A BENCHMARK WITH KNOWN CAUSES: CAN LANGUAGE MODELS EXPLAIN ML AND RL DECISIONS?

Anonymous authors
Paper under double-blind review

ABSTRACT

The quality of an explanation for an ML model or RL policy in a given situation is typically judged by readability or by alignment with another explainer. A more insightful test is whether the explanation identifies the cause of the action that the policy proposed. For such a test, we need decisions for which the cause is known. We build a benchmark of fifty such decisions: fifteen from tabular classifiers, where each feature's contribution is exact, and thirty-five from three reinforcement learning agents (value iteration on a grid world, tabular Q-learning on cart-pole and an upper-confidence-bound contextual bandit), where we keep a case only if removing one candidate cause and re-running the agent changes the decision. Each case carries four probe situations answered by re-running the system, so the answer key is exact. A program scores faithfulness against the known cause with no language model involved, and a weaker reader from a different model family scores comprehension by predicting what the system does in situations it has not seen. The audience named in the prompt decides whether the cause survives: a plain-language request scores 0.327 on faithfulness against 0.668 for a technical one, one added instruction to name the deciding factor recovers half that gap for five points of reading ease, and across ten open-weight models from six families ($1-42$ gigabytes) the plain request never exceeds 0.42. In the model's own next-word distribution, naming the audience alone lowers the probability of the true cause by 0.137 ($p = 0.0005$). A model judges how plainly a passage reads, agreeing with the formula on 97% of texts, however, it cannot count its words. Counting constraints therefore belong in code, while judgements of style can stay in the prompt. On the benchmark, a four-stage pipeline that fixes the cause before simplifying the language matches a single well-specified prompt on mean faithfulness (0.696 against 0.700); what it adds is that it checks the rules in code.

1 INTRODUCTION

A trained system makes a decision, and the person it affects, usually not a specialist, wants to know why. Given the system's technical record, a language model will write a fluent answer in plain words. This paper asks whether that answer names the reason the system acted on.

The tools that report a model's reason are usually built for specialists e.g., feature attribution tools such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017) report the reason as a weighted list of features, which an analyst can read and a patient cannot. The standard outputs for reinforcement learning agents, such as pixel saliency, reward-component charts and values spread over future time steps (Greydanus et al., 2018; Juozapaitis et al., 2019; Ruggeri et al., 2025), serve a lay reader even less. A language model asked for plain language produces readable text instead, but fluent model-written reasoning can give confident reasons that omit what drove the answer (Turpin et al., 2023), and Agarwal et al. (2024) argue that making explanations plausible to users can cost faithfulness, which matters most in high-stakes decisions.

Two gaps have kept anyone from measuring this trade-off—ground truth and comprehension. Evaluations of attribution often treat one attribution method as the answer key, although the common methods disagree on the same prediction (Krishna et al., 2024), and surveys of explainable reinforcement learning note the lack of specific measurement methods (Qing et al., 2022) and call for benchmarks that allow evaluation against ground truth (Milani et al., 2024). For comprehension,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

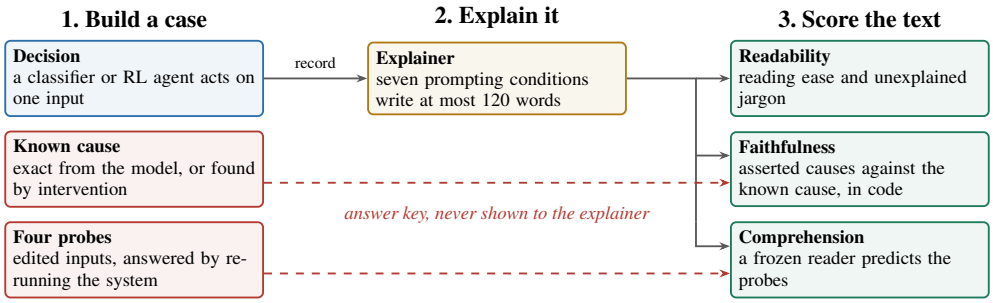

Figure 1: The benchmark in three steps. The known cause and the probe answers go straight to the scorers and never reach the explainer.

Explingo (Zytek et al., 2024) published a system closest to our setting, turning SHAP values into narrative and using a language model to grade it, yet Hase & Bansal (2020) found that people's ratings of explanations did not predict whether those explanations helped them predict the model.

We close both gaps with a benchmark in which the reason for each decision is known by construction, exactly from the model for classifiers and by intervention for agents, and a protocol that scores explanation text against that reason automatically (Figure 1). For agents, intervention means we remove one candidate cause from the world, re-run the agent, and keep the case only if the decision changes.

Our contributions are fourfold. (i) A benchmark of fifty single decisions, 15 from tabular classifiers and 35 from three reinforcement learning agents, each with an exact or interventionally established cause and four probe situations whose answers come from re-running the system. (ii) A scoring protocol in which a program decides faithfulness with no language model in the loop, and a weaker reader from a different model family measures comprehension on unseen situations. (iii) Evidence that asking for plain language drops the cause. The model recovers half the lost information when one added sentence asks the model to name the deciding factor. The model's next-word probabilities show it omitting the cause, and a separate test shows it can judge reading ease but cannot count words. (iv) A test of FD-CR, a pipeline that fixes the cause before simplifying and checks the language rules in code. It matches a single prompt carrying the same rules on mean faithfulness, and what it adds is a check of every explanation against those rules..

The section 2 reviews related work. The sections 3 and 4 describe the benchmark and the scoring protocol, section 5 reports what the benchmark shows, section 6 tests the pipeline against a single prompt, and sections 7 and 8 discuss limitations and the conclusion.

2 Related work

Explaining classifiers and agents. LIME fits a simple model near one input (Ribeiro et al., 2016), SHAP unifies several attribution methods as approximations of Shapley values (Lundberg & Lee, 2017), and Anchors gives if-then rules with a stated precision (Ribeiro et al., 2018). All three produce output for analysts. Rudin (2019) argues that post-hoc explanation of an opaque model is unreliable in principle, so we explain systems that are readable by construction and ask whether the *text* carries the reason. Bastings et al. (2022) plant a known shortcut so that the true reason is known in advance, and our injected features follow their protocol. For reinforcement learning agents, reward decomposition reports an action as a trade-off between named reward components (Juozapaitis et al., 2019). HIGHLIGHTS summarises a policy through important states (Amir & Amir, 2018), and Septon et al. (2023) combine the two in a human study. Saliency maps mark the pixels an agent attends (Greydanus et al., 2018), although Atrey et al. (2020) show with counterfactual tests that such maps suit exploring hypotheses better than explaining behaviour. TripleTree fits a readable global surrogate (Bewley & Lawry, 2021), and temporal policy decomposition spreads an action's value over future steps (Ruggeri et al., 2025). Each of these outputs a chart, a tree, a video or a table, and none comes with an evaluation against a known cause for a single decision.

