Formal Methods Australia, 5 June 2026

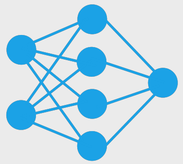
A Principled Programming Model for Constructing LLM-Powered Reasoning Tools

Aaron Bembenek
Research Fellow
The University of Melbourne

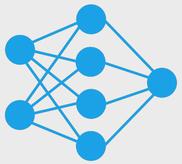
Context: Automated Reasoning (AR) tools

AI systems targeted towards correct logical reasoning

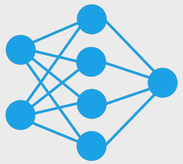
Combinatorial solvers (SAT, SMT, CHC, etc.)



Program synthesizers and verifiers



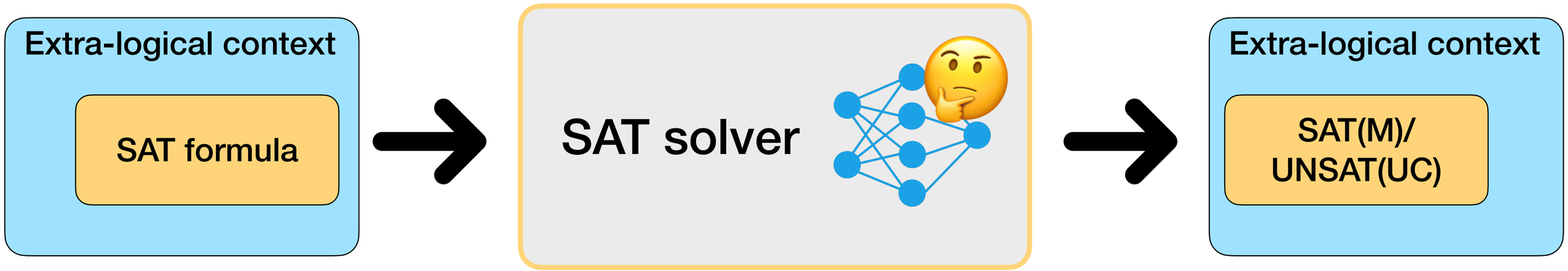
Automated theorem provers



- Traditionally based in symbolic AI
  - Pros: trustworthy and interpretable
  - Cons: lack of “intuition” hampers solving, explainability/debugging
- New approach: **neurosymbolic (NeSy)** architectures
  - Symbolic computation plus LLM “reasoning”

Thought experiment: NeSy SAT solver

What properties would we want it to have?



1. Use neural “intuition” to more effectively search solution space
2. Ingest extra-logical context about problem (what it models in real world)
3. Produce extra-logical context that explains solution (wrt real world)
4. Computation must remain a decision procedure (terminates with solution)

Question:

How can we build these ideal NeSy AR systems?
Seems hard to do using ad hoc approaches

My answer:

A principled programming model — based on new ***computational models*** and ***programming languages***

This talk

Problem: without a principled computational foundation, LLM-powered AR tools cannot meet their transformative potential

Solution: the NeSy Transition System (NSTS), a computational model for NeSy AR that can be basis for new programming languages

