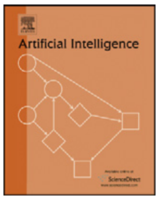
Artificial Intelligence 174 (2010) 585–596

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



# Information loss in knowledge compilation: A comparison of Boolean envelopes

Peter Schachte, Harald Søndergaard [*], Leigh Whiting, Kevin Henshall

*Department of Computer Science and Software Engineering, The University of Melbourne, Victoria 3010, Australia*

ARTICLE INFO

*Article history:*
Received 18 March 2009
Received in revised form 14 March 2010
Accepted 27 March 2010
Available online 1 April 2010

*Keywords:*
Boolean approximation
Co-clones
Knowledge bases
Tractable inference

ABSTRACT

Since Selman and Kautz's seminal work on the use of Horn approximation to speed up the querying of knowledge bases, there has been great interest in Boolean approximation for AI applications. There are several Boolean classes with desirable computational properties similar to those of the Horn class. The class of affine Boolean functions, for example, has been proposed as an interesting alternative to Horn for knowledge compilation. To investigate the trade-offs between precision and efficiency in knowledge compilation, we compare, analytically and empirically, four well-known Boolean classes, and their combinations, for ability to preserve information. We note that traditional evaluation which explores unit-clause consequences of random hard 3-CNF formulas does not tell the full story, and we complement that evaluation with experiments based on a variety of assumptions about queries and the underlying knowledge base.

## 1. Introduction: Boolean approximation

The problem of efficient inference from propositional knowledge is of fundamental importance for symbolic reasoning, as used for example in circuit verification. Knowledge compilation, as introduced by Selman and Kautz [16] is a particular technique that uses approximations of a knowledge base to speed up querying, at the expense of completeness. As is often the case with powerful ideas, the approach is simple. Let a knowledge base $\varphi$ be given and let $\varphi^{\uparrow}$ be a logical consequence (an upper approximation) of $\varphi$, with $\varphi^{\uparrow}$ belonging to some class (Horn, for example) of Boolean functions for which the question of logical consequence is tractable. For any query $\alpha$, a positive answer to the question "$\varphi^{\uparrow} \models \alpha$?" affirms "$\varphi \models \alpha$." A negative answer is of no help. However, if a lower approximation $\varphi^{\downarrow}$ is also available (as assumed in the Selman–Kautz framework), a negative answer to the question "$\varphi^{\downarrow} \models \alpha$?" can similarly be used to answer "$\varphi \models \alpha$" in the negative. Otherwise one has to fall back on some standard (expensive) method for resolving "$\varphi \models \alpha$," or answer "don't know."

We shall follow Kavvadias et al. [8] and refer to a unique best upper approximation as an "envelope" (and to the dual concept as a "core"—although we will not need that much, as cores are not well-defined for the classes discussed in this paper).

Three factors determine the effectiveness of knowledge compilation: (1) the time required to compute the approximations, amortised over all queries of the same knowledge base; (2) the time saved by evaluating queries using approximations; and (3) the proportion of queries for which the approximate knowledge base yields a definite answer. The second factor is determined by the choice of Boolean function class with tractable inference, and the first by availability of an efficient algorithm for calculating approximations in the class. Both of these aspects have received considerable attention,

[*] Corresponding author. Tel.: +61 3 8344 1342; fax: +61 3 9348 1184.
*E-mail addresses:* schachte@unimelb.edu.au (P. Schachte), harald@unimelb.edu.au (H. Søndergaard), leighwhiting@gmail.com (L. Whiting), kevin.henshall@gmail.com (K. Henshall).

0004-3702/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2010.03.003

including in-depth study [10]. Del Val [6] has shown that Selman and Kautz's Horn envelope algorithm carries over to all Boolean function classes closed under subsumption. He has proposed an improved algorithm that is applicable if, additionally, the complement of a class is closed under resolution. Moreover, del Val has discussed the case of first-order predicate logic, showing how the original concepts can be extended in that direction as well [6].

While computational questions have received much attention, less has been said about *information loss* in knowledge compilation. In their study, Selman and Kautz [16] analysed how well the generated approximations preserve information. They applied their method to a large number of hard propositional formulas in 3-CNF and analytically derived an estimate of how many queries of the forms $x$, $x \vee y$, and $x \vee y \vee z$ could be answered successfully, based on the approximations alone. The results were very encouraging, particularly as the less expressive class of conjunctive (unit Horn) functions were substituted for the Horn class to simplify analysis. One aim of this note is to extend and complement the Selman–Kautz analysis with an empirical analysis of fidelity, not only for Horn approximations, but also for (upper) Krom, contra-dual Horn, and affine approximations.

A second aim is to understand how fidelity varies with underlying assumptions about knowledge bases and queries. Selman and Kautz's analysis was for random hard 3-CNF knowledge bases. These are strong Boolean functions, close to half of which are unsatisfiable and lead to unsatisfiable approximations, while the rest typically have relatively few models. Each application of knowledge compilation is different, but it must be expected that some applications involve somewhat weaker Boolean functions, or at least that inconsistent knowledge bases be repaired. How well will approximation to different classes preserve information in that setting? Here we empirically measure information loss in approximation to the four different classes and their combinations, using three different test sources: (1) random Boolean functions, (2) random 3-CNF, including random hard 3-CNF (as in the Selman–Kautz analytical investigation), and (3) structured functions arising from encodings of combinatorial problems. We measure information loss in two different ways: by counting the number of models added in envelopes, and by calculating the fraction of random queries entailed by the original function but not by its approximation. Results from the different experiments are given in Sections 5–7.

A third aim is to investigate to what extent combinations of Boolean classes can improve the success rate for accelerated query-answering. This question presents itself naturally, owing to this observation (Proposition 2): For any query $\alpha$ which can be expressed in 3-CNF, whether a knowledge base $\varphi$ entails $\alpha$ can be decided completely from $\varphi$'s Horn and contra-dual Horn envelopes. An interesting corollary is that, given the assumptions used in Selman and Kautz's analysis [16], one can obtain not just better results by using both envelopes, but perfect ones—100% accuracy is achieved without the need for any lower approximations. Related conclusions are drawn in Section 8.

## 2. Boolean co-clones

Let $\mathcal{B} = \{0, 1\}$ and let $\mathcal{V}$ be a countably enumerable set of variables. A *valuation* $\mu : \mathcal{V} \to \mathcal{B}$ is an assignment of truth values to the variables in $\mathcal{V}$. Let $\mathcal{I} = \mathcal{V} \to \mathcal{B}$ denote the set of $\mathcal{V}$-valuations. A Boolean function over $\mathcal{V}$ is a function $\varphi : \mathcal{I} \to \mathcal{B}$. We let **B** denote the set of all Boolean functions over $\mathcal{V}$. The ordering on $\mathcal{B}$ is the usual: $x \leqslant y$ iff $x = 0 \vee y = 1$. **B** is ordered pointwise, so that the ordering relation corresponds exactly to classical entailment, $\models$. A valuation $\mu$ is a *model* for $\varphi$, denoted $\mu \models \varphi$, if $\varphi(\mu) = 1$. We use the usual connectives, including $+$ for exclusive or. We let $\langle x, y, z\rangle$ denote the *median*[1] operation: $\langle x, y, z\rangle = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$. These connectives will also be applied to valuations with the obvious intention (pointwise application).

In this study we are concerned with the four *Schaefer classes* [1]:

**Krom (K):** $\varphi$ is *Krom* iff for all valuations $\mu$, $\mu'$, and $\mu''$, $\langle \mu, \mu', \mu''\rangle \models \varphi$ whenever $\mu \models \varphi$, $\mu' \models \varphi$, and $\mu'' \models \varphi$. Syntactically, **K** is the set of functions that can be written in conjunctive normal form with at most two literals per clause, and its members are also referred to as 2-*CNF* or *bijunctive*.

**Horn (H):** $\varphi$ is *Horn* iff for all valuations $\mu$ and $\mu'$, $(\mu \wedge \mu') \models \varphi$ whenever $\mu \models \varphi$ and $\mu' \models \varphi$. **H** is the set of functions that can be written in conjunctive normal form $\bigwedge(\ell_1 \vee \cdots \vee \ell_k)$, $k \geqslant 0$, with at most one positive literal $\ell$ per clause.

