered ESD Φ and a disjunction Ψ of at most k atoms $\mathrm{K}\varphi_i$ or $\neg\mathrm{K}\varphi_i$, where all φ_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 Ψ (Prop. 6), and since bounded conjunction and satisfiability are polytime on ESD (Prop. 14 and 9), we can decide efficiently whether $\Phi \wedge \neg\Psi$ is unsatisfiable. □

However, we conjecture that *unbounded* clausal entailment (**CE**) is hard on ESD. Note that EBDD 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 $\bigvee_{i=1}^k \Phi_i$ 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 Φ, 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 $\vee$ connective. For conjunction, let Φ_i be an ESD equivalent to $\mathrm{K}(x_i = y_i) \vee \mathrm{K}(x_i \neq y_i)$. Clearly, $\sum_{i=1}^k |\Phi_i|$ 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 $\bigwedge_{i=1}^k \Phi_i \equiv \neg \bigvee_{i=1}^k \neg\Phi_i$, if negation were polysize, conjunction also would. □

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

Proposition 14. *Given two explicitly ordered ESDs Φ_1, Φ_2, an explicitly ordered ESD for $\Phi_1 \wedge \Phi_2$ can be computed in quadratic time.*

Proof sketch. One can design an algorithm akin to "*apply*" on OBDDs [Bryant, 1986], relying on rules (i) $(\Psi_1 \vee \Psi_2) \wedge \Psi \equiv (\Psi_1 \wedge \Psi) \vee (\Psi_2 \wedge \Psi)$ and (ii) $\mathsf{spl}(x, \Psi_1, \Psi_2) \wedge \mathsf{spl}(x, \Psi_3, \Psi_4) \equiv \mathsf{spl}(x, \Psi_1 \wedge \Psi_3, \Psi_2 \wedge \Psi_4)$ (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, $\mathrm{Fo}(x,M)$ is defined to be the structure $M|_x \cup M|_{\bar{x}}$. For a subjective formula Φ, $\mathrm{Fo}(x,\Phi)$ is any formula Ψ satisfying $\mathrm{Mod}(\Psi) = \{\mathrm{Fo}(x,M) \mid M \in \mathrm{Mod}(\Phi)\}$; 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 Φ and a propositional atom x, an ESD for $\mathrm{Fo}(x,\Phi)$ can be computed in polytime. However, given a* set *Y of atoms, it is not guaranteed that there is a polysize ESD for $\mathrm{Fo}(Y,\Phi)$.*

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

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 $\mathtt{L}_1$ is *at least as succinct* as another language $\mathtt{L}_2$, denoted by $\mathtt{L}_1 \leq_s \mathtt{L}_2$, if and only if there exists a polynomial P verifying that for any formula Φ_2 in $\mathtt{L}_2$, there exists an equivalent Φ_1 in $\mathtt{L}_1$ such that $|\Phi_1| \leq P(|\Phi_2|)$.

The succinctness relation is a preorder; we write $\mathtt{L}_1 \not\geq_s \mathtt{L}_2$ if both $\mathtt{L}_1 \not\leq_s \mathtt{L}_2$ and $\mathtt{L}_1 \not\geq_s \mathtt{L}_2$ hold, that is, if the two languages are incomparable with respect to succinctness. The following proposition shows that it is the case for two pairs of our three languages when restricted to positive epistemic formulas (we denote by $\mathtt{L}^+$ the language L restricted to positive formulas).

Proposition 17. $\mathtt{EDNF}^+ \not\geq_s \mathtt{EBDD}^+$ *and* $\mathtt{EDNF}^+ \not\geq_s \mathtt{ESD}^+$ *hold.*

Proof sketch. We first show $\mathtt{EBDD}^+ \not\leq_s \mathtt{EDNF}^+$. Let $(\varphi_n)_n$ be a family of DNFs such that for no polynomial P does there exist a family of OBDDs $(\psi_n)_n$ with $\forall n, \psi_n \equiv \varphi_n$ and $|\psi_n| \leq P(|\varphi_n|)$ (such a family exists since $\mathtt{OBDD} \not\geq_s \mathtt{DNF}$, see Darwiche & Marquis [2002]). Consider the family $(\mathrm{K}\varphi_n)_n$ of formulas in $\mathtt{EDNF}^+$ (actually, of epistemic atoms). It can be shown that the smallest representations of $\mathrm{K}\varphi_n$ in EBDD are of the form $\mathrm{K}\psi_n$ for some OBDD $\psi_n \equiv \varphi_n$. Since by assumption such ψ_n is exponentially larger than φ_n, we indeed get $\mathtt{EBDD}^+ \not\leq_s \mathtt{EDNF}^+$.

We can show $\mathtt{EDNF}^+ \not\leq_s \mathtt{EBDD}^+$ similarly, using OBDDs which have no equivalent polysize DNFs. Now, for ESD, using the construction of Prop. 6 we can show that a smallest ESD for $\mathrm{K}\varphi$ ($\varphi \in \mathtt{OBDD}$) has essentially the same size as φ, hence the proof of $\mathtt{EDNF}^+ \not\geq_s \mathtt{EBDD}^+$ works for $\mathtt{EDNF}^+ \not\geq_s \mathtt{ESD}^+$. □

As a corollary, we get $\mathtt{ESD} \not\geq_s \mathtt{EDNF}$ (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 $\mathtt{EBDD} \not\geq_s \mathtt{EDNF}$ [Bienvenu et al., 2010, Prop. 17].

