Towards Defeasible Reasoning Using Knowledge Compilation Techniques

Luke Slater[1,†], Thomas Meyer[1] and Jesse Heyninck[1,2]

[1]University of Cape Town and CAIR, Cape Town, South Africa
[2]Open University, Heerlen, Netherlands

Abstract
We explore the use of ordered binary decision diagrams (OBDDs) and multi-terminal binary decision diagrams (MTBDDs) for compiling and executing defeasible reasoning, focusing on System Z, also known as rational closure. We introduce an OBDD-based algorithm for System Z inference that parallels traditional SAT-based approaches. Additionally, we investigate the use of MTBDDs for encoding ordinal conditional functions (OCFs), providing a more general and unified knowledge compilation framework for defeasible reasoning tasks involving ranking functions. While our primary focus is on System Z, the proposed techniques are applicable to other forms of defeasible reasoning and conditional logic more broadly. Experimental results demonstrate that OBDDs can significantly accelerate inference. However, our evaluation is limited to synthetically generated belief bases, and further research is needed to validate benefits in real-world scenarios. Overall, our findings suggest that binary decision diagrams offer a promising alternative to SAT-based methods for defeasible reasoning, and warrant further investigation.

1. Introduction

Defeasible reasoning augments propositional logic with *conditionals*—statements of the form "if α, then typically β"—whose semantics are inherently non-monotonic. A popular way to interpret such conditionals is Spohn's *ordinal conditional functions* (OCFs) [1], which rank possible worlds by plausibility. System Z (also known as *rational closure*) constructs an OCF from a finite belief base of conditionals [2]. It was shown by Goldszmidt and Pearl (1990) to be equivalent to *rational closure* [4].

The implementation challenge. Although System Z enjoys elegant theory, computing its inference relation is $\mathsf{P}^{\mathsf{NP}}_{\|}$-complete [5]. Current practice therefore reduces each sub-problem to SAT and relies on off-the-shelf CDCL solvers. This strategy is easy to code but can become expensive on large or structurally complex inputs.

Knowledge compilation to the rescue. Knowledge compilation tackles intractability by preprocessing a formula into a target language that supports polynomial-time queries [6]. Ordered binary decision diagrams (OBDDs) are a well-known target whose strengths include constant-time equivalence checking and fast model counting [7]. Our first contribution shows how the classic SAT-based algorithm for System Z can be translated, step for step, into OBDD operations.

From OBDDs to ADDs. Because an OBDD stores only Boolean values in its sinks, a query may still have to inspect several ranks before reaching a conclusion. We therefore turn to multi-terminal BDDs (MTBDDs, or ADDs) [8, 9], whose sinks can carry arbitrary finite values. Encoding the rank itself in each sink yields a *single* diagram for the whole OCF and reduces every inference to a constant number of `apply` calls. The same idea extends to other ranking-based formalisms such as lexicographic closure [10] and c-representations [11], giving a unified compilation framework.

23rd International Workshop on Nonmonotonic Reasoning, November 11-13, 2025, Melbourne, Australia
[†]Corresponding author.
sltluk001@myuct.ac.za (L. Slater); tommie.meyer@uct.ac.za (T. Meyer); jesse.heyninck@ou.nl (J. Heyninck)
0009-0000-5376-159X (L. Slater); 0000-0003-2204-6969 (T. Meyer); 0000-0002-3825-4052 (J. Heyninck)

Preliminary results. On synthetically generated belief bases with up to 20 variables, our OBDD solver answers queries up to $100\times$ faster than a SAT baseline, while the MTBDD version is up to $50\times$ faster. The speed-up decreases with input size because random formulas lack the regularities that BDDs exploit for compactness, and real-world datasets for conditional logic are still rare. Nonetheless, the gains confirm that decision diagrams are a credible alternative to SAT.

Contributions. We summarize the main contributions of this paper as follows:

- An OBDD-based re-implementation of SAT-style System Z inference.
- An ADD compilation that directly encodes the System Z ranking function.
- An empirical comparison of SAT, OBDD, and ADD engines on synthetic data.

2. Preliminaries

In this section, we survey the necessary preliminaries on propositional logic (Sec. 2.1), conditional logic (Sec. 2.2), ordinal conditional functions (Sec. 2.3) and inductive inference (Sec. 2.4). Lastly, we provide the necessary preliminaries on ordered binary decision diagrams and diagram synthesis (Sec. 2.5).

2.1. Propositional Logic

Assuming a finite set of n boolean variables Σ, we fix a variable ordering $x_0, x_1, \ldots, x_{n-1}$. This ordering remains fixed for all formulas in our propositional language $\mathcal{L}_\Sigma$ (abbreviated to $\mathcal{L}$), which is constructed using the usual connectives of propositional logic. A classical *interpretation* or *possible world* is a mapping $\omega : \Sigma \mapsto \{\top, \bot\}$. The set of all such interpretations is Ω. We write $\omega \models \varphi$ if ω satisfies φ, and $\omega \not\models \varphi$ otherwise. The set of models of a formula φ is $\texttt{Mods}(\varphi) = \{\omega \in \Omega \mid \omega \models \varphi\}$. For clarity, we sometimes write $\varphi(v_0, v_1, \ldots, v_{n-1})$ to denote the formula obtained by replacing each instance of x_i in φ with $v_i \in \{\top, \bot, x_i\}$. If the assignment is clear or not relevant, we simply write $\varphi(\omega)$ to denote the formula resulting when every instance of x_i is replaced by its truth assignment in ω.

Definition 1 (conditioning). *The conditioning of φ on a variable x_i and constant $c \in \{\top, \bot\}$ is the formula obtained by replacing all instances of x_i in φ with c. We denote this by*

$$\varphi_{x_i=c} \;=\; \varphi(x_0, \ldots, x_{i-1}, c, x_{i+1}, \ldots, x_{n-1}).$$

In the literature, $\varphi_{x_i=\top}$ and $\varphi_{x_i=\bot}$ are often referred to the positive and negative *cofactors* of φ with respect to x_i.