**Contra-dual Horn (N):** $\varphi$ is *contra-dual*[2] Horn iff for all valuations $\mu$ and $\mu'$, $(\mu \vee \mu') \models \varphi$ whenever $\mu \models \varphi$ and $\mu' \models \varphi$. These are the functions that can be written in conjunctive normal form $\bigwedge(\ell_1 \vee \cdots \vee \ell_k)$, $k \geqslant 0$, with at most one negative literal $\ell$ per clause.

**Affine (A):** $\varphi$ is *affine* iff for all valuations $\mu$, $\mu'$, and $\mu''$, $(\mu + \mu' + \mu'') \models \varphi$ whenever $\mu \models \varphi$, $\mu' \models \varphi$, and $\mu'' \models \varphi$ [15]. A Boolean function is affine iff it can be written as a conjunction of terms $c_0 + c_1x_1 + c_2x_2 + \cdots + c_kx_k$, where $c_i \in \{0, 1\}$ and $x_i \in \mathcal{V}$ for all $i \in \{0, \ldots, k\}$.[3]

There are other classes of interest, including $k$-Horn [4] and $k$-quasi-Horn, for which envelopes are also well-defined. Some well-studied classes, however, including the class of *unate* functions and the class of *renamable Horn* functions, do not

[1] Or "majority" operation; we prefer the terminology and notation suggested by Knuth [9].

[2] We follow Halmos [7] in using this term.

[3] In the cryptography/coding community, "affine" is used for what Post [12] called an "alternating" function, that is, a function that can be written $c_0 + c_1x_1 + c_2x_2 + \cdots + c_kx_k$, $k \geqslant 0$. The class of alternating functions is not closed under conjunction.

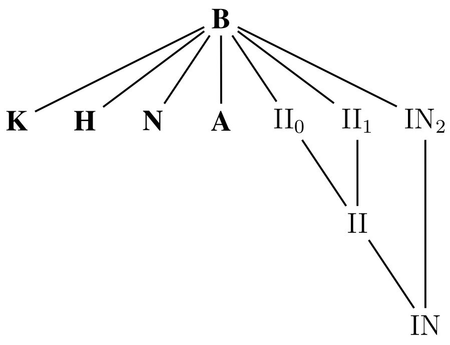


**Fig. 1.** Lattice of co-clones—top part.

provide unique best upper approximations.[4] It is inconvenient to have the question "Does $\psi$ follow from the approximation of $\varphi$?" answered non-deterministically, so we insist that the relation between a Boolean function and its envelope should be a proper *function*.[5] This corresponds to the technical requirement that a Boolean function class be a *co-clone* [11]—roughly a Boolean function collection that can be characterised via "model closure" under a set of connectives (as we did for the four classes above). It is well known that every co-clone contains 1 and is closed under conjunction and existential quantification. Hence for any co-clone $\mathcal{C}$, the $\mathcal{C}$-envelope of a Boolean function $\varphi$ is defined simply as $\bigwedge\{\psi \in \mathcal{C} \mid \varphi \models \psi\}$.

Recall that a *clone* is any family of functions, containing the projection functions and closed under composition. The Boolean clones form a lattice under subset ordering, and this lattice is (a subset of) Post's well-known lattice [12], from which one easily reads Post's famous functional completeness result for propositional logic. The relations between the clones, the lattice of co-clones, and the classes identified by Schaefer in his dichotomy result for generalised satisfiability [15] are well explained by Böhler et al. [1].

Fig. 1 shows the top end of the lattice of co-clones, including **K**, **H**, **N**, **A**, and **B**. The classes $\mathrm{II}_0$ and $\mathrm{II}_1$ are Schaefer's 0-*valid* and 1-*valid* classes, respectively.[6] That is, $\varphi \in \mathrm{II}_0$ iff $(\lambda v.0) \models \varphi$, and $\varphi \in \mathrm{II}_1$ iff $(\lambda v.1) \models \varphi$. The class $\mathrm{IN}_2$ is the set of Boolean functions $\varphi$ satisfying the model-closure constraint that $(\neg\mu) \models \varphi$ whenever $\mu \models \varphi$. The class IN is the intersection $\mathrm{II}_0 \cap \mathrm{II}_1 \cap \mathrm{IN}_2$ (in this expression we could equivalently leave out either of the first two).

For **K**, **H**, and **N**, there are well-known linear-time algorithms for deciding satisfiability (SAT) of formulas in CNF, and for **A**, satisfiability is decidable in polynomial time, assuming the usual matrix form representation for affine functions in modulo-2 arithmetic. Simple reductions to SAT show that entailment can similarly be decided in polynomial time for these classes. Let $\mathcal{C}$ be any one of **K**, **H**, **N**, or **A**. The entailment $\varphi \models \gamma_1 \wedge \cdots \wedge \gamma_n$ holds exactly when each $\gamma_i$ is a logical consequence of $\varphi$, so it suffices to consider entailment of each conjunct separately. But $\varphi \models \gamma$ amounts to the satisfiability of $\varphi \wedge \neg\gamma$, a formula which can be translated to a conjunction of $\mathcal{C}$ constraints in linear time. Moreover, $\varphi \wedge \neg\gamma$ belongs to $\mathcal{C}$ whenever $\varphi$ does.

The proposition below communicates the sense in which the four Schaefer classes are appropriate for this study. We have already argued why we focus on Boolean co-clones (for envelopes to be unique), and Proposition 1 then says that, assuming a CNF presentation, classes not contained in the Schaefer classes do not allow for efficient entailment tests, unless P = co-NP. To be precise, let $\mathcal{E}_{\mathcal{C}}$ be the *entailment problem* for class $\mathcal{C}$, an instance of which is of the form $(\varphi, \psi)$, with $\varphi \in \mathcal{C}$, and $\varphi$ and $\psi$ presented in CNF. The decision problem posed in $\mathcal{E}_{\mathcal{C}}$ is "Does $\varphi \models \psi$ hold?"

**Proposition 1.** *For each co-clone $\mathcal{C}$ not contained in* **K**, **H**, **N** *or* **A**, *$\mathcal{E}_{\mathcal{C}}$ is co-NP complete.*

**Proof.** The complement of $\mathcal{E}_{\mathcal{C}}$ has an obvious polynomial time verifier. For an instance $(\varphi, \psi)$, the verifier uses as certificate an interpretation that satisfies $\varphi$ but not $\psi$. Inspection of Fig. 1, together with the fact that all omitted co-clones are subsets of **K**, **H**, **N**, or **A**, then makes it clear that we only need to show $\mathcal{E}_{\mathrm{IN}}$ co-NP hard. We do this by providing a sequence of three polynomial-time mapping reductions $\mathit{UNSAT}_{\mathbf{B}} \overset{f}{\rightsquigarrow} \mathit{UNSAT}_{IN_2} \overset{g}{\rightsquigarrow} \mathcal{E}_{IN_2} \overset{h}{\rightsquigarrow} \mathcal{E}_{IN}$, where $\mathit{UNSAT}_{\mathcal{C}}$ is the unsatisfiability problem for class $\mathcal{C}$.

For the first reduction, consider $f$ defined by $f(\varphi) = \varphi \vee \varphi^\circ$, where $\varphi^\circ$ is $\varphi$'s contra-dual, that is, the negation of $\varphi$'s dual function. Note that $\varphi$ and $\varphi^\circ$ are equi-satisfiable and so $f(\varphi) \in IN_2$ is satisfiable iff $\varphi$ is. To see that a formula for $f$ can be computed in polynomial time, note that $f(\varphi) = \bigwedge\{\gamma \vee \eta \mid \gamma \in \varphi \wedge \eta \in \varphi^\circ\}$, and that $\varphi^\circ$ is obtained by simply changing

[4] For example, $x \rightarrow y$ and $x \leftarrow y$ are unate, but $x \leftrightarrow y$ is not, so the "unate envelope" of the latter is not well-defined.

