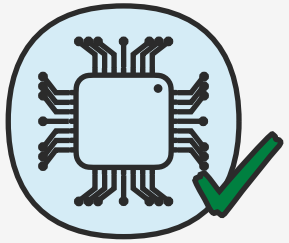
CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Neuro-Symbolic Methods for Reactive Synthesis, Repair, and Natural Language Formalization

Matthias Cosler | Oxford | November 17, 2023

Neuro-Symbolic Methods



System Correctness

Safety & Reliability

Security

Formal Guarantees

I. Formalization is difficult
II. Scalability Challenge

Image Generation

Robotics

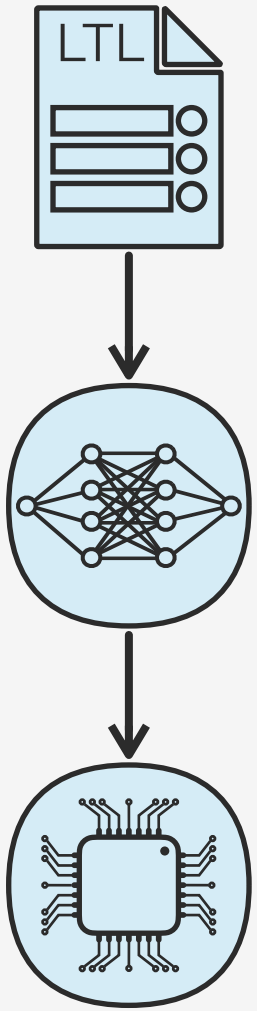
Natural Language Processing

Revolutionizing applications

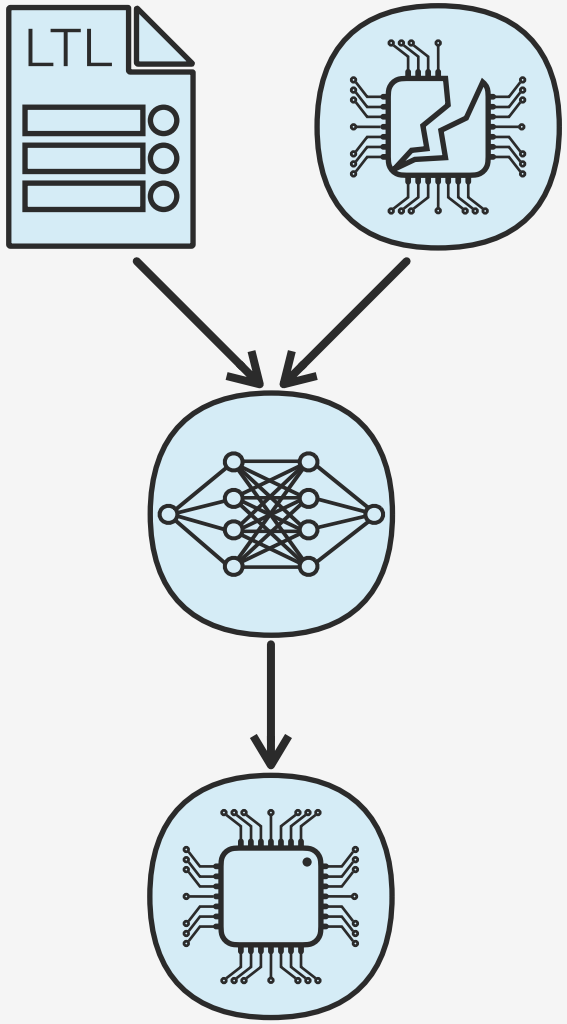
Can we combine the power of Deep Learning with guarantees from Formal Methods?

Neuro-Symbolic Methods for Temporal Logics

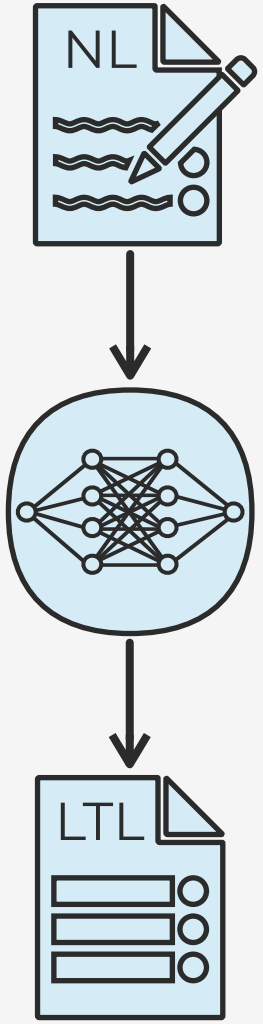
Neural Circuit Synthesis



Circuit Repair



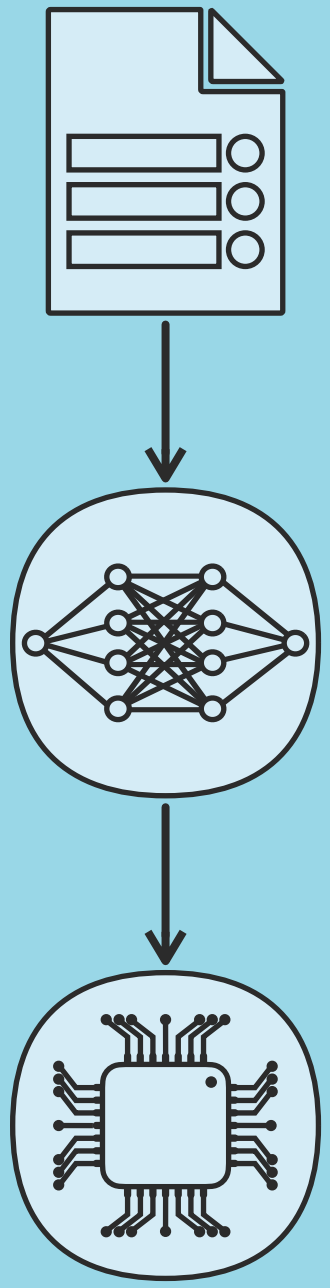
Natural Language to Temporal Logics



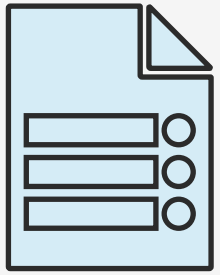
Neural Circuit Synthesis

F. Schmitt, C. Hahn, M. N. Rabe, and B. Finkbeiner, *Neural Circuit Synthesis from Specification Patterns*. NeurIPS 2021

M.C. , C. Hahn, A. Omar, and F. Schmitt, *NeuroSynt: A Neuro-symbolic Portfolio Solver for Reactive Synthesis*. (*under review*)



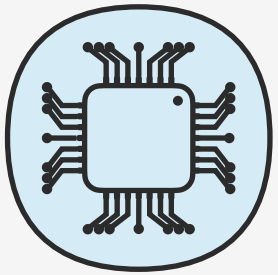
Reactive Synthesis



Assumptions & Guarantees φ
Linear-time Temporal Logic (LTL)

Globally → $\Box$ Eventually → $\Diamond$

$$\varphi = \Box\,(r \rightarrow \Diamond\, g)$$



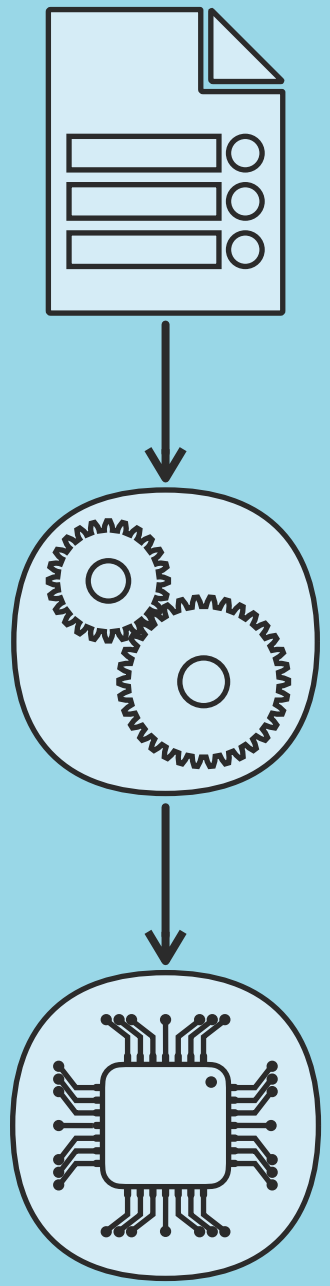
Circuit c
And-Inverter Graph

- Inputs
- Outputs
- Latches
- AND-gates
- Inverter

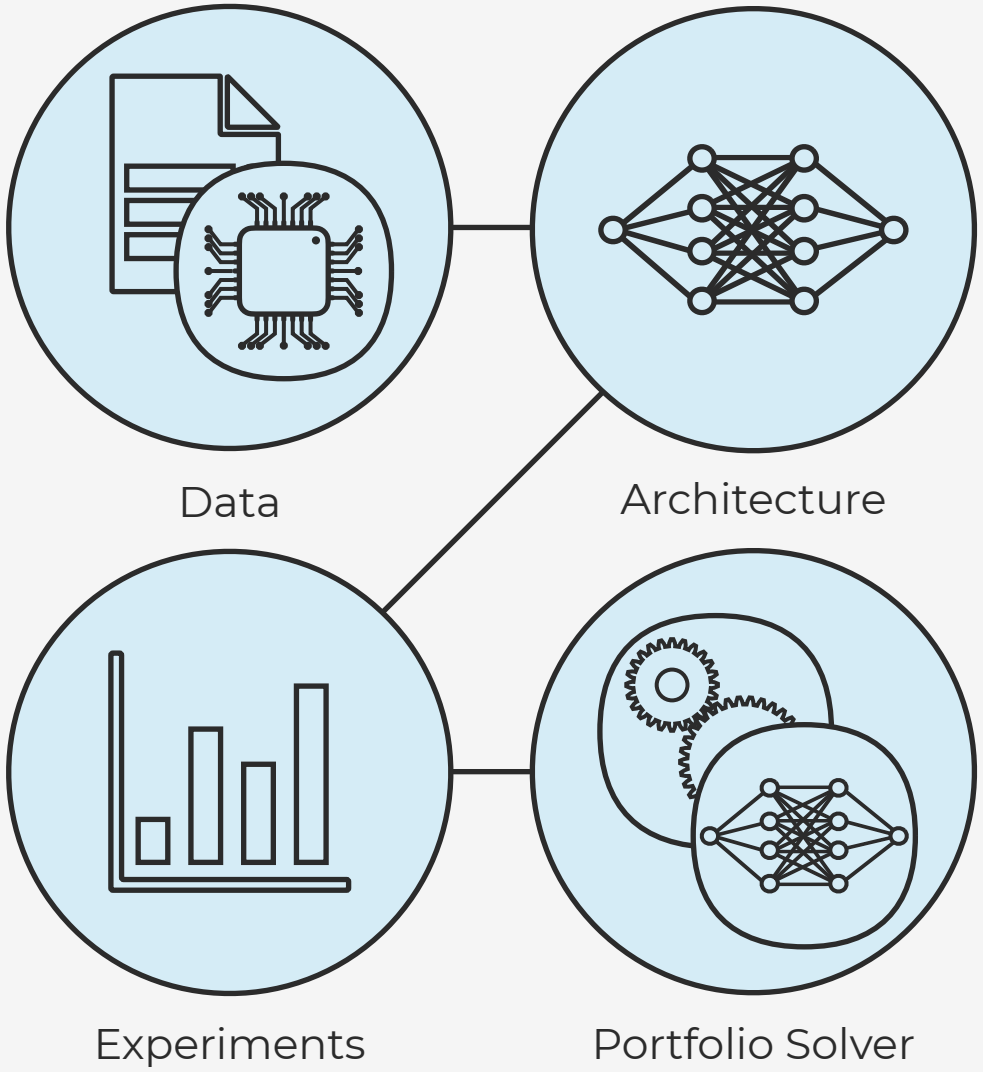
```
aag 12 5 2 5 5
...
14 24
...
22 14 12
24 23 17
```