2.2. Defeasible Reasoning With Conditionals

Propositional logic is extended to conditional logic by introducing *conditionals*, statements of the form $(\beta \mid \alpha)$. The language of propositional conditional logic is given by $(\mathcal{L} \mid \mathcal{L}) = \{(\beta \mid \alpha) \mid \alpha, \beta \in \mathcal{L}\}$. For a world ω, it is useful to define the semantics of conditionals as *generalized indicator functions* [12]:

$$(\beta \mid \alpha)(\omega) = \begin{cases} \top, & \text{if } (\alpha \wedge \beta)(\omega) = \top, \\ \bot, & \text{if } (\alpha \wedge \neg\beta)(\omega) = \top, \\ \times, & \text{otherwise.} \end{cases}$$

A world ω *verifies* a conditional δ if $\delta(\omega) = \top$, *falsifies* it if $\delta(\omega) = \bot$, and is irrelevant otherwise (i.e. when $\omega \models \neg\alpha$, denoted $\delta(\omega) = \times$). For a belief base Δ (a finite set of conditionals), we write:

$$\Delta(\omega) = \begin{cases} \top, & \text{if } \forall\delta \in \Delta, \delta(\omega) = \top, \\ \bot, & \text{if } \exists\delta \in \Delta, \delta(\omega) = \bot, \\ \times, & \text{otherwise.} \end{cases}$$

Intuitively, the value $\times$ marks worlds where the antecedent α of a conditional is false—so the statement is irrelevant—and, for a belief base Δ, it indicates that no conditional is falsified and at least one is irrelevant. To use classical reasoning tools, one often uses *materialization* $\mathcal{M}(\beta \mid \alpha) = \alpha \to \beta$, where $(\alpha \to \beta)$ is considered only to check falsifications. We extend this to a belief base Δ such that $\mathcal{M}(\Delta) = \bigwedge\{\mathcal{M}(\delta) \mid \delta \in \Delta\}$.

2.3. Ordinal Conditional Functions

Ordinal conditional functions (OCFs) or *ranking functions* are a well- known semantics for conditional logic introduced by Spohn [1988]. They assign to each world ω a non-negative integer reflecting its degree of *surprise.* Higher values indicate less plausibility, and ∞ impossibility.

Definition 2 (Ranking Function). *A ranking function is a mapping $\kappa : \Omega \to \mathbb{N} \cup \{\infty\}$. It extends to formulas by*

$$\kappa(\varphi) \;=\; \min\{\kappa(\omega) \mid \omega \models \varphi\},$$

with $min\emptyset = \infty$. We say κ accepts *$(\beta \mid \alpha)$ if $\kappa(\alpha \wedge \beta) < \kappa(\alpha \wedge \neg\beta)$. In other words, a ranking function accepts $(\beta \mid \alpha)$ if all the most plausible worlds satisfying α also satisfy β. Also, $|\kappa|$ denotes $|\{\, i \mid \exists\omega \in \Omega,\, \kappa(\omega) = i\}|$ for $i < \infty$.*

2.4. Inductive Inference and System Z

Inductive inference, or defeasible inference, involves the use of inference relations $\mid\!\sim_\Delta$ derived from a conditional belief base Δ, known as *inductive inference operators.* Ranking functions offer a common basis for generating a relation $\mid\!\sim^\kappa$, where $\alpha \mid\!\sim^\kappa \beta$ holds if κ accepts $(\beta \mid \alpha)$. Notable examples of such inference relations generated from ranking functions include System Z [2] (equivalent to rational closure [4]), lexicographic closure [10], and c-representations [11]. This paper focuses on System Z.

System Z partitions a finite belief base Δ into groups of conditionals (a Z-partitioning) such that each group collects conditionals mutually "tolerable." A conditional δ is *tolerated* by Δ if there is some ω for which $\delta(\omega) = \top$ and $\Delta(\omega) \neq \bot$. Classical formulas α can also be included in Δ by representing them as conditionals $(\bot \mid \neg\alpha)$. Such conditionals are inherently intolerable and are assigned to Δ_∞. The System Z ranking function κ^Z_Δ is then defined through this partitioning.

Definition 3 (Z-partitioning and Z-Ranking). *Let $Z_\Delta = (\Delta_0, \Delta_1, \dots, \Delta_\infty)$ be the Z-partitioning of Δ, where $\Delta_0 = \{\delta \in \Delta \mid \Delta$ tolerates $\delta\}$ and $(\Delta_1, \dots, \Delta_\infty)$ is the Z-partitioning of $\Delta \setminus \Delta_0$. The System Z ranking κ^Z_Δ is given by*

$$\kappa^Z_\Delta(\omega) \;=\; \min\{\, i \mid \big(\bigcup_{j\geq i} \Delta_j\big)(\omega) \neq \bot\} \quad (\min\emptyset = \infty).$$

Intuitively, the Z-ranking orders worlds by how badly they violate the belief base: worlds satisfying all high-priority (more tolerable) conditionals get lower ranks, while those that break less tolerable ones are pushed to higher ranks, with classically impossible worlds assigned rank ∞. Classical formulas thus establish an initial distinction between worlds with finite ranks $i \in \mathbb{N}$ and those deemed classically impossible (assigned rank ∞). Once the ranking function κ^Z_Δ has been generated for a given belief base Δ, it can then be used to perform inference under System Z.

Definition 4 (Z-inference). *Given a belief base Δ, the inference relation $\mid\!\sim^Z_\Delta$ corresponding to System Z inference is defined according to κ^Z_Δ such that $\alpha \mid\!\sim^Z_\Delta \beta$ iff κ^Z_Δ accepts $(\beta \mid \alpha)$.*

Example 1. *Let*

$$\begin{aligned}\Delta = \big\{&(w \mid b),\\ &(f \mid b),\\ &(\neg f \mid p),\\ &(\bot \mid \neg(p \!\to b))\big\},\end{aligned}$$

with b = bird, p = penguin, w = has wings, and f = can fly. System Z yields the partition

$$\begin{aligned}\Delta_0 &= \{(w \mid b),\ (f \mid b)\},\\ \Delta_1 &= \{(\neg f \mid p)\},\\ \Delta_\infty &= \{(\bot \mid \neg(p \to b))\}.\end{aligned}$$

The induced ranking function κ^Z_Δ assigns:

$$\begin{aligned}\kappa^Z_\Delta(p \wedge b \wedge f) &= 2,\\ \kappa^Z_\Delta(p \wedge b \wedge \neg f) &= 1,\\ \kappa^Z_\Delta(p \wedge \neg b) &= \infty,\\ \kappa^Z_\Delta(\neg p \wedge b \wedge f \wedge w) &= 0,\\ \kappa^Z_\Delta(\neg p \wedge b \wedge f \wedge \neg w) &= 1,\\ \kappa^Z_\Delta(\neg p \wedge b \wedge \neg f) &= 1,\\ \kappa^Z_\Delta(\neg p \wedge \neg b) &= 0.\end{aligned}$$

From this we infer, for instance:

$$b \wedge \neg f \not\mid\!\sim^Z_\Delta p, \quad b \wedge \neg f \not\mid\!\sim^Z_\Delta \neg p, \quad p \not\mid\!\sim^Z_\Delta w.$$

2.5. OBDD Based Knowledge Compilation

Knowledge compilation seeks to address the computational difficulty inherent in general propositional reasoning by dividing the task of query answering into two distinct phases:

Off-line Compile a general propositional formula $\varphi \in \mathcal{L}$ into a logically equivalent representation $F \in \mathcal{T}$, for a more computationally tractable target language $\mathcal{T}$. Although this compilation step may be exponentially expensive in the worst case, it is performed only once.

