

Specification-Guided Synthesis of Deadlock-Free Communication Protocol Refinements with Large Language Models

Yang Li Ping Hou Nobuko Yoshida

University of Oxford, Computer Science

ASE'26, Munich, Germany

Outline

- 1 Motivation
- 2 Syntropy Framework
- 3 Evaluation
- 4 Conclusion

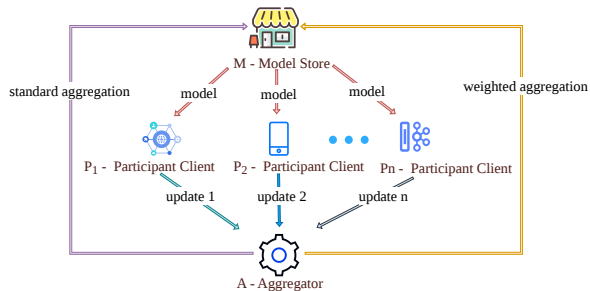
- LLMs are strong at code generation and program synthesis, but **lack behavioural guarantees** in systems with complex interactions.
- Distributed systems rely on **communication protocols**; as systems evolve, protocols must be **refined** while preserving correctness.
- Manual refinement is error-prone: even minor changes can introduce **deadlocks**.
- Refinement is **undecidable** in general – motivating automatic, LLM-based synthesis.
- **Research question:** How can protocol refinements be systematically synthesised while retaining behavioural correctness?

Example: Federated Learning Protocol

Roles: model store m , clients p, q , aggregator a

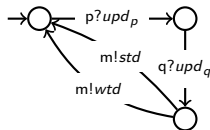
- 1 Model distribution: m sends model to all clients
- 2 Update submission: clients send updates to a
- 3 Aggregation result: a sends std/wtd result to m

Under asynchronous FIFO messaging, a may receive q 's update *before* p 's $\Rightarrow$ requires a **refined** local protocol.

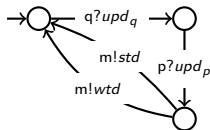


Session Trees and Asynchronous Subtyping

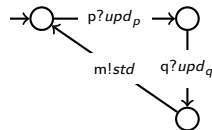
- Local session type for a : $T_a = \mu t. p?upd_p. q?upd_q. \oplus \{m!std.t, m!wtd.t\}$
- Reordering gives $T'_a = \mu t. q?upd_q. p?upd_p. \oplus \{m!std.t, m!wtd.t\} \Rightarrow T'_a \leq T_a$
- Asynchronous Multiparty Subtyping (AMS) allows:
 - C1 covariance (fewer outputs)
 - C2 contravariance (more inputs)
 - R1 anticipate input from p before a finite number of inputs not from p
 - R2 anticipate output to p before a finite number of inputs (from any participant), and before outputs not to p
- AMS is **undecidable** in general.



