An Override-Aware Classifier for Transparent AI

Ruvarashe Shalom Madzime[1][0009−0001−3469−9353], Louise Leenen[1][0000−0002−9212−550X], and Thomas Meyer[2][0000−0003−2204−6969]

[1] University of the Western Cape and CAIR, Cape Town, South Africa
4158028@myuwc.ac.za, lleenen@uwc.ac.za
[2] University of Cape Town and CAIR, Cape Town, South Africa
tmeyer@cair.org.za

Abstract. Many real-world decisions follow rules that hold in general but allow exceptions, such as "birds usually fly, unless they are penguins." Most interpretable classifiers struggle to capture this pattern, leading to explanations that feel less aligned with human reasoning. This paper introduces the Defeasible Horn Classifier with Exceptions (DHCE), a symbolic model that makes this reasoning structure explicit. Each rule combines a default with its linked exceptions, so predictions can be explained step by step without relying on post-hoc tools. DHCE is learned using Answer Set Programming, which searches for globally optimal rule sets while balancing accuracy and simplicity. The resulting models consist of ranked Horn rules that provide full traceability: users can see both why a decision applies and why it may be overridden. We evaluate DHCE on standard classification benchmarks and find that it matches or outperforms leading interpretable models, a performance level that prior work shows to be competitive with classical machine learning classifiers. By making prediction decisions inherently retractable, DHCE delivers accuracy alongside explanations that mirror how people reason, making it suited for domains where understanding why a rule no longer applies is as important as the prediction itself.

Keywords: Explainable AI (XAI) · Human-aligned reasoning · Rule-based Models · Interpretable Classification · Horn rules · Transparency · Answer Set Programming

1 Introduction

Machine-learning (ML) models are increasingly being used in various high-stakes domains [1]. While these models often perform well in terms of predictive accuracy, this alone is not sufficient for real-world acceptance. It would be beneficial if these models were also interpretable, transparent, and explainable.

Explainable AI (XAI) frameworks define these qualities distinctly. A model is transparent when its internal decision-making logic is fully accessible. It is interpretable when users can follow the path from input to output and make sense of how predictions were reached. It is explainable if it can generate human-understandable justifications for its predictions, even when its internal workings

are complex or opaque [2]. Doshi-Velez and Kim [2] proposed that interpretability means being able to work through a model's logic step by step. They highlighted simulatability (being simple enough to mentally simulate), modularity (separate parts handling specific tasks), and decomposability (where each part can be understood on its own). Rudin argued that in sensitive areas, such as healthcare or criminal justice, models should be designed to be interpretable from the start, not explained afterward with separate tools [3]. These views represent two main approaches: explanations added after training (post-hoc) and models built to be understandable from the outset (built-in interpretability).

Post-hoc methods such as LIME [4] and SHAP [5] attempt to explain the output of a trained model. They work by perturbing inputs and approximating the model's behaviour around specific data points [3]. However, Guidotti et al. [6] note that such tools only approximate black-box behaviour. When the model is highly complex, these approximations become unreliable, hiding key parts of the actual decision process [6]. This reinforces the value of interpretable models, where the internal logic is transparent and structured in a way that can be understood without external tools.

Interpretable models are often called white-box models because their reasoning is visible. One family is decision lists: sequences of if–then rules applied top-down, where the first matching rule determines the outcome. Variants include Falling Rule Lists, which prioritize rules with higher confidence [7], and CORELS, which uses exact search to find optimal short lists [8]. A related family is decision sets, where multiple rules may apply at once, each casting a vote so that the final decision is based on an aggregate [9]. Classical rule learners such as RIPPER [10] and FOIL [11] build lists or sets greedily to reduce error. ILASP [12] provides a logic-based alternative, using Answer Set Programming (ASP) to define and search for models with user-specified structure. ASP is a declarative form of logic programming: the user states rules a solution must satisfy, and the solver automatically explores all "answer sets" that meet them [13].

Both decision lists and decision sets are explicit, but they struggle to capture how humans often think: through defaults and exceptions. A classic example is "birds normally fly, unless they are penguins." In decision lists this requires placing the exception above the default as a separate rule, breaking their natural connection.

Another class of interpretable models learns a single Boolean expression to capture the full decision process. The Short Boolean Formula (SBF) approach [14] uses ASP to find the smallest expression (using $\neg$, $\wedge$, and $\vee$) that correctly classifies the data. These formulas are compact and competitive with standard classifiers such as Naïve Bayes [15]. A related approach learns tractable logical circuits, such as Ordered Binary Decision Diagrams (OBDDs) or Sentential Decision Diagrams (SDDs), which are equivalent to the classifier and support exact reasoning [16, 17].

Yet Boolean formulas and circuits often bury negations and overrides deep within the structure, without an explicit link between defaults and exceptions. They lack a clear notion of rule priority, which makes decisions harder to interpret.

In summary, post-hoc methods provide approximations of black-box models but reveal little about their true logic. Rule-based systems are transparent but struggle with exceptions. Boolean formulas and circuits achieve conciseness, yet hide reasoning in cases of conflict. None of these approaches support a natural "normally... unless..." narrative, where defaults and exceptions are directly connected. This gap limits interpretability in domains where understanding why a decision was made is as important as the outcome.

Human reasoning, by contrast, often follows a pattern of defaults and exceptions: accept what is normally true and withdraw that conclusion when an exception appears ("birds fly, unless they are penguins"). Formally, this is called defeasible reasoning. Kraus, Lehmann, and Magidor (KLM) [18] gave a formal statement of this idea: a defeasible rule says "if A then B" in ordinary cases, but allows later information to block the inference. Because new facts can overturn prior conclusions, these logics are non-monotonic.