On-line Efficiently answer queries, typically in polynomial time, using the previously compiled representation F.

The underlying rationale is that an initial, computationally intensive compilation phase significantly reduces the complexity of subsequent queries, effectively amortizing the initial compilation cost over multiple queries. A query q (e.g., satisfiability, entailment, or model counting) is considered *tractable* on a target language $\mathcal{T}$ if there exists an algorithm that, for every formula $F \in \mathcal{T}$, determines $q(F)$ within polynomial time relative to the size $|F|$ of its representation in $\mathcal{T}$. In this paper, the primary query under consideration is entailment checking. Consequently, our analysis will focus on two prominent target languages: CNF (Conjunctive Normal Form), the standard input format for most SAT solvers, and OBDD (Ordered Binary Decision Diagrams), commonly used for tractable and efficient entailment checking.

Definition 5 (CNF). *A formula φ is in the language* CNF *if it can be expressed as:*

$$\varphi = \bigwedge_{i=1}^{m} \vartheta_i, \quad \vartheta_i = \bigvee_{j=1}^{k_i} \ell_{ij},$$

where each literal ℓ_{ij} is either x or $\neg x$ for some variable $x \in \Sigma$. The size $|\varphi|$ is defined as the total number of literal occurrences in φ.

SAT solving on general CNF formulas is a well-known problem in NP, and thus CNF does not qualify as a tractable target language for entailment checking. The next target language we consider, OBDD, is in fact tractable for many key queries—most notably entailment checking. Ordered Binary Decision Diagrams were first proposed by Bryant [7], building on earlier work by Lee [13] and Akers [14], and remain the most prominent tractable representation in the literature.

Definition 6 (OBDD-structure)**.** *A graph Ψ is in the language* OBDD *if it is a rooted, directed acyclic graph with two types of nodes:*

- ***Variable nodes**: Each has an index $\mathrm{I}(\nu) < n$ and two child nodes $\mathrm{T}(\nu)$ and $\mathrm{F}(\nu)$ satisfying $\mathrm{I}(\nu) < \mathrm{I}(\mathrm{T}(\nu))$ and $\mathrm{I}(\nu) < \mathrm{I}(\mathrm{F}(\nu))$.*
- ***Sink nodes**: Each has a value $\mathrm{V}(\nu) \in \{\top, \bot\}$.*

Each variable node ν corresponds to a variable x_i where $\mathrm{I}(\nu) = i$. Any world ω traces a unique path from the root: going to $\mathrm{T}(\nu)$ if $\omega \models x_i$, and to $\mathrm{F}(\nu)$ if $\omega \models \neg x_i$, until a sink node is reached. Therefore, an OBDD is a graphical representation of the semantic behavior of all Boolean formulas equivalent to some formula α, defined recursively as follows.

Definition 7 (OBDD-representation)**.** *An OBDD Ψ with root ν represents all formulas equivalent to φ, where φ is defined as follows:*

- *If ν is a sink node, then $\varphi \equiv \top$ if $\mathrm{V}(\nu) = \top$, or $\varphi \equiv \bot$ if $\mathrm{V}(\nu) = \bot$.*
- *If ν is a variable node with $\mathrm{I}(\nu) = i$, then*

$$\varphi \equiv \left(x_i \wedge \varphi_{x_i=\top}\right) \vee \left(\neg x_i \wedge \varphi_{x_i=\bot}\right),$$

with $\mathrm{T}(\nu)$ and $\mathrm{F}(\nu)$ representing formulas equivalent to $\varphi_{x_i=\top}$ and $\varphi_{x_i=\bot}$, respectively.

This relies on *Shannon expansion* [15] (also known as Boole's expansion theorem [16]), where a formula is recursively decomposed into positive and negative cofactors for each variable in the ordering.

Definition 8 (Isomorphic)**.** *Two OBDDs, Ψ_1 and Ψ_2, are* isomorphic *provided there exists a bijection ψ between their nodes such that for every node ν_1 in Ψ_1 with $\psi(\nu_1) = \nu_2$ in Ψ_2, one of the following holds:*

- *Both ν_1 and ν_2 are sink nodes with $\mathrm{V}(\nu_1) = \mathrm{V}(\nu_2)$.*
- *Both ν_1 and ν_2 are variable nodes with $\mathrm{I}(\nu_1) = \mathrm{I}(\nu_2)$, and the bijection preserves their children: $\psi(\mathrm{T}(\nu_1)) = \mathrm{T}(\nu_2)$ and $\psi(\mathrm{F}(\nu_1)) = \mathrm{F}(\nu_2)$.*

Definition 9 (Reduced)**.** *An OBDD is fully reduced if it contains no redundant nodes such that $\mathrm{T}(\nu) = \mathrm{F}(\nu)$ and no pair of isomorphic subgraphs.*

In practice, an "OBDD" refers by default to a fully reduced diagram. This complete reduction yields a unique, canonical form for each Boolean function, making equivalence checking a straightforward graph-isomorphism test. Moreover, when compiling multiple functions, it is standard to embed them in a single shared DAG: identical subgraphs are factored out and reused, resulting in a very compact representation that exploits redundancy across formulas, an example of this is shown in Figure 4.

2.5.1. General Complexity of OBDD Operations

Most OBDD transformations rely on Bryant's `apply` algorithm [7], which performs Boolean operations (e.g. $\wedge$, $\vee$, $\oplus$) by simultaneously traversing two operand diagrams and at the same time producing the diagram resulting from the operation. In practice, one compiles a complex formula by recursively descending its parse tree and invoking `apply` at each connective—this "recursive-descent parsing" makes `apply` the workhorse of OBDD construction.

A crucial feature is that each reduced subgraph is assigned a unique numeric ID. Whenever `apply` produces a new subgraph, it checks whether an identical one (i.e. isomorphic) has been seen before; if so, it reuses that existing ID. This scheme means that two OBDDs represent the same function precisely when their root IDs coincide—equivalence checking reduces to a single ID comparison in $O(1)$ time.

