# Neural Network and Temporal Logic

罗炜麟

中山大学

2021-7-22 烟台

# Research Background

- Linear Temporal Logic (LTL)
  - temporal modal operators, e.g., next ($X$) and until ($U$),
  - formal verification, program synthesis, etc.
- Satisfiability Problem of LTL (LTL SAT)
  - answer SAT or UNSAT for a given LTL Boolean formula
  - PSPACE-complete
  - avoid common errors in LTL assertions, identify boundary conditions, etc.

# Research Background

1. Developing  neural approaches for LTL SAT over different paradigms

2. Exploring the relationship between the expressive ability of the neural network and temporal logic

# Enhancing Neural Temporal Reasoning with Logical Proof

# Motivation

- Make neural networks to tackle hard computational problems
  - propositional satisfiability problem[1]
  - combinatorial optimization problem[2]
  - approximate model counting[3]
- Not competitive with SOTA approaches
- Show some potential
  - Accuracy is considerably higher than random guessing.

- **Whether LTL SAT can be tackled by neural networks?**

# Motivation

- Neural Temporal Reasoning (NTR)
  - predict a trace satisfying a given satisfiable LTL formula

| Input | Output |
|---|---|
| $a\,\mathcal{U} \bigcirc b$ | $\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^{\omega}$ |

- Challenge: capture the semantics of LTL
  - is interpreted recursively
  - characterizes the features of sequences

# Motivation

- Transformer model has a good performance[4]
    - successfully capture the features of sequences
- Drawback
    - only focuses on recognizing the pattern of the combinations of the syntactic of formula and satisfying traces

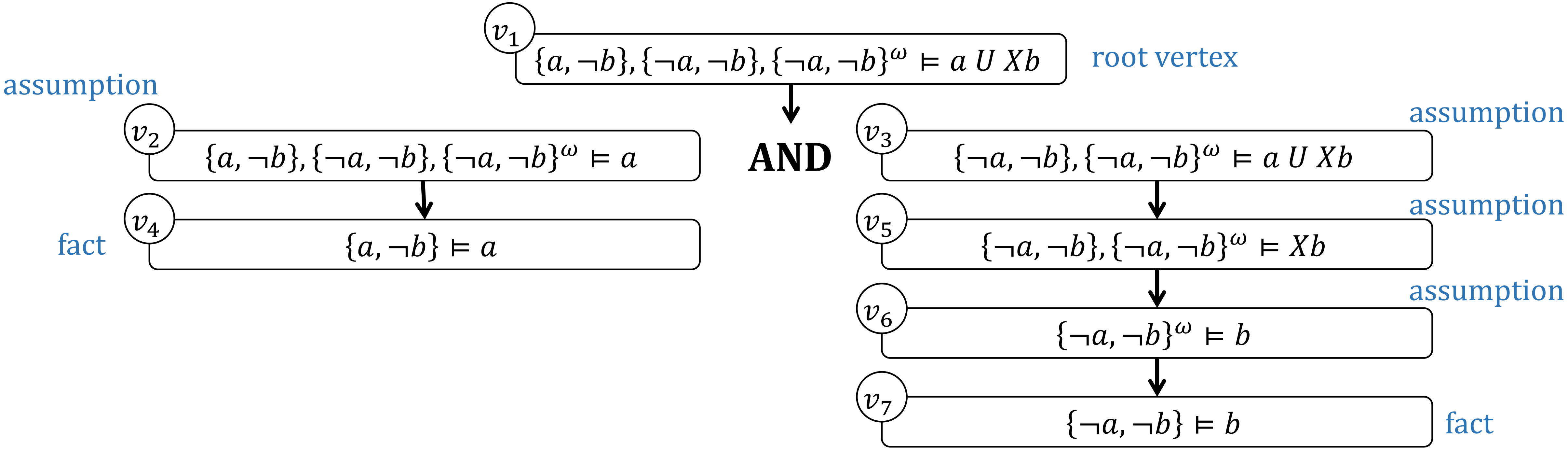
- **How to enhance neural networks temporal reasoning?**