Classic frameworks capture this intuition. Reiter's Default Logic [19] starts from facts, tentatively adds defaults as long as they remain consistent, and produces belief sets called extensions. Nute's Defeasible Logic [20] separates rules into strict (always valid) and defeasible (normally valid), resolving conflicts through explicit priorities. McCarthy's Circumscription [21] models normality by minimising "abnormal" predicates so that everything is assumed normal unless stated otherwise.

This paper introduces the Defeasible Horn Classifier with Exceptions (DHCE), an interpretable model that applies this "normally... unless..." pattern to classification tasks. DHCE is a learned rule model that captures defaults and exceptions without committing to a specific non-monotonic proof theory. Each rule contains both the default and its exception, producing a concise narrative: apply the first matching default, then check its attached exceptions. This yields step-by-step explanations that ordinary readers can follow.

Unlike rule lists that scatter exceptions or Boolean formulas that hide them in nested logic, DHCE keeps each default paired with its exception, ensuring clarity in every decision trace. On standard binary datasets, DHCE outperformed other interpretable models and reached accuracy levels comparable to classical ML methods, while still producing complete explanations for every prediction.

This paper makes three contributions: (1) a new defeasible model class with explicit exception handling, (2) an ASP encoding that finds globally optimal rule sets, and (3) empirical evidence that DHCE achieves competitive accuracy among interpretable models. Section 2 gives a motivating example, Section 3 formally defines the framework, Section 4 details the ASP encoding, Section 5 presents results, Section 6 provides a case study, and Section 7 concludes with discussion.

2 Motivating Example and Problem Statement

Consider a foundation that awards scholarships to students who either show academic merit, indicated by a high GPA (at least 3.7) and extensive community

service (at least 100 hours), or demonstrate financial need, reflected by low household income and first-generation college status. However, both paths to qualification are blocked if the student has a record of serious disciplinary misconduct or a dishonesty flag such as plagiarism. Let $Q = 1$ denote qualification, and $Q = 0$ rejection. The binary features recorded for each student are: G for high GPA, S for community service, I for low income, F for first-generation status, D for disciplinary record, and P for plagiarism. A value of 1 indicates the feature is True.

An interpretable classifier of this policy can be written as follows:

$$Q \leftarrow G \wedge S \qquad \text{(merit default)} \tag{1}$$
$$\neg Q \leftarrow G \wedge S \wedge D \qquad \text{(disciplinary exception)} \tag{2}$$
$$Q \leftarrow I \wedge F \qquad \text{(financial-need default)} \tag{3}$$
$$\neg Q \leftarrow I \wedge F \wedge P \qquad \text{(plagiarism exception)} \tag{4}$$

Each rule has the form $\mathit{Head} \leftarrow \mathit{Body}_1 \wedge \cdots \wedge \mathit{Body}_k$. The symbol "$\leftarrow$" denotes implication: if all body literals are true, the head literal must also be true. The connective "$\wedge$" denotes logical conjunction, and "$\neg$" is classical negation. Thus Rule 1 says that a student with a high GPA (G) AND extensive service (S) qualifies (Q); Rule 2 overrides that by implying rejection when serious misconduct (D) is also present; Rule 3 grants qualification for low income (I) together with first-generation status (F); and Rule 4 blocks that route when plagiarism (P) is detected.

Suppose an applicant, *Emily*, has the following feature values: $\mathrm{G} = 1,\ \mathrm{S} = 1,\ \mathrm{D} = 1,\ \mathrm{I} = 0,\ \mathrm{F} = 0,\ \mathrm{P} = 0$. She meets the merit criteria (1), but is overruled by the disciplinary exception (2), and is therefore rejected.

One symbolic option is to learn a single Boolean formula (SBF) built only from AND ($\wedge$), OR ($\vee$) and NOT ($\neg$) operators that map the features to a label [14]. Here disciplinary misconduct D blocks only the merit route (Rules 1 and 2), and plagiarism P blocks only the financial-need route (Rules 3 and 4). Accordingly, an equivalent SBF encodes each default with its own blocker:

$$(G \wedge S \wedge \neg D) \ \vee \ (I \wedge F \wedge \neg P) \tag{5}$$

Substituting Emily's values into (5) yields $(G \wedge S \wedge \neg D) = (1 \wedge 1 \wedge 0) = 0$ and $(I \wedge F \wedge \neg P) = (0 \wedge 0 \wedge 1) = 0$, so $Q = 0 \vee 0 = 0$. This classifies Emily correctly as $Q = 0$, but the reason is hidden: there is no explicit default, exception label, or clear trace of the override. Standard black box ML classifiers such as neural networks may also output $Q = 0$, but provide no explanation.

This scenario calls for a model that reflects how people naturally reason: "Normally merit qualifies, unless there's misconduct." DHCE captures this structure directly. For Emily, a DHCE model proceeds as follows:

1. It applies the merit default ($G \wedge S$), setting $Q = 1$.
2. It then finds the exception (D) holds, overriding the default.
3. It returns $Q = 0$ with a transparent explanation.

The result is a clear trace: "Merit default matched; disciplinary exception applied; outcome flipped to $Q = 0$." Unlike rule lists that separate exceptions, or single formulas that could hide them, DHCE provides a clear default-and-exception narrative. Section 3 formalises this framework.

3 Formal Framework for DHCE

This section looks at the structure of DHCE, an interpretable classifier that combines default rules with attached exceptions. It works on datasets where every instance is described by binary input features (0 or 1) and a binary label. The goal is to learn a rule set that mirrors how people reason about what *normally* holds and when an *exception* overrides it.

An atom is a basic, single-feature statement, for example GPA_high. A positive literal is simply an atom, a negative literal is the same atom preceded by classical negation, such as $\neg GPA\_high$. In the datasets used to experiment on DHCE, each binary feature becomes an atom. If the feature's value is 1 we write the positive literal $\mathsf{GPA\_high}$; if the value is 0 we write the negative literal $\neg GPA\_high$.