[5] Where there is no unique best upper approximation, two minimal upper approximations may suffer exponentially different information loss. Section 3 discusses this phenomenon (in the dual—for maximal lower approximations). Without an insistence on classes with unique best approximations, we would need to discuss actual approximation algorithms, and their probability of producing a better or worse minimal upper approximation.

[6] Here we use the nomenclature of Böhler et al. [1].

the sign of each literal in $\varphi$. The second reduction is performed by $g$, defined by $g(\varphi) = (\varphi, 0)$—clearly a polynomial-time reduction. For the third reduction, consider the function $h$ defined by

$$h(\varphi, \psi) = \begin{cases} (\varphi, \psi) & \text{if } \varphi \text{ is 0-valid or 1-valid} \\ (II(\varphi), II(\psi)) & \text{otherwise} \end{cases}$$

where $II$ yields the 0-valid 1-valid envelope. Note that $II(\varphi)$ can be obtained in polynomial time by replacing each $\varphi$-clause $\gamma$ by $\bigwedge\{u \vee \neg v \vee \gamma \mid u, v \text{ occur in } \varphi\}$. □

## 3. Lower and upper approximations

A class closed under disjunction offers a unique weakest implicant, or "core"—dual to the concept of an envelope. However, none of the classes we consider are closed under disjunction. In the absence of unique best lower approximations it is common to use *maximal* lower approximations from a given class.[7] Selman and Kautz [16] show how to derive a maximal lower Horn approximation by repeated use of so-called Horn strengthening (removal of $k-1$ positive literals from a clause that has $k > 1$ positive literals). There has been much interest in the task of finding maximal lower approximations, in particular for Horn approximations [2].

However, different maximal approximations may have staggeringly different information content, and so should be treated with great care. Zanuttini [17] elegantly makes this point for **K**, **H**, and **A** with a single example. He considers the Boolean $n$-place function that has the $2^{n-3}+2$ models $\{m100 \mid m \in \{0,1\}^{n-3}\} \cup \{00\ldots010, 00\ldots001\}$. Maximal lower Horn approximations include $\{00\ldots010\}$ and the exponentially larger $\{m100 \mid m \in \{0,1\}^{n-3}\}$. Maximal lower Krom approximations include $\{00\ldots010, 00\ldots001\}$ and the exponentially larger $\{m100 \mid m \in \{0,1\}^{n-3}\} \cup \{00\ldots010\}$. Maximal lower affine approximations are $\{00\ldots010, 00\ldots001\}$ and the exponentially larger $\{m100 \mid m \in \{0,1\}^{n-3}\}$.

Upper-approximation in the same classes avoids this kind of divergence. Nevertheless, for each class $\mathcal{C}$ studied here, the loss of information involved in $\mathcal{C}$ upper-approximation can be considerable: the number of models added to a function can be exponential in its number of variables. Below are examples of $n$-place Boolean functions that can produce the vacuous envelope 1. They show that the $\mathcal{C}$-envelope may in the worst case incur a great loss of information.

(1) With $n \geqslant 3$, the function $\bigwedge\{x_i \vee x_j \vee x_k \mid 1 \leqslant i < j < k \leqslant n\}$ is in **N** and has $n(n+1)/2+1$ models. Its **K** envelope has $2^n$ models.
(2) With $n \geqslant 2$, the function $\bigwedge\{\neg x_i \vee \neg x_j \mid 1 \leqslant i < j \leqslant n\}$ is in **H** (and **K**) and has $n+1$ models. Its **N** envelope, as well as its **A** envelope, have $2^n$ models.
(3) With $n \geqslant 2$, the function $\bigwedge\{x_i \vee x_j \mid 1 \leqslant i < j \leqslant n\}$ is in **N** (and **K**) and has $n+1$ models. Its **H** envelope, as well as its **A** envelope, have $2^n$ models.

Notably, the Horn envelope combined with the contra-dual Horn envelope achieves complete coverage for queries that are presented in 3-CNF:

**Proposition 2.** *Let $\varphi$ be a Boolean function and let $\varphi'$ and $\varphi''$ be its Horn and contra-dual Horn envelopes. Let $\psi$ be a Boolean formula in 3-CNF. Then $\varphi \models \psi$ iff, for each clause $C$ of $\psi$, $\varphi' \models C$ or $\varphi'' \models C$.*

**Proof.** The "if" part is immediate, as $\varphi \models \varphi'$ and $\varphi \models \varphi''$, and so, since each clause of $\psi$ is a logical consequence of $\varphi$, $\psi$ is too. For the "only if" part, assume that $\varphi \models \psi$ and let $C$ be an arbitrary clause of $\psi$. Then $\varphi \models C$, and since $C \in \mathbf{H}$ or $C \in \mathbf{N}$, $C$ is a logical consequence of either $\varphi'$ or $\varphi''$ (or both). □

Proposition 2 can be strengthened. Using results from del Val [5], one can show that the proposition's assertion holds even if we assume that $\varphi'$ is the 3-*CNF* Horn envelope and $\varphi''$ is the 3-*CNF* contra-dual Horn envelope. Of course, not all queries have a 3-CNF presentation; there are Boolean functions that cannot be so expressed.

## 4. Experimental method

Our aim is to empirically evaluate the loss of information incurred by the approximation of knowledge bases to classes that guarantee a unique best approximation and tractable entailment checking. Only (sub-classes of) the four Schaefer classes satisfy these requirements. As subclasses correspond to greater information loss, we consider only the four. We stress that no single representation is ideal for all four classes. CNF works well for **K**, **H**, and **N** but is less suitable for **A**. A modulo-2 arithmetic matrix form is ideal for **A**, but unsuitable for **K**, **H**, and **N**. Hence we use the more neutral Reduced Ordered Binary Decision Diagrams (ROBDDs) [3] and associated envelope algorithms [13,14]. From an efficiency point of view, our representation is adequate for all classes, but ideal for none. But since our focus is on measuring information loss,

[7] We avoid the commonly used term GLB for a maximal lower approximation, as it has a different use in lattice theory.