Memoization further accelerates `apply` by caching every computed result for a given pair of operand IDs and operator. If the same operation is requested again, `apply` simply looks up the cached outcome instead of re-descending the diagrams. Together, these ideas guarantee that `apply` runs in $O(|\Psi_1| \cdot |\Psi_2|)$ time in the worst case, while trivializing equivalence tests and leaving other queries—satisfiability in $O(1)$, model counting in $O(|\Psi|)$, and model evaluation in $O(n)$—as simple graph traversals or ID checks.

Operation	**Time Complexity**
Satisfiability	$O(1)$
`apply`	$O(\lvert\Psi_1\rvert \cdot \lvert\Psi_2\rvert)$
Equivalence checking	$O(1)$
Model counting	$O(\lvert\Psi\rvert)$
Model evaluation (assignment)	$O(n)$

Table 1
Time complexity of common queries and transformations on an OBDD Ψ, or two OBDDs Ψ_1 and Ψ_2.

2.5.2. Variable Ordering

The succinctness of an OBDD depends critically on the variable ordering chosen—a problem that is NP-hard in the worst case [17]. Thankfully, a range of effective heuristics (such as sifting, dynamic reordering, and even genetic-algorithm approaches) typically find good orders in practice. Even so, most arbitrary Boolean functions admit no compact OBDD representation, and exponential blow-up remains possible. The practical power of OBDDs, therefore, derives largely from the fact that meaningful, real-world formulas often exhibit substantial structural regularity and redundancy, which these diagrams can exploit to remain tractable.

3. A SAT–Based Implementation of System Z

Constructing the Z–partition of a conditional belief base is $\mathsf{FP}_{\|}^{\mathsf{NP}}$-complete, and performing Z–inference once this partition is known is $\mathsf{P}_{\|}^{\mathsf{NP}}$-complete [18, 5]. In the absence of an NP-oracle, the standard remedy is to reduce the relevant sub-tasks to instances of propositional satisfiability and to rely on a modern SAT solver—typically a conflict-driven clause-learning (CDCL) engine [19, 20, 21]—to answer each query.

3.1. Encoding the Z–partition

Definition 10 (Z-vector). *Let $(\Delta_0, \Delta_1, \ldots, \Delta_\infty)$ be the Z–partition of Δ. The Z-*vector $(\delta_0^Z, \ldots, \delta_n^Z)$ *is obtained by*

$$\delta_i^Z = \mathcal{M}\Big(\bigcup_{j \geq i} \Delta_j\Big),$$

where $\mathcal{M}$ materialises every conditional as an implication and n is the largest index that actually occurs:

$$n = \begin{cases} \min\{\, i \mid \Delta_i = \varnothing\} & \textit{if } \Delta_\infty \neq \varnothing, \\ \max\{\, i \mid \Delta_i \neq \varnothing\} & \textit{otherwise.} \end{cases}$$

Each component δ_i^Z is translated once into CNF and handed to the SAT solver as a separate formula.

3.2. SAT procedure for Z–inference

Freund's original SAT-based procedure [22], refined by Casini *et al.* [23], is reproduced in Algorithm 1. The routine scans the Z-vector from the lowest to the highest rank, searching for the first level whose conjunction with the antecedent α is satisfiable. Once such a level i is found, a second SAT call decides whether some world of the same rank also violates the consequent β.

Complexity. In the worst case the loop invokes the solver $|\kappa_\Delta^Z|$ times; each call can take $O(2^{|\Sigma|})$, giving a bound of $O(|\kappa_\Delta^Z| \cdot 2^{|\Sigma|})$.

```
**Input:** $Z$–vector $(\mathsf{CNF}(\delta_0^Z), \ldots, \mathsf{CNF}(\delta_n^Z))$; conditional $(\beta \mid \alpha)$
**Output:** **true** iff $\kappa_\Delta^Z$ accepts $(\beta \mid \alpha)$
$\alpha \leftarrow \mathsf{CNF}(\alpha)$
$\neg\beta \leftarrow \mathsf{CNF}(\neg\beta)$
$i \leftarrow 0$
**while** $i \leq n$ **do**
    $\delta \leftarrow \delta_i^Z \wedge \alpha$
    **if** $SAT(\delta)$ **then**
        **return** $\neg$SAT( $\delta \wedge \neg\beta$ )
    **end**
    $i \leftarrow i + 1$
**end**
**return** false
```

Algorithm 1: `ZinfSAT`

3.3. Correctness

Lemma 1. *For the Z-vector $(\delta_0^Z, \ldots, \delta_n^Z)$ of Δ,*

$$\{\omega \mid \kappa_\Delta^Z(\omega) = 0\} \; = \; \mathtt{Mods}(\delta_0^Z),$$

and for every $i > 0$,

$$\{\omega \mid \kappa_\Delta^Z(\omega) = i\} \; = \; \mathtt{Mods}(\delta_i^Z) \setminus \mathtt{Mods}(\delta_{i-1}^Z).$$

Proof. Straightforward induction on i, using $\kappa_\Delta^Z(\omega) = \min\{j \mid (\bigcup_{k \geq j} \Delta_k)(\omega) \neq \bot\}$. □

Theorem 2. *For any belief base Δ with Z-vector $(\delta_0^Z, \ldots, \delta_n^Z)$ and any conditional $(\beta \mid \alpha)$,* `ZinfSAT` *terminates and returns **true** exactly when κ_Δ^Z accepts $(\beta \mid \alpha)$.*

Proof. Termination follows from the bounded loop over i. For soundness, let $i = \kappa_\Delta^Z(\alpha)$. By Lemma 1, $\delta_i^Z \wedge \alpha$ is satisfiable and $\delta_j^Z \wedge \alpha$ is unsatisfiable for $j < i$, so the loop stops at the correct level. Acceptance then hinges on the (un)satisfiability of $\delta_i^Z \wedge \alpha \wedge \neg\beta$, which equals $\kappa_\Delta^Z(\alpha \wedge \neg\beta) = i$. Completeness is the converse argument. □

4. An OBDD–Based Implementation of System Z

Binary decision diagrams are a natural alternative to the SAT encoding of Sect. 3: equivalence checking is constant-time, Boolean operations are handled by a single `apply` routine. We therefore compile each level of the Z–partition into an OBDD, obtaining the *OBDD Z-vector* $(\Psi_0^Z, \ldots, \Psi_n^Z)$, and answer queries by a short sweep through that vector. An example of the optimal shared DAG representing the OBDD Z-vector for the belief base in Example 1 is illustrated in Figure 1.

