# Deep Learning for Temporal Logics

*Frederik Schmitt*, Christopher Hahn, Jens U. Kreber,
Markus N. Rabe, Bernd Finkbeiner

6th Conference on Artificial Intelligence and Theorem Proving

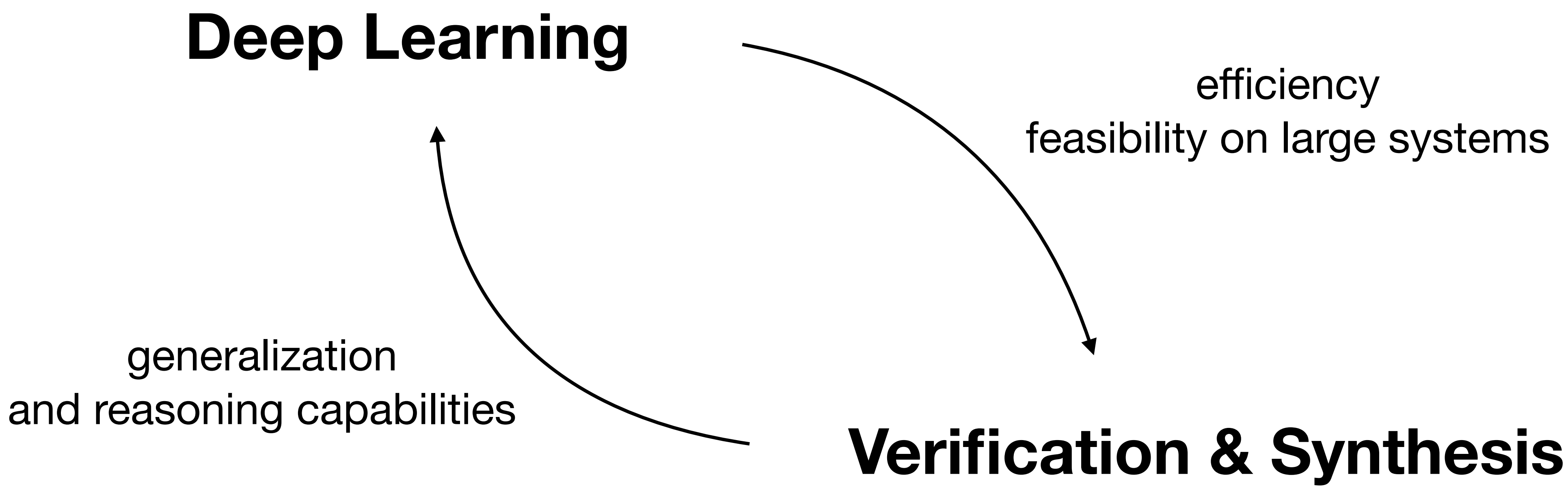
September 6, 2021

CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

UNIVERSITÄT
DES
SAARLANDES

Google Research

# Deep Learning for Formal Methods



# Examples of Related Work

## NeuroSAT

Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., Dill, D.L.: Learning a SAT Solver from Single-Bit Supervision. ICLR 2019

## FastSMT

Balunović, M., Bielik, P., Vechev, M.: Learning to Solve SMT Formulas. NeurIPS 2018

## DeepHOL

Bansal, K., Loos, S.M., Rabe, M.N., Szegedy, C., Wilcox, S.: HOList: An Environment for Machine Learning of Higher-Order Theorem Proving. ICML 2019

## DeepMath

Alemi, A. A., Chollet, F., Een, N., Irving, G., Szegedy, C., Urban, J.: DeepMath: Deep Sequence Models for Premise Selection. NeurIPS 2016

**...**

# Deep Learning for Temporal Logics

**Deep Learning**

efficiency
feasibility on large systems

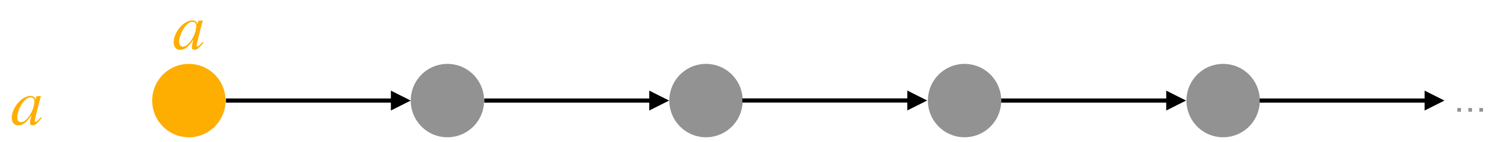
generalization
and reasoning capabilities

***Temporal Logics***

- *Trace generation Problem (PSPACE)*
- *Circuit synthesis Problem (2EXPTIME)*

# Linear-time Temporal Logic (LTL)[1]

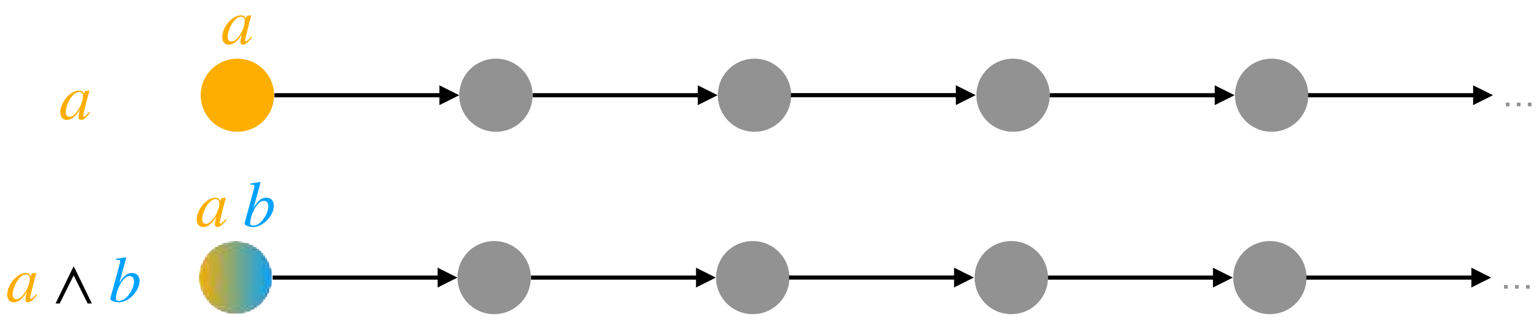
$\varphi, \psi ::= a \mid \mathsf{true} \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi \,\mathsf{U}\, \psi$ where $a \in AP$




[1] Pnueli, A.: The Temporal Logic of Programs. 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 1977

# Linear-time Temporal Logic (LTL)[1]

$$\varphi, \psi ::= a \mid \text{true} \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi \, \mathsf{U} \, \psi \;\; \text{where} \;\; a \in AP$$




[1] Pnueli, A.: The Temporal Logic of Programs. 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 1977

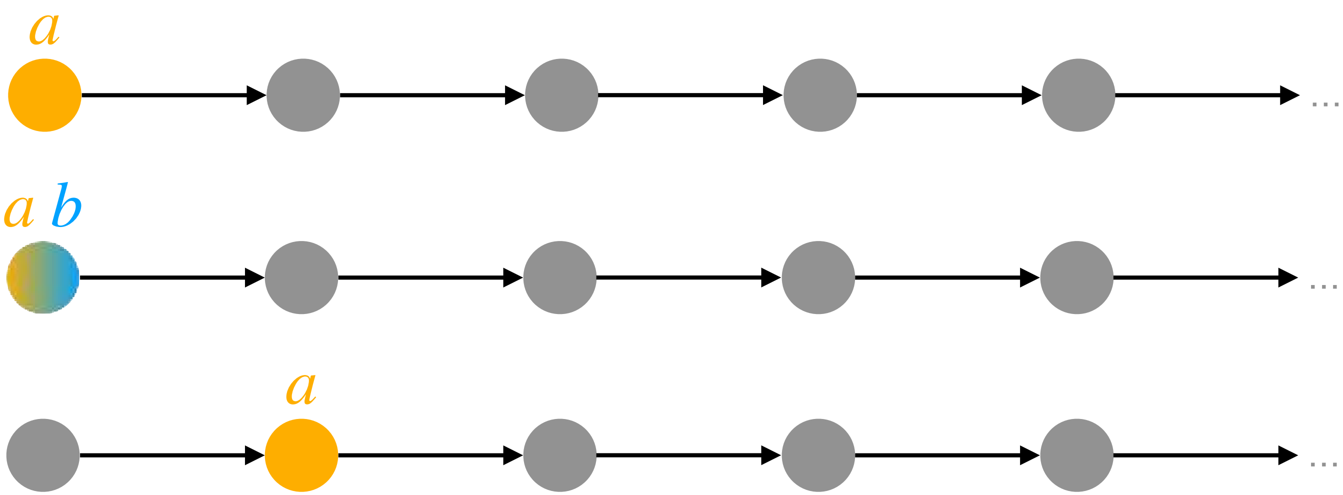
# Linear-time Temporal Logic (LTL)[1]

$$\varphi, \psi ::= a \mid \text{true} \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi \,\mathsf{U}\, \psi \quad \text{where } a \in AP$$

$a$

$a \wedge b$

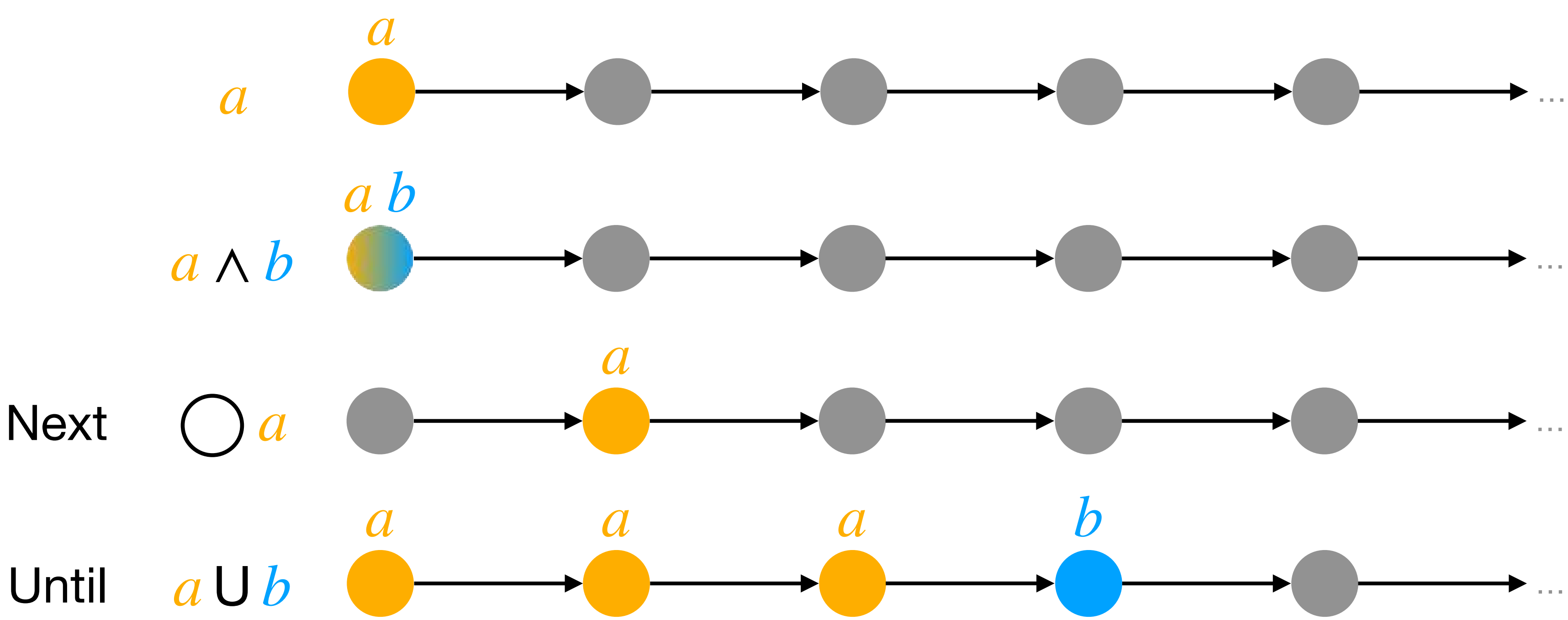
Next $\bigcirc a$




[1] Pnueli, A.: The Temporal Logic of Programs. 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 1977

