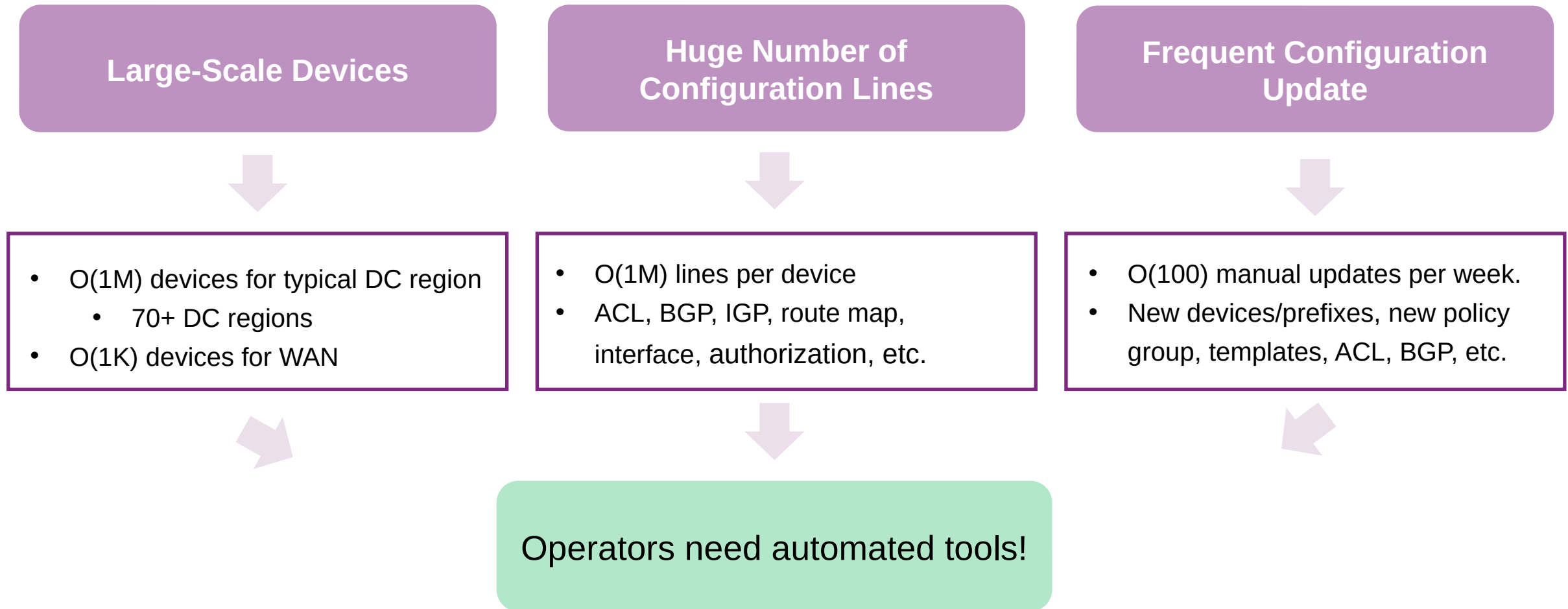


Towards Intelligent Network Configuration Management with LLMs

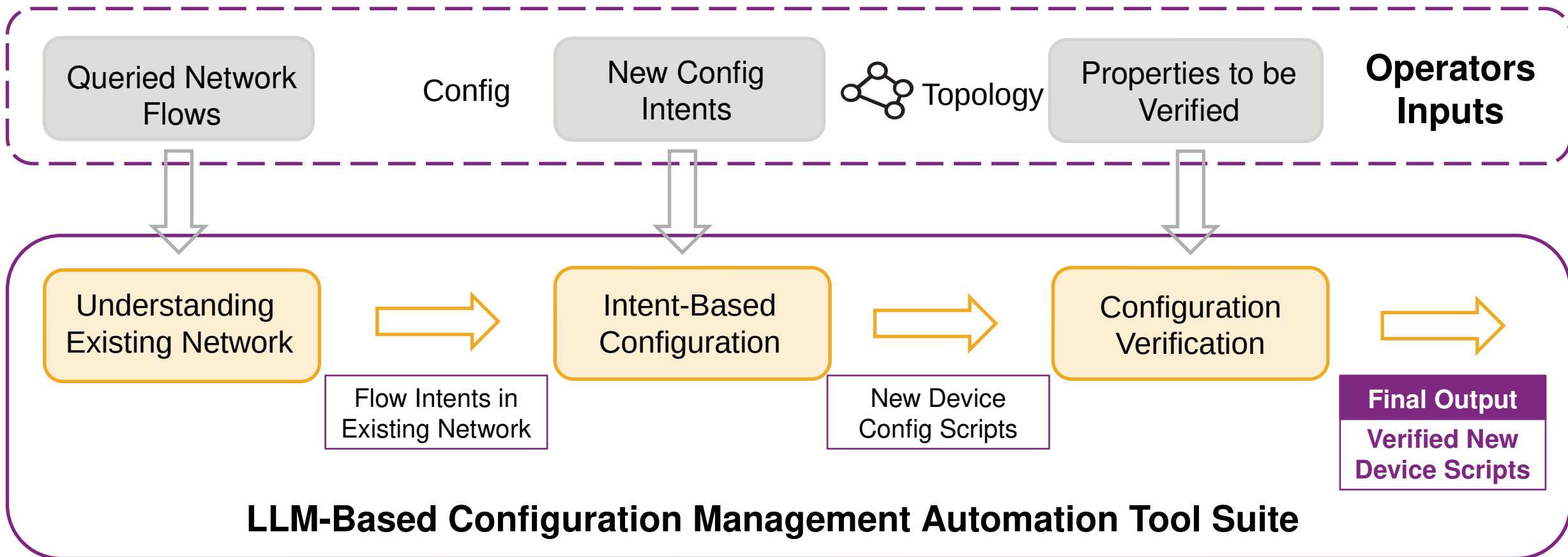
Wenlong Ding

PhD Candidate, CSE, CUHK

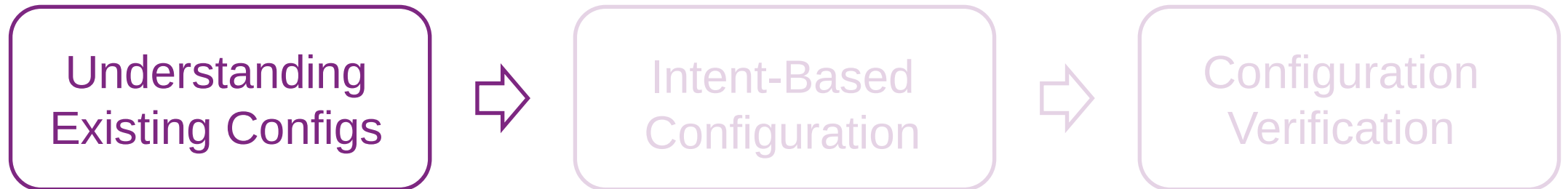
Configuration Management is Complex



Contribution Overview



In Search of Lost Time: Intent Recovery in Network Configurations



Intent Recovery in Existing Networks

Problem: given an observed network state of interest, infer *why* it exists with evidence.

- *Verification* asks **whether** a property holds.
- *Specification Mining* asks **what** properties hold.
- *Intent Recovery* asks **why** a property holds.



Why can 10.0.1.5 reach 10.0.2.8 on port 443?

Q: Why the problem exists?

A: Evolution of existing configuration is lossy due to context decay.

Q: Why hard?

A: Intent recovery is ill-posed with inherent ambiguity (e.g., config for intents are distributed across devices and times).

