Fast Algorithms for Lexicographic Inference

Jonas Klein , Matthias Thimm
Artificial Intelligence Group, University of Hagen, Germany
{jonas.klein,matthias.thimm}@fernuni-hagen.de

Abstract

We present two new algorithms for the problem of lexicographic inference from conditional knowledge bases. These algorithms are based on SAT and MaxSAT encodings of the underlying problem of lexicographic comparisons of classical interpretations and require only a polynomial number of SAT solver calls. In our experimental evaluation we show that our new algorithms significantly outperform the state of the art.

1 Introduction

Non-monotonic reasoning from conditional knowledge bases is a central topic within knowledge representation and reasoning [Lehmann and Magidor, 1989; Kraus *et al.*, 1990; Lehmann, 1995; Kern-Isberner, 2001; Nute and Cross, 2002; Haldimann *et al.*, 2025]. It addresses the inadequacy of reasoning with classical logic in the presence of *default* or *defeasible rules*, from now on called *conditionals*, such as "birds typically fly". Concrete approaches such as *rational closure* [Lehmann and Magidor, 1989] or *lexicographic inference* [Lehmann, 1995] typically rely on *plausibility orders* that rank classical interpretations in terms of how well they satisfy the conditionals of a given knowledge base. Lexicographic inference, which will be the focus of this paper, furthermore relies on the so-called Z-partitioning of the conditionals of the knowledge base, which sorts conditionals in terms of their exceptionality. Classical interpretations are then ranked according to a lexicographic comparison with respect to the number of falsified conditionals on the different levels of exceptionality. Lexicographic inference satisfies a series of desirable properties and is generally regarded as a plausible approach for non-monotonic reasoning from conditional knowledge [Benferhat *et al.*, 1993; Heyninck *et al.*, 2023].

As outlined above, lexicographic inference is defined on rather complex structures such as Z-partitionings and orderings over interpretations. It is therefore no surprise that the computational complexity of lexicographic inference is rather high [Cayrol *et al.*, 1998; Eiter and Lukasiewicz, 2000]. More concretely, the problem of deciding, whether a propositional formula ψ is lexicographically inferred by a propositional formula ϕ, given a set of conditionals Δ, is P^{NP}-complete [Cayrol *et al.*, 1998]. The challenge of addressing this problem algorithmically has, to the best of our knowledge, only been addressed in a recent paper by Haldimann et al. [2025], who develop an algorithm based on *minimal correction sets* and MaxSAT solver technology. In this paper, we present two new algorithms to address the very same problem. Our first algorithm relies on a compact representation of the lexicographic comparison criterion as a SAT encoding and thus avoids the costly (and repeated) computation of minimal correction sets. Given the Z-partitioning of a set of conditionals, our first algorithm repeatedly determines interpretations of increasing plausibility, by employing said SAT encoding, and depending on whether the finally found interpretation satisfies or violates the queried inference problem, can safely return the correct answer. Our second algorithm also employs (weighted) MaxSAT solver technology and first determines a maximal plausible interpretation for the satisfaction of the query. It then checks (using a single further SAT solver call) whether there is any interpretation for the violation of the query that is at least as plausible as that one, and can then safely return the correct answer. Our extensive experimental evaluation shows that both our new algorithms significantly outperform the algorithm from [Haldimann *et al.*, 2025].

In summary, the main contributions of this paper are as follows:

1. We revisit the computational complexity of lexicographic inference and identify the complexity of certain sub-problems that can be used for algorithmic purposes (Section 3)
2. We present two algorithms for lexicographic inference (Section 4)
3. We instantiate the relevant sub-procedures of our algorithms with suitable (Max)SAT encodings (Section 5)
4. We conduct an extensive experimental evaluation (Section 6)

We present necessary preliminaries in Section 2 and conclude in Section 7. Proofs of all technical results, as well as supplementary material, can be found in an extended version of this paper[1].

[1]Link to extended version: https://zenodo.org/records/20081561

2 Preliminaries

For a (finite) set At of atoms (also called *signature*) let $\mathcal{L}(\mathsf{At})$ be the corresponding propositional language constructed using the usual connectives $\wedge$ (*and*), $\vee$ (*or*), $\neg$ (*negation*) and $\rightarrow$ (*material implication*). A (classical) *interpretation* (also called *possible world*) ω for a propositional language $\mathcal{L}(\mathsf{At})$ is a function $\omega : \mathsf{At} \rightarrow \{\mathsf{t}, \mathsf{f}\}$. Let $\Omega(\mathsf{At})$ denote the set of all interpretations for At. We simply write Ω if the set of atoms is implicitly given. An interpretation ω *satisfies* (or is a *model* of) an atom $a \in \mathsf{At}$, denoted by $\omega \models a$, if and only if $\omega(a) = \mathsf{t}$. The satisfaction relation $\models$ is extended to formulas as usual. As an abbreviation we sometimes identify an interpretation ω with its *complete conjunction*, i. e., if $a_1, \ldots, a_n \in \mathsf{At}$ are those atoms that are assigned t by ω and $a_{n+1}, \ldots, a_m \in \mathsf{At}$ are those propositions that are assigned f by ω we identify ω by $a_1 \ldots a_n \overline{a_{n+1}} \ldots \overline{a_m}$ (or any permutation of this). For example, the interpretation ω on $\{a, b, c\}$ with $\omega(a) = \omega(c) = \mathsf{t}$ and $\omega(b) = \mathsf{f}$ is abbreviated by $a\overline{b}c$. For $\Phi \subseteq \mathcal{L}(\mathsf{At})$ we also define $\omega \models \Phi$ if and only if $\omega \models \phi$ for every $\phi \in \Phi$. We define the set of models $\mathsf{Mod}(X) = \{\omega \in \Omega(\mathsf{At}) \mid \omega \models X\}$ for every formula or set of formulas X. A formula or set of formulas X_1 *entails* another formula or set of formulas X_2, denoted by $X_1 \vdash X_2$, if $\mathsf{Mod}(X_1) \subseteq \mathsf{Mod}(X_2)$.

For $\alpha, \beta \in \mathcal{L}(\mathsf{At})$ the object $\delta = (\beta|\alpha)$ is called a *conditional* and models the defeasible rule "If α then usually β". A *conditional belief base* Δ is a set of conditionals. For every conditional $(\beta|\alpha)$ define $\Lambda_{(\beta|\alpha)} : \Omega \rightarrow \{v, f, n\}$ via

$$\Lambda_{(\beta|\alpha)}(\omega) = \begin{cases} v & \text{if } \omega \models \alpha \wedge \beta \\ f & \text{if } \omega \models \alpha \wedge \neg\beta \\ n & \text{if } \omega \models \neg\alpha \end{cases}$$

If $\Lambda_{(\beta|\alpha)}(\omega) = v$, we say that ω *verifies* $(\beta|\alpha)$, if $\Lambda_{(\beta|\alpha)}(\omega) = f$, we say that ω *falsifies* $(\beta|\alpha)$, and if $\Lambda_{(\beta|\alpha)}(\omega) = n$, we say that ω *is not applicable to* $(\beta|\alpha)$.