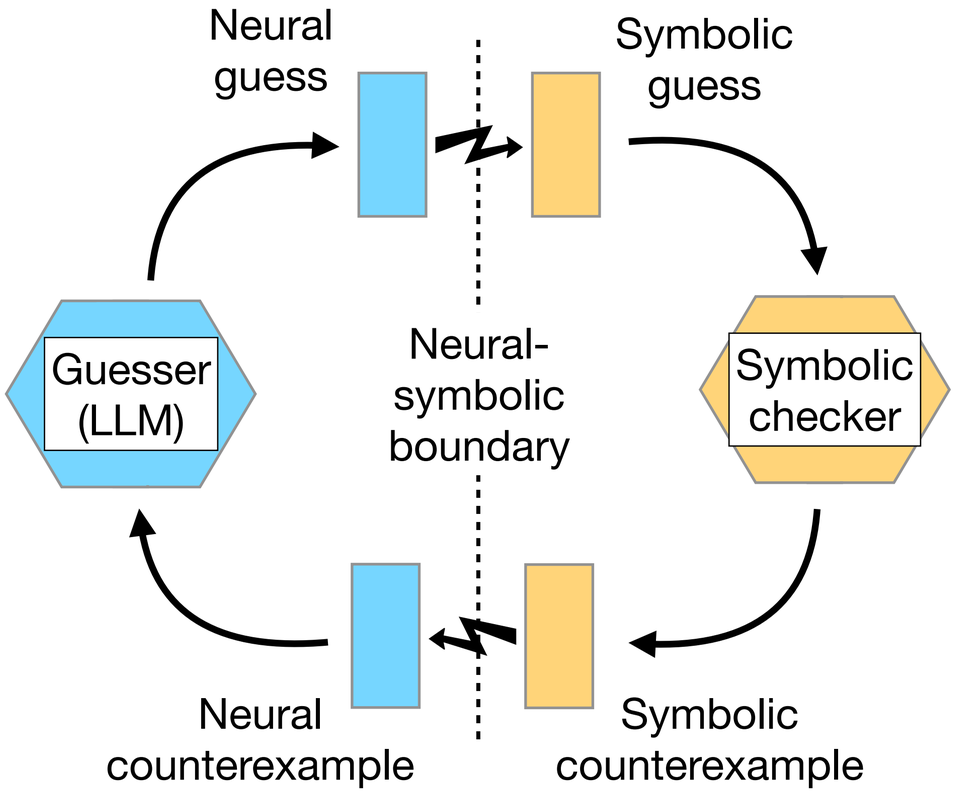
3-year fellowship on this topic starting in July

Needed: feedback and collaborators!!

Problem: Sequential NeSy

State of the practice: *sequential* NeSy

LLM calls and symbolic computation chained *in sequence*



Claim: sequential NeSy is not the right programming model for NeSy AR

Strict neural-symbolic boundaries: symbolic parts do not ingest, use, or produce “intuition”

Architecture forces logical reasoning to flow through illogical component (LLM)

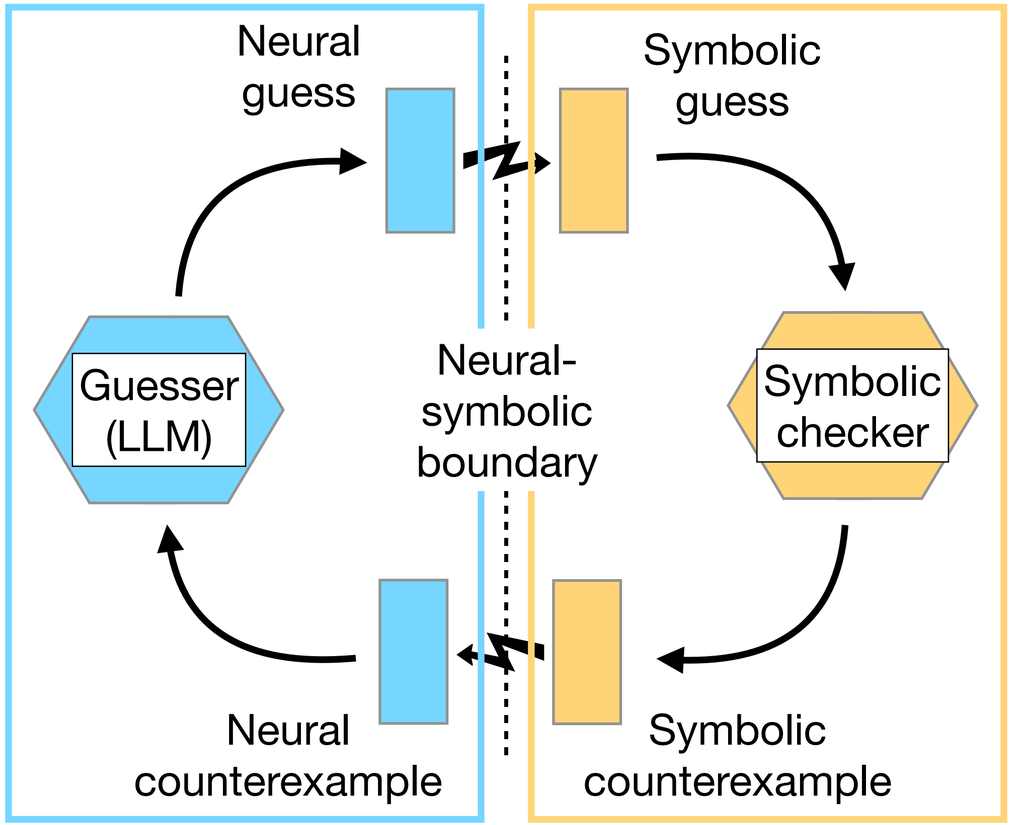
Hard to build systems with good computational properties (e.g., semi-decision procedures)