Q: Why deserve solving?

A: Recovered intents enable safe subsequent Intent-Based Networking (IBN).

Key Insight: Recover Intent via Context

Observation: The *snapshot* alone is underdetermined, but the original design leaves *residual traces* in surrounding artifacts.



Insight: Use context as regularization.

What are the useful residual traces and how to mine the context?

Context 1: Semantics

Data source: IPAM (names, tags, hierarchy for prefix/entity)
Ch1: **search-space explosion**
(many-to-one entity mapping)

Context 2: Provenance

Data source: config snapshot
Ch2: **reverse tracing**
(Distributed config across devices for flow intents)

Context 3: History

Data source: Git evolution
Ch3: **shadowing detection**
(Config of different intents override each other)

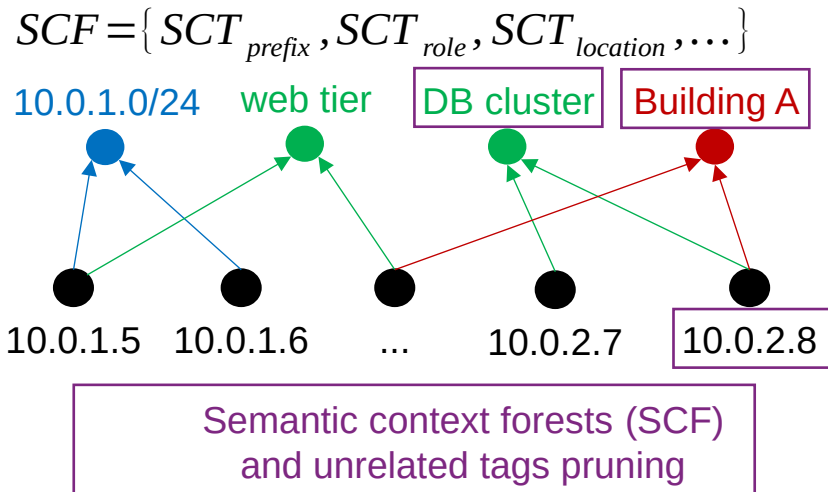
Context Mining Design: Three Engines

Semantic Engine

Input: IPAM (prefixes □ metadata tags)