A further line builds the explanation on an explicit causal model. Madumal et al. (2020) learns an action-influence structural causal model and answers "why" and "why not" questions from it; Yu et al. (2023) learns a causal world model and explains an action by the chain it triggers; and Olson et al. (2021) generates a counterfactual state, the smallest change to what the agent saw that would have changed what it did. These methods usually assume or learn a causal graph. We propose no new mechanism. Instead we supply an explainer-agnostic harness in which the cause of each decision is known, so anyone can score any mechanism (theirs, ours, or a language model asked directly) against the same answer: they intervene to *produce* an explanation, and we intervene to *establish the ground truth* that judges it.

Language models as explainers. Jacovi & Goldberg (2020) separate plausibility from faithfulness and argue that human judgements of an explanation's quality should not be used to evaluate its faithfulness. Turpin et al. (2023) bias a model's answer with a cue its written reasoning never mentions, Lanham et al. (2023) intervene on the reasoning chain, and Madsen et al. (2024) tests whether deleting a word the model calls important changes its prediction. These studies concern a model explaining itself. We study a model explaining a separate classifier or policy, which is the deployment case. Kroeger et al. (2023) prompts a model for feature importances and scores them against logistic-regression coefficients and perturbation tests, with LIME, SHAP and gradient methods as baselines. TalkToModel answers questions about a model in natural language, and its authors evaluate intent parsing and user preference (Slack et al., 2023). Explingo turns SHAP values into narrative and grades it with a language model (Zytek et al., 2024). It reports our trade-off from the inside: more worked examples raise fluency and conciseness and lower accuracy and completeness.

Evaluating explanations with readers. Doshi-Velez & Kim (2017) set out a ladder of evaluation from real users on real tasks down to proxy measures. Studies with people separate how an explanation is rated from what it lets a reader do. Hase & Bansal (2020) found the gains in predicting a model small and rare, and unrelated to how people rated the explanations. In pre-registered experiments with 3,800 participants, Poursabzi-Sangdeh et al. (2021) found that people shown a *clearer* model caught fewer of its large mistakes. Lage et al. (2019) found that longer explanations and new concepts raise response time and lower satisfaction, with new concepts costing more than length. We adopt the counterfactual simulatability of Chen et al. (2024): a good explanation lets a reader predict what the system does on a related but different input. Their counterfactuals come from a model; we produce ours, and their answers, by re-running the actual system. For readability we use Flesch Reading Ease, which scores text from sentence length and syllables (Flesch, 1948; Kincaid et al., 1975). Tanprasert & Kauchak (2021) show that chopping sentences up raises such scores, so we never use the score alone.

3 THE BENCHMARK

3.1 WHAT A CASE IS

A **case** is one decision by one system, with four parts: the input, the system, the decision, and a **technical record** of what a specialist would inspect. For a classifier the record holds the weights and exact per-feature contributions; for an agent, the score of every available action and the reward numbers. Every method we compare receives the same record. An **explanation** is at most 120 words of prose for a reader who has never studied the subject.

3.2 HOW THE REASON IS ESTABLISHED

Each case carries a small set of true causes. We fix them in one of three ways, never by our own judgement.

Exact—for logistic regression the contribution of feature j is $w_j(x_j - \mathbb{E}[x_j])$, its Shapley value when features are independent (Lundberg & Lee, 2017). We keep the smallest set of features whose absolute contributions reach 60% of the total, capped at three. For a depth-three decision tree the reason is the set of features tested on the root-to-leaf path.

By intervention—for every agent case we edit one candidate cause out of the world and re-run the same agent. The candidates are the goal, nearby lava, nearby walls and the slipperiness of the floor in the grid world; the pole state and the cart state in cart-pole; and the exploration bonus, the risk

penalty and each context feature in the bandit. A candidate counts as a cause only if removing it changes the decision, and we keep a case only when exactly one candidate does.

By planting—following Bastings et al. (2022), some cases carry a feature that should be irrelevant but that we made predictive during training. This yields two bandit cases for which the sole reason is a spurious correlation. In the clinical data the model learns the planted feature and gives it the largest weight of any feature (4.42 against at most 1.34), yet the feature decides only one of three patients, because a large weight on a value near its average contributes little. We report the count and build no claim on it.

One design choice changed our results. Deleting the goal changes almost every move an agent makes, so "is the goal a cause?" nearly always gets a yes and distinguishes nothing; our first pass labelled 48 of 65 usable states "goal". We ask what a person would ask instead: *why not head straight for the goal?* A case is goal-seeking when the agent takes the greedy move towards the goal; otherwise the cause is whichever nearby feature, once removed, sends the agent back to that move.

3.3 What the benchmark contains

The benchmark contains fifty cases, fifteen tabular (breast cancer and wine under logistic regression, iris under a decision tree, and three cases carrying the planted feature) and thirty five RL agent cases (15 grid world, 9 cart-pole, 11 bandit). We hold out ten cases. Every case carries four probe situations, 200 in total. Each probe is a real edit of the input or the world, and we answer it by running the system again. We construct the probes so that at least two different decisions occur among them, which penalises a reader who always gives the same answer. Only 71 of the 200 probes change the decision, a fact that matters for reading Section 5.

We label each agent case with one of five reasons: seeking the goal, avoiding an obstacle, exploring what it is unsure about, avoiding a risk, or following a spurious correlation. Each tabular case gets one of three: weighing feature evidence, following a threshold rule, or following a spurious correlation. Appendix A gives the counts.

4 Measuring the text

4.1 Faithfulness, decided by program

*An explanation is **faithful** when the causes it puts forward are the causes that drove the decision, and no others.* We score only the causes which the explanations assert. A cause counts when its name, or one of its listed aliases, appears inside a sentence that gives a reason: a sentence containing "because", "since", "in order to", "to avoid", "drove", "the reason" and similar phrases. A sentence that only describes the situation, such as "there is lava to the left", does not count, because it does not say the lava caused the move. We fixed the trigger phrases and the aliases before generating any explanation (for more detail, read Appendix D).

We compare the causes the explanation asserts with the known causes and report recall (the share of known causes it names), precision (the share of named causes that are correct) and their harmonic mean F_1, and whether the explanation states the **direction**: for a measurement, tying the observed level to the class it supports ("the largest radius is low, and low points to harmless"); for an agent, saying whether it moved toward the thing or away from it. The score is $0.5\,F_1 + 0.5\,\mathrm{direction}$, and zero if the text names no true cause. We fixed two refinements before the main run. Silence about direction does not count as an error, because the reader test already penalises incompleteness. In a grid world we treat the goal as neutral when it is not the distinctive reason, since the agent always tries to reach it. No language model takes part in this measurement.

