Learning from Logical Reasoning Steps through Process Supervision in Neuro-Symbolic Architectures

Acile Chahboun

October 20, 2026

Motivation: Correct Answer $\neq$ Correct Reasoning

Problem

- Neural models can produce the correct answer for the wrong reason
- They may rely on shortcuts or unintended correlations [6]
- Standard outcome supervision evaluates only the final prediction
- A correct output therefore does not guarantee a valid or generalizable reasoning process

Example

Given:

$$\text{Parent}(\text{Alice}, \text{Bob}) \qquad \text{Parent}(\text{Bob}, \text{Charlie})$$

Correct reasoning:

$$\text{Parent}(\text{Alice}, \text{Bob}) \land \text{Parent}(\text{Bob}, \text{Charlie}) \rightarrow \text{Grandparent}(\text{Alice}, \text{Charlie})$$

But suppose the training data accidentally contains the shortcut:

$$\text{SameSurname}(\text{Alice}, \text{Charlie}) \rightarrow \text{Grandparent}(\text{Alice}, \text{Charlie})$$

Why Outcome Supervision Fails

Both rules can produce:

Alice is Charlie's grandparent

Outcome supervision: ✓ correct answer

Problem: it cannot tell which reasoning process produced it ×

This raises the question:

Can we inspect the model's internal reasoning, extract it as logical rules, evaluate those rules, and use that evaluation during training?

Useful Terminology: Neuro-Symbolic AI

Combines neural learning with logical or symbolic structure.

Architecture Example: Neural Logic Machines (NLM) [4]: reasoning structure is learned from examples

1. Input facts are stored as relation tables. Example: $\mathrm{Parent}(x, y)$ is one table containing a value for every pair of people.
2. The architecture can add, reorder, or remove a variable (quantification-like operations) so different relations can be combined.
3. A small neural network looks at the available relations for each tuple of objects and produces new learned relation tables.
4. These learned relations are passed to the next layer, allowing several reasoning steps to be composed.

Neural Logic Machines: Example

$$\mathrm{Parent}(x, z),\ \mathrm{Parent}(z, y)$$

may lead one learned relation table to become high for pairs (x, y) that behave like:

$$\mathrm{Grandparent}(x, y)$$

The model is not given the Grandparent rule directly; the useful relational computation is learned from examples.

Useful Terminology: Logic Tensor Networks

Architecture Example: Logic Tensor Networks (LTN) [2]: supplied logical knowledge influences learning

1. Objects are represented numerically.
2. Predicates such as $\mathrm{Parent}(x, y)$ are differentiable functions returning truth values between 0 and 1.
3. Logical operators (AND, implication) and quantifiers ("for all") are also converted into differentiable numerical operations.
4. A supplied formula can therefore be assigned a numerical satisfaction value.
5. Gradient descent changes the learnable predicates/functions so that the supplied formulas become better satisfied.

Logic Tensor Networks: Example

$$\mathrm{Parent}(x, y) \to \mathrm{Related}(x, y)$$

can be supplied directly. If the model gives high truth to $\mathrm{Parent}(\mathrm{Alice}, \mathrm{Bob})$ but low truth to $\mathrm{Related}(\mathrm{Alice}, \mathrm{Bob})$, the formula has low satisfaction and training pushes the model toward satisfying it better.

Important Distinction

Neural Logic Machines learn more of the relational reasoning structure from examples, whereas Logic Tensor Networks can use logical formulas supplied beforehand during training.

Useful Terminology: Rule Extraction

Converts the behaviour of a trained model into symbolic rules that can be inspected. [1]

Pedagogical

Example: TREes PArroting Networks (TREPAN) [3]

1. Give examples to the trained neural network.
2. Record its predictions.
3. Train a decision tree to imitate those predictions.
4. Convert paths through the tree into IF–THEN rules.

$\rightarrow$ Pedagogical Extraction looks mainly at the model from the outside.

Rule Extraction: Decompositional

Example : Efficient CLAuse- wlse Rule Extraction (ECLAIRE) [5]

Step 1: Run examples through the model and record hidden activations.

Step 2: Learn a rule over hidden activations:

$$h_1 > 0.7 \land h_3 > 0.6 \rightarrow \text{Illness}$$

Step 3: For every patient, check whether the whole hidden condition is true or false.

Step 4: That TRUE/FALSE result becomes a temporary label attached to the patient's original features.

Step 5: Learn which original features predict that label:

$$\text{Fever} \land \text{Cough} \rightarrow (h_1 > 0.7 \land h_3 > 0.6)$$

Rule Extraction: Decompositional

Step 6: Substitute it into the hidden rule:

$$\text{Fever} \land \text{Cough} \rightarrow \text{Illness}$$

$\rightarrow$ Decompositional Extraction uses information from inside the model.

