Thank you!

Specula: Generating TLA+ specifications from system code using generative AI

Qian Cheng, Peiyang He, Yu Huang — Nanjing University
Ruize Tang — Microsoft Research
Emilie Ma, Finn Hackett, Ivan Beschastnikh — UBC
Yiming Su, **Tianyin Xu** — University of Illinois Urbana Champaign

Nvidia Formal Methods Mini-Conference, Nov. 20, 2025

Formal methods and TLA+

"Program testing can at best show the presence of bugs, but never their absence."

EWD

DOI:10.1145/2699417

Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

BY CHRIS NEWCOMBE, TIM RATH, FAN ZHANG, BOGDAN MUNTEANU, MARC BROOKER, AND MICHAEL DEARDEUFF

How Amazon Web Services Uses Formal Methods

S3 is just one of many AWS services that store and process data our customers have entrusted to us. To safeguard that data, the core of each service relies on fault-tolerant distributed algorithms for replication, consistency, concurrency control, auto-scaling, load balancing, and other coordination tasks. There are many such algorithms in the literature, but combining them into a cohesive system is a challenge, as the algorithms must usually be modified to interact properly in a real-world system. In addition, we have found it necessary to invent algorithms of our own. We work hard to avoid unnecessary complexity, but the essential complexity of the task remains high.

Complexity increases the probability of human error in design, code, and operations. Errors in the core of the system could cause loss or corruption of data, or violate other interface contracts on which our customers depend. So, before launching a service, we need to reach extremely high confidence that the core of the system is

Multi-Grained Specifications for Distributed System Model Checking and Verification

- April 22, 2025

This EuroSys 2025 paper wrestles with the messy interface between formal specification and implementation reality in distributed systems. The case study is ZooKeeper. The trouble with verifying something big like ZooKeeper is that the spec and the code don't match. Spec wants to be succinct and abstract; code has to be performant and dirty.

Our ZooKeeper story

P. Hunt, e al., ZooKeeper: Wait-free coordination for Internet-scale systems, USENIX ATC, 2010.

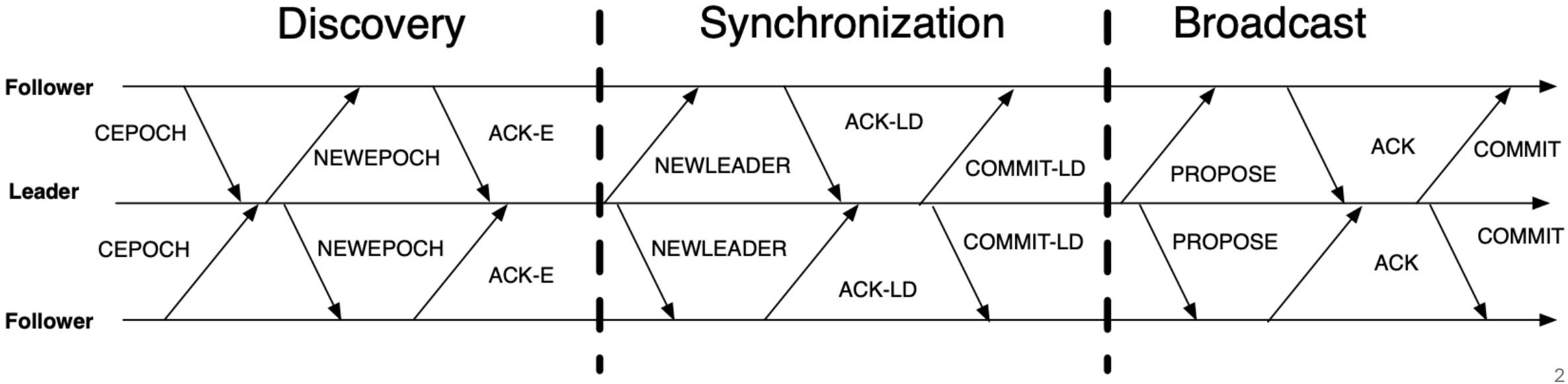
Our ZooKeeper story

Zab.tla

APACHE ZooKeeper™ implements the **Zab (ZooKeeper Atomic Broadcast)** protocol

CEPOCH = Follower sends its last promise to the prospective leader
NEWEPOCH = Leader proposes a new epoch e'
ACK-E = Follower acknowledges the new epoch proposal
NEWLEADER = Prospective leader proposes itself as the new leader of epoch e'
ACK-LD = Follower acknowledges the new leader proposal
COMMIT-LD = Commit new leader proposal

PROPOSE = Leader proposes a new transaction
ACK = Follower acknowledges leader proosal
COMMIT = Leader commits proposal



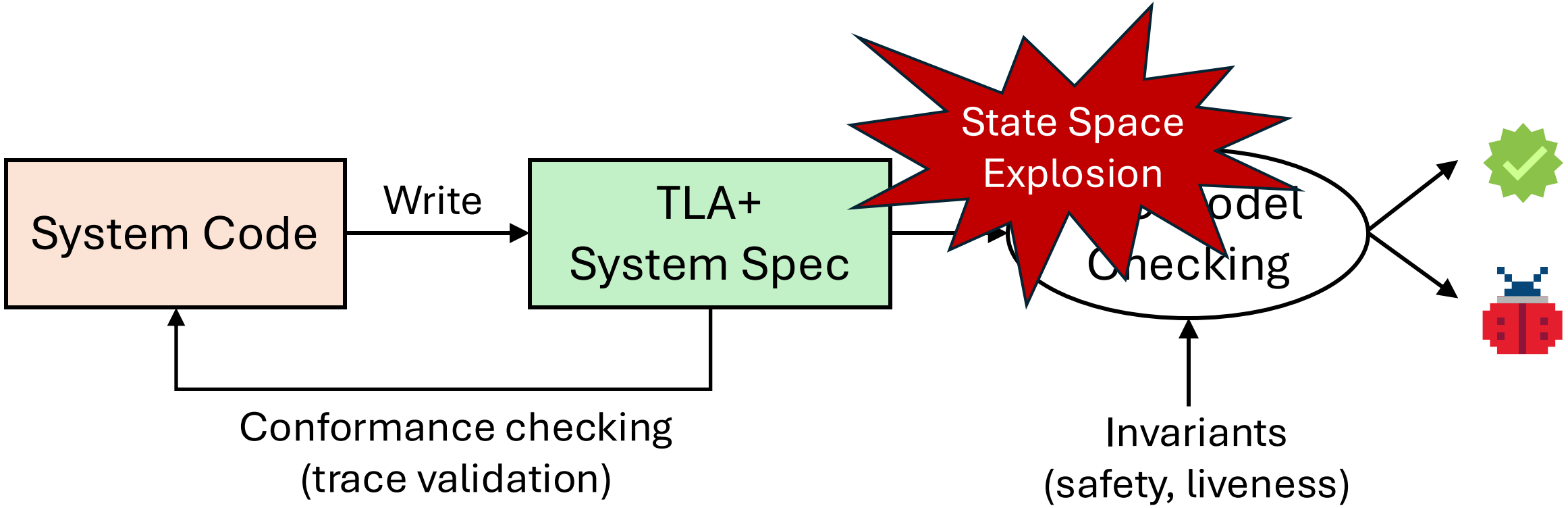
F. P. Junqueira, e al., Zab: High-performance broadcast for primary-backup systems, DSN, 2011.

