

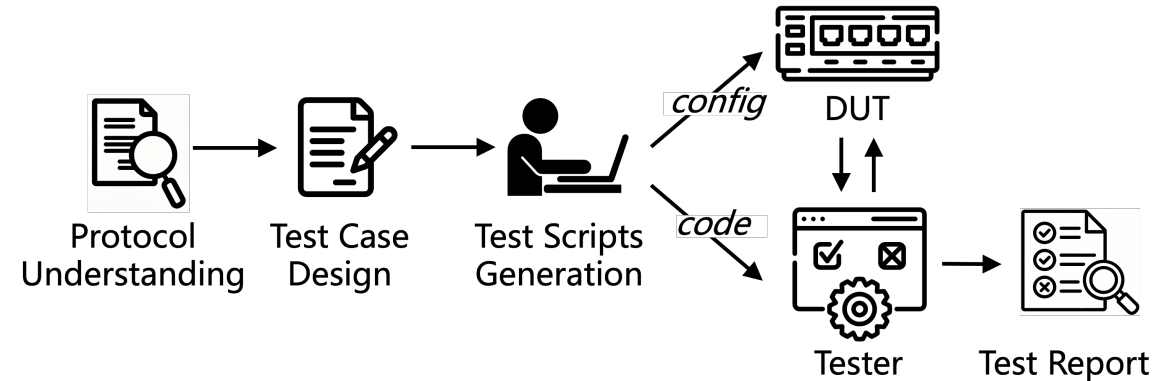
# Framework and Automation Levels for AI-Assisted Network Protocol Testing

draft-cui-nmrg-auto-test-01

Yong Cui, **Yunze Wei**, Kaiwen Chi, Xiaohui Xie

Tsinghua University

# Background



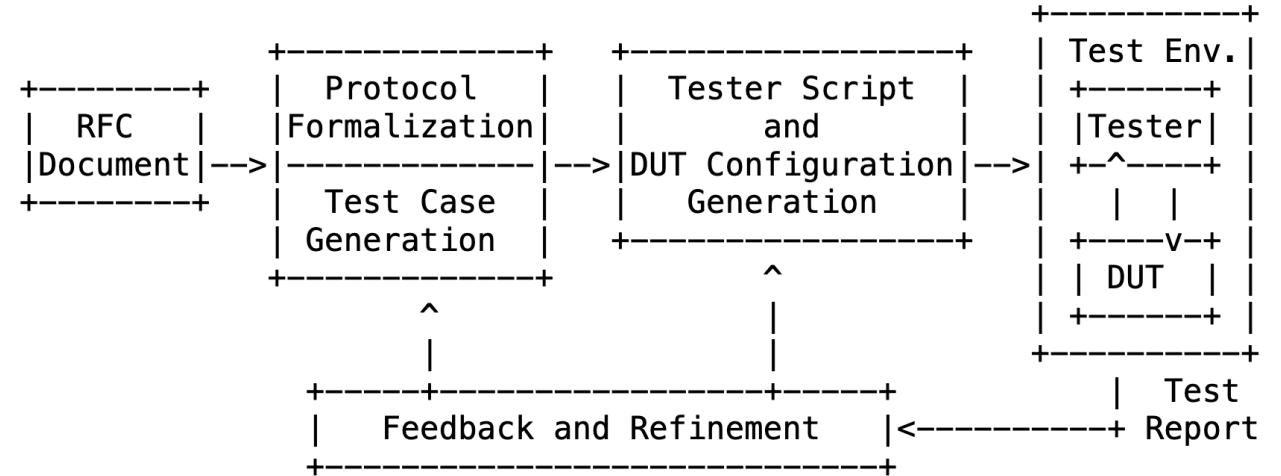
- Network Protocol Testing
  - Test protocol implementations to ensure conformance, performance, security, etc.
- Traditional Network Protocol Testing is Labor-Intensive
  - Engineers need to analyze RFCs documents, design test cases, convert them into executable scripts for both testers and devices under test (DUTs), execute the test and analyze reports
  - Drawbacks: **Low efficiency, Limited coverage**
  - Struggling to adapt to the rapid evolution of network protocols in new scenarios, such as Industrial Internet, Satellite Networks, and Modern DCNs.