Rule Extraction: Eclectic

Example: DEcision DEteCtion by rule extraction from neural networks (DEDEC) [8]

1. Train the neural network.
2. Inspect the network's learned weights to estimate which input variables are most important.
3. Use that internal information to decide which input combinations should be examined first.
4. Run those inputs through the trained network and observe its outputs.
5. Use those input-output cases to construct symbolic rules.

$\rightarrow$ Eclectic Extraction uses internal information + observed model behaviour (essentially combining the two approaches)

Useful Terminology: Process Supervision

Evaluates *how* the answer was reached, not only whether the final answer is correct. [7]

$$3 + 4 \times 2$$

Outcome supervision:

$$11 \quad \checkmark$$

Process supervision also checks:

$$4 \times 2 = 8 \quad \checkmark \qquad 3 + 8 = 11 \quad \checkmark$$

Idea: give training feedback on the reasoning process itself, rather than only the final result.

General Idea / Intuition Behind the Thesis

Main idea

Observe internals → Extract rules → Validate → Process loss → Train

1. Observe internal behaviour

Inspect which internal relational features activate while the model solves a task.

Example input:

$$\mathrm{Parent}(\mathrm{Alice}, \mathrm{Bob}), \qquad \mathrm{Parent}(\mathrm{Bob}, \mathrm{Charlie})$$

Suppose an internal learned relation h_1 becomes highly active for:

$$h_1(\mathrm{Alice}, \mathrm{Charlie})$$

2. Extract the Activation Rule

Determine which previous relations consistently cause that internal activation.

$$\mathrm{Parent}(x, z) \wedge \mathrm{Parent}(z, y) \rightarrow h_1(x, y)$$

A later internal step may be:

$$h_1(x, y) \rightarrow \mathrm{Grandparent}(x, y)$$

So the extracted chain becomes:

$$\mathrm{Parent} \wedge \mathrm{Parent} \rightarrow h_1 \rightarrow \mathrm{Grandparent}$$

3. Check Extraction Fidelity

Before judging the reasoning, we first check whether the extracted rules actually describe the model.

Example: when $\mathrm{Parent}(x, z) \wedge \mathrm{Parent}(z, y)$ holds, does $h_1(x, y)$ actually activate?

- Compare rule predictions with the model's real internal activations
- Repeat on unseen examples

4. Evaluate Reasoning Quality

Legality: Is each extracted step compatible with the architecture's logical operations and semantics?

Consistency: Do the extracted rules avoid contradictory conclusions?

Bad example: $h_1(x, y) \rightarrow \mathrm{Grandparent}(x, y)$ and $h_1(x, y) \rightarrow \neg\mathrm{Grandparent}(x, y)$

Coherence: Do the extracted steps connect into a meaningful chain $\mathrm{Parent} \wedge \mathrm{Parent} \rightarrow h_1 \rightarrow \mathrm{Grandparent}$, rather than being unrelated internal rules?

4. Evaluate Reasoning Quality: Relevance / Shortcut Detection

Is the model using information relevant to the task?

Desired:

$$\text{Parent}(x, z) \land \text{Parent}(z, y) \rightarrow \text{Grandparent}(x, y)$$

Shortcut:

$$\text{SameSurname}(x, y) \rightarrow \text{Grandparent}(x, y)$$

4. Evaluate Reasoning Quality: Known Ground Truth

Known ground-truth reasoning, where available

If the dataset was created using a known logical rule, compare the extracted rule with that rule.

Example: suppose the dataset was generated using $\mathrm{Bird}(x) \to \mathrm{CanFly}(x)$, but in the training data every bird also happens to have wings.

The model could therefore learn $\mathrm{HasWings}(x) \to \mathrm{CanFly}(x)$ and still fit the training examples.

Because we know which rule generated the dataset, we can distinguish the intended reasoning rule from a correlated shortcut.

5. Use Reasoning Quality for Training

Convert validation into a process loss:

$$\text{better reasoning} \rightarrow \text{lower } L_{\text{process}}$$

$$\text{illegal / inconsistent / incoherent / shortcut reasoning} \rightarrow \text{higher } L_{\text{process}}$$

Then train end-to-end using:

$$\boxed{L = L_{\text{process}}}$$

Our intuition
Do not supervise only the final answer, supervise the quality of the internal reasoning process used to reach it.

