Teaching LTL_f Satisfiability Checking to Neural Networks

Weilin Luo[1], Hai Wan [1,2*], Jianfeng Du [3,4,*], Xiaoda Li [1], Yuze Fu [1], Rongzhen Ye [1], Delong Zhang [1]

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2] Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China
[3] Guangdong University of Foreign Studies, Guangzhou, China
[4] Pazhou Lab, Guangzhou, China

Content

Content

Motivation

Linear temporal logic over finite traces (LTL$_f$) satisfiability checking: checking whether a given LTL$_f$ formula is satisfiable or unsatisfiable

- AI applications: reinforcement learning[15], program synthesis[14], and explainable AI[9]
- PSPACE-complete[7]

Motivation

Linear temporal logic over finite traces (LTL_f) satisfiability checking: checking whether a given LTL_f formula is satisfiable or unsatisfiable

- AI applications: reinforcement learning[15], program synthesis[14], and explainable AI[9]
- PSPACE-complete[7]

Related work

- *symbolic approaches*: *e.g.*, based on tableau[10] and based on SAT[6,11]
- sound and complete
- *no* symbolic approach scales to all datasets[11]

Motivation

Linear temporal logic over finite traces (LTL$_f$) satisfiability checking: checking whether a given LTL$_f$ formula is satisfiable or unsatisfiable

- AI applications: reinforcement learning[15], program synthesis[14], and explainable AI[9]
- PSPACE-complete[7]

Related work

- *symbolic approaches*: *e.g.*, based on tableau[10] and based on SAT[6,11]
- sound and complete
- *no* symbolic approach scales to all datasets[11]
- end-to-end neural networks to solve Boolean satisfiability (SAT) problem in *polynomial time*[4,12]

Motivation

Linear temporal logic over finite traces (LTL$_f$) satisfiability checking: checking whether a given LTL$_f$ formula is satisfiable or unsatisfiable

- AI applications: reinforcement learning[15], program synthesis[14], and explainable AI[9]
- PSPACE-complete[7]

Related work

- *symbolic approaches*: *e.g.*, based on tableau[10] and based on SAT[6,11]
- sound and complete
- *no* symbolic approach scales to all datasets[11]
- end-to-end neural networks to solve Boolean satisfiability (SAT) problem in *polynomial time*[4,12]

Whether LTL$_f$ satisfiability checking can be effectively tackled by end-to-end neural networks?

Content

Recursion

- syntactics of recursive definition: $\phi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{X}\varphi \mid \varphi_1 \ \mathsf{U} \ \varphi_2$

Recursion

- syntactics of recursive definition: $\phi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{X}\varphi \mid \varphi_1 \ \mathsf{U} \ \varphi_2$
- semantics of recursive definition

Example 1 (Recursion)

Let $\{p, q, r\}$ be a set of atomic propositions.

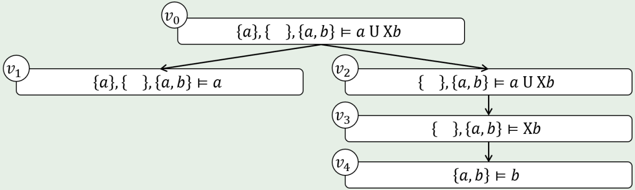


Figure 1: The semantics of a U (Xb) is recursive.

Permutation Invariant and Sequentiality

Example 2 (Permutation Invariant)

permutation invariance of sub-formulae

- $north \vee west \equiv west \vee north$
- $(north \vee west)\ \mathsf{U}\ door \equiv (west \vee north)\ \mathsf{U}\ door$

Permutation Invariant and Sequentiality

Example 2 (Permutation Invariant)

permutation invariance of sub-formulae

- $north \vee west \equiv west \vee north$
- $(north \vee west)\ \mathsf{U}\ door \equiv (west \vee north)\ \mathsf{U}\ door$

permutation invariance of atomic propositions

- $(north \vee door)\ \mathsf{U}\ west$
- $(north \vee west)\ \mathsf{U}\ door$

Permutation Invariant and Sequentiality

Example 2 (Permutation Invariant)

permutation invariance of sub-formulae

- $north \lor west \equiv west \lor north$
- $(north \lor west)\ \mathsf{U}\ door \equiv (west \lor north)\ \mathsf{U}\ door$

permutation invariance of atomic propositions

- $(north \lor door)\ \mathsf{U}\ west$
- $(north \lor west)\ \mathsf{U}\ door$

Example 3 (Sequentiality)

- $(door\ \mathsf{U}\ west) \land \mathsf{G}\neg door$ is satisfiable, while $(west\ \mathsf{U}\ door) \land \mathsf{G}\neg door$ is unsatisfiable
- $door\ \mathsf{U}\ west \not\equiv west\ \mathsf{U}\ door$

LTLfNet: TreeNN-based Embedding Model

Motivation

- **Recursion**: the architecture of recursive neural network (TreeNN)[3]
- **Permutation Invariance** and **Sequentiality**: aggregate functions fulfilling permutation invariance or sequentiality

LTLfNet: TreeNN-based Embedding Model

Motivation

- **Recursion**: the architecture of recursive neural network (TreeNN) [3]
- **Permutation Invariance** and **Sequentiality**: aggregate functions fulfilling permutation invariance or sequentiality

LTLfNet

- COMBINE with different learnable parameters for different logical operators

Algorithm 2: COMBINE

Input : an aggregated representation $\mathbf{r}$ of sub-formulae and the logical operator op.
Output : the combination representation $\mathbf{r}_{out}$.