# Proof of LTL

- Trace-formula Relation Tuple
  - (trace, satisfaction relation, LTL formula)
- Proof of LTL
  - is explicit reasoning step by step based on the semantics of LTL

# Proof of LTL

- Proof of LTL
  - Example: $\{a, \neg b\}, \{\neg a, \neg b\}, \{\neg a, \neg b\}^{\omega}$ and $a\ U\ Xb$



# Proof of LTL

- Existing works only learn the relationship between the formula and the trace.
  - lack learning the relationship between sub-formulae and sub-traces in the proof

- We suggest to enhance the neural networks for temporal reasoning by explicitly incorporating the proof that the trace satisfies the for mula into neural architectures.

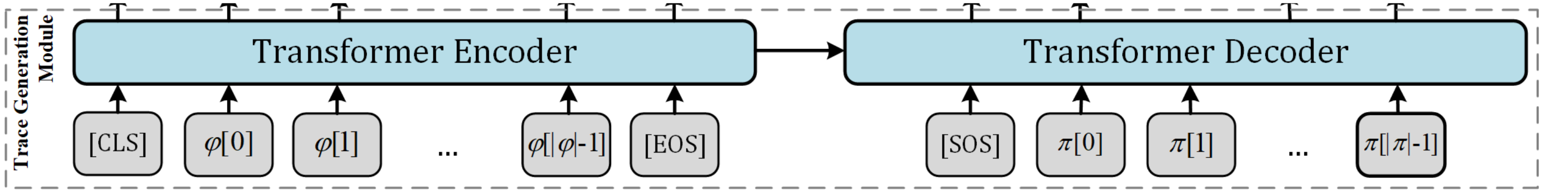
# Joint Trace Prediction and Proof Generation Model

- LTSatP
  - jointly makes temporal reasoning and generates a entire proof at one time

| Model | Input | Output |
|---|---|---|
| Finkbeiner *et al.* [2021] | $a\,\mathcal{U} \bigcirc b$ | $\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega$ |
| `LTSatP` | $a\,\mathcal{U} \bigcirc b$ | $\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega$<br>$(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a\,\mathcal{U} \bigcirc b)$<br>$(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a)$<br>$(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a\,\mathcal{U} \bigcirc b)$<br>$(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, \bigcirc b)$<br>$((\{a, b\})^\omega, \models, b)$ |

# Joint Trace Prediction and Proof Generation Model

- Architecture of LTSatP



# Iterative Proof Generation Model

- LTSatP-ite
  - given an assumption, predicts facts or assumptions supporting the input a ssumption

<table>
<tr><th>Model</th><th>Input</th><th>Output</th></tr>
<tr><td>Finkbeiner et al. [2021]</td><td>$a \,\mathcal{U} \bigcirc b$</td><td>$\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega$</td></tr>
<tr><td><code>LTSatP</code></td><td>$a \,\mathcal{U} \bigcirc b$</td><td>$\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega$<br>$(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a \,\mathcal{U} \bigcirc b)$<br>$(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a)$<br>$(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a \,\mathcal{U} \bigcirc b)$<br>$(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, \bigcirc b)$<br>$((\{a, b\})^\omega, \models, b)$</td></tr>
<tr><td><code>LTSatP-ite</code></td><td>0: $(?, \models, a \,\mathcal{U} \bigcirc b)$<br>1: $(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a \,\mathcal{U} \bigcirc b)$<br><br>2: $(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a \,\mathcal{U} \bigcirc b)$<br>3: $(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, \bigcirc b)$</td><td>$(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a \,\mathcal{U} \bigcirc b)$<br>$(\{a, \neg b\}\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a)$ and<br>$(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, a \,\mathcal{U} \bigcirc b)$<br>$(\{\neg a, \neg b\}(\{a, b\})^\omega, \models, \bigcirc b)$<br>$((\{a, b\})^\omega, \models, b)$</td></tr>
</table>