The central concepts for lexicographical inference [Lehmann, 1995] are as follows.

Definition 1. Let Δ be a conditional belief base.

1. A conditional δ is *tolerated* by Δ if there is $\omega \in \Omega$ with $\Lambda_\delta(\omega) = v$ and $\Lambda_{\delta'}(\omega) \neq f$ for all $\delta' \in \Delta$.
2. The *Z-Partitioning* $Z(\Delta)$ of Δ is recursively defined as follows:
   (a) $Z(\Delta) = (\Delta)$ if every $\delta \in \Delta$ is tolerated by Δ
   (b) $Z(\Delta) = (\Delta^0, \Delta^1, \ldots, \Delta^k)$ if Δ^0 is the set of all $\delta \in \Delta$ that are tolerated by Δ and $Z(\Delta \setminus \Delta^0) = (\Delta^1, \ldots, \Delta^k)$.

Note that the Z-partitioning is not well-defined for every conditional belief base (e. g., it is undefined for $\Delta = \{(\neg a|a)\}$). We say a conditional belief base Δ is *consistent* iff $Z(\Delta)$ is well-defined, i. e., that $Z(\Delta)$ exists as defined in Definition 1. For consistent Δ with Z-Partitioning $Z(\Delta) = (\Delta^0, \Delta^1, \ldots, \Delta^k)$ define

$$\xi^i_\Delta(\omega) = \{\delta \in \Delta^i \mid \Lambda_\delta(\omega) = f\}$$

for all $i = 1, \ldots, n$. In other words, $\xi^i_\Delta(\omega)$ is the set of conditionals in the i-th partition of $Z(\Delta)$ that are falsified by ω.

For $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{N}$ we write $(a_1, \ldots, a_n) <^{\text{lex}} (b_1, \ldots, b_n)$ if there is $k \in \{1, \ldots, n\}$ such that $a_k < b_k$ and $a_i = b_i$ for all $i = k+1, \ldots, n$. We also write $(a_1, \ldots, a_n) =^{\text{lex}} (b_1, \ldots, b_n)$ if $a_i = b_i$ for all $i = 1, \ldots, n$ and define $\leq^{\text{lex}}$ accordingly.

For the definition of lexicographic inference we use the characterisation from [Haldimann *et al.*, 2025].

Definition 2. Let Δ be a consistent conditional belief base with $Z(\Delta) = (\Delta^0, \Delta^1, \ldots, \Delta^n)$.

- For $\omega, \omega' \in \Omega$ we write $\omega \prec^{\text{lex}}_\Delta \omega'$ iff $(|\xi^0_\Delta(\omega)|, \ldots, |\xi^n_\Delta(\omega)|) <^{\text{lex}} (|\xi^0_\Delta(\omega')|, \ldots, |\xi^n_\Delta(\omega')|)$. Let $\preceq^{\text{lex}}_\Delta$ be defined accordingly.
- For formulas $\phi, \psi \in \mathcal{L}(\mathsf{At})$ we write $\phi \mathrel{|\!\sim^{\text{lex}}_\Delta} \psi$ iff for all $\omega' \in \mathsf{Mod}(\phi \wedge \neg\psi)$ there is $\omega \in \mathsf{Mod}(\phi \wedge \psi)$ with $\omega \prec^{\text{lex}}_\Delta \omega'$.

In other words, a formula ψ can be lexicographically inferred by a formula ϕ, given the conditional knowledge in Δ, iff for all interpretations ω' that falsify the conditional $(\psi|\phi)$, we can find another interpretation ω that verifies $(\psi|\phi)$ and is more plausible according to $\prec^{\text{lex}}_\Delta$. Moreover, an interpretation ω is more plausible than an interpretation ω', according to $\prec^{\text{lex}}_\Delta$, iff ω falsifies strictly less conditionals of some "level" i in $Z(\Delta)$ and falsifies the same number of conditionals in all higher "levels" of $Z(\Delta)$. Intuitively, this means that ω is more compatible with the less exceptional conditionals of Δ.

Example 1. We consider the classical penguins example, see, e. g., [Lehmann and Magidor, 1989; Haldimann *et al.*, 2025]. We consider the signature $\mathsf{At} = \{B, P, F, W\}$ with

B ="(we see a) bird" P ="penguin"
F ="flying" W ="wings"

and the conditional knowledge base

$$\Delta = \{(B|P), (F|B), (\neg F|P), (W|B)\}$$

that models that penguins are usually birds, birds usually fly, penguins usually do not fly, and birds usually have wings.

Note that the interpretation $B\overline{P}FW$ verifies both $(F|B)$ and $(W|B)$ and does not falsify $(B|P)$ and $(\neg F|P)$. Moreover, the interpretation $BP\overline{F}W$ verifies both $(B|P)$ and $(\neg F|P)$. This gives us the Z-Partitioning of Δ as $Z(\Delta) = (\Delta^0, \Delta^1)$ with

$$\Delta^0 = \{(F|B), (W|B)\} \qquad \Delta^1 = \{(B|P), (\neg F|P)\}$$

Then we get, for example, that

$$\xi^0_\Delta(BPFW) = \emptyset \qquad \xi^0_\Delta(BPF\overline{W}) = \{(W|B)\}$$
$$\xi^1_\Delta(BPFW) = \{(\neg F|P)\} \qquad \xi^1_\Delta(BPF\overline{W}) = \{(\neg F|P)\}$$

and therefore $BPFW \prec^{\text{lex}}_\Delta BPF\overline{W}$. The complete order $\prec^{\text{lex}}_\Delta$ over $\Omega(\Delta)$ is given in Table 1. We can now infer, for example, that penguins usually have wings, i. e., $P \mathrel{|\!\sim^{\text{lex}}_\Delta} W$, since for all $\omega' \in \mathsf{Mod}(P \wedge \neg W)$ there is $\omega \in \mathsf{Mod}(P \wedge W)$ with $\omega \prec^{\text{lex}}_\Delta \omega'$. In particular, we have $BP\overline{F}W \prec^{\text{lex}}_\Delta \omega'$ for all $\omega' \in \mathsf{Mod}(P \wedge \neg W)$.

6	$\overline{B}PF\overline{W}, \overline{B}PFW$
5	$BPF\overline{W}$
4	$BPFW, \overline{B}P\overline{FW}, \overline{B}P\overline{F}W$
3	$BP\overline{FW}, B\overline{P}F\overline{W}$
2	$B\overline{P}FW, B\overline{P}F\overline{W}, BP\overline{F}W$
1	$\overline{B}PFW, \overline{B}PFW, \overline{B}PFW, \overline{B}P\overline{FW}, B\overline{P}FW$

Table 1: The order $\prec_\Delta^{\text{lex}}$ from Example 1, structured into the corresponding layers, i. e., we have $\omega' \preceq_\Delta^{\text{lex}} \omega$ and $\omega \preceq_\Delta^{\text{lex}} \omega'$ for all ω and ω' in the same layer and $\omega' \prec_\Delta^{\text{lex}} \omega$ if ω' is in layer i and ω' is in layer j and $i < j$.

