Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Compiling Defeasible Inference: A Dynamic Approach to System Z

Luke Slater[1] , **Thomas Meyer**[1] , **Jesse Heyninck**[1,2]
[1]University of Cape Town and CAIR, Cape Town, South Africa
[2]Open University, Heerlen, Netherlands
sltluk001@myuct.ac.za, tommie.meyer@uct.ac.za, jesse.heyninck@ou.nl

Abstract

Non-monotonic reasoning is essential for drawing plausible conclusions from incomplete information. Many approaches model changing belief states using Ordinal Conditional Functions (OCFs), which assign degrees of surprise to possible worlds. This paper demonstrates how OCFs are ideally suited for the knowledge compilation paradigm, particularly with Reduced Ordered Binary Decision Diagrams (ROBDDs). We introduce a compilation pipeline for System Z, a prominent ranking-based semantics, which pre-compiles a conditional knowledge base into a set of materialized theories represented by ROBDDs. This compilation enables polynomial-time conditional entailment and efficient, incremental updates, avoiding costly re-computation. We further extend this approach using Algebraic Decision Diagrams (ADDs) to directly compile the entire ranking function, facilitating direct and efficient implementation of complex belief revision operations such as Spohn conditioning.

1 Introduction

Defeasible reasoning formalizes the ability to draw plausible yet defeasible conclusions. Here we focus on default-style approaches that represent such information as a conditional knowledge base—a finite set of defeasible conditionals of the form "if α, then typically β". A variety of semantics for conditional knowledge bases have been studied, a prominent approach models an agent's epistemic state using an Ordinal Conditional Function (OCF) (Spohn 1988). The focus of this paper is Pearl's System Z (Pearl 1990), a canonical ranking-based semantics that constructs an OCF from a knowledge base and uses it to perform defeasible inference. Goldszmidt and Pearl also showed that System Z is equivalent to Lehmann and Magidor's rational closure (Lehmann and Magidor 1992; Goldszmidt and Pearl 1990).

Ranking-based defeasible reasoning, in particular with System Z, is inherently *dynamic*. The central semantic object is not a static knowledge base but a changing belief state that is queried and revised repeatedly as new defaults and observations arrive. Adding a new default to the knowledge base can alter this ranking, and thus change which defeasible conclusions follow. The computational problem is therefore not only to answer a single entailment query, but to maintain and interrogate an evolving ranking under a long sequence of interleaved queries and updates.

In current implementations, the System Z ranking is typically constructed implicitly by first assigning each default to a plausibility stratum via an initial stratification stage that requires many calls to a SAT solver. Conditional entailment is then reduced to a further sequence of SAT checks over these strata. Moreover, when a new default is added to the knowledge base, the preprocessing is rerun from scratch.

From a dynamic point of view, this is conceptually unsatisfying. If belief change is even moderately well behaved, we expect most of the ranking to persist under small updates, with only a comparatively small region of the state space shifting in plausibility. In systems such as System Z, these changes take place in a layered manner, with worlds moving between finitely many ranks and defaults being reassigned to nearby strata while the overall stratification can often remain recognizably intact.

In this paper we propose an alternative perspective based on the knowledge compilation paradigm (Darwiche and Marquis 2002). Instead of treating queries and updates as independent computations over a fixed base, we compile (and maintain) the belief state itself into decision-diagram representations that support fast, local operations. The goal is to use compilation as a principled way to obtain reusable semantic artifacts: inference runs in time polynomial in the size of the compiled representation, and updates can be implemented incrementally by repairing only the affected compiled components.

Concretely, we study two complementary compilation strategies. First, we exploit the layered nature of System Z by compiling its stratified classical theories into Reduced Ordered Binary Decision Diagrams (ROBDDs) (Bryant 1986). This yields a "materialized" backbone of strata that supports polynomial-time conditional entailment with respect to the compiled diagram sizes, and it admits an incremental update procedure that locally repairs the affected layers when new defaults are added. Second, we compile the ranking function itself into an Algebraic Decision Diagram (ADD) (Bahar et al. 1993; Fujita, McGeer, and Yang 1997). In this representation, conditional entailment and belief revision operations such as Spohn conditioning (Spohn 1988) reduce to algebraic transformations on a single multi-valued diagram, again running in time polynomial in the size of the compiled structure.

Our contributions are summarized as follows:

- We present an ROBDD-based compilation pipeline for System Z that materializes its stratified theories as reusable decision-diagram artifacts and supports polynomial-time System Z conditional entailment (in the size of the compiled diagrams).
- We develop an incremental update algorithm that integrates new defaults by locally repairing the compiled strata, enabling efficient reuse rather than full recomputation.
- We compile the entire System Z ranking function into an ADD, yielding polynomial-time procedures (in ADD size) for conditional entailment and for belief revision via Spohn conditioning.
- We prove soundness, completeness, and complexity of the proposed constructions, and provide an initial empirical comparison with a SAT-based baseline.

Roadmap. Section 2 covers preliminaries (OCFs, System Z, and ROBDDs); Section 3 develops the ROBDD materialization with polytime entailment and incremental updates; Section 4 compiles the ranking into ADDs and implements Spohn conditioning; Section 5 evaluates. Supplementary Material A provides pseudocode for existing implementation algorithms; Supplementary Material B contains proofs of all theorems. Supplementary material (proofs and implementation pseudocode) is available at DOI: https://doi.org/10.5281/zenodo.20613113.

2 Preliminaries

2.1 Propositional Logic

Assuming a finite set of n Boolean variables Σ, we fix an ordering $x_0, x_1, \ldots, x_{n-1}$. We denote the general propositional language by $\mathcal{L}$, which is constructed using the usual Boolean connectives. We denote worlds/interpretations by ω. The set of all such interpretations is Ω. We write $\omega \models \varphi$ if ω satisfies φ, and $\omega \not\models \varphi$ otherwise. We write $\varphi \models \vartheta$ if φ entails ϑ, and $\varphi \not\models \vartheta$ otherwise. We use $Mod(\varphi)$ to denote the models of φ.

2.2 Conditional Knowledge Bases

Propositional logic is extended to conditional logic by introducing *defeasible conditionals* (sometimes referred to here as defaults). Conditionals are statements of the form "if α, then typically β," which we will denote here as $r = \alpha \mathrel{|\!\sim} \beta$, with $\alpha, \beta \in \mathcal{L}$. For a world ω, it is useful to define the semantics of conditionals as follows (De Finetti 1974):

$$r(\omega) = \begin{cases} \mathrm{T}, & \text{if } \omega \models \alpha \wedge \beta, \\ \mathrm{F}, & \text{if } \omega \models \alpha \wedge \neg\beta, \\ \times, & \text{otherwise.} \end{cases}$$

A world ω *verifies* a conditional r if $r(\omega) = \mathrm{T}$, *falsifies* it if $r(\omega) = \mathrm{F}$, and is *irrelevant* otherwise (i.e., when $\omega \models \neg\alpha$, denoted $r(\omega) = \times$). A conditional plays a similar role to a classical implication, but with a semantics that makes the antecedent α a strict precondition for evaluation.

Definition 1 (Conditional Knowledge Base)**.** *A conditional knowledge base $\mathcal{K} = (\Gamma, \Delta)$ is a pair where $\Gamma \subseteq \mathcal{L}$ is a finite set of classical propositional formulas, and Δ is a finite set of defeasible conditionals of the form $\alpha \mathrel{|\!\sim} \beta$.*

2.3 Ordinal Conditional Functions

Ordinal conditional functions (OCFs) or *ranking functions* are a well-known semantics for conditional knowledge bases (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. *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 the conditional $\alpha \mathrel{|\!\sim} \beta$ if

$$\kappa(\alpha \wedge \beta) < \kappa(\alpha \wedge \neg\beta).$$

Ranking functions are used for representing a "belief state," and come with various transformations for adapting its beliefs, one good example of this is Spohn conditioning.