# Draft Overview (Recap)

This draft introduces:

## 1. A framework for AI-assisted protocol testing

- Protocol formalization
- Test case generation
- Tester script & DUT configuration generation
- Execution and feedback



- **Input**
  - RFC document
- **Output**
  - Test cases, Tester script, DUT configuration
  - Test Report (Final result)

# Draft Overview (Recap)

This draft introduces:

## 2. Automation Maturity Levels (0-5) for protocol testing

- A reference model describing the evolution from manual testing to fully autonomous testing systems.
- This serves as a technology roadmap for protocol testing automation

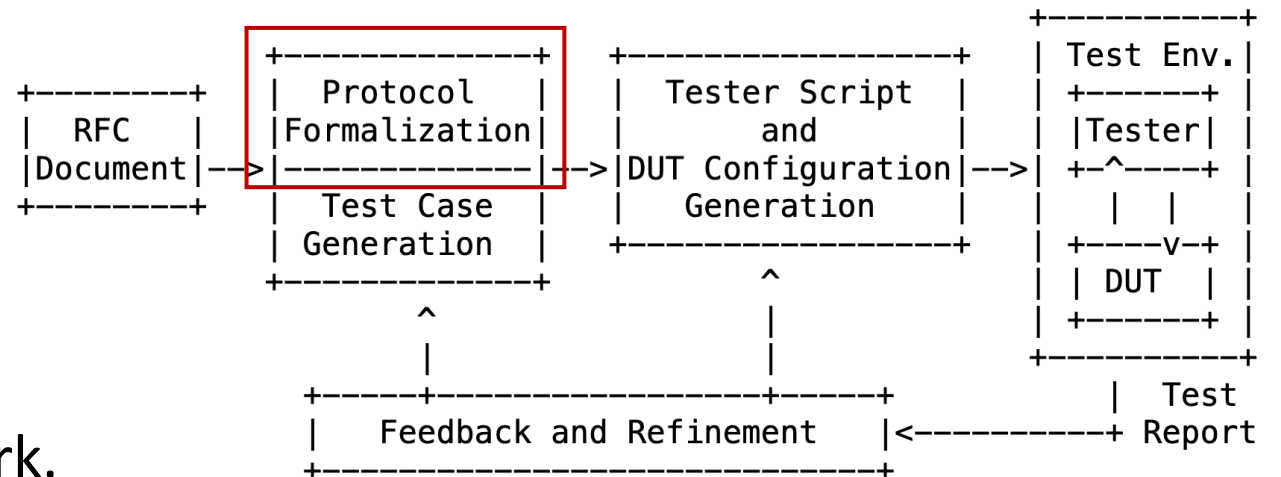
## 3. A Concrete LLM-based example for automated testing

| Level | RFC Interpretation              | Test Asset Generation & Execution                      | Result Analysis & Feedback                        | Human Involvement                       |
|-------|---------------------------------|--|---|---|
| 0     | Manual reading                  | Fully manual scripting and CLI-based execution         | Manual observation and logging                    | Full-time intervention                  |
| 1     | Human-guided parsing tools      | Script templates with tool-assisted execution          | Manual review with basic tools                    | High (per test run)                     |
| 2     | Template-based extraction       | Basic autogen of config & scripts for standard cases   | Rule-based validation with human triage           | Moderate (Manual correction and tuning) |
| 3     | Rule-based semantic parsing     | Parameterized generation and batch orchestration       | ML-assisted anomaly detection                     | Supervisory confirmation                |
| 4     | Structured model interpretation | Objective-driven synthesis with end-to-end automation  | Correlated failure analysis and report generation | Minimal (strategic input)               |
| 5     | Adaptive protocol modeling      | Self-adaptive generation and self-optimizing execution | Predictive diagnostics and remediation proposals  | None (optional audit)                   |

# Key Updates in draft-01 (Framework)

- Change *Protocol Understanding* to *Protocol Formalization*
  - v00: *Protocol Understanding*, a loosely defined extraction process with Human Intent as an explicit input alongside RFC documents
  - v01: *Protocol Formalization*, a rigorous, structured encoding process promoting comprehensive test case coverage in subsequent steps.