# Linear-time Temporal Logic (LTL)[1]

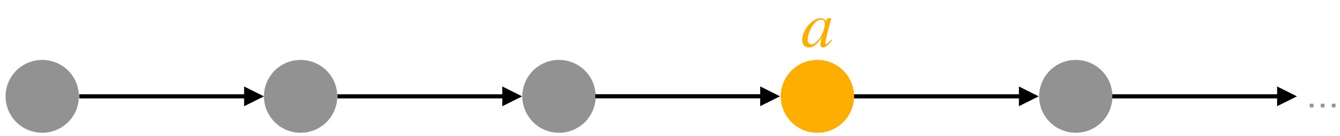
$$\varphi, \psi ::= a \mid \text{true} \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc\, \varphi \mid \varphi\, \mathsf{U}\, \psi \quad \text{where } a \in AP$$




[1] Pnueli, A.: The Temporal Logic of Programs. 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 1977

# Linear-time Temporal Logic (LTL)[1]

Eventually

$\Diamond a := \text{true} \, \mathsf{U} \, a$ 


Globally

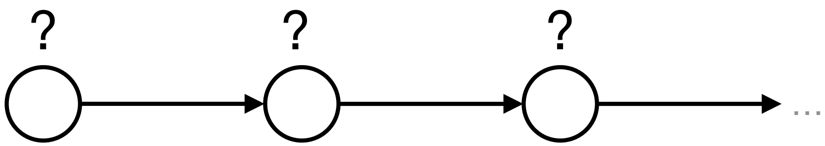
$\Box a := \neg \Diamond \neg a$ 


[1] Pnueli, A.: The Temporal Logic of Programs. 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 1977

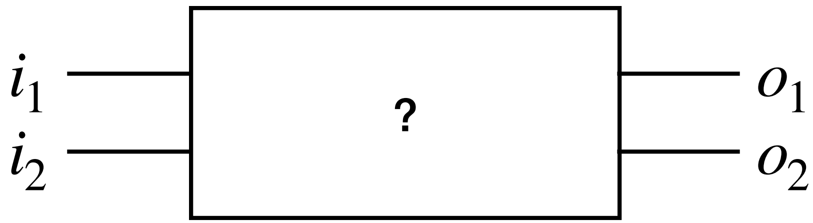
# Part 1:
# Trace Generation

# Part 2:
# Circuit Synthesis

Trace $\pi \models$ LTL Formula $\varphi$




Circuit $C \models$ LTL Specification $\varphi$

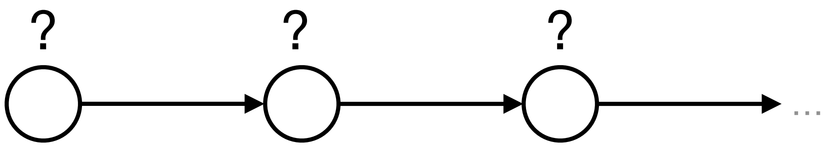



Hahn, C., S., F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.: Teaching Temporal Logics to Neural Networks. ICLR 2021
S., F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural Circuit Synthesis from Specification Patterns. arXiv Preprint 2021

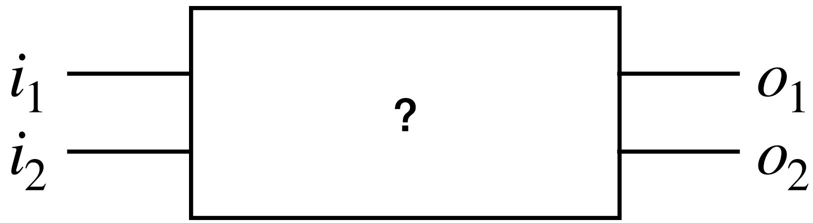
# Part 1: Trace Generation

# Part 2: Circuit Synthesis

Trace $\pi \models$ LTL Formula $\varphi$




Circuit $C \models$ LTL Specification $\varphi$

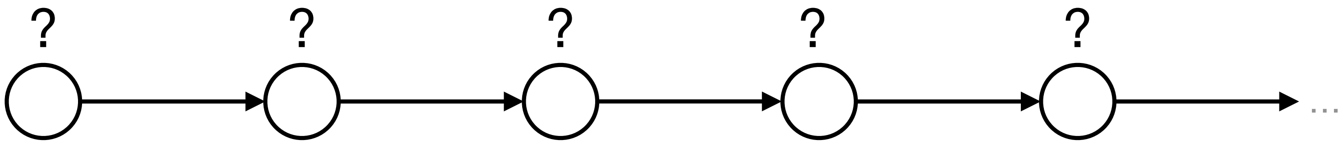




Hahn, C., S., F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.: Teaching Temporal Logics to Neural Networks. ICLR 2021
S., F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural Circuit Synthesis from Specification Patterns. arXiv Preprint 2021

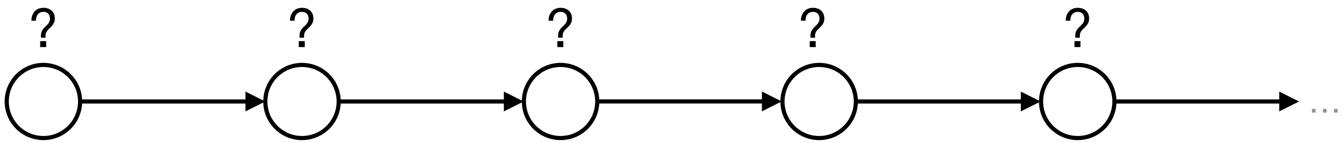
# Trace Generation Problem

$$(a \mathsf{U} (b \wedge c)) \wedge (a \mathsf{U} (\neg b \wedge c)) \wedge (a \mathsf{U} (\neg b \wedge \neg c))$$



# Trace Generation Problem

$$(a \,\mathsf{U}\, (b \wedge c)) \wedge (a \,\mathsf{U}\, (\neg b \wedge c)) \wedge (a \,\mathsf{U}\, (\neg b \wedge \neg c))$$

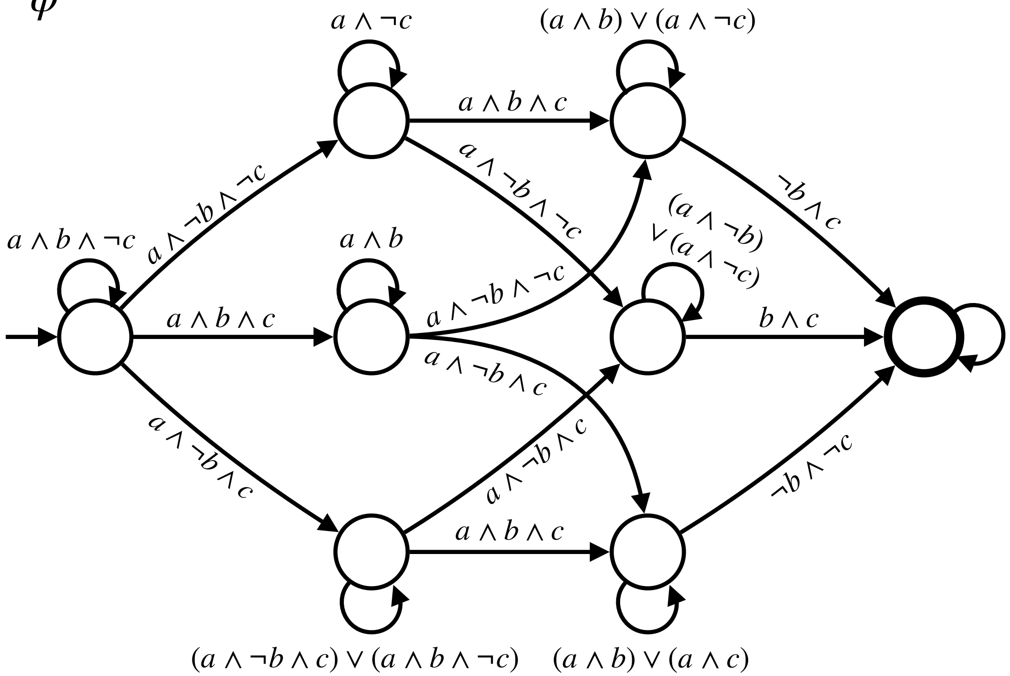



The LTL Trace Generation Problem is PSPACE-complete.

# Classic Trace Generation

$$\varphi = (a \,\mathsf{U}\, (b \land c)) \land (a \,\mathsf{U}\, (\neg b \land c)) \land (a \,\mathsf{U}\, (\neg b \land \neg c))$$

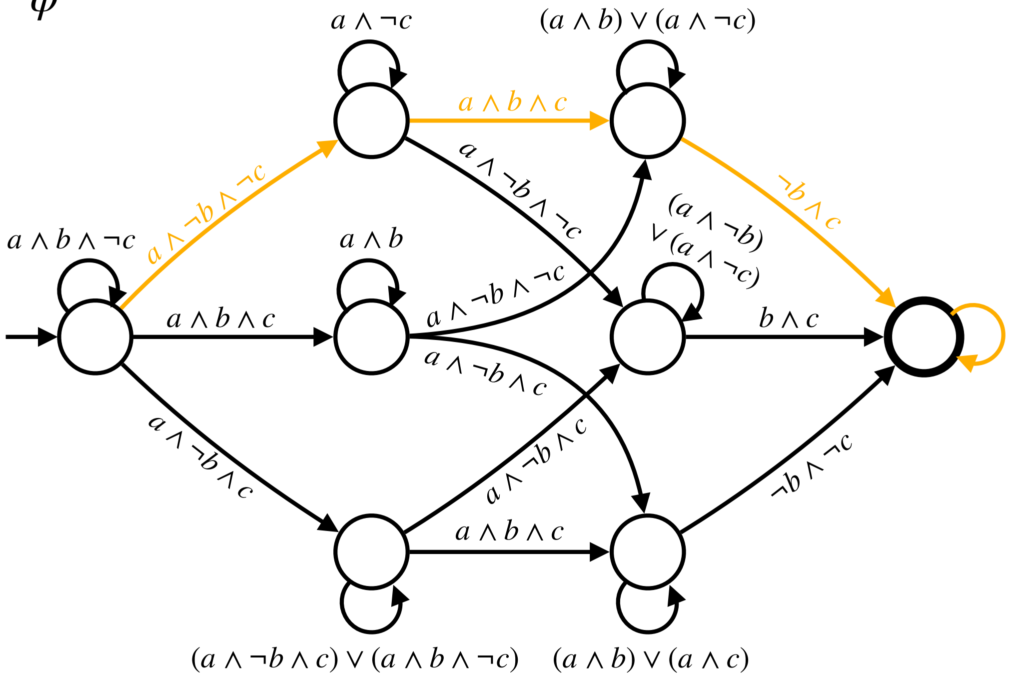
Büchi Automaton $A_\varphi$



# Classic Trace Generation

$$\varphi = (a \mathsf{U} (b \wedge c)) \wedge (a \mathsf{U} (\neg b \wedge c)) \wedge (a \mathsf{U} (\neg b \wedge \neg c))$$