Definition 3 (Spohn Conditioning (Spohn 1988))**.** *Let κ be a ranking function and $\varphi \in \mathcal{L}$. Learning φ with firmness $m \in \mathbb{N} \cup \{\infty\}$ yields the revised ranking function:*

$$\kappa_{[\varphi,m]}(\omega) = \begin{cases} \kappa(\omega) - \kappa(\varphi), & \text{if } \omega \models \varphi, \\ \kappa(\omega) - \kappa(\neg\varphi) + m, & \text{if } \omega \models \neg\varphi. \end{cases}$$

Conditioning re-normalizes φ-worlds by subtracting $\kappa(\varphi)$ and penalizes $\neg\varphi$-worlds by m, i.e., a Bayes-style update with a finite likelihood penalty. The parameter m controls the relative penalty assigned to $\neg\varphi$-worlds in the revised ranking: taking $m = \infty$ rules out $\neg\varphi$-worlds, while finite values of m preserve them at a positive rank, thereby retaining more structure for subsequent iterated changes.

2.4 System Z

This paper focuses on System Z, introduced by Pearl (Pearl 1990), and widely recognized as the poster child for ranking-based semantics. System Z partitions Δ into groups of conditionals according to how they "tolerate" one another.

Definition 4 (Tolerance (Pearl 1990))**.** *Let $\mathcal{K} = (\Gamma, \Delta)$, $r \in \Delta$ is tolerated by $\mathcal{K}$ if there exists $\omega \in Mod(\Gamma)$ such that*

$$r(\omega) = T \quad \textit{and} \quad \forall r' \in \Delta : (r'(\omega) \neq F).$$

Tolerated conditionals are intuitively the most broadly applicable or plausible rules in Δ. This suggests that we should separate Δ into layers ordered by tolerance: the first layer collects all maximally "general" defaults, the next layer all those tolerable once the first are set aside, and so on.

Definition 5 (Z-partitioning (Pearl 1990))**.** *Let $\mathcal{K} = (\Gamma, \Delta)$. The Z-partitioning $\Delta^Z_{\mathcal{K}} = (\Delta_0, \Delta_1, \ldots, \Delta_\infty)$ of $\mathcal{K}$ is defined such that $\Delta_0 = \{r \in \Delta \mid \mathcal{K} \text{ tolerates } r\}$ and $(\Delta_1, \ldots, \Delta_\infty)$ is the Z-partitioning of $(\Gamma, \Delta \setminus \Delta_0)$.*

Conditionals that are never tolerated are grouped in Δ_∞. Once each $r \in \Delta$ has an assigned "generality weight", defeasible entailment reduces to a task of model comparison. Intuitively, worlds that falsify only high-weight (i.e. less plausible) defaults are preferred over those that falsify low-weight (more plausible) ones.

Definition 6 (System Z Ranking Function (Pearl 1990))**.** *Let $\Delta_{\mathcal{K}}^{Z} = (\Delta_0, \Delta_1, \ldots, \Delta_\infty)$ be the Z-partitioning of $\mathcal{K}$. The OCF defining the semantics of $\mathcal{K}$ under System Z is*

$$\kappa_Z^{\mathcal{K}}(\omega) = \begin{cases} \infty, & \text{if } \omega \not\models \Gamma, \\ \min\{i \mid \forall r \in \bigcup_{j \geq i} \Delta_j, r(\omega) \neq F\}, & \text{otherwise.} \end{cases}$$

A conditional $\alpha \mid\sim \beta$ is then defeasibly entailed by a conditional knowledge base $\mathcal{K}$ under System Z if it is accepted by the ranking function $\kappa_Z^{\mathcal{K}}$.

Definition 7 (System Z Entailment)**.** *A conditional r is defeasibly entailed by a conditional knowledge base $\mathcal{K}$ according to its System Z semantics if it is accepted by $\kappa_Z^{\mathcal{K}}$.*

$$\mathcal{K} \mid\sim_Z r \iff \kappa_Z^{\mathcal{K}} \models r$$

Henceforth, we assume that $\Delta_{\mathcal{K}}^{Z}$ is given in the form $(\Delta_0, \ldots, \Delta_{k-1}, \Delta_\infty)$, where $k - 1 = \min_i \Delta_i \neq \Delta_\infty$.

An Illustrative Example: Tweety's Changing Prospects We consider an example with the following vocabulary: $t =$ "*is Tweety*," $b =$ "*is a bird*," $n =$ "*is nailed to table*," $r =$ "*is nailed to rocket*," and $f =$ "*can fly*". Along with the following conditional knowledge base $\mathcal{K} = (\Gamma, \Delta)$:

$$\begin{array}{lcl} \Gamma & = & \{t \to b,\ t \to n\} \\ \Delta & = & \{b \mid\sim f,\ b \wedge n \mid\sim \neg f,\ b \wedge n \wedge r \mid\sim f\} \end{array}$$

Although birds typically fly ($b \mid\sim f$), the more specific default $b \wedge n \mid\sim \neg f$ overrides it for birds nailed to a table. System Z partitions Δ into three layers:

$$\begin{array}{lcl} \Delta_0 & = & \{b \mid\sim f\} \\ \Delta_1 & = & \{b \wedge n \mid\sim \neg f\} \\ \Delta_2 & = & \{b \wedge n \wedge r \mid\sim f\} \end{array}$$

System Z therefore yields $\mathcal{K} \mid\approx_Z t \mid\sim \neg f$. However, if we then learn that the table is itself nailed to a rocket and add $t \to r$ to Γ, System Z revises its conclusion: the more specific default now applies, and we obtain $\mathcal{K} \mid\approx_Z t \mid\sim f$.

Implementation The `BaseRank` (Supplementary Material A) algorithm introduced by Casini *et al.* (originally proposed for rational closure) is commonly used to compute the Z-partitioning for a given conditional knowledge base (Casini, Meyer, and Varzinczak 2019), which is based on a procedure originally proposed by Freund (Freund 1998). The algorithm iteratively peels off all rules whose antecedent is still "compatible" with the current remainder of the base, indicating that they are tolerated. The `RationalClosure` (Supplementary Material A) algorithm then successively discards defaults in lower layers until α becomes consistent; the remaining layers then determine whether β is preferred over $\neg\beta$. Both problems are known to be $\mathsf{FP}^{\mathsf{NP}}/\mathsf{P}^{\mathsf{NP}}$-complete (Eiter and Lukasiewicz 2000). In standard practice, each entailment check is performed using a regular SAT solver. In the worst case, `BaseRank` performs $|\Delta|^2$ entailment tests, while `RationalClosure` needs at most $|\Delta|$ per query.

2.5 ROBDD Based Knowledge Compilation

Knowledge compilation addresses the inherent intractability of general propositional reasoning by splitting it into two distinct phases (Darwiche and Marquis 2002):

Off-line. A propositional formula $\varphi \in \mathcal{L}$ is preprocessed—potentially at exponential cost—into an equivalent form $\vartheta \in \mathcal{T}$, where $\mathcal{T}$ is chosen specifically according to its tractability properties.

On-line. Once compiled, otherwise intractable reasoning tasks can be performed in polynomial time relative to $|\vartheta|$.

A target language must support polynomial-time queries not to simplify all tasks, but to shift complexity into the compiled form itself—any inherent hardness appears as increased representation size. Optimally compiling a formula then reduces to finding its smallest equivalent in the target language. If a compact form is found, subsequent queries can be computed in time proportional to its size, providing a useful predictor of online performance. Once a complex (sub)problem is compiled, its result can then amortize the initial compilation cost online, and can even be used in solving other seemingly unrelated problems that share common subproblems . ROBDDs are a well-known and highly effective example of this principle (Bryant 1986).