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

Proposition 18. *It holds that* $\mathtt{ESD} \not\leq_s \mathtt{EBDD}$.

Proof sketch. Let Φ be the EBDD $\bigwedge_{i=1}^n \neg\mathrm{K}(\varphi_i)$, with φ_i an OBDD for $x \leftrightarrow x_i$, and assume $\forall i, x < x_i$. The smallest ESD equivalent to Φ is $\bigvee_{I \subseteq \{1,\ldots,n\}} \mathsf{spl}(x, \bigwedge_{i\in I} \neg\mathrm{K}\bar{x}_i, \bigwedge_{i\notin I} \neg\mathrm{K}x_i)$ (not proven here for space reasons), which is exponentially larger than Φ. □

Proposition 19. $\mathtt{ESD}^+$ *is strictly more succinct than* $\mathtt{EBDD}^+$ *(and, as a corollary, it holds that* $\mathtt{EBDD} \not\geq_s \mathtt{ESD}$*).*

Proof. A formula in $\mathtt{EBDD}^+$ is simply a disjunction of positive atoms $\mathrm{K}\phi_i$, with each ϕ_i an OBDD. From Proposition 6 and the fact that $\vee$ is a connective in ESD, we get $\mathtt{ESD}^+ \leq_s \mathtt{EBDD}^+$.

Now consider the family of formulas $(\Phi_n)_n$, with $\Phi_n = \bigwedge_{i=1}^n (\mathrm{K}x_i \vee \mathrm{K}\bar{x}_i)$. It can be seen that the only EBDD equivalent to Φ_n is $\bigvee_t (\mathrm{K}t)$, where t 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 Φ_n and has size linear in n; hence $\mathtt{ESD}^+ \not\geq_s \mathtt{EBDD}^+$. This in turn entails $\mathtt{ESD} \not\geq_s \mathtt{EBDD}$, and $\mathtt{EBDD} \not\geq_s \mathtt{ESD}$ with Proposition 18. □

Interestingly, note that representing in $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ the family $(\Phi_n)_n$ used in the previous proof requires exponential space for any choice of $\mathtt{L},\mathtt{L}'$, since the proof works at the epistemic level. This shows that even the language built as the union of all $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ languages (over all languages $\mathtt{L},\mathtt{L}'$) 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 $\mathrm{O}\varphi$, while $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ cannot (whatever $\mathtt{L},\mathtt{L}'$). This gives an example of formulas which are *not* positive and on which ESD is more succinct than $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ as a whole.

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

Proposition 20. *There exist a polynomial P and an algorithm transforming any formula Φ in* $\mathtt{ESD}^+$ *into the (unique) equivalent formula Ψ in* $\mathtt{EBDD}^+$ *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 $\mathsf{spl}(x,\Phi_1 \vee \Phi_2,\Phi_3)$ by $\mathsf{spl}(x,\Phi_1,\Phi_3) \vee \mathsf{spl}(x,\Phi_2,\Phi_3)$ (and symmetrically for disjunctions in the other child). Clearly, this process converges to an equivalent ESD which is structurally equal to Ψ. 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 $\mathtt{EBDD}^+$ can support queries that $\mathtt{ESD}^+$ does not support, this proposition guarantees that $\mathtt{ESD}^+$ will only perform polynomially worse than $\mathtt{EBDD}^+$ 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.