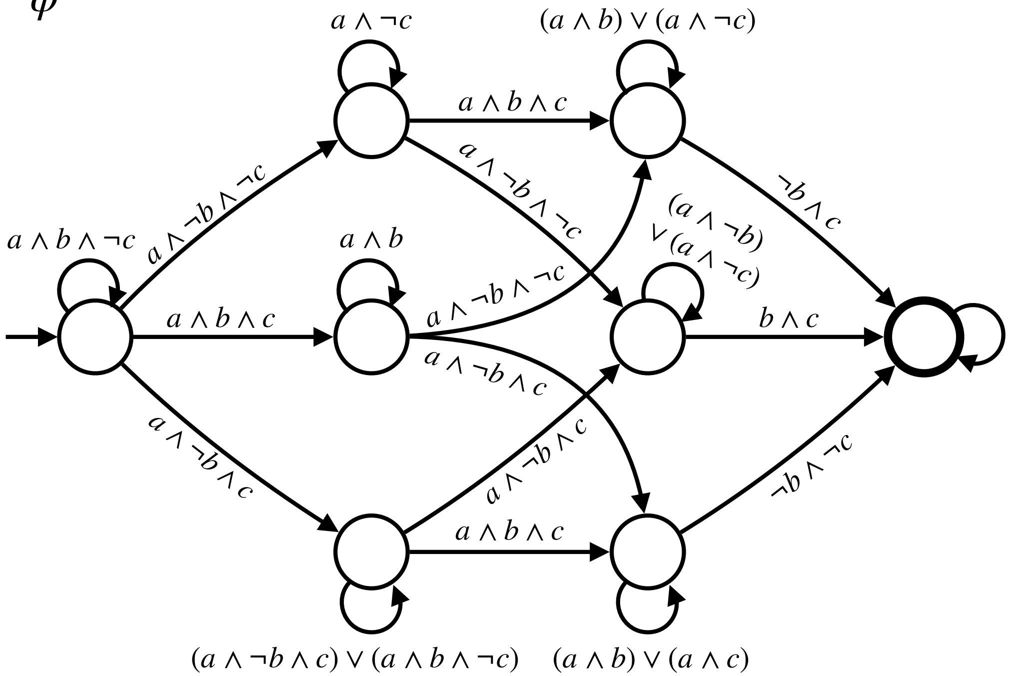
Büchi Automaton $A_\varphi$



# Classic Trace Generation

$$\varphi = (a \mathbin{\mathsf{U}} (b \land c)) \land (a \mathbin{\mathsf{U}} (\neg b \land c)) \land (a \mathbin{\mathsf{U}} (\neg b \land \neg c))$$

Büchi Automaton $A_\varphi$



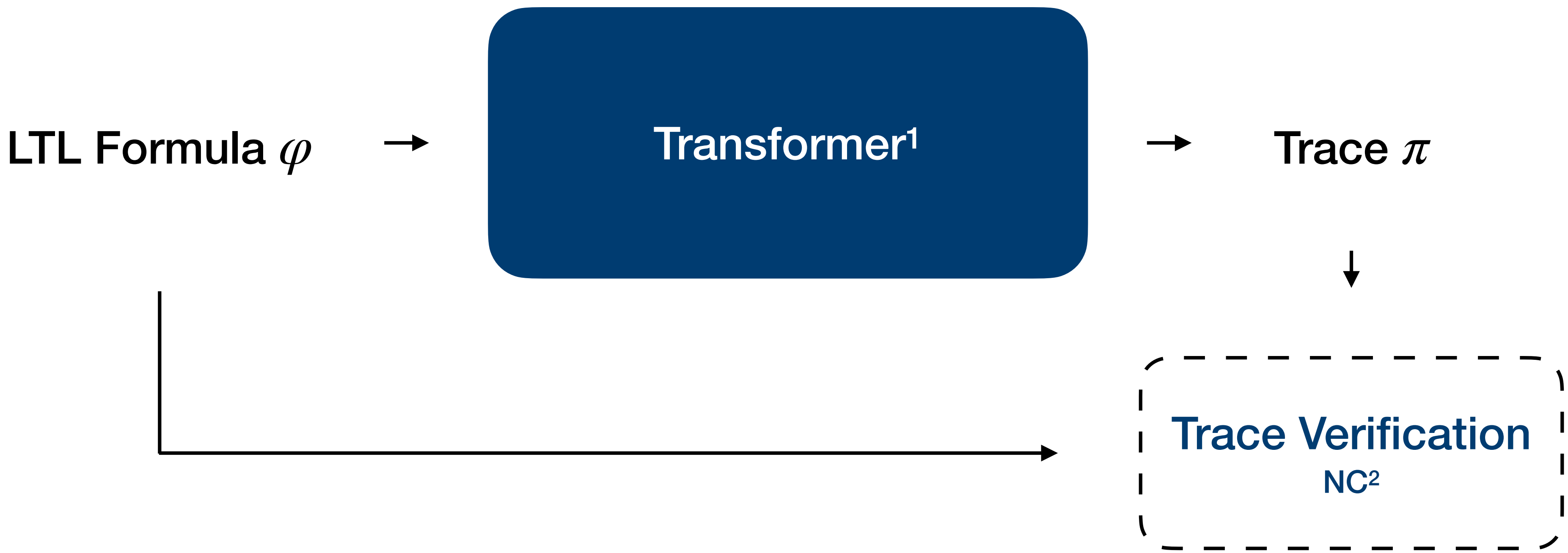

The number of states can be exponential in $|\varphi|$.

# *Neural* Trace Generation

LTL Formula $\varphi$ → Transformer[1] → Trace $\pi$

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. NeurIPS 2017

# Neural Trace Generation




[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. NeurIPS 2017

[2] Kuhtz, L., Finkbeiner, B.: LTL Path Checking is Efficiently Parallelizable. ICALP 2009

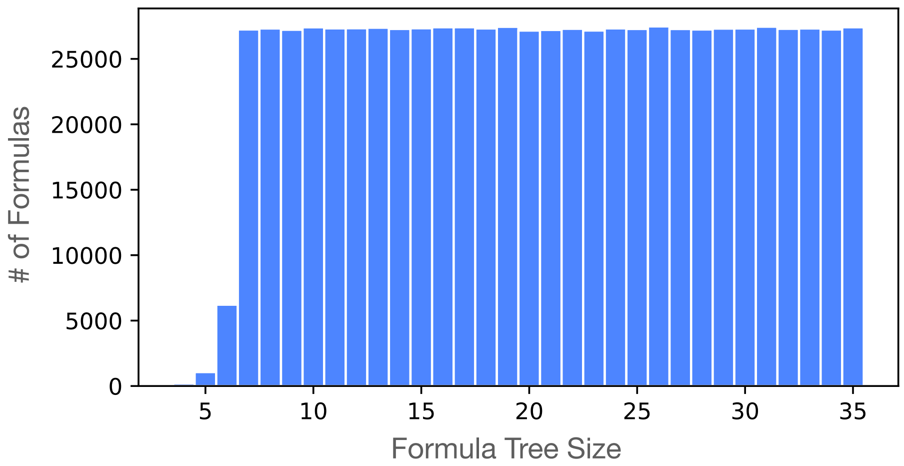
# Neural Trace Generation
## Datasets

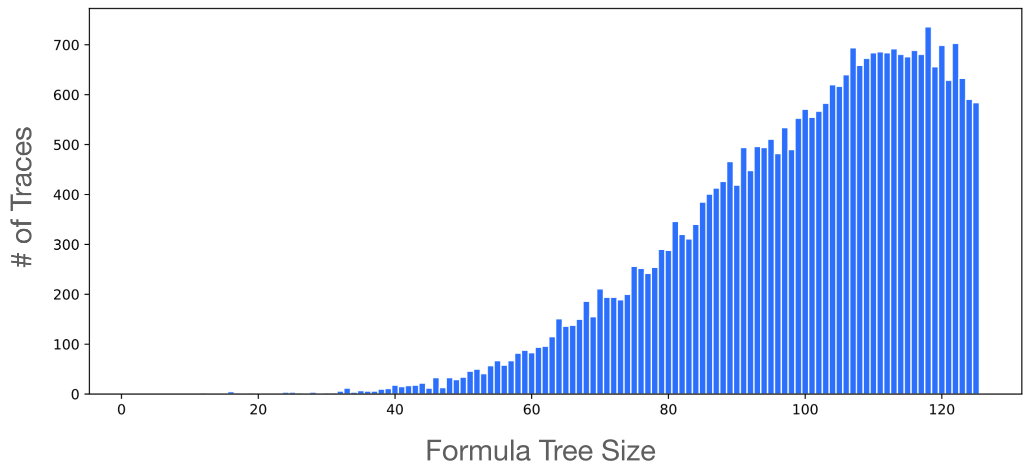
### LTLRandom35

- 1,000,000 formula-trace pairs
- Generated randomly
- Uniformly distributed in formula size

### LTLPattern126

- 1,664,487 formula-trace pairs
- Constructed from formula patterns[1]
- Conjunctions of patterns are hard to solve

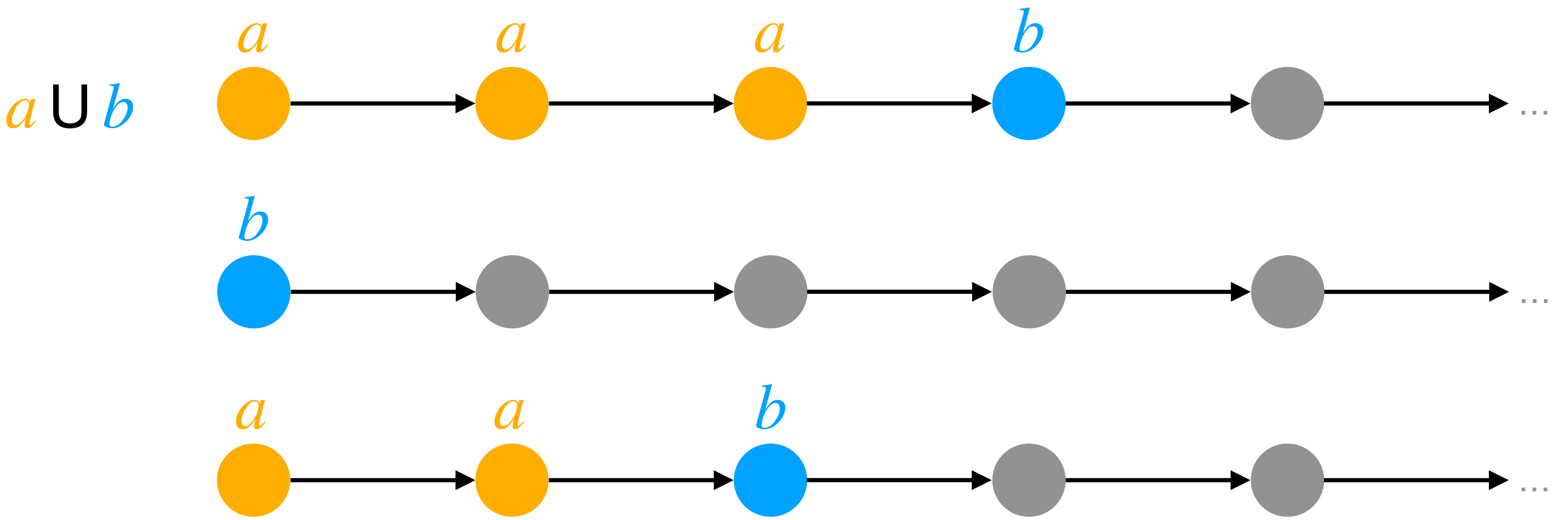



[1] Dwyer, M. B., Avrunin, G.S., Corbett, J.C.: Property Specification Patterns for Finite-State Verification. 2017