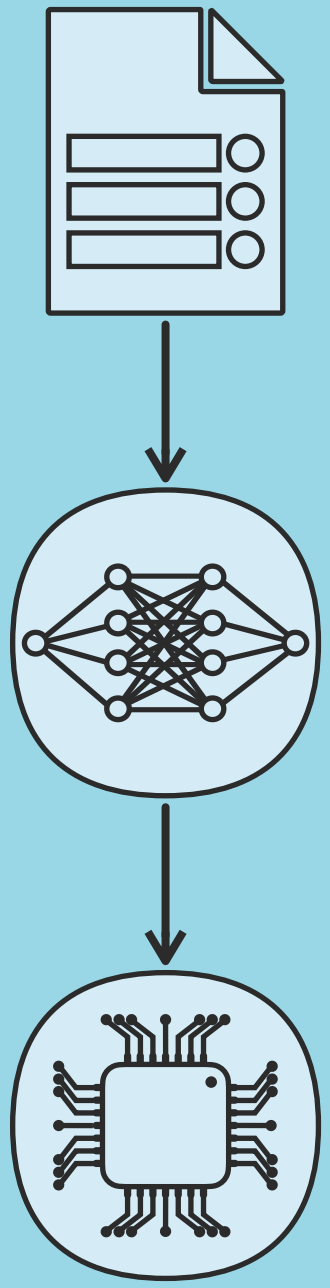
2-EXPTIME complete →

Construct c such that $c \models \varphi$



Neural Circuit Synthesis





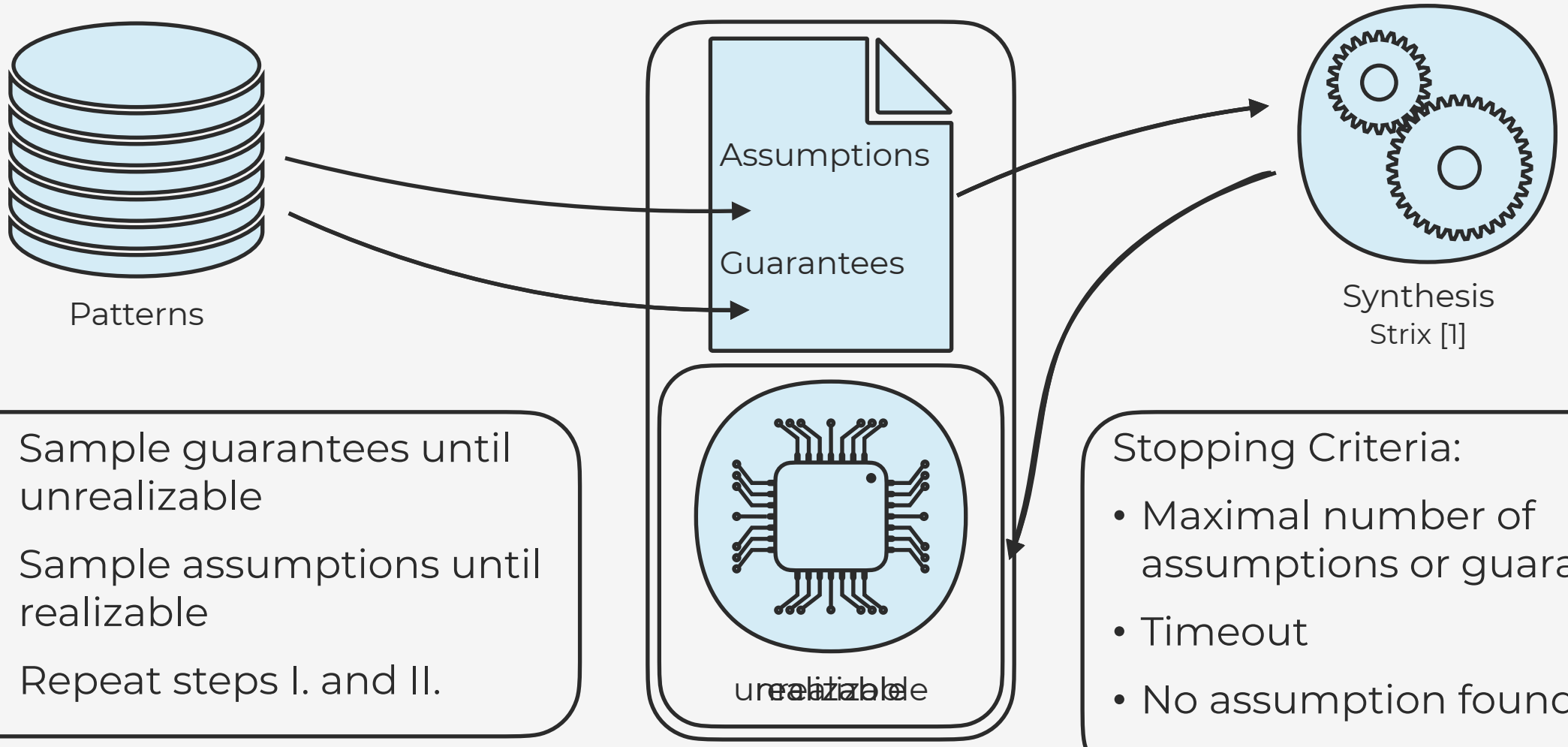
Data
Neural Circuit Synthesis

- Supervised training
- **Input**: assumptions & guarantees
- **Target**: circuit & realizable flag

assumptions	guarantees	realizable	circuit
	$\square\,(r_0 \rightarrow \diamondsuit\, g_0)$, $\square\,(r_1 \rightarrow \diamondsuit\, g_1)$, $\square\,(r_2 \rightarrow \diamondsuit\, g_2)$, $\square\,(r_3 \rightarrow \diamondsuit\, g_3)$, $\ldots$	yes	aag 12 5 2 5 5 ... 14 24 ... 22 14 13 24 23 17

Data Generation

Neural Circuit Synthesis



I. Sample guarantees until unrealizable

II. Sample assumptions until realizable

III. Repeat steps I. and II.

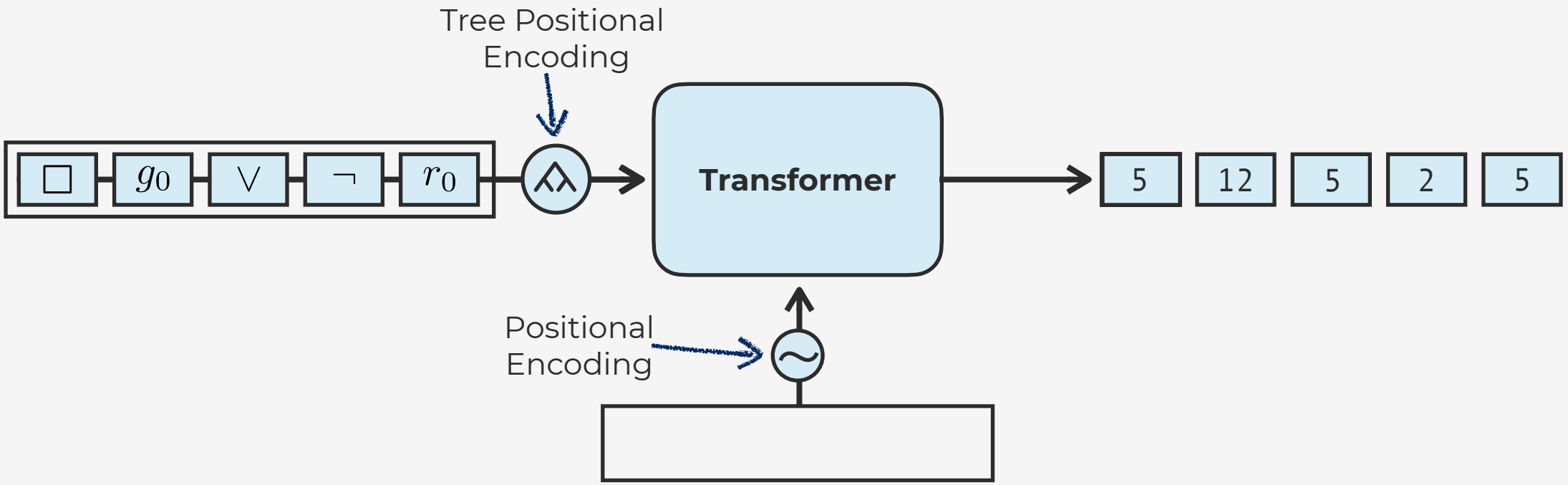
Stopping Criteria:

- Maximal number of assumptions or guarantees
- Timeout
- No assumption found

[1] P. J. Meyer, S. Sickert, and M. Luttenberger. *Strix: explicit reactive synthesis strikes back!* CAV 2018.