# Experiments and Analysis

- NTR evaluation
  - 80w/10w/10w
  - train in [5,20)
  - test in [5,20)

| Model | Syntactic | Semantic | Total |
|---|---|---|---|
| Transformer[4] | 0.43 | 0.52 | 0.95 |
| LTSatP | **0.66** | 0.33 | **0.99** |
| LTSatP-ite | 0.65 | 0.34 | 0.98 |

# Experiments and Analysis

- Generalizability evaluation
  - 80w/10w/10w
  - train in [5,20)
  - test in [20,35), [35,50)
  - test in another random formulae dataset, denoted as **RF**[5]
  - test in pattern formulae dataset, denoted as **PF**[5]

# Experiments and Analysis

- Generalizability evaluation

| Model | [20,35) | | | [35,50) | | |
|---|---|---|---|---|---|---|
| | Syntactic | Semantic | Total | Syntactic | Semantic | Total |
| Transformer[4] | 0.08 | 0.62 | 0.70 | 0.01 | 0.60 | 0.61 |
| LTSatP | 0.14 | 0.58 | 0.72 | **0.02** | 0.60 | **0.62** |
| LTSatP-ite | **0.20** | 0.61 | **0.80** | **0.02** | 0.59 | 0.61 |

| Model | RF | | | PF | | |
|---|---|---|---|---|---|---|
| | Syntactic | Semantic | Total | Syntactic | Semantic | Total |
| Transformer[4] | 0.32 | 0.54 | 0.86 | 0.08 | 0.92 | 0.99 |
| LTSatP | **0.57** | 0.39 | **0.96** | 0.06 | 0.94 | 0.99 |
| LTSatP-ite | 0.51 | 0.36 | 0.88 | **0.10** | 0.89 | 0.99 |

# Experiments and Analysis

- Varying training data size
  - 8w/10w/10w
  - train in [5,20)
  - test in [5,20), [20,35), [35,50)
  - test in another random formulae dataset, denoted as **RF**[5]
  - test in pattern formulae dataset, denoted as **PF**[5]

# Experiments and Analysis

- Varying training data size

| Model | [5,20) | | | [20,35) | | | [35,50) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Syntactic | Semantic | Total | Syntactic | Semantic | Total | Syntactic | Semantic | Total |
| Transformer[4] | 0.30 | 0.58 | 0.88 | 0.04 | 0.61 | 0.65 | 0.00 | 0.58 | 0.58 |
| LTSatP | **0.53** | 0.43 | **0.96** | 0.10 | 0.59 | 0.69 | **0.02** | 0.58 | 0.60 |
| LTSatP-ite | **0.53** | 0.41 | 0.94 | **0.13** | 0.62 | **0.75** | **0.02** | 0.59 | **0.61** |

| Model | RF | | | PF | | |
|---|---|---|---|---|---|---|
| | Syntactic | Semantic | Total | Syntactic | Semantic | Total |
| Transformer[4] | 0.27 | 0.49 | 0.76 | **0.02** | 0.82 | 0.84 |
| LTSatP | **0.46** | 0.45 | **0.91** | 0.01 | 0.84 | **0.85** |
| LTSatP-ite | 0.40 | 0.40 | 0.80 | 0.01 | 0.71 | 0.72 |

# Conclusion

1. Utilizing proof in predicting a satisfying trace can indeed improve the predictive ability and generalization ability of the model.

