AnSerApp workshop co-located with the
21st International Conference on Network and Service Management (CNSM 2025)
27-31 October 2025 – Bologna, Italy

BIG DATA & COMPUTE CONTINUUM RESEARCH LAB

Investigating Neurosymbolic AI for Intent-based Service Management

Lorenzo Colombi*, Sara Cavicchi*,

Filippo Poltronieri*, Mauro Tortonesi*, Cesare Stefanelli* and Pal Varga†

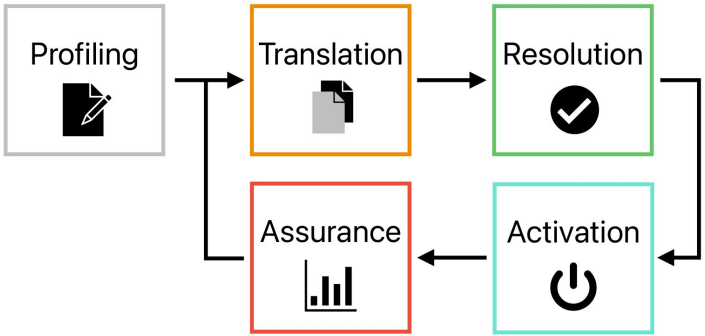
*** University of Ferrara**, Ferrara, Italy
† **Budapest University of Technology and Economics**, Budapest, Hungary

Università degli Studi di Ferrara

Background: ZSM and Intent-based Service Management

- **Zero-touch network and Service Management (ZSM)** is a paradigm aimed at fully automating network and service lifecycle operations. ZSM promotes **vendor-neutral**, **closed-loop automation** systems capable of performing tasks such as self-configuration, self-healing, and self-optimization.

- **Intent-based Service Management** is a key enabler of ZSM. It allows **high-level expressions** of **management objectives** instead of low-level configurations. The goal is **simplify** the **management** for the operators and better aligns configurations with business objectives.

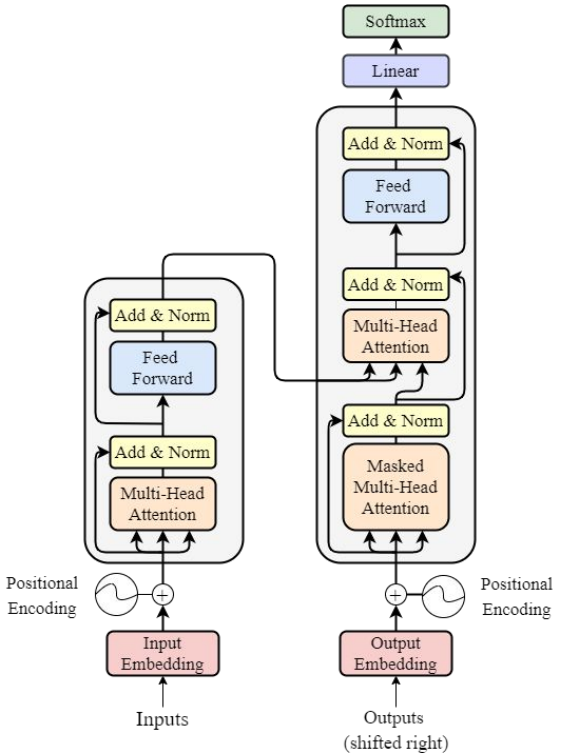


Five stages of intent processing

Background: LLMs for Intent Processing

- Intent-based Service Management often relies on **rigid approaches** to define intent statements, **limiting expressiveness** and restricting the range of possible configurations.

- **LLMs** can **interpret** and **refine** complex **natural** language **inputs** into consistent, structured machine-readable policies.



Microsoft Phi-4

Qwen

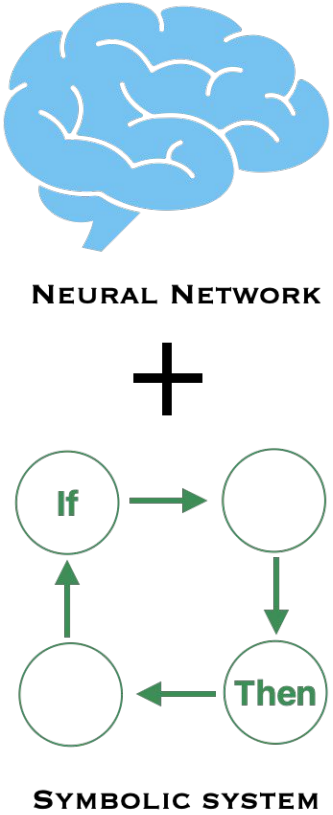
MISTRAL AI_

Gemma

Claude

Background: Neurosymbolic AI (NeSy)

- **Neural models** excel at learning patterns from data and interpreting unstructured inputs, but **are non deterministic** and act as opaque "**black boxes**".
- **Symbolic AI** relies on **explicit rules and logic**, making its decisions **explainable** and **deterministic**, though less flexible.
- **NeSy** enables intents to be expressed **flexibly** in natural language, while ensuring that the resulting **policies** remain **explainable**, and aligned with resource constraints and service dependencies.