As hinted at in the above example, we briefly note here that the "for all" and "there is" parts of the definition of lexicographic inference above (see Definition 2) can be swapped without changing the meaning. This representation will be used to show the correctness of our algorithms later.

Proposition 1. *Let Δ be a consistent conditional belief base and $\phi, \psi \in \mathcal{L}(\mathsf{At})$. Then $\phi \mathrel{|\!\sim}^{lex}_\Delta \psi$ iff there is $\omega \in \mathsf{Mod}(\phi \wedge \psi)$ such that $\omega \prec^{lex}_\Delta \omega'$ for all $\omega' \in \mathsf{Mod}(\phi \wedge \neg\psi)$.*

In other words, a formula ψ can be lexicographically inferred by a formula ϕ, given the conditional knowledge in Δ, iff there is an interpretation ω that verifies $(\psi|\phi)$ and is more plausible, according to $\prec_\Delta^{\text{lex}}$, than all interpretations ω' that falsify the conditional $(\psi|\phi)$.

In this paper, we are interested in the following computational problem:

LEXINFERENCE **Input**: Conditional belief base Δ, formulas $\phi, \psi \in \mathcal{L}(\mathsf{At})$
Output: YES iff $\phi \mathrel{|\!\sim}^{\text{lex}}_\Delta \psi$

3 The Computational Complexity of Lexicographic Inference, Revisited

It has been shown in [Cayrol *et al.*, 1998], see also the discussion in [Eiter and Lukasiewicz, 2000], that LEXINFERENCE is P^{NP}-complete. The class P^{NP}, also denoted as Δ_2^{P}, is the class of decision problems that can be solved by an algorithm that runs in deterministic polynomial time and has access to an NP-oracle, i. e., a polynomial-time algorithm that can instantly solve an NP-complete problem in each step, cf. [Papadimitriou, 1994]. In the following, we make some more concrete observations on certain sub-problems of LEXINFERENCE that allows us to devise an effective algorithm in the next section.

We first note that $Z(\Delta)$ can be determined by successive calls to an NP-oracle.

Proposition 2. *Let δ be a conditional and Δ a consistent conditional belief base.*

1. *Deciding whether δ is tolerated by Δ is NP-complete.*
2. *Determining $Z(\Delta)$ can be accomplished in polynomial time with at most $|\Delta|(|\Delta|+1)/2$ calls to an NP-oracle.*

Central to our algorithms is the following observation.

Proposition 3. *Given $\phi \in \mathcal{L}$, $\omega \in \Omega$, and the Z-partitioning $Z(\Delta)$ of a consistent conditional belief base Δ, deciding whether there is $\omega' \in \Omega$ with $\omega' \in \mathsf{Mod}(\phi)$ and $\omega' \preceq^{lex}_\Delta \omega$ is NP-complete.*

Observe that the statement of the previous proposition remains true if we use $\prec_\Delta^{\text{lex}}$ instead of $\preceq_\Delta^{\text{lex}}$.

Algorithm 1 Algorithm FastLexi

Input: Conditional belief base Δ, formulas $\phi, \psi \in \mathcal{L}(\mathsf{At})$
Output: YES iff $\phi \mathrel{|\!\sim}^{\text{lex}}_\Delta \psi$

```
1: Z = ComputeZ(Δ)
2: ω' ← null
3: while true do
4:     ω ← SmallerModel(φ ∧ ψ, ω', Z, Δ, ≺^lex_Δ)
5:     if ω = null then
6:         return NO
7:     ω' ← SmallerModel(φ ∧ ¬ψ, ω, Z, Δ, ⪯^lex_Δ)
8:     if ω' = null then
9:         return YES
```

4 The Algorithms

Our first algorithm FastLexi is depicted in Algorithm 1. It takes a conditional belief base Δ and two formulas ϕ and ψ as input and starts with computing the Z-partitioning of Δ, using a sub-routine ComputeZ that will be implemented via SAT-encodings in Section 5 and is based on the algorithm sketched in the proof of Proposition 2. In the **while**-loop in lines 3–9, the algorithm then iteratively determines models of $\phi \wedge \psi$ and $\phi \wedge \neg\psi$ that are increasingly smaller wrt. the lexicographic order. More specifically, the sub-routine SmallerModel$(\gamma, \omega, Z, \Delta, \odot)$ with $\odot \in \{\prec_\Delta^{\text{lex}}, \preceq_\Delta^{\text{lex}}\}$ finds an interpretation $\omega' \in \mathsf{Mod}(\gamma)$ such that $\omega' \odot \omega$ (or returns "null" if no such ω' can be found). If $\omega =$ null then SmallerModel$(\gamma, \omega, Z, \Delta, \odot)$ returns just any model of γ. The sub-routine SmallerModel will also be implemented using SAT-encodings in Section 5. Here, we only note that according to Proposition 3, SmallerModel can be implemented using a single NP oracle call.

If the algorithm FastLexi terminates in line 6, this means we have not found an interpretation ω that is a model of $\phi \wedge \psi$ and strictly more plausible as the model ω' of $\phi \wedge \neg\psi$. In this case the algorithm returns NO. If the algorithm terminates in line 9, this means we have not found an interpretation ω' that is a model of $\phi \wedge \neg\psi$ and at least as plausible as the model ω of $\phi \wedge \psi$. In this case, the algorithm returns YES.

The formal correctness of the algorithm and its theoretical runtime are given by the following two theorems.

Theorem 1. *Assume Δ is consistent. Algorithm FastLexi returns* YES *if and only if $\phi \mathrel{|\!\sim}^{lex}_\Delta \psi$.*

Theorem 2. *Assume Δ is consistent. Algorithm FastLexi runs in polynomial time and makes at most $|\Delta|(|\Delta|+1)/2 + 2|\Delta|$ calls to an NP-oracle.*

Note that the additional assumption in the above two theorems, namely Δ being consistent, is a technical one. In fact, we can determine the consistency of Δ while the Z-Partitioning of Δ is computed in the sub-routine ComputeZ without additional effort (see below), and abort the algorithm

Algorithm 2 Algorithm FastLexiMax

Input: Conditional belief base Δ, formulas $\phi, \psi \in \mathcal{L}(\mathsf{At})$
Output: YES iff $\phi \mathrel{|\!\sim}^{\text{lex}}_{\Delta} \psi$

1: $Z = \mathsf{ComputeZ}(\Delta)$
2: $\omega \leftarrow \mathsf{SmallestModel}(\phi \wedge \psi, \Delta, Z)$
3: **if** $\mathsf{SmallerModel}(\phi \wedge \neg\psi, \omega, Z, \Delta, \preceq^{\text{lex}}_{\Delta}) = \text{null}$ **then**
4: **return** YES
5: **return** NO

if Δ is found to be inconsistent. We omitted this aspect from Algorithm 1 (and also for our second algorithm FastLexiMax below) to avoid the additional case differentiation.