Agentic AI is in sequential NeSy

Sequential NeSy is built on a limiting conceptual model

A piece of data is, at any one time,

- exclusively neural (interpreted according to **learned connotation**)
- exclusively symbolic (interpreted according to **formal denotation**)



But every datum is inherently “neurosymbolic”!

`password123`

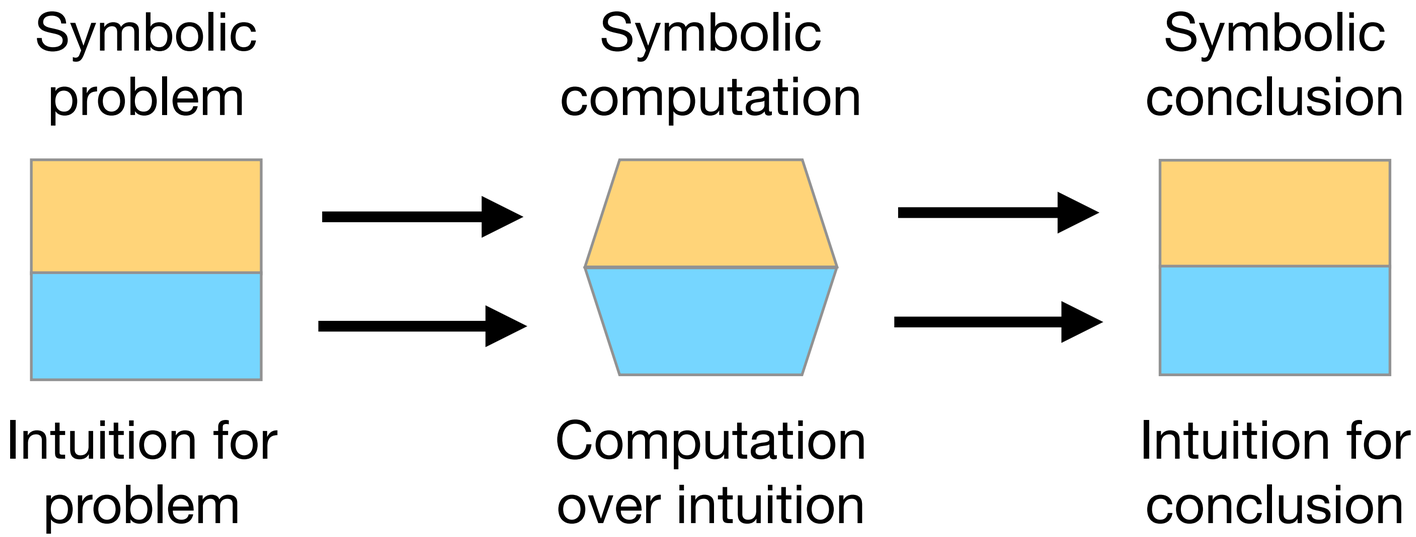
A particular string of characters (formal denotation)

A poor password choice (learned connotation)

Desire: NeSy AR tools that holds both interpretations simultaneously

- Connotations guide logical reasoning
- Act of logical reasoning updates connotations

Data are NeSy ⟹ NeSy computation should be over intuition and symbols *in parallel*



Problem: Sequential NeSy is natural consequence of existing prog. languages

So, let's build new programming infrastructure for parallel NeSy!