Protocol specs (e.g., Zab.tla) are insufficient

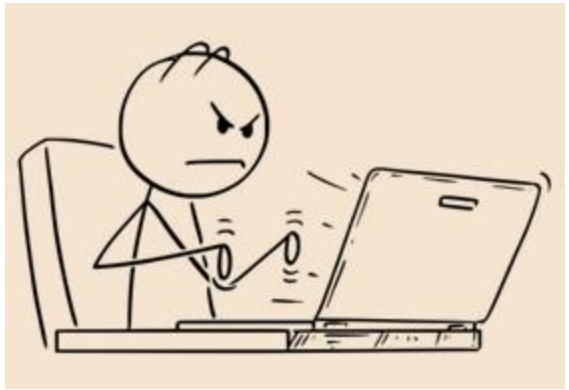
- **Protocols do not address performance optimizations**
  - One **atomic action** in Zab is implemented by several concurrent operations in ZooKeeper as an optimization.
  - The protocol spec *cannot* address concurrency bugs
- **Zab does not describe leader election**
  - ZooKeeper implements Fast Leader Election (FLE)
  - The protocol spec *cannot* address unexpected FLE cases

https://github.com/apache/zookeeper/blob/master/zookeeper-specifications/protocol-spec/Zab.tla

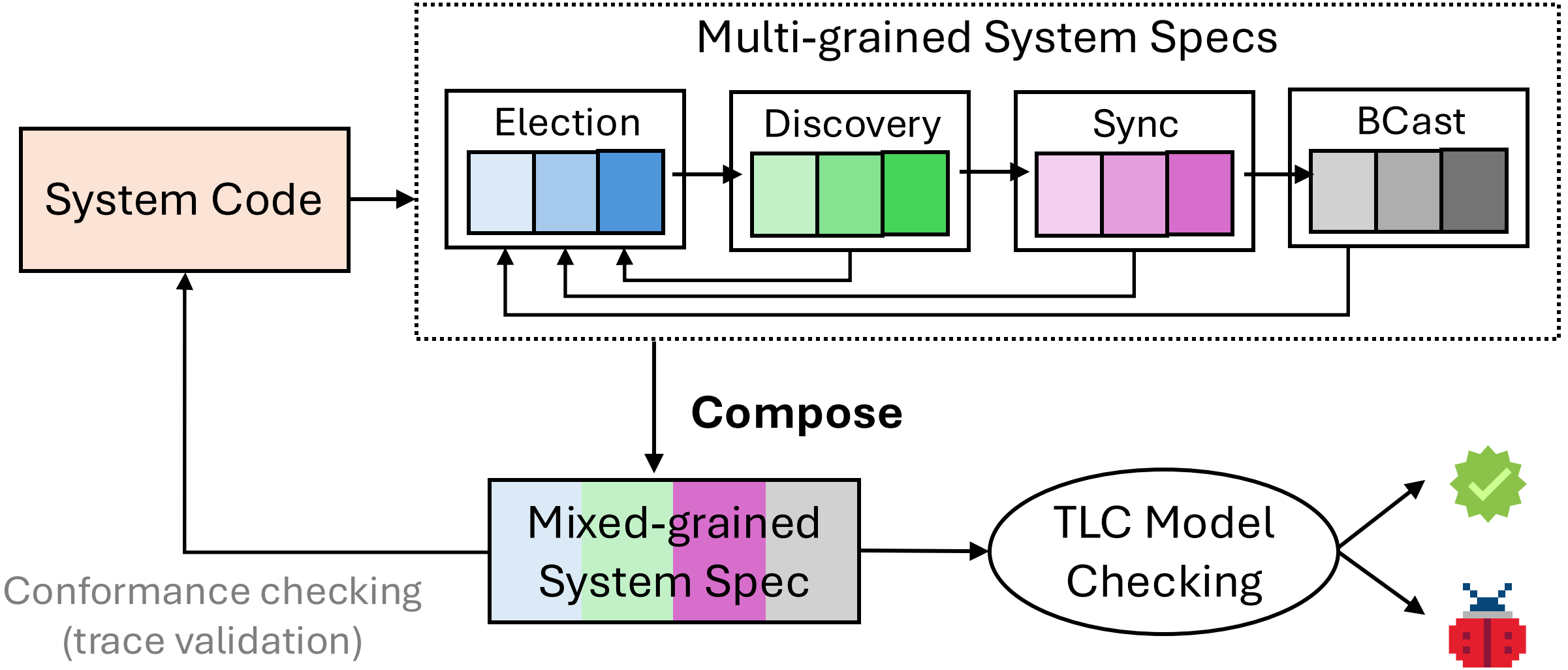
From system code to System Specs



“Write” means



From system code to System Specs



L. Ouyang, e al., Multi-Grained Specifications for Distributed System Model Checking and Verification, EuroSys, 2025.

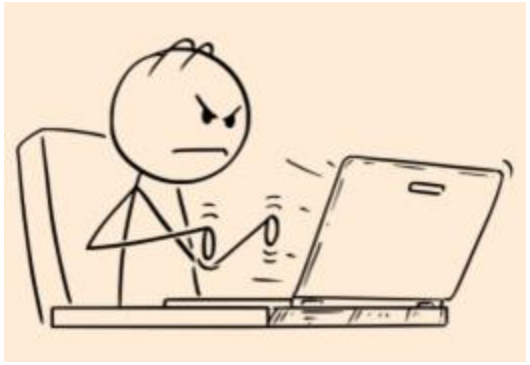
From system code to System Specs

5 years of a PhD student's life

- Read and understand ZooKeeper code
- Learn TLA+
- Write one TLA+ system spec
- Write multi-grained TLA+ system specs
- Update TLA+ system specs for every new version

N * 5 years for N PhD students (e.g., concurrency reasoning for OS synchronization mechanisms [USENIX ATC '25])

Promises of generative AI



5 years of a PhD student's life

- Read and understand ZooKeeper code
- Learn TLA+
- Write one TLA+ system spec
- Write multi-grained TLA+ system specs
- Update TLA+ system specs for every new version

5 LLM calls

Promises of generative AI

- Read and understand ZooKeeper code
- Learn TLA+
- Write one TLA+ system spec **Focus**
- Write multi-grained TLA+ system specs
  → If AI can generate one, it can generate multiple
- Update TLA+ system specs for every new version
  → Do we still need incremental updates?

AI agent for system modeling

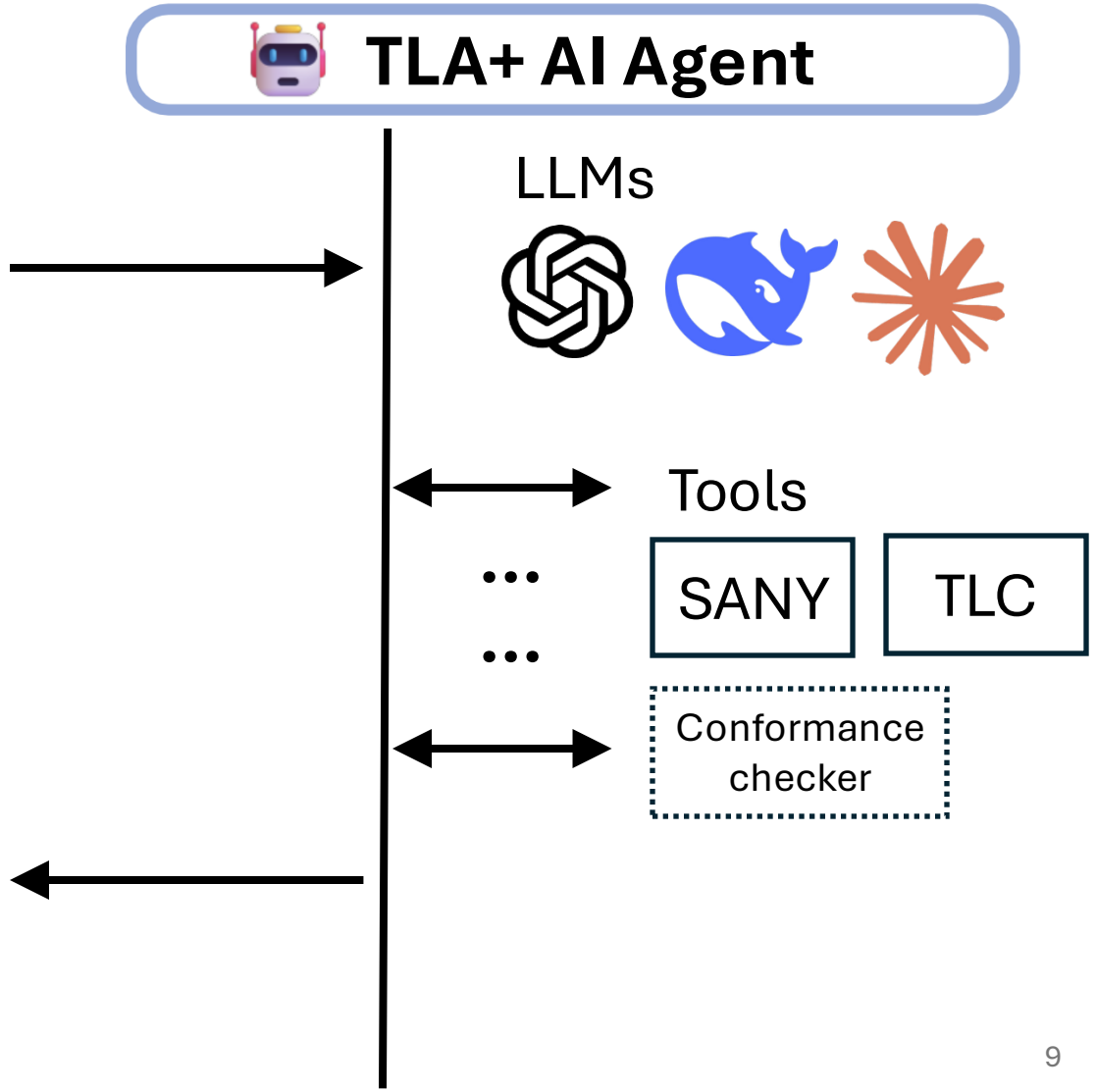
System Code

Etcd Raft. Distributed key-value store with Raft consensus for strong consistency

Code: raft.go | storage.go | node.go

TLA+ System Spec

```
1  ---- MODULE etcd_raft ----
2  CONSTANTS Server, Value
3  VARIABLES state, term, log
4  Init == term = [i \in Server |-> 0]
5  Next == Timeout \/ Election \/ Heartbeat
```



I want to use LLMs like GPT-5 like SWE-agent to write a TLA+ formal model for a given system implementation such as etcd Raft. The input will be the code of the system and I expect the output to be a perfect formal model written in TLA+ specification language. Do you think GPT-5 can do this?