2. However, we believe that the neural network still cannot understand the semantics of temporal operators.

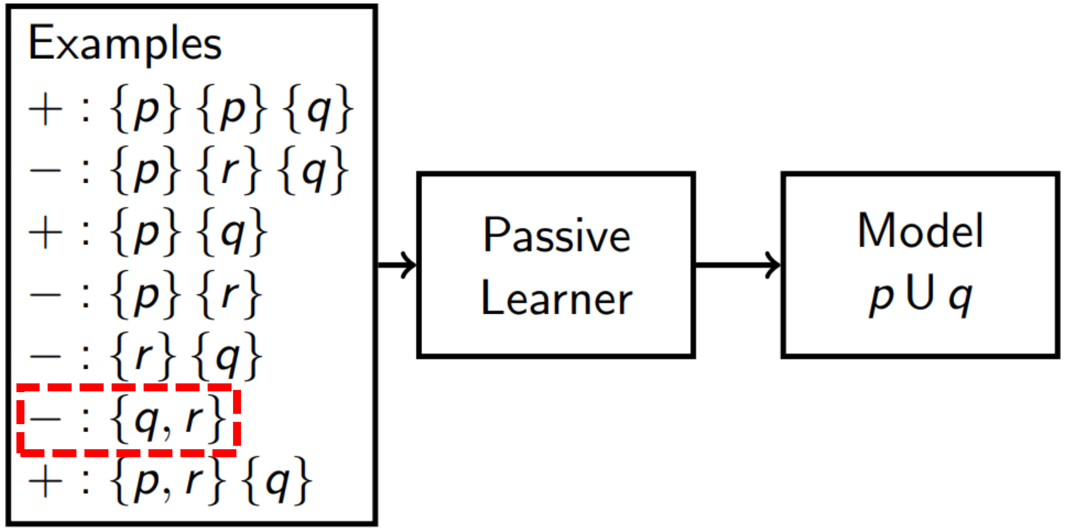
# LTLf Classifier for Path Can Be Expressed by GNN

# Motivation

- Expressiveness of GNN
  - Weisfeiler-Lehman (WL) test[6,7]
  - FOC2 classifier[8]
- We proof that GNNs can express the LTLf classifier on the directed path graph.
- This result provides a new solution to the problem of LTLf learning.
  - suitable for large-scale data
  - utilize GPU to improve performance

# Motivation

- LTLf Learning
  - learning LTLf formulae that characterize the high-level behavior of a system based on observation traces in planning
  - Input:
    - $E^+ = \{e_1^+, \dots, e_{m_1}^+\}$
    - $E^- = \{e_1^-, \dots, e_{m_2}^-\}$
  - Output:
    - An LTLf formula $\phi$

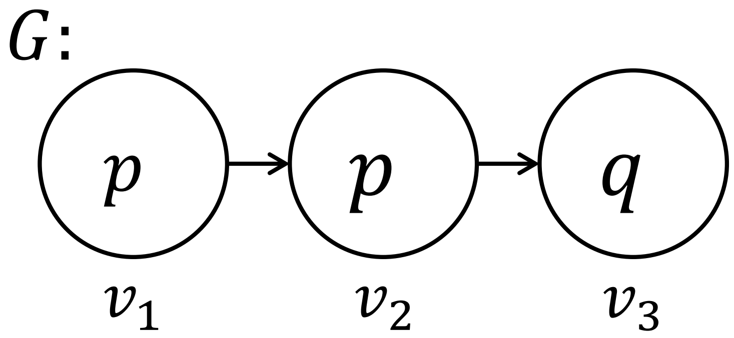


# Motivation

- No methods to learn template-free LTLf formulae from noise data
  - alternating finite automata (AFA) + SAT[9]
  - pattern formula + Bayesian inference[10]
- Learning template-free formulae and handling noise data is of value for the practical applications
  - more perfect characterization
  - the traces usually contain noise
- Challenge
  - large search space for template-free formulae
  - negative influence of noisy data for search bias

# Relationship between LTLf and GNN

- **Definition 1.** Given a directed path graph $G$, a node $v$ of $G$, and an LTLf Boolean formula $\phi$, iff the path $\pi$ starting from $v$ to the end $v_e$ of $G$ such that $\mathrm{L}(\pi) \vDash \phi$, then $v$ *satisfies* $\phi$ denoted by $(G, v) \vDash_L \phi$.