Solution: NeSy Transition System

Example: loop invariant inference

Key to automating software verification

```
int x, y, z;

…

while (…) {
  // invariant over x, y, z

  …

}
```

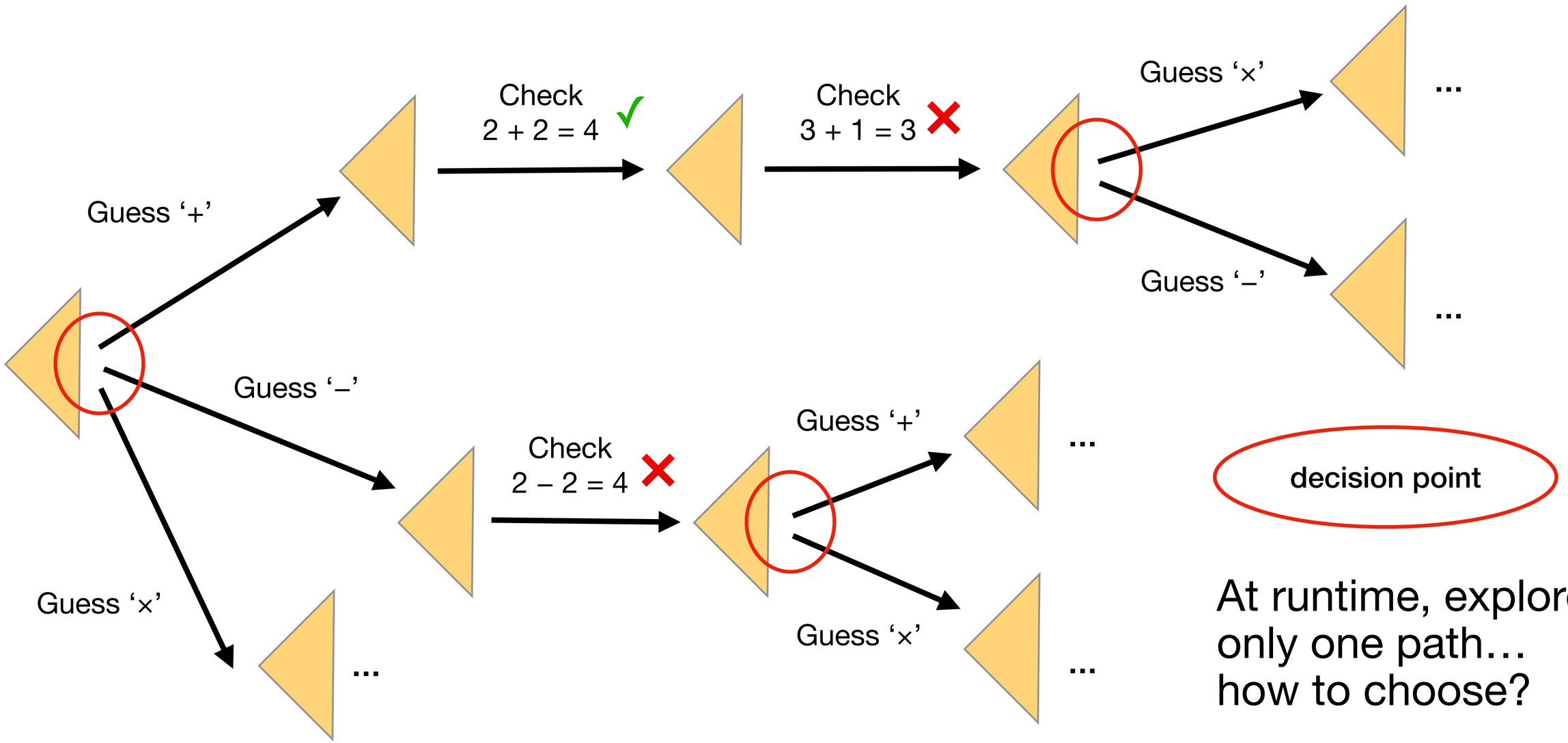
Data from traces:

x	y	z
2	2	4
3	1	3
2	4	8

Problem: choose an operator $\oplus \in \{ +, -, \times \}$ such that $x \oplus y = z$ fits the data