In plain, simple language:

No — we are not there yet.

LLMs like GPT-5 **cannot** take a big real system (like etcd's Raft code) and automatically spit out a **perfect** TLA+ formal model.

They can help.
They can draft pieces.
They can save you a lot of time.

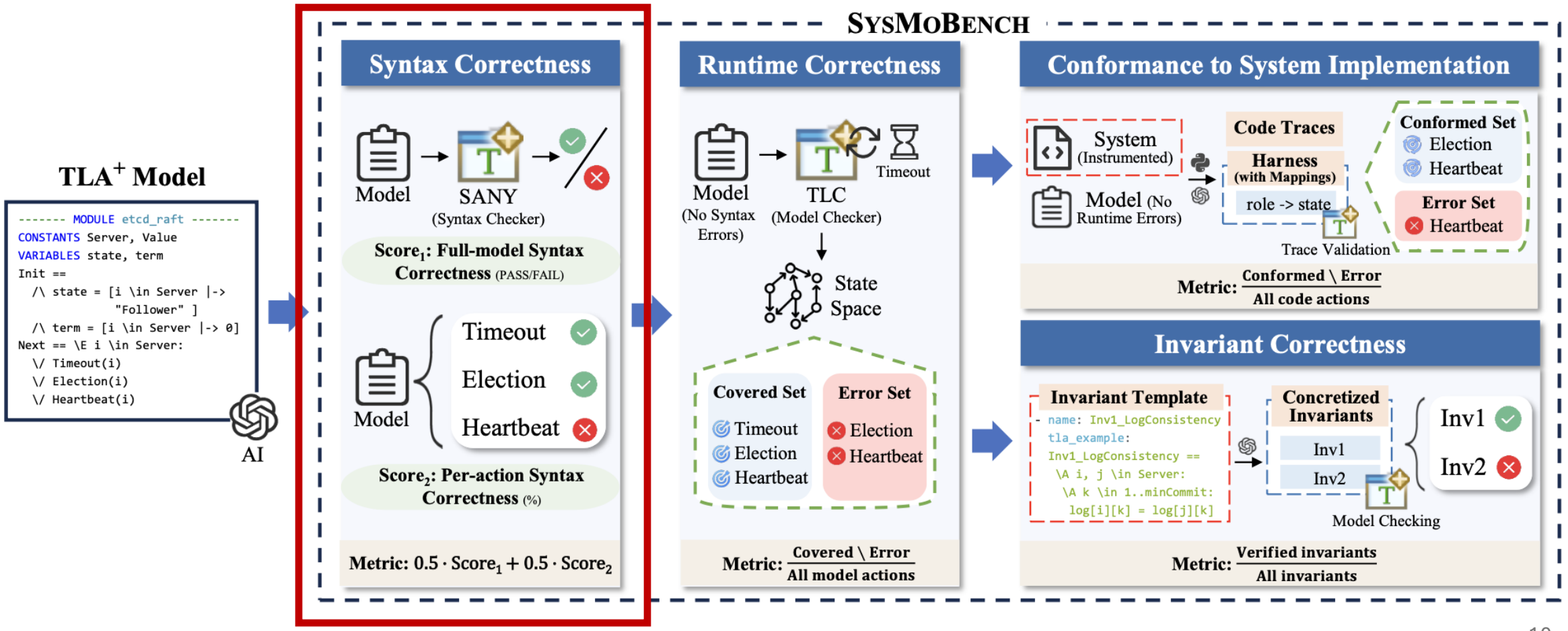
But they **cannot** fully understand the system, abstract it correctly, and produce a flawless formal spec on their own.

So the honest answer is:

We're closer, but not close enough yet. Not in 2025.

Benchmark on system modeling agents

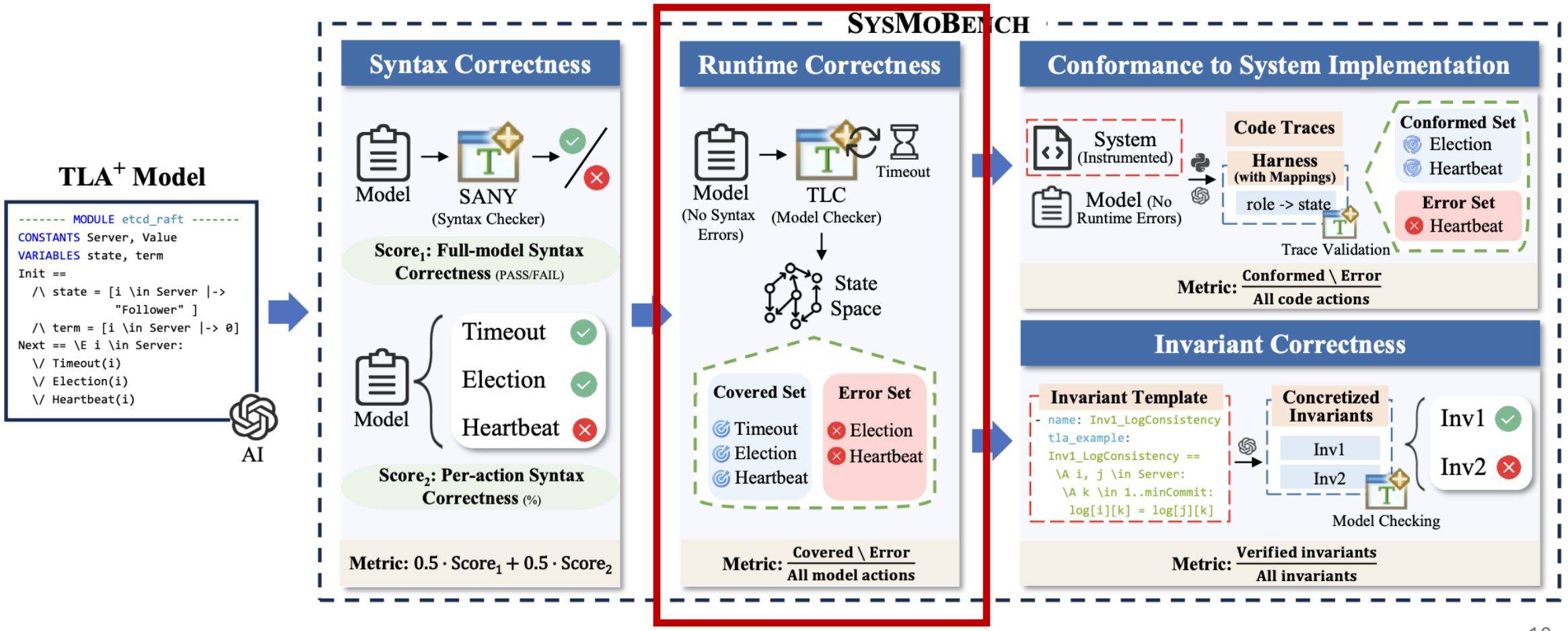
- **Automated metrics on the quality of a TLA+ model**