4.1. OBDD procedure for Z–inference

Let Ψ_α and Ψ_β be the OBDDs for α and β. Algorithm 2 (`ZinfOBDD`) tests successive ranks until it finds the first i such that $\Psi_i^Z \wedge \Psi_\alpha$ is satisfiable; a second test then checks whether any world of that same rank falsifies β.

Complexity. An `apply` call on two diagrams of size $|\Psi_i^Z|$ and $|\Psi_\alpha|$ runs in $O(|\Psi_i^Z| \cdot |\Psi_\alpha|)$. In the worst case the loop iterates $|\kappa_\Delta^Z|$ times, giving

$$O\big(|\Psi_\alpha| \;\cdot \textstyle\sum_{i=0}^{|\kappa_\Delta^Z|} |\Psi_i^Z|\big),$$

which improves on the SAT bound $O(|\kappa_\Delta^Z| \cdot 2^{|\Sigma|})$ whenever the diagrams are compact.

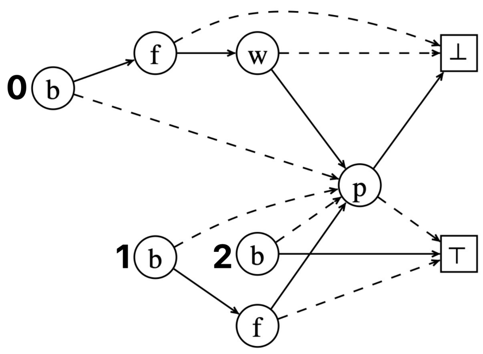

Figure 1: Shared DAG representing Z-vector from Example 1

```
Input: $Z$-vector $(\mathsf{OBDD}(\Psi_0^Z), \ldots, \mathsf{OBDD}(\Psi_n^Z))$; conditional $(\beta \mid \alpha)$
Output: true iff $\kappa_\Delta^Z$ accepts $(\beta \mid \alpha)$
$\Psi_\alpha \leftarrow \mathsf{OBDD}(\alpha)$
$\neg\Psi_\beta \leftarrow \mathsf{OBDD}(\neg\beta)$
$i \leftarrow 0$
while $i \leq n$ do
    $\Psi \leftarrow \Psi_i^Z \wedge \Psi_\alpha$
    if $\Psi \neq \bot$ then
        return $\big(\Psi \wedge \neg\Psi_\beta\big) = \bot$
    end
    $i \leftarrow i + 1$
end
return false
```

Algorithm 2: `ZinfOBDD`

4.2. Correctness

Theorem 3. *Given the OBDD Z-vector $(\Psi_0^Z, \ldots, \Psi_n^Z)$ and an OBDD conditional $(\Psi_\beta \mid \Psi_\alpha)$, `ZinfOBDD` terminates and returns **true** exactly when κ_Δ^Z accepts $(\beta \mid \alpha)$.*

Proof. Termination is immediate from the bounded loop. By Lemma 1, $\Psi_i^Z \wedge \Psi_\alpha \neq \bot$ holds precisely when $\kappa_\Delta^Z(\alpha) = i$. The additional test $\Psi \wedge \neg\Psi_\beta = \bot$ is therefore equivalent to $\kappa_\Delta^Z(\alpha \wedge \neg\beta) > \kappa_\Delta^Z(\alpha)$, i.e. to the acceptance criterion for $(\beta \mid \alpha)$ under System Z. □

5. An ADD Representation of System Z

An *algebraic decision diagram* (ADD) [9, 8] is a rooted, directed acyclic graph analogous to an OBDD, except that its sink nodes may carry values from an arbitrary finite domain R rather than just $\top, \bot$. ADDs inherit the efficient `reduce` and `apply` algorithms of OBDDs, making them a powerful compilation target for a wide range of reasoning tasks. By generalizing this compilation target to encompass arbitrary ranking functions, we obtain a single, coherent knowledge-compilation framework for all defeasible-reasoning formalisms based on ordinal conditional functions. In particular, this unified approach subsumes not only System Z but also other ranking-based inference mechanisms such as lexicographic closure [10] and c-representations [11]. Because the compiled structure directly encodes the ranking function, it additionally permits efficient execution of model-centric operations—including model checking, model enumeration, and model counting—without resorting to costly SAT-based procedures. We therefore refer to any multi-terminal BDD over $\mathbb{N} \cup \{\infty\}$ that represents a ranking function as a *ranked binary decision diagram* (RBDD).

5.1. Representation

Definition 11 (RBDD). *A reduced ADD Ψ^κ whose sinks are labelled by non-negative integers or ∞ is an* RBDD. *Every variable node ν (index $\mathrm{I}(\nu)$) has high and low children $\mathrm{T}(\nu)$ and $\mathrm{F}(\nu)$ consistent with the variable ordering. Additionally, we record at every node the set of reachable ranks*

$$\mathrm{R}(\nu) = \begin{cases} \{\mathrm{V}(\nu)\} & \nu \textit{ sink}, \\ \mathrm{R}(\mathrm{T}(\nu)) \cup \mathrm{R}(\mathrm{F}(\nu)) & \textit{otherwise}, \end{cases}$$

which later allows $O(1)$ existence checks, and $O(n)$ model checking for a given rank.

We say any graph adhereing to these requirements is in the language RBDD. Before giving the recursive semantics of an RBDD, we must introduce the notion of conditioning, which captures how a ranking function behaves when one variable is fixed. Conditioning yields sub-ranking functions corresponding to each truth value of the chosen variable, and will serve as the basis for the node-wise interpretation of the diagram.

Definition 12 (Conditioning / cofactor of a ranking function). *Let $\kappa : \Omega \to \mathbb{N} \cup \{\infty\}$ be a ranking function over the set of worlds Ω, and let $x_i \in \Sigma$ be a propositional variable. For $c \in \{\top, \bot\}$, the* conditioning *(or* cofactor*) of κ with respect to $x_i = c$ is the function*

$$\kappa_{x_i=c}(\omega) = \kappa\big(\omega[x_i \mapsto c]\big), \quad c \in \{\top, \bot\},$$

where $\omega[x_i \mapsto c]$ denotes the world obtained from ω by fixing the value of x_i to c while leaving all other variables unchanged.