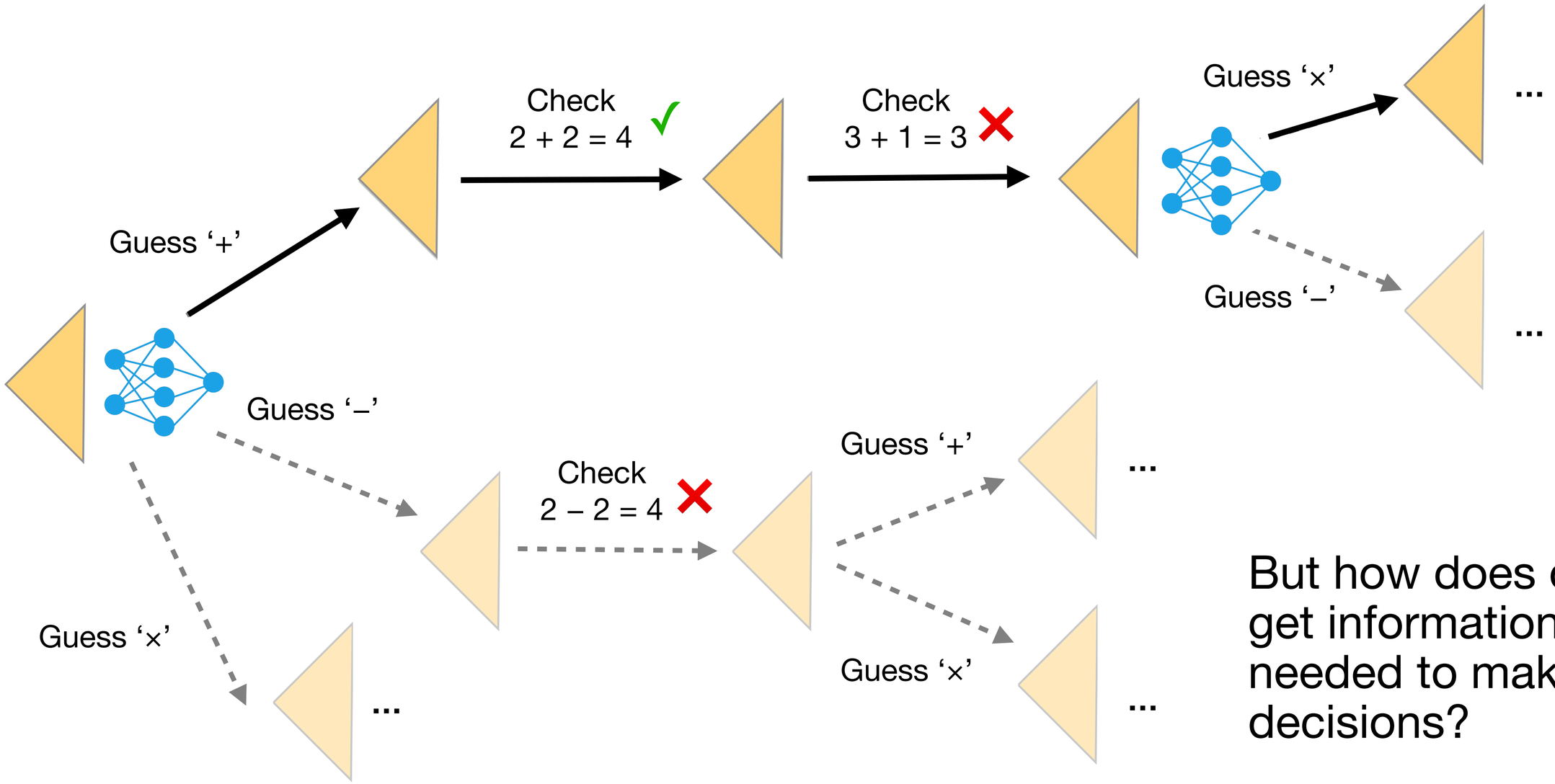
Next few slides: a parallel NeSy solver for this problem

Starting point: a logical transition system



At runtime, explore only one path… how to choose?

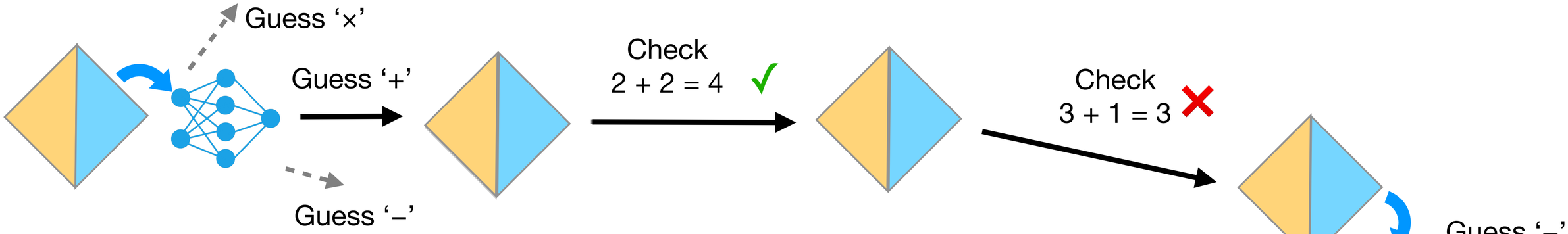
At decision points, query intuitive oracle (LLM)



But how does oracle get information needed to make decisions?