1 $\mathbf{r}' \leftarrow \mathrm{ReLU}(\mathbf{W}_{0,op} \cdot \mathbf{r})$
2 $\mathbf{r}_{out} \leftarrow \mathbf{W}_{1,op} \cdot \mathbf{r}' + \mathbf{W}_{2,op} \cdot \mathbf{r}$
3 **return** $\mathbf{r}_{out} / \|\mathbf{r}_{out}\|_2$

- aggregation function for each operator
  - permutation invariance: mean pooling
  - sequentiality: concatenate

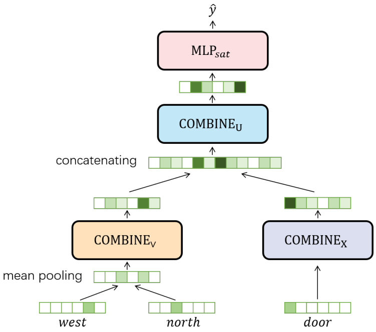


Figure 2: LTLfNet embeds $(west \vee north)\ \mathrm{U}\ \mathrm{X} door$

Content

Dataset

synthetic dataset

- *randltl* tool in the SPOT framework to generate random formulae
- formula size is in the interval $[20, 100)$
- the number of different atomic propositions is less than 100
- 16K/2K formulae for training/validating
- 6 test sets with different size intervals: $[20, 100)$, $[100, 120)$, $[120, 140)$, $[140, 160)$, $[160, 180)$, and $[180, 200)$ (2K formulae for each)

Dataset

synthetic dataset

- *randltl* tool in the SPOT framework to generate random formulae
- formula size is in the interval $[20, 100)$
- the number of different atomic propositions is less than 100
- 16K/2K formulae for training/validating
- 6 test sets with different size intervals: $[20, 100)$, $[100, 120)$, $[120, 140)$, $[140, 160)$, $[160, 180)$, and $[180, 200)$ (2K formulae for each)

large-scale datasets[11]

- *LTL-as-LTL$_f$*: 4668 formulae coming from LTL satisfiability checking
- *LTL$_f$-Specific*: 1700 formulae generated by common LTL$_f$ patterns
- *NASA-Boeing*: real-world LTL$_f$ specifications
- *DECLARE*: 112 LTL$_f$ patterns widely used in the business process management

Competitor

`Transformer`: transformer-based embedding model

- generating LTL satisfiable traces by training a Transformer model [8]
- **Permutation Invariance** and **Sequentiality**: Multi-head Self-attention [4]

`RGCN`: R-GCN-based embedding model

- relational graph convolutional network (R-GCN) embeds commands in LTL to train an agent to make command-compliant decisions [13]
- **Recursion**: GNNs provide better inductive bias [13]
- **Permutation Invariance**: aggregate functions that is exchangeable [1,2,4,12,16]

`TreeNN`: TreeNN-based embedding model

- does not apply different COMBINE functions to different operators

Competitor

`Transformer`: transformer-based embedding model

- generating LTL satisfiable traces by training a Transformer model[8]
- **Permutation Invariance** and **Sequentiality**: Multi-head Self-attention[4]

`RGCN`: R-GCN-based embedding model

- relational graph convolutional network (R-GCN) embeds commands in LTL to train an agent to make command-compliant decisions[13]
- **Recursion**: GNNs provide better inductive bias[13]
- **Permutation Invariance**: aggregate functions that is exchangeable[1,2,4,12,16]

`TreeNN`: TreeNN-based embedding model

- does not apply different COMBINE functions to different operators

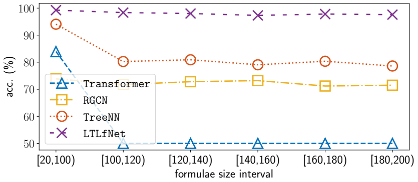
`CDLSC`: SOTA symbolic approach to LTL_f satisfiability checking[11]
`nuXmv`: SOTA approaches to model checking[5]

Evaluation on Synthetic Datasets

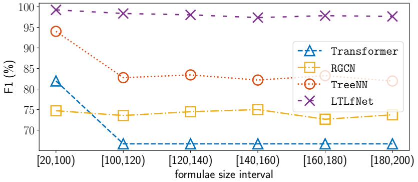
Model	acc. (%)	pre. (%)	rec. (%)	F1 (%)	time (s)
Transformer	83.95	93.58	72.90	81.96	**5.63**
RGCN	73.85	72.31	77.30	74.72	97.01
TreeNN	94.05	94.54	93.50	94.02	13.83
LTLfNet (our)	**99.25**	**98.91**	**99.60**	**99.25**	14.72

Table 1: Evaluation on the synthetic datasets as the same size of training formulae ($[20, 100)$).