- $\pi = v_1, v_2, v_3, \mathrm{L}(\pi) = \{p\}\{p\}\{q\}$
  - $\{p\}\{p\}\{q\} \vDash pUq$, therefore $(G, v_1) \vDash_L pUq$
  - $\{p\}\{p\}\{q\} \vDash p$, therefore $(G, v_1) \vDash_L p$
- $\pi = v_2, v_3, \mathrm{L}(\pi) = \{p\}\{q\}$
  - $\{p\}\{q\} \vDash pUq$, therefore $(G, v_2) \vDash_L pUq$

# Relationship between LTLf and GNN

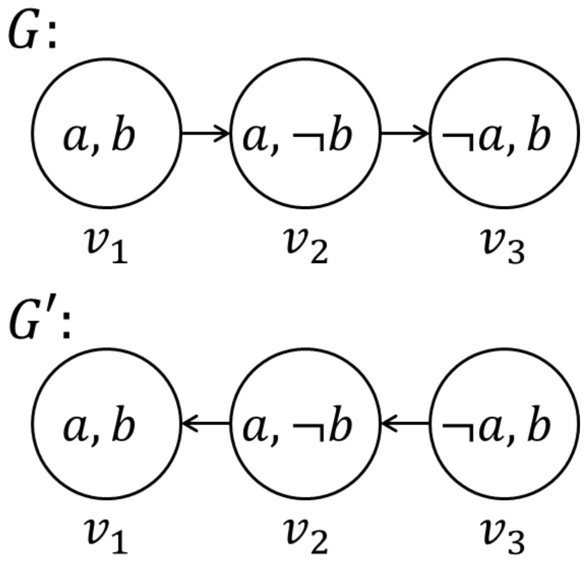
- **Definition 2.** A GNN classifier $\mathcal{A}$ *captures* a LTLf classifier $\phi$ for path, if for every directed path graph $G$ and node $v$ of $G$, it holds that $\mathcal{A}(G, v) = true$ iff $(G, v) \vDash_L \phi$.

# Relationship between LTLf and GNN

- **Proposition 1.** Each LTLf classifier for path is captured by a simple homogeneous aggregate-combine GNN (AC-GNN).
  - AC-GNN
    - $\boldsymbol{x}_v^{(i)} = \mathrm{COM}^{(i)}(\boldsymbol{x}_v^{(i-1)}, \mathrm{ACC}^{(i)}(*))$
  - simple and homogeneous
    - $\mathrm{COM}^{(i)}(\boldsymbol{x}_1, \boldsymbol{x}_2) = f(\boldsymbol{x}_1 \boldsymbol{C} + \boldsymbol{x}_2 \boldsymbol{A} + \boldsymbol{b})$

# Relationship between LTLf and GNN

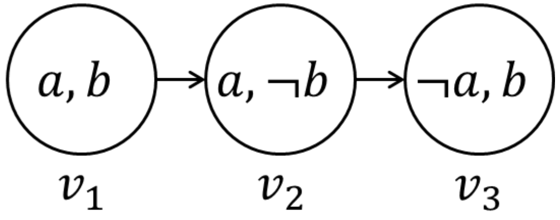
- Given $\phi = a\ U\ Xb$, for every directed path graph, e.g., $G$. We get the AC-GNN $\mathcal{A}(\boldsymbol{C}, \boldsymbol{A}, \boldsymbol{b})$ capturing $\phi$ as follows:
  - Let $\phi_1 = a\ U\ Xb$, $\phi_2 = Xb$, $\phi_3 = a$, and $\phi_4 = b$.
  - The neighbors of $\mathcal{A}$ is $G'$.
  - Embedding of node: $\boldsymbol{x}_v^{(i)} = [x_{\phi_1}, x_{\phi_2}, x_{\phi_3}, x_{\phi_4}]$
    - $\boldsymbol{x}_{v_1}^{(0)} = [0,0,1,1]$, $\boldsymbol{x}_{v_2}^{(0)} = [0,0,1,0]$, $\boldsymbol{x}_{v_2}^{(0)} = [0,0,0,1]$
  - single iteration
    - $\boldsymbol{x}_v^{(i)} = f(\boldsymbol{x}_v^{(i-1)}\boldsymbol{C} + \sum_{u \in \mathcal{N}(v)} \boldsymbol{x}_u^{(i-1)} \boldsymbol{A} + \boldsymbol{b})$
    - $f(x) = \min(\max(0, x), 1)$
    - $\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$, $\boldsymbol{b} = [-1 \quad 0 \quad 0 \quad 0]$