Conditioned on different values of x_i, each sub-ranking function $\kappa_{x_i=c}$ restricts κ to a simpler domain, capturing its behavior when x_i is permanently fixed. By iterating this process over successive variables, we generate a hierarchy of increasingly specialized ranking functions. An RBDD compactly represents this entire hierarchy: each internal node corresponds to one such conditioning step, and its outgoing edges lead to the sub-ranking functions that arise when the node's variable is set to $\top$ or $\bot$.

Definition 13 (Recursive semantics of an RBDD). *Let ν be the root of Ψ^κ. The ranking function represented by the diagram is defined recursively:*

- ***Sink node.*** *If ν is a sink, then $\kappa(\omega) = \mathrm{V}(\nu)$ for all worlds ω.*
- ***Variable node.*** *If $\mathrm{I}(\nu) = i$, denote by $\kappa^+ = \kappa_{x_i=\top}$ the function represented by $\mathrm{T}(\nu)$ and by $\kappa^- = \kappa_{x_i=\bot}$ the function represented by $\mathrm{F}(\nu)$. Then for every world ω*

$$\kappa(\omega) = \begin{cases} \kappa^+(\omega) & \textit{if } \omega \models x_i, \\ \kappa^-(\omega) & \textit{otherwise.} \end{cases}$$

The optimal RBDD for the belief base in Example 1 is shown in Figure 2.

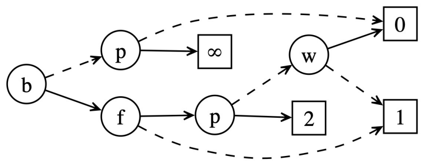

Figure 2: MTBDD representing κ^Z_Δ

5.2. Compilation

The usual OBDD reduction rules—eliminating redundant nodes and merging isomorphic sub-graphs—carry over unchanged, and we henceforth treat every RBDD as fully reduced unless noted otherwise. As argued earlier, the practical construction of large decision diagrams proceeds by successive applications of the `apply` algorithm: complex structures are built from a collection of smaller, already-compiled components. For System Z we first compile each level of the Z-partitioning into an OBDD, obtaining the Z-vector $(\Psi_0^Z, \dots, \Psi_n^Z)$ such that $\Psi_i^Z = \mathsf{OBDD}(\delta_i^Z)$. These OBDDs can then be relabelled and combined to yield a single RBDD encoding the entire System Z ranking function, as stated next.

Theorem 4. *Let $(\Psi_0^Z, \dots, \Psi_n^Z)$ be the OBDD Z-vector of a belief base Δ. Replace every sink label $\top$ in Ψ_i^Z by the integer i and every sink label $\bot$ by ∞. Then*

$$\Psi_\kappa^Z \;=\; \min\bigl(\Psi_0^Z, \Psi_1^Z, \dots, \Psi_n^Z\bigr)$$

is an RBDD that represents the System Z ranking function κ_Δ^Z.

Proof. By Lemma 1, the OBDD Ψ_i^Z (after relabelling) evaluates to a finite rank $\le i$ precisely on those worlds whose System Z rank is at most i, and to ∞ otherwise. Taking the pointwise minimum of these diagrams therefore assigns to each world ω the smallest i for which $\kappa_\Delta^Z(\omega) = i$; if no such i exists, every operand contributes ∞ and the minimum is ∞. Consequently the resulting diagram encodes exactly κ_Δ^Z. □

5.3. Defeasible Entailment

We now show how a constant number of `apply` operations suffices to decide whether a conditional $(\beta \mid \alpha)$ is accepted by a ranking function κ represented as an RBDD Ψ_κ. The key is an operator that *restricts* a rank to cases in which a Boolean formula holds and maps all other cases to ∞.

Definition 14 (Restriction operator $\propto$). *For $v_1 \in \mathbb{N} \cup \{\infty\}$ and $v_2 \in \{\top, \bot\}$ define*

$$v_1 \;\propto\; v_2 \;=\; \begin{cases} v_1 & \text{if } v_2 = \top, \\ \infty & \text{if } v_2 = \bot. \end{cases}$$

Given an RBDD Ψ_κ and an OBDD Ψ_φ, the diagram $\Psi_\kappa \propto \Psi_\varphi$ is obtained by applying this operator node-wise via the standard `apply` algorithm. The result is an RBDD that represents the restricted *ranking function*

$$\kappa_\varphi(\omega) = \begin{cases} \kappa(\omega) & \text{if } \omega \models \varphi, \\ \infty & \text{otherwise.} \end{cases}$$

Let Ψ_α and Ψ_β be the OBDDs for α and β, respectively, and set

$$\nu_1 = \mathrm{root}(\Psi_\kappa \propto \Psi_\alpha), \qquad \nu_2 = \mathrm{root}\bigl((\Psi_\kappa \propto \Psi_\alpha) \propto \neg\Psi_\beta\bigr).$$

Theorem 5. *The ranking function κ accepts the conditional $(\beta \mid \alpha)$ iff*

$$\min \mathrm{R}(\nu_1) \;<\; \min \mathrm{R}(\nu_2).$$

Proof. If $\kappa(\alpha) = \infty$ then $\Psi_\kappa \propto \Psi_\alpha$ is a single sink labelled ∞, and the second synthesis yields the same; the inequality fails and the conditional is rejected. Otherwise $\min \mathrm{R}(\nu_1) = \kappa(\alpha)$ and $\min \mathrm{R}(\nu_2) = \kappa(\alpha \wedge \neg\beta)$. Hence

$$\min \mathrm{R}(\nu_1) < \min \mathrm{R}(\nu_2) \quad\Longleftrightarrow\quad \kappa(\alpha) < \kappa(\alpha \wedge \neg\beta),$$

which is equivalent to the acceptance condition $\kappa(\alpha \wedge \beta) < \kappa(\alpha \wedge \neg\beta)$. □

Consequently, a single entailment check requires just three apply operations—two applications of the restriction operator and one final rank comparison—yielding a worst-case time complexity of

$$O\big(|\Psi_\kappa| \cdot |\Psi_\alpha| \cdot |\Psi_\beta|\big),$$

which is typically governed by the size of Ψ_κ. In contrast to the OBDD-based method, which may invoke apply up to $|\kappa|$ times, our RBDD approach bounds calls to a constant.

5.4. General Complexity of RBDD Operations

Recall that each node ν stores the set of reachable ranks $\mathrm{R}(\nu)$. This additional cache—absent from a plain ADD—lets us answer several rank-specific questions without exploring the whole diagram: for instance, testing whether any world has rank i is just the membership check $i \in \mathrm{R}(\text{root})$, and generating a witness of rank i requires at most one edge choice per variable. Combined with the canonical `reduce`/`apply` machinery inherited from OBDDs, these cached rank sets yield the bounds in Table 2. As usual, $|\Psi_\alpha|$ and $|\Psi_\beta|$ are assumed to be tiny compared with $|\Psi^\kappa|$.