Definition 8 (ROBDD)**.** *A BDD G is a rooted DAG. Each node ν is either a branch node, assigned an index $\mathrm{I}(\nu)$ and having two children $\mathrm{T}(\nu)$ and $\mathrm{F}(\nu)$, or a sink node, with a value $\mathrm{V}(\nu) \in \{1, 0\}$. A ROBDD is a BDD that is:*

- *Ordered: For every branch node ν, if ν' is a child branch node, then $\mathrm{I}(\nu) < \mathrm{I}(\nu')$.*
- *Reduced: It contains no isomorphic sub-diagrams and no ν where $\mathrm{T}(\nu) = \mathrm{F}(\nu)$.*

Its size $|G|$ is the total number of nodes.

The semantics of ROBDDs are defined as follows. For any Boolean function f denote its positive and negative cofactors with respect to variable x_i by

$$\begin{array}{lcl} f_{x_i} & = & f(x_0, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_{n-1}) \\ f_{\overline{x}_i} & = & f(x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_{n-1}) \end{array}$$

A ROBDD G with root ν encodes a unique function f defined recursively via Shannon expansion (Shannon 1949):

$$f = \begin{cases} \mathrm{V}(\nu) & \text{if } \nu \text{ is a sink,} \\ (x_i \cdot f_{x_i}) + (\overline{x}_i \cdot f_{\overline{x}_i}) & \text{otherwise,} \end{cases}$$

where the diagrams rooted at $\mathrm{T}(\nu)$ and $\mathrm{F}(\nu)$ encode f_{x_i} and $f_{\overline{x}_i}$. Every propositional formula $\varphi \in \mathcal{L}$ therefore has a canonical ROBDD G_φ representing its Boolean function f_φ. Reduction guarantees canonicity by "skipping" any variable x_i that a sub-function does not depend on ($f_{x_i} = f_{\overline{x}_i}$). After compilation, model checking takes $\mathcal{O}(n)$ time, model counting $\mathcal{O}(|G|)$, and equivalence testing $\mathcal{O}(1)$. Canonicity means each function is completely determined by its node triple (I, T, F); this locality lets recursive algorithms cache and reuse results via dynamic programming. Canonicity also reduces optimal search to the task of finding a good variable ordering, which is NP-complete (Bollig and Wegener 1996).

Operations The `apply` algorithm is a recursive procedure that computes the ROBDD for $f \diamond g$, given the ROBDDs for f and g and any Boolean operator $\diamond$. At its core is the

observation that if we choose the top-most variable to split on, say x_i, the expansion for the combined function is:

$$f \diamond g \;=\; \overline{x}_i \cdot (f_{\overline{x}_i} \diamond g_{\overline{x}_i}) \;+\; x_i \cdot (f_{x_i} \diamond g_{x_i})$$

A recursive dynamic programming procedure traverses both diagrams in parallel while constructing the result of the operation. Memoization ensures that each node pair is processed at most once, yielding a worst-case complexity of $\mathcal{O}(|G_1| \cdot |G_2|)$, where G_1 and G_2 are the input diagrams. Folding `apply` over a sequence $G_0 \diamond_0 G_1 \diamond_1 \cdots \diamond_{i-1} G_i$ remains polynomial for fixed (bounded) i, but if i is unbounded the intermediate or resulting ROBDDs may blow up (Bryant 1986; Darwiche and Marquis 2002). Negation is made $\mathcal{O}(1)$ through the use of complemented edges (Brace, Rudell, and Bryant 1990).

Implementation A multi-rooted, shared DAG with a global variable ordering is used to represent a set of functions: each node is in one-to-one correspondence with a function f and points to the nodes of its cofactors f_{x_i} and $f_{\overline{x}_i}$. A global hash table called the *unique table* maps every triple to its node in the DAG. A global cache called the *computed table* serves as a memory function for all recursive algorithms (Brace, Rudell, and Bryant 1990); for `apply`, it maps the triples of any two functions together with an operator ID to the triple of the result. Optimal search is performed through another dynamic-programming-style technique known as *dynamic variable reordering* (Rudell 1993), which swaps adjacent variables and updates the DAG locally instead of reconstructing it from scratch. Most search heuristics are then built around this variable swap primitive. Modern BDD libraries such as CUDD (Somenzi 1997; Somenzi 2015) embody decades of optimization: they hide these mechanisms behind a simple API, run extremely efficiently, and perform optimal search automatically through a large suite of built-in dynamic reordering heuristics (Rudell 1993; Drechsler, Becker, and Göckel 1996; Aloul, Markov, and Sakallah 2003; Bollig, Löbbing, and Wegener 1996; Drechsler, Drechsler, and Günther 1998).

Algebraic Decision Diagrams Algebraic Decision Diagrams (ADDs) are the multi-terminal analogue of OBDDs: sinks store values from an algebraic domain D (e.g., $\mathbb{N} \cup \{\infty\}$) (Bahar et al. 1993; Fujita, McGeer, and Yang 1997). With a fixed variable order, the standard ordering/reduction rules still yield a canonical DAG for each $f : \mathbb{B}^n \to D$, so all the infrastructure described previously for ROBDDs applies *for free* to ADDs. The `apply` recursion is identical, replacing the Boolean operator by any algebraic operator $\star : D \times D \to D$. Mature libraries like CUDD already support ADDs and their operations, allowing both Boolean and pseudo-Boolean reasoning within a unified environment.

3 ROBDD Based System Z

We now instantiate the knowledge-compilation viewpoint for System Z using classical ROBDDs as the target language. The aim is to replace repeated SAT-based optimisation by a one-off compilation step, after which the core System Z tasks are mediated by decision diagrams rather than by fresh calls to a solver. Throughout, products like $\boldsymbol{\Gamma} \cdot \boldsymbol{\varphi}$ denote a single `apply` call with conjunction, and $\prod S$ is the left-associative fold of conjunction over the set S. All complexity bounds in this section assume a fixed variable ordering. Set operations on ROBDDs are pointer-level because nodes are uniquely shared via the global unique table and referenced by ID.

3.1 Compilation and Preprocessing

Utilizing ROBDDs requires a systematic way to represent a conditional expression $\alpha \mid\sim \beta$ using classical formulas. A straightforward option is to encode α and β as two separate ROBDDs. While this is correct, it is not aligned with how System Z algorithms actually use conditionals. In practice, they rely on two components: the *applicability condition* α, and the *materialization* $\alpha \to \beta$, which specifies what must hold whenever the condition applies.

Definition 9. *Let $r = \alpha \mid\sim \beta$ be a conditional. Its ROBDD-based compilation is the pair*

$$\boldsymbol{r} \;=\; (\mathsf{ROBDD}(\alpha), \mathsf{ROBDD}(\alpha \to \beta)).$$

We use $\mathsf{ROBDD}(\varphi)$ to denote the bottom-up compilation of φ into its ROBDD representation $\boldsymbol{\varphi}$ using `apply`. Since α and β can be any general propositional formula, compilation from r to $\boldsymbol{r}$ is exponential in the worst case. We will write $\boldsymbol{r}[0]$ for the compiled antecedent $\mathsf{ROBDD}(\alpha)$ and $\boldsymbol{r}[1]$ for the compiled materialization $\mathsf{ROBDD}(\alpha \to \beta)$. Note that $\overline{\boldsymbol{r}[1]}$ (complemented edge) encodes $\neg(\alpha \to \beta) \equiv \alpha \wedge \neg\beta$, which we will use repeatedly for entailment checks.