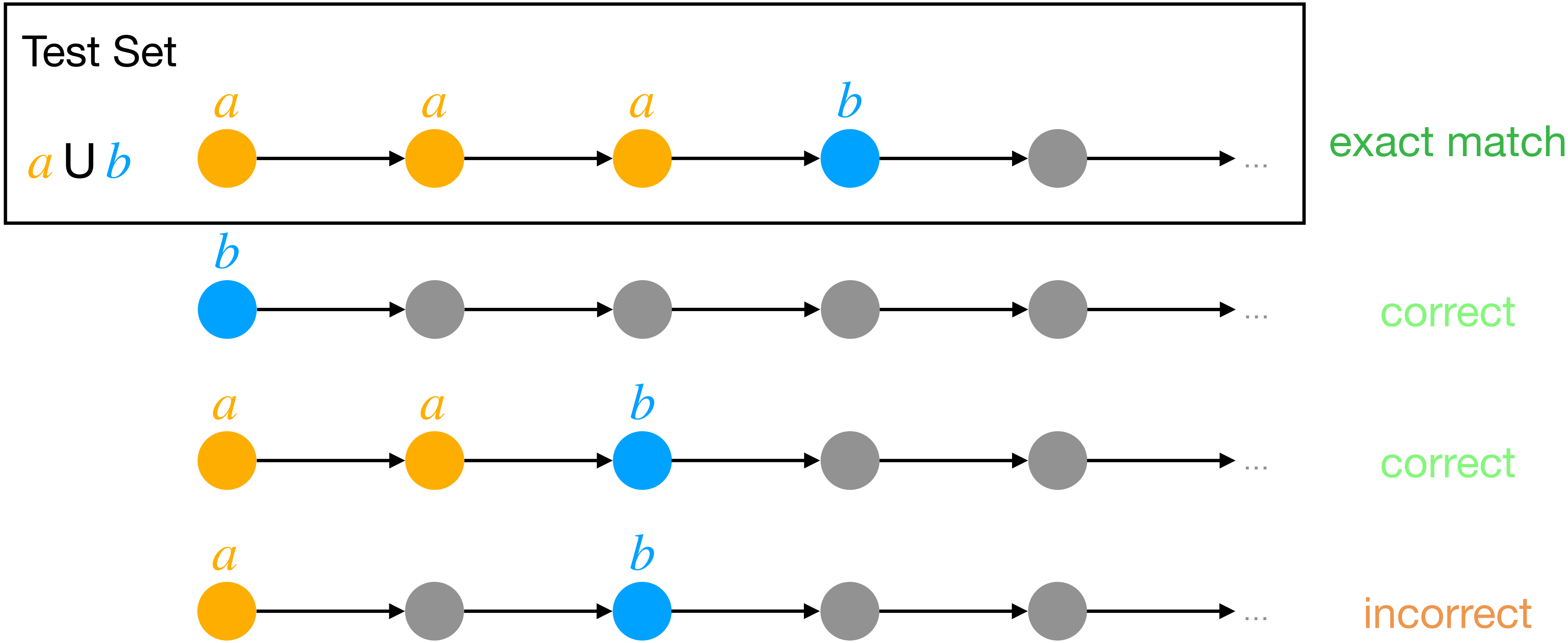
# Neural Trace Generation

## Performance Measures



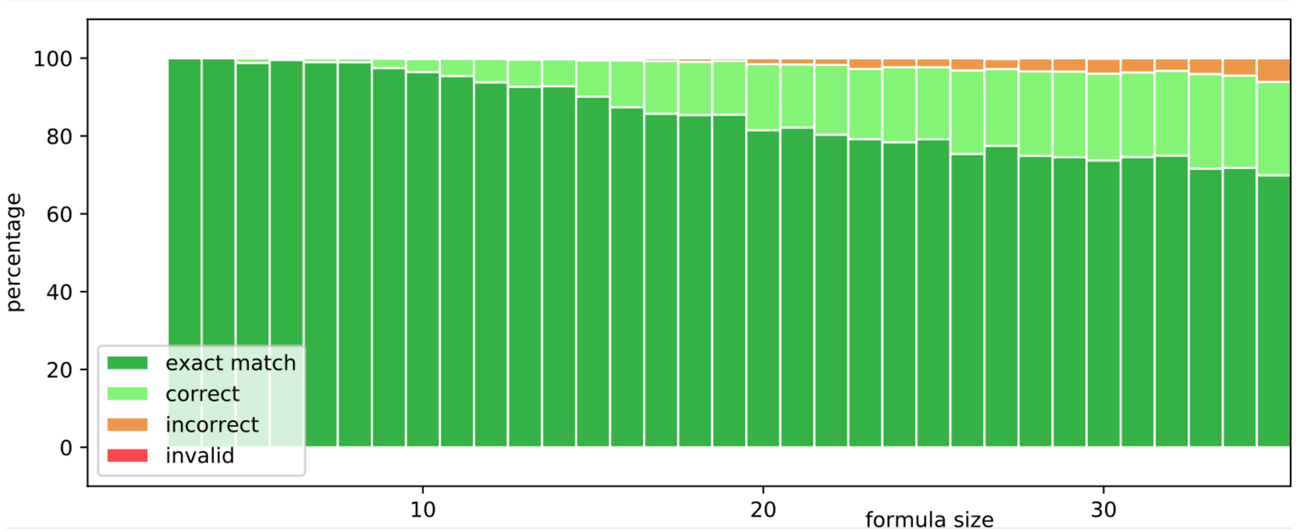
# Neural Trace Generation

## Performance Measures



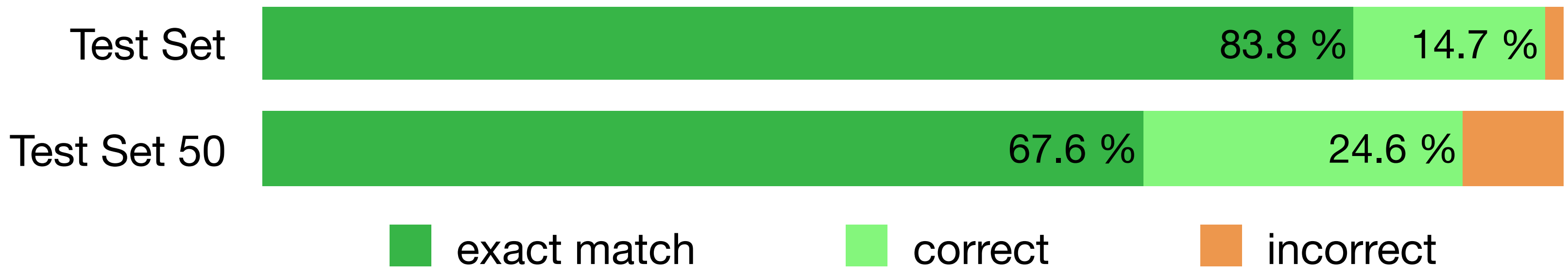
# Neural Trace Generation

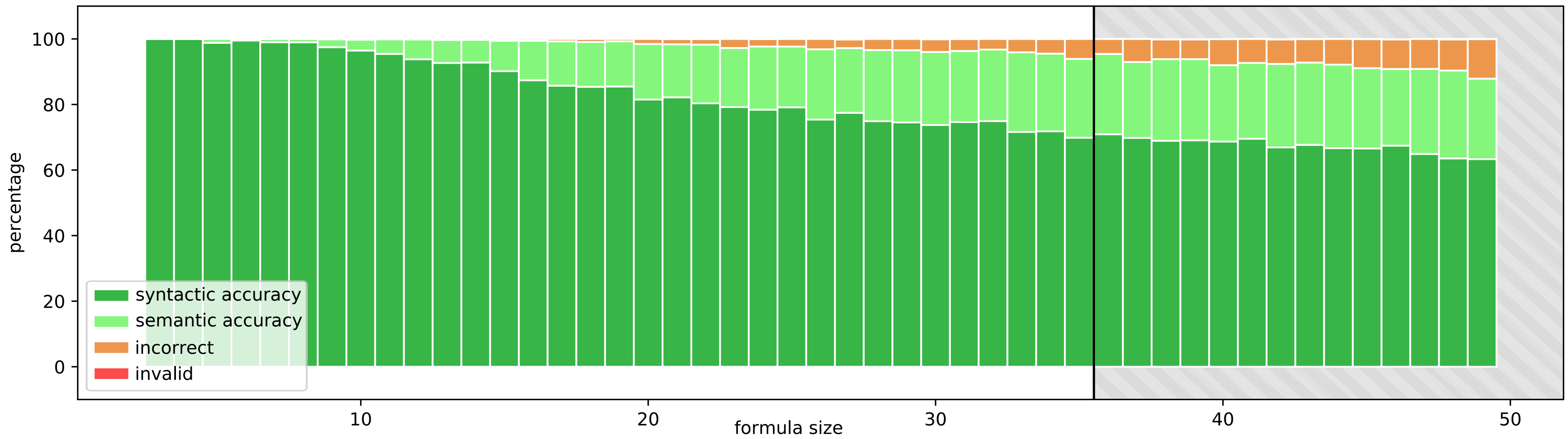
## Results - LTLRandom35

# Neural Trace Generation

## Results - LTLRandom35

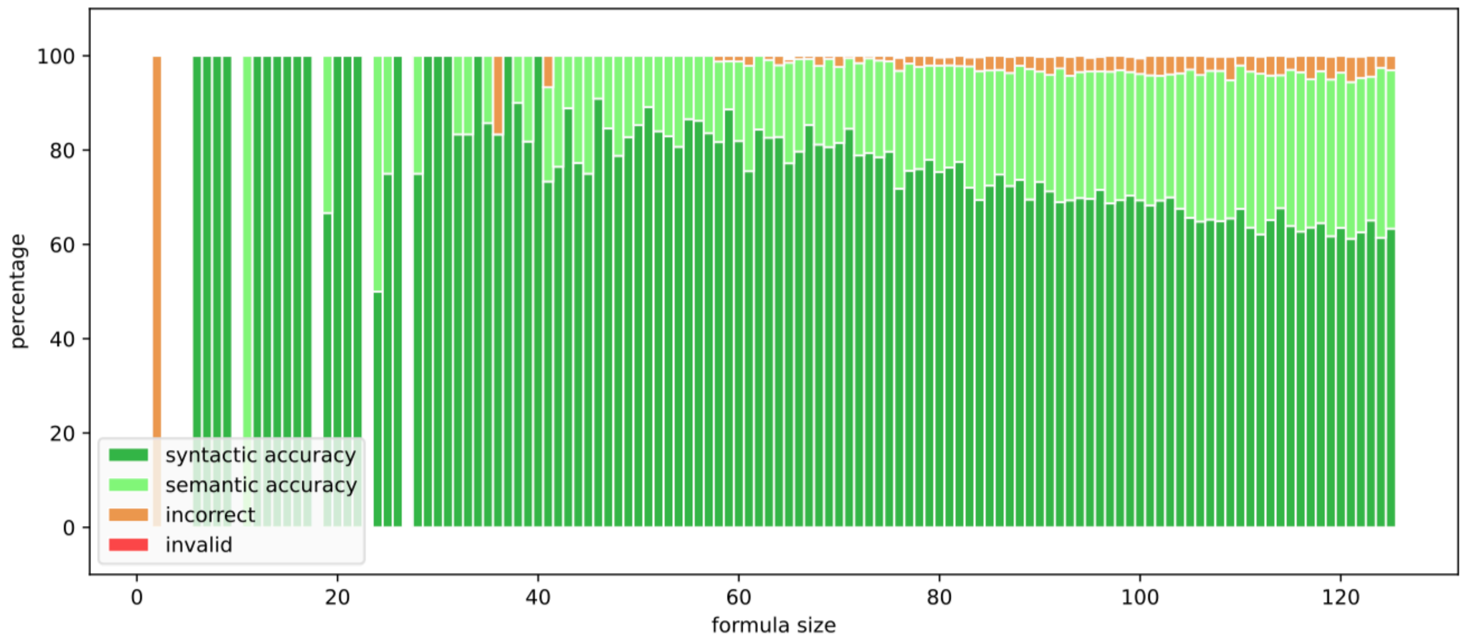



Tree positional encoding: Shiv, V. L. and Quirk, C.: Novel positional encodings to enable tree-based transformers. NeurIPS 2019