A Horn rule is a clause that takes the shape $t \leftarrow B_1, B_2, \ldots, B_k$ where the head t is a single positive atom (the conclusion) and each body literal B_i is either a positive or a negative literal (a condition). Horn rules avoid nested AND/OR/NOT structures, so they read naturally as "*if all these conditions hold, then conclude t*". Strictly restricting every rule within DHCE to be in Horn rule format has three practical benefits:

1. It is traceable: each prediction comes from one rule, optionally overridden by its local exception.
2. Placing the exception right next to its default makes the "normally ... unless ..." reasoning explicit.
3. Their very simple syntax keeps the search space small for the ASP encoding used by DHCE (see Section 4).

Returning to the scholarship case in Section 2, DHCE would express the logic as:

$$t \leftarrow \mathsf{GPA\_high}, \mathsf{Service} \quad \text{(default)} \tag{6}$$

$$\neg t \leftarrow \mathsf{GPA\_high}, \mathsf{Service}, \mathsf{Disciplinary} \quad \text{(exception)} \tag{7}$$

This clearly shows: a student with strong merit normally qualifies, unless disqualified by misconduct. The logic is transparent, the override is explicit, and the explanation mirrors how a human reasons.

Before the DHCE system can learn rules from the dataset, the dataset has to be converted into propositional form. In this form all features describing each example must be expressed using basic true/false values. To do this, every column in the dataset is turned into a set of binary features. For numeric values like GPA or age, this is done by using the median as a cut-off point and then creating a

new feature that is true if the original value is above that threshold and false otherwise. We adopt median thresholding for numeric features to stay aligned with the SBF baseline, which Booleanises its datasets in the same way; this preserves an apples-to-apples comparison while keeping the representation simple and label-agnostic. For example, a feature GPA_high might be true if the GPA is above 3.7. For categorical features, a technique called one-hot encoding is used, which means that each possible category gets its own feature set to 1 if it applies, and 0 otherwise. After this step the table contains only 0s and 1s.

A DHCE model works by learning a ranked list of logical Horn rules that describe patterns within the data. The simplest kind of rule is a default rule, which describes what normally holds true (see Table 1). This means that $t = 1$ (the positive label) is predicted whenever all the conditions B_1 through B_k hold for a given instance.

However, there are often exceptions to such general patterns. An exception rule is used to override a default when more specific information is available (Table 1). This says that even if the default conditions B_1 through B_k hold, the conclusion $t = 1$ is cancelled if extra conditions E_1 through E_m also hold. In that case, the prediction flips to $t = 0$. Multiple exceptions per default are allowed in our encoding. The number is bounded by the constant (`maxE`), which we set during the model search. Each exception is evaluated only when its parent default matches, since exceptions are attached to that default. We do not allow exceptions to exceptions in the current design. This choice keeps every override local to its default and preserves a short, readable trace where each default appears next to its own exception.

After evaluating all defaults and their attached exceptions, the DHCE model introduces two final rule layers to guarantee full coverage of the dataset. First comes a fallback rule, a single-literal Horn rule added when no previous default or exception applies (Table 1). Here, F is the single literal that best classifies the remaining uncovered training examples. The selection is made by scoring each available literal against the uncovered cases and choosing the one that minimises misclassification. Like defaults, the fallback rule may include exceptions that cancel its effect. If even the fallback does not match, the catch-all rule fires. The catch-all has an empty body (Table 1); it always fires and predicts the majority class in the training data. For example, if most training examples have label $t = 1$, then the catch-all predicts $t = 1$ for any unmatched instance. The two layers are kept distinct by design to preserve traceability and clear tie-breaking.

Therefore, each instance can be traced back to a specific rule layer: default, exception, fallback, or catch-all, providing users with a transparent trace that is not available in black-box models.

DHCE classifies each instance by applying its rules in a clear, step-by-step order, using built-in priority handling to decide which rule should apply first. The model is defeasible by nature: conclusions can be withdrawn if stronger evidence appears. Rules are assigned numeric priorities to control conflict resolution. Learned default rules occupy the top levels, with ascending values (e.g., 0, 10, 20, ...), where lower numbers indicate higher priority. Each exception inherits its

Table 1. Formulas used in DHCE rules

Rule Type	Formula
Default	$t \leftarrow B_1, B_2, \ldots, B_k$
Exception	$\neg t \leftarrow B_1, B_2, \ldots, B_k, E_1, \ldots, E_m$
Fallback	$t \leftarrow F$
Catch-all	$t \leftarrow$

parent default's priority, ensuring they are evaluated together. The fallback rule and its possible exception are placed just below all defaults, and the catch-all rule sits at the very bottom. When an input is received, DHCE evaluates all applicable rules and selects the one with the lowest priority number. If multiple rules share this priority, the system chooses the rule with the most conditions in its body, favoring specificity.

This setup means that exceptions only apply when their parent default matches, and they always win in a conflict because their body strictly contains the default's body. Keeping exceptions at the same priority as their parent default avoids extra complexity and keeps the reasoning path local and easy to follow. The evaluation unfolds in a fixed order: scan defaults, apply the first match, check for an exception, otherwise move to the fallback, and finally the catch-all. This process guarantees that each instance receives exactly one label. Because each prediction is backed by a rule with a known priority and body, users can always trace why one rule overruled another.

To learn DHCE models, ASP is used. ASP is a declarative framework for hard search problems. A program states rules and constraints that any solution must satisfy, and a solver searches for stable models that meet those conditions. ASP is widely used in knowledge representation and reasoning [13]. Clingo is the standard toolchain. It grounds the program to a propositional form and then calls clasp. Clasp is the solver that performs the search and optimisation [22].

In this work ASP generates candidate DHCE rule sets, checks structural constraints such as uniqueness and ordering, and optimises for accuracy and simplicity. The resulting models classify the training data while keeping explanations readable.