Algorithm FastLexi iteratively seeks more plausible (wrt. $\prec^{\text{lex}}_{\Delta}$) interpretations and always switches between interpretations verifying or violating the query under consideration. However, Proposition 1 also allows for another approach to check for a valid inference, namely by first determining some *minimal* interpretation that verifies the query and then checking whether there is another interpretation that violates it and is at least as plausible as the previously found query. Algorithm FastLexiMax, depicted in Algorithm 2, formalises this idea. Here, the sub-routine $\mathsf{SmallestModel}(\gamma, \Delta, Z)$ finds a $\omega \in \mathsf{Mod}(\gamma)$ such that there is no $\omega' \in \mathsf{Mod}(\gamma)$ with $\omega' \prec^{\text{lex}}_{\Delta} \omega$. As we will see in the next section, $\mathsf{SmallestModel}(\gamma, \Delta, Z)$ can be implemented through an encoding as a weighted MaxSAT problem and a single call to a (weighted) MaxSAT solver.

Theorem 3. *Assume Δ is consistent. Algorithm FastLexiMax returns* YES *if and only if $\phi \mathrel{|\!\sim}^{lex}_{\Delta} \psi$.*

Theorem 4. *Assume Δ is consistent. Algorithm FastLexiMax runs in polynomial time and makes at most $|\Delta|(|\Delta|+1)/2 + |\Delta| + 1$ calls to an NP-oracle.*

The above theorem assumes that SmallestModel requires $|\Delta|$ calls to an NP-oracle. Since we will encode SmallestModel as a weighted MaxSAT-problem and MaxSAT-solvers use more sophisticated search procedures than linear search, this value can be regarded as a very rough upper bound. Still, from a theoretical point of view, FastLexiMax is already more effective than FastLexi (compare Theorems 2 and 4).

5 Encodings for ComputeZ, SmallerModel, and SmallestModel

We first implement the sub-routines ComputeZ and SmallerModel from Algorithms 1 and 2 using SAT-solving technology [Biere *et al.*, 2021] and encode the underlying decision problems as satisfiability problems. For the rest of this section, let $\Delta = \{\delta_1, \ldots, \delta_m\}$ be a conditional belief base with $\delta_i = (\beta_i|\alpha_i)$ for $i = 1, \ldots, m$, and let $\phi \mathrel{|\!\sim}^{\text{lex}}_{\Delta} \psi$ be the query under consideration.

We first consider ComputeZ. Central to computing the Z-Partitioning of Δ is the question of tolerance. Recall that a conditional δ is *tolerated* by Δ if there is $\omega \in \Omega$ with $\Lambda_\delta(\omega) = v$ and $\Lambda_{\delta'}(\omega) \neq f$ for all $\delta' \in \Delta$. Proposition 2, item 1, noted that this question can be answered with a single NP-oracle call, i. e., using a single query to a SAT-solver. In fact, it can be easily seen that $\delta = (\beta|\alpha)$ is *tolerated* by Δ if

$$\begin{aligned}&\Phi_{\delta,\Delta}\\ &= \alpha \wedge \beta \wedge (\neg\alpha_1 \vee \alpha_1 \wedge \beta_1) \wedge \ldots \wedge (\neg\alpha_m \vee \alpha_m \wedge \beta_m)\\ &\equiv \alpha \wedge \beta \wedge (\neg\alpha_1 \vee \beta_1) \wedge \ldots \wedge (\neg\alpha_m \vee \beta_m)\end{aligned}$$

is satisfiable. Given an appropriate encoding of $\Phi_{\delta,\Delta}$ in conjunctive normal form, its satisfiability can be decided by a single call to a SAT-solver. Following the algorithm sketch in the proof of Proposition 2, item 2, we can implement ComputeZ using at most $|\Delta|(|\Delta|+1)/2$ calls to the SAT-solver. More concretely, we can check for each $\delta \in \Delta$, whether δ is tolerated by Δ, collect the positive instances in the first partition Δ^0, and recursively continue with the remaining conditionals. Moreover, we can also test consistency of Δ while determining $Z(\Delta)$: if at any point we cannot find a conditional that is tolerated by the remaining conditionals, we can cancel the computation as then Δ is inconsistent.

We now turn to the encoding of SmallerModel which can be represented as a single satisfiability problem according to Proposition 3. In order to encode SmallerModel we require the use of *cardinality constraints* [Wynn, 2018; Bittner *et al.*, 2019], in particular *at-most-k*-constraints. For a set of variables X and a natural number $k \in \mathbb{N}$ a constraint of the form $\text{at\_most}(X, k)$ is true if and only if at most k variables of X have the truth value t. A constraint $\text{at\_most}(X, k)$ can be naively represented through

$$\text{at\_most}(X, k) = \bigwedge_{X' \subseteq X, |X'| = |X| - k} \left(\bigvee_{x \in X'} \neg x \right)$$

For example, for $X = \{x_1, x_2, x_3, x_4\}$ and $k = 2$ we have

$$\begin{aligned}&\text{at\_most}(\{x_1, x_2, x_3, x_4\}, 2)\\ &= (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_4)\\ &\quad \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_3 \vee \neg x_4)\end{aligned}$$

Note that this naive encoding (also called *binomial encoding*) is inefficient in practice, since $\text{at\_most}(X, k)$ grows in general exponentially. There exist other encodings that make use of auxiliary variables and only grow polynomially in size, e. g., the sequential counter encoding [Sinz, 2005] (which is polynomial in the number of variables and the parameter k) or the approach of [Bailleux *et al.*, 2009] (which is only polynomial in the number of variables). We abstract from the concrete implementation of $\text{at\_most}(X, k)$ and only assume that it can be represented in polynomial size wrt. the number of variables in the signature, see [Wynn, 2018; Bittner *et al.*, 2019] for some more discussion. In our implementation (see Section 6), we used a variant of the totalizer [Bailleux and Boufkhad, 2003] cardinality constraint encoding called kmtotalizer [Morgado *et al.*, 2014].

We consider first a query of the form $\mathsf{SmallerModel}(\gamma, \omega', Z, \Delta, \prec^{\text{lex}}_{\Delta})$ (see line 4 of Algorithm 1) for an arbitrary formula γ, the difference when considering $\preceq^{\text{lex}}_{\Delta}$ instead of $\prec^{\text{lex}}_{\Delta}$ (for line 7 of Algorithm 1) will be discussed afterwards. Let the Z-Partitioning of Δ be given by $Z(\Delta) = (\Delta^0, \ldots, \Delta^k)$. Observe first that given the

interpretation ω', the numbers $|\xi^0_\Delta(\omega')|, \ldots, |\xi^k_\Delta(\omega')|$ can be determined in polynomial time before encoding the problem.