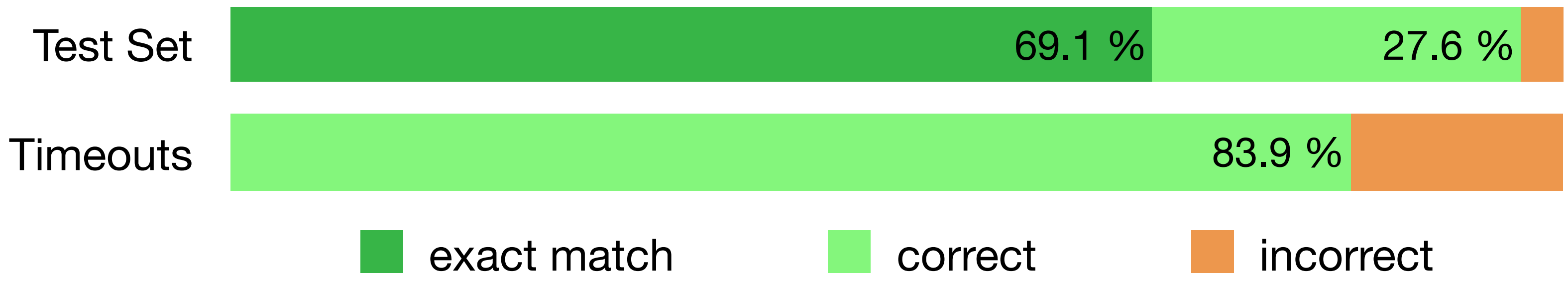
# Neural Trace Generation

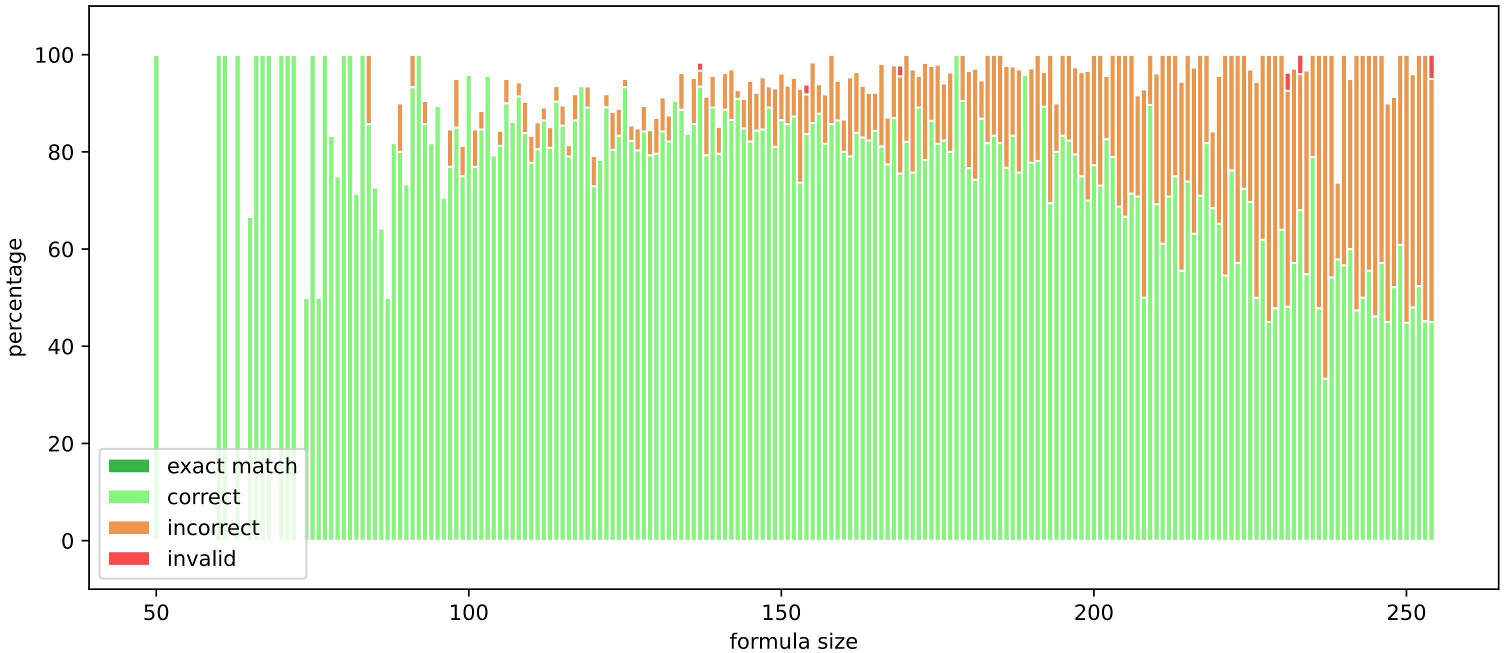
## Results - LTLPattern126

# Neural Trace Generation

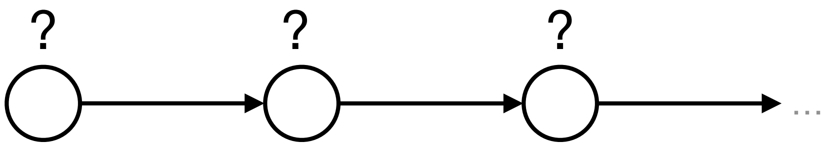
## Results - LTLPattern126



## Part 1: Trace Generation

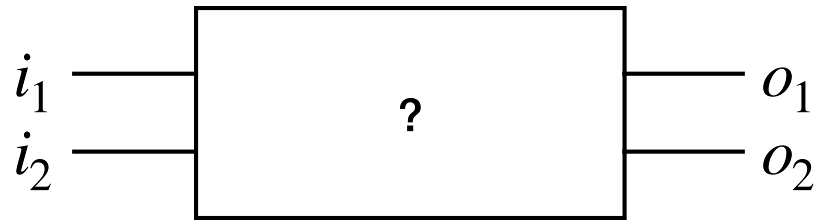
Trace $\pi \models$ LTL Formula $\varphi$




- Semantic generalization
- Generalization to larger formulas with tree positional encoding

## Part 2: Circuit Synthesis

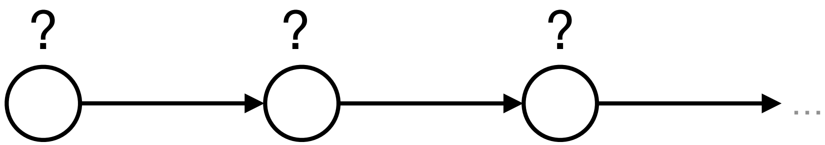
Circuit $C \models$ LTL Specification $\varphi$




Hahn, C., S., F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.: Teaching Temporal Logics to Neural Networks. ICLR 2021
S., F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural Circuit Synthesis from Specification Patterns. arXiv Preprint 2021

# Part 1:
# Trace Generation

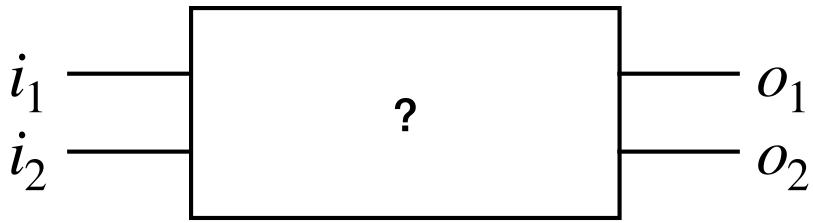
Trace $\pi \vDash$ LTL Formula $\varphi$




- Semantic generalization
- Generalization to larger formulas with tree positional encoding

# Part 2:
# Circuit Synthesis

Circuit $C \vDash$ LTL Specification $\varphi$




Hahn, C., S., F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.: Teaching Temporal Logics to Neural Networks. ICLR 2021
S., F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural Circuit Synthesis from Specification Patterns. arXiv Preprint 2021

# Church's Problem

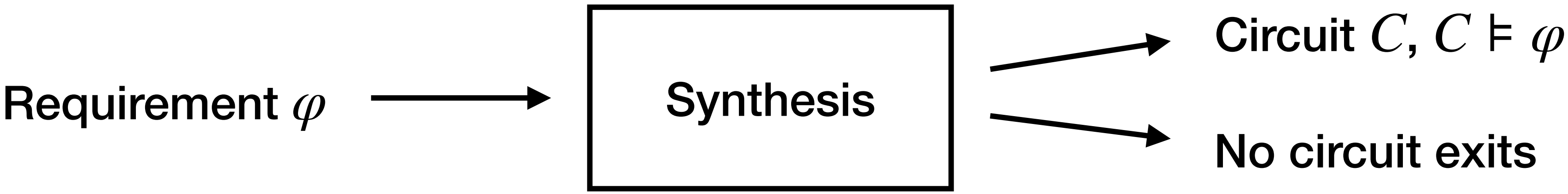
"Given a requirement which a circuit is to satisfy (…). The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit)."

**Alonzo Church, 1957**

# Church’s Problem

“Given a requirement which a circuit is to satisfy (…). The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit).”

**Alonzo Church, 1957**



# Church's Problem

"Given a requirement which a circuit is to satisfy (…). The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit)."

**Alonzo Church, 1957**

Requirement $\varphi$ → Synthesis → Circuit $C$, $C \models \varphi$

Synthesis → No circuit exits

# Classic LTL Synthesis

**Bounded Synthesis**[1]

**Game-based Synthesis**[2]

Specification

increase bound

Constraint System

Deterministic Automaton

Game on State Graph

Circuit / Unrealizable

The LTL Synthesis Problem is 2EXPTIME-complete[3].