Operation on RBDDs	**Time complexity**
Rank lookup $\kappa(\omega)$	$O(n)$
Existence of rank i ($i \in \mathrm{R}(\text{root})$)	$O(1)$
Diagram equivalence (root ID comparison)	$O(1)$
Binary synthesis (`apply`)	$O(\|\Psi_1\| \cdot \|\Psi_2\|)$
Conditioning by formula α	$O(\|\Psi^\kappa\| \cdot \|\Psi_\alpha\|)$
Entailment ($\beta \mid \alpha$)	$O(\|\Psi^\kappa\| \cdot \|\Psi_\alpha\| \cdot \|\Psi_\beta\|)$
Model counting $\kappa(\omega) = i$	$O(\|\Psi^\kappa\|)$

Table 2
Complexities of common queries and transformations on a ranked binary decision diagram Ψ^κ. Constant-time tests rely on the cached rank sets $\mathrm{R}(\nu)$, whereas `apply`-based operations inherit the usual $O(|\Psi_1| \cdot |\Psi_2|)$ worst case from OBDDs.

6. A Brief Comparison

System-Z inference can be driven by a direct SAT encoding, by per-level OBDDs, or by a single *ranked* ADD. The SAT route is parameter-free and benefits from decades of solver engineering, but every query incurs several solver calls whose worst-case time remains exponential. OBDDs move most work off-line: once the Z-vector is compiled, an inference requires at most $|\kappa|$ `apply` calls, yielding large speed-ups when the diagram stays compact. Replacing the Z-vector by one ADD folds all ranks into one structure, cutting each query to *three* `apply` calls and enabling richer model-level questions. Because a fully reduced ADD is *canonical* for its variable ordering, two belief bases induce the same diagram iff they are equivalent under the chosen OCF semantics; a single root-ID comparison thus gives the *first tractable equivalence test* for ranking-based formalisms. Neither decision-diagram method dominates the other—for some variable orderings an ADD can grow larger than the sum of its OBDD components—but *in the worst case, where a query must examine every rank, the ADD per-query cost is likely to be much lower.*

Aspect	SAT	OBDD	ADD
Compile effort	None	Per rank	Whole ranking
Ordering sens.	None	High	Very high
Per-query cost	$\|\kappa\|\times$ SAT	$\|\kappa\|\times$ `apply`	$3\times$ `apply`
Model queries	Costly	Easy	Easiest

$(\lvert\Delta\rvert, \lvert\Sigma\rvert)$	(6,6)	(8,8)	(10,10)	(12,12)	(14,14)	(16,16)	(18,18)	(20,20)
SAT	$122.22\,\mu s$	$146.37\,\mu s$	$161.17\,\mu s$	$172.28\,\mu s$	$180.78\,\mu s$	$190.69\,\mu s$	$197.93\,\mu s$	$203.24\,\mu s$
OBDD	$1.22\,\mu s$	$1.97\,\mu s$	$2.31\,\mu s$	$2.78\,\mu s$	$3.56\,\mu s$	$4.86\,\mu s$	6.17	$7.92\,\mu s$
Speedup	100.18	74.3	69.77	61.97	50.78	39.23	32.08	25.66
RBDD	$2.4\,\mu s$	$4.06\,\mu s$	$4.98\,\mu s$	$6.82\,\mu s$	$8.38\,\mu s$	$11.01\,\mu s$	$14.39\,\mu s$	$18.53\,\mu s$
Speedup	50.93	36.05	32.36	25.26	21.57	17.32	13.75	10.97

Table 3
Preliminary runtimes (microseconds) averaged over 1000 queries per setting.

7. Experimental Results

We evaluated `ZinfOBDD`, `ZinfSAT`, and inference with RBDDs on the synthetically generated CLKR-PS004 dataset [24], which contains 100 belief bases with 10 queries each, spanning various $(|\Delta|, |\Sigma|)$ pairs. We used the CaDiCal solver for SAT [25], which uses CDCL, and CUDD for OBDD and ADD operations [26]. All were implemented in C++. Correctness of our implementation was also verified against the implementation used in [27]. The evaluation was run on an M1 Pro with 16GB of RAM. We used CUDD's built in dynamic variable ordering algorithm to find a good variable ordering for representing the OBDDs and ADDs. For our small-scale preliminary analysis we restricted our experiments to belief bases with $|\Sigma| \leq 20$. For fairness, the time required to construct each $(\Psi_\beta \mid \Psi_\alpha)$ using diagram synthesis was included in the total inference time for each query.

Table 3 summarizes the average runtimes (in microseconds) over 1000 queries per setting. At smaller problem sizes, the OBDD and RBDD approaches are up to three orders of magnitude faster than the SAT based approach, demonstrating that, when OBDDs and RBDDs are relatively compact, they can significantly speed up inference. As $|\Delta|$ and $|\Sigma|$ grow, however, the speedup narrows in both cases. This behavior is expected in synthetic data, as purely synthetic belief bases often lack the redundancies and symmetries OBDDs exploit for efficiency with real-world data. It is well-known that almost all formulas require at least $2^n/2n$ nodes even for the optimal ordering [28]. We also observe that the RBDD approach is consistently approximately half as efficient as the OBDD approach. This is likely because the queries generated for the dataset were not designed to enforce the worst case, where all ranks must be checked until a result is achieved. In most instances, both `ZinfSAT` and `ZinfOBDD` only needed to evaluate the lower ranks.

Nonetheless, our results indicate that both OBDDs and RBDDs could offer significant performance advantages over SAT-based methods. Since large-scale, real-world conditional logic datasets remain rare, the exact gains are difficult to quantify and will depend on the specific application. At the very least, these findings demonstrate that these techniques warrant further investigation in real-world applications.

8. Conclusion and Future Work

Decision diagrams represent a promising alternative to SAT-based methods for performing inference in System-Z reasoning. Using per-rank OBDDs significantly reduces query time, while employing a single MTBDD further restricts the worst-case computational cost to three `apply` operations, irrespective of the number of ranks. Our empirical evaluation on synthetic datasets demonstrates notable performance improvements, achieving of up to two orders of magnitude.

Two main factors currently influence the practical adoption of these methods:

- **Variable ordering:** Existing heuristics are general-purpose and may not exploit characteristics unique to conditional logic. Tailoring these heuristics specifically for conditional logic could further reduce diagram sizes or prevent significant size growth.