(a) FSM for a



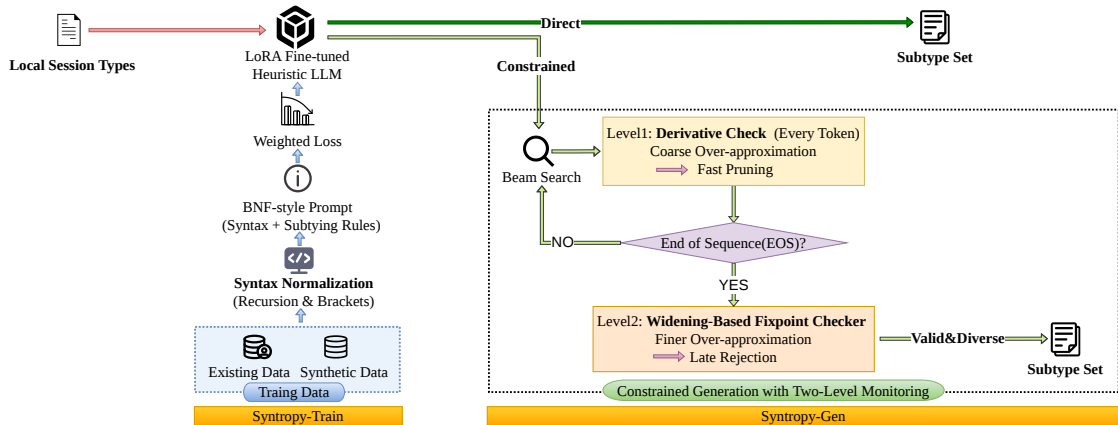
(b) Reordered FSM for a



(c) Covariant FSM for a

Figure: FSMs for role a

Syntropy: Overview



Syntropy-Train: Learning Subtype Generation

- Fine-tune open-source LLMs (e.g. Qwen2.5-Coder-7B-Instruct) with **LoRA** (rank 64, $\alpha = 32$) to generate subtypes conditioned on a supertype.
- Training data: literature-derived MPST specs + synthetic supertypes, encoded as $\{S, [\{T_1, l_1\}, \dots, \{T_n, l_n\}], n\}$.
- Transformation rules used as prompt context:

Rule	Meaning
Identity	subtype identical to supertype
Unfold	unfold recursion
RefA / RefB	reorder receive & send (R1/R2)
RefOut / RefIn	covariance / contravariance (C1/C2)

- Weighted token-level loss: 1.0 on subtype tokens, 0.2 on auxiliary fields (labels, counts).

Syntropy-Gen: Constrained Generation

- **Level 1 (Derivative Check):** per-token, prune beams whose partial session tree T' fails $\text{Feasible}(T', T_S)$.
- **Level 2 (Widening-Based Fixpoint Check):** on EOS, verify full tree via $\text{SimCheck}(T_f, T_S)$ with widening for termination on recursive types.

Algorithm Constrained Generation with Two-Level Monitoring

Require: supertype S , language model M , beam size k

Ensure: set of candidate subtypes Ω

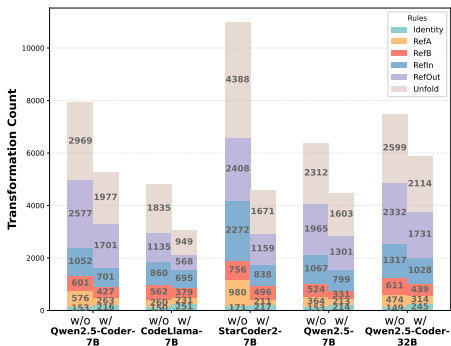
```
1:  $\mathbb{T}_S \leftarrow \text{PARSE}(S)$ 
2:  $\text{Beams} \leftarrow \{(\epsilon, 0.0)\}$ 
3:  $C \leftarrow \emptyset$ 
4: for each decoding step do
5:   for each  $(\text{seq}, \text{score}) \in \text{Beams}$  do
6:     for each  $(t, p) \in \text{Top}_{2k}(M)$  do
7:        $\text{seq}' \leftarrow \text{seq} \cdot t$ 
8:        $\mathbb{T}' \leftarrow \text{PARSEPREFIX}(\text{seq}')$  ▷ Level 1 (Derivative Check)
9:       if  $\mathbb{T}' \neq \perp$  and  $\text{FEASIBLE}(\mathbb{T}', \mathbb{T}_S)$  then
10:        if  $t = \text{EOS}$  then
11:           $\mathbb{T}_f \leftarrow \text{PARSE}(\text{seq}')$  ▷ Level 2 (Widening-Based Fixpoint Check)
12:          if  $\text{SIMCHECK}(\mathbb{T}_f, \mathbb{T}_S)$  then
13:             $\Omega \leftarrow \Omega \cup \{\text{seq}'\}$ 
14:        else
15:           $C \leftarrow C \cup \{(\text{seq}', \text{score} + \log p)\}$ 
16:    $\text{Beams} \leftarrow \text{Top}_k(C)$ 
```

Experimental Setup

- **Models:** Qwen2.5-Coder-7B (primary), CodeLlama-7B, StarCoder2-7B, Qwen2.5-7B, Qwen2.5-Coder-32B.
- **Data:** 704 training supertypes $\rightarrow$ 10,800 (supertype, subtype) pairs; 100 test supertypes (43 literature + 57 synthetic).
- **Metrics:**
 - Syntactic validity
 - Semantic validity
 - Reordering vs. variance validity
 - Transformation rule distribution
 - Output complexity