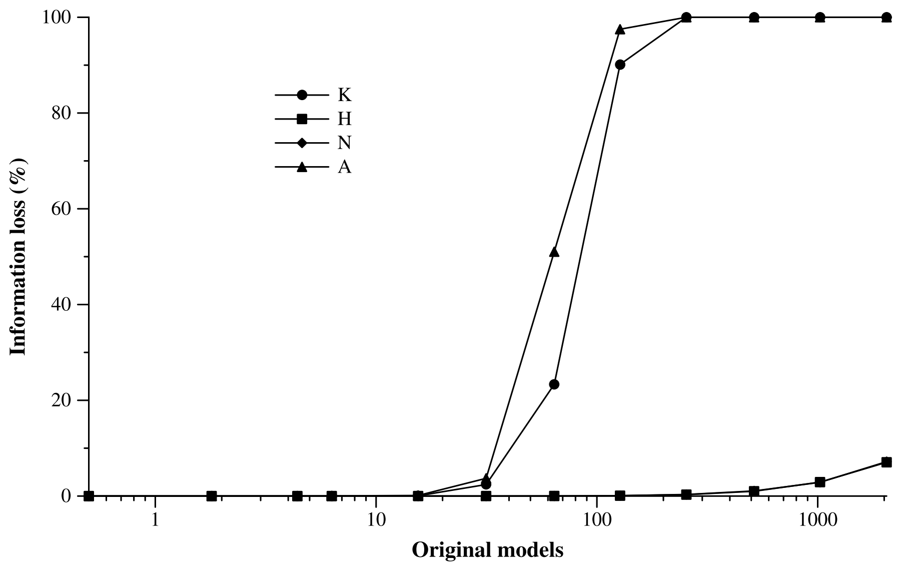


**Fig. 2.** Information lost (models gained) by envelopes (random functions).

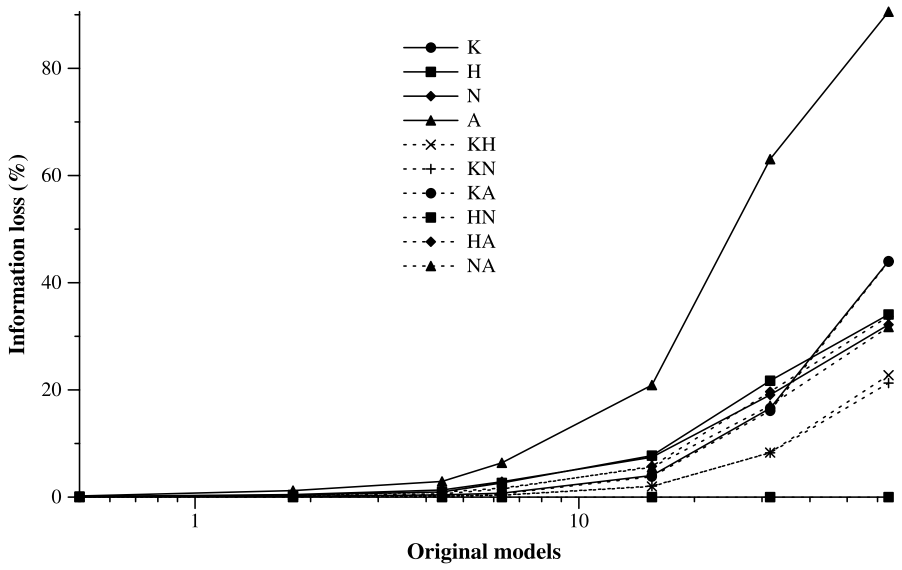


**Fig. 3.** 3-CNF queries dropped (percentages) by envelopes (random functions).

we are less concerned about the most efficient representation for a given class. ROBDDs provide a uniform base for our experiments, and also allow us to generate Boolean functions that are truly random, in the sense that every valuation has an equal probability of being a model.

## 5. First experiment: randomly generated Boolean functions

We first explore information loss due to approximation in random Boolean functions. To this end, we generate Boolean functions of 24 variables such that each valuation has probability $p$ of being a model, varying $p$ from 1 in $2^{15}$ to 1 in $2^{24}$.

For each random Boolean function we evaluate the information loss from taking **K**, **H**, **N** and **A** envelopes in two different ways. Firstly, we compute the number of models of the original function and of each of its envelopes, and consider that every model of an envelope that is not a model of the original function is information lost by that approximation. We then calculate the percentage of loss as follows. Let $m$ be the number of models of the original function and $n$ be the number of models of its envelope. The percentage of information loss is then $100 \cdot (n - m)/(2^{24} - n)$. The results of this calculation for each envelope over functions of varying strengths is presented in Fig. 2. The number of models of the original function is

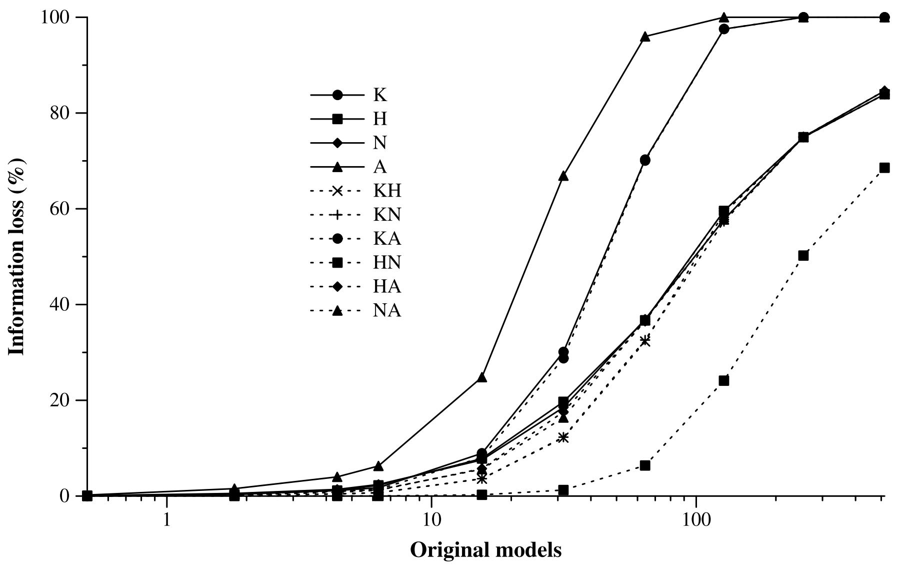


**Fig. 4.** 6-CNF queries dropped (percentages) by envelopes (random functions).

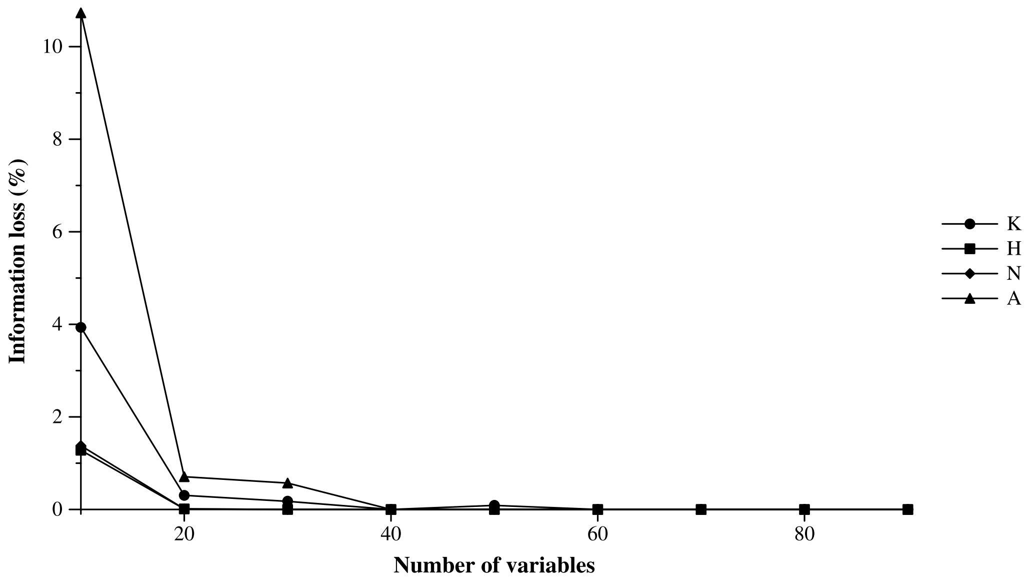


**Fig. 5.** Information lost vs. function size (random functions).

presented logarithmically along the $X$ axis, with the strongest functions on the left, while the $Y$ axis shows the percentage of information loss. Note that all approximations lie imperceptibly above the $X$ axis (with **A** and **K** slightly worse than **H** and **N**) up to functions with about 16 models. **H** and **N** are indistinguishable.

Our second appraisal of information loss counts the proportion of random queries that are correctly answered by the approximation. A query is "dropped" if it is entailed by the original function but not by the envelope. We follow the tradition in the literature of using single clause 3-CNF queries (that is, disjunctions of three literals). We also consider 6-CNF queries, to see if weaker queries give different results. We do not consider queries consisting of conjunctions of clauses, as they could be handled by separately checking if each clause is entailed. Each query is formed by generating 3 (or 6) literals at random, ensuring that no two literals involve the same variable, and each envelope is tested with 1000 3-CNF and 1000 6-CNF queries, considering only queries that are indeed entailed by the original function. All of this is repeated for 100 random original formulas, and averages are calculated. We show no results where fewer than 1% of the random queries (1000 queries overall) are entailed by the original function.