Architecture - Transformer

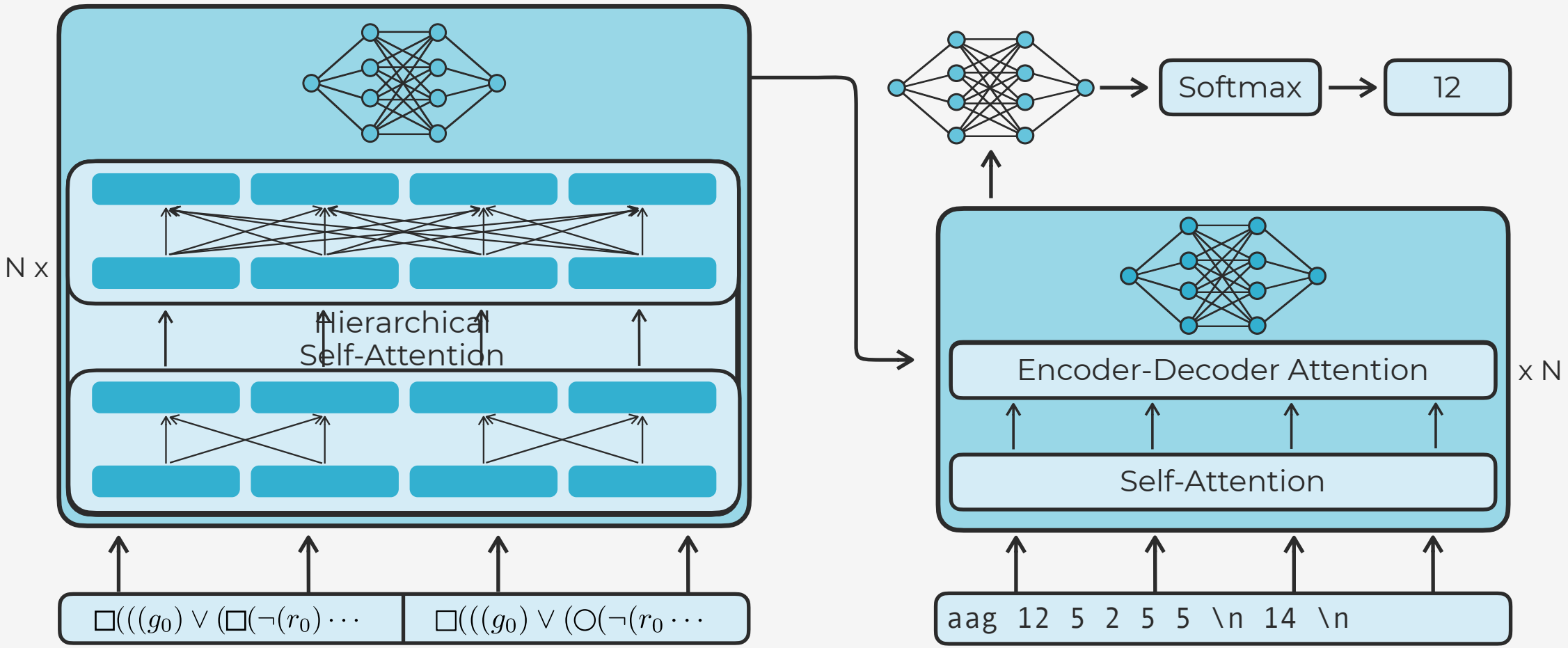
Neural Circuit Synthesis



- Sequence-to-Sequence Model
- Step-by-Step Prediction

Architecture - Hierarchical Transformer

Neural Circuit Synthesis



Experiments
Neural Circuit Synthesis

	test	large
syntactic accuracy	38.6%	10.2%
semantic accuracy	84.2%	57.7%

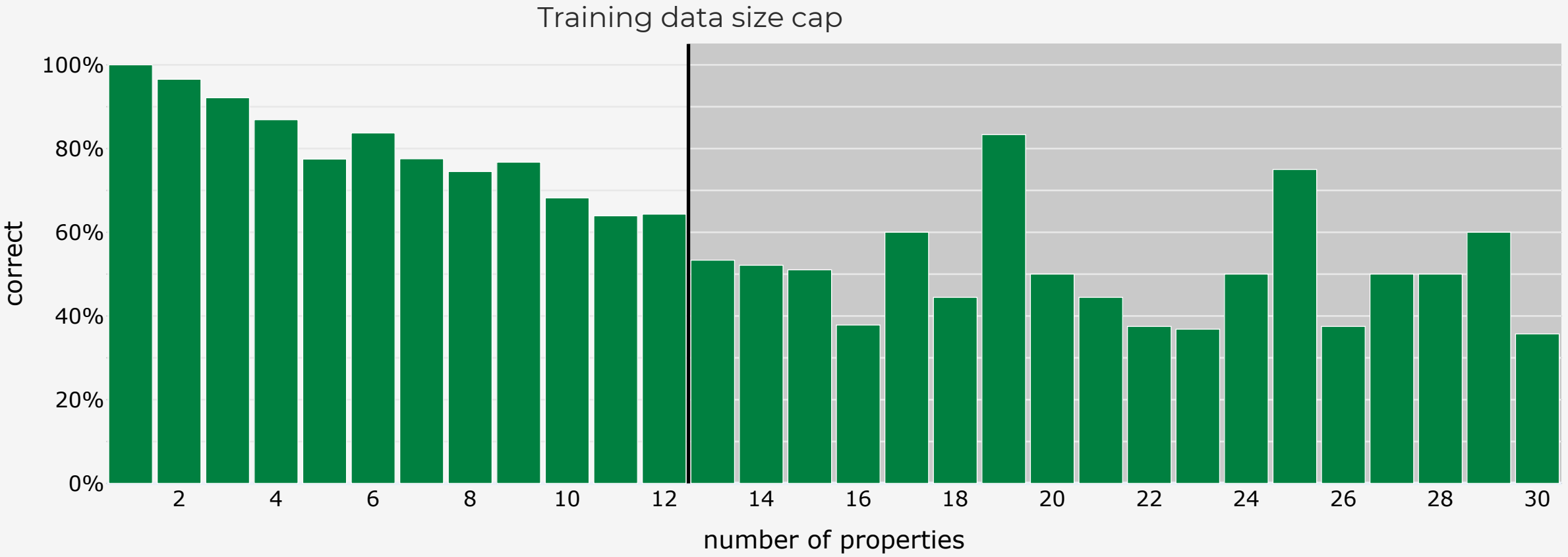
Generalization to alternative solutions

Generalization to larger specifications

* SYNTCOMP 2022 benchmark: small - training size constraints | large - evaluation size constraints | full - no constraints

Experiments - Size Generalization

Neural Circuit Synthesis



Generalization to more properties. Joined dataset of `test` and `large`

Experiments
Neural Circuit Synthesis

	test	large	timeouts	syntcomp-small*	syntcomp-large*	syntcomp-full*
syntactic accuracy	38.6%	10.2%				
semantic accuracy	84.2%	57.7%	33 %	65.8%	54.5%	34.8%

Generalization to alternative solutions

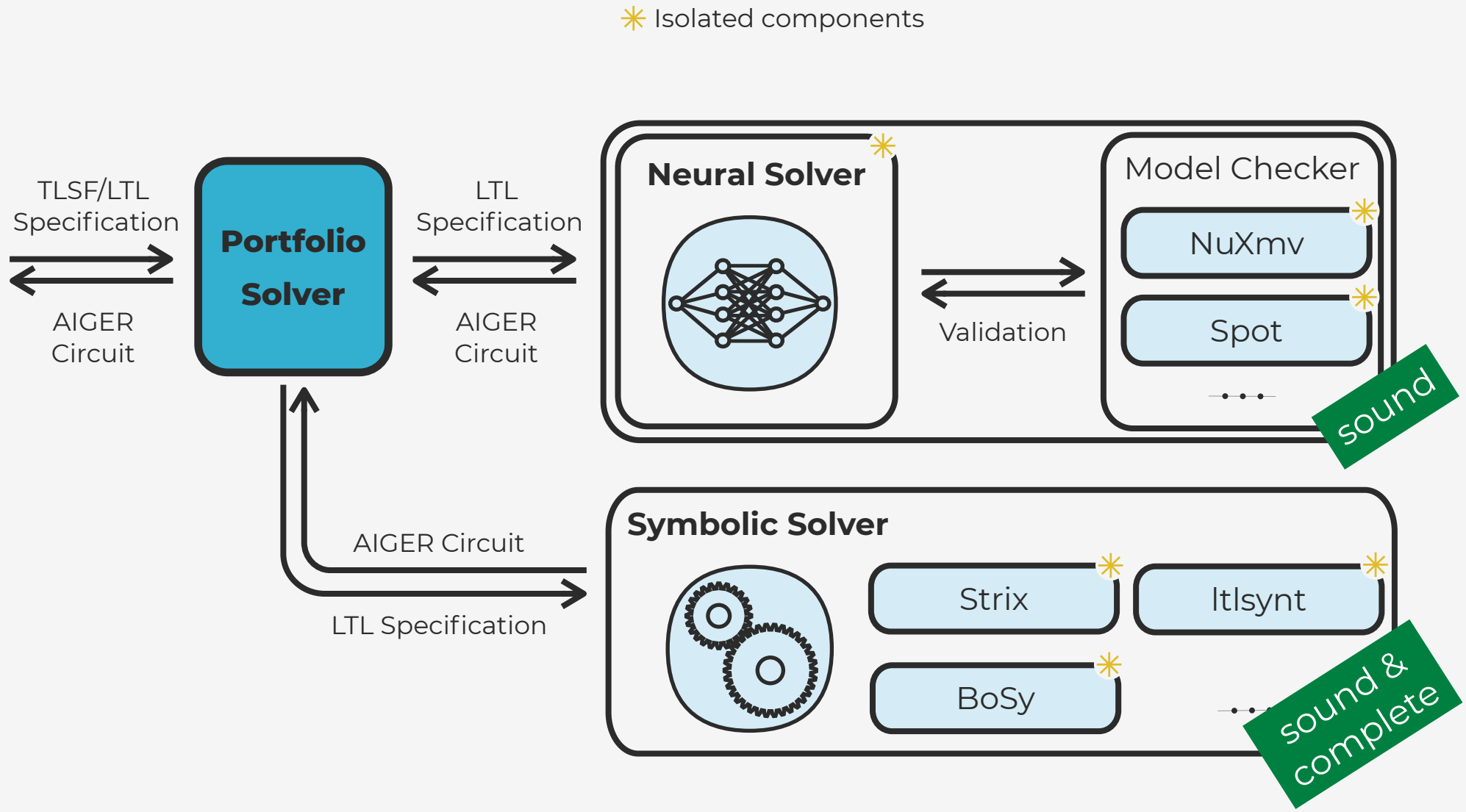
Generalization to larger specifications

Generalization to harder specifications

Generalization to out-of-distribution benchmarks

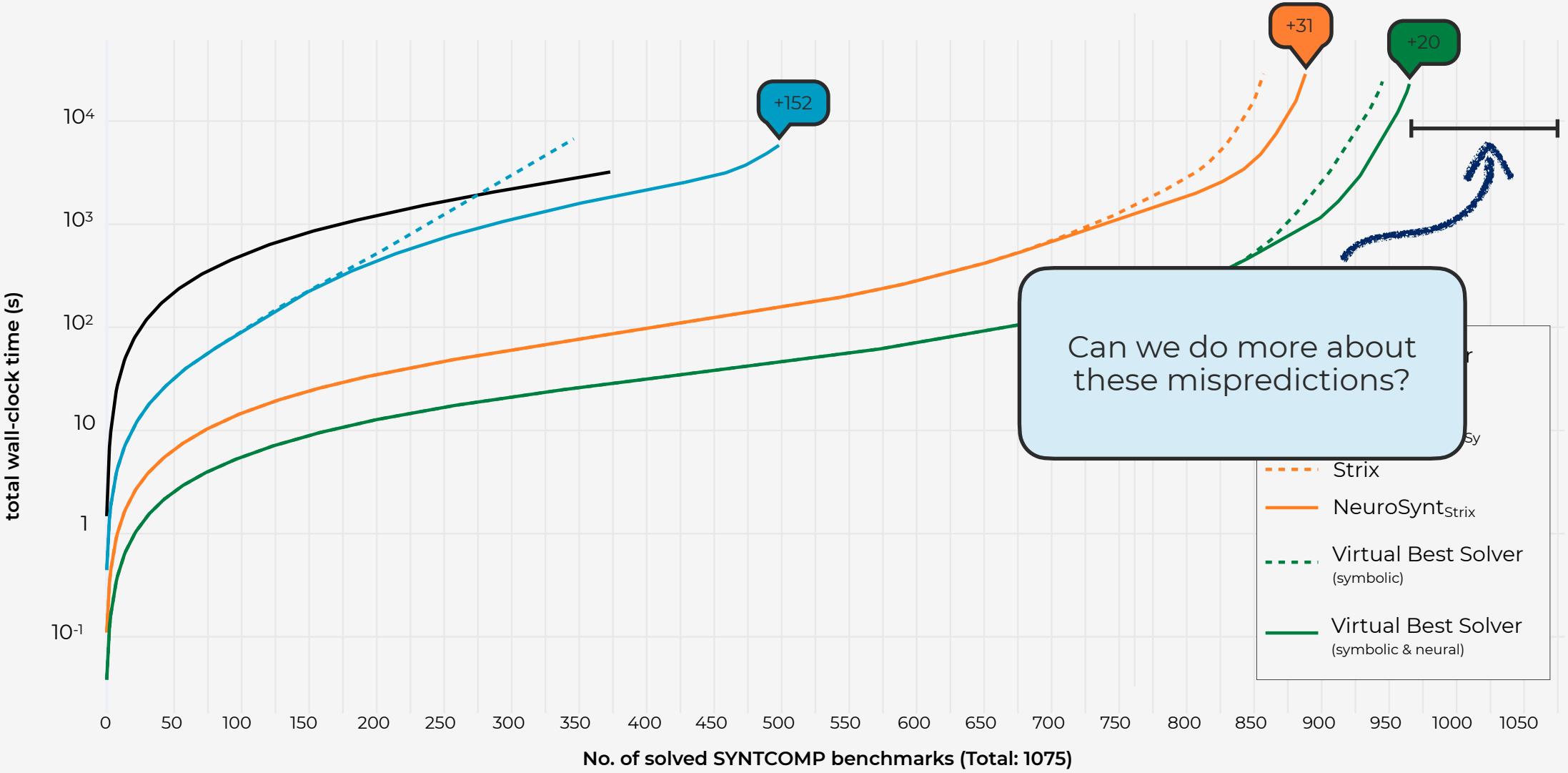
* SYNTCOMP 2022 benchmark: small - training size constraints | large - evaluation size constraints | full - no constraints

