MONASH University

Formal Explainability for Artificial Intelligence in Dynamic Environments

Jaime Cuartas Granada

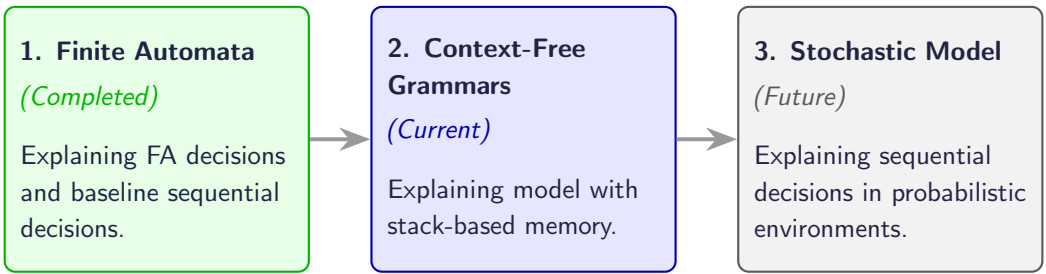
26-February-2026

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

Project Summary & Refinements

Summary

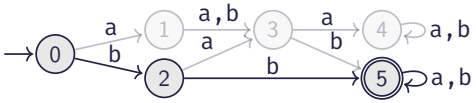
Goal: Deliver explanations for sequential decision-making models.



Completed Work: Finite Automata

Explaining Finite Automata (Completed)

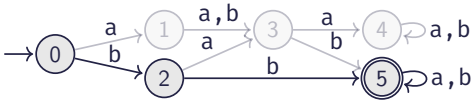
- FA is a mapping from an input $w \in \Sigma^*$ to a class $\mathcal{K} = \{\text{Accept}, \text{Reject}\}$.



Input w**:**	b b b b b	$\rightarrow$ Accept
AXp 1:	b b Σ Σ Σ	$\rightarrow$ Guarantees Accept $L(\text{bb}\Sigma\Sigma\Sigma) \subseteq L(\mathcal{A})$
AXp 2:	Σ Σ b Σ Σ	$\rightarrow$ Guarantees Accept $L(\Sigma\Sigma\text{b}\Sigma\Sigma) \subseteq L(\mathcal{A})$

Explaining Finite Automata (Completed)

- 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 $L(\Sigma\texttt{b}\Sigma\texttt{bb}) \not\subseteq L(\mathcal{A})$
CXp 2:	b Σ Σ b b	$\rightarrow$ Enables Reject $L(\texttt{b}\Sigma\Sigma\texttt{bb}) \not\subseteq L(\mathcal{A})$

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 (like balanced parentheses):

Accepted Words are Fragile:

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

Rejected Words are Robust:

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

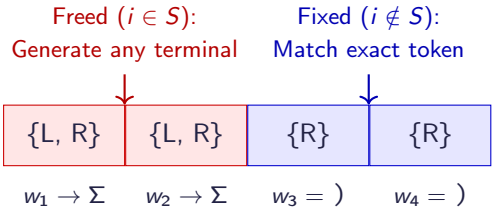
Challenge: How do we formally extract these explanations?

The Modified CYK Approach

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

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

$$B \to \mathbf{L}\, B\, \mathbf{R}\, B \quad \text{(R1)}$$
$$B \to \mathbf{L}\, B\, \mathbf{R} \quad \text{(R2)}$$
$$B \to \mathbf{L}\, \mathbf{R}\, B \quad \text{(R3)}$$
$$B \to \mathbf{L}\, \mathbf{R} \quad \text{(R4)}$$
$$\mathbf{L} \to (\quad \text{(U1)}$$
$$\mathbf{R} \to) \quad \text{(U2)}$$



Result: Now the first two indices can act as (via variable **L**.

The Ambiguity Problem & Phase 3 Transition

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

Example: The rejected word $w =))))$, yields two valid minimal corrections:

Correction A: (())

Nested structure

Correction B: ()()

Sequential structure

- By transitioning to **Probabilistic Context-Free Grammars (PCFGs)** trained on real-world datasets, we can assign likelihoods to grammar rules.

Phase 3: Deducing Responsibility 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:

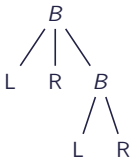
$B \to \mathrm{L}\,B\,\mathrm{R}\,B$	(11.1%)
$B \to \mathrm{L}\,B\,\mathrm{R}$	(11.1%)
$B \to \mathrm{L}\,\mathrm{R}\,B$	(33.3%)
$B \to \mathrm{L}\,\mathrm{R}$	(44.4%)
$\mathrm{L} \to ($	(100%)
$\mathrm{R} \to)$	(100%)

Correction A: (())

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

Likelihood

Correction B: ()()



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

Likelihood

Conclusion: The algorithm prioritizes (()) as a better correction for))))).