One might wonder whether queries that cannot be answered for one approximation might be handled successfully by another. This is of interest beyond selecting the most effective approximation: one might well use two separate types of

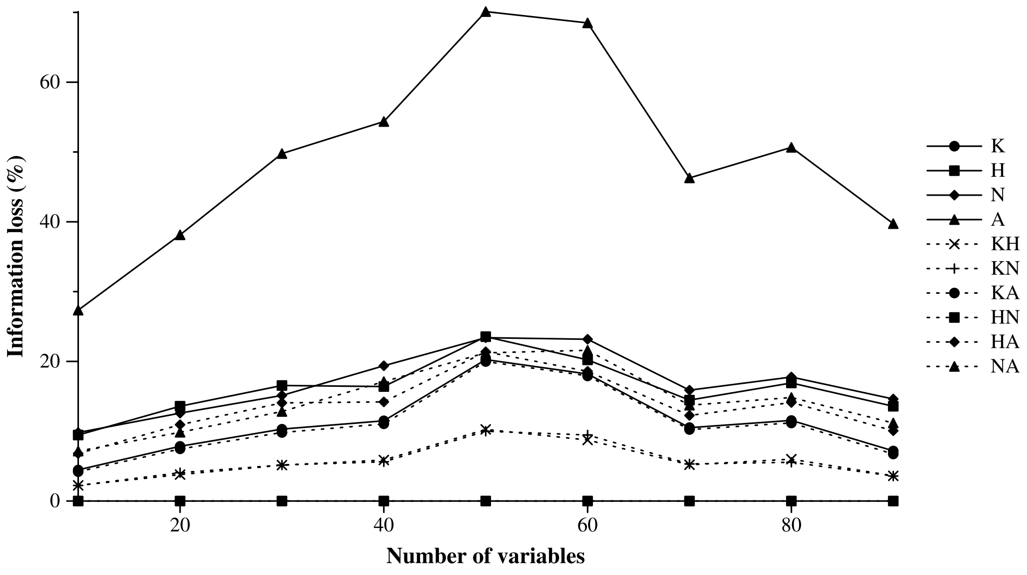


**Fig. 6.** 3-CNF queries dropped vs. function size (random functions).

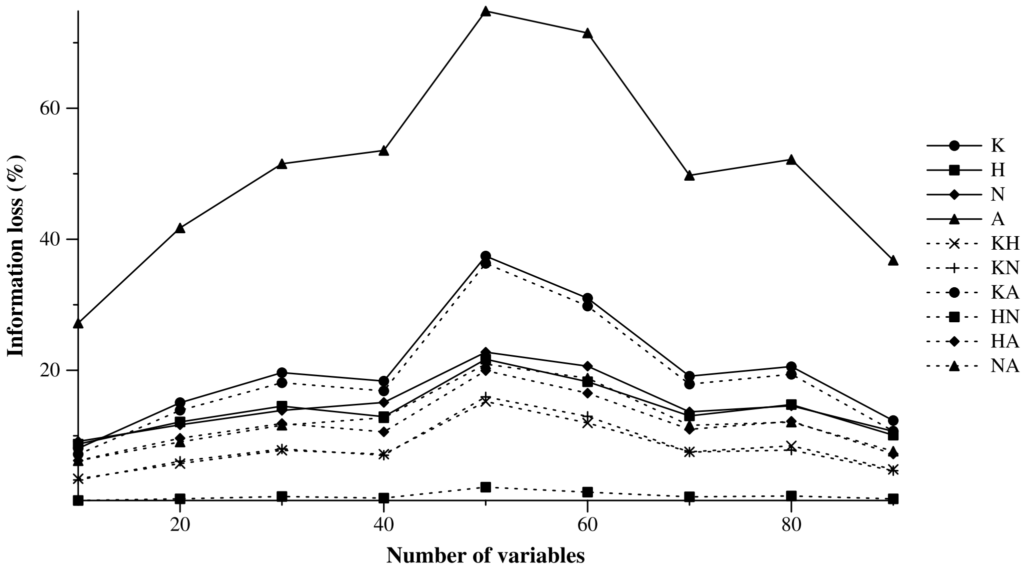


**Fig. 7.** 6-CNF queries dropped vs. function size (random functions).

approximation, and successfully answer the query if either is able to answer it. Although this requires computing two approximations of a knowledge base, it may be worthwhile in applications where the knowledge base does not change often. However it will be wasteful if both approximations tend to fail for the same queries. Thus we additionally present results for each pair of approximation domains, where we consider a pair to answer a query successfully if *either* does (or both do, of course). Figs. 3 and 4 show the results for 3-CNF and 6-CNF queries, respectively.

We also confirmed that these results hold for formulas of different numbers of variables, ranging from 10 to 90, but similar strengths. Fig. 5 also shows the percentage of information loss computed as above plotted against the number of variables in each randomly generated function. In each case, for a function of $n$ variables, we set the probability of each valuation being a model such that there would be approximately $n$ models. These results would seem to show little information loss once the number of variables rises above 20 or 30, however Figs. 6 and 7, showing the percentage of 3-CNF and 6-CNF queries dropped by each approximation, refute this conclusion. Here again we fix the probability of each valuation being a model to approximately $n$ in $2^n$. As above, in all cases, 100 random Boolean functions were used, and

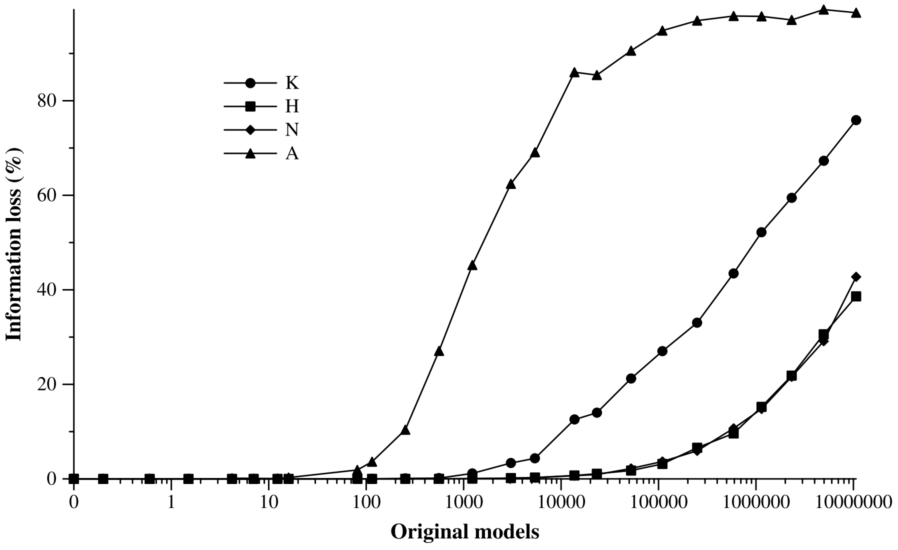


**Fig. 8.** Information lost (models gained) by envelopes (3-CNF functions).

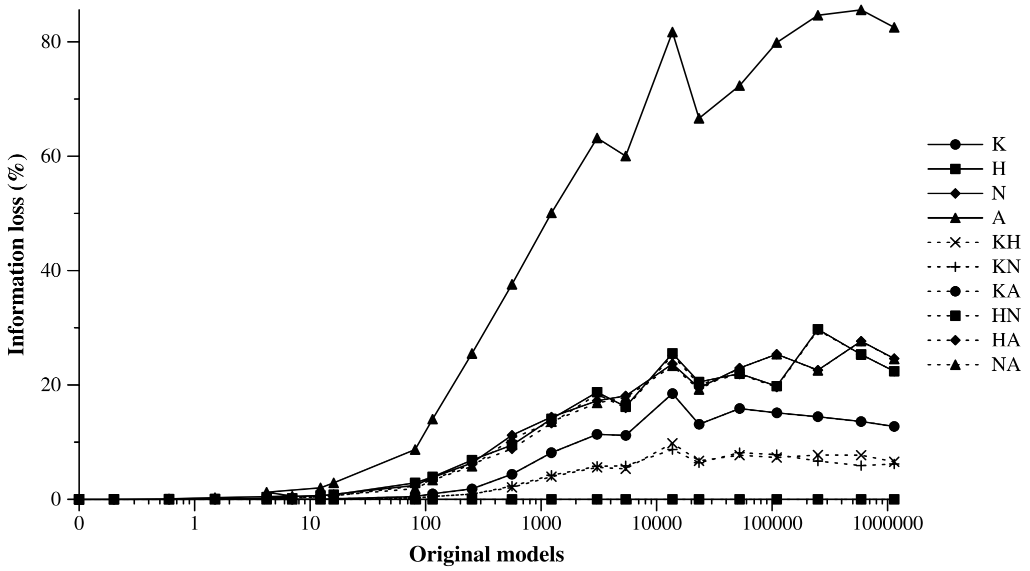


**Fig. 9.** 3-CNF queries dropped (percentages) by envelopes (3-CNF functions).

1000 3-CNF and 1000 6-CNF random test queries were used for each approximation. The charts show a fairly consistent degree of information loss, when measured by queries dropped, regardless of number of variables.[8]

## 6. Second experiment: random 3-CNF including hard 3-CNF

The second experiment is like the first, except that arbitrary random functions have been replaced by random functions expressible as 3-CNF formulas. This provides a baseline for comparison with related work which has almost exclusively considered the (hard) 3-CNF case. In the first experiment we generated formulas for a variety of model-probability distributions. Here instead we generate formulas that have a variety of clause-to-variable ratios. (Each formula is translated to ROBDD form, and the various approximations are then produced from that.) In all cases the formulas denote 24-place Boolean functions, but explore a range of numbers of clauses from 8 to 148, including 103, which yields hard 3-CNF functions.