Portfolio Solver
Neural Circuit Synthesis



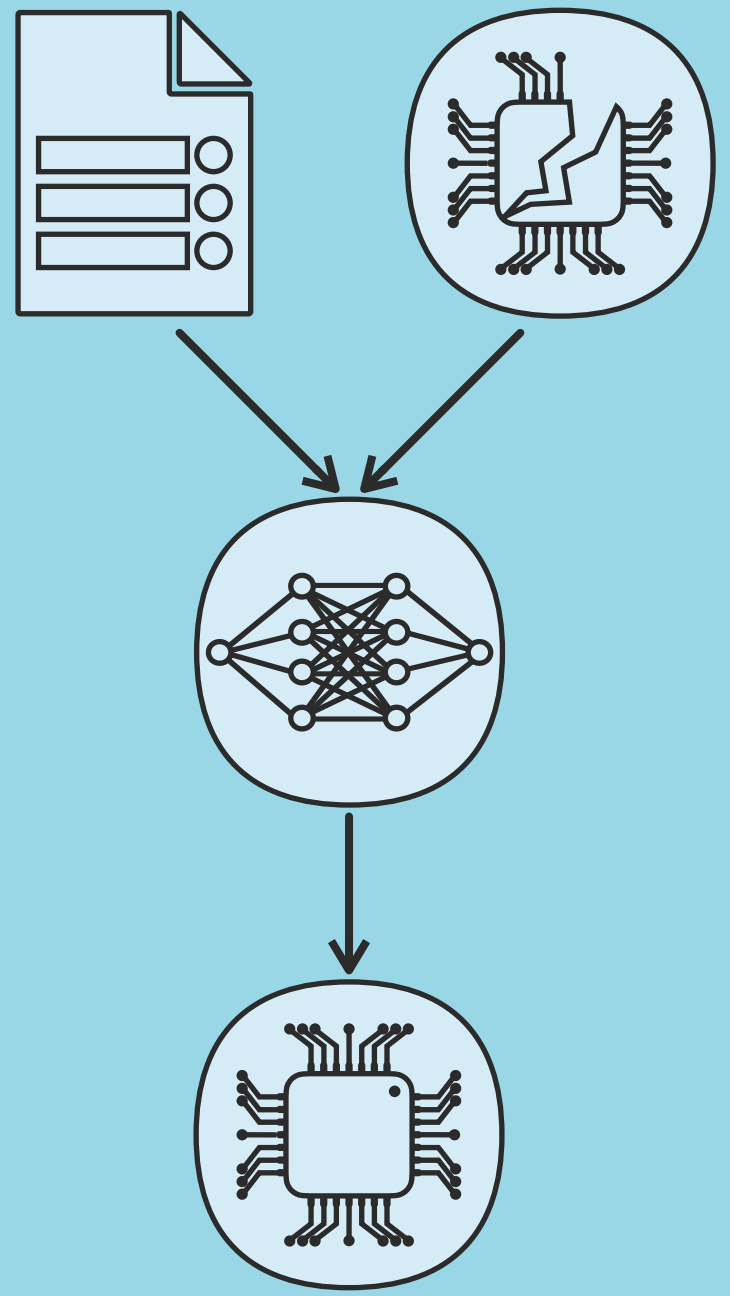
Portfolio Solver Evaluation

Neural Circuit Synthesis



Circuit Repair

M. C., F. Schmitt, C. Hahn, and B. Finkbeiner. Iterative Circuit Repair Against Formal Specifications. ICLR 2023



Circuit Repair

Assumptions & Guarantees φ
Linear-time Temporal Logic (LTL)

Circuit c'
And-Inverter Graph (AIGER)

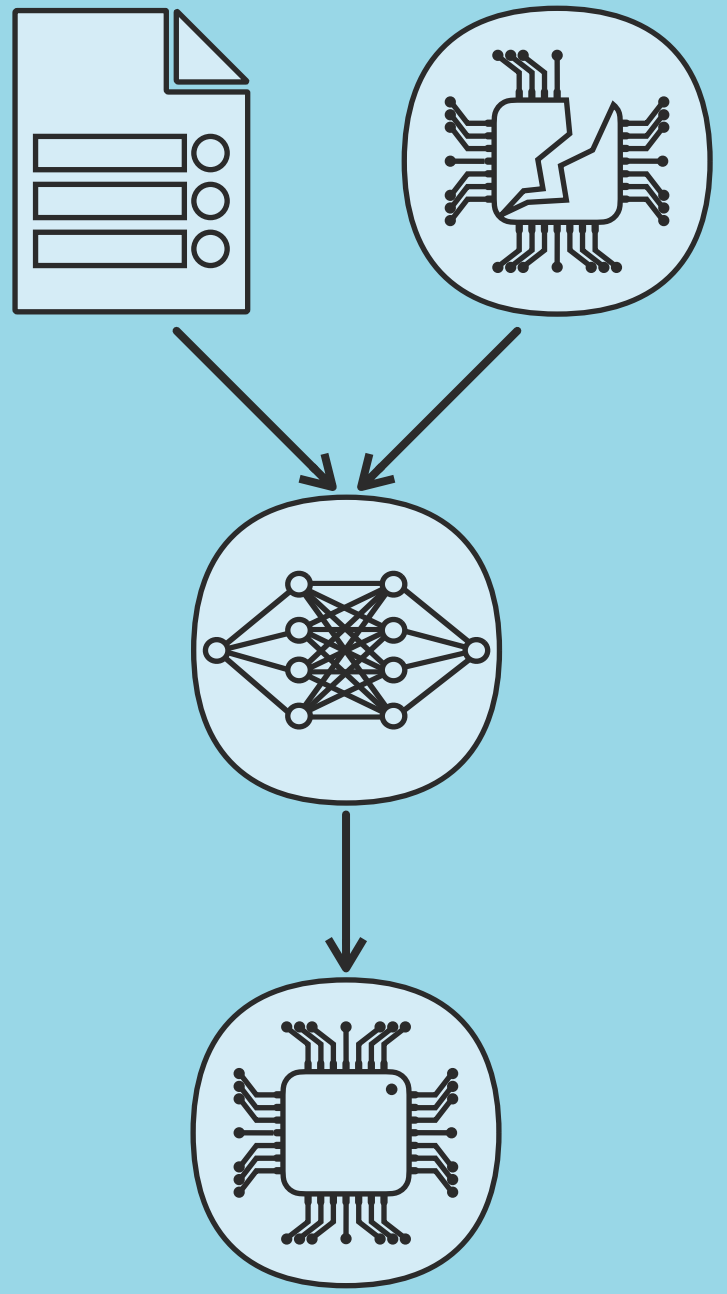
Faulty Circuit c
And-Inverter Graph (AIGER)

- Inputs
- Outputs
- Latches
- AND-gates
- Inverter

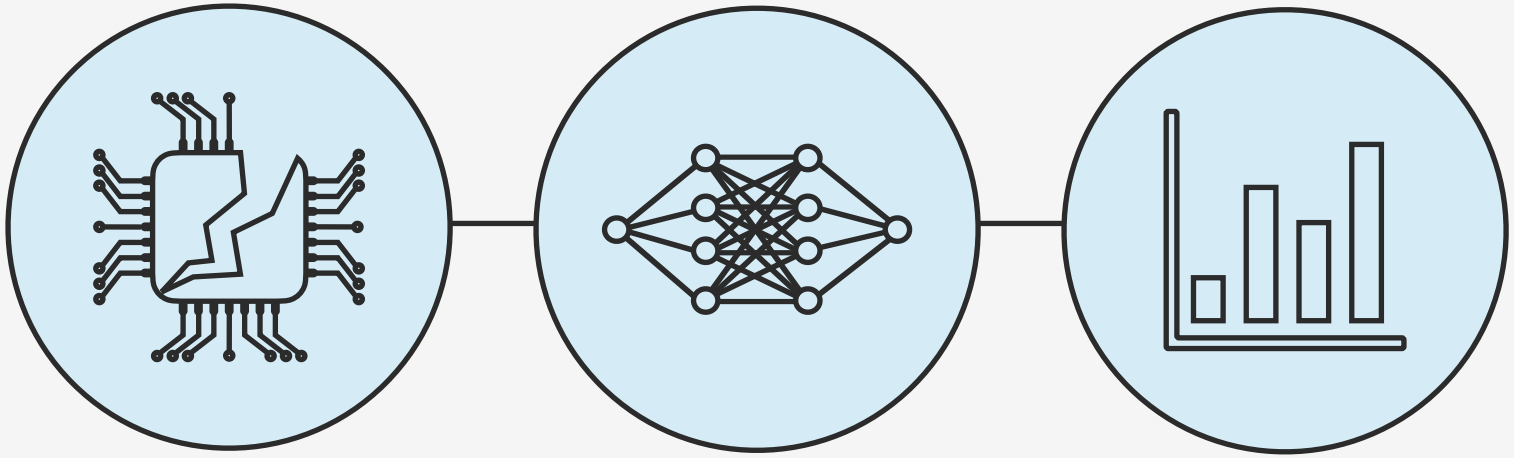
```
aag 12 5 2 5 5
...
14 24
...
22 14 12
24 23 17
```

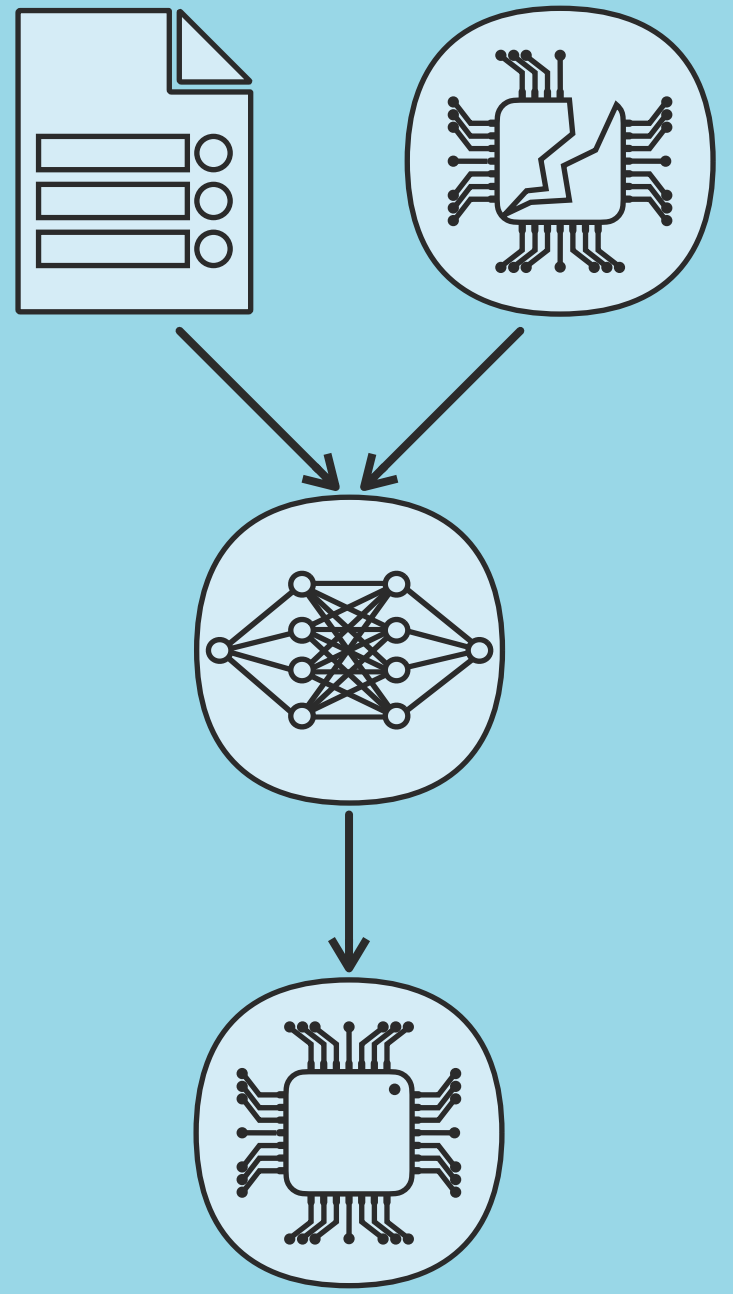
- $c \not\models \varphi$
- c might not be syntactically valid