Output: Multi-level abstractions

Operations: LLM-based SCF building and pruning



Get multi-level semantic candidates

Provenance Engine

Input: Config snapshot & flow query

Output: Contributing policy rules

Operations: Rules extraction and LLM-based affinity-based filtering

Get all configs on trace-route devices
tracert(10.0.1.5, 10.0.2.8)

Affinity-based filtering for intent flows

Contributing Lines

- A.cfg: ACL https-access
- B.cfg: route-map ...
- ...

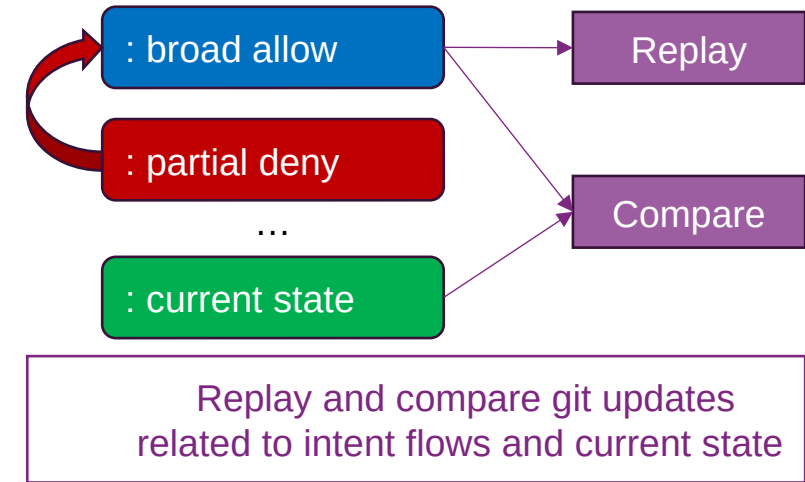
Get grouped contributing policies

Temporal Engine

Input: Git history of config changes

Output: Overridden rule sequence

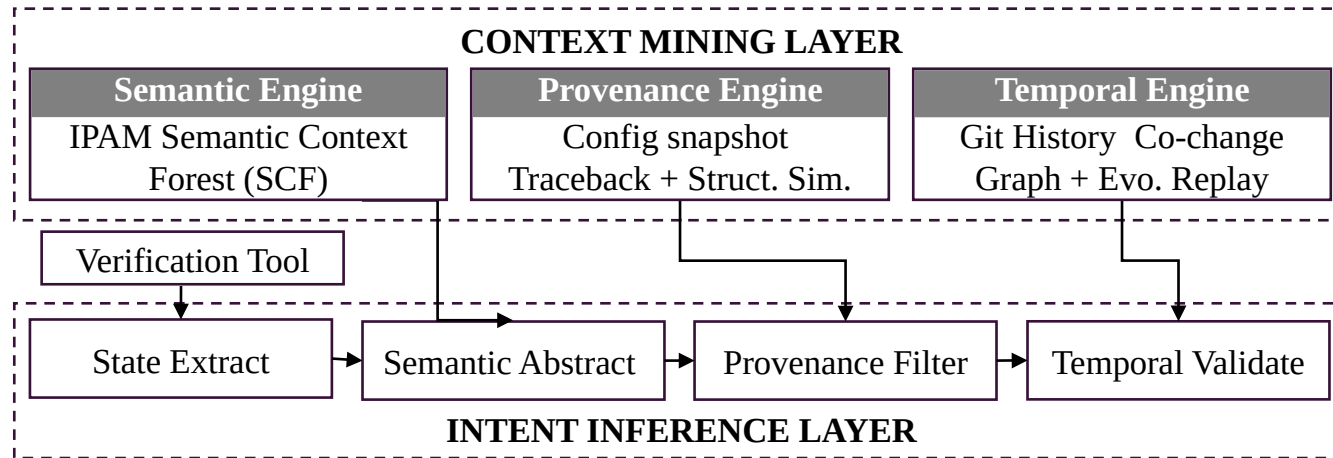
Operations: LLM-based replay of Git history to detect shadowed rules



Get a timeline of related rules

Preliminary framework & results

- **Preliminary Design: deterministic heuristic pipeline** that progressively applies each context dimension as filters.