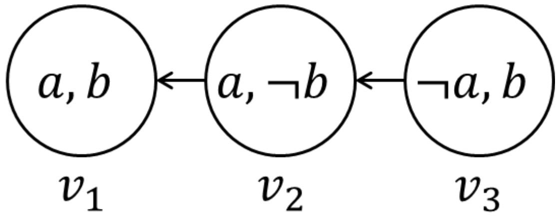
# Relationship between LTLf and GNN

- Given $\phi = a\ U\ Xb$, for every directed path graph, e.g., $G$. We get the AC-GNN $\mathcal{A}(\boldsymbol{C}, \boldsymbol{A}, \boldsymbol{b})$ capturing $\phi$ as follows:
  - Let $\phi_1 = a\ U\ Xb$, $\phi_2 = Xb$, $\phi_3 = a$, and $\phi_4 = b$.
  - single iteration
    - $\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} -1 & 0 & 0 & 0 \end{bmatrix}$
    - If $f_i \in \mathbb{P}$, then $\boldsymbol{C}_{ji} = 1$;
    - If $f_i = Xf_j$, then $\boldsymbol{A}_{ji} = 1$;
    - If $f_i = f_j\ U\ f_k$, i.e., $f_i = f_k \vee (f_j \wedge Xf_i)$,
      - then $\boldsymbol{C}_{ki} = 2$, $\boldsymbol{C}_{ji} = 1$, $\boldsymbol{A}_{ii} = 1$, and $\boldsymbol{b}_i = -1$;

$G$:




$G'$:



# GNN-based LTLf Learning

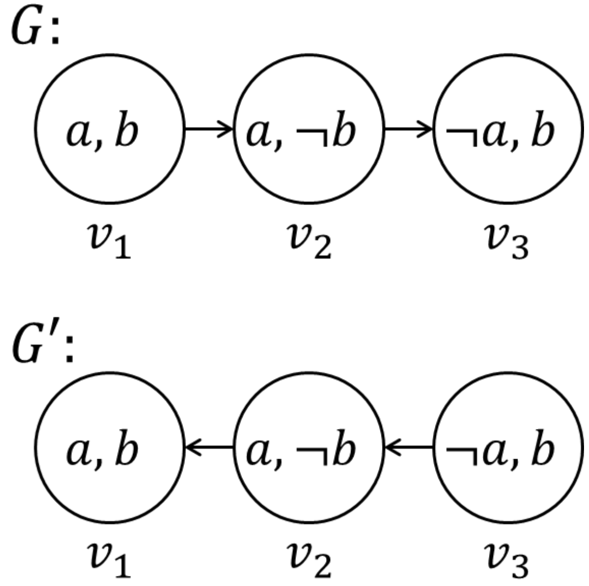
- Overview

$$
\begin{aligned}
+ &: \{p\}\,\{p\}\,\{q\} \\
- &: \{p\}\,\{r\}\,\{q\} \\
+ &: \{p\}\,\{q\} \\
- &: \{p\}\,\{r\} \\
- &: \{r\}\,\{q\} \\
- &: \{q, r\} \\
+ &: \{p, r\}\,\{q\}
\end{aligned}
$$

$\Rightarrow$ GNN Classifier $\Rightarrow$ GNN Interpreter $\Rightarrow$ $p \,\mathrm{U}\, q$

# GNN-based LTLf Learning

- GNN Classifier
  - encode a trace as directed chain graph
  - Example




- K-embedding of node
  - $\boldsymbol{x}_v^{(i)} = [x_1, x_2, \dots, x_K]$, where $x_j$ denoted as $\left(\boldsymbol{x}_r^{(i)}\right)_j$