Maintain synchronized intuition state

By giving intuitive interpretation to each symbolic step



Intuition state

We need help finding a loop invariant over x, y, and z for the following program: …

Choose an operator $\oplus \in \{ +, -, \times \}$ such that $x \oplus y = z$ fits this data: …

You guessed ‘+’.

Indeed, 2 + 2 = 4.

But, 3 + 1 = 4, not 3 (as required).

counter-example

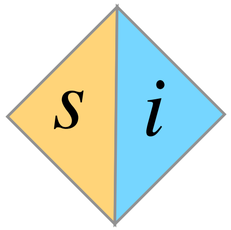
Make a new guess, choosing between ‘–’ and ‘×’.

You guessed ‘×’.

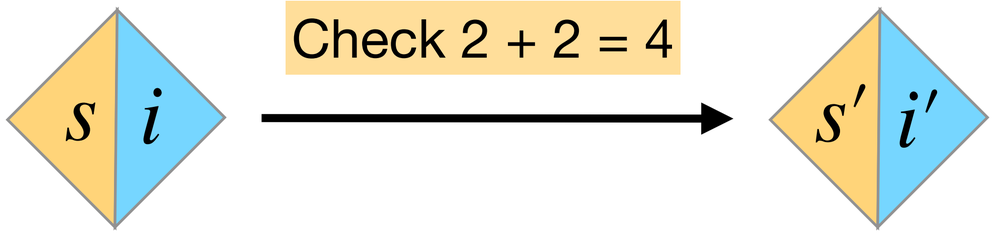
…

Recap:

1. Pair symbolic state with intuition

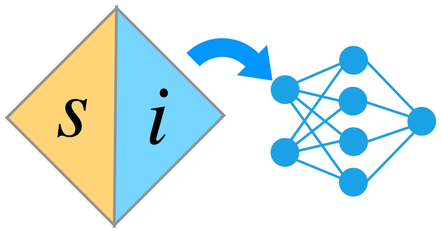


2. Update intuition state using intuition for symbolic step



$$i' = concat(i, \text{“Indeed, 2 + 2 = 4”})$$

3. Query oracle with accumulated intuition



What we achieved:

System ingests, uses, and produces intuition ✓

Intuition evolves over course of computation ✓

System preserves logical behavior ✓

Generalizing: NeSy Transition System (NSTS)

Labeled Transition System

States S, actions A

$step : S \times A \to \mathcal{P}(S)$

+

Intuition domains I, J, Δ (Text)

$update : I \times \Delta \to I$ (Concat)

$infer : S \times I \to A \times J$ (Call LLM)

Label $\lambda : (S \times A \times S) \times J \to \Delta$

Given state and running intuition, infer action & provide intuitive rationale

$$infer(s, i) = \langle a, j \rangle$$

$$s' \in step(s, a)$$

$$\lambda(s, a, s', j) = \delta$$

$$i' = update(i, \delta)$$

$$\langle s, i \rangle \Longrightarrow \langle s', i' \rangle$$

Step according to symbolic system

Generate intuition for transition

Update running intuition

Configuration is state and running intuition

The NSTS Framework

1st view: an instantiation of the parallel NeSy architecture

2nd view: a recipe for lifting symbolic computation to NeSy computation

RQ 1: Can we build practical NeSy AR tools on top of NSTS framework?

RQ 2: Can we build NeSy programming languages based on NSTS model?

RQ 3: Can we embed NSTS component inside neural architecture?

Interested in collaborating? Let me know!