Construct c' from c such that $c' \models \varphi$



Circuit Repair





Data
Circuit Repair

- Supervised training
- **Input**: assumptions & guarantees
- **Target**: circuit & realizable flag

assumptions	guarantees	realizable	target circuit
	$\square\,(r_0 \rightarrow \diamondsuit\, g_0)$, $\square\,(r_1 \rightarrow \diamondsuit\, g_1)$, $\square\,(r_2 \rightarrow \diamondsuit\, g_2)$, $\square\,(r_3 \rightarrow \diamondsuit\, g_3)$, ...	yes	aag 12 5 2 5 5 ... 14 24 ... 22 14 13 24 23 17

Data
Circuit Repair

- Supervised training
- **Input**: assumptions & guarantees & faulty circuit
- **Target**: circuit & realizable flag

assumptions	guarantees	faulty circuit	realizable	target circuit
	$\square(r_0 \rightarrow \diamondsuit g_0)$, $\square(r_1 \rightarrow \diamondsuit g_1)$, $\square(r_2 \rightarrow \diamondsuit g_2)$, $\square(r_3 \rightarrow \diamondsuit g_3)$, ...	aag 11 5 2 5 4 ... 14 24 ... 22 19 17	yes	aag 12 5 2 5 5 ... 14 24 ... 22 14 13 24 23 17

Data Generation
Circuit Repair

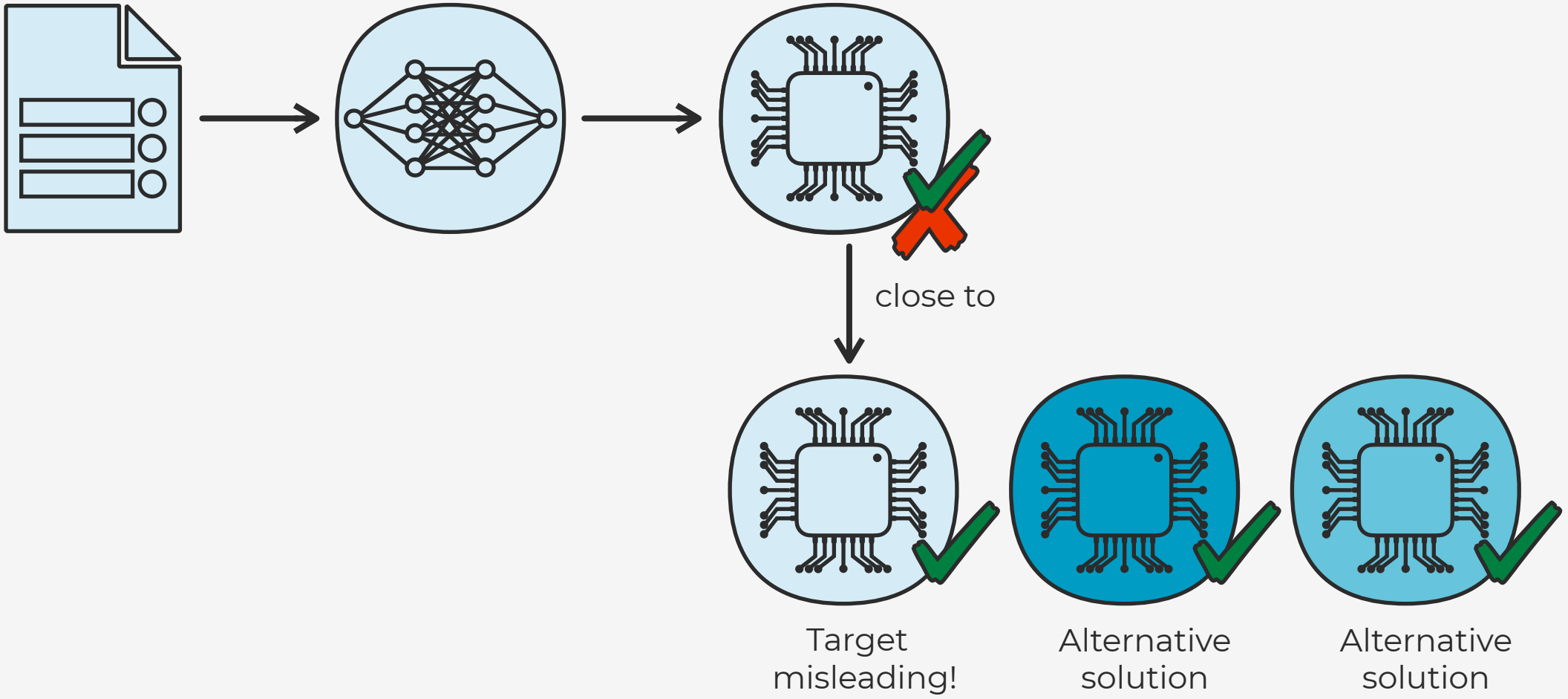
- Fault-injection algorithm
- Collecting mispredictions of Neural Circuit Synthesis

assumptions	guarantees	faulty circuit	realizable	target circuit

	$\square\,(r_0 \rightarrow \diamondsuit\, g_0)$,
	$\square\,(r_1 \rightarrow \diamondsuit\, g_1)$,
	$\square\,(r_2 \rightarrow \diamondsuit\, g_2)$,
	$\square\,(r_3 \rightarrow \diamondsuit\, g_3)$,
	$\ldots$
		`aag 11 5 2 5 4`
		`...`
		`14 24`
		`...`
		`22 19 17`
			yes	`aag 12 5 2 5 5`
				`...`
				`14 24`
				`...`
				`22 14 13`
				`24 23 17`

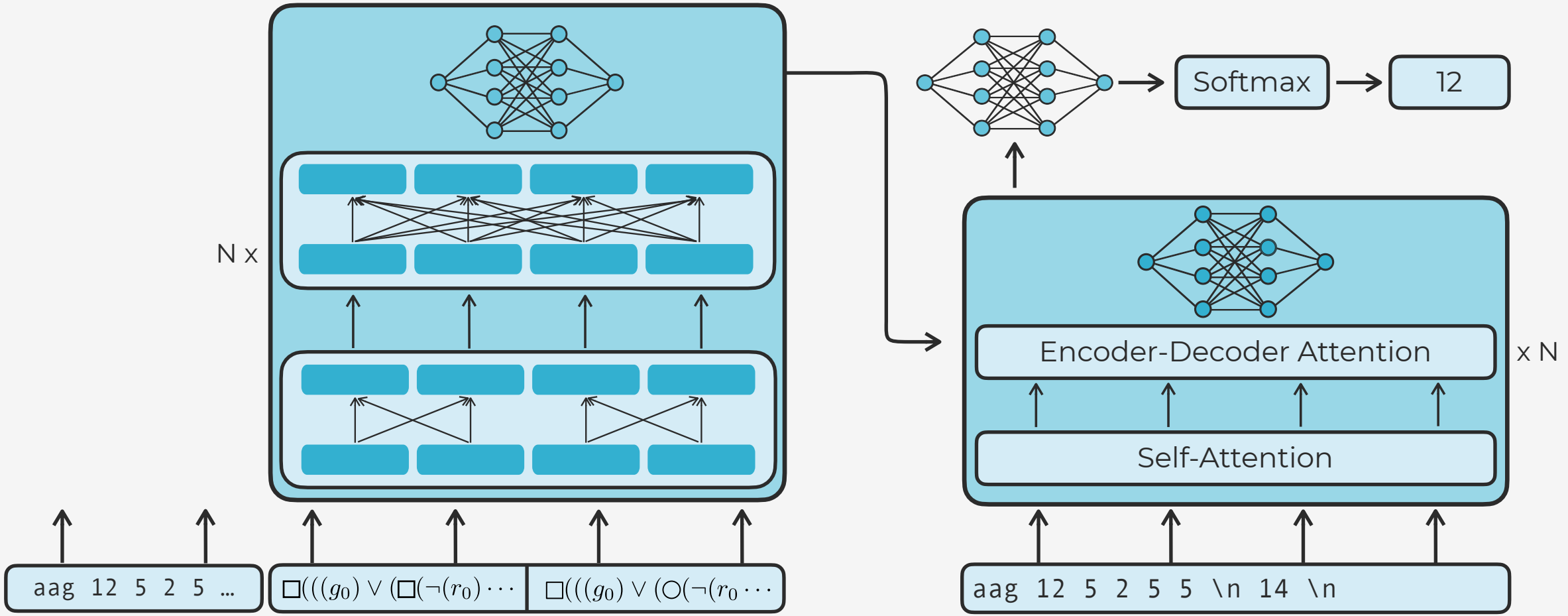
Data Generation - Collecting Mispredictions

Circuit Repair



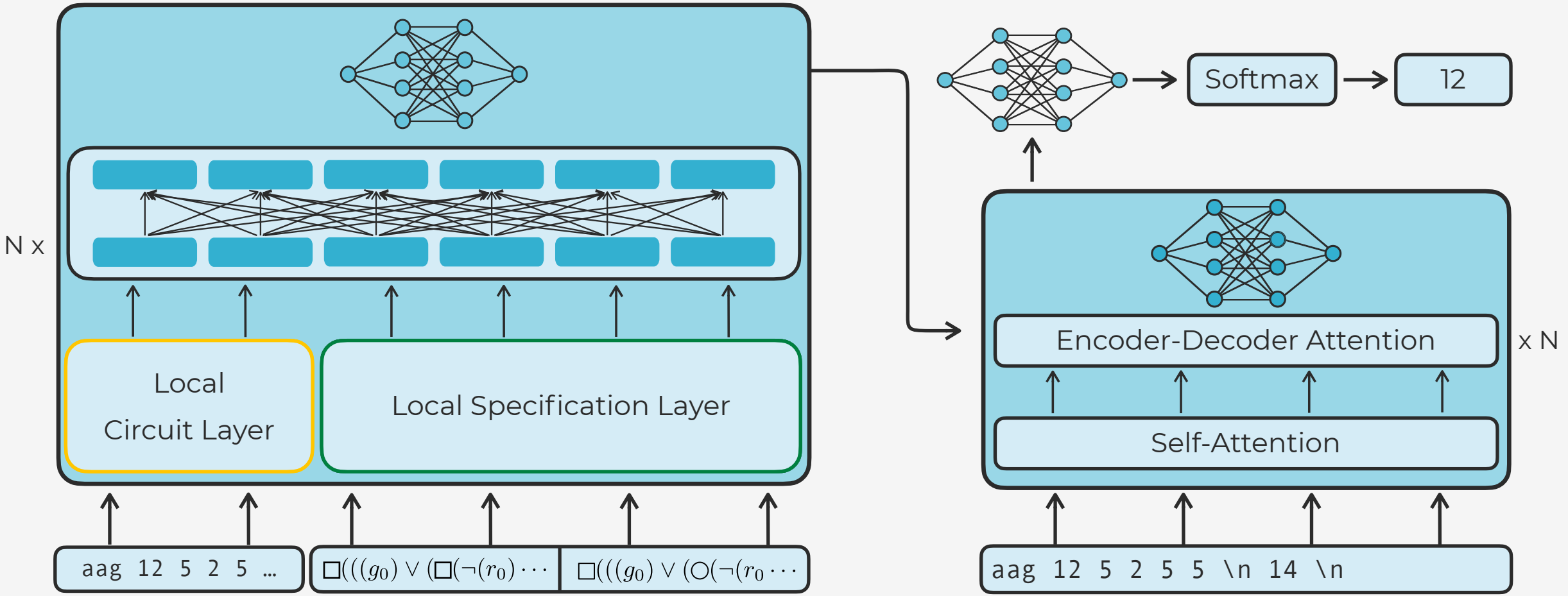
Architecture - Separated Hierarchical Transformer