Definition 10. *Let $\mathcal{K} = (\Gamma, \Delta)$ be a conditional knowledge base, with $\Delta = \{r_j\}_{j=0}^{m-1}$. Its ROBDD-based compilation is the pair*

$$\boldsymbol{\mathcal{K}} \;=\; (\boldsymbol{\Gamma}, \boldsymbol{\Delta}).$$

Where $\boldsymbol{\Gamma} = \mathsf{ROBDD}(\bigwedge \Gamma)$ and $\boldsymbol{\Delta} = \{\boldsymbol{r}_j\}_{j=0}^{m-1}$.

The pair $(\boldsymbol{\Gamma}, \boldsymbol{\Delta})$ therefore gives a reusable compiled vocabulary for the knowledge base: once this has been constructed, it does not need to be recompiled, and any further compilation only concerns formulas that arise later in queries or updates. On its own, however, this representation does not yet support our inference algorithms in polynomial time. A further preprocessing step is required to reorganize the compiled knowledge base according to System Z semantics. The first step involves assigning each compiled conditional its corresponding plausibility rank.

Definition 11. *Let $\Delta_{\mathcal{K}}^{Z} = (\Delta_0, \ldots, \Delta_{k-1}, \Delta_\infty)$ be the Z-partitioning of $\mathcal{K}$. Its ROBDD-based compilation is*

$$\boldsymbol{\Delta}_{\mathcal{K}}^{\boldsymbol{Z}} = (\boldsymbol{\Delta}_0, \ldots, \boldsymbol{\Delta}_{k-1}, \boldsymbol{\Delta}_\infty).$$

Where $\boldsymbol{\Delta}_i = \{\boldsymbol{r} \mid r \in \Delta_i\}$.

While $\boldsymbol{\Delta}_{\mathcal{K}}^{\boldsymbol{Z}}$ keeps track of the rank of each compiled conditional, System Z inference does not operate directly on these sets, but on the materialized classical theories they induce. For each rank i, one considers a theory Γ_i that combines Γ with the materializations of all defaults in layers Δ_j with $j \geq i$. The resulting sequence $\Gamma_0, \Gamma_1, \ldots, \Gamma_{k-1}$ is the Z–materialization of $\mathcal{K}$, and its compiled form will serve as

the main semantic backbone for our ROBDD-based procedures. The models of Γ_i are precisely the worlds whose System Z rank is at most i.

Definition 12. *The ROBDD-based materialization of $\mathcal{K}$ under its System Z semantics is the tuple*

$$\mathbf{\Gamma}^{\mathbf{Z}}_{\mathcal{K}} = (\mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_{k-1}, \mathbf{\Gamma}_\infty).$$

Where for each stratified layer $\mathbf{\Gamma}_i$ (for $i < k$) we have

$$\mathbf{\Gamma}_i = \prod \bigcup_{j \geq i} \{\boldsymbol{r}[1] \mid \boldsymbol{r} \in \mathbf{\Delta}_j\} \cdot \mathbf{\Gamma}.$$

Figure 1 shows an optimally sized shared DAG for the Z-materialization of the Tweety example. The substantial node sharing reflects semantic overlap between the materialized layers.

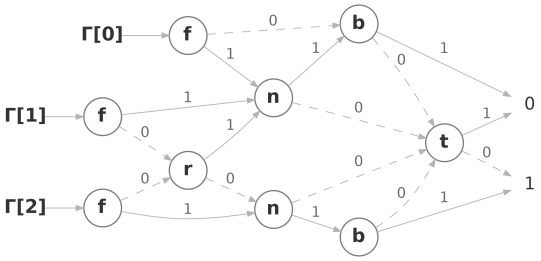

Figure 1: Shared DAG representing the materialization of the Tweety example.

This materialization will not only support predictable and efficient inference, it will also enable caching, incremental compilation, and component reuse. The overall preprocessing step is summarized by the `Preprocess` algorithm. After preprocessing, the shared size of $\mathbf{\Gamma}^{\mathbf{Z}}_{\mathcal{K}}$ should be minimized using optimal search strategies like dynamic variable reordering. This will have a direct effect on the efficiency of inference and update.

Algorithm 1 `Preprocess`

Input: $\mathcal{K} = (\mathbf{\Gamma}, \mathbf{\Delta})$
Output: $\mathbf{\Gamma}^{Z}_{\mathcal{K}} = (\mathbf{\Gamma}_0, \ldots, \mathbf{\Gamma}_{k-1})$, $\mathbf{\Delta}^{\mathbf{Z}}_{\mathcal{K}} = (\mathbf{\Delta}_0, \ldots, \mathbf{\Delta}_{k-1}, \mathbf{\Delta}_\infty)$
$i \leftarrow 0$
$E_i \leftarrow \mathbf{\Delta}$
while $E_i \neq E_{i-1}$ **do**
    $\mathbf{\Gamma}_i \leftarrow \prod\{\boldsymbol{r}[1] \mid \boldsymbol{r} \in E_i\} \cdot \mathbf{\Gamma}$
    $E_{i+1} \leftarrow \{\boldsymbol{r} \in E_i \mid \mathbf{\Gamma}_i \cdot \boldsymbol{r}[0] = 0\}$
    $\mathbf{\Delta}_i \leftarrow E_i \setminus E_{i+1}$
    $i \leftarrow i + 1$
$\mathbf{\Delta}_\infty \leftarrow E_{i-1}$
$\mathbf{\Gamma}_\infty \leftarrow \mathbf{\Gamma}_{i-1}$
if $E_{i-1} = \emptyset$ **then**
    $k \leftarrow i - 1$
else
    $k \leftarrow i$
return $(\mathbf{\Gamma}_0, \ldots, \mathbf{\Gamma}_{k-1}), (\mathbf{\Delta}_0, \ldots, \mathbf{\Delta}_{k-1}, \mathbf{\Delta}_\infty)$

Theorem 1. *The time complexity of the `Preprocess` algorithm is bounded by*

$$\mathcal{O}\Big(\big(|\mathbf{\Gamma}| + m \cdot \max_{\boldsymbol{r} \in \mathbf{\Delta}} |\boldsymbol{r}[0]|\big) \cdot \prod_{j=0}^{m-1} \max_{\boldsymbol{r} \in \mathbf{\Delta}} |\boldsymbol{r}[1]|\Big),$$

and requires exponential time in the worst case.

Theorem 2. *`Preprocess` is sound/complete for computing $\mathbf{\Gamma}^{\mathbf{Z}}_{\mathcal{K}}$ and $\mathbf{\Delta}^{\mathbf{Z}}_{\mathcal{K}}$.*

3.2 Inference

We now focus on two important queries that can now be answered in polynomial time given the compiled materialization of $\mathcal{K}$. First, given the OBDD encoding $\boldsymbol{\varphi}$, we can compute $\kappa^{\mathcal{K}}_{Z}(\varphi)$ with a binary search over $\mathbf{\Gamma}^{\mathbf{Z}}_{\mathcal{K}}$ using the `MinTheory` algorithm, which returns k if $\kappa^{\mathcal{K}}_{Z}(\varphi) = \infty$. This is possible because the sequence of materialized theories in $\mathbf{\Gamma}^{\mathbf{Z}}_{\mathcal{K}}$ becomes strictly weaker with increasing i. The primary operation performed at each iteration of the search is $\mathbf{\Gamma}_i \cdot \boldsymbol{\varphi} = 0$, which is performed at most $\log k$ times. This general approach is still closely based on the `RationalClosure` (Supplementary Material A) algorithm.