One direction: NSTS logic programming

```
:- candidate(Op),
   check(Op, 2, 2, 4),
   check(Op, 3, 1, 3),
   check(Op, 2, 4, 8).

candidate(plus).
candidate(minus).
candidate(mult).

check(plus, X, Y, Z)  :- X + Y is Z.
check(minus, X, Y, Z) :- X - Y is Z.
check(mult, X, Y, Z)  :- X * Y is Z.
```

Problem specification as top-level query

Grammar of candidate solutions

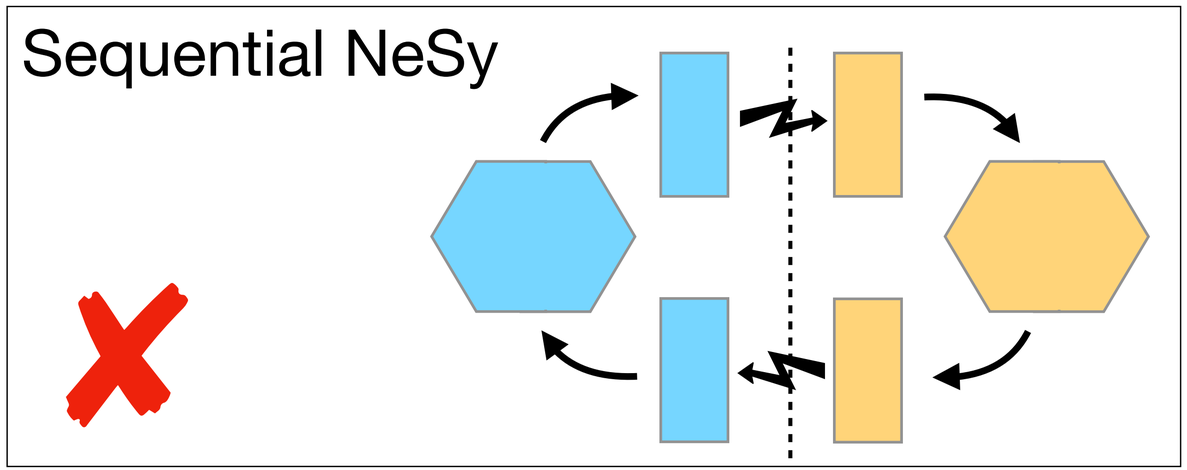
Rules for checking correctness of candidate solution

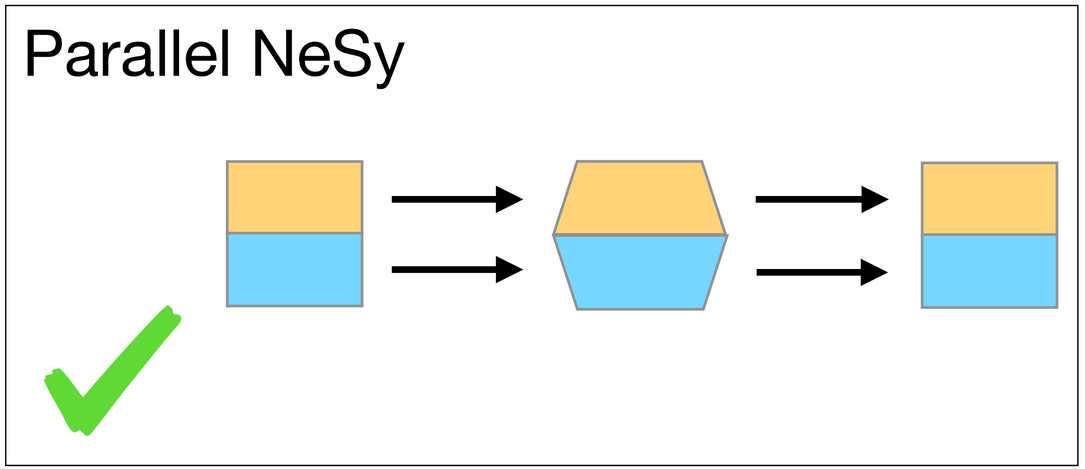
Programmer writes *symbolic* program that is run on *intuition-powered* machine

Declarative programming in age of LLMs? (cf. vibe coding, prompt engineering)

Conclusion

NeSy Transition Systems: a principled computational model for LLM-powered AR





Now is the time for foundational work on constructing NeSy reasoning systems!

aaronbembenek.github.io/blog/