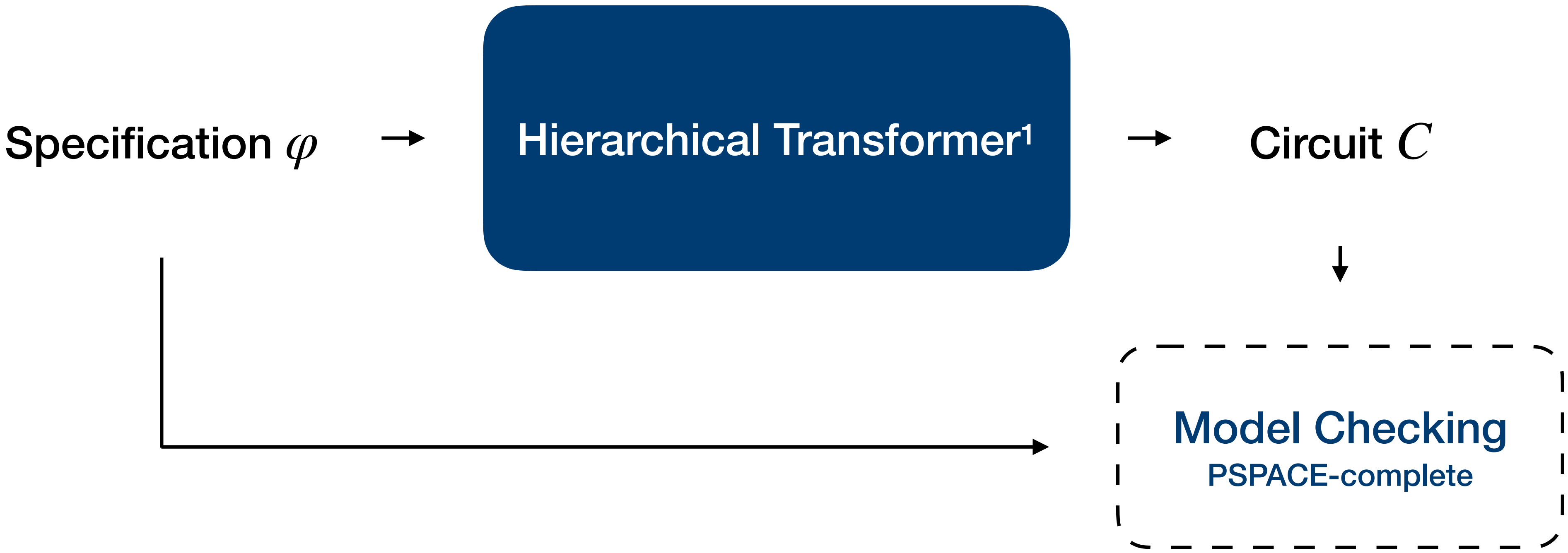
[1] Schewe, S., Finkbeiner, B.: Bounded Synthesis. ATVA 2007
[2] Büchi, J.R., Landweber, L.H.: Solving Sequential Conditions by Finite-State Strategies. Transactions of the American Mathematical Society Vol. 183 1969
[3] Pnueli, A., Rosner, R.: On the Synthesis of a Reactive Module. POPL 1989

# *Neural* LTL Synthesis

Specification $\varphi$ → Hierarchical Transformer[1] → Circuit $C$

[1] Li, W., Yu, L., Wu, Y., Paulson, L.C.: IsarStep: a Benchmark for High-level Mathematical Reasoning. ICLR 2021

# Neural LTL Synthesis




[1] Li, W., Yu, L., Wu, Y., Paulson, L.C.: IsarStep: a Benchmark for High-level Mathematical Reasoning. ICLR 2021

# Neural LTL Synthesis

## Data Generation from Specification Patterns

$$\begin{aligned} & \square(r_1 \Rightarrow \diamondsuit g_1) \\ \wedge\ & \square(r_2 \Rightarrow \diamondsuit g_2) \\ \wedge\ & \square \neg(g_1 \wedge g_2) \end{aligned}$$

Response

Response

Mutual Exclusion

$$\begin{aligned} AP &= I \cup O \\ I &= \{r_1, r_2\} \\ O &= \{g_1, g_2\} \end{aligned}$$

# Neural LTL Synthesis

## Data Generation from Specification Patterns

$$\square(r_1 \Rightarrow \Diamond g_1)$$ Response

$$\boxed{\wedge}\ \square(r_2 \Rightarrow \Diamond g_2)$$ Response

$$\boxed{\wedge}\ \square\neg(g_1 \wedge g_2)$$ Mutual Exclusion

$$AP = I \cup O$$
$$I = \{r_1, r_2\}$$
$$O = \{g_1, g_2\}$$

Conjunctions of smaller guarantees

# Neural LTL Synthesis

## Data Generation from Specification Patterns

$$\square(r_1 \Rightarrow \lozenge g_1)$$
$$\boxed{\wedge}\ \square(r_2 \Rightarrow \lozenge g_2)$$
$$\boxed{\wedge}\ \square\neg(g_1 \wedge g_2)$$

Response

Response

Mutual Exclusion

$$AP = I \cup O$$
$$I = \{r_1, r_2\}$$
$$O = \{g_1, g_2\}$$

Conjunctions of smaller guarantees

Frequent patterns

# Neural LTL Synthesis

## Data Generation from Specification Patterns

$$\square(r_1 \Rightarrow \bigcirc(\neg g_2 \mathsf{U} g_1))$$ Prioritized Response

$$\wedge\ \square(r_2 \Rightarrow \Diamond g_2)$$ Response

$$\wedge\ \square\neg(g_1 \wedge g_2)$$ Mutual Exclusion

$$AP = I \cup O$$
$$I = \{r_1, r_2\}$$
$$O = \{g_1, g_2\}$$

Conjunctions of smaller guarantees

Frequent patterns

# Neural LTL Synthesis

## Data Generation from Specification Patterns

$$\boxed{\Rightarrow}\quad \Box\Diamond\neg r_1$$

$$\Box(r_1 \Rightarrow \bigcirc(\neg g_2 \mathsf{U} g_1))$$

$$\boxed{\wedge}\ \Box(r_2 \Rightarrow \Diamond g_2)$$

$$\boxed{\wedge}\ \Box\neg(g_1 \wedge g_2)$$

Infinitely Often $\neg r_1$

Prioritized Response

Response

Mutual Exclusion

$$AP = I \cup O$$
$$I = \{r_1, r_2\}$$
$$O = \{g_1, g_2\}$$

Conjunctions of smaller guarantees

Frequent patterns

Assumptions

# Neural LTL Synthesis

## Data Generation from Specification Patterns

$\square \Diamond \neg r_1$ — Infinitely Often $\neg r_1$

$\boxed{\Rightarrow}$

$\square (r_1 \Rightarrow \bigcirc(\neg g_2 \mathsf{U} g_1))$ — Prioritized Response

$\boxed{\wedge}\ \square (r_2 \Rightarrow \Diamond g_2)$ — Response

$\boxed{\wedge}\ \square \neg(g_1 \wedge g_2)$ — Mutual Exclusion

$$AP = I \cup O$$
$$I = \{r_1, r_2\}$$
$$O = \{g_1, g_2\}$$

Conjunctions of smaller guarantees

Frequent patterns

Assumptions

$$\text{assumption}_1 \wedge \ldots \wedge \text{assumption}_m \Rightarrow \text{guarantee}_1 \wedge \ldots \wedge \text{guarantee}_n$$