Q. Cheng, e al., SysMoBench: Evaluating AI on Formally Modeling Complex Real-World Systems, arXiv, 2025.

Benchmark on system modeling agents

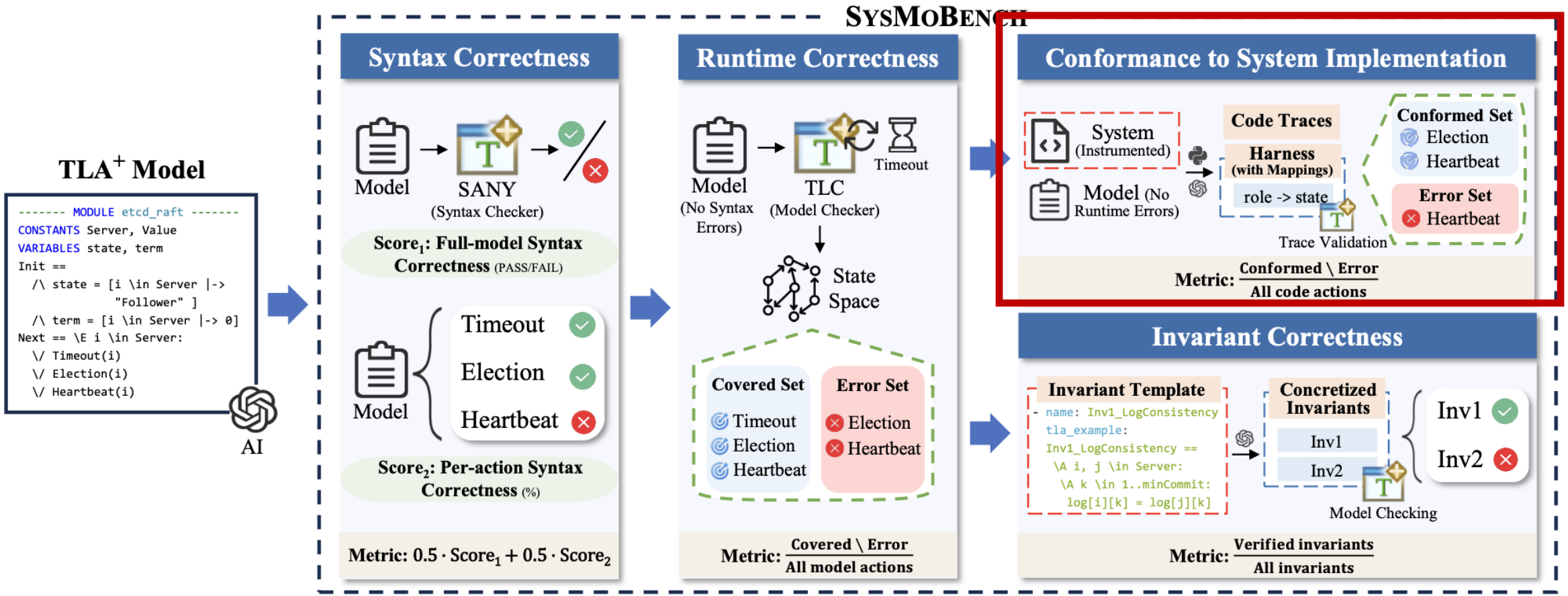
- **Automated metrics on the quality of a TLA+ model**



Q. Cheng, e al., SysMoBench: Evaluating AI on Formally Modeling Complex Real-World Systems, arXiv, 2025.

Benchmark on system modeling agents

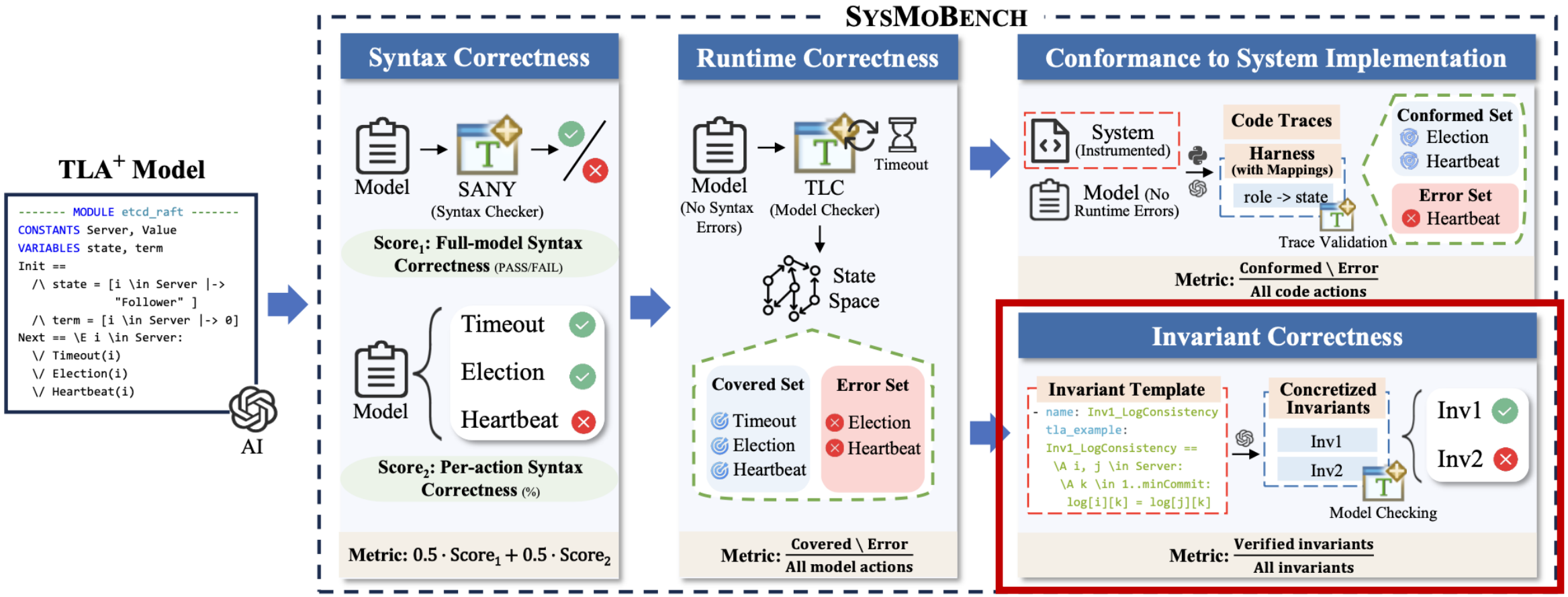
- **Automated metrics on the quality of a TLA+ model**



Q. Cheng, e al., SysMoBench: Evaluating AI on Formally Modeling Complex Real-World Systems, arXiv, 2025.

Benchmark on system modeling agents

- **Automated metrics on the quality of a TLA+ model**



Q. Cheng, e al., SysMoBench: Evaluating AI on Formally Modeling Complex Real-World Systems, arXiv, 2025.

Benchmark results

Spinlock Asterinas

LLM	Syntax	Runtime	Conformance	Invariant
Claude-Sonnet-4	100.00% ✓	100.00% ✓	100.00%	100.00%
GPT-5	100.00% ✓	100.00% ✓	80.00%	100.00%
Gemini-2.5-Pro	100.00% ✓	100.00% ✓	80.00%	85.71%
DeepSeek-R1	100.00% ✓	100.00% ✓	80.00%	100.00%

