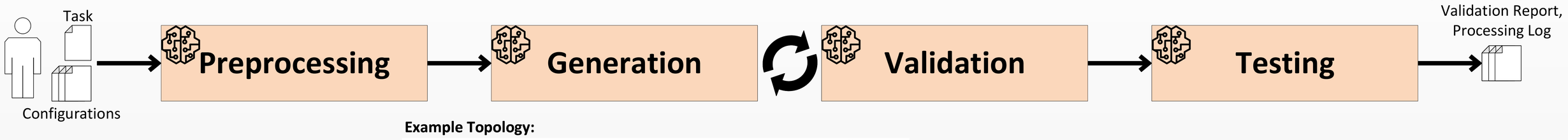
FACULTY OF INFORMATION TECHNOLOGY

Michal Rozsíval, Petr Matoušek, Jaromír Kotala
Brno University of Technology, Božetěchova 2, Czech Republic
irozsival@fit.vut.cz, matousp@fit.vutbr.cz, xkotal01@stud.fit.vut.cz

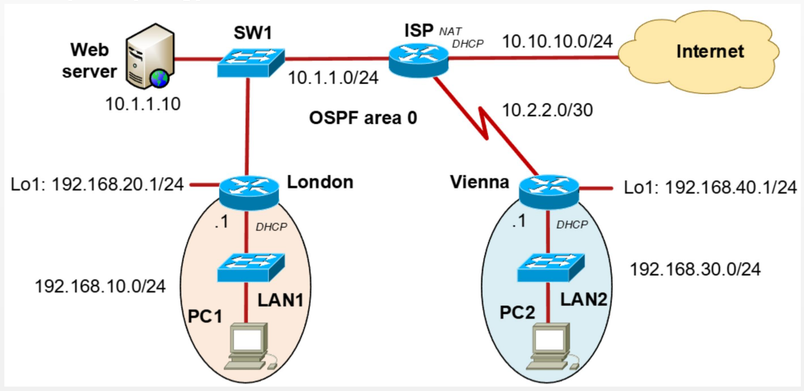
nes fit

Multi-Agent LLM System for Cisco Router Configuration

Accurate network device configuration requires careful updates to ensure correct operational behavior. We propose a multi-agent LLM system that emulates the decision-making process of a human network expert: it decomposes tasks into concrete objectives, derives structured update plans, and applies updates systematically. Correctness is ensured through extensible validation functions, currently based on pyBatfish. The system generates tests derived from the original task, enabling users to confirm that the resulting configurations fulfill all intended objectives.



Example Topology:



Task: Configure packet filtering for LAN at Vienna. LAN 192.168.30.0/24 is allowed to communicate with 192.168.40.0/24 (all traffic). LAN 192.168.30.0/24 can access web server at the server 10.1.1.10. Allow DNS and ICMP communication for all local networks. No other traffic is allowed to originate from LAN 192.168.30.0/24

Configurations: Vienna.cfg, London.cfg, ISP.cfg

Expert Actions

1. Analyses the task and identifies objectives.

2. Determines the sequence of actions to achieve the objectives.

3. Identifies configurations relevant to the objective.

4. Determines the update order of configurations based on dependencies and constraints.

5. Decides how to modify the configuration considering current configuration and context.

6. Updates the configuration file.

7. Verifies that applied changes align with the intended modification.

8. Verifies that updated configurations satisfy the objective requirements.

9. Tests the overall functionality of updated configurations against the original task.

LLM Agent Workflow

Preprocessing

1. Task Decomposition: divides the task into focused subtasks covering required configuration changes.

2. Subtask Expansion: defines a structured sequence of high-level steps, removing implicit assumptions.

Generation

3. Configuration Filtering: selects configurations needing updates or providing essential context.

4. Configuration Ordering: determines the optimal update sequence for consistency.

5. Update Planning: devises a modification strategy using high-level steps and contextual dependencies.

6. Configuration Updating: applies planned Changes by editing configuration to meet subtask objectives.

Validation

7a. Consistency Check: verifies compliance of updated configurations with their plan.