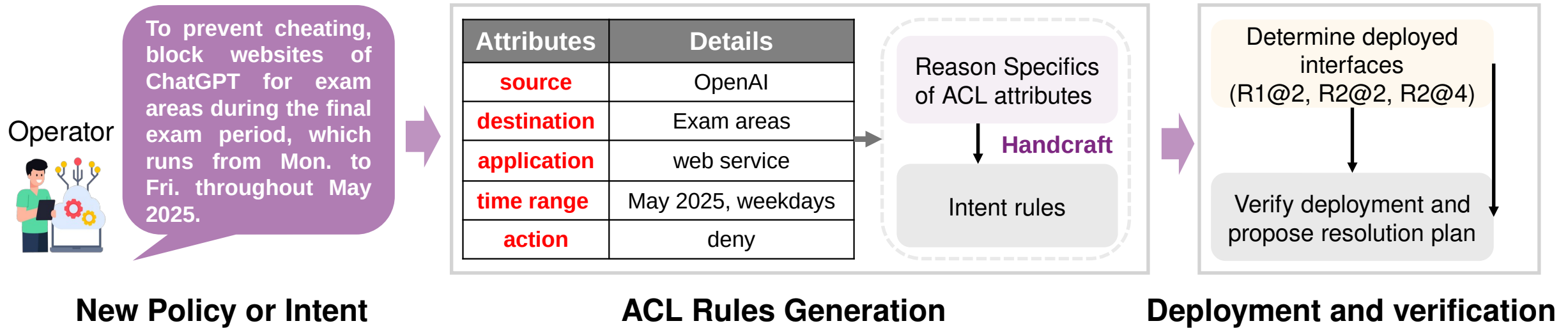
- The full pipeline achieves **0.71** Exact Match accuracy and **0.79** Intent F1 score.

- **Future work:** A joint-optimization framework, as the contexts in the three dimensions are intertwined.

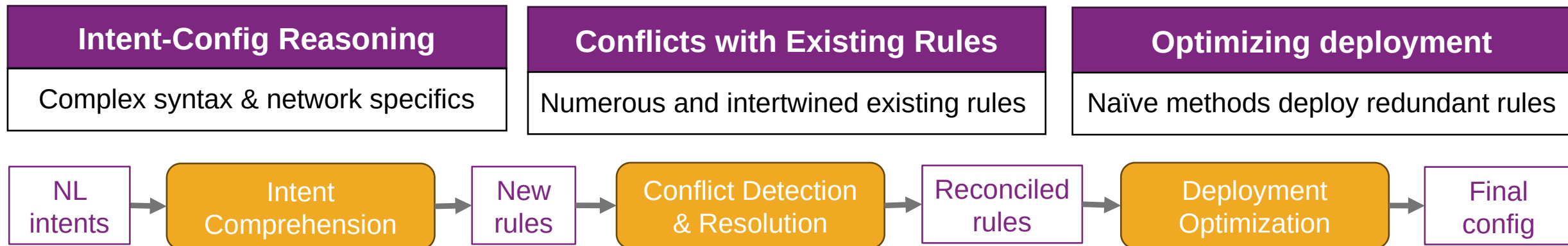
Xumi: Automating Conflict-Aware ACL Configuration with Natural Language Intents



Common Manual ACL Configuration & Automated Pipeline



➤ Automation Challenges & Xumi's pipeline

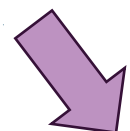


Intent Comprehension

Field	Format Specification	Chain-of-Thought Reasoning
source		
destination		
application		
time		
action	deny or permit or protect	

ACL IR Templates **Feedback Prompts**

LLM is prone to hallucinations



src	OpenAI: ([R _A @1, R _B @1], [104.18.33.170/32, 184.105.99.79/32, ...])
dst	Exam Area: ([R ₁ @2, R ₂ @2], [10.0.1.0/24]), ([R ₂ @4], [10.0.2.0/24])
app	HTTP: [tcp(80)], HTTPS: [tcp(443)],
time	Absolute: 1 May 2025 to 31 May 2025, Periodic: weekdays
action	deny

Result IR



Enumerations

Intent ACL Rules:

- S1:** 104.18.33.170/32, 10.0.1.0/24, tcp(80), 2024/05:weekdays, deny
- S2:** 104.18.33.170/32, 10.0.2.0/24, tcp(80), 2024/05:weekdays, deny
- S3:** 184.105.99.79/32, 10.0.1.0/24, tcp(80), 2024/05:weekdays, deny
- S4:** 184.105.99.79/32, 10.0.2.0/24, tcp(80), 2024/05:weekdays, deny

...