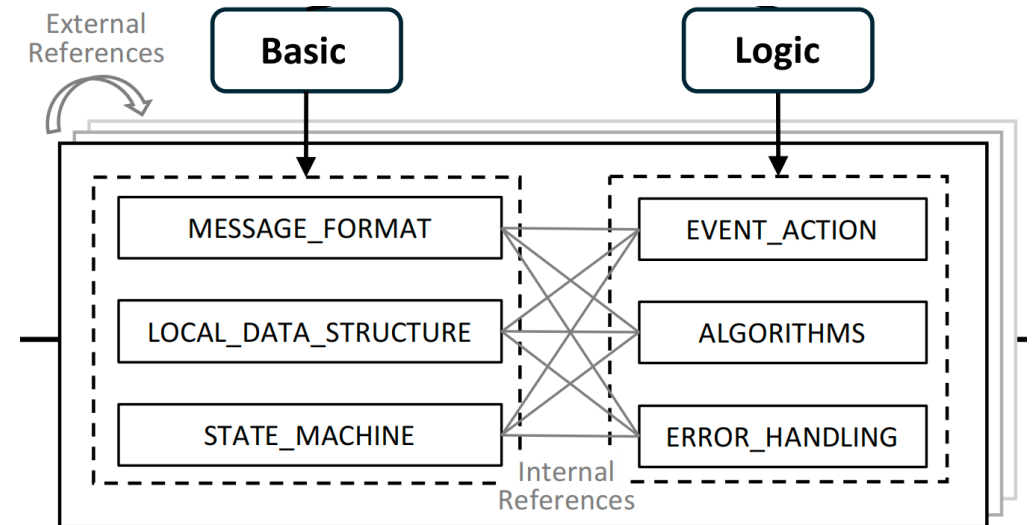
- Protocol Formalization is explicitly defined as a stage in the Figure.
- Human Intent is removed as a named input to emphasize the formalized, systematic framework.



# Key Updates in draft-01 (LLM-based Example)

## + Protocol Formalization

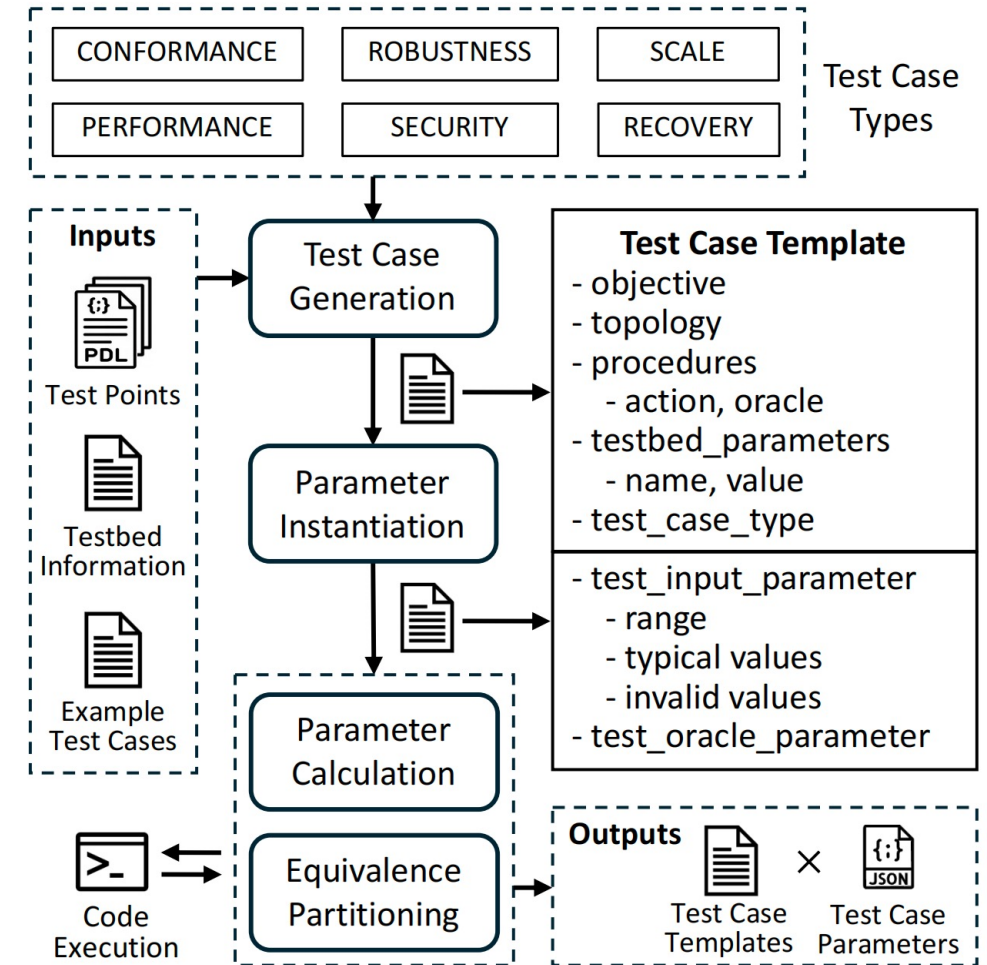
- **Basic:** Message format, local data structures, finite state machines, etc.
- **Logic:** Event-action rules, algorithms, error handling, etc.
- Captures both internal and external relationships.
- Provides more precise guidance with reduced ambiguity, enabling Large Language Models (LLMs) to effectively generate and refine test cases with clear context and minimal interpretation.