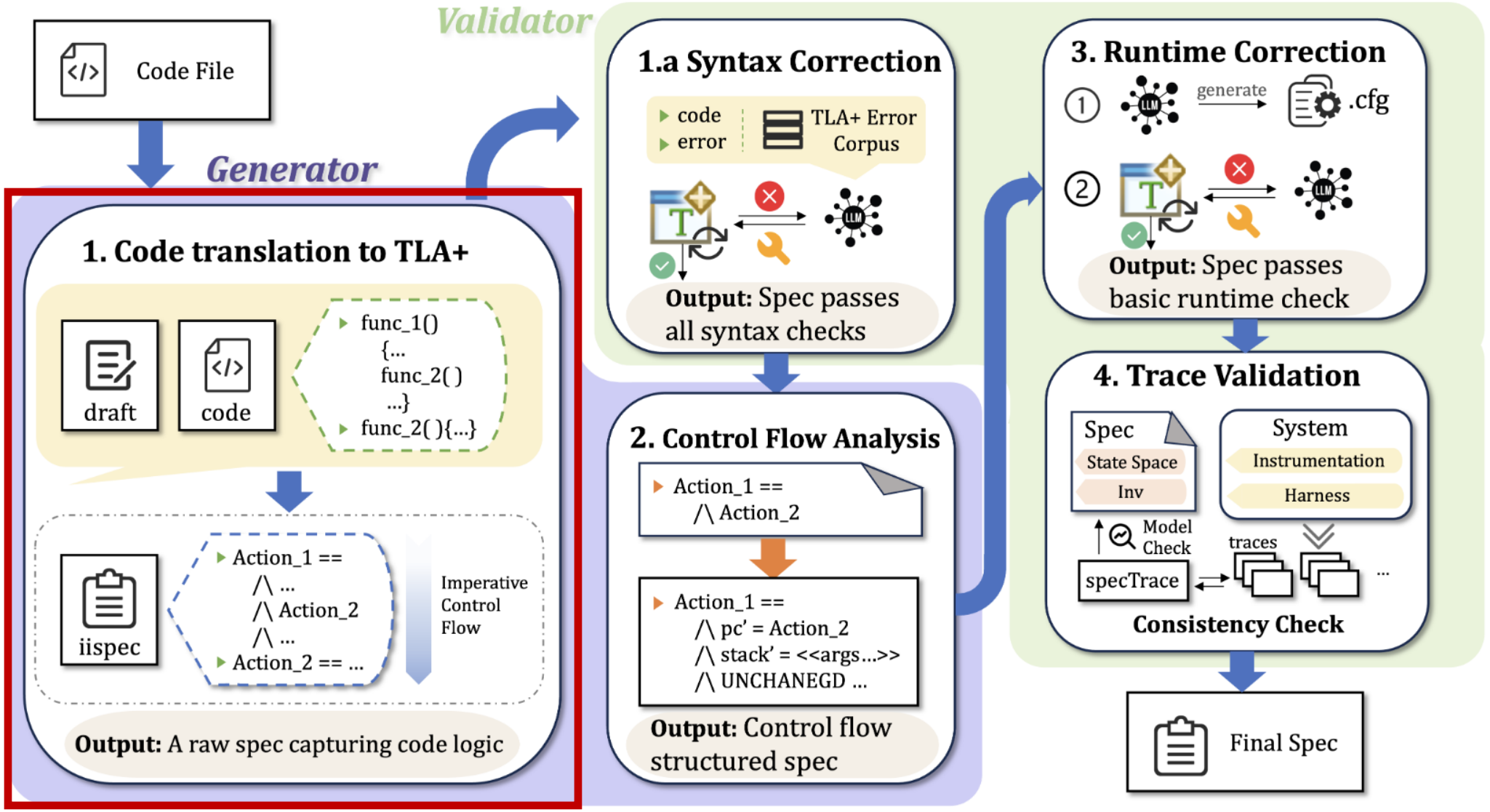
Raft etcd

LLM	Syntax	Runtime	Conformance	Invariant
Claude-Sonnet-4	100.00% ✓	25.00% ✓	7.69%	69.23%
GPT-5	47.87% ✗	-	-	-
Gemini-2.5-Pro	50.00% ✗	-	-	-
DeepSeek-R1	50.00% ✗	-	-	-

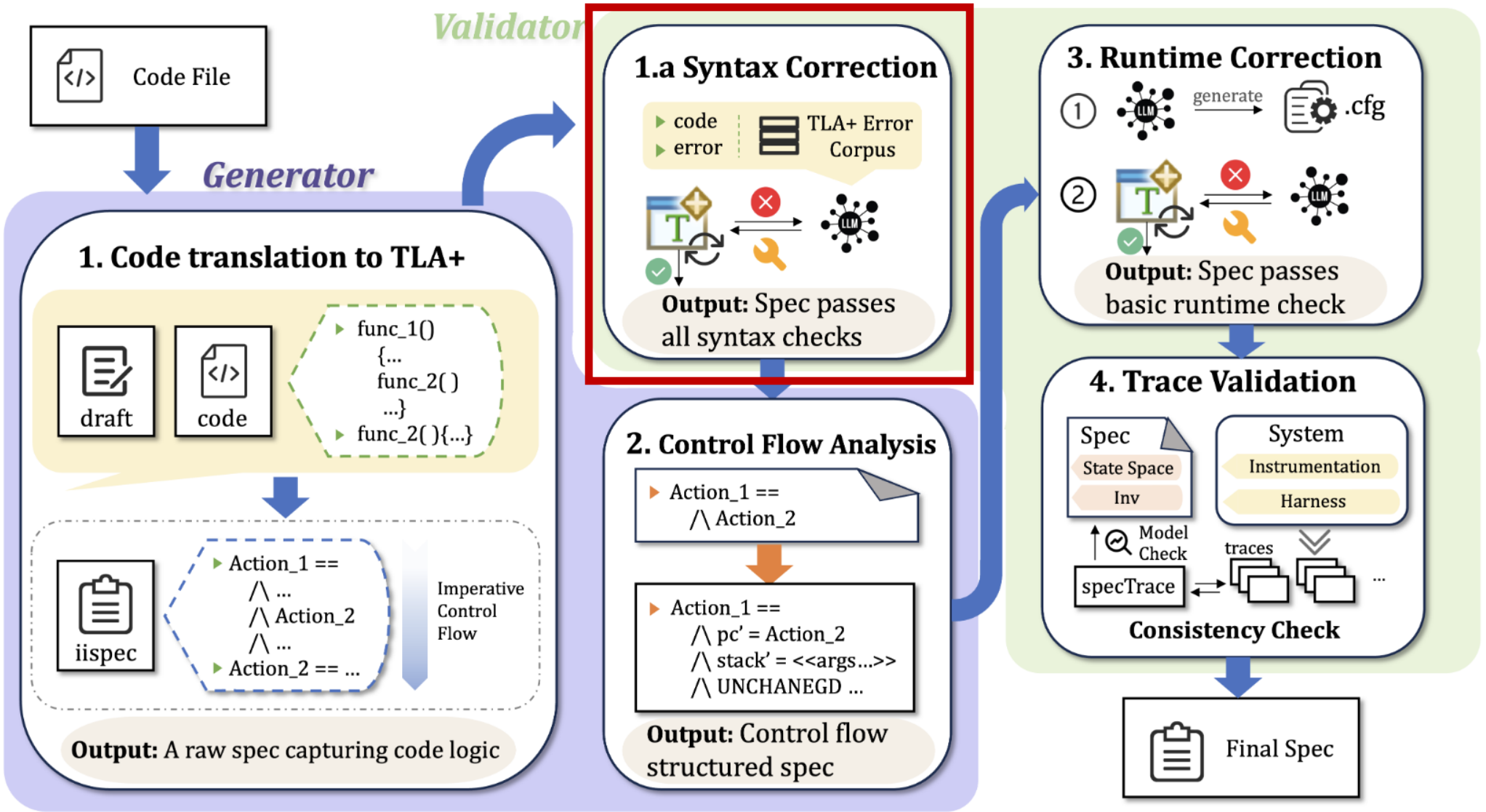
Specula (github.com/specula-org)

- **Leverages AI's code translation capabilities**
  - Code translation is easier than system comprehension
  - Translating from source code to TLA+
- **Takes a neurosymbolic approach**
  - Control-flow analysis to transform translated specs
    - Single-assignment constraint, UNCHANGED requirement, state annotation
- **Auto-corrects errors iteratively**
  - Incorporates feedback (e.g. TLC, trace validation, RAG)