Concretely, clingo grounds the DHCE template and hands it to clasp, which searches for stable models under a lexicographic objective. The encoding proposes defaults up to `maxD`, and for each default up to `maxE` attached exceptions, with each body limited by `maxBody`. For each candidate rule set the program simulates classification on the training rows to count errors and records simple size measures. Optimisation proceeds in levels: minimise training error first, then prefer fewer defaults, fewer exceptions, and shorter bodies. During model selection a grid over `maxD`, `maxE`, and `maxBody` is scanned. The configuration with the lowest test error is chosen, breaking ties by total literal count.

Consider the following example. Suppose `maxD` $= 1$, `maxE` $= 2$, `maxBody` $= 2$. A candidate default $t \leftarrow a_1, a_2$ is proposed. Two exceptions can attach to this default, for example $\neg t \leftarrow a_1, a_2, e_1$ and $\neg t \leftarrow a_1, a_2, e_2$. The solver tests these

candidates on the training rows. If the default reduces error and some mistakes occur only when e_1 holds, the first exception is kept; if a distinct set of mistakes occurs with e_2, the second exception is also kept, provided both stay within the size bounds. If multiple candidates tie on error, the version with shorter bodies is preferred. A single literal fallback and an empty body catch-all are then added so that every instance receives a label. This illustrates that more than one exception may attach to the same default while the priority and tie breaking keep the override local and readable.

To evaluate how well DHCE performs, this study adopts the same experimental setup used by Jaakkola et al. [14], who introduced the SBF method. Like DHCE, their method uses ASP to learn interpretable models from binary (true/false) data. Both systems are evaluated using ten random 50/50 splits of each dataset, half the data for training, and half for testing. The goal is to compare the training and test error across different splits and see how DHCE performs relative to SBF.

The prediction error for a learned rule set $\mathcal{R}$ is the proportion of examples on which its prediction $\hat{y}\mathcal{R}(x)$ disagrees with the true label $y(x)$. We write S for the training set and T for the test set (with sizes $|S|$ and $|T|$), let x range over the instances in either set, and use the indicator $\mathbf{1}[\cdot]$ that returns 1 when its argument is true and 0 otherwise. With this notation the training error and test error rates are

$$\text{Errortrain}(\mathcal{R}) = \frac{1}{|S|} \sum_{x \in S} \mathbf{1}\big[\hat{y}\mathcal{R}(x) \neq y(x)\big], \tag{8}$$

$$\text{Errortest}(\mathcal{R}) = \frac{1}{|T|} \sum_{x \in T} \mathbf{1}\big[\hat{y}_{\mathcal{R}}(x) \neq y(x)\big]. \tag{9}$$

These error values are used to find the best rule set for each split. All results are stored inside the ASP generated answer sets for traceability.

This lexicographic optimisation ensures that the final model is not only accurate but also as simple and interpretable as possible. Because DHCE and SBF both use ASP, the same kind of data, and the same evaluation process, they are compatible for a comparison. It allows for directly assessing whether adding defeasible rule structure improves interpretability without affecting performance.

4 Implementation and Experimental Setup

SBF has been shown to compete with Naïve Bayes and decision trees [14]. DHCE seeks similar or better predictive accuracy while delivering more interpretable classifications. During evaluation, the system logs training error, test error, total rule set size, and per layer coverage, which are analysed in Section 5.

This section explains how DHCE was implemented, how it was evaluated, and why certain design decisions were made. All three datasets used in this study are publicly available from the UCI Machine Learning Repository [23], and they were the same datasets used in the SBF study [14]. The Breast Cancer Wisconsin (Original) dataset predicts whether a tumor is benign or malignant

based on microscopic cytology features. The Statlog (German Credit) dataset assesses whether a loan applicant is a good or bad credit risk, using financial and demographic attributes. The Ionosphere dataset classifies radar signal returns as "good" or "bad" for identifying ionospheric structures.

Each dataset was preprocessed to convert it into Boolean (true/false) form. As mentioned before, for numeric features, median thresholding was used and for categorical features, one-hot encoding was used. After this transformation, every data point is represented as a flat vector of 0s and 1s; concretely: Breast-Cancer has 683 rows with 10 Boolean attributes (cytology: benign/malignant), German-Credit has 1000 rows with 21 Boolean attributes (credit risk: good/bad), and Ionosphere has 351 rows with 35 Boolean attributes (radar signal: good/bad).

In the DHCE implementation, each example is encoded in ASP as a set of `val/3` atoms: `val(RowID, a(Attribute, Category?), 0/1).` This means that the feature 'a(Attribute, Category)' has value 0 or 1 for a given row (data point). The optional `Category` is only used when the feature comes from a one-hot encoded categorical variable, as in the German-Credit dataset. For example, `val(7, a(3,2), 1).` means that row 7 has category 2 active for attribute 3. In contrast, datasets like Breast-Cancer and Ionosphere use simpler encodings like `a(5)` for attribute 5.

Each dataset ends with a special column for the target label, this becomes the attribute to be predicted in DHCE. The exact target label positions are: `a(10)` for Breast Cancer, `a(21)` for German-Credit, and `a(35)` for Ionosphere.

To ensure a fair comparison with SBF, the evaluation protocol described by Jaakkola *et al.* [14] was followed. Each dataset was split into ten random 50/50 training–test partitions using fixed seeds.A driver script handles this protocol. It writes each split into grounded ASP facts, runs the solver with various parameter settings, and logs the results. For each split, the best model is selected based on test error. If several configurations tie on error, the system chooses the one with the fewest literals. This evaluation method aligns with SBF's evaluation method to support an apples-to-apples comparison.

The DHCE ASP encoding has three tasks: it constructs a layered Horn rule list (defaults with attached exceptions, a single literal fallback, and an empty body), it simulates those rules on the training rows to count errors, and it exposes error and size summaries that the optimiser can minimise. Concretely, the encoding proposes bodies for learned defaults up to `maxD`; for each default it allows up to `maxE` attached exceptions, and all bodies obey `maxBody`. The fallback slot is constrained to exactly one literal and the slot to an empty body. This produces a compact, ranked structure where overrides stay local to the default they qualify.