Now our encoding $\Phi_{\gamma,\omega',\Delta,Z}$ for the sub-routine $\mathsf{SmallerModel}(\gamma, \omega', Z, \Delta, \prec^{\text{lex}}_\Delta)$ consists of five components:

$$\Phi_{\gamma,\omega',\Delta,Z} = \Phi^1_\gamma \wedge \Phi^2_\Delta \wedge \Phi^3_{\omega',\Delta,Z} \wedge \Phi^4_{\omega',\Delta,Z} \wedge \Phi^5_Z$$

which we define in the following.

1. Φ^1_γ represents the fact that we seek a model of γ, i. e., γ must be evaluated to true: $\Phi^1_\gamma = \gamma$
2. Φ^2_Δ encodes which conditionals are falsified by the model we are seeking. For that, let $F_1, \ldots, F_m$ be auxiliary variables, where F_i denotes that the conditional $\delta_i \in \Delta$ is falsified by the model we are seeking, for $i = 1, \ldots, m$. Therefore, we encode
$$\Phi^2_\Delta = \bigwedge_{i=1}^{m} ((\alpha_i \wedge \neg\beta_i) \Leftrightarrow F_i)$$
3. $\Phi^3_{\omega',\Delta,Z}$ counts the number of falsified conditionals in each layer of the Z-Partitioning and compares that number to the corresponding one for ω'. For that, let $LE_0, \ldots, LE_k$ be auxiliary variables, where LE_j denotes that the model under consideration falsifies fewer or equally many conditionals in Δ^i than ω'. We write
$$\Phi^3_{\omega',\Delta,Z} = \bigwedge_{j=0}^{k} \Big(\text{at\_most}(\{F_i \mid \delta_i \in \Delta^j\}, |\xi^j_\Delta(\omega')|) \Leftrightarrow LE_j\Big)$$
4. Similarly, $\Phi^4_{\omega',\Delta,Z}$ checks for each layer of $Z(\Delta)$ when the current model falsifies strictly fewer conditionals than ω'. For that, let $L_0, \ldots, L_k$ be auxiliary variables, where L_i denotes that the model under consideration falsifies strictly fewer conditionals in Δ^i than ω'. We write
$$\Phi^4_{\omega',\Delta,Z} = \bigwedge_{j=0}^{k} \Big(\text{at\_most}\big(\{F_i \mid \delta_i \in \Delta^j\}, |\xi^j_\Delta(\omega')| - 1\big) \Leftrightarrow L_j\Big)$$
5. Φ^5_Z finally encodes that the model we seek is lexicographically smaller than ω', i. e., there is $h \in \{0, \ldots, k\}$ such that L_h is true and LE_i is true for all $i = h+1, \ldots, k$:
$$\Phi^5_Z = \bigvee_{h=0}^{k} \left(L_h \wedge \bigwedge_{i=h+1}^{k} LE_i\right)$$

It should be clear that the size of $\Phi_{\gamma,\omega',Z}$ is polynomial in Δ, $Z(\Delta)$, γ, and the size of the underlying signature (given that we use polynomial encodings for the at_most-constraints).

The following theorem shows that $\Phi_{\gamma,\omega',Z}$ is a suitable SAT-encoding (given an appropriate transformation into conjunctive normal form) of the sub-routine $\mathsf{SmallerModel}$.

Theorem 5. *$\Phi_{\gamma,\omega',\Delta,Z}$ is satisfiable if and only if there is an interpretation ω with $\omega \in \mathsf{Mod}(\gamma)$ and $\omega \prec^{lex}_\Delta \omega'$.*

Since in case of satisfiability, SAT-solvers can return a model (a *witness*) of the satisfiability, we can implement $\mathsf{SmallerModel}$ by either extracting that model (in the original signature) or return null. Also note that we can encode $\mathsf{SmallerModel}(\gamma, \omega', Z, \Delta, \preceq^{\text{lex}}_\Delta)$ instead of $\mathsf{SmallerModel}(\gamma, \omega', Z, \Delta, \prec^{\text{lex}}_\Delta)$ by simply changing Φ^5_Z to
$$\hat{\Phi}^5_Z = \Phi^5_Z \vee \bigwedge_{i=0}^{k} LE_i$$
to accommodate the case that ω' and the model we seek are lexicographically equivalent.

We now turn to the encoding of $\mathsf{SmallestModel}(\gamma, \Delta, Z)$ from Algorithm 2, which we will encode as a weighted MaxSAT problem, i. e., a pair (Φ, Ψ) where Φ is a propositional formula encoding the *hard constraints* and $\Psi = \{w_1 : l_1, \ldots, w_k : l_k\}$ is a set of weighted propositional literals (the weight of literal l_i is $w_i \in \mathbb{N}$, for $i = 1, \ldots, k$) that encodes the *soft constraints* (note that we only need this simple form of weighted MaxSAT problems). A solution to (Φ, Ψ) is then a propositional interpretation ω that satisfies Φ such that the value $\sum_{\omega \models l_i} w_i$ is maximal. Now our encoding $\Phi^\dagger_{\gamma,\Delta,Z}$ of the hard constraints for $\mathsf{SmallestModel}(\gamma, \Delta, Z)$ is defined via $\Phi^\dagger_{\gamma,\Delta,Z} = \Phi^1_\gamma \wedge \Phi^2_\Delta$ with Φ^1_γ and Φ^2_Δ as above. Define now $Z(i) = j$ if $\delta_i \in \Delta^j$ for $i = 1, \ldots, m$. Then the encoding $\Psi^\dagger_{\Delta,Z}$ of the hard constraints for $\mathsf{SmallestModel}(\gamma, \Delta, Z)$ is defined via $\Psi^\dagger_{\Delta,Z} = \{w_{Z(i)} : \neg F_i \mid i = 1, \ldots, m\}$ with $w_0 = 1$ and $w_j = (|\Delta^{j-1}| + 1)w_{j-1}$ for $j = 1, \ldots, k$. Be reminded that F_i is true if the model under consideration falsifies conditional δ_i. With the above definition, each non-falsification of a conditional in Δ^j is rewarded with weight w_j, for $j = 0, \ldots, k$. Furthermore, the weight of a conditional in Δ^j is larger than the sum of all weights of conditionals in $\Delta^0, \ldots, \Delta^{j-1}$, so not falsifying conditionals in higher "levels" of $Z(\Delta)$ is more beneficial for obtaining a large value of the solution than not falsifying conditionals in lower "levels" of $Z(\Delta)$. We obtain the following correctness result.