<Src, Dst> Gateway Interfaces of Rules:

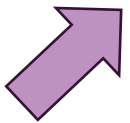
- S1, S3:** src: [R_A@1, R_B@1], dst: [R₁@2, R₂@2],
- S2, S4:** src: [R_A@1, R_B@1], dst: [R₂@4]

...

Result ACL Rules with Gateway Information

Endpoint	Gateway(s)+Interface(s)	Prefix(es)
Laboratory	Router 1, Ethernet0/2 Router 2, Ethernet0/2	10.0.1.0/24
Teach		0/24
Ex		0/24
		0/24
OpenAI	Router A, Ethernet0/1 Router B, Ethernet0/1	104.18.33.170/32 184.105.99.79/32 ...

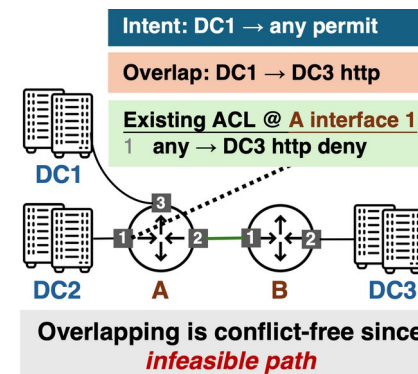
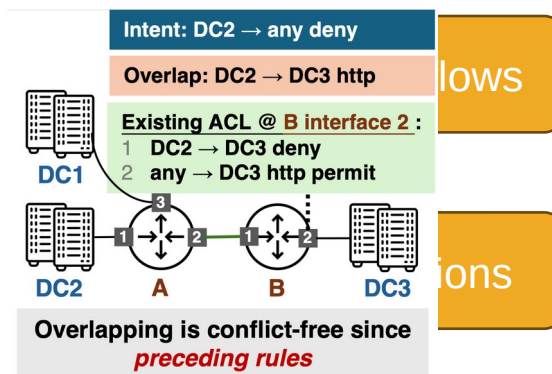
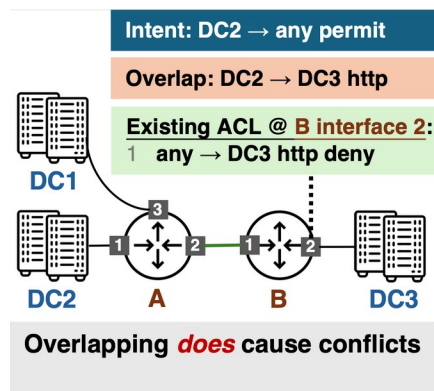
LLM lacks of network-specific information



Semantics-Network Mapping Table (SNMT)

Conflict Detection & Resolution

- Naive condition checking leads to false-positive detection results



TMF-Based Conflict Detection

- Truly Matched Flow (TMF) Calculation
 - with

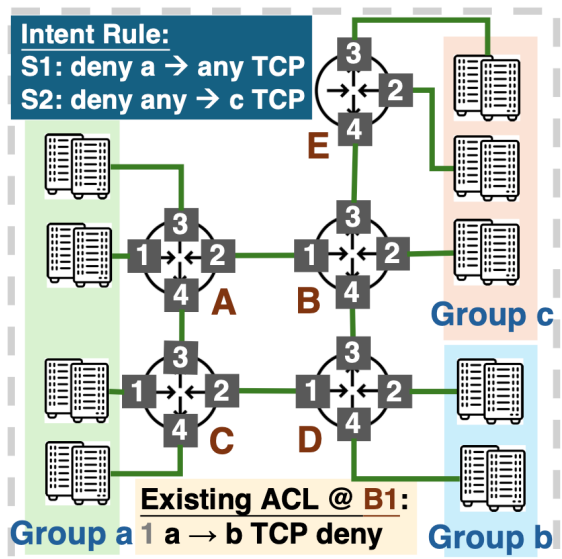
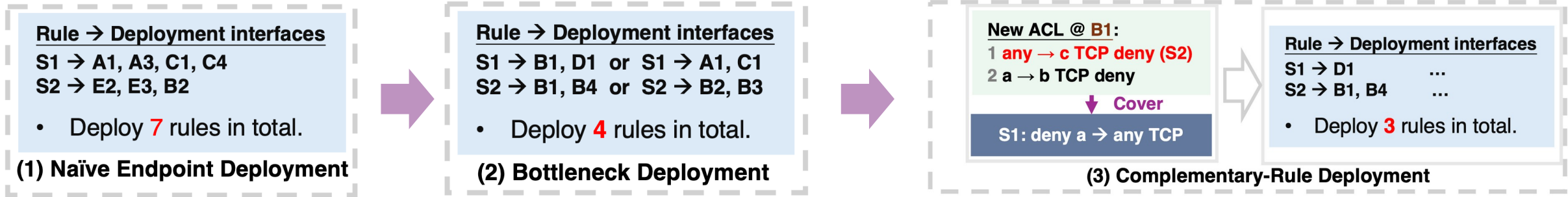
Interface-Path Validation