Circuit Repair



Architecture - Separated Hierarchical Transformer

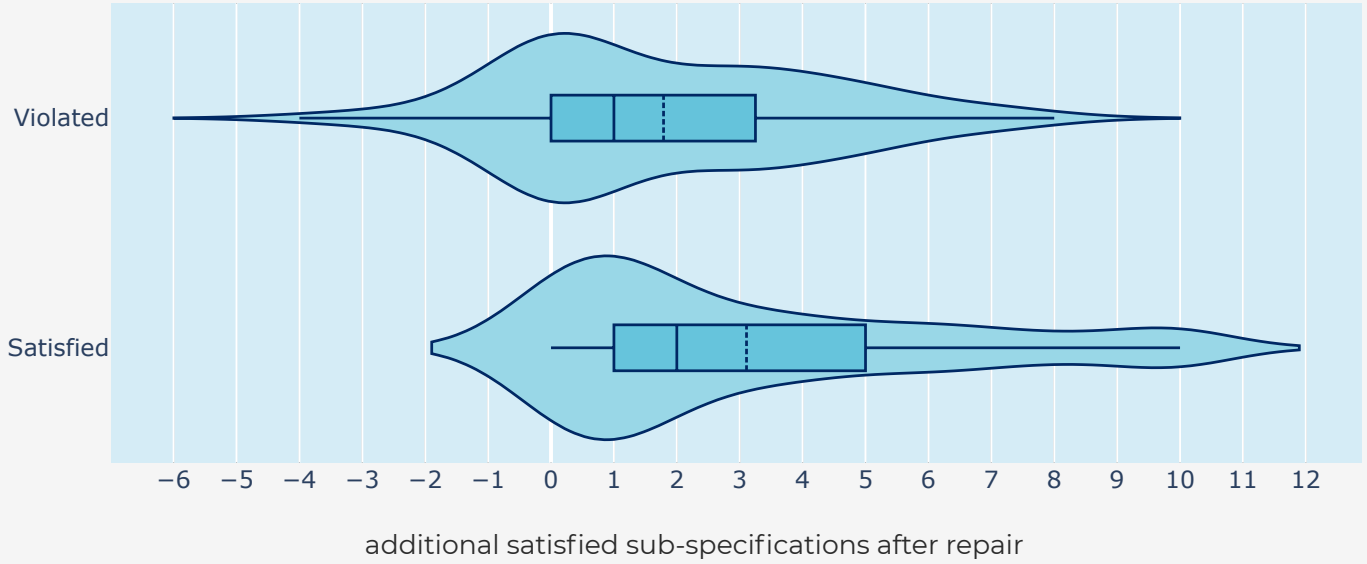
Circuit Repair

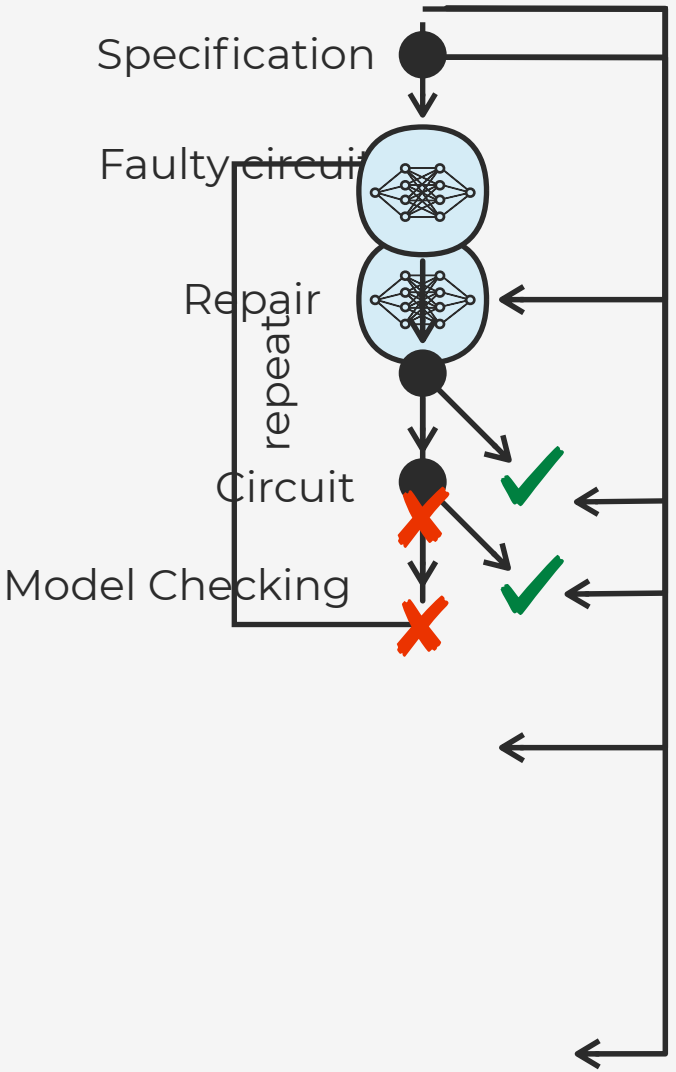


Experiments - Repair

Circuit Repair

Correct predictions: **84.2%**



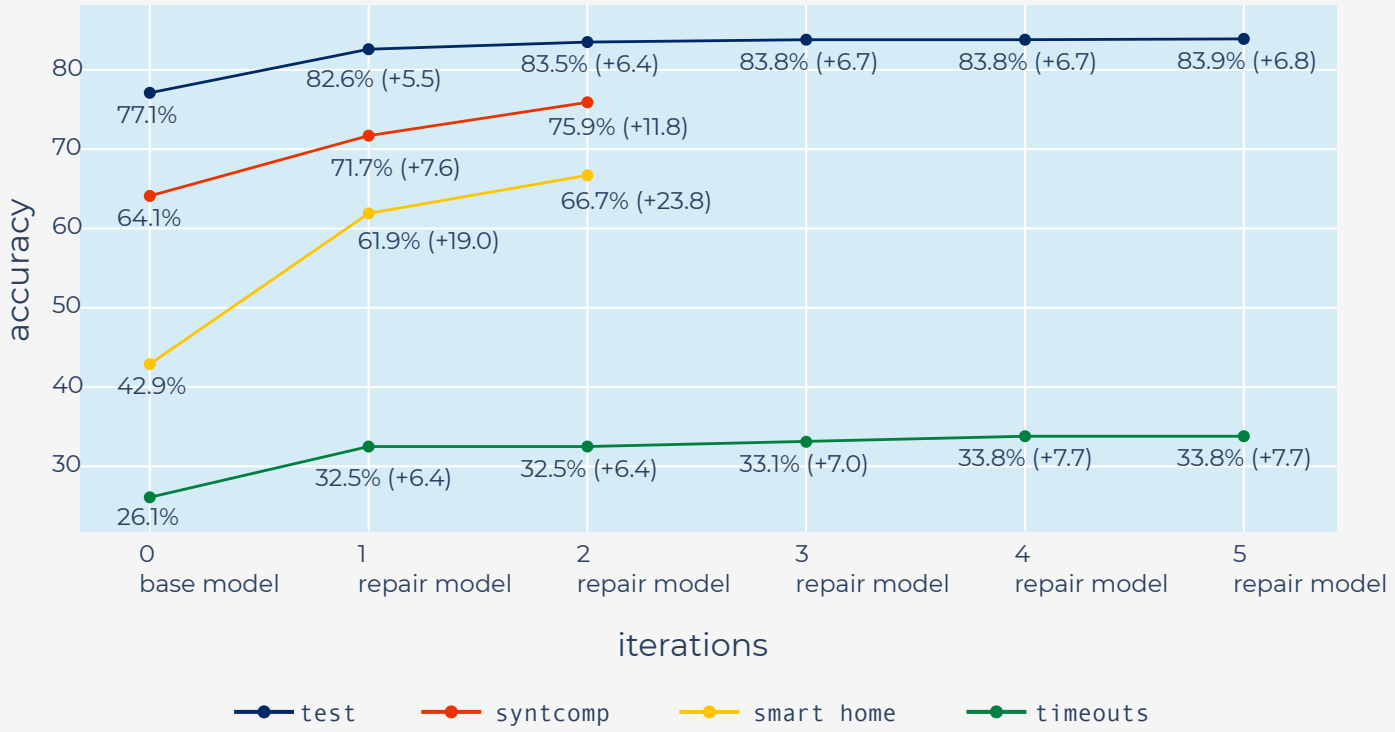


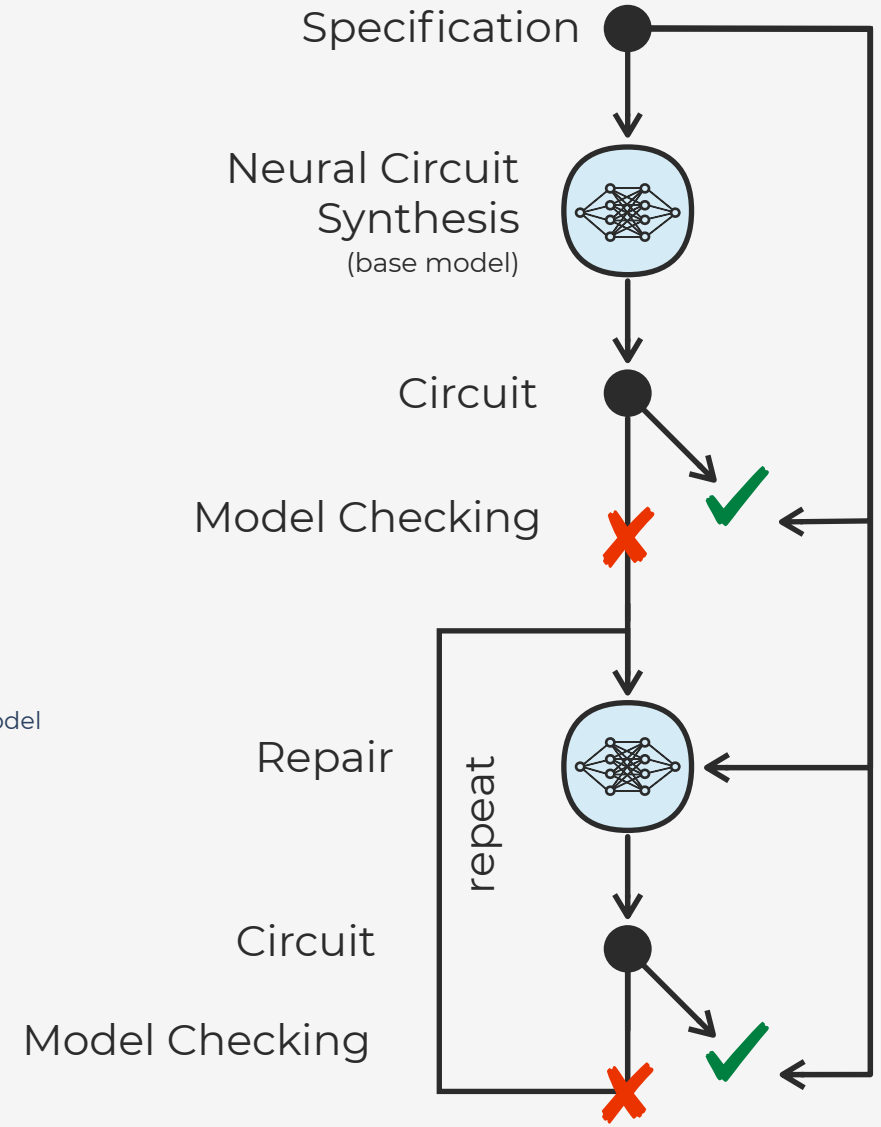
Wrong predictions are less faulty than before!