Algorithm 2 `MinTheory`

Input: $\mathbf{\Gamma}^{\mathbf{Z}}_{\mathcal{K}} = (\mathbf{\Gamma}_0, \ldots, \mathbf{\Gamma}_{k-1})$, $\boldsymbol{\varphi}$
Output: $\min\{\, i < k \mid \mathbf{\Gamma}_i \cdot \boldsymbol{\varphi} \neq 0\} \cup \{k\}$
$low \leftarrow 0, \quad high \leftarrow k$
while $low < high$ **do**
    $mid \leftarrow \lfloor (low + high)/2 \rfloor$
    if $mid < k$ **and** $\mathbf{\Gamma}_{mid} \cdot \boldsymbol{\varphi} \neq 0$ **then**
        $high \leftarrow mid$
    else
        $low \leftarrow mid + 1$
return low

Theorem 3. *The time complexity of the `MinTheory` algorithm is bounded by*

$$\mathcal{O}\Big(\big(\max_{j} |\mathbf{\Gamma}_j|\big) \cdot |\boldsymbol{\varphi}| \cdot \log k\Big),$$

and requires polynomial time in the worst case.

Theorem 4. *`MinTheory` is sound/complete for computing $\kappa^{\mathcal{K}}_{Z}(\varphi)$.*

Algorithm 3 `Inference`

Input: $(\mathbf{\Gamma}_0, \ldots, \mathbf{\Gamma}_{k-1})$, $\boldsymbol{r}$
Output: 1 if inference, else 0
$i \leftarrow$ `MinTheory`$((\mathbf{\Gamma}_0, \ldots, \mathbf{\Gamma}_{k-1}), \boldsymbol{r}[0])$
if $i = k$ **then**
    $result \leftarrow 1$
else
    $result \leftarrow \mathbf{\Gamma}_i \cdot \overline{\boldsymbol{r}[1]} = 0$
return $result$

The `MinTheory` algorithm is then directly applicable to checking entailment of a conditional r given $\boldsymbol{r}$, as summarized by the `Inference` algorithm. A call to `MinTheory` returns the minimal rank i where the antecedent is consistent. The algorithm then performs a final check $\boldsymbol{\Gamma}_i \cdot \overline{\boldsymbol{r}[1]} = 0$ to determine if any worlds at that rank satisfy $\alpha \wedge \neg\beta$.

Theorem 5. *The time complexity of the* `Inference` *algorithm is bounded by*

$$\mathcal{O}\Big(\big(\max_j |\boldsymbol{\Gamma}_j|\big) \cdot \big(|\boldsymbol{r}[0]| \cdot \log k + |\boldsymbol{r}[1]|\big)\Big),$$

and requires polynomial time in the worst case.

Theorem 6. `Inference` *is sound/complete for System Z conditional entailment.*

3.3 Update

Updates are typically addressed by rerunning the `BaseRank` (Supplementary Material A) algorithm from scratch. In a compiled setting, however, this is inefficient. Instead of rebuilding the ranking, updates can be made through local, layered adjustments to the existing compilation.

Algorithm 4 `Update`

```
1: In: Γ, (Γ_0, ..., Γ_{k-1}, Γ_∞), (Δ_0, ..., Δ_{k-1}, Δ_∞), r
2: Out: new (Γ_0, ..., Γ_{k-1}, Γ_∞), (Δ_0, ..., Δ_{k-1}, Δ_∞)
3: v ← 0,   C ← {r}
4: i ← MinTheory((Γ_0, ..., Γ_{k-1}), r[0])
5: while C ≠ ∅ do
6:     if i = k then
7:         Γ_i ← Γ · Γ_∞,  Δ_i = ∅,  k ← k + 1
8:     Δ_i ← Δ_i ∪ C
9:     Γ_{i-v} ← ∏{r'[1] | r' ∈ C} · Γ_i
10:    C ← {r' ∈ Δ_i | Γ_{i-v} · r'[0] = 0}
11:    if |C| ≠ |Δ_i| then
12:        Δ_{i-v} ← Δ_i \ C
13:    else
14:        v ← v + 1
15:        if i = k − 1 then
16:            Γ_∞ ← Γ_∞ · Γ_{i-v}
17:            Δ_∞ ← Δ_∞ ∪ C,  C ← ∅
18:    i ← i + 1
19: k ← k − v
20: return (Γ_0, ..., Γ_{i-v-1}, Γ_i, ..., Γ_{k+v-1}, Γ_∞),
    (Δ_0, ..., Δ_{i-v-1}, Δ_i, ..., Δ_{k+v-1}, Δ_∞)
```

The `Update` algorithm incrementally integrates a new conditional r into the compiled Z-ranking. Line 4 finds the least i whose theory is consistent with r's antecedent; adding r's materialization below this point is redundant because those layers already entail $\neg r[0]$. The loop then walks upward from i. If $i = k$, we materialize a fresh top rank as $\boldsymbol{\Gamma} \cdot \boldsymbol{\Gamma}_\infty$, set $\boldsymbol{\Delta}_i = \emptyset$, and increase k. At each step we carry the current bump set $\mathcal{C}$ by inserting it into the read-side old layer $\boldsymbol{\Delta}_i$ (line 8), and we write the compacted theory at index $i{-}v$ by conjoining it only with the newly bumped conditionals (line 9). We then recompute the next bump set as those rules in $\boldsymbol{\Delta}_i$ whose antecedents are inconsistent under $\boldsymbol{\Gamma}_{i-v}$ (line 10). If all rules in $\boldsymbol{\Delta}_i$ are bumped (line 11), the layer vanishes and we increment v; if this occurs at the last finite layer ($i = k-1$), we finish by updating $\boldsymbol{\Gamma}_\infty$ and $\boldsymbol{\Delta}_\infty$, then clear $\mathcal{C}$. Otherwise (partial bump), we write the survivors to $\boldsymbol{\Delta}_{i-v}$, increment i, and continue. Reads always use the old index i; writes use $i{-}v$ to keep the partition contiguous. The algorithm is likely to be most efficient when changes are local—that is, when r bumps few conditionals and the cascade is shallow.

Theorem 7. *The time complexity of the* `Update` *algorithm is bounded by* ($\boldsymbol{\Delta}' = \boldsymbol{\Delta} \cup \{\boldsymbol{r}\}$)

$$\mathcal{O}\Big(\big(|\boldsymbol{\Gamma}| + (m+1) \cdot \max_{\boldsymbol{r'} \in \boldsymbol{\Delta}'} |\boldsymbol{r'}[0]|\big) \cdot \prod_{j=0}^{m} \max_{\boldsymbol{r'} \in \boldsymbol{\Delta}'} |\boldsymbol{r'}[1]|\Big),$$

and requires exponential time in the worst case.

Theorem 8. `Update` *is sound/complete for single-rule insertion under System Z (partitioning and materialization).*

4 ADD Based System Z

We now shift from ROBDD materializations to an ADD encoding of the ranking function. ADDs provide a natural extension that supports direct representation of the ranking function itself. Viewed as a Shannon expansion of κ, the ADD factors per-variable choices while exposing *inter*-rank dependencies, complementing the *intra*-rank theories captured by ROBDDs.

In what follows, a ITE (if-then-else) operator will be used to mediate between ROBDDs and ADDs. Given a Boolean decision diagram $\boldsymbol{\varphi}$ and two ADDs $\boldsymbol{F}$ and $\boldsymbol{G}$, the diagram $\mathsf{ITE}(\boldsymbol{\varphi}, \boldsymbol{F}, \boldsymbol{G})$ represents the function that maps each world ω to $\boldsymbol{F}(\omega)$ if $\omega \models \varphi$ and to $\boldsymbol{G}(\omega)$ otherwise. Operationally, ITE is a primitive ternary version of the usual binary `apply` operator and has a worst case complexity of $\mathcal{O}(|\boldsymbol{\varphi}| \cdot |\boldsymbol{F}| \cdot |\boldsymbol{G}|)$. For a ranking ADD $\boldsymbol{H}$ we write $\min(\boldsymbol{H})$ for its smallest terminal value. Binary operations on ranking ADDs are taken pointwise using the tropical (min–plus) semiring $(\mathbb{N} \cup \{\infty\}, \cdot, +)$, where for $a, b \in \mathbb{N} \cup \{\infty\}$ we set $a \cdot b := \min\{a, b\}$ and let $a+b$ be ordinary addition extended by $a + \infty = \infty + a = \infty$.