# GNN-based LTLf Learning

- GNN Classifier $\mathcal{A}(\boldsymbol{C}, \boldsymbol{A}, \boldsymbol{b})$
  - Message passing
    - $\boldsymbol{x}_v^{(i)} = f(\boldsymbol{x}_v^{(i-1)}\boldsymbol{C} + \sum_{u \in \mathcal{N}(v)} \boldsymbol{x}_u^{(i-1)} \boldsymbol{A} + \boldsymbol{b})$
    - $f(x) = \min(\max(0, x), 1)$
  - K+L iterations
  - Loss function
    - $L_{ACC} = \frac{1}{N}\sum_i -(y_i \cdot \log\left(\left(\boldsymbol{x}_r^{(K+L)}\right)_1\right) + (1 - y_i) \cdot \log(1 - \left(\boldsymbol{x}_r^{(K+L)}\right)_1))$
    - $L_{INT} = \left\|\boldsymbol{C}^{\mathrm{T}}\right\|_1 + \left\|\boldsymbol{A}^{\mathrm{T}}\right\|_1$
    - Minimize $L_{ACC} + \lambda_1 L_{ACC} + \lambda_1 L_{INT}$

# GNN-based LTLf Learning

- GNN Interpreter
  - If $f_i \in \mathbb{P}$, then $\boldsymbol{C}_{ji} = 1$;
  - If $f_i = f_j \wedge f_k$, then $\boldsymbol{C}_{ji} = 1$, $\boldsymbol{C}_{ki} = 1$, and $\boldsymbol{b}_i = -1$;
  - If $f_i = \neg f_j$, then $\boldsymbol{C}_{ji} = -1$ and $\boldsymbol{b}_i = 1$;
  - If $f_i = X f_j$, then $\boldsymbol{A}_{ji} = 1$;
  - If $f_i = f_j\ U\ f_k$, then $\boldsymbol{C}_{ki} = 2$, $\boldsymbol{C}_{ji} = 1$, $\boldsymbol{A}_{ii} = 1$, and $\boldsymbol{b}_i = -1$;
  - ...
- Challenge: the interpreter is not unique.

# Conclusion

1. GNNs can express the LTLf classifier on the directed path graph.

2. This result provides a interesting solution to the problem of LTLf learning.

3. The interpretation of the parameters of the GNN classifier is still a challenge.

# References


[1] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In ICLR, 2019.

[2] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In ICLR, 2019.

[3] Ralph Abboud, ˙Ismail ˙Ilkan Ceylan, and Thomas Lukasiewicz. Learning to reason: Leveraging neural networks for approximate DNF counting. In AAAI, pages 3097–3104, 2020.

[4] Bernd Finkbeiner, Christopher Hahn, Markus N. Rabe, and Frederik Schmitt. Teaching temporal logics to neural networks. In ICLR, 2021.

[5] Kristin Y. Rozier and Moshe Y. Vardi. LTL satisfiability checking. volume 4595, pages 149–167, 2007.

[6] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In AAAI, pages 4602–4609, 2019.

[7] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In ICLR, 2019.

[8] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In ICLR, 2020.

[9] Alberto Camacho and Sheila A. McIlraith. Learning interpretable models expressed in linear temporal logic. In ICAPS, pages 621–630, 2019.

[10] Joseph Kim, Christian Muise, Ankit Shah, Shubham Agarwal, and Julie Shah. Bayesian inference of linear temporal logic specifications for contrastive explanations. In IJCAI, pages 5591–5598, 2019.

# Thank You

# Experiments and Analysis

- Competitors
  - Camacho and McIlraith[10]
  - BayesLTL[11]
- Benchmarks (follow [10])
  - blocks
  - rovers
  - satellite
  - storage
  - TPP
  - Zenotravel

# Experiments and Analysis

- Result

# Backup

- Semantics of LTL
  - $\pi_t \vDash p$ iff $p \in s_t, p \in \mathbb{P}$
  - $\pi_t \vDash \neg\varphi$ iff $\pi_t \nvDash \varphi$
  - $\pi_t \vDash \varphi_1 \vee \varphi_2$ iff $\pi_t \vDash \varphi_1$ or $\pi_t \vDash \varphi_2$
  - $\pi_t \vDash F\varphi$ iff $\pi_{t+1} \vDash \varphi$
  - $\pi_t \vDash \varphi_1 U \varphi_2$ iff $\exists k \geq t, \pi_k \vDash \varphi_2$ and $\forall t \leq j < k, \pi_j \vDash \varphi_1$