Theorem 6. *If $\hat{\omega} : \mathsf{At} \cup \{F_1, \ldots, F_m\} \to \{\mathsf{t}, \mathsf{f}\}$ is a solution for $(\Phi^\dagger_{\gamma,\Delta,Z}, \Psi^\dagger_{\Delta,Z})$ then for $\omega : \mathsf{At} \to \{\mathsf{t}, \mathsf{f}\}$ with $\omega(x) = \hat{\omega}(x)$ for all $x \in \mathsf{At}$, we have $\omega \in \mathsf{Mod}(\gamma)$ and there is no $\omega' \in \mathsf{Mod}(\gamma)$ with $\omega' \prec^{lex}_\Delta \omega$.*

6 Experimental Evaluation

In this section, we present the results of an experimental analysis comparing the performance of the two proposed algorithms, FastLexi and FastLexiMax, with LexInf from [Haldimann *et al.*, 2025]. In their work, Haldimann *et al.* (2025) also compared their algorithm to a naive Java implementation of lexicographic inference named LexJ. However, as LexJ did not perform competitively, we refrain from including it in this evaluation.

The remainder of this section is structured as follows. In Section 6.1, the experimental setup is described, including the implementation, the used datasets, the performance metrics, and the experimental procedure. Subsequently, Section 6.2 presents the results of the experimental analysis.

6.1 Setup

Implementation We implemented the algorithms FastLexi and FastLexiMax in Python, using the SAT toolkit PySAT [Ignatiev *et al.*, 2024]. FastLexi is parameterizable with different SAT solvers, including CaDiCaL (rel-1.9.5) [Biere *et al.*, 2024], Glucose (4.2.1) [Audemard and Simon, 2009], MapleLCMDistChronoBT [Kochemazov *et al.*, 2019], and MiniSAT (2.2) [Sorensson and Een, 2005]. The MaxSAT version FastLexiMax uses the RC2 [Ignatiev *et al.*, 2018] solver. The implementation is open source and available on GitHub[2]. For the LexInf solver, we implemented a wrapper that offers an identical solver interface to our solvers. We also replaced the LexInf parser for belief bases and queries with our implementation, as the original parser was unable to handle instances with a large number of conditionals.

Datasets In total, we used 3 different data sets for the evaluation: (1) *CLKR-PS005*, (2) *CLKR-Buster*, (3) *AwAKB*.

The *CLKR-PS005* [Beierle *et al.*, 2024] dataset contains 100 randomly generated belief bases and 1000 (10 per belief base) queries for each combination of signature size $|\Sigma|$ (ranging from 6 to 120) and belief base size $|\Delta|$ (ranging from 6 to 160), resulting in a total of 2700 belief bases and 27000 queries. The dataset *CLKR-PS005* is the same dataset that was used in the evaluation of [Haldimann *et al.*, 2025]. For more information on the random belief base generator see [von Berg *et al.*, 2024]

The dataset *CLKR-Buster* was created using the same generator as the previous dataset, but with larger signatures (200-3000) and a larger number of conditionals (200-3000). The data set consists of 270 belief bases, each with 10 queries, for a total of 2700 queries.

The *AwAKB* [Kuhlmann *et al.*, 2022] comprises 1920 knowledge bases derived from the *Animals with Attributes* (AwA) dataset, a widely used benchmark in machine learning. AwA contains 50 animal categories described by 85 binary attributes. The authors applied the Apriori algorithm [Agrawal *et al.*, 1993] to extract association rules from AwA and then treated each mined rule as a propositional implication (rules were restricted to contain only 4 literals). We adapted the data set by interpreting the implications as conditionals. Since the number of conditionals varies greatly, we have further grouped the dataset into 5 subsets based on the number of conditionals: (1) *tiny*, (2) *small*, (3) *medium*, (4) *large*, and (5) *huge*. All subsets contain 336 instances.

Metrics We compared the performance of all algorithms in terms of (1) Belief Base Coverage (Cov_b), (2) Query Coverage (Cov_q), and (3) Penalized Average Runtime 2 (PAR2).

The belief base coverage Cov_b is the ratio between completely solved belief bases and the total number of belief bases. A belief base is fully solved when all associated queries have been solved. Similarly, the query coverage Cov_q is defined as the ratio between solved queries and the total number of solved queries.

The PAR2 score is defined as the average runtime (in seconds) of solved instances, plus twice the timeout value, allowing runtime to be considered while still setting a strong focus on instance coverage. For example, for a 300-second timeout, the runtime of unsolved instances is counted as $2 * 300 = 600$ seconds.

[2]https://github.com/aig-hagen/Luthor

	CLKR-PS005		
Algorithm	**Cov_b**	**Cov_q**	**PAR2**
$\text{FastLexi}_{\text{CaDicaL}}$	1.0	1.0	0.743
$\text{FastLexi}_{\text{Glucose}}$	1.0	1.0	**0.707**
$\text{FastLexi}_{\text{Maple}}$	1.0	1.0	0.724
$\text{FastLexi}_{\text{MiniSAT}}$	1.0	1.0	0.721
FastLexiMax	1.0	1.0	0.733
LexInf	0.919	0.974	19.282
VBS	1.0	1.0	0.691
	CLKR-Buster		
Algorithm	**Cov_b**	**Cov_q**	**PAR2**
$\text{FastLexi}_{\text{CaDicaL}}$	1.0	1.0	10.307
$\text{FastLexi}_{\text{Glucose}}$	0.926	0.992	15.983
$\text{FastLexi}_{\text{Maple}}$	0.989	0.999	10.205
$\text{FastLexi}_{\text{MiniSAT}}$	0.844	0.980	24.514
FastLexiMax	1.0	1.0	**5.723**
LexInf	0.778	0.964	34.444
VBS	1.0	1.0	5.723

Table 2: Overview of belief base coverage (**Cov_b**), query coverage (**Cov_q**), and penalized average runtime (**PAR2**) on the **CLKR-PS005** and **CLKR-Buster** benchmarks. Subscripts of FastLexi denote the SAT solver used. Best results are shown in **bold**.

Procedure In our experiments, we investigated 6 different algorithms, consisting of the SAT version of FastLexi, parameterized with four different SAT solvers (CaDiCaL, Glucose, MapleLCMDistChronoBT, and MiniSAT), as well as the MaxSAT version FastLexiMax and LexInf. Each algorithm was executed for each belief base and its 10 queries. We set a 5 minutes (300 seconds) timeout for each query. If an instance was not solved within this limit, it was terminated, marked as timed out, and its runtime set to the timeout value. In addition, we computed a *virtual best solver* (VBS) by selecting the best runtime for each query. All experiments were conducted on a machine running Ubuntu 20.04 with an Intel Xeon E5 3.4 GHz CPU and 192 GB of RAM.