4.1 Preprocessing

The ADD-based compilation of the System Z ranking function reuses the ROBDD representations of the Z-materialisation layers $\boldsymbol{\Gamma}^{\boldsymbol{Z}}_{\boldsymbol{\mathcal{K}}} = (\boldsymbol{\Gamma}_0, \ldots, \boldsymbol{\Gamma}_k)$. The key observation is that each $\boldsymbol{\Gamma}_i$ already identifies all worlds whose rank is at most i. Turning $\boldsymbol{\Gamma}_i$ into an ADD that assigns rank i to exactly those worlds, and ∞ to all others, is simply an instance of the ITE operator: we use $\boldsymbol{\Gamma}_i$ as the Boolean guard and choose between the constant ADDs i and ∞.

Definition 13. *Let* $\boldsymbol{\Gamma}^{\boldsymbol{Z}} = (\boldsymbol{\Gamma}_0, \ldots, \boldsymbol{\Gamma}_k)$ *be the ROBDD compilation of the Z-materialisation. The ADD compilation of the System Z ranking function is*

$$\boldsymbol{\kappa}^{\boldsymbol{Z}}_{\boldsymbol{\mathcal{K}}} \;=\; \prod_{i=0}^{k-1} \mathsf{ITE}\big(\boldsymbol{\Gamma}_i,\; \boldsymbol{i},\; \boldsymbol{\infty}\big), \tag{1}$$

where the product is taken with respect to the pointwise minimum, and $\boldsymbol{i}$ and $\boldsymbol{\infty}$ are the constant (single terminal) ADDs that assign these values to every world.

Intuitively, $\boldsymbol{\kappa}^{\boldsymbol{Z}}_{\mathcal{K}}$ is an additional compilation layer on top of the ROBDD-based Z-materialization: it does not replace the Boolean strata, but summarizes them at the level of the full ranking function. This summary is particularly useful for tasks that manipulate ranks directly, such as belief revision or comparison of worlds. Constructing $\boldsymbol{\kappa}^{\boldsymbol{Z}}_{\mathcal{K}}$ requires k applications of ITE and $k-1$ applications of the semiring product and, like the underlying ROBDD preprocessing, is an offline procedure with worst-case exponential complexity.

Theorem 9. *The time complexity of the procedure described by Equation 1 is bounded by*

$$\mathcal{O}\Big(\prod_{i=0}^{k-1} |\boldsymbol{\Gamma}_i|\Big),$$

and requires exponential time in the worst case.

Theorem 10. *Equation 1 is sound and complete for compiling the ADD representation $\boldsymbol{\kappa}^{\boldsymbol{Z}}_{\mathcal{K}}$ of the ranking function $\kappa^{Z}_{\mathcal{K}}$.*

Figure 2 shows the same Tweety example represented as an ADD. Compared with the shared DAG in Figure 1, the ADD appears to exploit rank-level dependencies more directly, yielding a more compact representation.

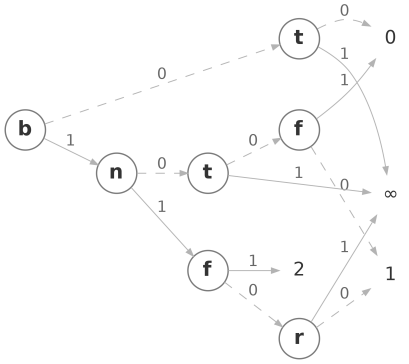

Figure 2: ADD representation of Tweety example.

4.2 Inference

With the entire ranking function captured in a single ADD $\boldsymbol{\kappa}^{\boldsymbol{Z}}_{\mathcal{K}}$, inference tasks that were complex and difficult to implement in the SAT-based setting become straightforward algebraic manipulations. We illustrate this here with Spohn conditioning and conditional entailment.

Spohn Conditioning. Given the compiled ROBDD representation $\boldsymbol{\varphi}$ of φ the conditioning operation on a compiled ranking function $\boldsymbol{\kappa}$ can be implemented in two simple steps. First, we read off the offsets

$$k_\varphi \;=\; \min\big(\mathsf{ITE}(\boldsymbol{\varphi}, \boldsymbol{\kappa}, \boldsymbol{\infty})\big),\; k_{\neg\varphi} \;=\; \min\big(\mathsf{ITE}(\overline{\boldsymbol{\varphi}}, \boldsymbol{\kappa}, \boldsymbol{\infty})\big),$$

which coincide with $\kappa(\varphi)$ and $\kappa(\neg\varphi)$, respectively. Intuitively, the two ITE calls create separate "views" of the ranking in which either the φ-worlds or the $\neg\varphi$-worlds keep their original ranks while all other worlds are sent to ∞, so that the minimum in each view simply returns the rank of the best φ- and $\neg\varphi$-worlds. Second, we form two shifted copies of $\boldsymbol{\kappa}$ and use the Boolean diagram for φ as a guard:

$$\boldsymbol{\kappa}_{[\varphi,m]} \;=\; \mathsf{ITE}\big(\boldsymbol{\varphi},\; \boldsymbol{\kappa} - k_\varphi,\; \boldsymbol{\kappa} - k_{\neg\varphi} + m\big). \qquad (2)$$

Conceptually, this pastes the two shifted rankings back into a single diagram: on φ-worlds it reports the renormalised ranks starting at 0, and on $\neg\varphi$-worlds it reports the renormalised ranks starting at m, while worlds that are already impossible remain at ∞. Here, the symbols "$+$" and "$-$" denote ordinary integer addition and subtraction on the terminal values of the ranking ADD, applied pointwise; in an implementation such as CUDD these shifts are realised by simple linear traversals of the diagrams. In the special case $m = \infty$, the second branch collapses to the constant $\boldsymbol{\infty}$, so the mask and offset for $\neg\varphi$ need not be constructed.

Theorem 11. *The conditioning operator given in Equation* (2) *can be computed with a worst–case running time*

$$\mathcal{O}\big(|\boldsymbol{\varphi}| \cdot |\boldsymbol{\kappa}|^2\big),$$

and therefore runs in polynomial time in the sizes of the input diagrams.

Theorem 12. *Equation 2 is sound/complete for Spohn conditioning.*

Conditional Entailment. Let $\boldsymbol{\kappa}$ be the ADD encoding the ranking κ, and let $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ be the compiled decision diagrams for α and β. Then, in the non-vacuous case where α has at least one model, the acceptance test for a conditional $\alpha \mid\!\sim \beta$ can be expressed as

$$\min\big(\mathsf{ITE}(\boldsymbol{\alpha}\cdot\boldsymbol{\beta},\, \boldsymbol{\kappa},\, \boldsymbol{\infty})\big) \;<\; \min\big(\mathsf{ITE}(\boldsymbol{\alpha}\cdot\overline{\boldsymbol{\beta}},\, \boldsymbol{\kappa},\, \boldsymbol{\infty})\big). \qquad (3)$$

If instead $\min(\mathsf{ITE}(\boldsymbol{\alpha}, \boldsymbol{\kappa}, \boldsymbol{\infty})) = \infty$, then no world satisfies α and the conditional is accepted vacuously under the usual convention.