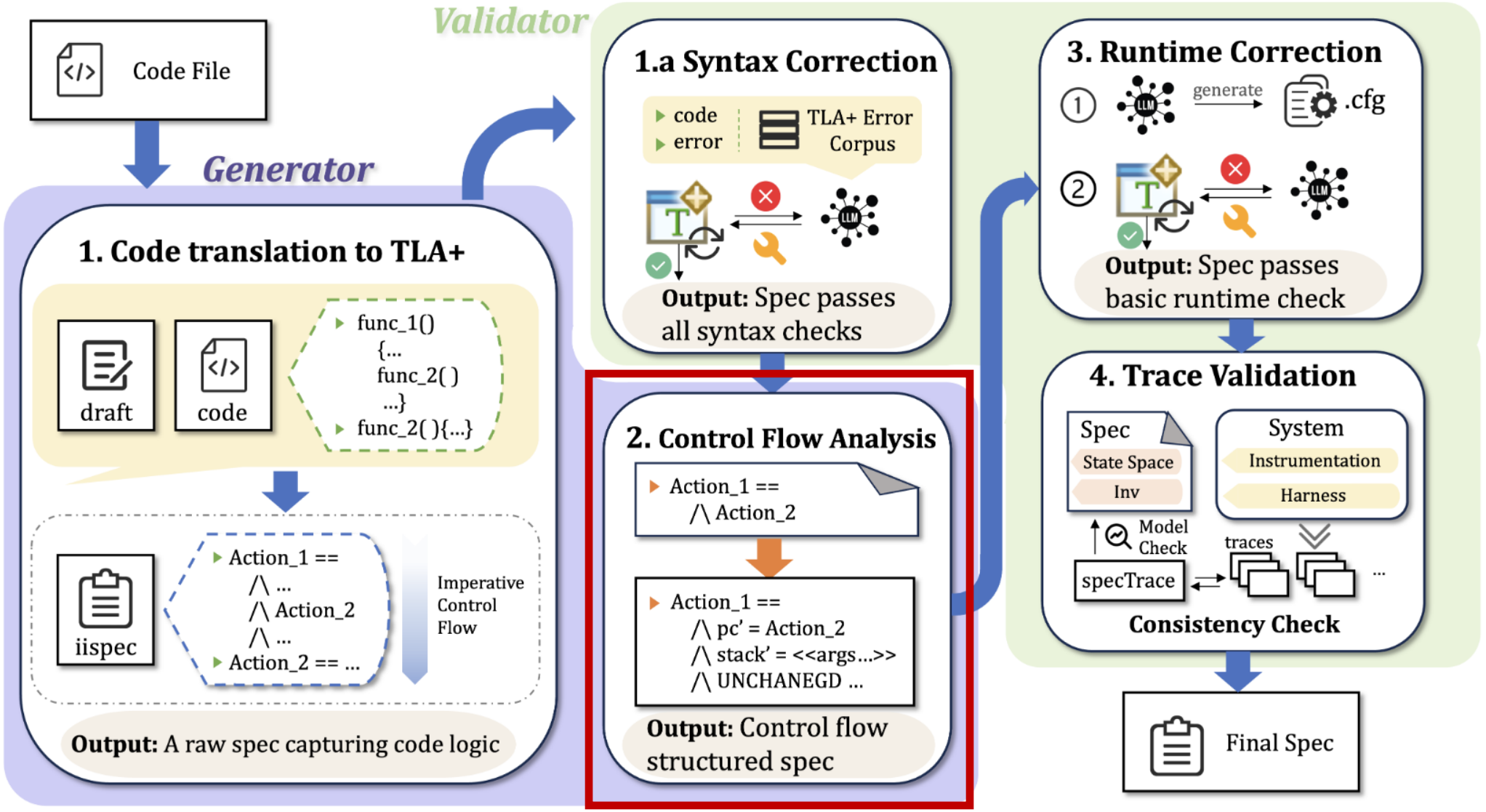
Specula design overview



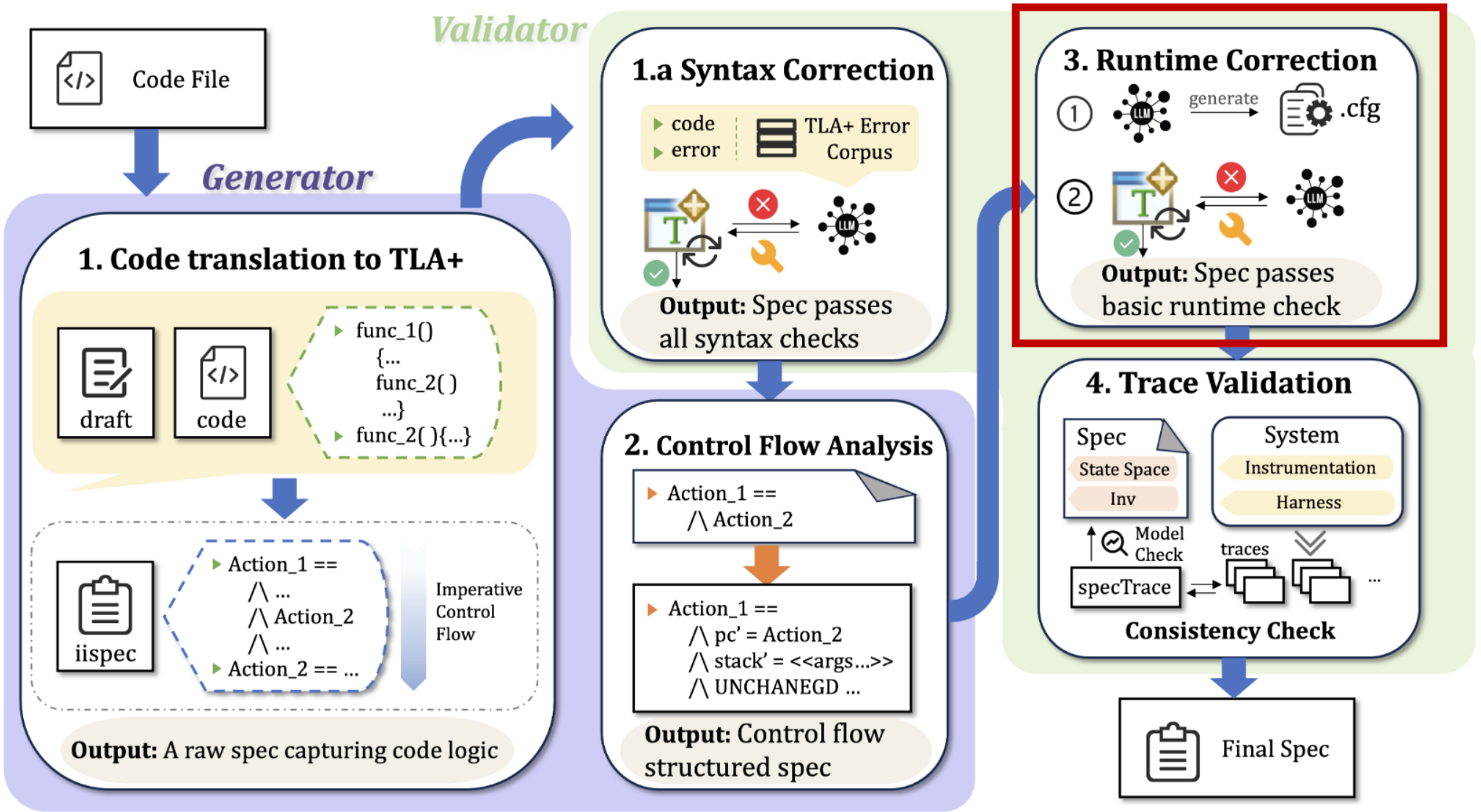
Specula design overview



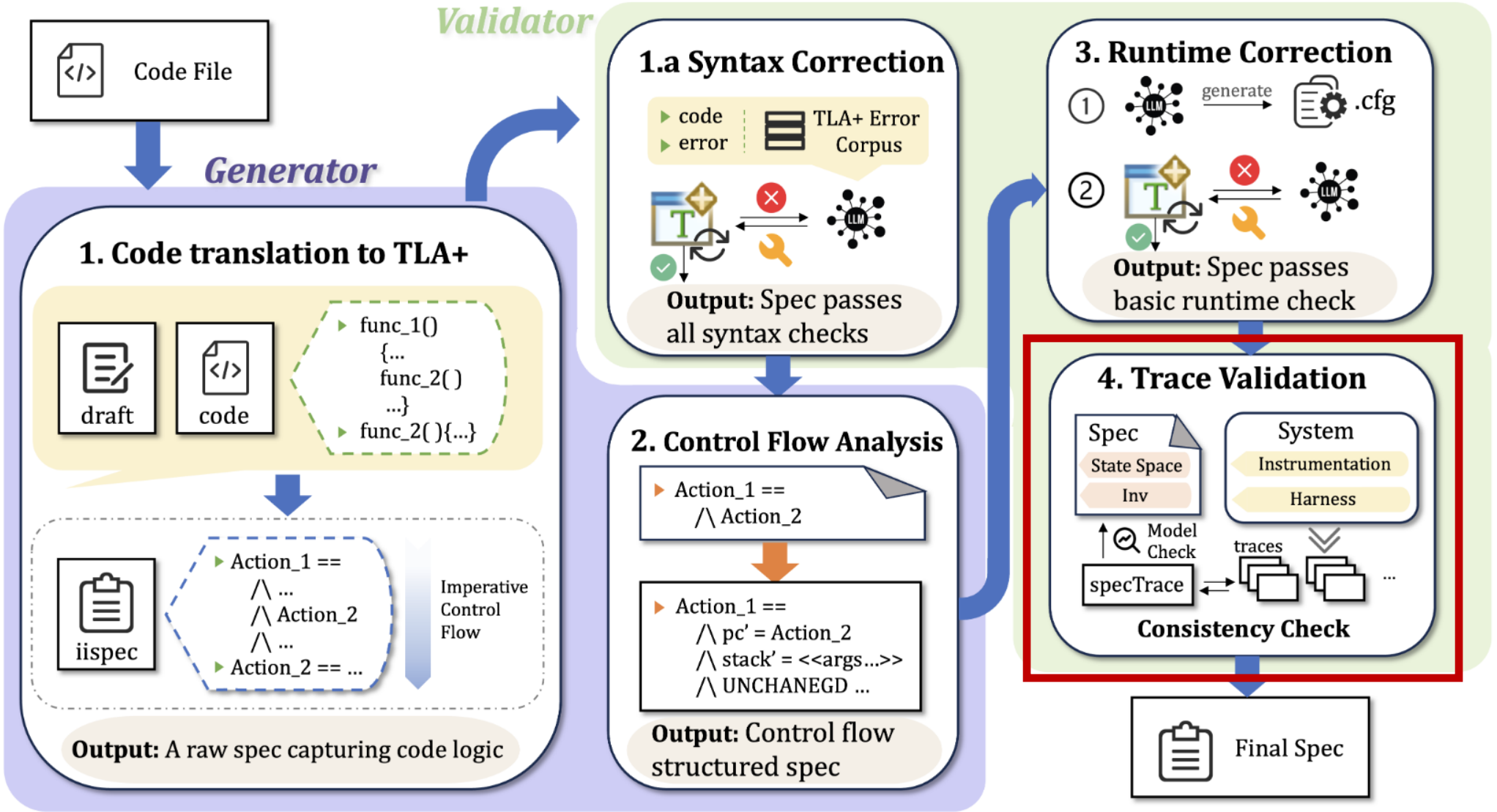
Specula design overview



Specula design overview



Specula design overview



Specula demo

- **Use Etcd's Raft as the target system artifact**
- **Specula is currently an internal tool**
  - We will manually fix errors (unlike in the benchmark)
- **Demo code available in the repo**
  - github.com/specula-org/Specula/tree/main/demo/etcd

Step-1: Code-to-spec translation

Code

```
func (r *raft) becomeFollower(term uint64,
  lead uint64) {
  r.step = stepFollower
  r.reset(term)
  ...
}
```

Prompt

You are a TLA+ specification expert.
Classify each function into:
- CORE protocol logic
- SUPPORTING utilities
- UTILITY code

- Critical logic: Set state = ...
- Modeling approach: Full

A draft summarizes code behavior in a natural language

Step-1: Code-to-spec translation

Code + Draft

```
func (r *raft) becomeFollower(term uint64,
  lead uint64) {}
```

```
### becomeFollower
- Purpose: Transition to follower
-    Critical logic: Set state = ...
- Modeling approach: Full
```

Translated Spec Prompt