Correct predictions after 4 repetitions: **87.5% (+3.3)**

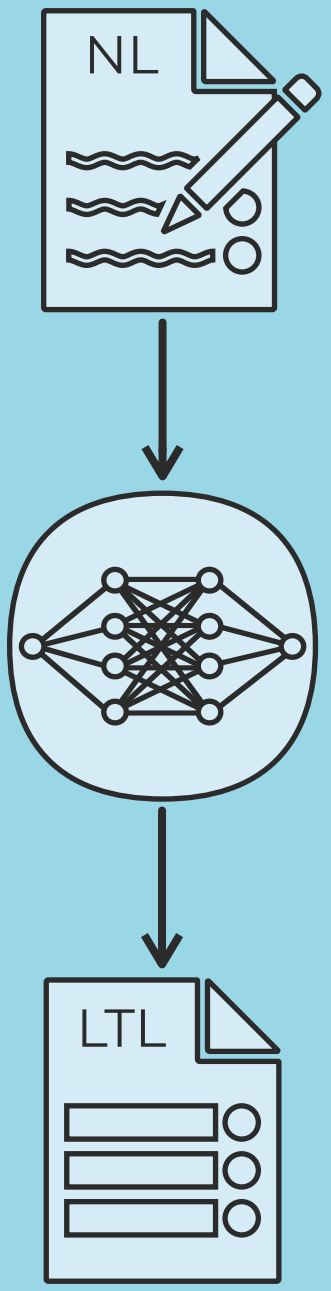
Experiments - Synthesis

Circuit Repair





Natural Language to Temporal Logics



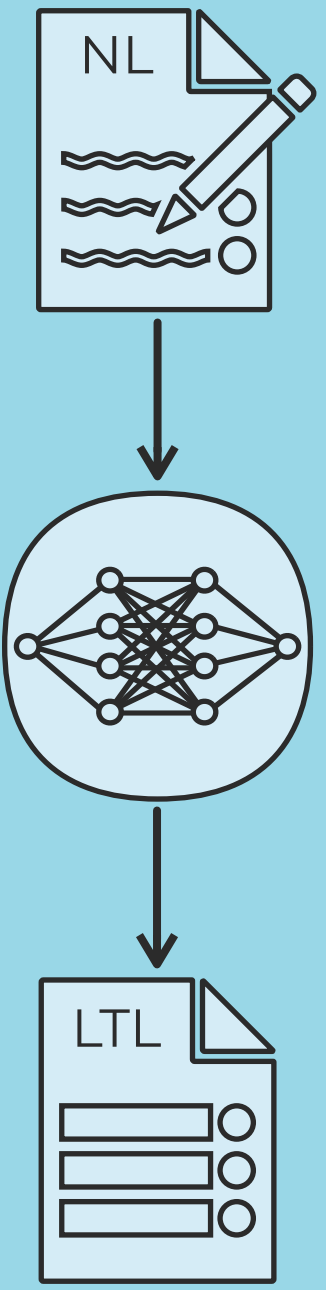
M. C., C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel. *nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models*. CAV 2023

Natural Language to Temporal Logics

The vehicle should maintain the set speed when cruise control is activated.

$$\square(cruise\_control \rightarrow maintain\_speed)$$

Natural Language is unstructured

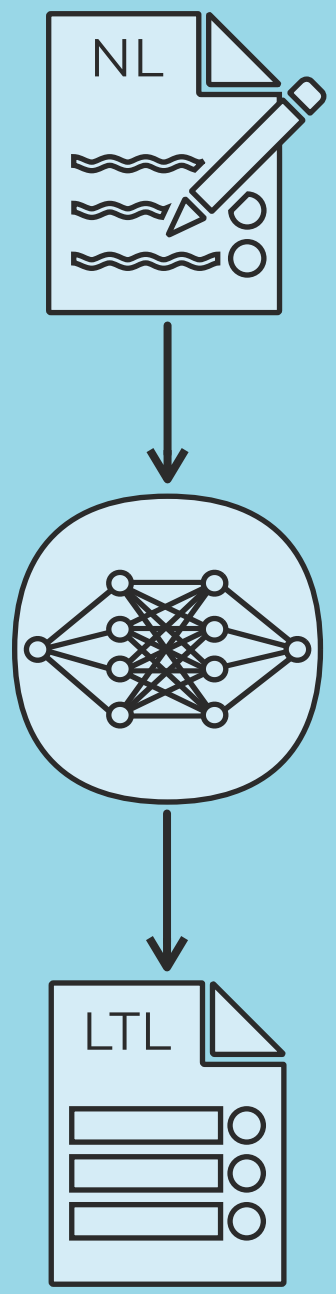


Natural Language to Temporal Logics

Whenever a holds, b must hold in the next two steps

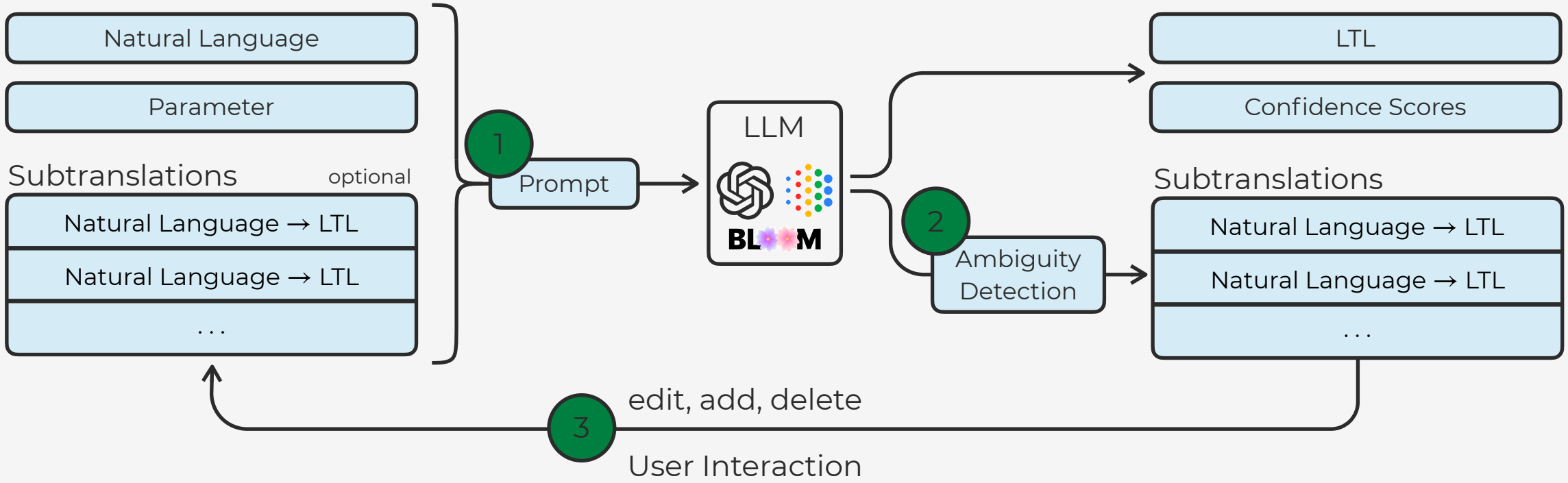
$$\square(a \rightarrow (b \vee \bigcirc b))$$
$$\square(a \rightarrow (b \wedge \bigcirc b))$$
$$\square(a \rightarrow \bigcirc(b \wedge \bigcirc b))$$
$$\square(a \rightarrow \bigcirc\bigcirc b)$$

Natural Language is ambiguous



Design

Natural Language to Temporal Logics



Prompt
Natural Language to Temporal Logics

Chain-of-thought prompting

Translate the following natural language sentences into an LTL formula and explain your translation step by step.

Specification language context

Remember that X means "next", U means "until", G means “globally", F means "finally", which means GF means "infinitely often". The formula should only contain atomic propositions or operators &, ~, ->, <->, X, U, G, F.

Few-shot prompting (3x)

Natural Language: Globally if a holds then c is true until b.
Given translations: {}
Explanation: "a holds" from the input translates to the atomic proposition a. "c is true until b" from the input translates to the subformula c U b. "if x then y" translates to an implication x -> y, so "if a holds then c is true until b" translates to an implication a -> c U b. "Globally" from the input translates to the temporal operator G.
Explanation dictionary: `{"a holds": "a", "c is true until b": "c U b", "if a holds then c is true until b": "a -> c U b", “Globally": "G"}`
So the final LTL translation is G a -> c U b. `FINISH`

Interactive prompt

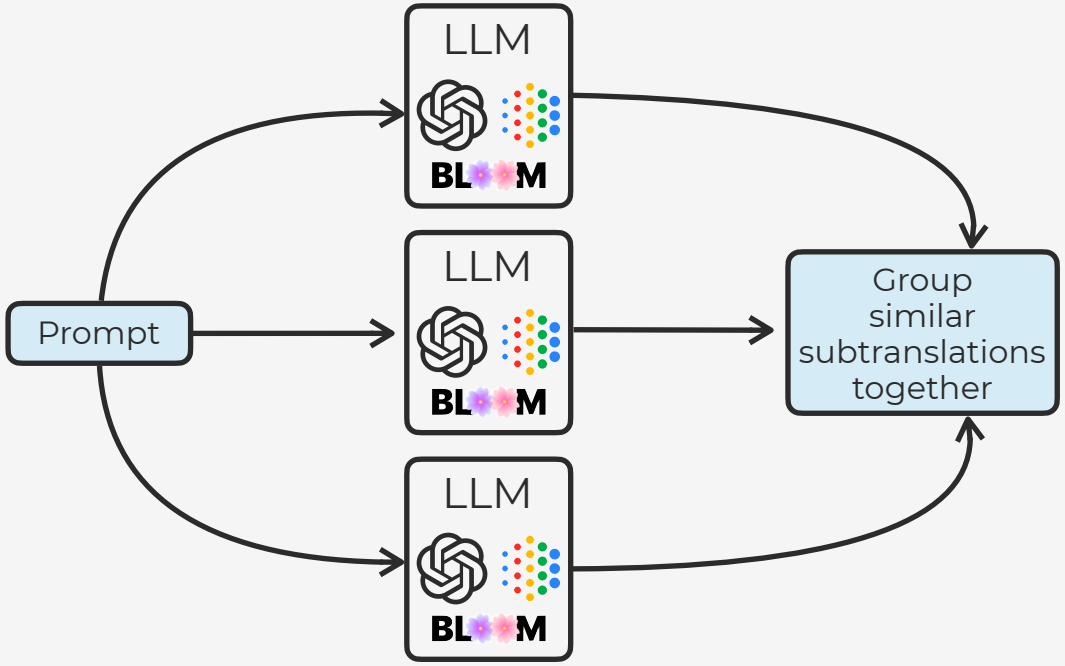
Natural Language: ...
Given translations: ...