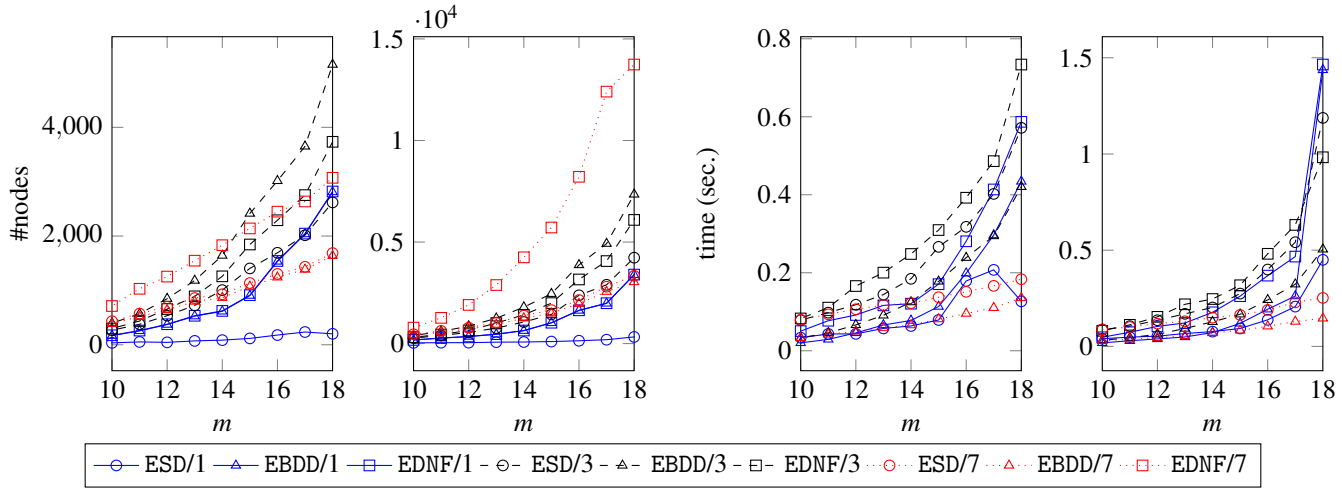

Figure 2: Results for positive feedbacks. From left to right: size for $n = 15$; size for $n = 30$; time for $n = 15$; time for $n = 30$. Each curve corresponds to a pair language/term size. We do not report time for EDNF with $t = 7$, which was very bad.

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 $test(\varphi_i)$. For each run, we drew m actions of the form $test(\varphi_i)$ ($i = 1,\ldots,m$), with φ_i a random (uniform, satisfiable) term of a given size t. Then, starting from $\Phi_0 = \top$, we iteratively computed the current set of belief states Φ_i, $i = 1,\ldots,m$, by progressing Φ_{i-1} through $test(\varphi_i)$, that is, by computing $\Phi_{i-1} \wedge (\mathrm{K}\varphi_i \vee \mathrm{K}\neg\varphi_i)$. For instance, with $t = 3$, a possible term was $x_4 \wedge \bar{x}_1 \wedge x_2$, yielding progression by $\mathrm{K}(x_4 \wedge \bar{x}_1 \wedge x_2) \vee \mathrm{K}(\bar{x}_4 \vee x_1 \vee \bar{x}_2)$.

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 $m = 1,\ldots,18$. 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 Φ_m, and the time taken for computing it iteratively from Φ_0. It can be seen that ESD provides the most compact representations, especially for small terms: as terms get larger (e.g., $t = 7$), feedbacks $\mathrm{K}\varphi_i$ are most 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 $\Phi_m \models (\mathrm{K}x_i \vee \mathrm{K}\neg x_i)$ for all atoms x_i. The results (not reported here for lack of space) show that all three languages are very efficient at this, even when Φ_m is large.

We performed a second set of experiments, with the same setup except that for each feedback $\mathrm{K}\varphi_i$ and $\mathrm{K}\neg\varphi_i$, a polarity was drawn uniformly: for instance, the i-th action could yield an epistemic progression by $\mathrm{K}\varphi_i \vee \neg\mathrm{K}\neg\varphi_i$. 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 $test(\varphi_i)$'s (with positive feedbacks) and progression by *ontic* actions similar to conditional STRIPS actions, such as $\psi = (x_1 \wedge x_2' \wedge x_3') \vee (\bar{x}_1 \wedge x_1' \wedge \bar{x}_3')$, which sets x_2, x_3 to $\top$ in states satisfying x_1, and x_1 to $\top$ and x_3 to $\bot$ in other states. Progressing a belief state Φ_{i-1} by an ontic action ψ essentially consists in computing $\Phi_{i-1} \wedge \mathrm{K}\psi$ 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-DNF$_{\mathrm{DNF,CNF}}$ and s-S5-DNF$_{\mathrm{OBDD,OBDD}}$ (called EDNF and EBDD 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 & Brafman, 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

[Bienvenu et al., 2010] Meghyn Bienvenu, Hélène Fargier, and Pierre Marquis. Knowledge compilation in the modal logic S5. In *Proc. 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[Bonet & Geffner, 2014] Blai Bonet and Hector Geffner. Belief tracking for planning with sensing: Width, complexity and approximations. *J. Artificial Intelligence Research*, 2014.

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[Bryant, 1992] Randal E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[Darwiche & Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artificial Intelligence Research*, 17:229–264, 2002.

[Fagin et al., 1995] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[Herzig et al., 2003] Andreas Herzig, Jérôme Lang, and Pierre Marquis. Action representation and partially observable planning using epistemic logic. In *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1067–1072, 2003.

[Hoffmann & Brafman, 2005] Jörg Hoffmann and Ronen I. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proc. 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 71–80, 2005.

[Huang & Darwiche, 2005] Jinbo Huang and Adnan Darwiche. DPLL with a trace: From SAT to knowledge compilation. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 156–162, 2005.

[Iocchi et al., 2004] Luca Iocchi, Thomas Lukasiewicz, Daniele Nardi, and Riccardo Rosati. Reasoning about actions with sensing under qualitative and probabilistic uncertainty. In *Proc. 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 818–822. IOS Press, 2004.

[Levesque, 1990] Hector J. Levesque. All I know: A study in autoepistemic logic. *Artificial Intelligence*, 42(2-3):263–309, 1990.

[Marquis, 2015] Pierre Marquis. Compile! In *Proc. 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 4112–4118, 2015.

[Meinel & Theobald, 1998] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications*. Springer, 1998.

[Petrick & Bacchus, 2004] Ronald P. A. Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 2–11, 2004.