You are a TLA+ specification expert.
- Implement ALL core functions identified in the draft.
- Preserve execution and conditional logic.
- Model core logic fully; abstract helpers; omit utility code.

currentTerm[s] THEN None ELSE votedFor[s]]

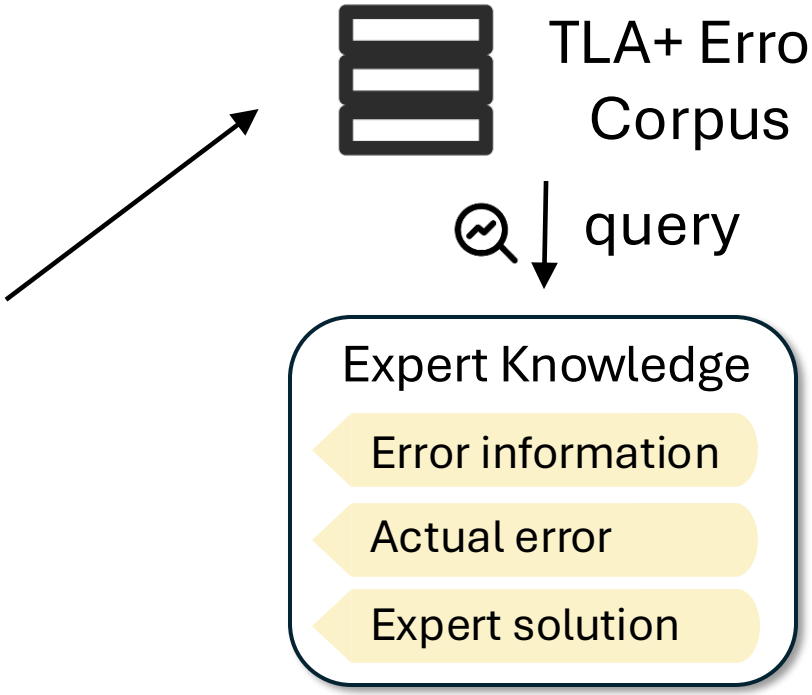
One-to-one mapping from core functions to TLA+ actions
- A core function starts the procedure of a state transition

Step-1.a: Fixing syntax errors using RAG

```
stepFollower(s, m) ==
 /\ IF stmt
    THEN messages' = messages \cup {new_m}
 /\ ...
```

Error info: **Encountered** "/\"
at line 375, column 8 and token "}"

Unrelated to the actual error: Missing ELSE branch



RAG: Expert-style knowledge base to help agents automatically correct syntax issues

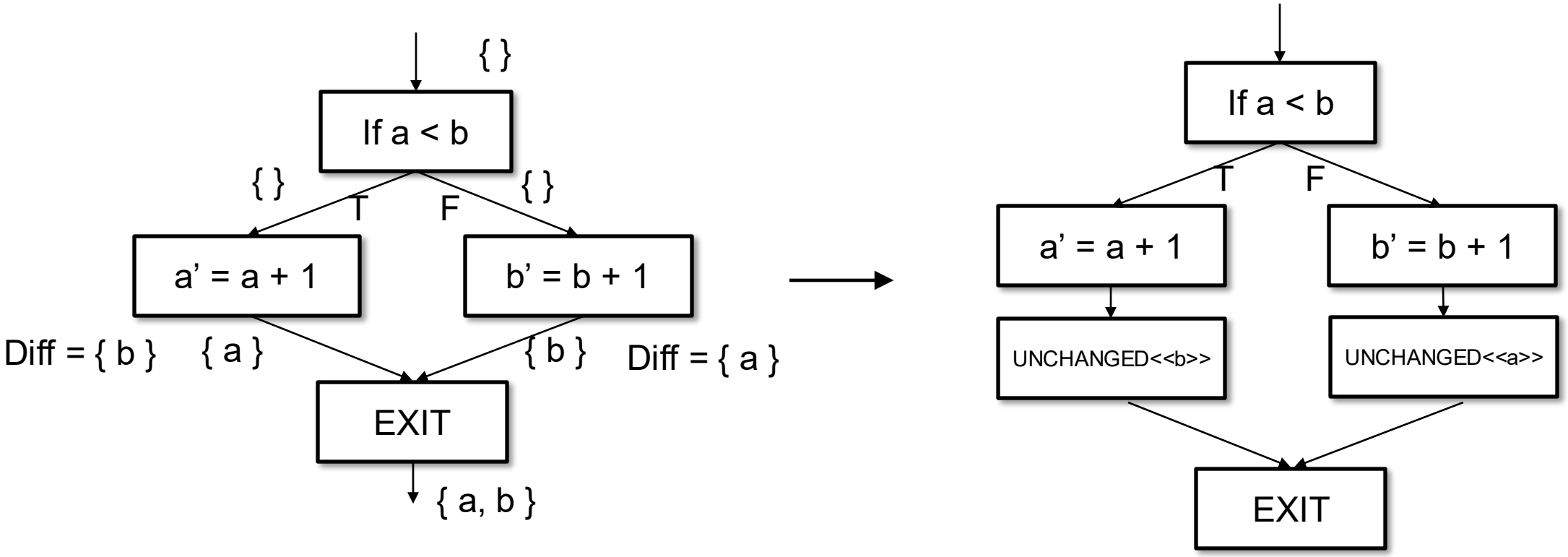
Step-2: Fixing non-atomic actions

```
poll(s, from, msgType, granted) ==
  /\ votesGranted' = new_votes
  /\ IF stmt THEN becomeFollower(...)

becomeFollower(s, term, leader) ==
  /\ votesGranted' = [votesGranted EXCEPT ![s] = [t \in Server |-
> FALSE]]
```

```
poll ==
  /\ votesGranted' = new_votes
  /\ IF stmt THEN pc' = Poll_1

poll_1 ==
  /\ becomeFollower(...)
  /\ ...
```