6.2 Results

CLKR Table 2 shows the results in terms of belief base coverage (Cov_b), query coverage (Cov_q), and penalized average runtime (PAR2) on the CLKR-PS005 and CLKR-Buster benchmarks. Starting with the CLKR-PS005 benchmark, the results show that all versions of FastLexi (FastLexiMax, $\text{FastLexi}_{\text{CaDicaL}}$, $\text{FastLexi}_{\text{Glucose}}$, $\text{FastLexi}_{\text{Maple}}$, and $\text{FastLexi}_{\text{MiniSAT}}$) achieve perfect query coverage and, therefore, perfect belief base coverage. Furthermore, the results show that all versions yield comparable PAR2 scores, with $\text{FastLexi}_{\text{Glucose}}$ achieving the best result of 0.707. The remaining FastLexi versions attain scores between 0.721 ($\text{FastLexi}_{\text{MiniSAT}}$) and 0.743 ($\text{FastLexi}_{\text{CaDicaL}}$). The LexInf solver achieves query coverage of 0.974 and belief base coverage of 0.919. LexInf also performs worse in terms of PAR2 with a score of 19.282.

For the CLKR-Buster benchmark, which has significantly more conditionals and a larger signature, only two

	Tiny			Small			Medium			Large			Huge		
Algorithm	**Cov$_b$**	**Cov$_q$**	**PAR2**	**Cov$_b$**	**Cov$_q$**	**PAR2**	**Cov$_b$**	**Cov$_q$**	**PAR2**	**Cov$_b$**	**Cov$_q$**	**PAR2**	**Cov$_b$**	**Cov$_q$**	**PAR2**
FastLexi$_{\text{CaDicaL}}$	1.0	1.0	0.445	1.0	1.0	0.634	1.0	1.0	1.781	1.0	1.0	10.595	0.679	0.779	215.055
FastLexi$_{\text{Glucose}}$	1.0	1.0	0.443	1.0	1.0	0.628	1.0	1.0	1.724	1.0	1.0	10.097	0.702	0.781	210.162
FastLexi$_{\text{Maple}}$	1.0	1.0	**0.436**	0.994	0.999	1.074	0.982	0.998	4.614	0.836	0.983	25.559	0.399	0.738	231.602
FastLexi$_{\text{MiniSAT}}$	1.0	1.0	0.441	1.0	1.0	0.628	1.0	1.0	1.725	1.0	1.0	10.005	0.690	0.780	210.228
FastLexiMax	1.0	1.0	0.444	1.0	1.0	**0.585**	1.0	1.0	**1.292**	1.0	1.0	**6.319**	0.780	0.790	**177.979**
LexInf	1.0	1.0	0.552	0.577	0.623	227.254	0.063	0.180	494.140	0.006	0.149	514.695	0.0	0.106	548.393
VBS	1.0	1.0	0.433	1.0	1.0	0.582	1.0	1.0	1.289	1.0	1.0	6.313	0.780	0.790	177.841

Table 3: Overview of belief base coverage (**Cov**$_b$), query coverage (**Cov**$_q$), and penalized average runtime (**PAR2**) on the Animals with Attributes (AwA) dataset, grouped by the number of conditionals into five subsets: (1) **tiny**, (2) **small**, (3) **medium**, (4) **large**, and (5) **huge**. Subscripts of FastLexi denote the SAT solver used. Best results are shown in **bold**.

algorithms achieve full coverage: FastLexi$_{\text{CaDicaL}}$ and FastLexiMax. The other FastLexi variants lie between 0.844 (FastLexi$_{\text{MiniSAT}}$) and 0.989 (FastLexi$_{\text{Maple}}$) for Cov$_b$ and between 0.980 (FastLexi$_{\text{MiniSAT}}$) and 0.999 (FastLexi$_{\text{Maple}}$) in terms of Cov$_q$ values. LexInf achieves the worst coverage values with a Cov$_b$ value of 0.778 and a Cov$_q$ value of 0.964. In terms of PAR2 scores, FastLexiMax achieves the best at 5.723 (on par with VBS), nearly twice as fast as the next best algorithm, FastLexi$_{\text{Maple}}$ (10.205), and closely followed by FastLexi$_{\text{CaDicaL}}$ (10.307). The LexInf algorithm shows the worst performance, with a PAR2 score of 34.444, which is noticeably worse than the worst of the FastLexi variants, FastLexi$_{\text{MiniSAT}}$, with a score of 24.514.

AwAKB Table 3 summarizes belief base coverage (Cov$_b$), query coverage (Cov$_q$), and penalized average runtime (PAR2) on the AwAKB dataset, grouped into *tiny*, *small*, *medium*, *large*, and *huge* subsets by the number of conditionals. On the tiny (7–73 conditionals) subset, all algorithms achieve perfect query and belief base coverage and show very similar runtimes. The best PAR2 is obtained by FastLexi$_{\text{Maple}}$ with 0.436, while the remaining variants fall into a very narrow range between 0.441 (FastLexi$_{\text{MiniSAT}}$) and 0.445 (FastLexi$_{\text{CaDicaL}}$). The LexInf algorithm likewise attains complete coverage, with a comparable PAR2 score of 0.552.

For the *small* subset (73–465 conditionals), coverage remains perfect for FastLexiMax, FastLexi$_{\text{CaDicaL}}$, FastLexi$_{\text{Glucose}}$, and FastLexi$_{\text{MiniSAT}}$, and is only marginally lower for FastLexi$_{\text{Maple}}$ with a Cov$_b$ of 0.994 and a Cov$_q$ value of 0.999. However, the results show a substantial decline in coverage for the LexInf algorithm, with a Cov$_b$ value of 0.577 and a Cov$_q$ value of 0.623. This fact is also reflected in the PAR2 scores: LexInf achieves a score of 227.254, which is significantly worse than all other algorithms. For comparison: The worst FastLexi version for this subset,FastLexi$_{\text{Maple}}$, achieves a PAR2 score of 1.074 and the best version, FastLexiMax, achieves a value of 0.585.

The trend observed so far is continued with the *medium* and large subsets.

Finally, the *huge* (9535–96097 conditionals) subset is the only one where none of the algorithms is able to solve all queries. Here, FastLexiMax achieves the overall best performance, combining the highest coverage (Cov$_b = 0.78$, Cov$_q = 0.79$) with the lowest PAR2 (177.979), very similar to the VBS (177.841). The remaining SAT-backed variants show lower belief base coverage (0.679–0.702) and higher PAR2 values (approximately 210–215). FastLexi$_{\text{Maple}}$ performs worst of all FastLexi variants in this subset, with a Cov$_b$ of 0.399 and a Cov$_q$ of 0.738, and the highest PAR2 (231.602). Nevertheless, these results are still comparable than those of LexInf on the small subset which are on average 160 times smaller then the huge instances. The LexInf algorithm is not able to fully solve a single belief base, indicated by a Cov$_b$ of 0 and a low Cov$_q$ of 0.106. Accordingly, LexInf achieves the highest PAR2 score of 548.393, approaching the worst possible score of 600.

