



MONASH University

Formal Explainability for Artificial Intelligence in Dynamic Environments

Jaime Cuartas Granada

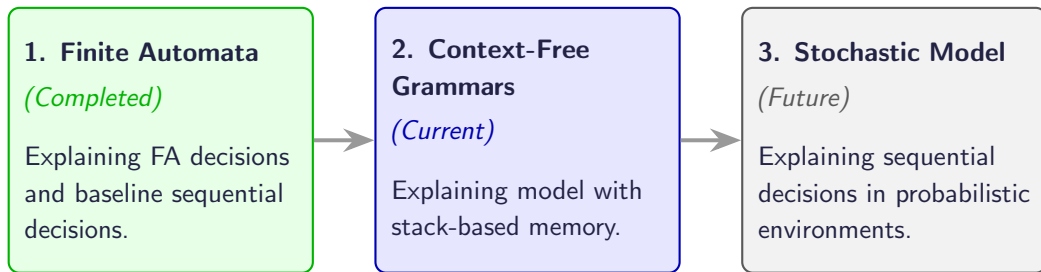
Supervisors: Alexey Ignatiev, Peter J. Stuckey, Julian Gutierrez

26-February-2026

Department of Data Science and Artificial Intelligence (DSAI), Monash University, Australia

Project Summary

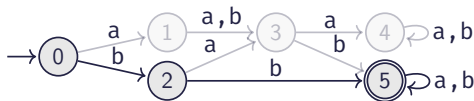
Goal: Deliver explanations for sequential decision-making models.



Completed Work: Finite Automata

Explaining Finite Automata (Completed)

- A Finite Automaton (FA) is a mapping from an input $w \in \Sigma^*$ to a class $\mathcal{K} = \{\text{Accept}, \text{Reject}\}$.



Input w :

b	b	b	b	b
---	---	---	---	---

 \rightarrow Accept

AXp 1:

b	b	Σ	Σ	Σ
---	---	----------	----------	----------

 \rightarrow Guarantees Accept $L(bb\Sigma\Sigma\Sigma) \subseteq L(\mathcal{A})$

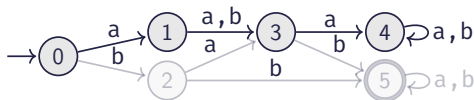
AXp 2:

Σ	Σ	b	Σ	Σ
----------	----------	---	----------	----------

 \rightarrow Guarantees Accept $L(\Sigma\Sigma b\Sigma\Sigma) \subseteq L(\mathcal{A})$

Explaining Finite Automata (Completed)

- A Finite Automaton (FA) is a mapping from an input $w \in \Sigma^*$ to a class $\mathcal{K} = \{\text{Accept}, \text{Reject}\}$.



Input w :

b	b	b	b	b
---	---	---	---	---

 \rightarrow Accept

CXp:

Σ	b	Σ	b	b
----------	---	----------	---	---

 \rightarrow Enables Reject $L(\Sigma b \Sigma b b) \not\subseteq L(\mathcal{A})$

a	b	a	b	b
---	---	---	---	---

Why Finite Automata Are Not Enough

- $L = \{a^n b^n \mid n \geq 1\}$ cannot be described by a FA.
- For a rejected word like aaaabb, standard parsers identify the error at the end.

Source Code:

```
1 int main(){
2     for(int i=0; i<10; i++){ // Error
3         printf("hello");
4     }
5 }
```

Parser Output:

```
error: expected '}' at end of input
5 | }
  | ^
```

Current Research: Explaining CFGs

Fragility of Acceptance vs. Robustness of Rejection

In Context-Free Languages, consider the balanced parentheses language $L = \{(), (()), ()(), \dots\}$:

Accepted Words are Fragile:

- $w = ()()$
-))((), (((), ())), (()((,
- ()()

Rejected Words are Robust:

- $w =))))$
- (()), ()()
-)))),))))

Challenge: How do we formally extract these explanations?

Methodology: The Modified CYK Algorithm

- **The Goal:** Verify if a candidate set S is a valid CXp.

Example: Rejected word $w =))))$. Test candidate $S = \{1, 2\}$.