# Key Updates in draft-01 (LLM-based Example)

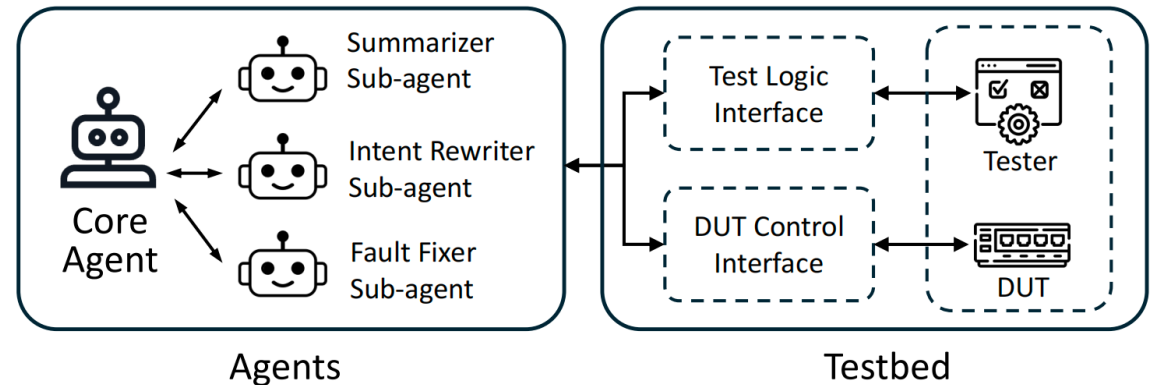
## + Test Case Templates with Decoupled Parameters

- Decoupling test case templates from parameter instantiation enhances LLM reliability when handling well-scoped subtasks.
- Parameter calculation and equivalence partitioning to improve testing efficiency
- A hybrid approach combining model-free (LLM) and model-based (coding) methods



# Key Updates in draft-01 (LLM-based Example)

- From Single Agent to Multi-Agent Test Code Generation
  - **v00**: A single agent performing multiple roles (test code generation, testbed interaction, debugging)
  - **v01**: A multi-agent framework where the core agent generates test code, while sub-agents handle other auxiliary tasks such as test intent rewriting and bug fixing



# Next Steps

- Further Refinement and Standardization of the Draft
  - Discuss key parts of the draft that are worth standardization
  - Refine these parts with more details
- Discussion on Automation Testing Evaluation
  - One of the key challenges in AI-assisted protocol testing is defining and agreeing upon effective **evaluation metrics** for test cases
  - Collaboration with BMWG?
- More Industry Collaborations
  - Ensuring that the proposed framework meets practical real-world needs
- **We sincerely appreciate your valuable comments and feedback!**

# Thank You!

draft-cui-nmrg-auto-test-1

Yong Cui, **Yunze Wei**, Kaiwen Chi, Xiaohui Xie

Tsinghua University