# Neural LTL Synthesis

## Data Generation from Specification Patterns

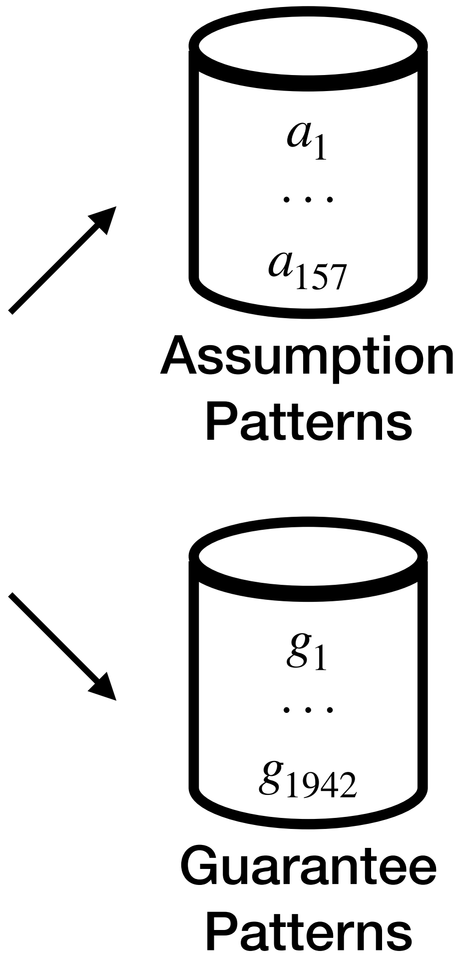
**346 SYNTCOMP[1] Specifications**

[1] http://www.syntcomp.org

# Neural LTL Synthesis

## Data Generation from Specification Patterns

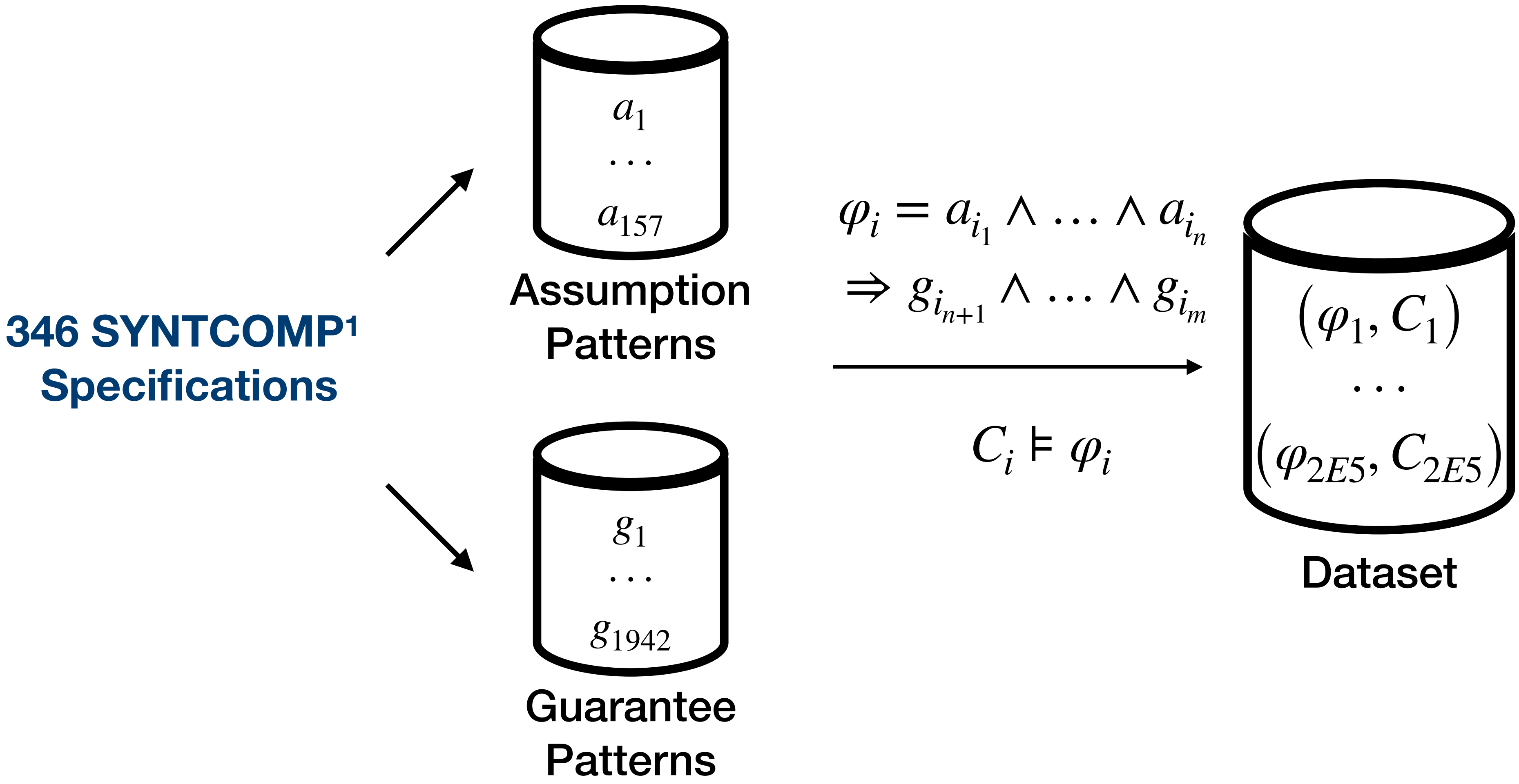
346 SYNTCOMP[1]
Specifications




[1] http://www.syntcomp.org

# Neural LTL Synthesis

## Data Generation from Specification Patterns




[1] http://www.syntcomp.org

# Neural LTL Synthesis

## Hierarchical Transformer[1]

AIGER Circuit $C$

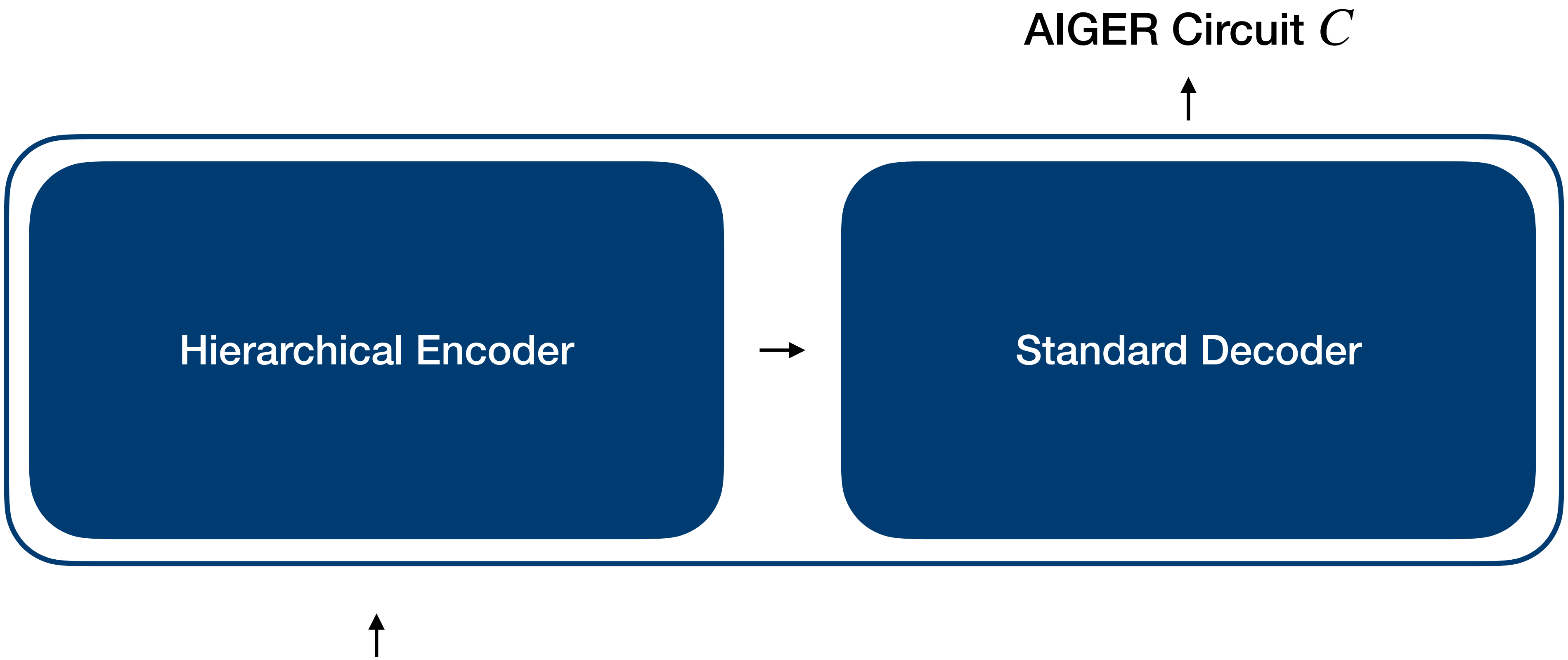
Hierarchical Transformer

$\square(r_1 \Rightarrow \Diamond g_1)$ $\square(r_2 \Rightarrow \Diamond g_2)$ $\square\neg(g_1 \wedge g_2)$

[1] Li, W., Yu, L., Wu, Y., Paulson, L.C.: IsarStep: a Benchmark for High-level Mathematical Reasoning. ICLR 2021

# Neural LTL Synthesis

## Hierarchical Transformer[1]

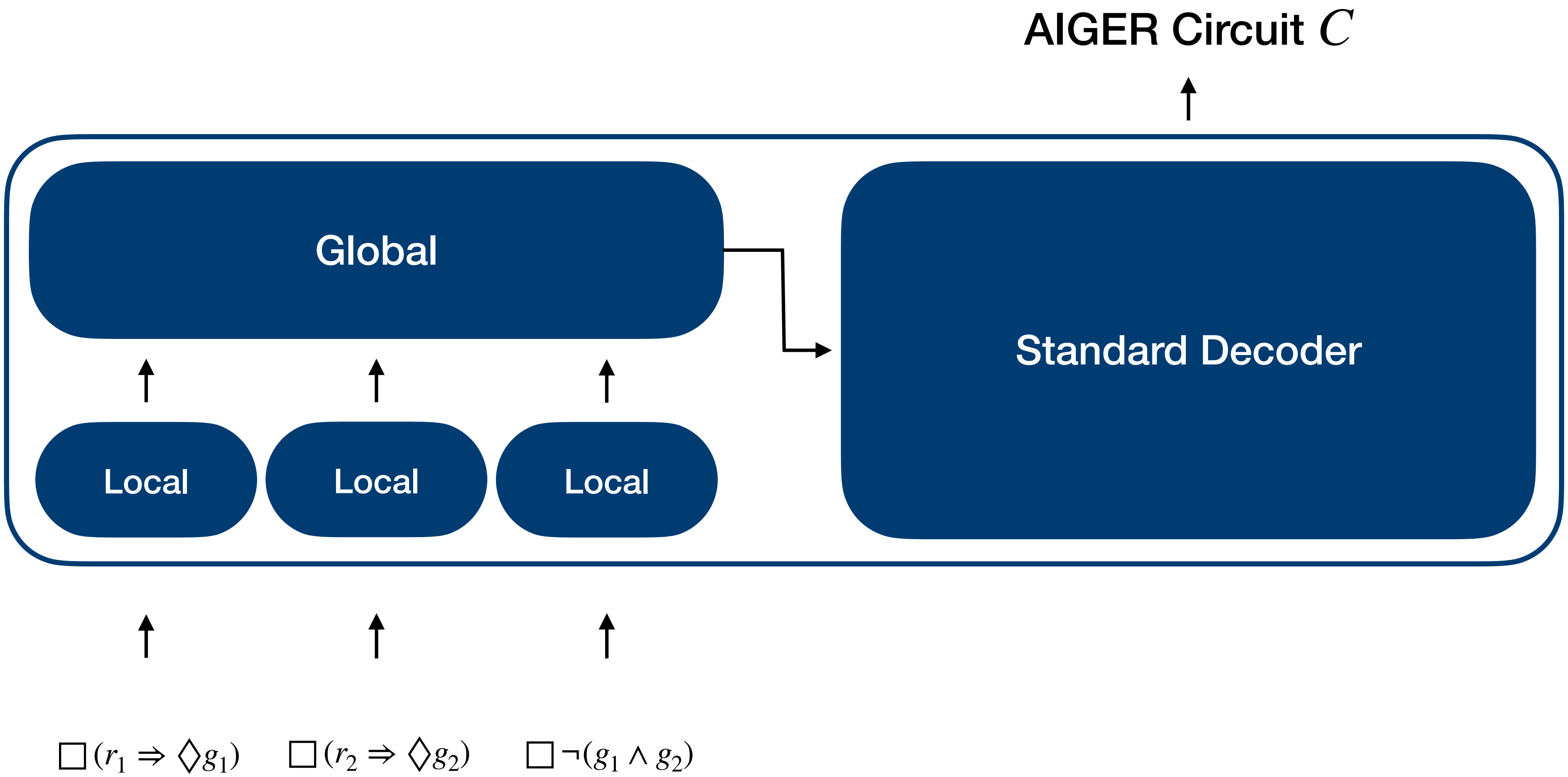



$\square(r_1 \Rightarrow \Diamond g_1) \quad \square(r_2 \Rightarrow \Diamond g_2) \quad \square\neg(g_1 \wedge g_2)$

[1] Li, W., Yu, L., Wu, Y., Paulson, L.C.: IsarStep: a Benchmark for High-level Mathematical Reasoning. ICLR 2021

# Neural LTL Synthesis

## Hierarchical Transformer[1]

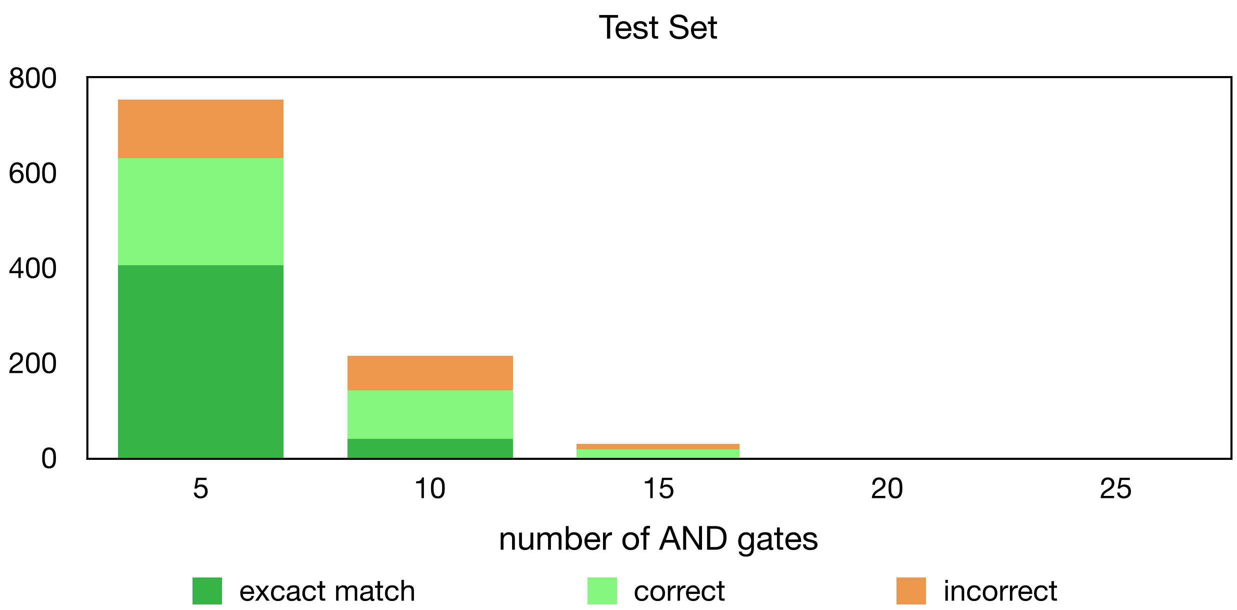



[1] Li, W., Yu, L., Wu, Y., Paulson, L.C.: IsarStep: a Benchmark for High-level Mathematical Reasoning. ICLR 2021