Why Layer-by-Layer Rule Extraction Might Be Possible

1. What happens in an ordinary neural network?

Suppose the network receives two inputs:

$$x_1 = \mathrm{Parent}(\mathrm{Alice}, \mathrm{Bob})$$

$$x_2 = \mathrm{Parent}(\mathrm{Bob}, \mathrm{Charlie})$$

A neuron in the first hidden layer receives x_1 and x_2, multiplies them by learned weights, adds a bias, and produces an activation:

$$x_1,\ x_2 \to h_1$$

Why Layer-by-Layer Rule Extraction Might Be Possible

The next layer then receives h_1 together with other hidden activations:

$$h_1,\ h_2,\ \ldots \rightarrow h_3$$

Eventually the network produces the output:

$$\mathrm{Grandparent}(\mathrm{Alice}, \mathrm{Charlie})$$

So the network already computes the answer through a sequence of smaller layer-to-layer transformations.

Why Layer-by-Layer Rule Extraction Might Be Possible

2. Rule extraction can approximate some of these transformations

Suppose we observe the network on many examples and find:

h_1 becomes highly active whenever both x_1 and x_2 are true.

Then we may approximate its behaviour with:

$$x_1 \wedge x_2 \rightarrow h_1$$

or:

$$\mathrm{Parent}(\mathrm{Alice}, \mathrm{Bob}) \wedge \mathrm{Parent}(\mathrm{Bob}, \mathrm{Charlie}) \rightarrow h_1$$

Why Layer-by-Layer Rule Extraction Might Be Possible

A later transformation might be approximated as:

$$h_1 \to \mathrm{Grandparent}(\mathrm{Alice}, \mathrm{Charlie})$$

So instead of explaining the whole network at once, we can try to explain one transformation at a time.

Why Layer-by-Layer Rule Extraction Might Be Possible

3. Why might neuro-symbolic models make this easier?

In an ordinary neural network, h_1 is usually just an anonymous numerical feature. We know it has some value, but the architecture does not tell us what objects or relation it represents.

In a Neural Logic Machine, hidden features keep a relational structure. For example, instead of only:

$$h_1 = 0.95$$

we can have:

$$h_1(\mathrm{Alice}, \mathrm{Charlie}) = 0.95$$

Why Layer-by-Layer Rule Extraction Might Be Possible

$$h_1(\text{Alice}, \text{Charlie}) = 0.95$$

meaning: learned relation feature h_1 is highly active specifically for the pair Alice–Charlie.

The same feature also has values for:

$$h_1(\text{Alice}, \text{Bob}), \quad h_1(\text{Bob}, \text{Charlie}), \quad \ldots$$

We can then ask:

What relational conditions make $h_1(x, y)$ become active?

Why Layer-by-Layer Rule Extraction Might Be Possible

Perhaps we discover:

$$\mathrm{Parent}(x, z) \wedge \mathrm{Parent}(z, y) \rightarrow h_1(x, y)$$

That structure may make the hidden computation easier to express as a symbolic rule.

Why Layer-by-Layer Rule Extraction Might Be Possible

4. Why extract locally?

Instead of asking:

"What single rule explains the entire network?"

We want to ask smaller questions:

What makes h_1 activate?

then:

How does the next layer use h_1?

Why Layer-by-Layer Rule Extraction Might Be Possible

So we investigate:

$$\text{Layer}_0 \rightarrow \text{Layer}_1$$

then:

$$\text{Layer}_1 \rightarrow \text{Layer}_2$$

then:

$$\text{Layer}_2 \rightarrow \text{Output}$$

Our hypothesis
Smaller layer-to-layer transformations may be easier to extract faithfully, and neuro-symbolic models may make them easier to interpret because their internal representations already preserve objects and relations.

Broad Goals

1. **Extract the model's intermediate reasoning**
   Recover symbolic rules that describe what happens inside the model.
2. **Find suitable neuro-symbolic architectures**
   Determine which architectures expose internal representations that can be extracted reliably.
3. **Validate the extracted reasoning**
   First check that the rules faithfully represent the model, then evaluate their reasoning quality.
4. **Define measurable reasoning-quality criteria**
   For example: legality, consistency, coherence, relevance, and agreement with known rules where available.

Broad Goals

5. **Turn reasoning quality into a training signal**
   Use the validation results to construct a process loss.

6. **Investigate process-only training**
   Train the model end-to-end using the process signal rather than supervising only the final answer.

7. **Evaluate the result**
   Test reasoning quality, task performance, generalization, interpretability, and computational cost.

Broad Thesis Framework

1. **Observe internal model behaviour**
   Record activations or learned relational features.

2. **Extract symbolic rules**
   Example:
   $$\mathrm{Parent}(x, z) \wedge \mathrm{Parent}(z, y) \rightarrow h_1(x, y)$$

3. **Check extraction fidelity**
   Compare the extracted rule with the model's actual internal activations, including on unseen examples.

Broad Thesis Framework

4. **Evaluate reasoning quality**
   Check criteria such as legality, consistency, coherence, relevance, and known ground-truth rules where available.