4.2 Comprehension, measured on situations the reader has not seen

A weaker reader model, smaller than the explainer, from a different family sees the case we explained, the decision taken, the explanation, and then a **different** situation. It must say what the system does. It never sees the record, the weights or the attributions, and we get the answer key by running the actual system. We report accuracy over the four probes, written sim_{cf}, and the same reader's accuracy with no explanation at all.

Table 1: **Results for the seven conditions on the 40 main cases.** Faithfulness and comprehension range from 0 to 1, shown as the mean ± half the 95% bootstrap interval. Jargon is the number of technical terms left unexplained per explanation. Rules met is the share of explanations that satisfy every rule in Section 6.

method	faithfulness	comprehension	reading ease	jargon	rules met
Technical prompt	0.668 ± 0.102	0.606 ± 0.081	25.5	1.5	0.000
Chain-of-thought	0.643 ± 0.126	0.613 ± 0.081	41.7	0.8	0.025
LIME/SHAP verbalised	0.602 ± 0.129	0.669 ± 0.078	35.8	0.5	0.025
Plain-language prompt	0.327 ± 0.123	0.625 ± 0.072	82.2	0.0	0.375
Plain language, but named	0.510 ± 0.134	0.619 ± 0.075	77.2	0.0	0.475
Same rules, prompt only (no checking)	0.700 ± 0.127	0.650 ± 0.075	67.8	0.4	0.400
FD-CR (ours)	0.696 ± 0.123	0.656 ± 0.081	73.2	0.0	0.600

An explanation states the decision it explains, so the reader can repeat that decision as the answer. Showing the original case and the probe as separate sections is the solution.

What a comprehension score must beat. Because only 71 probes change the decision, a reader who ignores the explanation and repeats the decision it was shown scores 0.645, and the best constant answer scores 0.650. The weak reader with no explanation scores 0.594. Therefore, a comprehension score should be compared with these three baselines, not with random guessing: a score of 0.62 is well above chance but below what a reader gets by repeating the original decision.

4.3 Reading ease and jargon

We record Flesch Reading Ease and count the technical terms used without a plain-language gloss at first use, against a frozen list. An explained technical word costs nothing; an unexplained word costs one. Writing fragments can raise reading ease (Tanprasert & Kauchak, 2021), so we also record mean sentence length and count anything below eight words per sentence as a failure.

5 What the benchmark shows

We compare seven ways of writing an explanation from the same record: **technical** (explain precisely, for a data scientist); **chain-of-thought** (reason step by step, then explain); **LIME/SHAP verbalised**, the Explingo setting (Zytek et al., 2024), fed real LIME and KernelSHAP output for tabular cases and the standard local artifact for each agent; **plain language** (explain to someone with no technical background); **plain, but named**, the plain prompt plus one sentence telling the model to name the deciding factor; **rules only**, the full list of written rules in one prompt; and **FD-CR**, the pipeline of Section 6. Every condition has the same 120-word limit. The explainer is `qwen3.8` and the weaker reader `gemma4:e4b`. We compute bootstrap confidence intervals over cases and pair every comparison, with p-values from a sign-flip permutation test. Appendix D gives the exact wording of every prompt.

5.1 How the audience is named governs whether the cause survives

The plain-language prompt produces the easiest text of anything we tested, reading ease 82.2, and names the real cause least often, 0.327. The technical prompt shows the opposite pattern: 0.668 at reading ease 25.5, with 1.5 undefined technical terms per explanation.

The plain prompt never asks the model to name the cause. The **plain, but named** condition keeps the plain-language instruction word for word and adds one sentence asking the model to name the deciding factor by its exact name and say which way it pushed. Faithfulness rises from 0.327 to 0.510 ($+0.183$, interval 0.035 to 0.335, $p = 0.027$), and reading ease stays close to the plain prompt (77.2 against 82.2). The full rule list reaches 0.700, a further 0.190 (interval 0.023 to 0.348, $p = 0.032$). One sentence therefore recovers about half of the lost faithfulness, and the remaining rules recover the rest. Because this condition still asks for plain writing, the loss comes from the missing request to name the cause.

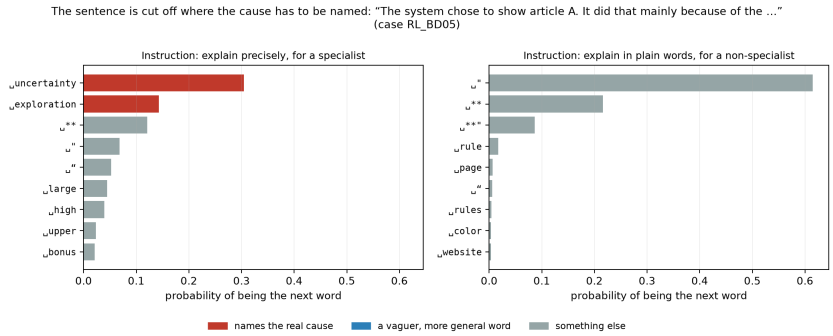

Figure 2: Next-word probabilities where the cause must be named. Red bars name the real cause. For a specialist, the model's top choices are "uncertainty" and "exploration". For a non-specialist, none of its top choices names the cause.

The effect is far greater on tabular cases than on agent cases. On tabular cases the plain prompt scores only 0.056, so almost no causal content survives; adding the one sentence raises this to 0.433, and the full rule list to 0.800. On agent cases the same three conditions score 0.490, 0.557 and 0.640, a much smaller spread. The difference comes from the vocabulary. A grid world has a few concrete objects, such as a wall, lava and the goal, whose names stay the same in plain words. A clinical record has eight measurements that differ only by name, and plain writing drops the names.

5.2 Comprehension does not separate the methods

Explanations barely help the reader. With no explanation, the main reader scores 0.594 on the probes. With one, it scores between 0.606 and 0.681, and no method beats another significantly. Repeating the original decision already scores 0.645.

We repeated the test with a stronger reader (`gemma3:27b`) and a weaker one (`qwen2.5:1.5b`). The stronger reader rises from 0.670 with no explanation to at most 0.745 with one. The weaker reader gains at most 0.030. With both readers the methods still do not differ, and their order changes from reader to reader. We had expected the weaker reader to give a harder test, and it did not. The result matches Hase & Bansal (2020): explanations help a reader a little, and how they are written makes no measurable difference.

5.3 Why plain language loses the cause

We stop the model where the causal noun must appear (*"The system decided benign. This happened mainly because of the . . . "*) and read the probability of every possible next word under two instructions that differ only in the audience they name (Figure 2).