Feature Attribution: The Ambiguity Problem

Abductive Explanations (Errors)

$w =))))$

$)))),))))$

Future Work:

If $()()$ is a more common structure than $(())$. We can weigh these contrastive explanations using **Probabilistic CFGs (PCFGs)**.

The Open Question: How can we prioritize AXps?

Methodology: The Modified CYK Algorithm

- To verify if a set of indices S is a valid CXp, we must check if replacing those tokens with wildcards (Σ) allows the word to be accepted.
- We achieve this by modifying the base case of the **CYK Parsing Algorithm**:

Modified Base Case Initialization

For each token w_i at index i:

- **If $i \notin S$ (Fixed):** $T[i, i] = \{A \in V \mid A \to w_i\}$
- **If $i \in S$ (Freed/Wildcard):** $T[i, i] = \{A \in V \mid \exists \alpha \in \Sigma, A \to \alpha\}$

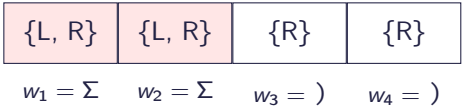
Result: If the Start symbol S appears at the top of the table $T[1, n]$, then the candidate set S is a valid CXp.

Visualizing Extraction: Greedy Deletion

Algorithm Insight: We start by freeing the *entire* word, then greedily lock tokens back into place one by one. If locking a token breaks the parsing, it belongs in our minimal CXp.

Example: $w =$))))), checking CXp candidate $S = \{1, 2\}$

Wildcards Fixed indices lock to $R \to)$



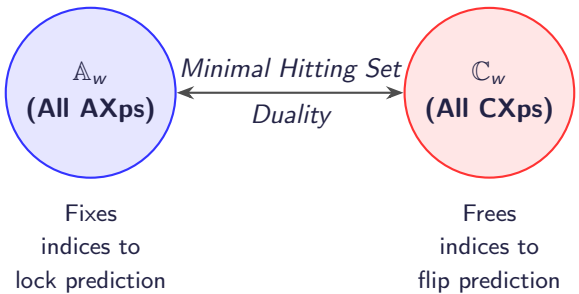
Because indices 1 and 2 can now act as L ($L \to ($), the CYK table successfully resolves to the Start variable, proving $\{1, 2\}$ is a valid explanation.

Current Status: CFG Explanations

- **Theoretical Foundation:**
  - Formalized the extension of AXps/CXps to Context-Free Languages.
  - Proved the correctness of the Modified CYK approach for wildcard evaluation.
- **Algorithmic Implementation:**
  - Currently implementing the greedy extraction algorithm (ExtractCXp).
  - Testing extraction performance on standard balanced parenthesis (Dyck) languages.
- **Next Immediate Step:** Benchmarking the computational complexity of the CYK-based extraction and drafting the manuscript for this second phase of the thesis.

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).



Outcomes: Enumeration & Milestone

- **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 rigorous formal language properties.

Phase 1 Milestone Achieved

Status: Completed.

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

- *Transitioning to Phase 2:* With the baseline for regular languages established, we now scale the complexity to languages requiring memory.

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 $A \to BC$ (two variables) or $A \to a$ (one terminal).

Original Grammar (G):

$$B \to (\ B\ )\ B$$
$$B \to (\ B\ )$$
$$B \to (\ )\ B$$
$$B \to (\ )$$

Converted CNF (G'):

$$B \to N\,B \quad N \to U\,R \quad L \to ($$
$$B \to U\,R \quad U \to L\,B \quad R \to)$$
$$B \to P\,B \quad P \to L\,R$$
$$B \to L\,R$$

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: $w = (\,(\,)\,)$

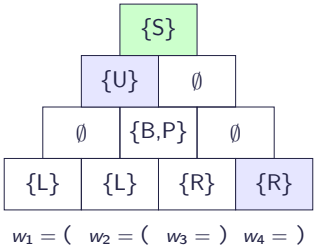
CNF Grammar:

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

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

$L \to ($

$R \to)$

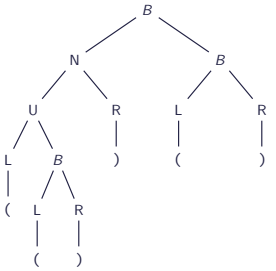


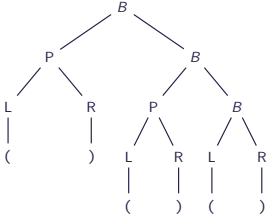
Step 1: Initialize the base row using terminal rules ($L \to ($, $R \to)$).

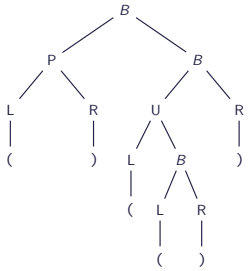
Step 2: Build upward. E.g., cell {X} is formed because $X \to SR$, matching {S} and {R} below it.

Step 3: The word is accepted because the Start variable S appears in the top cell!

Probabilistic Resolution of Ambiguity







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