5. **Define a process loss**
   Better reasoning $\rightarrow$ lower loss; poorer reasoning $\rightarrow$ higher loss.

6. **Train end-to-end using the process loss**

To summarize:

$$\boxed{\text{Observe} \rightarrow \text{Extract} \rightarrow \text{Validate} \rightarrow L_{\text{process}} \rightarrow \text{Train}}$$

Current Main Decisions

Decompositional Rule Extraction

Focus on internal model behaviour, not only input-output imitation.

Reason: the thesis is about understanding and evaluating the model's internal reasoning process.

Architecture Grouping

Group architectures by how much logical/domain knowledge is supplied beforehand versus learned from examples.

Current Main Decisions

Architecture Grouping

More learned from examples: Eg. Neural Logic Machines
Relational structure is provided by the architecture, while useful reasoning patterns are learned during training.

More logical knowledge supplied: Eg. Logic Tensor Networks
Logical formulas can be provided directly and their satisfaction can influence training.

Current Main Decisions

Priority Architectures

Prioritize architectures where more of the reasoning structure is learned or induced from examples.

Reason: the thesis aims to recover and evaluate reasoning learned by the model itself, rather than mainly reasoning that was specified beforehand.

Architectures that already use supplied logical formulas during training are still useful for comparison, but are less central to this question.

Expected Contributions: If Our Approach Works

1. **Identify suitable neuro-symbolic architectures**
   Analyze which architectures expose internal representations that can be meaningfully extracted.

2. **Develop a method for extracting intermediate reasoning**
   Recover symbolic rules describing how internal representations are transformed.

3. **Provide symbolic descriptions of learned reasoning**
   Make parts of the model's internal reasoning process inspectable.

4. **Evaluate extraction fidelity**
   Test whether the extracted rules actually reproduce the model's internal behaviour, including on unseen examples.

Expected Contributions: If Our Approach Works

5. **Develop criteria for reasoning-quality evaluation**
   For example: legality, consistency, coherence, relevance, and agreement with known rules where available.

6. **Develop a process-based training approach**
   Convert reasoning-quality evaluation into a process loss and train end-to-end using that signal.

7. **Evaluate the effect on reasoning and generalization**
   Test whether the approach improves reasoning quality, task performance, interpretability, and generalization.

Risks / Failure Cases / Potential Difficulties

1. **No compact symbolic explanation**
   Some internal representations may be too distributed or complex to describe with clear or simple rules.
2. **Insufficient extraction fidelity**
   Extracted rules may not accurately reproduce the model's actual internal behaviour.
3. **Poor generalization of extracted rules**
   Rules may fit observed activations but fail to remain faithful on unseen examples.
4. **Architecture dependence**
   Different neuro-symbolic architectures may require entirely different extraction methods.

Risks / Failure Cases / Potential Difficulties

5. **Difficult reasoning-quality evaluation**
   Criteria such as legality, consistency, coherence, and relevance may be difficult to define and measure reliably.

6. **Validation may not produce a useful process loss**
   A reasoning-quality score may not translate effectively into a training signal.

7. **High computational cost**
   Repeated extraction and validation across many layers, internal features, or training iterations may be expensive.

8. **Process-only training may fail**
   The process signal alone may not be sufficient to learn useful task behaviour.

Risks / Failure Cases / Potential Difficulties

9. **Negative results are still informative**
   Failures will be analyzed and reported, including which part of the pipeline failed and why.

Any questions?



References i

Robert Andrews, Joachim Diederich, and Alan B. Tickle.

Survey and critique of techniques for extracting rules from trained artificial neural networks.

Knowledge-Based Systems, 8(6):373–389, 1995.

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger.

Logic tensor networks.

Artificial Intelligence, 303, 2022.

Mark W. Craven and Jude W. Shavlik.

Extracting tree-structured representations of trained networks.

In *Advances in Neural Information Processing Systems 8*, 1995.

References ii

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou.
Neural logic machines.
In *International Conference on Learning Representations*, 2019.

Mateo Espinosa Zarlenga, Zohreh Shams, and Mateja Jamnik.
Efficient decompositional rule extraction for deep neural networks.
arXiv preprint arXiv:2111.12628, 2021.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann.
Shortcut learning in deep neural networks.
Nature Machine Intelligence, 2:665–673, 2020.

References iii



Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe.

Let's verify step by step.

arXiv preprint arXiv:2305.20050, 2023.



Alan B. Tickle, Marian Orlowski, and Joachim Diederich.

DEDEC: Decision detection by rule extraction from neural networks.

Technical report, Neurocomputing Research Centre, Queensland University of Technology, 1994.