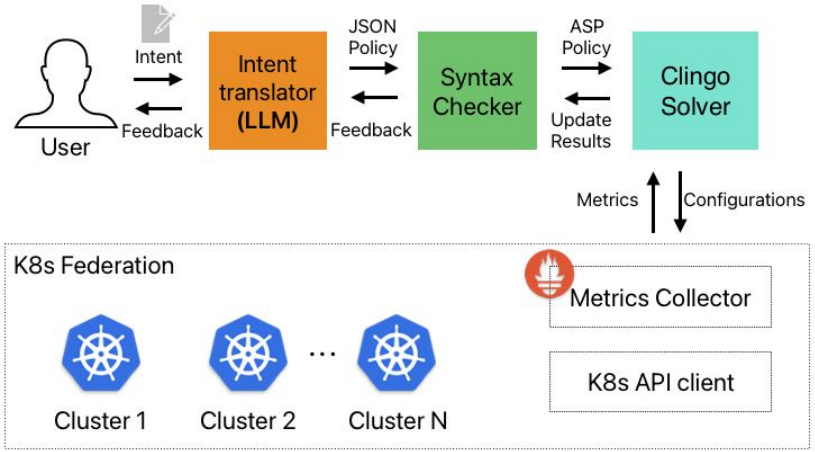
NeSy for intent-based service placement

1. We used an **LLM** to **interpret** user **requests** made in natural language. It assesses the instructions for clarity and, if they are ambiguous, engages the user in a dialogue to obtain the necessary refinements.

2. The **LLM** then **converts** the request **in a machine-readable** format (JSON).

3. **Syntax Checker** module analyzes the generated JSON to ensure that its **structure** is **syntactically correct** and it is compliant to the **required schema**.

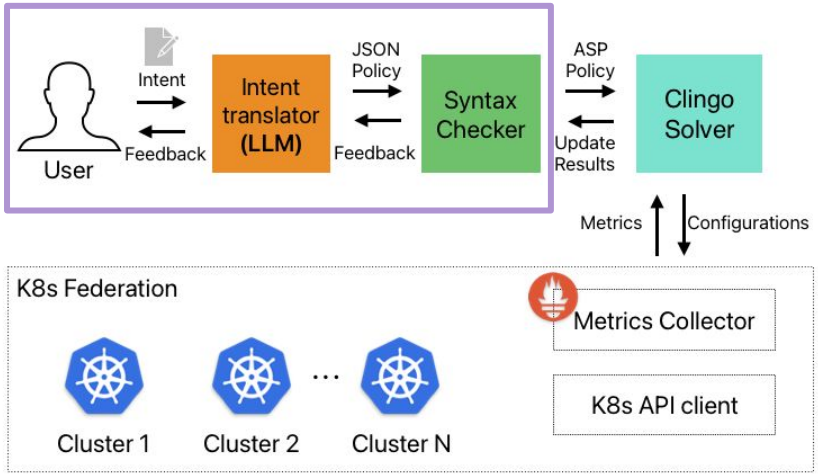
4. The **JSON** file is **converted** in a set of executable **low-level policies**. This translation is done by mapping the structured representation into a corresponding set of **ASP rules.***



**Answer Set Programming (ASP) is a form of declarative programming designed to solve difficult search and optimization problems.*

JSON intermediate representation

- We used **JSON** as intermediate representation since it is both **machine** and **human readable.**
- This permit to **decouple** the translation step from the subsequent ones.
- In our initial experiment, we defined for each service a set of **constraints** and a list of **dependencies**.



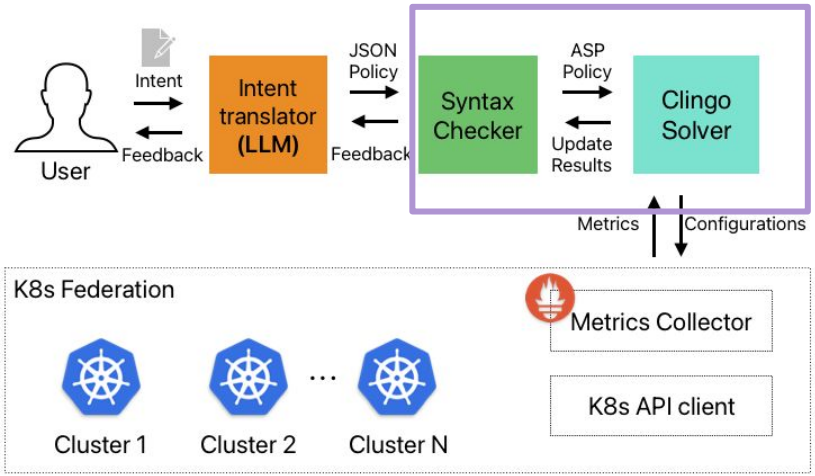

```
{"serviceId": 5, "constraints": {"cpu": 3,
"ram": 32, "latency": null, "cost": null,
"gpu": 1}, "goal": "-mincpu", "depends_on":
[3, 4], "same_cluster": [1] }
```

Answer Set Programming (ASP) formulation

- **ASP** is a **rule-based language** for knowledge representation and to solve optimization problem, developed within the field of logic programming.