Across 48 cases the model names the specific cause in 61.8% of sampled continuations under the technical instruction and 47.5% under the plain-language one. We expected substitution: the model writing "size" where the record says "worst radius". Some cases show it, but across all 48 the rate of vaguer words stays flat ($p = 0.67$). The share of continuations that name neither the cause nor anything standing in for it grows from 0.212 to 0.335. Written for a non-specialist, the explanation more often leaves the cause out. A reader can ask what a vague word means; a missing cause gives them nothing to ask about. The drop is about twice as large on tabular cases as on agent cases.

5.4 A model can judge how text reads, but cannot count

The reading-ease formula counts words and syllables; a model reads subword tokens. Over 204 distinct words the two counts agree for only 54% and correlate at $r = 0.51$. The mismatch is worst where it matters: jargon words average 2.25 syllables per 1.38 tokens, ordinary words 1.75 per 1.31 tokens.

We then gave the model sixty passages of 81 to 120 words and asked it for two numbers, the word count and the reading ease (Figure 3). Its word counts miss by 13.8 words on average and barely

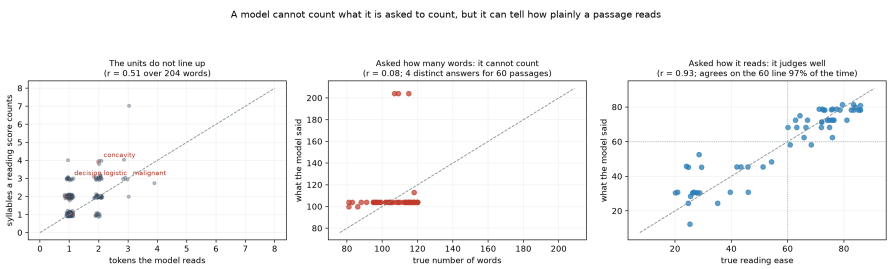

Figure 3: Left: syllables against tokens for 204 words. Middle: the model's word counts against the true counts for 60 passages. Right: its reading-ease estimates against the true scores.

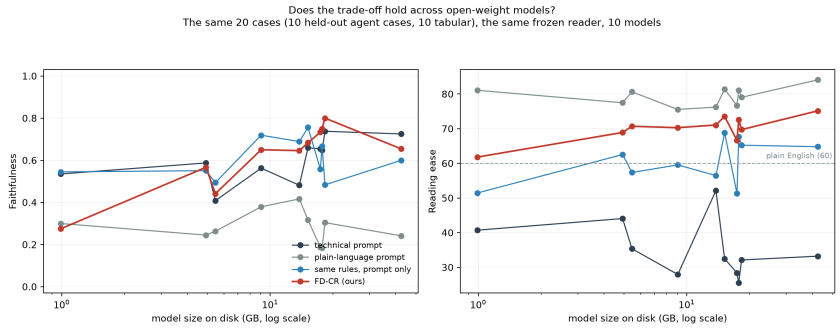

Figure 4: Faithfulness (left) and reading ease (right) for ten open-weight models on the same twenty cases, against model size. The plain-language prompt never exceeds 0.42 faithfulness at any size.

track the truth ($r = 0.085$). It gave only four distinct answers across the sixty passages, and answered 104 for fifty-four of them. Its reading-ease estimates are close: they miss by 6.2 points, correlate with the true score at $r = 0.932$, and put 97% of passages on the correct side.

We designed FD-CR on the assumption that a model could not work to a reading-ease target, so it computes reading ease in code. These results support a narrower rule. A model can judge how plainly a passage reads, so a target such as "reading ease = 60" can stay in the prompt. A count, such as the 120-word limit, has to be checked in code.

5.5 The trade-off holds across ten open-weight models

We ran the same twenty cases (the ten held-out cases and ten tabular ones) through ten models from six families, ranging from about 1 to 42 gigabytes on disk, and scored every model with the same weaker reader (Figure 4). With the plain-language prompt, faithfulness ranges from 0.18 to 0.42 across the ten models, with a mean of 0.284. Larger models do no better: the correlation with size is $r = -0.156$, which on ten models shows no trend. Of all our results, this one holds most widely, across explainers, settings, readers and model sizes.

5.6 How far the benchmark can be trusted

We ran four checks on the benchmark itself; Appendix B gives the details. **Exact causes agree with sampled Shapley values.** Our tabular ground truth, $w_j(x_j - \mathbb{E}[x_j])$, equals the Shapley value only if features are independent, and clinical features are correlated. Therefore, we compute the KernelSHAP on the same instances and training data. In all 11 logistic-regression cases, the feature KernelSHAP ranks first is the one our benchmark records as the cause; the independence assumption changes the order of minor features only. **Most grid-world states have a single cause.** Of the 109 states we examined, one candidate flips the decision in 60%, none does in 25%, two or more do in 7%, and the action is ambiguous in 8%. Section 7 discusses the multi-cause states. **The alias list does not drive the score.** We deleted a random quarter of the aliases and rescored, twenty times.

Table 2: **FD-CR against a single prompt with the same rules**, on the 40 main cases. Difference is FD-CR minus the single prompt.

measure	FD-CR	prompt only	difference
reading ease	73.2	67.8	5.4
undefined technical terms	0.000	0.375	-0.375
every rule satisfied	0.600	0.400	0.200
faithfulness, tabular cases	0.933	0.800	+0.133
faithfulness, agent cases	0.553	0.640	-0.087

Each condition's mean faithfulness moved by at most 0.027, and the ranking of methods stayed the same in all 20 runs. **The ranking does not depend on how we combine the two parts of the score.** Across seven ways of combining cause and direction, the plain-language prompt ranks last every time, and the rules-only prompt and FD-CR rank first or joint first in five.

6 What a pipeline adds, and what a prompt already gives

Section 5 suggests a remedy: fix the cause first, simplify afterwards, and check the language rules with a program. We use a program rather than a judge model because model critics are unreliable: language models abandon correct answers when a user pushes back (Sharma et al., 2024), and their own critiques can make answers worse (Huang et al., 2024). FD-CR has four stages. First, the model names the decisive factor from the record. Second, it rewrites the explanation in plain language under explicit rules: at most 120 words, state the factor, explain every technical word at first use, at most 2 technical words per sentence, a target reading ease of 60, and sentences of 10 to 20 words. Third, a program checks each rule and sends the failures back for repair. Fourth, we sample 4 candidates and pick one without using the ground truth, with a *third* model as reader so that we never tune a method on the model that scores it.