When conditionals are available in compiled form, as in the ROBDD-based pipeline, we can reuse this structure directly. A compiled conditional $\boldsymbol{r}$ stores $\boldsymbol{r}[0] = \mathsf{ROBDD}(\alpha)$ and $\boldsymbol{r}[1] = \mathsf{ROBDD}(\alpha \to \beta)$. Since $\alpha \wedge (\alpha \to \beta)$ is classically equivalent to $\alpha \wedge \beta$, the Boolean mask for $\alpha \wedge \beta$ can be taken as $\boldsymbol{r}[0] \cdot \boldsymbol{r}[1]$. For the counterexamples we can omit $\boldsymbol{r}[0]$: the negation $\overline{\boldsymbol{r}[1]}$ encodes $\neg(\alpha \to \beta)$, which is equivalent to $\alpha \wedge \neg\beta$, so it already serves as the mask for $\alpha \wedge \neg\beta$.

Theorem 13. *The time complexity of the entailment test in Equation 3, applied to a compiled conditional $\boldsymbol{r}$, is bounded by*

$$\mathcal{O}\big(|\boldsymbol{\kappa}| \cdot |\boldsymbol{r}[0]| \cdot |\boldsymbol{r}[1]|\big),$$

and is polynomial in the sizes of $\boldsymbol{\kappa}$ and $\boldsymbol{r}$ in the worst case.

Theorem 14. *The ADD-based test in Equation* (3) *is sound and complete for defeasible entailment with respect to the ranking κ.*

Remark. A full overview of OCF-based queries and transformations is beyond the scope of this paper. The ADD encoding permits several additional operations to be carried out in polynomial time with respect to the size of the compiled diagram. Polynomial-time model counting over ADDs is especially useful for aggregating ranks over sets of worlds. The encoding also enables $O(1)$ equivalence checking between compiled OCFs (via root/pointer identity).

5 Experimental Results

5.1 Experiment Setup

We implemented all algorithms in C++ using CUDD (Somenzi 2015). For comparison, we also developed SAT-based variants using the more lightweight Kissat SAT solver (Biere et al. 2020). We additionally tested more heavyweight solvers like CaDiCaL and Z3, but given the relatively small problem sizes, solver overhead dominated runtime, exaggerating ROBDD speedup. Experiments were run on a 16GB M1 Pro.

Because there are currently no large, real-world datasets for conditional logic, our evaluation primarily relies on randomly generated benchmarks, supplemented by two non-random medium-sized datasets from the medical domain. Results are presented in two blocks:

1. *Synthetic benchmarks:* Based on the publicly available *CLKR-PS004* benchmark (Beierle, Haldimann, and Schwarzer 2024). Each row represents 100 randomly generated datasets, each with N propositional variables and N defeasible conditionals, plus 10 inference/update conditionals. Preprocessing times are averaged over 100 datasets, while inference and update times are averaged over 1000 queries.
2. *Medical datasets:* Two publicly available datasets (Beierle, Haldimann, and Schwarzer 2024). **M1** corresponds to `med0004_anaemiengesamt` (24 variables, 15 conditionals), and **M2** to `med0001_thoraxschmerz` (61 variables, 33 conditionals). Inference and update times are averaged over 10 queries each. These were the two largest publicly available conditional knowledge bases we could locate.

We cap experiments at $N \leq 40$ for pragmatic reasons. Random Boolean functions are a known worst case for ROBDDs, with sizes practically guaranteed to grow rapidly regardless of variable ordering (Wegener 1994), leading to diminishing returns as N grows. These results serve as a proof-of-concept illustration of how the proposed approaches compare despite the adversarial conditions for ROBDDs.

5.2 Main Comparison

In Table 1 we report: *Total preprocessing* (SAT: CNF conversion + preprocess; ROBDD: compile + preprocess + reorder), *Inference* (inference time + query compilation/CNF conversion), and *Update* (update time + query compilation/CNF conversion). Columns of the form SAT/X report the speedup ratio obtained by dividing the SAT baseline runtime by the runtime of method X. For the compiled engine, *total preprocessing* includes compilation of each conditional into ROBDD form, the preprocessing algorithm on the compiled structures, and a single dynamic variable reordering pass using basic sifting (Rudell 1993), after which the variable order is fixed.

We draw three main points from our evaluation. First, although ROBDD sizes grow rapidly on random formulas, they still deliver substantial online speedups for both inference and updates when $N \leq 30$. Second, SAT-based updates offer little benefit over standard preprocessing, whereas our incremental `Update` on ROBDDs leverages component reuse to achieve clear gains. Third, on the structured medical datasets, the decision-diagram methods sustain their advantage—even at $N = 61$—demonstrating practical scalability beyond synthetic benchmarks. Interestingly, on these larger structured instances the ADD compilation yields faster inference than the ROBDD materialization, as one would expect. While the extra terminal values can make ADDs larger on random formulas, on structured knowledge bases they can better exploit rank-level dependencies induced by the conditional base.

Amortisation. A simple break-even calculation shows when the additional offline preprocessing cost of compilation is amortised by faster online operations. For the synthetic benchmarks, this break-even point is typically reached after only a handful of inference queries at moderate sizes (about 10 inferences at $N = 6$, 5 at $N = 12$, and 8 at $N = 20$), but it can grow substantially when compilation becomes expensive (about 91 inferences at $N = 30$). On the structured medical datasets, the picture is more favorable: M1 breaks even after roughly one inference, while M2 amortises after about 43 inference queries.Updates shorten these break-even points dramatically. Already at $N = 12$ and $N = 20$, a single update compensates the extra preprocessing cost, and even at $N = 30$ the overhead is recovered after about 6 updates. For the medical datasets, compilation is likewise quickly repaid by updates: M2 breaks even after roughly 3 updates, and on M1 a single update more than pays for compilation. Overall, whenever the compiled diagrams remain reasonably compact, the one-off compilation overhead is justified after on the order of tens of mixed inference/update operations (and often much sooner when updates are frequent); although our evaluation is constrained by the limited availability of large, real-world conditional-logic datasets, the results consistently indicate that compilation can pay off in the intended dynamic regime of many interleaved queries and updates. Of course, whether SAT solving or knowledge compilation is preferable will always depend on the problem: SAT may win when preprocessing cost is critical, while compiled representations pay off when compressible structure exists and many queries or updates are needed. What we have done is formalize the knowledge-compilation paradigm for ranking-based defeasible reasoning and System Z, allowing for principled selection between these complementary techniques.

	Total preprocessing (µs)		Inference (µs)					Update (µs)		
N	SAT	ROBDD	ROBDD	SAT	SAT/ROBDD	ADD	SAT/ADD	ROBDD	SAT	SAT/ROBDD
6	290.0	928.1	2.0	65.9	33.7	3.5	18.7	3.4	222.8	66.4
8	463.6	1108.9	2.9	88.3	30.4	5.4	16.4	5.6	368.3	65.6
10	669.1	1283.6	3.6	107.4	29.6	7.3	14.8	8.0	507.7	63.8
12	851.3	1429.8	4.7	122.7	26.2	9.4	13.1	11.3	664.2	58.8
14	1077.9	1691.6	5.8	136.2	23.4	11.5	11.8	16.7	823.9	49.3
16	1311.5	2071.5	8.0	147.1	18.5	15.1	9.7	26.8	1007.6	37.6
18	1569.5	2645.4	10.3	160.6	15.7	20.9	7.7	39.1	1161.3	29.7
20	1795.2	3069.2	12.1	170.0	14.0	22.8	7.5	54.7	1392.7	25.4
30	3551.2	16 834.9	87.2	233.3	2.7	111.6	2.1	600.3	2714.8	4.5
40	5734.6	198 917.1	701.4	291.6	0.4	775.3	0.4	6021.9	4569.9	0.8
M1	797.1	873.3	5.3	89.4	17.0	10.6	8.4	24.6	743.2	30.2
M2	1478.3	4454.0	21.3	90.1	4.2	14.2	6.4	288.3	1416.5	4.9

Table 1: Results.