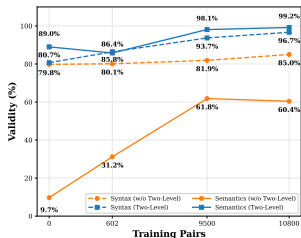
RQ1: Benchmark Performance & Generalisation

- Without Monitoring: syntactic 85.0%, semantic **60.4%** on Qwen2.5-Coder-7B.
- With Monitoring: syntactic **96.7%**, semantic **99.2%** on Qwen2.5-Coder-7B.
- Across all 5 LLMs With Monitoring: syntactic 95.4%–98.1%, semantic up to **99.5%** $\Rightarrow$ strong cross-model generalisation.
- Transformation distribution: Within each LLM, it remains comparable; across different LLMs, it is generally consistent, though some variation exists between models.



RQ2–RQ4: Data Scale, Ablation & Robustness

RQ2 – Training data scale RQ3 – Ablation



- w/o Two-Level Monitoring: semantic validity drops to 60.4%
- w/o Prompt: semantic validity 93.9%
- w/o Fine-tuning: semantic validity 89.0%, less diversity
- Full Syntropy: best overall (99.2% semantic validity)

RQ4 – Robustness to reordering vs. variance

- Both reordering (RefA/RefB) and variance (RefIn/RefOut) achieve high semantic validity: **89.8%–100%** across models.
- Variance transformations are generated **2–3× more often** than reordering $\Rightarrow$ model prefers simpler patterns.
- Reordering frequency can be partially increased via targeted constraints.

Comparison with Frontier LLMs

- GPT-5.5: 18% benchmark coverage; DeepSeek-V4-Pro: 4% coverage.
- Fine-tuned Syntropy models: **100% coverage** across all five backbones.
- Frontier models show comparable validity *when they do respond*, but at much lower coverage and fewer candidates per supertype (8.22/5.75 vs. 19.20–35.30).
- Conclusion: Syntropy is necessary for comprehensive protocol refinement.

Conclusion & Future Work

- **Syntropy**: first framework combining LLMs with MPST to synthesise protocol refinements with guaranteed deadlock freedom.
- Two-level monitoring (derivative check + widening-based fixpoint check) is essential for semantic correctness.
- Achieves 95.6%–99.5% validity, generalises across 5 LLMs, produces diverse non-trivial refinements.
- **Future work**: choreographies, contract-based models, temporal logics; verifier-guided iterative refinement; integration into practical development workflows.

Specification-Guided Synthesis of Deadlock-Free Communication Protocol Refinements with Large Language Models

Thank you!

Questions?

Appendix: Transformation Rules - Details

Rule	Prompt	Example
Identity	Subtype identical to supertype.	–
Unfold	Unfold recursion via substitution of the recursive variable.	VALID $\text{REC_X_OPEN } p?m; p!m'; \text{REC_X_CLOSE} \Rightarrow$ $p?m; p!m'; \text{REC_X_OPEN } p?m; p!m'; \text{REC_X_CLOSE}$
RefA	Move recv $p?m$ earlier when roles differ.	VALID $p?m; q?m'; T \leq q?m'; p?m; T$ INVALID $p?m; p?m'; T \leq p?m'; p?m; T$
RefB	Move send $p!m$ earlier across independent actions.	VALID $p!m; q?m'; T \leq q?m'; p!m; T$ INVALID $p!m; p!m'; T \leq p!m'; p!m; T$
RefOut	Internal choice: subtype offers fewer labels.	VALID $\text{lbrace } p!m; T_1 \text{ rbrace} \leq$ $\text{lbrace } p!m; T_1, p!m'; T_2 \text{ rbrace}$ INVALID $\text{lbrace } p!m; T_1, p!m'; T_3 \text{ rbrace} \leq$ $\text{lbrace } p!m; T_1, p!m'; T_2 \text{ rbrace}$
RefIn	External choice: subtype accepts more labels.	VALID $\text{lbrace } p?m_1; T_1, p?m_2; T_2, p?m_3; T_3 \text{ rbrace} \leq$ $\text{lbrace } p?m_1; T_1, p?m_2; T_2 \text{ rbrace}$ INVALID $\text{lbrace } p?m_1; T_1 \text{ rbrace} \leq$ $\text{lbrace } p?m_1; T_1, p?m_2; T_2 \text{ rbrace}$