$B \rightarrow \mathbf{L R}$ (R1)

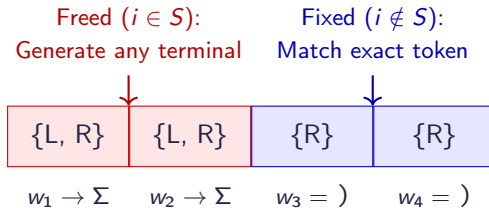
$B \rightarrow \mathbf{L R B}$ (R2)

$B \rightarrow \mathbf{L B R}$ (R3)

$B \rightarrow \mathbf{L B R B}$ (R4)

$\mathbf{L} \rightarrow ($ (U1)

$\mathbf{R} \rightarrow)$ (U2)



Now the first two indices can act as (via variable \mathbf{L} .

- **The Limitation of Standard CFGs:** Our algorithm successfully extracts minimal CXps, but it treats all valid corrections equally.

Example: The rejected word $w = \text{)}}))$ yields two valid minimal corrections:

Correction A: $(())$

Nested structure

Correction B: $()()$

Sequential structure

Priorities with PCFGs

- By training a **Probabilistic CFG (PCFG)** on real-world datasets, we can assign probabilities to rules to calculate the *most likely* explanation.

Learned PCFG Distributions:

$B \rightarrow \mathbf{L R}$ (44.4%)

$B \rightarrow \mathbf{L R B}$ (33.3%)

$B \rightarrow \mathbf{L B R}$ (11.1%)

$B \rightarrow \mathbf{L B R B}$ (11.1%)

$\mathbf{L} \rightarrow ($ (100%)

$\mathbf{R} \rightarrow)$ (100%)

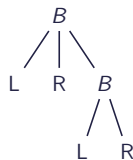
Correction A: **(())**



$$0.\bar{1} \times 0.\bar{4} \approx 4.9\%$$

Likelihood

Correction B: **(())**



$$0.\bar{3} \times 0.\bar{4} \approx 14.8\%$$

Likelihood

Abductive Explanations (Errors)

$w = \text{))})$

))),))

The Open Question:

How can we prioritise **Abductive Explanations**?

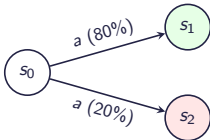
Is a structural failure at index 1 probabilistically “worse” than a failure at indices 2 and 3?

Future Work: Explaining Stochastic Environments

- PCFGs introduce probabilities to static sequences. The ultimate goal of this thesis is to scale these formalisms to explain AI in **dynamic, stochastic environments**.

What is a Stochastic Environment?

Systems (like MDPs) where executing an action a yields a probability distribution over future states.



What does an explanation look like?

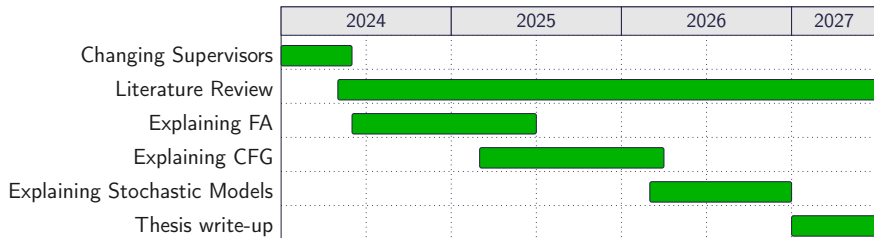
Explanations shift from absolute logical guarantees to probabilistic bounds.

The Open Questions:

- How do we define minimality for AXps/CXps when trajectories are infinite and outcomes are uncertain?

Project Management

Project Plan & Timeline

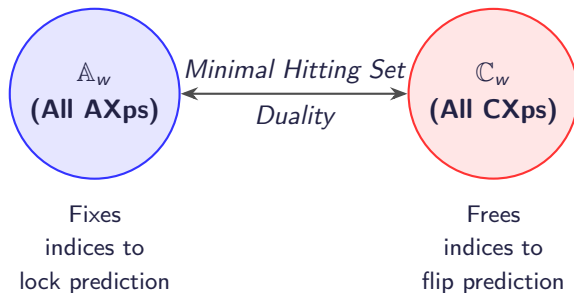


Thank You

Questions?

The Minimal Hitting Set Duality

- **The Core Relationship:** Abductive and contrastive explanations share a duality [?].
- Every AXp is a minimal hitting set of the complete set of CXps, and vice versa.
- To flip a prediction (CXp), you must free at least one token from every reason that guarantees the current prediction (AXp).



- **Algorithmic Contribution:**

- Leveraged this duality to develop algorithms for the formal **enumeration** of explanations in Finite Automata.
- Successfully maps the abstract concepts of XAI onto formal language.

Phase 1 Milestone Achieved

Status: Completed.