Misclassification is recorded whenever the predicted label differs from the ground truth for a training row. Optimisation is lexicographic: minimise training errors first; among ties prefer fewer defaults; then fewer exceptions; then the sum of default body lengths; then the sum of exception body lengths. The label index is dataset specific as described earlier in this section (a(10) for Breast Cancer, a(21) for German Credit, a(35) for Ionosphere). This summary keeps the learning

description self contained while making clear how DHCE differs from flat formula classifiers such as SBF.

To control the complexity of the models it learns, DHCE searches over a small grid of rule-size limits. This grid defines three aspects of model structure: the maximum number of default rules (`maxD`), the number of exceptions allowed per default (`maxE`), and the number of literals allowed in the body of any rule (`maxBody`). The solver explores all combinations of these values to generate a range of candidate rule sets. The grid used in this study includes $\texttt{maxD} \in \{3, 4\}$, $\texttt{maxE} \in \{1, 2\}$, and $\texttt{maxBody} \in \{1, 2, 3, 4\}$, which was found to offer a good trade-off between flexibility and runtime in pilot experiments.

For each train–test split, the solver evaluates every rule-size setting from the grid and selects the model with the lowest test error. If multiple models achieve the same accuracy, the one with the fewest total literals is preferred. This helps prevent overfitting by favouring simpler rule sets when possible.

Internally, the ASP encoding further enforces regularisation by disallowing redundant exceptions and ensuring that each rule covers at least one training instance not already handled by higher-priority rules. These structural constraints guide the solver toward non-redundant models.

SBF has already been shown to compete with traditional classifiers such as Naïve Bayes and decision trees [14]. DHCE aimed to achieve similar or better predictive accuracy while delivering more interpretable, classifications. During evaluation, the system logged key metrics, including training error, test error, total rule set size, and per-layer rule coverage. This was all saved and used for analysis in Section 5.

5 Results

The evaluation in this section builds directly on the protocol introduced in Section 3 and implemented as described in Section 4. As outlined earlier, each benchmark dataset was partitioned into ten fixed 50/50 train–test splits. For each split, DHCE searched over a grid of rule structures and selected the rule set that achieved the lowest test error, breaking ties using the smallest total number of literals. This protocol matches the setup used by Jaakkola et al. [14], enabling a direct comparison with SBF. The results presented next, summarise the characteristics of the best models discovered.

Figure 1 reports the lowest test error across all splits for each dataset. DHCE outperforms SBF on Breast-Cancer, reducing error by 1.5 percentage points. On German-Credit and Ionosphere, both models achieve identical error rates. These results confirm that the addition of defeasible rule layers does not degrade predictive accuracy, and in some cases can improve it.

When computing the DHCE rule sets, no strict timeout was imposed. Instead, the solver was allowed to run until it found an optimal rule set for each best split. This ensured that all models used in evaluation were globally optimal with respect to the specified objective. Figure 2 shows the total runtime per dataset. The German-Credit dataset required the longest solving time (7.3 hours), which

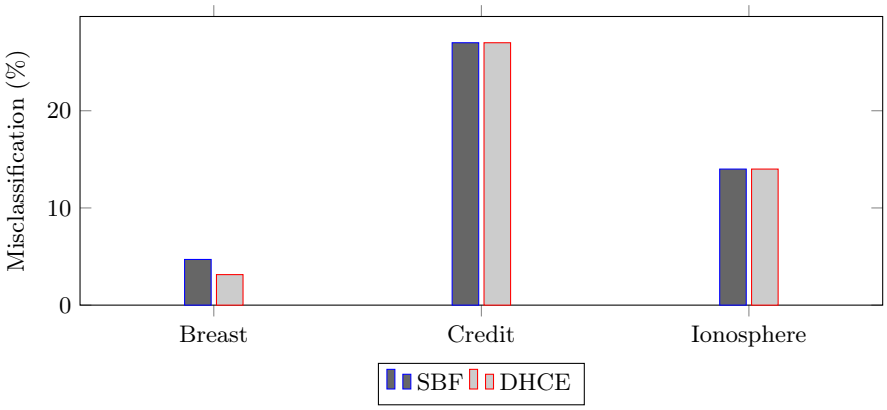

Fig. 1. Best-split misclassification rate.

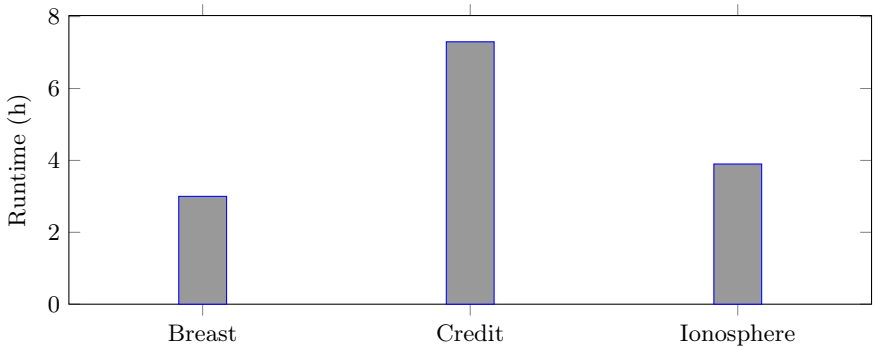

Fig. 2. Runtime optimal DHCE model (best split).

aligns with its size of 1 000 rows and 21 Boolean attributes. In contrast, the Breast-Cancer dataset (683 rows, 9 attributes) was solved in 3.0 hours, while Ionosphere (351 rows, 34 attributes) took 3.9 hours. These differences suggest that both dataset size and attribute dimensionality affect solving time. Across all datasets, the average runtime was approximately 4.7 hours. These times are reasonable for offline rule discovery, especially given the interpretability and structure DHCE provides in return.