FD-CR reaches faithfulness 0.696 at reading ease 73.2, with no unexplained technical terms. Its reading ease is 47.7 points above the technical prompt ($p < 0.0001$) and 9.0 below the plain prompt. Its faithfulness matches the technical prompt ($+0.027$, $p = 0.70$) and is well above the plain prompt ($+0.369$, $p = 0.0003$). A single prompt with the same rules, and no draft, checking, repair or selection, matches FD-CR on average: 0.700 against 0.696, a paired difference of -0.004 (interval -0.090 to 0.096, $p = 0.96$). We committed to reporting this comparison before running it. Most of the gain therefore comes from writing the rules down: the same model scores 0.327 with the plain prompt and 0.700 with the rule list ($p < 0.0001$).

The extra stages improve individual explanations and leave the average unchanged (Table 2). Checking in code enforces the rules; asking in a prompt does not. The single prompt leaves 0.375 unexplained technical terms per explanation and meets every rule in 40% of explanations, against 60% for FD-CR. Pinning the cause first also matters: removing that stage costs 0.190 in faithfulness at the same reading ease ($p = 0.020$). It helps on the tabular cases, where the cause can be identified.

Pinning the cause hurts when the cause is hard to identify. On grid-world cases FD-CR scores 0.364 against 0.644 for the technical prompt, even though it names the fewest false causes of any method (0.17 against 0.83). When the true reason is a hazard two squares away or a slippery floor, the first stage decides "it was heading for the goal", and every later stage keeps that choice. Methods that name several causes get credit for whichever one is right. On the slippery-floor case, FD-CR without candidate selection names 4 causes and scores 0.75 at a precision of 0.33. We tested two fixes, both stated in advance, on all 40 cases: asking the first stage the benchmark's contrastive question (-0.000, $p = 1.00$), and hiding the decision phrase as a whole rather than word by word during selection ($+0.013$, $p = 1.00$). Neither helped, and both still score 0.364 on the grid world. The problem is the premise itself: the pipeline assumes the first stage can pin down one cause, and here it cannot. We did not try a third fix, because one chosen after seeing these results would fit the method to the test set.

The pipeline is worth its extra calls when every explanation must meet the rules. Practitioners should avoid it when identifying the cause is itself the hard part, because there it turns a hedge into a confident error.

7 LIMITATIONS

Our findings hold for systems whose reasons we can know exactly or find by intervention: logistic regression, a depth-three tree, value iteration, a Q-table and a bandit. We chose them because we need a ground truth we can trust, so we do not show that the findings carry over to deep networks. For those, the planted-shortcut protocol of Bastings et al. (2022), which our injected features already follow, is the practical route. The tabular causes are exact only when features are independent, and clinical features are not. KernelSHAP agrees with them in every logistic-regression case (Section 5.6), but they remain a well-supported approximation rather than a certainty.

Each case has exactly one reason. This matches how people explain things (Miller, 2019) and keeps the scoring unambiguous, but it leaves out the 7% of grid-world states in which more than one candidate changes the decision. A labelled set of multi-cause cases, scored with partial credit, would cover them. We measure comprehension with three model readers, from 1 to 27 billion parameters, and no human participants. This is the third level of evaluation in Doshi-Velez & Kim (2017). Because we release the cases, probes and answer keys, a human study can test the same numbers directly.

The mechanism analysis needs next-word probabilities, so we make no claim about closed commercial models. The mechanisms depend on things those models share (a technical input, subword tokens and instruction following), and our code runs on any model that exposes next-word probabilities. Finally, 40 main cases are enough to detect the large effects we report but not small ones. We flag the comparisons that lack power. On the 10 held-out cases several tests return $p = 1.00$, which means no evidence either way, not evidence that the methods are equal.

8 CONCLUSION

We asked whether a language model can explain another system's decision in plain words without changing the reason, and whether that trade-off can be optimised. Answering required building the instrument first, and the instrument turned out to be the contribution. With it we find that a bare request for plain language costs most of the causal content on every model we tried, whatever its size, that one added sentence naming the deciding factor recovers half, and that the remaining rules recover the rest. Naming the audience drives the model off the causal noun, usually off the subject altogether, while comprehension separates none of the conditions across a twentyfold range of reader size.

Practitioners need no special method: write the constraints down, which is worth 0.373 of faithfulness; add a program to check them when a quietly broken rule is unacceptable, since checking buys a guarantee rather than a better average; and avoid pinning a cause early where identifying it is the hard part. We release the benchmark, every explanation, the frozen scoring code and the three faults we found in our own measurement, so that the next test of a claim about explanation quality can run against a known answer.

AI USE STATEMENT

In this work, we used generative AI tools to help assist us with the selection of prompts for our experiments and also with writing the initial draft. We also used AI for extracting related papers. We have reviewed all AI-assisted work. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 3 describes how we build every case and its ground truth, and Section 4 gives the scoring protocol. Appendix D describes how we cache runs and freeze and hash the benchmark files. Appendix D gives the exact prompt for every condition, together with the trigger phrases, alias tables and technical-term list the scorer uses. The full codebase will be released with the paper in the later stage.

REFERENCES

C. Agarwal, S. H. Tanneru, and H. Lakkaraju. Faithfulness vs. plausibility: On the (un)reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614*, 2024.

D. Amir and O. Amir. HIGHLIGHTS: Summarizing agent behavior to people. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

A. Atrey, K. Clary, and D. Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

J. Bastings, S. Ebert, P. Zablotskaia, A. Sandholm, and K. Filippova. "Will you find these shortcuts?" A protocol for evaluating the faithfulness of input salience methods for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022. doi: 10.18653/v1/2022.emnlp-main.64.

T. Bewley and J. Lawry. TripleTree: A versatile interpretable representation of black box agents and their environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. doi: 10.1609/aaai.v35i13.17360.

Y. Chen, R. Zhong, N. Ri, C. Zhao, H. He, J. Steinhardt, Z. Yu, and K. McKeown. Do models explain themselves? Counterfactual simulatability of natural language explanations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. arXiv:2307.08678.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

R. Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3), 1948. doi: 10.1037/h0057532.

S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding Atari agents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

P. Hase and M. Bansal. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. doi: 10.18653/v1/2020.acl-main.491.

J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024.

A. Jacovi and Y. Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. doi: 10.18653/v1/2020.acl-main.386.

Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI Workshop on Explainable Artificial Intelligence*, 2019.

J. P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom. Derivation of new readability formulas for Navy enlisted personnel. Technical report, Naval Technical Training Command, 1975.

S. Krishna, T. Han, A. Gu, S. Wu, S. Jabbari, and H. Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. *Transactions on Machine Learning Research*, 2024. arXiv:2202.01602.

N. Kroeger, D. Ley, S. Krishna, C. Agarwal, and H. Lakkaraju. In-context explainers: Harnessing LLMs for explaining black box models. *arXiv preprint arXiv:2310.05797*, 2023.