[8] The fluctuations across the graphs are due to random variation, and the fact that we approximated the $n/2^n$ model probability as 1 in $2^{n-\lfloor 0.5+\log_2(n)\rfloor}$. The systematic error caused by this rounding would tend to raise values for 50, 30, 60, and 70 variables, and lower them for 90, relative to the others.

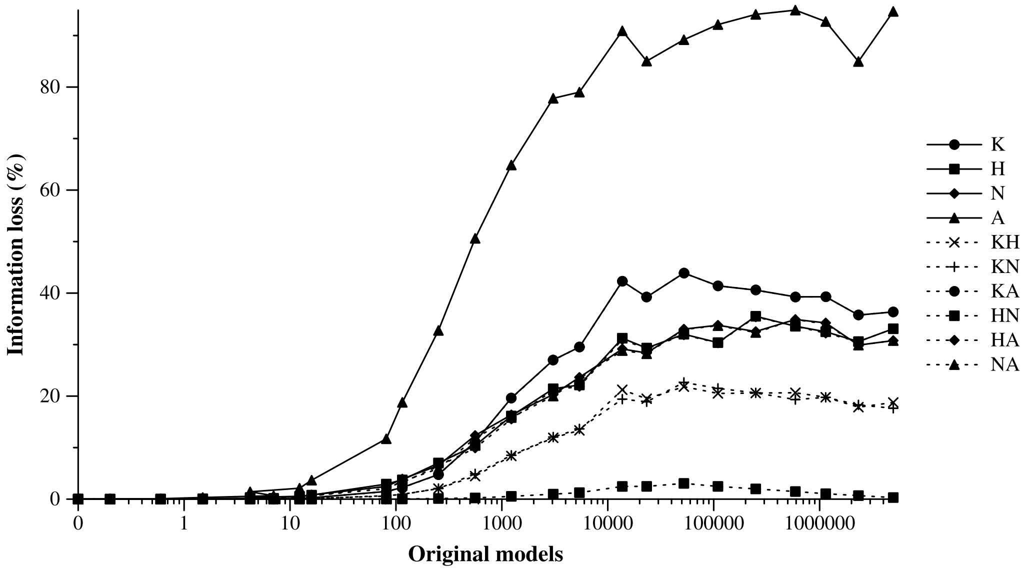


**Fig. 10.** 6-CNF queries dropped (percentages) by envelopes (3-CNF functions).

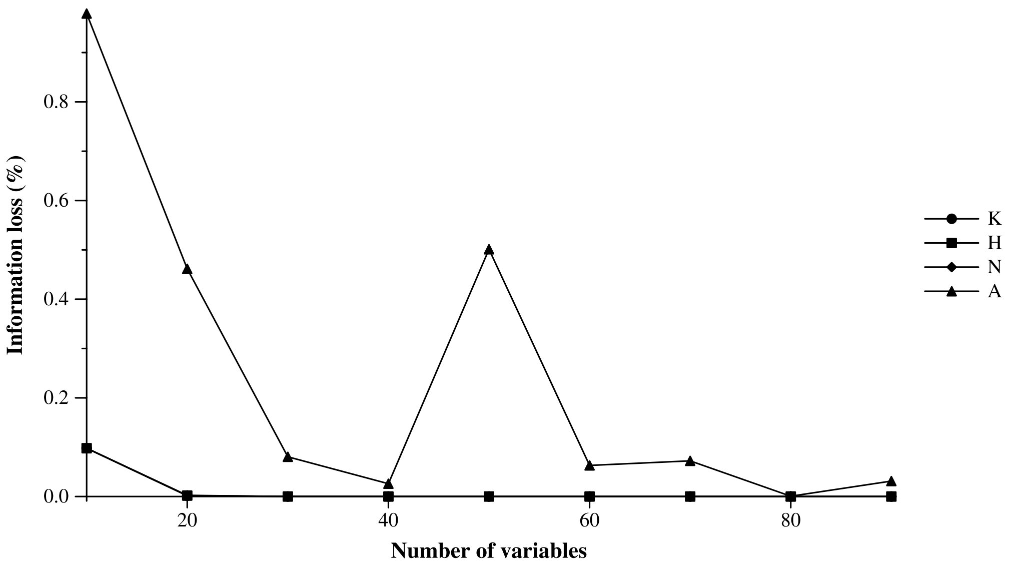


**Fig. 11.** Information lost vs. function size (3-CNF functions).

As in the first experiment, we measure information loss in terms of the models added by each envelope. For a given number of clauses, we again test 100 random functions, this time 3-CNF. The result is shown in Fig. 8. The point corresponding to 103 clauses (the hard 3-CNF case) falls around 2 models on the $X$ axis. Although it is not readily discernible from the figures, at the hard 3-CNF point, on average **K** approximation adds 1 model out of a possible $2^{24}$, **H** and **N** each add 4, and **A** adds 211.

Similarly, the proportions of 3-CNF and 6-CNF queries dropped by the various approximations (over 1000 tests each) are presented in Figs. 9 and 10, again including results for pairs of envelopes. At the hard 3-CNF point, for both 3- and 6-CNF queries, **K** performs best and **A** worst (still dropping less than 1% of queries).

Again we have verified that these results are fairly insensitive to the size of the functions involved. Fig. 11 shows information lost (as percentage of models gained) through approximation for hard 3-CNF formulas of sizes ranging from 10 to 90 variables (43–387 clauses). Figs. 12 and 13 show information loss as measured by dropped queries, for hard 3-CNF formulas over the same size range. Once again, despite all approximations gaining a very small percentage of models, the proportion of queries dropped is not negligible, and is fairly independent of size. Note that **A** approximation loses the most information, **K** the least, and the combination of **H** and **N** again answers all 3-CNF and almost all 6-CNF queries correctly.