The winning SBF rules are compact Boolean expressions, summarised per dataset in Table 2 [14]. These rules are notably short, 3 to 4 literals each. However, their simplicity comes at the cost of interpretability: the entire decision logic is compressed into a single, flat formula, encoding all reasoning inside nested logical operations. In contrast, Table 3 presents the explicit DHCE rule sets that match or exceed the accuracy of SBF. While DHCE rules use more literals overall: 7 for German-Credit and 9 for both Breast-Cancer and Ionosphere, the increase in size enables significantly clearer logic decomposition. Each rule is assigned

to a named layer (Default, Exception, Fallback, Catch-all), offering a faithful "normally ... unless ..." structure that mirrors human decision-making. These named layers allow users to see exactly when a rule applies and what specific exception (if any) overrides it.

Table 2. Best-performing SBF formulas per dataset.

Dataset	**SBF Formula**
Breast-Cancer (4 lit.)	$\neg((a[1] \wedge a[6]) \vee a[5]) \wedge a[3]$
German-Credit (3 lit.)	$\neg(a[1,1] \wedge a[2]) \vee a[17,4]$
Ionosphere (4 lit.)	$((a[8] \wedge a[12]) \vee a[15]) \wedge a[1]$

The SBF formulas in Table 2 are short, but they combine all logic into a single compact expression, which can be difficult to read. For example, the Breast-Cancer rule uses nested operations like $\neg((a[1] \wedge a[6]) \vee a[5]) \wedge a[3]$, which the user must unpack for understanding. In contrast, the DHCE rule sets in Table 3 break down the logic into separate layers: defaults, exceptions, fallback, and catch-all. This separation makes it easier to see what normally holds, when it is overridden, and how each case is handled. Although DHCE uses more literals overall, this extra detail improves clarity and helps explain each prediction step by step.

Table 3. DHCE best-split rule sets across datasets

Dataset	**Rule Type**	**Rule**	**Exception(s)**
Cancer (9 literals)	Default 1	$1 \leftarrow a(2,0)$	$0 \leftarrow a(2,0) \wedge a(8,1)$
	Default 2	$1 \leftarrow a(7,0)$	$0 \leftarrow a(7,0) \wedge a(8,1)$
	Fallback	$1 \leftarrow a(1,1)$	$0 \leftarrow a(1,1) \wedge a(3,1)$
	Catch-all	majority label 1	–
Credit (7 literals)	Default 1	$1 \leftarrow a(16,4) = 1$	$0 \leftarrow a(16,4) = 1 \wedge a(4,10) = 1$ $0 \leftarrow a(16,4) = 1 \wedge a(4,8) = 1$
	Fallback	$1 \leftarrow a(16,4) = 0$	$0 \leftarrow a(16,4) = 0 \wedge a(3,1) = 1$
	Catch-all	majority class 1	–
Ionosphere (9 literals)	Default 1	$1 \leftarrow a(1,0)$	$0 \leftarrow a(1,0) \wedge a(15,1)$
	Default 2	$1 \leftarrow a(12,0)$	$0 \leftarrow a(12,0) \wedge a(33,0)$
	Fallback	$1 \leftarrow a(1,1)$	$0 \leftarrow a(1,1) \wedge a(8,0)$
	Catch-all	majority label 1	–

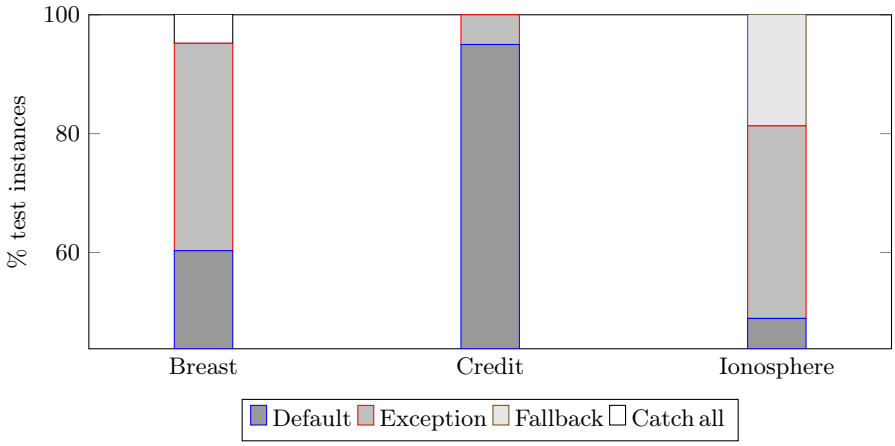

Fig. 3. Layer coverage of the *best* DHCE rule set on each dataset.

Figure 3 shows for the Breast-Cancer dataset how DHCE explains predictions using Default rules 60.3% of the time and Exception rules 34.9%, with only 4.9% handled by the Catch-all. In Ionosphere, 48.9% of test instances are covered by Defaults, 32.4% by Exceptions, and 18.8% by the Fallback rule—again, no reliance on Catch-all. On German-Credit, a dominant Default rule covers 95.0% of predictions, with only 5.0% requiring an Exception. DHCE uniquely provides this layered breakdown, providing transparency for the user.

DHCE shows strong overall performance across all benchmarks. In terms of accuracy, it matches or exceeds SBF performance, notably achieving a 1.54 percentage-point improvement on the Breast-Cancer dataset in Figure 1. More importantly, DHCE offers an advantage in interpretability: its rules are structured explicitly using "normally ... unless ..." clauses (see Table 3), making the reasoning process transparent and easy to follow, unlike Boolean expressions used by SBF. From a classifier-complexity perspective, DHCE maintains typically 7–9 literals while remaining practical in terms of runtime. As shown in Figure 2, model training takes between 3 and 7.3 hours, making the method feasible for applications where interpretability is a priority.