- Check whether target interface is on any path of the overlapping flow

Resolutions with Operators

- Protect Mechanism*: Invite operators propose subset of conflict flows that preserve existing rules

Deployment Optimization



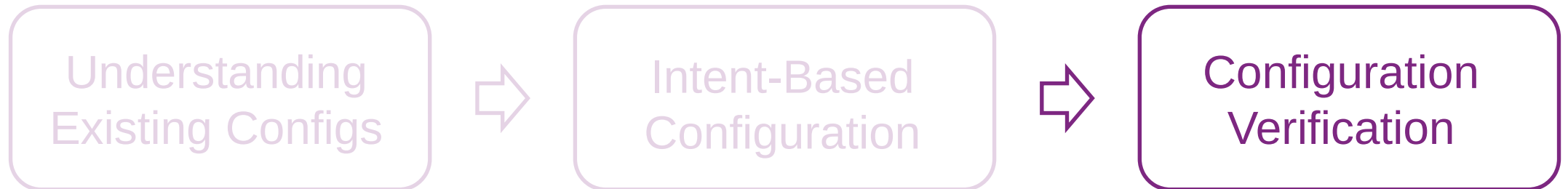
Solution: Joint Optimization Formulation

- Jointly considers two novel observations in one formulation

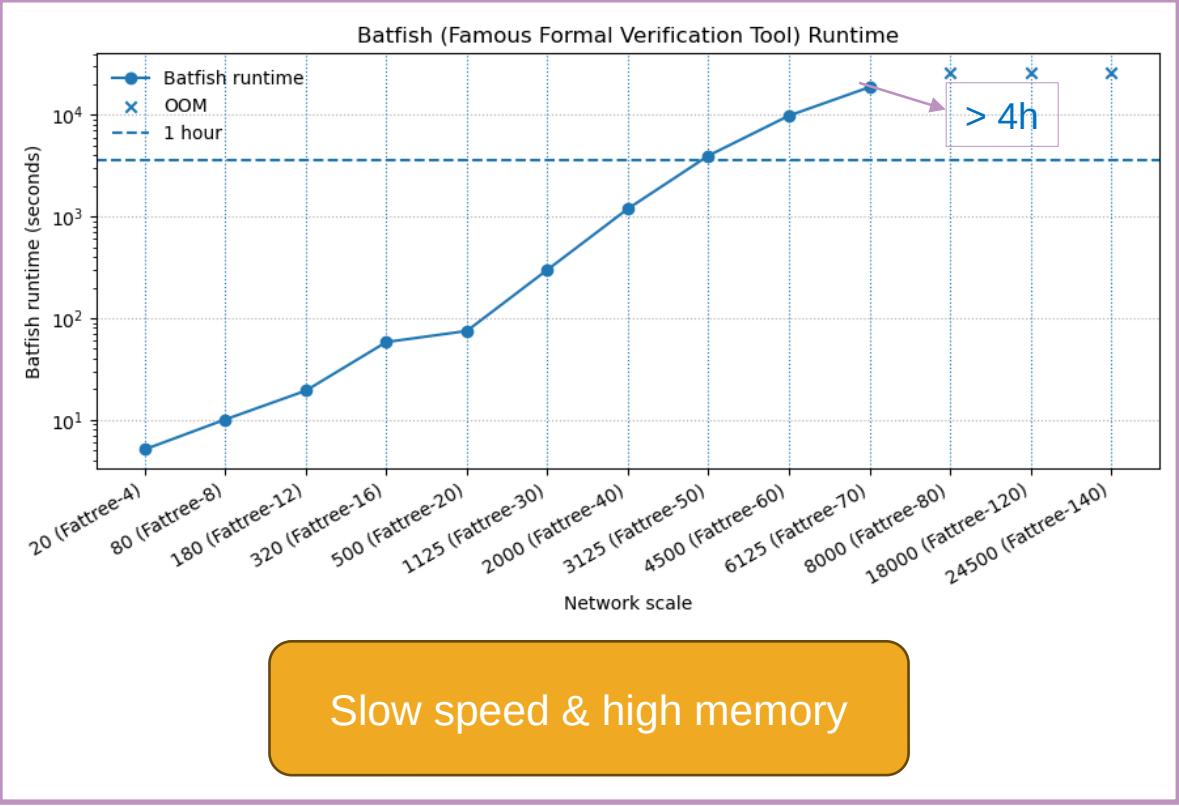
Results Highlight

- **Evaluated (Synthetic) Networks**
 - *CampusNet*: 41 WAN routers, 66 links (Mimic CUHK network)
 - *CloudNet*: 171 WAN routers, 219 links (Mimic production large cloud)
 - *ExtremeNet*: 1026 WAN routers, 1236 links (6× large cloud, Mimic future cloud)
- **Intent Comprehension**
 - Up to 8 minutes with GPT-4o
 - < 3 feedbacks rounds