Summary All evaluated FastLexi variants consistently outperform LexInf across all benchmarks. Furthermore, the performance differences increase with growing instance size, indicating that the advantages of FastLexi become more pronounced on larger problem instances. For instance, even for very large instances, such as those in the AwAKB huge subset, FastLexi maintains competitive performance, highlighting its efficiency and scalability. Moreover, differences can also be observed between the MaxSAT and SAT variants. Specifically, the FastLexiMax solver achieves the best results on nearly all benchmarks, yielding results similar to those of the VBS. However, among the SAT-based FastLexi variants, no single version consistently outperforms the others, as their performance varies across benchmarks.

7 Discussion and Conclusion

We presented two new algorithms for lexicographic inference from conditional knowledge bases. Our algorithms are based on compact SAT encodings of the lexicographic comparison criterion and make effective use of SAT and MaxSAT solving technology. In our experimental evaluation we showed, that both algorithms significantly outperform the state of the art, while the MaxSAT approach shows even more superior performance.

For future work we intend to apply the developed ideas to other inference operators for conditional knowledge bases, such as (disjunctive) rational closure [Lehmann and Magidor, 1989; Booth and Varzinczak, 2025], c-representations [Kern-Isberner, 2001], System Z [Pearl, 1990], or System W [Komo and Beierle, 2022].

Ethical Statement

There are no ethical issues.

References

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[Audemard and Simon, 2009] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, volume 9, pages 399–404, 2009.

[Bailleux and Boufkhad, 2003] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 108–122. Springer, 2003.

[Bailleux *et al.*, 2009] Olivier Bailleux, Yacine Boufkhad, and Olivier Roussel. New encodings of pseudo-Boolean constraints into CNF. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2009.

[Beierle *et al.*, 2024] Christoph Beierle, Jonas Haldimann, and Leon Schwarzer. CLKR – Conditional Logic and Knowledge Representation. *KI – Künstliche Intelligenz*, 38:61–67, 2024.

[Benferhat *et al.*, 1993] Salem Benferhat, Claudette Cayrol, Didier Dubois, Jérôme Lang, and Henri Prade. Inconsistency management and prioritized syntax-based entailment. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 640–647. Morgan Kaufmann, 1993.

[Biere *et al.*, 2021] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[Biere *et al.*, 2024] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024.

[Bittner *et al.*, 2019] Paul Maximilian Bittner, Thomas Thüm, and Ina Schaefer. SAT encodings of the At-Most-k constraint. In Peter Csaba Ölveczky and Gwen Salaün, editors, *Software Engineering and Formal Methods*, pages 127–144, Cham, 2019. Springer International Publishing.

[Booth and Varzinczak, 2025] Richard Booth and Ivan Varzinczak. On the disjunctive rational closure of a conditional knowledge base. *Artif. Intell.*, 348:104418, 2025.

[Cayrol *et al.*, 1998] Claudette Cayrol, Marie-Christine Lagasquie-Schiex, and Thomas Schiex. Nonmonotonic reasoning: From complexity to algorithms. *Ann. Math. Artif. Intell.*, 22(3-4):207–236, 1998.

[Eiter and Lukasiewicz, 2000] Thomas Eiter and Thomas Lukasiewicz. Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artif. Intell.*, 124(2):169–241, 2000.

[Haldimann *et al.*, 2025] Jonas Haldimann, Aron Spang, Lars-Phillip Spiegel, and Christoph Beierle. Implementing lexicographic inference using partial MaxSAT. In Kai Sauerwald and Matthias Thimm, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 18th European Conference, ECSQARU 2025, Hagen, Germany, September 23-26, 2025, Proceedings*, volume 16099 of *Lecture Notes in Computer Science*, pages 301–315. Springer, 2025.

[Heyninck *et al.*, 2023] Jesse Heyninck, Gabriele Kern-Isberner, Thomas Andreas Meyer, Jonas Philipp Haldimann, and Christoph Beierle. Conditional syntax splitting for non-monotonic inference operators. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 6416–6424. AAAI Press, 2023.

[Ignatiev *et al.*, 2018] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. RC2: a python-based MaxSAT solver. *MaxSAT Evaluation*, 2018:22, 2018.

[Ignatiev *et al.*, 2024] Alexey Ignatiev, Zi Li Tan, and Christos Karamanos. Towards universally accessible SAT technology. In *SAT*, pages 4:1–4:11, 2024.

[Kern-Isberner, 2001] Gabriele Kern-Isberner. *Conditionals in Nonmonotonic Reasoning and Belief Revision*. Number 2087 in Lecture Notes in Computer Science. Springer-Verlag, 2001.

[Kochemazov *et al.*, 2019] Stepan Kochemazov, Oleg Zaikin, Victor Kondratiev, and Alexander Semenov. Maplelcmdistchronobt-dl, duplicate learnts heuristic-aided solvers at the SAT race 2019. *Proceedings of SAT Race*, pages 24–24, 2019.

[Komo and Beierle, 2022] Christian Komo and Christoph Beierle. Nonmonotonic reasoning from conditional knowledge bases with system W. *Ann. Math. Artif. Intell.*, 90(1):107–144, 2022.

[Kraus *et al.*, 1990] Sarit Kraus, Daniel J. Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.

[Kuhlmann *et al.*, 2022] Isabelle Kuhlmann, Anna Gessler, Vivien Laszlo, and Matthias Thimm. A comparison of ASP-based and SAT-based algorithms for the contension inconsistency measure. In *International Conference*

on Scalable Uncertainty Management, pages 139–153. Springer, 2022.

[Lehmann and Magidor, 1989] Daniel Lehmann and Menachem Magidor. What does a conditional knowledge base entail? In Ron Brachman and Hector Levesque, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Canada, 1989. Morgan Kaufmann.

[Lehmann, 1995] Daniel Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15(1):61–82, 1995.

[Morgado *et al.*, 2014] Antonio Morgado, Alexey Ignatiev, and Joao Marques-Silva. MSCG: Robust core-guided MaxSAT solving: System description. *Journal on Satisfiability, Boolean Modelling and Computation*, 9(1):129–134, 2014.

[Nute and Cross, 2002] Donald Nute and Charles Cross. Conditional logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 4, pages 1–98. Kluwer Academic Publishers, second edition edition, 2002.

[Papadimitriou, 1994] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pearl, 1990] Judea Pearl. System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In Rohit Parikh, editor, *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge, Pacific Grove, CA, USA, March 1990*, pages 121–135. Morgan Kaufmann, 1990.

[Sinz, 2005] Carsten Sinz. Towards an optimal CNF encoding of Boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[Sorensson and Een, 2005] Niklas Sorensson and Niklas Een. Minisat v1.13 - a SAT solver with conflict-clause minimization. *SAT*, 2005(53):1–2, 2005.

[von Berg *et al.*, 2024] Martin von Berg, Arthur Sanin, and Christoph Beierle. An implementation of nonmonotonic reasoning with c-representations using an SMT solver. *Int. J. Approx. Reason.*, 175:109285, 2024.

[Wynn, 2018] Ed Wynn. A comparison of encodings for cardinality constraints in a SAT solver. *CoRR*, abs/1810.12975, 2018.