6 A Case Study

This section shows how DHCE mirrors human reasoning more closely than a single Boolean formula, even when both models reach the correct prediction. For this example, we consider a Breast-Cancer patient whose features activate both a default and an attached exception in the DHCE rule set.

To give context, Table 4 lists the ten columns that appear in the Breast-Cancer dataset. The first nine are input attributes extracted from microscopic images and then binarised at the median: a value of 0 means the measurement falls in the lower half of the training data, and 1 means it lies in the upper half [23]. The

last column, $a[10]$, is the *target label* supplied by the pathologist, in every model used here a label of 0 is interpreted as *benign* and 1 as *malignant*.

Using the feature definitions given in Table 4, consider a patient with binarised cytology profile $(\mathrm{CT}, \mathrm{UCS}, \mathrm{UCSh}, \mathrm{MA}, \mathrm{SECS}, \mathrm{BN}, \mathrm{BC}, \mathrm{NN}, \mathrm{MIT}) = (0, 0, 0, 0, 0, 0, 0, 1, 0)$. This individual has $a[2] = 0$ (cells are uniformly sized). However, prominent nucleoli are also present ($a[8] = 1$), a known override feature. These conditions trigger both a default and its linked exception in the DHCE rule set (see Table 3). Default 1 fires (see Table 3): $1 \leftarrow a[2]{=}0$ "Having uniform cell size normally leads to a prediction of malignant." Exception overrides: $0 \leftarrow a[2]{=}0 \wedge a[8]{=}1$ "Having uniform cell size normally leads to a malignant prediction, unless prominent nucleoli are also present, in which case the prediction is benign." Final DHCE prediction: benign (0). SBF also predicts class 0, using its learned 4-literal formula: $\neg((a[1] \wedge a[6]) \vee a[5]) \wedge a[3]$. Substituting these values yields: $\neg((0 \wedge 0) \vee 0) \wedge 0 = 0$, which matches DHCE's output. However, the SBF formula does not show clearly how this decision was made. It is difficult to tell which features led to the result, and there is no indication that one rule may have been overridden by another. All of the reasoning is packed into a single expression, which hides the steps that were followed. In contrast, DHCE shows its reasoning step by step. It clearly indicates that a default rule first predicted a malignant tumour, but this was then changed because an exception applied. This kind of explanation is easier to follow and matches how people usually reason.

This example also reflects what happens more broadly. On the full Breast-Cancer test set, DHCE made fewer errors than SBF,improving accuracy by 1.5 percentage points (Figure 1). It also gave a clear rule-layer explanation for all test cases, showing whether the result came from a default, an exception, or the catch-all rule (see Table 3). Even when both models give the same answer, DHCE gives a better decision trace by showing both what normally happens and when that rule no longer applies.

Table 4. Binarised Breast-Cancer columns ($a[1]-a[9]$ indicate 0 = below median, 1 = above; $a[10]$ is the diagnosis.)

Index	**Description**	**0 = Lower half**	**1 = Upper half**
$a[1]$	Clump Thickness (CT)	Thin clumps	Dense clumps
$a[2]$	Uniformity Cell Size (UCS)	Uniform cells	Size variation
$a[3]$	Uniformity Cell Shape (UCSh)	Regular shapes	Irregular shapes
$a[4]$	Marginal Adhesion (MA)	Weak adhesion	Strong bonding
$a[5]$	Single Epithelial Cell Size	Small cells	Large cells
$a[6]$	Bare Nuclei (BN)	Few nuclei	Many nuclei
$a[7]$	Bland Chromatin (BC)	Fine texture	Coarse texture
$a[8]$	Normal Nucleoli (NN)	Not visible	Prominent
$a[9]$	Mitoses (MIT)	Low activity	Elevated activity
$a[10]$	**Diagnosis (label)**	**Benign**	**Malignant**

7 Discussion and Conclusion

The experimental results confirm that DHCE achieves competitive accuracy while preserving its transparent *"normally ... unless ..."* structure. On the Breast-Cancer dataset [23] it reduced error from 4.7% to 3.14%, and on German-Credit and Ionosphere [23] it matched the 27.0% and 14.0% baselines, showing that attaching an explicit exception to every default does not compromise performance (See Figure 1). DHCE demonstrates how a model built from defaults and exceptions can tell the user both *why* a default fired and *when* it was overridden.

Overfitting is controlled by keeping small bounds on maxD, maxE, and maxBody. When accuracy ties, selection prefers fewer rules and shorter bodies, which acts as regularisation. The encoding prevents redundancy by requiring each new default to cover at least one previously uncovered instance and by disallowing duplicate exception bodies. Finally, models are chosen by the lowest test error over repeated splits, so large train-test gaps are penalised.

Several related studies also fuse explanations with defeasible reasoning. Brewka and Ulbricht's work on *strong explanations* shows how to identify the minimal facts that guarantee a conclusion despite exceptions [24]. Everett *et al.* [25] extend propositional KLM logic [18] with an explanation layer that highlights the rules ensuring a conclusion in the most normal situation. Chama *et al.* [26] adapt this idea to description logic, ranking worlds by normality and returning the smallest set of rules and exceptions that preserve a conclusion in the top-ranked worlds. Rienstra [27] applies similar ideas to Boolean classifiers with background knowledge, refining Darwiche and Hirth's monotonic "sufficient reasons" [28] into "ideal reasons" that are concise, knowledge-aware, and free of redundancy.

These approaches differ in technique but share a single goal: tell users not only *what* the system decided but also *why* it stands and *when* it would change. DHCE pursues this same spirit from a learning perspective. Instead of adding an explanation layer, DHCE learns rules that already combine defaults with their local exceptions, so every prediction comes with a trace. In DHCE, the structure itself—defaults, exceptions, and their priorities—forms the explanation.