I. Lage, E. Chen, J. He, M. Narayanan, B. Kim, S. Gershman, and F. Doshi-Velez. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*, 2019.

T. Lanham, A. Chen, A. Radhakrishnan, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

A. Madsen, S. Chandar, and S. Reddy. Are self-explanations from large language models faithful? In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. doi: 10.18653/v1/2024.findings-acl.19.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. arXiv:1905.10958.

S. Milani, N. Topin, M. Veloso, and F. Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7), 2024. doi: 10.1145/3616864.

T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 2019. doi: 10.1016/j.artint.2018.07.007.

Matthew L. Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence*, 295:103455, 2021. doi: 10.1016/j.artint.2021.103455.

F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. Wortman Vaughan, and H. Wallach. Manipulating and measuring model interpretability. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021. doi: 10.1145/3411764.3445315.

Y. Qing, S. Liu, J. Song, Y. Zhou, K. Chen, H. Wang, and M. Song. A survey on explainable reinforcement learning: Concepts, algorithms, challenges. *arXiv preprint arXiv:2211.06665*, 2022.

M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016. doi: 10.1145/2939672.2939778.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. doi: 10.1609/aaai.v32i1.11491.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1, 2019. doi: 10.1038/s42256-019-0048-x.

F. Ruggeri, A. Russo, R. Inam, and K. H. Johansson. Explainable reinforcement learning via temporal policy decomposition. *arXiv preprint arXiv:2501.03902*, 2025.

Y. Septon, T. Huber, E. André, and O. Amir. Integrating policy summaries with reward decomposition for explaining reinforcement learning agents. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics (PAAMS)*, 2023. doi: 10.1007/978-3-031-37616-0_27.

M. Sharma, M. Tong, T. Korbak, D. Duvenaud, A. Askell, S. R. Bowman, et al. Towards understanding sycophancy in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

D. Slack, S. Krishna, H. Lakkaraju, and S. Singh. Explaining machine learning models with interactive natural language conversations using TalkToModel. *Nature Machine Intelligence*, 2023. doi: 10.1038/s42256-023-00692-8.

T. Tanprasert and D. Kauchak. Flesch-Kincaid is not a text simplification evaluation metric. In *Proceedings of the Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, 2021. doi: 10.18653/v1/2021.gem-1.1.

M. Turpin, J. Michael, E. Perez, and S. R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. Explainable reinforcement learning via a causal world model. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2023. arXiv:2305.02749.

A. Zytek, S. Pido, S. Alnegheimish, L. Berti-Equille, and K. Veeramachaneni. Explingo: Explaining AI predictions using large language models. In *Proceedings of the IEEE International Conference on Big Data*, 2024. doi: 10.1109/bigdata62323.2024.10825114.

A BENCHMARK COMPOSITION AND RESULTS BY REASON

Table 3 breaks the fifty cases down by setting. Table 4 breaks them down by the true reason for the decision and gives each condition's faithfulness on the 40 main cases.

Table 3: **The fifty cases by setting.** Quality is test accuracy for the classifiers and, for cart-pole, the share of the 500-step maximum the agent reaches. The grid world and the bandit are solved exactly, so they have no quality score.

setting	what decides	main	held out	probes	quality
breast cancer	logistic regression	5	0	20	0.95
breast cancer, planted feature	logistic regression, planted feature	3	0	12	0.99
wine	logistic regression	3	0	12	0.98
iris	decision tree, depth three	4	0	16	0.98
grid world	value iteration	11	4	60	-
cart-pole	Q-learning on a coarse grid	6	3	36	1.00
bandit	contextual bandit, upper confidence bound	8	3	44	-
total		40	10	200	

Table 4: **Case counts and faithfulness by the true reason for the decision.** Faithfulness is the mean over the 40 main cases.

true reason	main	held out	Technical prompt	Chain-of-thought	LIME/SHAP verbalised	Plain-language prompt	Same rules, prompt only	FD-CR (ours)
weighing feature evidence	11	0	0.720	0.894	0.826	0.076	0.826	0.924
following a threshold rule	4	0	0.517	0.412	0.417	0.000	0.729	0.958
seeking the goal	4	2	0.875	0.688	0.875	0.917	1.000	1.000
avoiding an obstacle	5	2	0.200	0.200	0.167	0.250	0.067	0.000
exploring what it is unsure of	4	2	0.667	0.917	0.750	0.750	1.000	0.667
avoiding a risk	10	4	0.792	0.633	0.600	0.433	0.667	0.517
following a spurious correlation	2	0	0.833	0.250	0.000	0.000	0.500	1.000

B FULL RESULTS

Table 5: **All conditions on the 40 main cases.** Rows starting with a dash are FD-CR with one stage removed or changed (Section 6). Gain is comprehension minus the no-explanation score of 0.594. Words is the mean length of an explanation.

method	faithfulness	comprehension	gain	combined	reading ease	jargon	words	rules met
Technical prompt	0.668 ± 0.102	0.606 ± 0.081	0.013	0.553	25.5	1.5	107.5	0.000
Chain-of-thought	0.643 ± 0.126	0.613 ± 0.081	0.019	0.518	41.7	0.8	93.3	0.025
LIME/SHAP verbalised	0.602 ± 0.129	0.669 ± 0.078	0.075	0.542	35.8	0.5	103.4	0.025
Plain-language prompt	0.327 ± 0.123	0.625 ± 0.072	0.031	0.291	82.2	0.0	111.5	0.375
Plain language, but named	0.510 ± 0.134	0.619 ± 0.075	0.025	0.443	77.2	0.0	112.7	0.475
Same rules, prompt only (no checking)	0.700 ± 0.127	0.650 ± 0.075	0.056	0.588	67.8	0.4	104.8	0.400
FD-CR (ours)	0.696 ± 0.123	0.656 ± 0.081	0.062	0.592	73.2	0.0	109.0	0.600
- without best-of-N	0.735 ± 0.104	0.681 ± 0.078	0.087	0.645	68.7	0.0	110.8	0.450
- without repair loop	0.680 ± 0.121	0.669 ± 0.081	0.075	0.591	70.9	0.0	111.7	0.575
- without reader in selection	0.708 ± 0.123	0.662 ± 0.081	0.069	0.601	74.3	0.0	108.7	0.600
- without separate faithful draft	0.506 ± 0.136	0.613 ± 0.084	0.019	0.423	74.2	0.0	104.4	0.450
- contrastive first stage	0.696 ± 0.128	0.625 ± 0.069	0.031	0.568	70.6	0.0	110.6	0.425
- decision phrase masked as a phrase	0.708 ± 0.123	0.662 ± 0.081	0.069	0.601	73.5	0.0	108.9	0.600

Held-out cases. All ten held-out cases come from reinforcement learning. Reading ease holds up there: 74.0 against 27.0 for the technical prompt (interval 41.0 to 53.5, p = 0.0015), with no undefined

technical terms against 1.2, while faithfulness does not separate. At n = 10 the permutation test returns p = 1.00 for several comparisons, meaning no evidence either way.