Output: The formal definitions, duality proofs, and enumeration algorithms have been compiled and submitted to **ICALP 2026**.

Why $L = \{a^n b^n \mid n \geq 1\}$ cannot be described by a FA?

Intuition: Finite Automata cannot count arbitrarily high because they have no external memory (like a stack).

Pumping Lemma for Regular Languages: For any regular language, there is a “pumping length” p . If a string in the language is longer than p , the FA will get into a loop.

If we assume $L = \{a^n b^n \mid n \geq 1\}$ is regular, we could take a word $a^p b^p$. The FA must loop while reading the a's. This means we could "pump" (repeat) that loop of a's, generating a word like $a^{p+k} b^p$.

The FA would still accept this new word because it uses the exact same path to reach the accepting state, but $a^{p+k} b^p$ breaks the $n = n$ rule.

It leads to a contradiction, proving L is not regular and cannot be described by an FA.

Methodology: Parsing a word

- The CYK algorithm requires the grammar to be in **Chomsky Normal Form (CNF)**.
- Every rule is either $A \rightarrow BC$ (two variables) or $A \rightarrow a$ (one terminal).

Original Grammar (G):

$$B \rightarrow (B) B$$

$$B \rightarrow (B)$$

$$B \rightarrow () B$$

$$B \rightarrow ()$$

Converted CNF (G'):

$$B \rightarrow NB \quad N \rightarrow UR \quad L \rightarrow ($$

$$B \rightarrow UR \quad U \rightarrow LB \quad R \rightarrow)$$

$$B \rightarrow PB \quad P \rightarrow LR$$

$$B \rightarrow LR$$

Note: We introduce variables L and R for the terminals, and helper variables (N, U, P) to break down rules longer than two symbols.

The CYK Algorithm: Bottom-Up Parsing

Input Word: $w = (())$

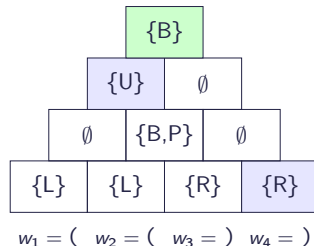
CNF Grammar:

$B \rightarrow NB \mid UR \mid PB \mid LR$

$N \rightarrow UR \quad U \rightarrow LB \quad P \rightarrow LR$

$L \rightarrow ($

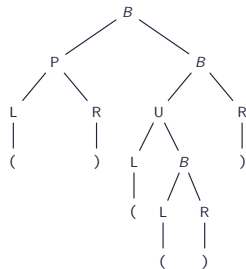
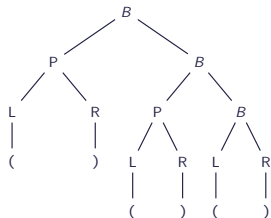
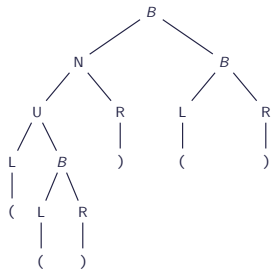
$R \rightarrow)$



Step 1: Initialise the base row using terminal rules ($L \rightarrow ($, $R \rightarrow)$).

Step 2: Build upward. E.g., cell $\{B\}$ is formed because $B \rightarrow UR$

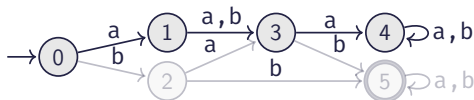
Probabilistic Resolution of Ambiguity



Rule	Count	Probability (P)
$B \rightarrow L R$	4	$4/9 = 44.\bar{4}\%$
$B \rightarrow P B$	3	$3/9 = 33.\bar{3}\%$
$B \rightarrow N B$	1	$1/9 = 11.\bar{1}\%$
$B \rightarrow U R$	1	$1/9 = 11.\bar{1}\%$

Explaining Finite Automata (Completed)

- A Finite Automaton (FA) is a mapping from an input $w \in \Sigma^*$ to a class $\mathcal{K} = \{\text{Accept}, \text{Reject}\}$.



Input w :

b	b	b	b	b
---	---	---	---	---

 \rightarrow Accept

CXp 1:

Σ	b	Σ	b	b
----------	---	----------	---	---

 \rightarrow Enables Reject $L(\Sigma b \Sigma b b) \not\subseteq L(\mathcal{A})$

CXp 2:

b	Σ	Σ	b	b
---	----------	----------	---	---

 \rightarrow Enables Reject $L(b \Sigma \Sigma b b) \not\subseteq L(\mathcal{A})$