6 Related Work

Existing Z/RC implementations are SAT-based (Beierle, Spang, and Haldimann 2024), and provide no dedicated incremental update algorithm. A number of alternative OCF-based semantics exist (e.g., lexicographic closure (Lehmann 1995), c-represenations (Kern-Isberner 2002)), some of which take the Z-ranking as baseline and refine it. Our ADD encoding applies to these as well, however additional work is required to formalize their encoding process. We note that non-canonical Sentential Decision Diagrams are also applicable to our algorithms with the same polytime guarantees (Darwiche 2011). However, without canonicity one loses pointer-identity equivalence, and caching/memorization, weakening incremental reuse and other implementation properties that our engineering relies on. With canonical SDDs we lose polytime apply (Broeck and Darwiche 2015).

Our work is also related to recent approaches to compact epistemic-state representations for iterated belief change. Liberatore studies succinct representations and algorithms for iterated and mixed revisions (Liberatore 2024; Liberatore 2023), and Schwind et al. investigate computational aspects of iterated belief change over compactly represented total preorders (Schwind et al. 2020). Our work differs in that it takes a knowledge-compilation perspective, aiming to obtain compact representations in the tractability sense: once compiled, reasoning tasks are performed efficiently in the size of the compiled representation.

To our knowledge, this is the first work to provide an end-to-end decision-diagram pipeline for System Z/Rational Closure with (i) polytime general conditional entailment, (ii) a concrete incremental update algorithm, and (iii) an ADD-based OCF representation supporting polynomial-time Spohn conditioning.

6.1 Future Work

Several directions follow naturally from our work. A first direction is semantic generality. System Z and rational closure are only part of the broader ranking-based landscape (e.g., lexicographic closure and other refinements). Extending the compilation pipeline requires (i) identifying the computational objects these semantics induce, (ii) encoding them into suitable compiled representations, and (iii) characterising which operations remain efficient in compiled size.

A second direction is to expand the library of ADD-level operators for OCFs. Conditioning is one important update, but ranking functions support a wider family of belief-change operations. A principled operator library would make compiled OCFs closer in spirit to probabilistic circuits, where compilation enables many queries and transformations within a shared infrastructure.

A third direction is structure-aware heuristics. Variable ordering remains the dominant practical bottleneck. Ranking-based knowledge bases exhibit structure that could guide ordering and decomposition (e.g., layered Z-partitions, shared antecedents/consequents, or domain constraints). Exploiting these patterns could expand the range of instances where compilation is effective.

7 Conclusion

The enduring challenge for any rational agent is not merely to form plausible beliefs, but to gracefully adapt them in the face of new evidence. This work has forged a direct path from the abstract principles of ranking-based semantics to a concrete, computational artifact. Where System Z points the way to a rational belief state, our approach provides the means to inhabit, maintain, and optimize it.

AI Declaration

The authors used large language models to assist with writing and proof suggestions. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the accuracy, originality, and integrity of the final manuscript.

Acknowledgments

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

References

Aloul, F. A.; Markov, I. L.; and Sakallah, K. A. 2003. FORCE: a fast and easy-to-implement variable-ordering heuristic. GLSVLSI '03, 116–119. New York, NY, USA: Association for Computing Machinery.

Bahar, R.; Frohm, E.; Gaona, C.; Hachtel, G.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, 188–191.

Beierle, C.; Haldimann, J.; and Schwarzer, L. 2024. CLKR: Conditional Logic and Knowledge Representation. *KI - Künstliche Intelligenz* 38(1–2):61–67.

Beierle, C.; Spang, A.; and Haldimann, J. 2024. Using SAT and Partial MaxSAT for Reasoning with System Z and System W.

Biere, A.; Fazekas, K.; Fleury, M.; and Heisinger, M. 2020. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Balyo, T.; Froleyks, N.; Heule, M.; Iser, M.; Järvisalo, M.; and Suda, M., eds., *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, 51–53. University of Helsinki.

Bollig, B., and Wegener, I. 1996. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers* 45(9):993–1002.

Bollig, B.; Löbbing, M.; and Wegener, I. 1996. On the effect of local changes in the variable ordering of ordered decision diagrams. *Information Processing Letters* 59(5):233–239.

Brace, K.; Rudell, R.; and Bryant, R. 1990. Efficient implementation of a BDD package. In *27th ACM/IEEE Design Automation Conference*, 40–45.

Broeck, G., and Darwiche, A. 2015. On the Role of Canonicity in Knowledge Compilation. *Proceedings of the AAAI Conference on Artificial Intelligence* 29.

Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers* C-35(8):677–691.

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

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Int. Res.* 17(1):229–264.

Darwiche, A. 2011. SDD: a new canonical representation of propositional knowledge bases. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, 819–826. AAAI Press.

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

Drechsler, R.; Becker, B.; and Göckel, N. 1996. Genetic algorithm for variable ordering of OBDDs. *Computers and Digital Techniques, IEE Proceedings* - 143:364 – 368.

Drechsler, R.; Drechsler, N.; and Günther, W. 1998. Fast exact minimization of BDDs. In *Proceedings of the 35th Annual Design Automation Conference*, DAC '98, 200–205. New York, NY, USA: Association for Computing Machinery.

Eiter, T., and Lukasiewicz, T. 2000. Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artificial Intelligence* 124(2):169–241.

Freund, M. 1998. Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence* 98(1):209–235.

Fujita, M.; McGeer, P. C.; and Yang, J. C.-Y. 1997. Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation. *Formal Methods in System Design* 10(2):149–169.

Goldszmidt, M., and Pearl, J. 1990. On the Relation Between Rational Closure and System-Z. In *Proceedings of the Third International Workshop on Nonmonotonic Reasoning, May 31 – June 3*, 130–140.

Kern-Isberner, G. 2002. Handling conditionals adequately in uncertain reasoning and belief revision. *Journal of Applied Non-Classical Logics* 12:215–237.

Lehmann, D., and Magidor, M. 1992. What does a conditional knowledge base entail? *Artificial Intelligence* 55(1):1–60.

Lehmann, D. 1995. Another perspective on default reasoning. *Annals of mathematics and artificial intelligence* 15:61–82.

Liberatore, P. 2023. Mixed Iterated Revisions: Rationale, Algorithms, and Complexity. *ACM Trans. Comput. Logic* 24(3).

Liberatore, P. 2024. Representing states in iterated belief revision. *Artif. Intell.* 336(C).

Pearl, J. 1990. 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, 121–135. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rudell, R. 1993. Dynamic variable ordering for ordered binary decision diagrams. In *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, 42–47.

Schwind, N.; Konieczny, S.; Lagniez, J.-M.; and Marquis, P. 2020. On Computational Aspects of Iterated Belief Change. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 1770–1776. International Joint Conferences on Artificial Intelligence Organization. Main track.

Shannon, C. E. 1949. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal* 28(1):59–98.

Somenzi, F. 1997. Cudd : Cu decision diagram package. *Public Software, University of Colorado*.

Somenzi, F. 2015. *CUDD: CU Decision Diagram Package, Release 3.0.0*. Dept. of Electrical, Computer, and Energy Engineering, University of Colorado Boulder, Boulder, CO, USA.

Spohn, W. 1988. Ordinal Conditional Functions. A Dynamic Theory of Epistemic States. In Harper, W. L., and Skyrms, B., eds., *Causation in Decision, Belief Change, and Statistics, vol. II*. Kluwer Academic Publishers.

Wegener, I. 1994. The size of reduced OBDD's and optimal read-once branching programs for almost all Boolean functions. *IEEE Transactions on Computers* 43(11):1262–1269.