- **Splitting one complex action into multiple atomic actions**
- **Avoiding multiple assignments silently skipped by TLC**

Step-2: Auto-inserting UNCHANGED

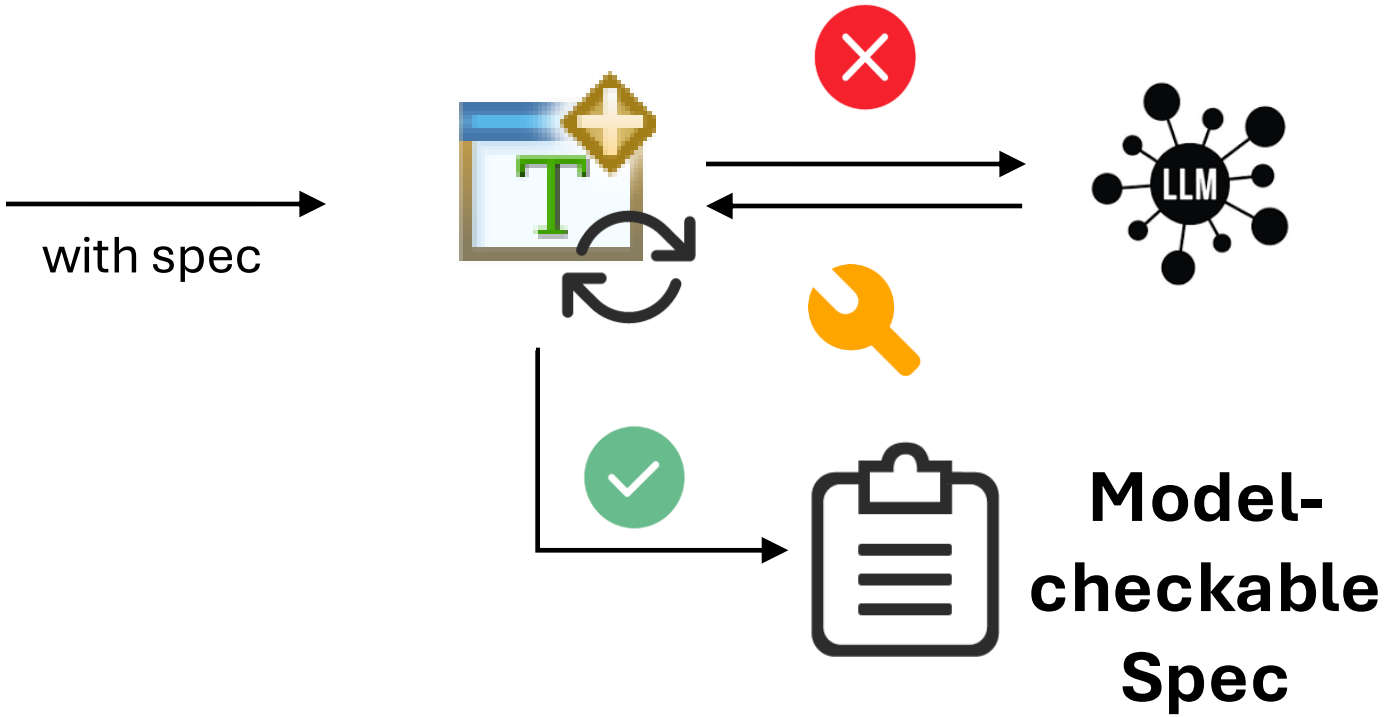


Using Control-Flow Analysis (CFA) to track modified variables
→ Automatically inserts UNCHANGED statements

Step-3: Runtime correction

.cfg file

```
SPECIFICATION Spec

CONSTANTS

Server = {1, 2, 3}

MaxTerm = 3

MaxLogLen = 3

None = 0

Nil = "Nil"
```



- Using the LLM to generate the configuration file
- Running TLC for one hour (pass if no error)

Step-4: Automated instrumentation

```
// tickElection is run by followers and candidates
after r.electionTimeout.
func (r *raft) tickElection() {
traceSpecAction("tickElection", r)
r.electionElapsed++
}
```

```
1 received MsgPreVoteResp from 2 at term 1
1 has received 2 MsgPreVoteResp votes and 0 vote rejections
1 became candidate at term 2
1 [logterm: 1, index: 3] sent MsgVote request to 2 at term 2
1 [logterm: 1, index: 3] sent MsgVote request to 3 at term 2
1 received MsgVoteResp from 1 at term 2
1 has received 1 MsgVoteResp votes and 0 vote rejections
3 [term: 1] received a MsgVote message with higher term from 1 [term: 2]
2 [term: 1] received a MsgVote message with higher term from 1 [term: 2]
2 became follower at term 2
2 [logterm: 1, index: 3, vote: 0] cast MsgVote for 1 [logterm: 1, index: 3

3 became follower at term 2
3 [logterm: 1, index: 3, vote: 0] cast MsgVote for 1 [logterm: 1, index: 3

1 received MsgVoteResp from 2 at term 2
1 has received 2 MsgVoteResp votes and 0 vote rejections
1 became leader at term 2
```

- **Running a three-node Etcd cluster**
- **Collecting implementation-level execution traces**

Step-4: Trace validation

```
Computing initial states...
Finished computing initial states: 1 distinct state generated
<< "Failed matching the trace to (a prefix of) a behavior:",
   [ state |->
        << [ op |-> "Update",
             args |->
                 << [ 1 |-> "StateFollower",
                      2 |-> "StateFollower",
                      3 |-> "StateFollower" ] >>,
             path |-> <<>> ] >>,
     currentTerm |->
         << [ op |-> "Update",
              args |-> <<[1 |-> 1, 2 |-> 0, 3 |-> 0]>>,
              path |-> <<>> ] >>,
     event |-> "Step" ],
   "TLA+ debugger breakpoint hit count 4" >>  FALSE
Error: Assumption line 65, col 5 to line 68, col 84 of module
2 states generated, 2 distinct states found, 0 states left on
The depth of the complete state graph search is 2.
Finished in 01s at (2025-11-16 21:41:22)
Command failed (exit 10): java -cp /home/ubuntu/specula/lib/t
race.tla
```

```
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
  calculated (optimistic):  val = 5.9E-11
68791 states generated, 43696 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 47.
The average outdegree of the complete state graph is 1 (minimum is 0, the ma
e is 6).
```

- **Refining (manually) the spec using feedback**
- **Obtaining a system spec that conforms to the code**

Specula results (no human assist)

Raft etcd

Agent	LLM	Syntax	Runtime	Conformance	Invariant
Basic Agent	Claude-Sonnet-4	100.00% ✓	25.00% ✓	7.69%	69.23%
	GPT-5	47.87% ✗	-	-	-
	Gemini-2.5-Pro	50.00% ✗	-	-	-
	DeepSeek-R1	50.00% ✗	-	-	-
Specula	Claude-Sonnet-4	100.00% ✓	66.67% ✓	15.38%	92.31%
	GPT-5	100.00% ✓	20.00% ✗	-	-
	Gemini-2.5-Pro	44.44% ✗	-	-	-
	DeepSeek-R1	100.00% ✓	0.00% ✗	-	-
Trace Learning	Claude-Sonnet-4	50.00% ✗	-	-	-
	GPT-5	48.78% ✗	-	-	-
	Gemini-2.5-Pro	42.31% ✗	-	-	-
	DeepSeek-R1	47.73% ✗	-	-	-

Observations

- **LLMs can model simple systems like spinlock but are limited in comprehending complex ones like Raft.**
  - Would love to see the TLAiBench results
- **Conformance checking is critically important.**
  - LLMs often memorize the TLA+ specs in its training data
  - Can be made into feedback loops
- **Leveraging AI's coding ability as a short-term solution**
  - Many efforts in advancing "system intelligence"
- **LLMs do ok on safety, but poorly on liveness**
  - Can LLMs do temporal reasoning?

Ongoing work

- **Continuously improving Specula's effectiveness**
  - e.g., leveraging runtime traces
- **Bug finding as the utility of generated TLA+ specs**
  - Using the generated TLA+ model to find old and new bugs
- **Human-in-the-loop designs**
  - Maintainability of AI-generated TLA+ specs
- **How to make Specula a tool for the community?**
  - Any feedback is highly appreciated