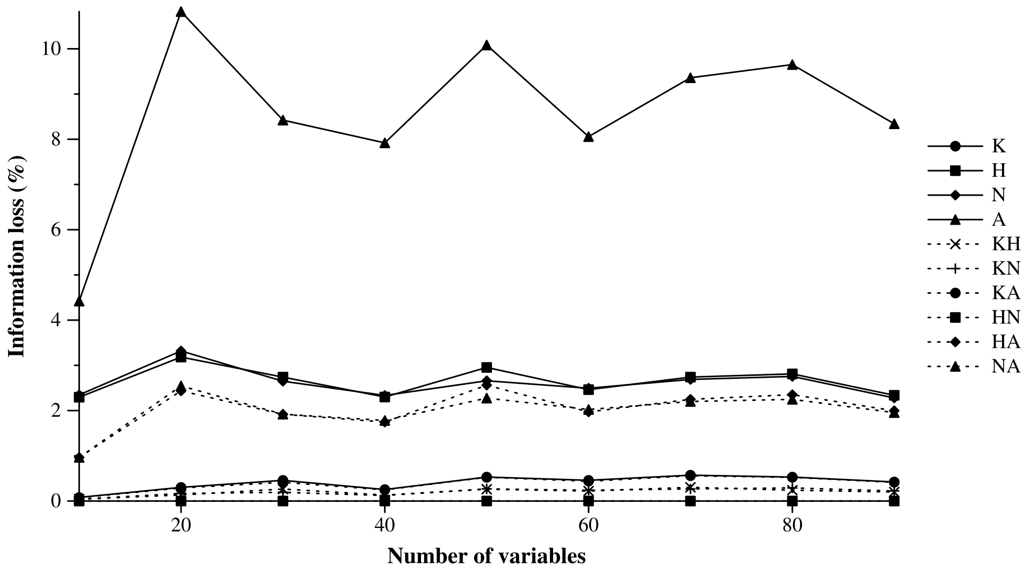


**Fig. 12.** 3-CNF queries dropped vs. function size (3-CNF functions).

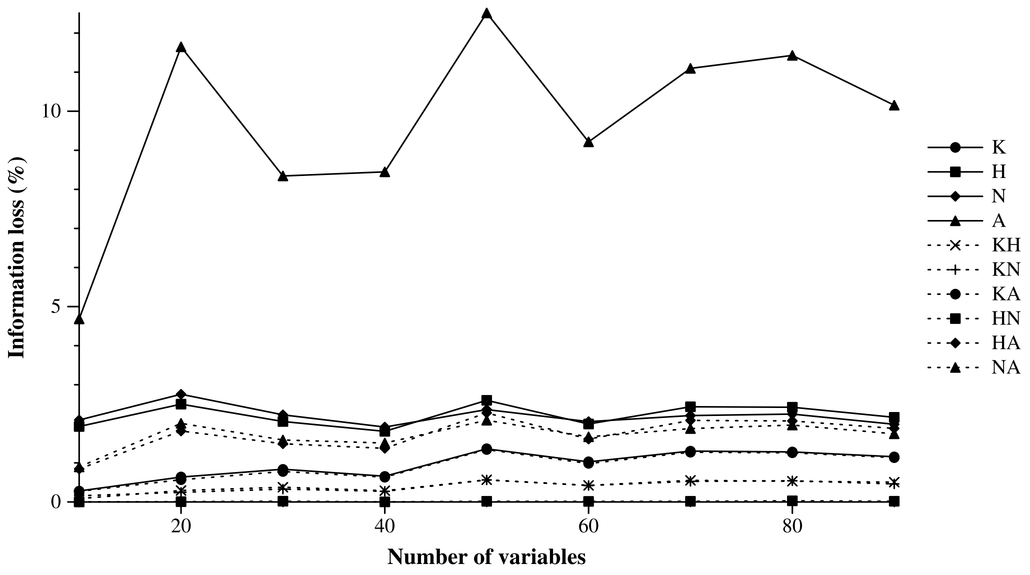


**Fig. 13.** 6-CNF queries dropped vs. function size (3-CNF functions).

## 7. Third experiment: structured Boolean functions

Randomly generated Boolean functions are arguably unrepresentative of real world knowledge bases that show some structure, so the third experiment investigates information loss for structured Boolean functions, sourced from SAT-based problem-solving. Table 1 gives the size of some sample problems and shows how many additional models are created by the various envelopes. Note that although some of the numbers are large, in no case is more than three percent of the possible models added. To give a better understanding of what this means for typical query-answering, Table 2 shows the percentage of 3-CNF queries dropped by the various envelopes and their pairwise combinations, and Table 3 does the same for 6-CNF queries. Each formula used is satisfiable and has been queried with 1000 random 3-CNF queries and 1000 random 6-CNF queries. The formulas are: `ais6.cnf`, an all-interval-series instance from SATLIB; `colour.cnf`, 4-colouring a map of the 7 western-most contiguous US states; `queens`*n*`.cnf`, solving the *n*-queens problem; `sudoku`*n*`.cnf`, solving a $4 \times 4$ sudoku instance using a unary encoding with *n* squares already filled; `sud1_`*n*`.cnf`, solving a $9 \times 9$ sudoku instance using a binary encoding with *n* squares already filled; `sud2_`*n*`.cnf`, similar, but a different instance; and `sv4.cnf`, disproving a software verification assertion. Note that for varying *n*, the `queens`*n*`.cnf` tests involve varying numbers of variables

**Table 1**
Information lost (number of models gained) by envelopes (structured functions).

| | Variables | Clauses | **K** | **H** | **N** | **A** |
|---|---|---|---|---|---|---|
| Ais6 | 61 | 581 | 43,890 | 178 | 672,555 | 1,048,552 |
| Colour | 28 | 97 | 10,425 | 2913 | 5,688,357 | 2,096,912 |
| Queens5 | 26 | 276 | 276 | 26 | 951 | 246 |
| Queens6 | 37 | 497 | 1 | 1 | 11 | 4 |
| Sudoku1 | 64 | 241 | 915,128 | 1968 | 15,132,387 | 131,000 |
| Sudoku2 | 64 | 242 | 2282 | 152 | 5991 | 1006 |
| Sudoku3 | 64 | 243 | 376 | 58 | 775 | 244 |
| Sudoku4 | 64 | 244 | 16 | 13 | 33 | 10 |
| Sud1_22 | 324 | 15,883 | 418,750 | 8945 | 241,026 | 16,777,170 |
| Sud1_23 | 324 | 15,887 | 64 | 55 | 270 | 119 |
| Sud1_24 | 324 | 15,891 | 1 | 3 | 4 | 1 |
| Sud2_26 | 324 | 15,899 | 431 | 229 | 239 | 244 |
| Sv4 | 58 | 150 | 896 | 901 | 247 | 0 |

**Table 2**
3-CNF queries dropped (percentages) by envelopes (structured functions).

| | **K** | **H** | **N** | **A** | **KH** | **KN** | **KA** | **HN** | **HA** | **NA** |
|---|---|---|---|---|---|---|---|---|---|---|
| Ais6 | 5.3 | 8.5 | 77.7 | 87.7 | 0.5 | 4.8 | 5.3 | 0.0 | 0.5 | 77.7 |
| Colour | 0.0 | 0.0 | 100.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 |
| Queens5 | 8.4 | 4.2 | 81.6 | 85.8 | 4.2 | 4.2 | 8.4 | 0.0 | 4.2 | 81.6 |
| Queens6 | 0.0 | 0.0 | 18.2 | 18.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 18.2 |
| Sudoku1 | 2.8 | 0.9 | 25.5 | 28.1 | 0.6 | 2.2 | 2.8 | 0.0 | 0.6 | 25.3 |
| Sudoku2 | 1.6 | 1.2 | 11.6 | 16.9 | 0.6 | 1.0 | 1.6 | 0.0 | 0.9 | 11.0 |
| Sudoku3 | 1.5 | 1.4 | 8.9 | 14.2 | 0.6 | 0.9 | 1.5 | 0.0 | 0.8 | 8.0 |
| Sudoku4 | 0.5 | 0.6 | 2.8 | 4.6 | 0.1 | 0.4 | 0.5 | 0.0 | 0.1 | 2.2 |
| Sud1_22 | 0.1 | 0.4 | 1.8 | 4.7 | 0.0 | 0.1 | 0.1 | 0.0 | 0.2 | 1.7 |
| Sud1_23 | 0.0 | 0.2 | 0.3 | 0.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.2 |
| Sud1_24 | 0.0 | 0.3 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.1 |
| Sud2_26 | 0.1 | 0.8 | 1.1 | 2.4 | 0.0 | 0.1 | 0.1 | 0.0 | 0.5 | 0.7 |
| Sv4 | 0.3 | 0.5 | 0.8 | 0.0 | 0.1 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 |

**Table 3**
6-CNF queries dropped (percentages) by envelopes (structured functions).

| | **K** | **H** | **N** | **A** | **KH** | **KN** | **KA** | **HN** | **HA** | **NA** |
|---|---|---|---|---|---|---|---|---|---|---|
| Ais6 | 8.6 | 2.0 | 64.1 | 87.0 | 2.0 | 5.4 | 8.5 | 0.3 | 1.9 | 64.1 |
| Colour | 6.5 | 2.3 | 100.0 | 99.8 | 2.3 | 6.5 | 6.5 | 2.3 | 2.3 | 99.8 |
| Queens5 | 9.7 | 4.9 | 64.4 | 84.1 | 4.9 | 1.4 | 8.6 | 0.0 | 4.0 | 62.8 |
| Queens6 | 0.3 | 0.3 | 8.8 | 7.4 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 | 7.4 |
| Sudoku1 | 8.4 | 2.4 | 36.0 | 43.2 | 2.2 | 6.3 | 8.3 | 0.9 | 2.1 | 35.2 |
| Sudoku2 | 2.5 | 1.9 | 12.5 | 21.2 | 0.9 | 1.3 | 2.5 | 0.0 | 1.3 | 11.6 |
| Sudoku3 | 1.6 | 1.8 | 8.1 | 14.7 | 0.9 | 0.4 | 1.6 | 0.0 | 1.0 | 7.2 |
| Sudoku4 | 0.4 | 1.1 | 2.5 | 3.0 | 0.4 | 0.0 | 0.4 | 0.0 | 0.4 | 1.4 |
| Sud1_22 | 0.3 | 0.6 | 1.1 | 3.5 | 0.1 | 0.1 | 0.3 | 0.0 | 0.5 | 0.8 |
| Sud1_23 | 0.0 | 0.2 | 0.4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 | 0.3 |
| Sud1_24 | 0.0 | 0.1 | 0.3 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| Sud2_26 | 0.2 | 0.8 | 0.4 | 1.8 | 0.0 | 0.2 | 0.1 | 0.0 | 0.5 | 0.2 |
| Sv4 | 0.3 | 0.3 | 0.3 | 0.0 | 0.2 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 |

but similar numbers of models, while each group of Sudoku tests involve the same number of variables, but considerable differences in the numbers of models.