- **Conflict Detection**
 - 3.3× higher detection accuracy on average
 - < 2 hours for all cases.
- **Deployment Optimization**
 - 38.8% rule reduction on average
 - < 3 minutes for all cases

NAC: Selective Verification at Scale with Network Analysis Copilot



Formal Verification: Scalability Problem



Config changes hourly

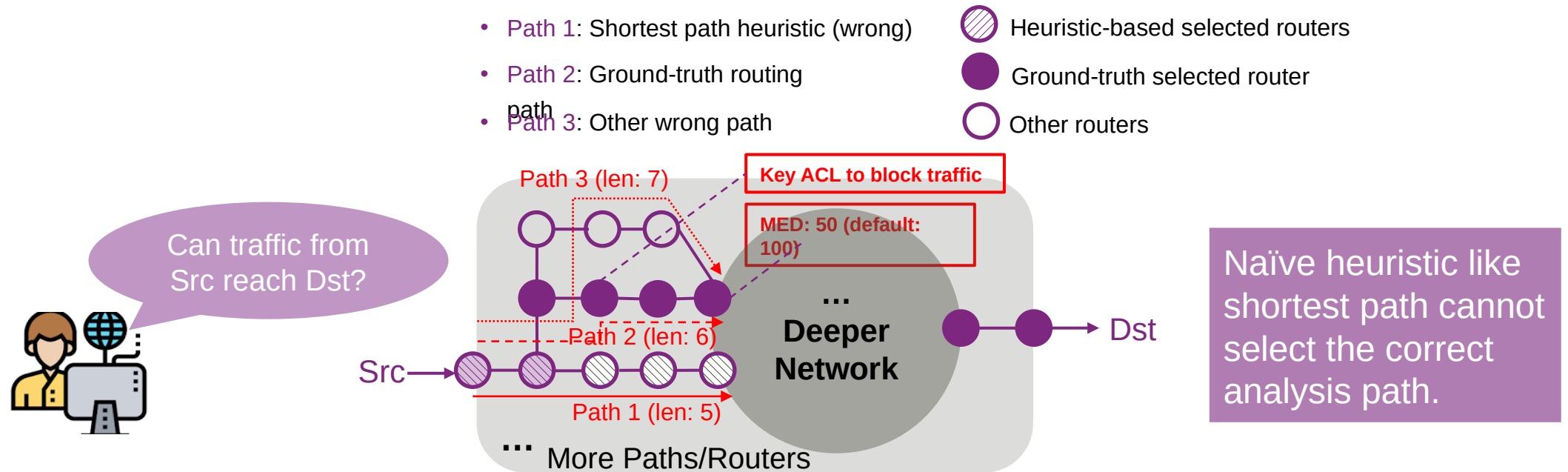
Large cloud has O(100) regions; Each with O(10k) devices

Strict timeliness & large cloud network

Formal verification tool (like SOTA Batfish) cannot scale to meet global (large) network requirement.

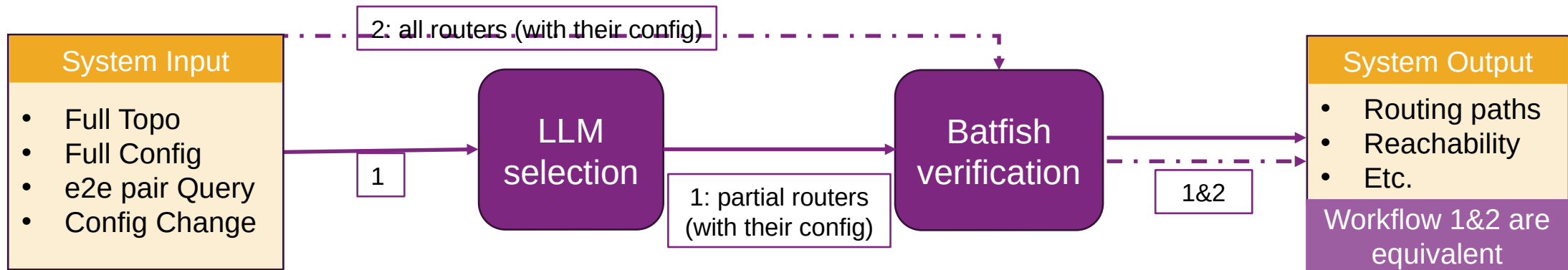
Selective Verification: Error-prone Heuristic Selection

- Selective Verification: Only partial routers with configs for formal verification.