- **Benchmarks:** Real-world knowledge bases, such as regulatory rules or semantic-web ontologies, frequently contain structural redundancies absent in synthetic datasets. Developing and releasing comprehensive benchmark datasets is crucial for objectively evaluating these methods.

Future work will focus on several key directions:

1. Developing structure-aware reordering methods to optimize variable ordering.
2. Investigating incremental compilation techniques to efficiently update diagrams in response to changes in the underlying belief base without complete reconstruction.
3. Extending the MTBDD framework to support more expressive ranking-based reasoning formalisms.
4. Conducting extensive experiments using large-scale, realistic datasets as they become available.
5. Exploring a broader range of knowledge compilation targets, including sentential decision diagrams (SDDs).

Overall, our findings indicate that decision-diagram approaches offer reliable and efficient inference for defeasible reasoning, with considerable potential for further optimization and expansion to alternative knowledge compilation representations.

Acknowledgments

This work is based on the research supported in part by the National Research Foundation of South Africa (REFERENCE NO: SAI240823262612, AHPMDS250623326170).

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5 in order to: Improve writing style. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] W. Spohn, Ordinal conditional functions. a dynamic theory of epistemic states, in: W. L. Harper, B. Skyrms (Eds.), Causation in Decision, Belief Change, and Statistics, vol. II, Kluwer Academic Publishers, 1988.

[2] J. Pearl, System Z: A Natural Ordering of Defaults with Tractable Applications to Nonmonotonic Reasoning, in: Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge, TARK '90, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, p. 121–135.

[3] M. Goldszmidt, J. Pearl, On the relation between rational closure and system-z, in: Proceedings of the Third International Workshop on Nonmonotonic Reasoning, May 31 – June 3, 1990, pp. 130–140.

[4] D. Lehmann, M. Magidor, What does a conditional knowledge base entail?, Artificial Intelligence 55 (1992) 1–60.

[5] M. Goldszmidt, J. Pearl, Qualitative probabilities for default reasoning, belief revision, and causal modeling, Artificial Intelligence 84 (1996) 57–112. URL: https://www.sciencedirect.com/science/article/pii/0004370295000909. doi:https://doi.org/10.1016/0004-3702(95)00090-9.

[6] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artif. Int. Res. 17 (2002) 229–264.

[7] Bryant, Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers C-35 (1986) 677–691.

[8] M. Fujita, P. C. McGeer, J. C.-Y. Yang, Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation, Formal Methods in System Design 10 (1997) 149–169.

[9] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, F. Somenzi, Algebraic decision diagrams and their applications, in: Proceedings of 1993 International Conference on Computer Aided Design (ICCAD), 1993, pp. 188–191. doi:10.1109/ICCAD.1993.580054.

[10] D. Lehmann, Another perspective on default reasoning, Annals of mathematics and artificial intelligence 15 (1995) 61–82.

[11] G. Kern-Isberner, Handling conditionals adequately in uncertain reasoning and belief revision, Journal of Applied Non-Classical Logics 12 (2002) 215–237. doi:10.3166/jancl.12.215-237.

[12] B. De Finetti, Theory of probability (2 vols.), John Wiley & Sons, 1974.

[13] C. Y. Lee, Representation of switching circuits by binary-decision programs, The Bell System Technical Journal 38 (1959) 985–999.

[14] Akers, Binary decision diagrams, IEEE Transactions on Computers C-27 (1978) 509–516.

[15] C. E. Shannon, The synthesis of two-terminal switching circuits, The Bell System Technical Journal 28 (1949) 59–98.

[16] G. Boole, An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities, volume 2, Walton and Maberly, 1854.

[17] B. Bollig, I. Wegener, Improving the variable ordering of OBDDs is NP-complete, IEEE Transactions on Computers 45 (1996) 993–1002.

[18] T. Eiter, T. Lukasiewicz, Default reasoning from conditional knowledge bases: Complexity and tractable cases, Artificial Intelligence 124 (2000) 169–241. URL: https://www.sciencedirect.com/science/article/pii/S0004370200000734. doi:https://doi.org/10.1016/S0004-3702(00)00073-4.

[19] J. Marques Silva, K. Sakallah, GRASP-A new search algorithm for satisfiability, in: Proceedings of International Conference on Computer Aided Design, 1996, pp. 220–227. doi:10.1109/ICCAD.1996.569607.

[20] R. J. Bayardo, R. C. Schrag, Using CSP look-back techniques to solve real-world SAT instances, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence, AAAI'97/IAAI'97, AAAI Press, 1997, p. 203–208.

[21] J. Marques-Silva, K. Sakallah, GRASP: a search algorithm for propositional satisfiability, IEEE Transactions on Computers 48 (1999) 506–521. doi:10.1109/12.769433.

[22] M. Freund, Preferential reasoning in the perspective of Poole default logic, Artificial Intelligence 98 (1998) 209–235. URL: https://www.sciencedirect.com/science/article/pii/S0004370297000532. doi:https://doi.org/10.1016/S0004-3702(97)00053-2.

[23] G. Casini, T. Meyer, I. Varzinczak, Taking defeasible entailment beyond rational closure, in: Logics in Artificial Intelligence: 16th European Conference, JELIA 2019, Rende, Italy, May 7–11, 2019, Proceedings 16, Springer, 2019, pp. 182–197.

[24] C. Beierle, J. Haldimann, L. Schwarzer, CLKR: Conditional Logic and Knowledge Representation, KI - Künstliche Intelligenz 38 (2024) 61–67. URL: http://dx.doi.org/10.1007/s13218-024-00842-z. doi:10.1007/s13218-024-00842-z.

[25] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, F. Pollitt, CaDiCaL 2.0, in: Computer Aided Verification (CAV 2024), volume 14681 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 133–152.

[26] F. Somenzi, Cudd: Cu decision diagram package, release 3.0.0, 2015. URL: https://sourceforge.net/projects/cudd-mirror/files/cudd-3.0.0.tar.gz, accessed 2025-10-16.

[27] C. Beierle, A. Spang, J. Haldimann, Using SAT and Partial MaxSAT for Reasoning with System Z and System W (2024).

[28] H.-T. Liaw, C.-S. Lin, On the OBDD-Representation of General Boolean Functions , IEEE Transactions on Computers 41 (1992) 661–664. URL: https://doi.ieeecomputersociety.org/10.1109/12.144618. doi:10.1109/12.144618.