# Neural LTL Synthesis Results

## Test Set

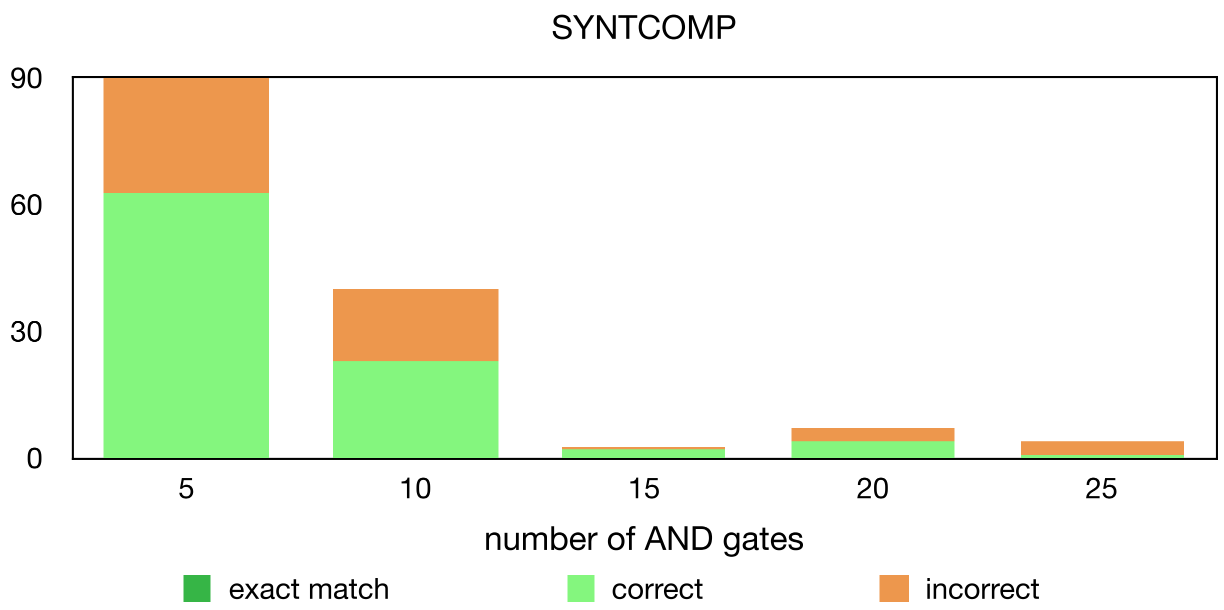
| | Beam Size 1 | Beam Size 4 | Beam Size 8 | Beam Size 16 |
|---|---|---|---|---|
| Test Set | 53.6 | 70.4 | 75.8 | 79.9 |



# Neural LTL Synthesis Results

## SYNTCOMP[1]

| | Beam Size 1 | Beam Size 4 | Beam Size 8 | Beam Size 16 |
|---|---|---|---|---|
| Test Set | 53.6 | 70.4 | 75.8 | 79.9 |
| SYNTCOMP | 51.9 | 60.0 | 63.6 | 66.8 |

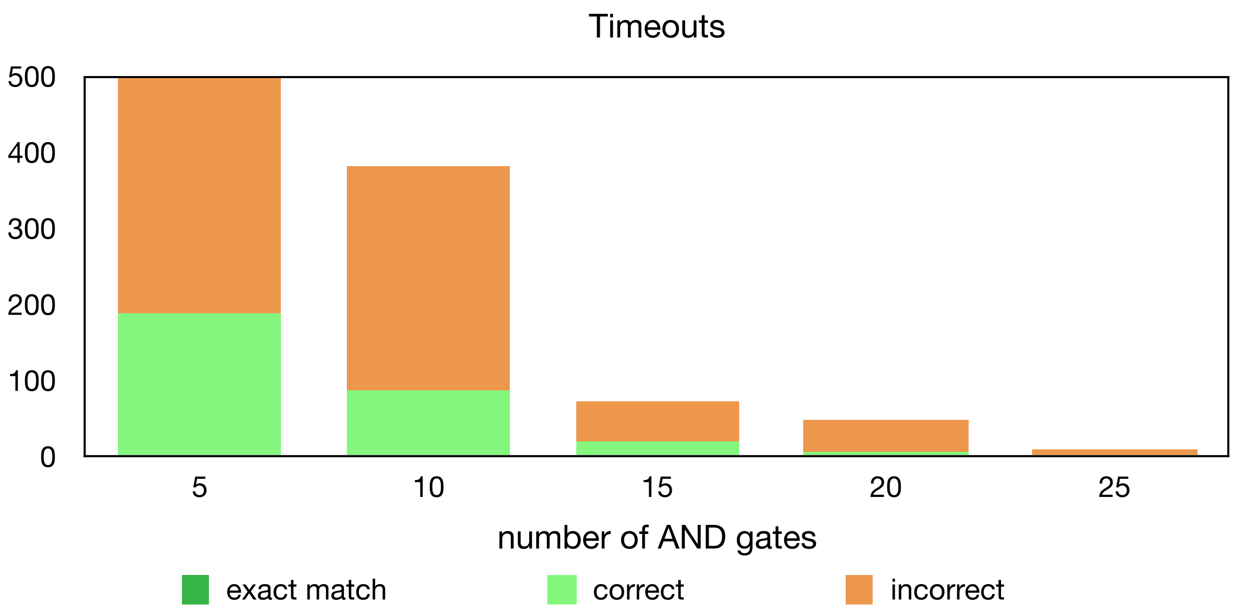



[1] http://www.syntcomp.org

# Neural LTL Synthesis Results

## Timeouts

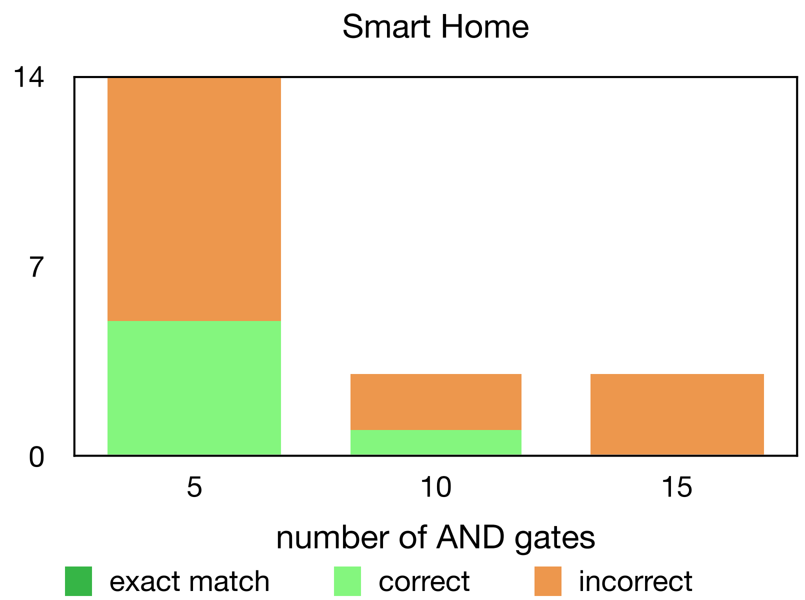
| | Beam Size 1 | Beam Size 4 | Beam Size 8 | Beam Size 16 |
|---|---|---|---|---|
| Test Set | 53.6 | 70.4 | 75.8 | 79.9 |
| SYNTCOMP | 51.9 | 60.0 | 63.6 | 66.8 |
| Timeouts | 11.7 | 21.1 | 25.9 | 30.1 |



# Neural LTL Synthesis Results

## Smart Home[1]

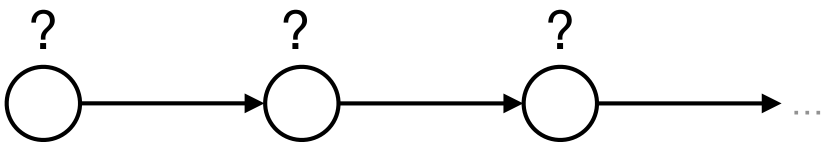
| | Beam Size 1 | Beam Size 4 | Beam Size 8 | Beam Size 16 |
|---|---|---|---|---|
| Test Set | 53.6 | 70.4 | 75.8 | 79.9 |
| SYNTCOMP | 51.9 | 60.0 | 63.6 | 66.8 |
| Timeouts | 11.7 | 21.1 | 25.9 | 30.1 |
| Smart Home | 22.9 | 31.4 | 44.8 | 40.0 |




[1] J.A.R.V.I.S. TSL/TLSF Benchmark Suite, 2021.

# Part 1: Trace Generation

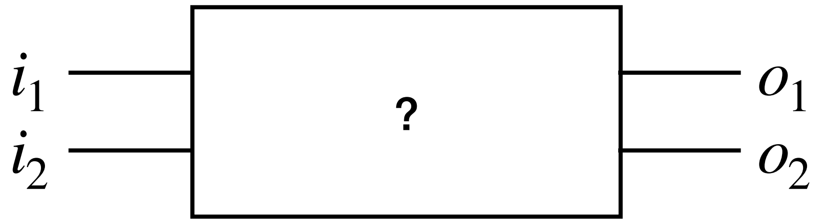
**Trace $\pi \models$ LTL Formula $\varphi$**




- Semantic generalization
- Generalization to larger formulas with tree positional encoding

# Part 2: Circuit Synthesis

**Circuit $C \models$ LTL Specification $\varphi$**

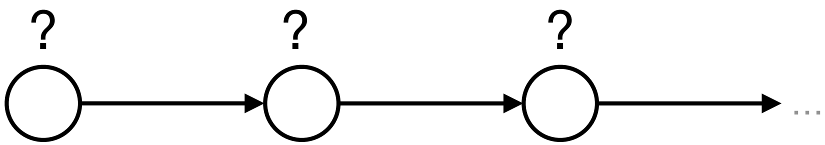



- Circuit synthesis end-to-end
- Generalizes to SYNTCOMP benchmarks

Hahn, C., S., F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.: Teaching Temporal Logics to Neural Networks. ICLR 2021
S., F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural Circuit Synthesis From Specification Patterns. arXiv Preprint 2021

# Part 1: Trace Generation

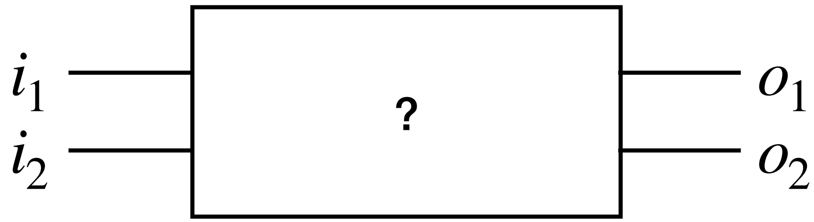
**Trace $\pi \models$ LTL Formula $\varphi$**




- Semantic generalization
- Generalization to larger formulas with tree positional encoding

# Part 2: Circuit Synthesis

**Circuit $C \models$ LTL Specification $\varphi$**




- Circuit synthesis end-to-end
- Generalizes to SYNTCOMP benchmarks

With deep learning new types of fast algorithms for verification and synthesis can be developed.

Hahn, C., S., F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.: Teaching Temporal Logics to Neural Networks. ICLR 2021
S., F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural Circuit Synthesis From Specification Patterns. arXiv Preprint 2021