Robustness checks. These numbers support Section 5.6. Against KernelSHAP on the same instance, the exact contribution picks the same leading feature in ten of eleven logistic-regression cases, and the median rank correlation between the two magnitude orderings is 1.00. The seven ways of combining the two faithfulness components are either component alone, three weightings, a hierarchical form in which direction counts only when the cause was recalled, and recall or precision alone. Both exceptions to the ordering follow from the definitions: on recall alone the technical prompt leads, because it names several candidate causes and collects credit for the one that is right, and on precision alone FD-CR leads, because it names one.

C Readers and explainers

Table 6: **Ten open-weight explainers on the same twenty cases**, all scored by the same frozen reader. Size is on disk.

explainer	family	size (GB)	technical (faithful)	plain prompt (faithful)	prompt only (faithful)	FD-CR (faithful)	technical (reading ease)	FD-CR (reading ease)
qwen2.5:1.5b	Qwen2.5	0.99	0.54	0.30	0.55	0.28	40.70	61.80
llama3.1:8b	Llama	4.92	0.59	0.24	0.55	0.57	44.10	68.90
glm4:9b	GLM-4	5.46	0.41	0.26	0.49	0.44	35.40	70.70
phi4:latest	Phi-4	9.05	0.56	0.38	0.72	0.65	28.00	70.30
gpt-oss:20b	gpt-oss	13.79	0.48	0.42	0.69	0.65	52.10	71.10
mistral-small3.2:latest	Mistral	15.18	0.66	0.32	0.76	0.68	32.50	73.50
gemma3:27b	Gemma3	17.40	0.65	0.19	0.56	0.73	28.40	66.60
qwen3.8:latest	Qwen3.5	17.74	0.65	0.18	0.67	0.75	25.50	72.60
hf.co/unsloth/GLM-4.7-Flash-GGUF:Q4_K_M	GLM-4.7	18.31	0.74	0.30	0.48	0.80	32.10	69.70
llama3.3:70b	Llama	42.52	0.72	0.24	0.60	0.65	33.20	75.10

FD-CR benefits from scale more than any other condition: r = +0.870 against the logarithm of size on disk, against +0.600 for the technical prompt. In practice this is a threshold. Below about thirteen gigabytes FD-CR averages 0.484 against 0.577 for stating the rules in one prompt and loses on four models out of four; above it, FD-CR averages 0.711 against 0.626 and wins on four of six. A four-stage procedure asks a model to do four things well, and a model that cannot is better served by a single prompt.

One model, `gpt-oss:20b`, returned an empty answer on 19 of its first 20 cases: with a 6000-token budget it produced 22,195 characters of reasoning and no answer. We added a fallback that retries with a larger budget and then with reasoning capped, after which it answers. We record empty output as missing and do not score it as zero, because a model that produced nothing has not written a bad explanation.

D Reproduction details and prompts

We cache every model call on disk by the exact request, so we can interrupt and resume a run without changing any number. The benchmark files are frozen, and every run log begins by printing their hashes. We keep three models apart: the explainer under study, a reader used *inside* our method to choose candidates, and the frozen scoring reader, which no method touches. The last two must differ, or we would tune the method on the metric that judges it.

Prompts. Every condition receives the same block first: the situation, the decision, the options and the technical record, written below as `<RECORD>`. The released code gives it in full.

Technical prompt.

```
<RECORD>
You are writing for a data scientist who knows this kind of model well.
Explain why the system made this decision. Be precise about which factor drove
it and why. Use at most 120 words. Write prose, no bullet points, no headings.
```

Chain-of-thought.

```
<RECORD>
First think step by step about which single factor drove this decision and how
you know. Then write the explanation.
Format:
THINKING: <your reasoning>
EXPLANATION: <at most 120 words of prose>
```

LIME/SHAP verbalised (the record is replaced by the tool's output).

```
<RECORD>
The record above is the output of a standard explanation tool.
Turn that output into a natural-language explanation of the decision for
someone who is not a specialist. Use at most 120 words. Prose only.
```

Plain language. The *plain, but named* condition inserts the marked sentence before the word limit and changes nothing else.

```
<RECORD>
Explain why the system decided this, to someone with no technical background
at all - imagine a twelve-year-old. Use short sentences and everyday words.
[plain, but named only:] One thing you must do: name the single factor that
decided the outcome, using the exact name it has in the record above, and say
which way it pushed. Do not replace that name with a general word.
Use at most 120 words. Prose only, no bullet points.
```

Rules only (the same rules as FD-CR, one prompt, nothing checked).

```
<RECORD>
Write an explanation for someone with no technical training, following every rule:
1. At most 120 words.
2. Say plainly that the decision was driven by: the factor that actually decided
   the outcome. Do not replace it with a different reason and do not drop it.
3. The first time you use any technical word, explain it in plain words in the
   same sentence, in brackets or after a dash.
4. At most 2 technical words in any one sentence.
5. Everyday words. Aim for a reading ease score of 60 or more.
6. Write proper sentences of about ten to twenty words. Do not write a list of
   three-word fragments; chopped-up text is not clearer.
7. Say which way the factor pushed: say whether it was high or low, or whether
   the system was moving towards it or away from it.
8. Do not quote numbers unless the number helps the reader.
9. Prose only. No bullet points, no headings, no preamble.

Write only the rewritten explanation.
```

FD-CR uses a draft prompt, then a rewrite prompt carrying the same rule list, then a repair prompt that quotes the specific failures the program found.

Triggers, aliases and technical terms. Faithfulness counts a cause only when a sentence carrying one of 68 reason-giving phrases ("because", "driven by", "deciding factor", "in order to" and others) names it. The scorer matches each known cause by its own name or by one of a list of alternatives, so "exit", "target" and "goal square" all count as the grid-world goal. A technical term counts against an explanation when the text uses it without a plain-word gloss at first appearance.

E WORKED EXAMPLES

We show one case for each of the five systems: the lowest-numbered case in its setting (for logistic regression, the breast-cancer model). The two conditions shown are the endpoints of the main finding. The released data contains every explanation produced in every condition.