- LTLfNet outperforms other approaches and keeps the high performance even when formulae become larger.
- The architecture fulfilling more logical properties has better scale generalizability.



(a) Accuracy



(b) F1 score

Figure 3: Results of different approaches on test sets with formulae of different sizes.

Evaluation on Large Scale Datasets

Model	*LTL-as-LTL$_f$* acc. (%)	F1 (%)	time (s)	*LTL$_f$-Specific* acc. (%)	F1 (%)	time (s)	*NASA-Boeing* acc. (%)	F1 (%)	time (s)	*DECLARE* acc. (%)	F1 (%)	time (s)
CDLSC	100.00	100.00	75,979.65	100.00	100.00	27.47	100.00	100.00	3,604.41	100.00	100.00	60,905.95
nuXmv	100.00	100.00	75,560.60	100.00	100.00	2,483.38	100.00	100.00	3,695.72	100.00	100.00	18,096.68
Transformer	47.33	62.63	12.98	61.71	61.27	4.25	98.39	99.19	1.34	100.00	100.00	3.71
RGCN	39.17	55.16	4,412.20	54.18	70.28	3,309.81	100.00	100.00	216.92	100.00	100.00	6,854.66
TreeNN	88.65	93.94	311.08	54.18	70.28	101.42	100.00	100.00	27.50	100.00	100.00	170.78
LTLfNet (our)	89.77	94.61	327.25	54.18	70.28	130.93	100.00	100.00	28.70	100.00	100.00	177.36

Table 2: Evaluation on the large-scale datasets.

- The neural approaches are much faster than the symbolic approaches generally.
- Neural approaches are able to learn biases that are widely present in industrial datasets.
- LTLfNet achieves highly confident prediction for LTL$_f$ satisfiability checking in relatively short time.

Content

Conclusion and Future Work

Conclusion:

1. By designing the neural architecture (`LTLfNet`) to characterize logical properties of LTL_f, end-to-end neural networks can learn to check LTL_f satisfiability.
2. Experimental results show the competitive results of `LTLfNet` compared with the SOTA symbolic approaches.

Conclusion and Future Work

Conclusion:

1. By designing the neural architecture (LTLfNet) to characterize logical properties of LTL_f, end-to-end neural networks can learn to check LTL_f satisfiability.
2. Experimental results show the competitive results of LTLfNet compared with the SOTA symbolic approaches.

Future work:

1. improve our approach to generalize across distributions
2. evaluate our approach in LTL satisfiability checking
3. extend our approach to generate a trace as evidence of satisfiability

References I

[1] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *ICLR*, 2019.

[2] Benedikt Bünz and Matthew Lamm. Graph neural networks and boolean satisfiability. *CoRR*, abs/1702.03592, 2017.

[3] Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. In *ICLR*, 2017.

[4] Chris Cameron, Rex Chen, Jason S. Hartford, and Kevin Leyton-Brown. Predicting propositional satisfiability via end-to-end learning. In *AAAI*, pages 3324–3331, 2020.

[5] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuxmv symbolic model checker. In *CAV*, pages 334–342, 2014.

[6] Valeria Fionda and Gianluigi Greco. The complexity of LTL on finite traces: Hard and easy fragments. In *AAAI*, pages 971–977, 2016.

[7] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.

[8] Christopher Hahn, Frederik Schmitt, Jens U. Kreber, Markus Norman Rabe, and Bernd Finkbeiner. Teaching temporal logics to neural networks. In *ICLR*, 2021.

[9] 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.

[10] Jianwen Li, Lijun Zhang, Geguang Pu, Moshe Y. Vardi, and Jifeng He. Ltlf satisfiability checking. In *ECAI*, volume 263, pages 513–518, 2014.

References II

[11] Jianwen Li, Geguang Pu, Yueling Zhang, Moshe Y. Vardi, and Kristin Y. Rozier. Sat-based explicit ltlf satisfiability checking. *Artif. Intell.*, 289:103369, 2020.

[12] 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*, pages 1–11, 2019.

[13] Pashootan Vaezipoor, Andrew C. Li, Rodrigo Toro Icarte, and Sheila A. McIlraith. Ltl2action: Generalizing LTL instructions for multi-task RL. In *ICML*, volume 139, pages 10497–10508, 2021.

[14] Shengping Xiao, Jianwen Li, Shufang Zhu, Yingying Shi, Geguang Pu, and Moshe Y. Vardi. On-the-fly synthesis for LTL over finite traces. In *AAAI*, pages 6530–6537, 2021.

[15] Yaqi Xie, Fan Zhou, and Harold Soh. Embedding symbolic temporal knowledge into deep sequential models. In *ICRA*, pages 4267–4273, 2021.

[16] Wenjie Zhang, Zeyu Sun, Qihao Zhu, Ge Li, Shaowei Cai, Yingfei Xiong, and Lu Zhang. Nlocalsat: Boosting local search with solution prediction. In *IJCAI*, pages 1177–1183, 2020.

Thank you for your listening!