Naïve manual (heuristic) selection is error-prone without knowing holistic configs/routing situations across the entire network.

LLM-Based Semantic Analysis: Overview



Key designs: Use LLM-based semantic analysis to infer routing states before and after configuration changes and identify affected routers across the entire network.

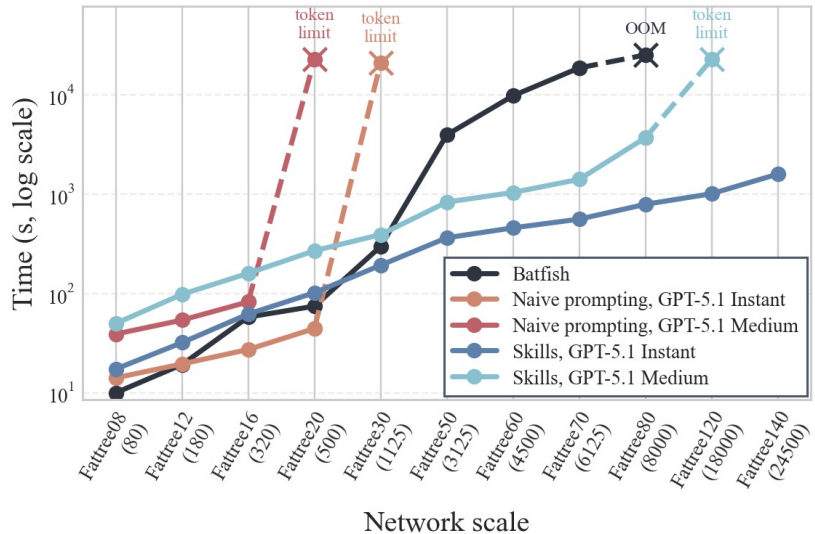
- This approach offers higher confidence than heuristics, as LLMs can rapidly reason across protocols—analysis that is infeasible for humans or formal tools in large networks.

Selective Results

➤ LLM-based evaluation methods

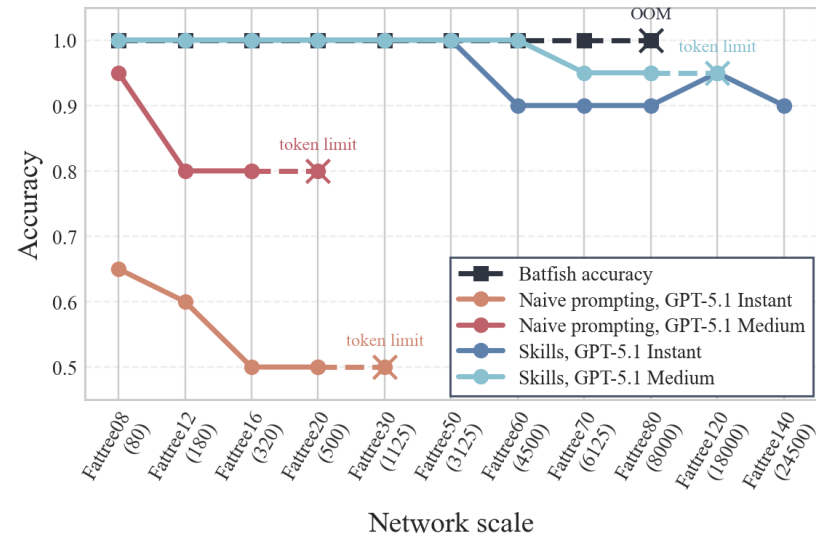
- **Naïve prompting:** Provide all configuration and topology information to the LLM with detailed instructions.
- **Agent with skills:** The LLM mainly acts as a planner, using topology/configuration extraction tools and design docs to complete the task.

• Task completion time



Naïve prompting full config/topo information still suffers from scalability issues due to context window limits.

• Accuracy



Letting the LLM *plan more and analyze less* through agents better unlocks its potential.

Evaluated on an arbitrary MED change (20 runs)

Ongoing Work

Detailed implementations and more supported functions

Further evaluations on real production topology and configs

Seek for opportunity for real deployment

...