ML00 · breast_cancer_logreg · decision: **benign** · true reason: **worst radius (low)** (feature-evidence)

Technical prompt (faithfulness 0.83, reading ease 20, undefined technical words 1). The benign prediction was overwhelmingly driven by the `worst radius` feature. With a value of 13.36 and a negative weight of -2.2, this feature contributed +6.41 to the benign score, indicating that a lower worst radius strongly signals benignity in this model. This positive evidence effectively overcame the opposing signal from `mean radius` (value 11.76, weight 1.36), which contributed -3.24 toward benignity. The remaining features provided negligible net influence, with contributions ranging from +0.33 to -0.001. Consequently, the decision hinges on the model's interpretation that a relatively small maximum radius is a decisive indicator of benign status, outweighing the moderate mean radius.

Plain-language prompt (faithfulness 0.00, reading ease 87, undefined technical words 0). Imagine the computer is a detective looking for clues. It checks eight different things about the cell. Most clues point to "benign,"

which means safe. The biggest clue is the cell's size. It is quite small. Small cells are usually not dangerous, so this clue strongly supports the "safe" guess. Another clue is how smooth the cell is. It is fairly smooth, which also helps the "safe" side. There is one clue that worries the detective a little. It looks at the cell's texture, and that part is a bit rough. This pushes the guess slightly toward "dangerous." But the "safe" clues are much stronger.

ML05 · iris_tree · decision: **virginica** · true reason: **petal length (cm) (high), petal width (cm) (high)** (threshold-rule)

Technical prompt (faithfulness 0.67, reading ease 20, undefined technical words 2). The decision tree's logic hinges almost entirely on petal dimensions, rendering sepal measurements irrelevant to the final classification. The first split immediately isolates the sample because its petal length of 6.1 cm exceeds the 2.35 cm threshold, bypassing the setosa branch. Subsequent splits further confirm this trajectory: the petal width of 2.3 cm surpasses the 1.65 cm cutoff, and the petal length again exceeds the 4.85 cm limit. These three consecutive "no" answers force the path exclusively toward the virginica leaf. Essentially, the model identifies that large petals are the definitive discriminator here; since this specimen exhibits extreme petal size relative to the dataset, it is unambiguously classified as virginica, regardless of its average sepal width.

Plain-language prompt (faithfulness 0.00, reading ease 84, undefined technical words 0). Imagine you are sorting flowers into three boxes. The system asks simple yes-or-no questions to find the right box. First, it checks if the petal is short. Your flower has a very long petal, so the answer is no. Next, it checks if the petal is thin. Yours is wide, so that is also no. Finally, it asks if the petal is medium-sized. Again, yours is too long. Because your flower has long, wide petals, it does not fit in the boxes for the other two types. It only fits in the last box. That box is for the virginica flower. The system just followed a list of rules to match your flower's size to the correct group.

RL00 · gridworld · decision: **right** · true reason: **goal (toward)** (goal-seeking)

Technical prompt (faithfulness 1.00, reading ease 30, undefined technical words 2). The system selected "right" because it yields the highest expected future score of +1.0000, directly maximizing the immediate reward of reaching the goal. With zero slip probability, the robot deterministically transitions to the goal state, securing the +1.0 terminal reward minus the negligible step cost. In contrast, moving up or left incurs a step penalty and delays the goal, resulting in discounted future values of +0.8245. Moving down is suboptimal at +0.9100, likely due to a longer path or higher cumulative step costs before reaching the goal. Since the discount factor (0.95) heavily weights immediate rewards, the direct path to the goal dominates all alternatives, making "right" the unique optimal action under the value iteration policy.

Plain-language prompt (faithfulness 0.83, reading ease 85, undefined technical words 0). Imagine you are a robot on a grid. You want to reach the green goal square. You also want to avoid the hot lava and take as few steps as possible. The robot looks at every direction it can go. It calculates how good each choice is. Going right leads straight to the goal. That is the best possible score. Going up or left takes extra steps, which costs a little bit of points. Going down hits a wall, which is useless. Because going right gets the robot to the finish line immediately without any penalties, it is the smartest move. The robot chooses right because it wins the game fastest and safest.

RL_BD00 · bandit · decision: **show article C** · true reason: **how little this option has been tried (toward)** (uncertainty-exploration)

Technical prompt (faithfulness 0.67, reading ease 28, undefined technical words 0). The selection of Article C was driven entirely by the exploration bonus, which overwhelmed its negative predicted payoff. With only two prior impressions, the system's uncertainty estimate was maximal, yielding a bonus of +3.6965. This far exceeded the bonuses for Articles A (+1.6531) and B (+0.3734), whose higher impression counts (10 and 196, respectively) had already reduced their uncertainty. Although Article B had the highest predicted payoff (+0.7703), its low uncertainty bonus resulted in a lower total score (+1.1437) compared to C's total (+3.1110). The spread penalty was zero for all options, so variance did not influence the ranking. Consequently, the UCB rule prioritized resolving the high uncertainty surrounding Article C over exploiting the known, albeit modest, performance of Article B.

Plain-language prompt (faithfulness 0.67, reading ease 88, undefined technical words 0). Imagine you are picking a card from a box. You have three cards: A, B, and C. Card B has been picked many times, so you know what to expect. Card A has been picked a few times. But Card C has only been picked twice. Because you don't know much about Card C, the system gets very curious. It adds a big "bonus" for this curiosity. Even though Card C might not be the best choice right now, the system wants to learn more about it. It thinks, "I should try this one to see if it's actually good." So, it picks Card C to gather more information.

RL_CP00 · cartpole · decision: **push left** · true reason: **the pole is tipping (away)** (risk-avoidance)

Technical prompt (faithfulness 0.00, reading ease 27, undefined technical words 1). The decision to push left stems from the Q-learning agent's learned value function, which assigns a higher expected return to this action in

the specific state bucket (1, 1, 1, 1). Although the pole is currently leaning left and accelerating further left, the cart is already positioned right of center and moving right. Pushing left serves a dual purpose: it counteracts the pole's leftward angular velocity while simultaneously decelerating the cart's rightward drift, preventing it from exiting the track. The score differential (+94.15 vs. +91.36) indicates that the agent has learned this combined corrective action yields a marginally higher long-term survival probability than pushing right, which would exacerbate the cart's rightward momentum and potentially overshoot the boundary before the pole stabilizes.

Plain-language prompt (faithfulness 0.00, reading ease 91, undefined technical words 0). Imagine you are pushing a shopping cart with a tall stick balanced on top. The cart is moving to the right. The stick is leaning to the left and falling that way. To stop it from falling, you need to push the cart to the left. This moves the base under the stick to catch it. The computer learned this by playing the game many times. It knows that pushing left works better here. It gave this move a higher score than pushing right. Pushing right would make the stick fall faster. So, the system chose the smart move to keep the stick standing up. It is like catching a falling book by moving your hand under it.