Ambiguity Detection

Natural Language to Temporal Logics

Whenever a holds, b must hold in the next two steps

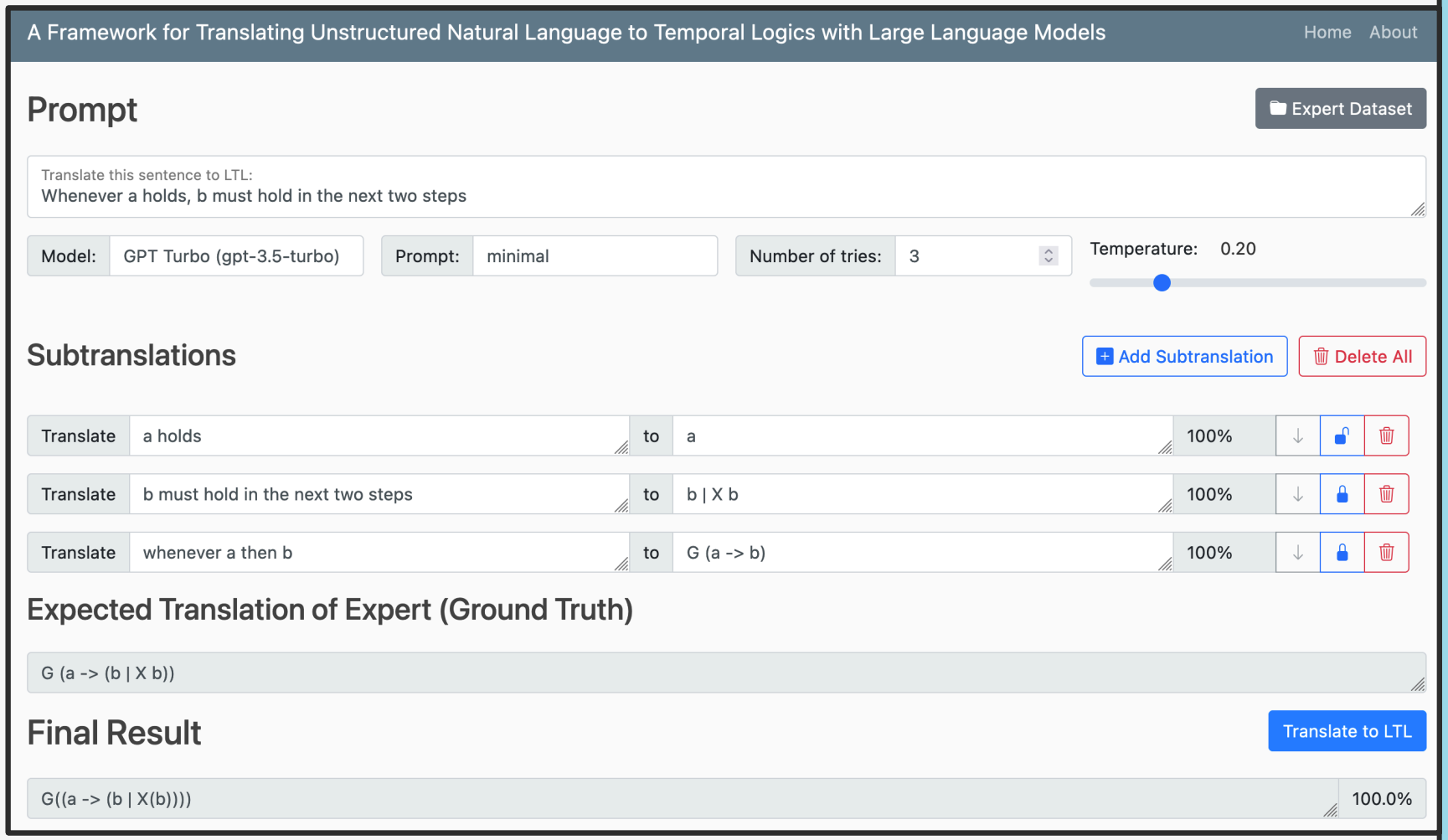
$$\square(a \rightarrow (b \vee \bigcirc b))$$
$$\square(a \rightarrow (b \wedge \bigcirc b))$$
$$\square(a \rightarrow \bigcirc(b \wedge \bigcirc b))$$
$$\square(a \rightarrow \bigcirc\bigcirc b)$$



Ambiguity cannot be resolved automatically

User Interaction

Natural Language to Temporal Logics



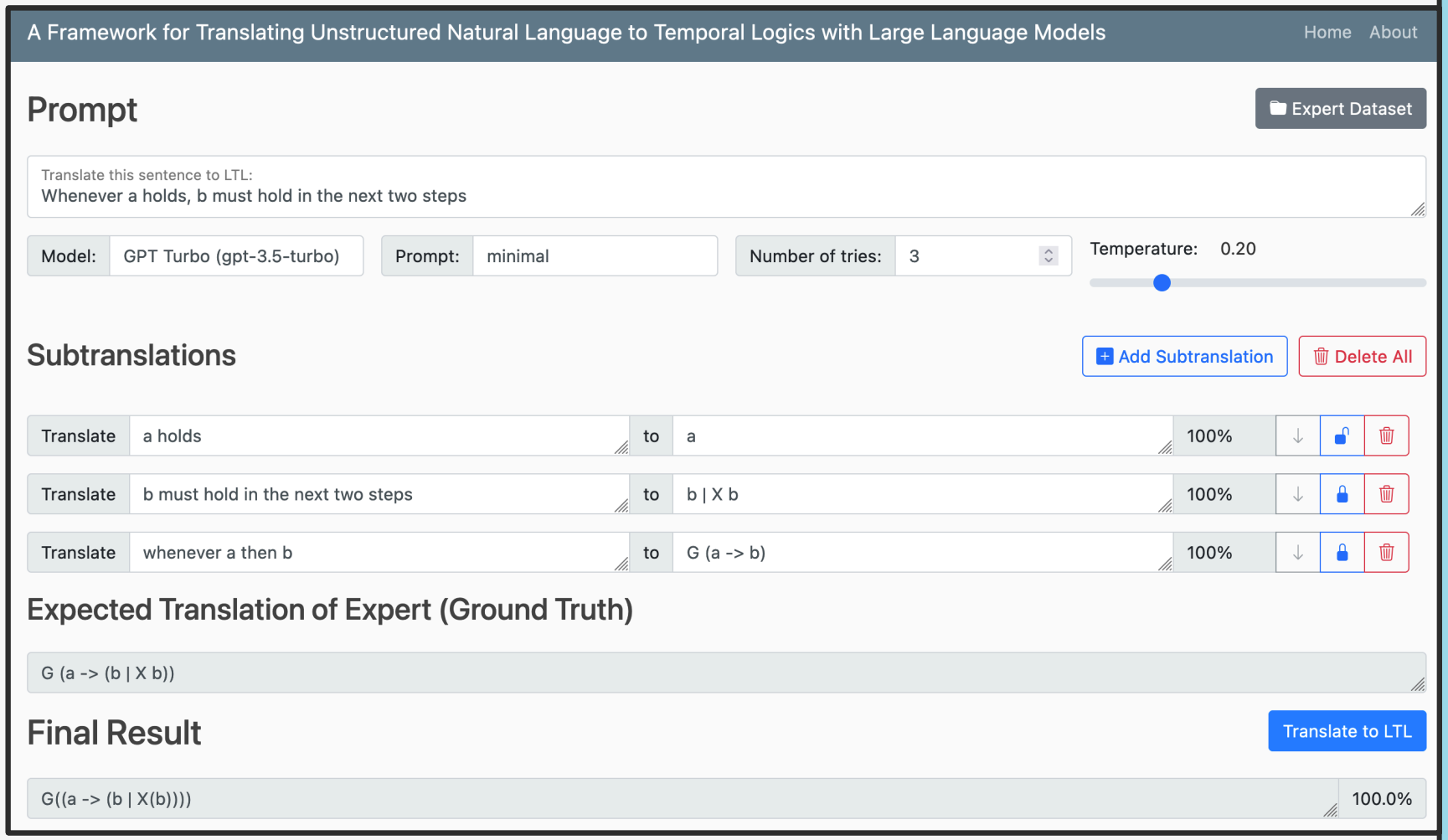
I. Globally missing

➡ Refine the meaning of whenever

II. Remove ambiguity

User Interaction

Natural Language to Temporal Logics



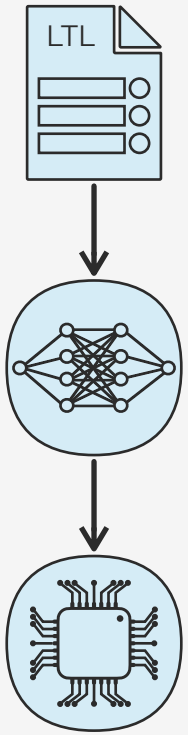
Interactive Experiment:

- Up to 3 interaction loops
- 1.4 interaction loops on average
- Expert dataset

31/36 (86.1%)

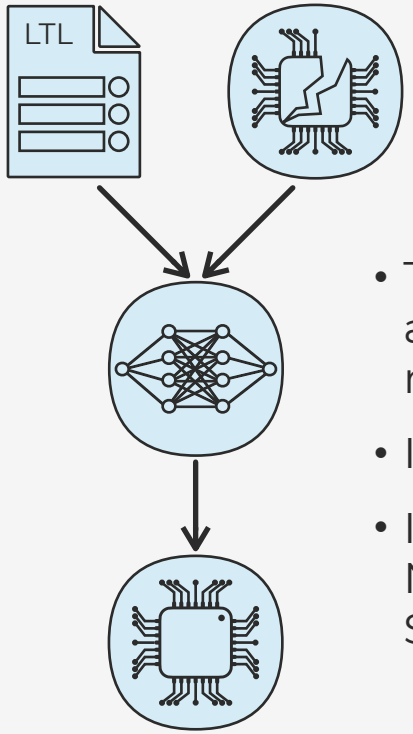
Summary

Neural Circuit Synthesis



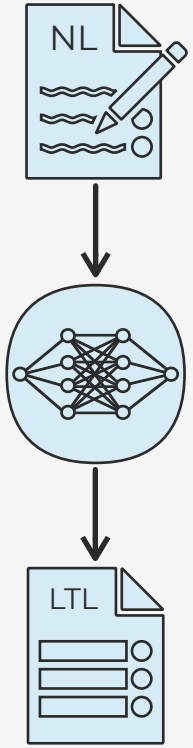
- End-to-End transformer architecture
- Generalization to larger, harder, and OOD instances
- Sound and complete integration into a portfolio solver

Circuit Repair



- Transformer architecture for multimodal input
- Iterative approach
- Improvement to Neural Circuit Synthesis

Natural Language to Temporal Logics



- Large Language Models
- Unstructured natural language
- Ambiguity detection and resolving