At present, DHCE handles only binary features and labels. A natural next step is to extend the default–exception scheme to multi-class problems and numeric or ordinal attributes. Training currently relies on Clingo, which guarantees optimal rule sets but scales poorly as feature counts grow. Future research into incremental grounding, heuristic pruning, or anytime search could improve runtime for larger datasets. Another direction is to compare DHCE directly with post-hoc tools like LIME or SHAP [4], clarifying its practical advantages. Finally, embedding DHCE in an explicit non-monotonic framework could provide formal guarantees that further support its use in real applications.

Acknowledgements

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

References

[1] Andreas Fuster, Paul Goldsmith-Pinkham, Tarun Ramadorai, and Ansgar Walther. "Predictably Unequal? The Effects of Machine Learning on Credit Markets". In: *The Journal of Finance* 77.1 (2022), pp. 5–47. DOI: 10.1111/jofi.13090.

[2] Finale Doshi-Velez and Been Kim. *Towards a Rigorous Science of Interpretable Machine Learning*. arXiv preprint arXiv:1702.08608. 2017. DOI: 10.48550/arXiv.1702.08608.

[3] Cynthia Rudin. *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. arXiv preprint arXiv:1811.10154. 2019. DOI: 10.48550/arXiv.1811.10154.

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[5] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. arXiv preprint arXiv:1705.07874. 2017. DOI: 10.48550/arXiv.1705.07874.

[6] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. "A Survey of Methods for Explaining Black Box Models". In: *ACM Computing Surveys* 51.5 (2019). DOI: 10.1145/3236009.

[7] Fulton Wang and Cynthia Rudin. "Falling Rule Lists". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Vol. 38. Proceedings of Machine Learning Research. PMLR, 2015, pp. 1013–1022.

[8] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. "Learning Certifiably Optimal Rule Lists for Categorical Data". In: *Journal of Machine Learning Research* 18 (2018), pp. 1–78.

[9] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. "Interpretable Decision Sets: A Joint Framework for Description and Prediction". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1675–1684. DOI: 10.1145/2939672.2939874.

[10] William W. Cohen. "Fast Effective Rule Induction". In: *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1995, pp. 115–123.

[11] J. Ross Quinlan and R. Michael Cameron-Jones. "FOIL: A Midterm Report". In: *European Conference on Machine Learning*. Springer, 1993, pp. 3–20.

[12] Mark Law, Alessandra Russo, and Krysia Broda. *The ILASP System for Inductive Learning of Answer Set Programs*. arXiv preprint arXiv:2005.00904. 2020. DOI: 10.48550/arXiv.2005.00904.

[13] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. "Answer Set Programming at a Glance". In: *Communications of the ACM* 54.12 (2011), pp. 92–103. DOI: 10.1145/2043174.2043195.

[14] Reijo Jaakkola, Tomi Janhunen, Antti Kuusisto, Masood Feyzbakhsh Rankooh, and Miikka Vilander. "Short Boolean Formulas as Explanations in Practice". In: *Logics in Artificial Intelligence (JELIA 2023)*. Vol. 14281. Lecture Notes in Artificial Intelligence. 2023, pp. 90–105.

[15] Pedro Domingos and Michael Pazzani. "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss". In: *Machine Learning* 29.2–3 (1997), pp. 103–130. DOI: 10.1023/A:1007413511361.

[16] Jinbo Huang and Adnan Darwiche. "The Language of Search". In: *Journal of Artificial Intelligence Research* 29 (2007), pp. 191–219. DOI: 10.1613/jair.2097.

[17] Adnan Darwiche. "SDD: A New Canonical Representation of Propositional Knowledge Bases". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. 2011, pp. 819–826. DOI: 10.5591/978-1-57735-516-8/IJCAI11-143.

[18] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. "Nonmonotonic Reasoning, Preferential Models and Cumulative Logics". In: *Artificial Intelligence* 44.1–2 (1990), pp. 167–207.

[19] Raymond Reiter. "A Logic for Default Reasoning". In: *Artificial Intelligence* 13.1–2 (1980), pp. 81–132. DOI: 10.1016/0004-3702(80)90014-4.

[20] Donald Nute. "Defeasible Reasoning". In: *Handbook of Philosophical Logic*. Vol. 3. 1987, pp. 353–395.

[21] John McCarthy. "Circumscription—A Form of Non-Monotonic Reasoning". In: *Artificial Intelligence* 13.1–2 (1980), pp. 27–39.

[22] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[23] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2019.

[24] Gerhard Brewka and Markus Ulbricht. "Strong Explanations for Nonmonotonic Reasoning". In: *Description Logic, Theory Combination, and All That*. Springer, 2019, pp. 135–146. DOI: 10.1007/978-3-030-22102-7_6.

[25] Lloyd Everett, Emily Morris, and Thomas Meyer. "Explanation for KLM-Style Defeasible Reasoning". In: *Proceedings of the Southern African Conference for Artificial Intelligence Research*. 2021, pp. 192–207.

[26] Victoria Chama, Steve Wang, Thomas Meyer, and Giovanni Casini. "Defeasible Justification for KLM-Style Logic". In: *Proceedings of the 37th International Workshop on Description Logics*. 2024.

[27] Tjitze Rienstra. "Explaining Boolean Classifiers with Non-Monotonic Background Theories". In: *Artificial Intelligence and Machine Learning — 35th Benelux Conference, BNAIC/Benelearn 2023, Revised Selected Papers*. Vol. 2187. Communications in Computer and Information Science. Springer, 2025, pp. 174–188. DOI: 10.1007/978-3-031-74650-5_10.

[28] Adnan Darwiche and Pierre Marquis. "On the (Complete) Reasons Behind Decisions". In: *Proceedings of the 24th European Conference on Artificial Intelligence*. 2020, pp. 2407–2414. DOI: 10.3233/FAIA200375.