Again, the results show that **HN** is the strongest, giving almost perfect results. Also, considering the `Sudoku`*n* and `Sud1_`*n* results, we again see that stronger Boolean functions (larger *n*) suffer less loss of information and dropped queries. These tests further show a bias not exhibited by the earlier tests: approximation loses more or less information, and drops fewer or more queries, depending on the idiosyncrasies of the knowledge bases and queries involved. Consider, for example, Table 3, where for `queens5`, **K** does not improve on **H** approximation, while it improves **N** to surpass the **KH** combination. Also, contrary to earlier results, **A** approximation performs better than **N** in many cases.

## 8. Conclusions

We have considered four classes of Boolean functions for use in knowledge compilation from the standpoint of information and consequence preservation, and showed that these are the only maximally precise classes guaranteeing unique best approximation and polynomial time entailment checking. Empirical evaluation of these approximations leads to several interesting observations:

- On the whole, knowledge compilation works well for strong knowledge bases.
- Querying pairs of domains is beneficial—for the right combinations of domains. In particular, **HN**, the combination of Horn and its contra-dual, uniformly gives the best results. This combination always answers 3-CNF queries precisely, with no need for lower approximation. As shown in Fig. 10 and Table 3, **HN** also drops very few 6-CNF queries for 3-CNF knowledge bases.
- Affine approximation (**A**) generally performs worst of the domains we have considered, and none of the other domains benefit much from combination with **A**. In a few of the structured test cases, **A** outperformed **N**, due to the nature of the knowledge base; in particular, the `Sv4` test case happens to be affine. For random knowledge bases, **A** always performs worst, though note that all test queries were disjunctive, not affine. It is difficult to recommend affine approximation for knowledge compilation, except in cases where the knowledge base or queries are expected to be nearly affine.
- Krom approximation (**K**), however, does appear to be useful. For 3-CNF knowledge bases, **K** approximations answer more 3-CNF queries correctly than any of the other approximations. For strong knowledge bases—3-CNF functions with a clause:variable ratio of at least 2.6 or Boolean functions with around 64 models—the phenomenon extends to 6-CNF queries. Remarkably, this is true despite the fact that both **H** and **N** generally give stronger approximations than **K**. It is also interesting that the precision of the **KH** and **KN** combinations is significantly better than any single domain, only being surpassed by **HN**.
- Whether we test random or 3-CNF knowledge bases, **H** and **N** rarely diverge. This is not surprising: these domains are contra-duals, so on average unbiased tests tend to behave similarly for **H** and **N** approximation. However it is striking that **H** and **N** behave very differently for almost all of the structured tests, probably owing to a natural human tendency to favour implications with a single consequent over implications with a single antecedent.
- As shown by Figs. 6, 7, 12, and 13, the query-answering performance of each approximation or combination is independent of the number of variables of the knowledge base. However, query-answering performance is very sensitive to the strength of the knowledge base, as shown by Figs. 3, 4, 9, and 10. The structured tests, all of which involved rather strong knowledge bases, but range from 26 to 324 variables, confirm this (Tables 2 and 3).
- Conclusions should not be drawn from the *hard* 3-CNF case alone. Of the 100 random hard 3-CNF functions in our experiments, 75 had fewer than 3 models. Those are all, by definition, already in both the Krom and the affine classes.

The last two points suggest that *the success of knowledge compilation is very sensitive to the nature of the knowledge base and queries themselves*. Where queries fall into a class with tractable inference, approximation of the knowledge base to that class may lose information, but will not drop any queries. Considering only the case of 3-CNF queries on approximations of hard 3-CNF knowledge bases does not adequately predict the behaviour for other shapes of knowledge bases and queries.

## Acknowledgements

We thank the reviewers, who all provided us with detailed and helpful advice. In particular, the comments made after Proposition 2 are due to the reviewers.

## References

[1] E. Böhler, N. Creignou, S. Reith, H. Vollmer, Playing with Boolean blocks, part II: Constraint satisfaction problems, ACM SIGACT News 35 (1) (2004) 22–35.
[2] Y. Boufkhad, Algorithms for propositional KB approximation, in: Proc. 15th Nat. Conf. Artificial Intelligence, AAAI Press/MIT Press, 1998, pp. 280–285.
[3] R. Bryant, Graph-based algorithms for Boolean function manipulation, IEEE Transactions on Computers C 35 (8) (1986) 677–691.
[4] R. Dechter, J. Pearl, Structure identification in relational data, Artificial Intelligence 58 (1992) 237–270.
[5] A. del Val, An analysis of approximate knowledge compilation, in: Proc. IJCAI'95, vol. 2, 1995, pp. 830–836.
[6] A. del Val, First order LUB approximations: Characterization and algorithms, Artificial Intelligence 162 (2005) 7–48.
[7] P.R. Halmos, Lectures on Boolean Algebras, Springer-Verlag, 1963.
[8] D. Kavvadias, C. Papadimitriou, M. Sideri, On Horn envelopes and hypergraph transversals, in: K. Ng, P. Raghavan, N. Balasubramanian, F. Chin (Eds.), Proc. Fourth Int. Symp. Algorithms and Computation, in: Lecture Notes in Computer Science, vol. 762, Springer-Verlag, 1993, pp. 399–405.
[9] D.E. Knuth, Introduction to Combinatorial Algorithms and Boolean Functions, The Art of Computer Programming, vol. 4, Addison-Wesley, 2008.
[10] P. Liberatore, Compilation of Intractable Problems and Its Application to Artificial Intelligence, PhD thesis, University of Rome "La Sapienza", Italy, 1998.
[11] N. Pippenger, Theories of Computability, Cambridge University Press, 1997.
[12] E.L. Post, The Two-Valued Iterative Systems of Mathematical Logic, Princeton University Press, 1941. Reprinted in M. Davis, Solvability, Provability, Definability: The Collected Works of Emil L. Post, Birkhäuser, 1994, pp. 249–374.
[13] P. Schachte, H. Søndergaard, Boolean approximation revisited, in: I. Miguel, W. Ruml (Eds.), Proc. SARA 2007, in: Lecture Notes in Artificial Intelligence, vol. 4612, Springer-Verlag, 2007, pp. 329–343.
[14] P. Schachte, H. Søndergaard, L. Whiting, K. Henshall, An algorithm for affine approximation of binary decision diagrams, Chicago Journal of Theoretical Computer Science, in press.
[15] T.J. Schaefer, The complexity of satisfiability problems, in: Proc. Tenth Ann. ACM Symp. Theory of Computing, 1978, pp. 216–226.
[16] B. Selman, H. Kautz, Knowledge compilation and theory approximation, Journal of the ACM 43 (2) (1996) 193–224.
[17] B. Zanuttini, Approximation of relations by propositional formulas: Complexity and semantics, in: S. Koenig, R. Holte (Eds.), Proc. SARA 2002, in: Lecture Notes in Artificial Intelligence, vol. 2371, Springer-Verlag, 2002, pp. 242–255.