7b. Consistency Coverage: ensures all necessary consistency validations are addressed.

8a. Integrity Check: validates the coherence of all updated configurations.

8b. Integrity Coverage: confirms integrity validations cover all modified aspects.

Testing

9. Test Design: formulates tests to verify updated configurations satisfy the original task.

Example: Output Generated by LLM

Subtask: Add a packet filter rule at the Vienna site permitting all traffic from source 192.168.30.0/24 to destination 192.168.40.0/24.

Generated steps:
1) Create a packet filtering policy that will control traffic originating from 192.168.30.0/24.
2) Add a rule with action permit matching: protocol any, source IP 192.168.30.0/24, destination IP 192.168.40.0/24, source port any, destination port any.
3) Apply this packet filtering policy to the interface that faces network 192.168.30.0/24 for outbound traffic.
4) Position the permit rule before any rule in the same policy that could deny or override traffic from 192.168.30.0/24 to 192.168.40.0/24.

Filtering:
- Vienna.cfg [select]: The device hostname is 'Vienna', which matches the TASK descriptor, and it has interfaces configured with IP addresses 192.168.30.1 and 192.168.40.1.

Generated instructions:
1) Create a packet filtering policy that will control traffic originating from 192.168.30.0/24.
2) Add an ordered rule with action permit matching protocol any, source IP 192.168.30.0/24, destination IP 192.168.40.0/24, source port any, and destination port any.
3) Apply this packet filtering policy to interface Ethernet0/0 for outbound traffic.
4) Position the permit rule before any rule in the same policy that could deny or override traffic from 192.168.30.0/24 to 192.168.40.0/24.

Generated updates:
```
+ip access-list extended PF_LAN30
+ permit ip 192.168.30.0 0.0.0.255 192.168.40.0 0.0.0.255
+ deny ip any any
…
+ ip access-group PF_LAN30 out
```

Generated validation:
- validate_syntax: Ensures the router configuration syntax is correct before applying packet filtering rules, preventing misconfigurations.
- validate_interface_exists: Confirms that the interfaces relevant to LAN 192.168.30.0/24 and the path to the web server 10.1.1.10 exist, which is essential for applying packet filters.

Generated validation coverage (suggested validation aspects)**:**
- Packet filter rule correctness and direction: The task requires denying any other traffic from LAN 192.168.30.0/24; validation must confirm that appropriate ACL or firewall rules exist and are applied in the correct direction (inbound or outbound) on the relevant interface Use 'show access-lists' or 'show firewall rules' to verify that a deny rule for source 192.168.30.0/24 exists and is correctly applied to the interface connected to that LAN. Confirm rule direction (inbound) matches intended traffic flow.
- Packet filter application to interface: Even if ACLs are defined, they must be applied to the correct interface and direction to take effect. Run 'show run interface <interface>' or equivalent to confirm the ACL or filter is bound to the LAN-facing interface in the correct direction (inbound).

Generated tests:
- Allow all traffic from 192.168.30.0/24 to 192.168.40.0/24: LAN30 must be able to communicate with LAN40 using any IP protocol. *Approach:* 1) On the router, clear ACL counters: clear ip access-list counters PF_LAN30. 2) From a host in 192.168.30.0/24 (GW 192.168.30.1), ping 192.168.40.1. Expect replies. 3) From the same host, open a TCP session to 192.168.40.1 on a non-web port (e.g., telnet 192.168.40.1 22 or nc -vz 192.168.40.1 22). Expect either a banner or immediate refusal, not a timeout. 4) On the router, show ip access-lists PF_LAN30 and verify the counter for the permit ip 192.168.30.0/24 to 192.168.40.0/24 entry incremented.

Acknowledgments: This work is supported by the project “Novel AI-Driven Process Automation for Simplifying and Enhancing Telecommunication Processes (Spojit)”, 2024-2026, no. FW10010014, funded by the Technology Agency of the Czech Republic.