- **ASP solvers** (such as **Clingo**) **compute** all the **stable models**, which represent the set of literals that are true according to the program's rules. In our case, the stable models are **all the possible service-cluster** matching **solutions**.

- ASP is **deterministic** and inherently **explainable**, providing transparency and reliability in the decision-making process.



ASP rule that compute the total CPU and RAM usage for a single cluster:

```
resources(C_id, T_cpu, T_ram) :-
      cluster(C_id,C_cpu, C_ram),
      T_cpu = #sum{S_cpu, C_id:match(S_id, C_id),
            service(S_id, S_cpu, _)},
      T_ram = #sum{S_ram, Cluster_id: match(
            S_id, C_id), service(S_id, _, S_ram)}.
```

Experimental evaluation

- We conducted **two experiments** to validate the proposed approach, separately testing the two main components of the neurosymbolic architecture: the **LLM** and the **ASP** solver.
- We compared the capabilities of different open-source **LLMs** in **generating** syntactically corrected **JSON** files.
- We measured the **times** to **compute** all the **stable models** varying the ASP problem complexity.

TABLE I: Summary of Experimental Setups

Specification	LLM Intent Translation	ASP Experiement
Operating System	Linux (CentOS)	Linux (Ubuntu)
RAM	32GB DDR4	32GB DDR4
CPU	AMD EPYC 9224	Intel Core i7-9750H
GPU	Nvidia L40S	Nvidia RTX 3060

Experimental evaluation: LLM Intent Translation

- Using **llama.cpp**, we **prompted** the **LLMs** with **natural language** inputs alongside a predefined JSON **template** and a series of illustrative **examples**.
- An **output** was deemed **valid** if it could be successfully deserialized into a **JSON** object, contained all the **required keys** and assigned **appropriate** data **types** and **legal values** to each field.
- We tried a **few-shot** approach prompting the model when the **JSON** was **not valid appending** the **error message** into the prompt.

Model	1-s Acc.	2-s Acc.	3-s Acc.	Mean #Prompts	Inference Time (s)
DeepSeek-R1-Distill-Qwen-14B-Q6_K	0.48	0.83	0.83	1.42	16.74
google_gemma-3-27b-it-Q6_K	0.06	1.00	1.00	1.94	2.99
google_gemma-3-12b-it-Q6_K	0.00	0.67	0.67	2.00	1.50
Meta-Llama-3.1-8B-Instruct-Q6_K_L	0.16	0.66	0.67	1.77	0.95
qwen3-8b-q6_k	0.03	0.56	0.66	2.12	63.84
Qwen3-32B-Q6_K	0.00	0.00	0.72	2.21	87.50
qwen2.5-coder-7b-instruct-q6_k	0.39	0.83	0.83	1.53	0.97

TABLE II: Model accuracy at different shot levels, average prompts to get valid JSON (**Mean #Prompts**), and **Inference Time**.

Experimental evaluation: ASP execution time

- We **measured** the **execution time** of **Clingo**, to **compute** all possible **stable models**, or the service-to-cluster matching solutions that satisfy the specified constraints.
- We **randomly generated** a series of **clusters** and **services** with their respective **constraints** using Python and the Clingo Python module.
- Although **computational time increases** with the **number of clusters** due to more potential service-to-cluster combinations, it **remains** within an **acceptable** threshold, and the parallel version is significantly faster.

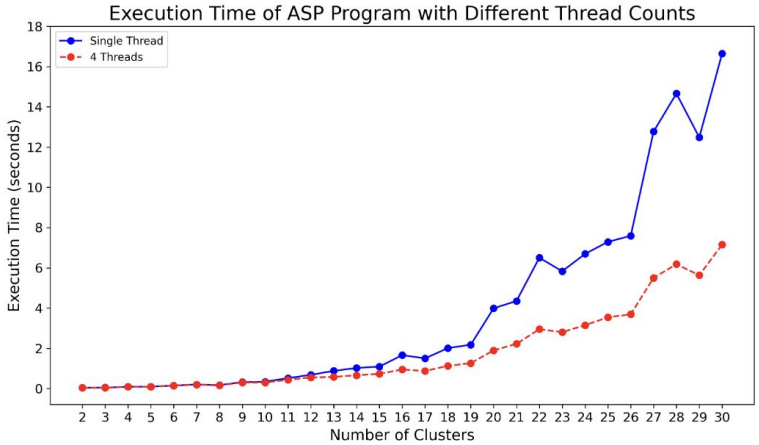


Fig. 3: Service-to-Cluster ASP-based filtering execution time with one and four threads.

Conclusion and future work

- We demonstrated the **feasibility** of combining open-source **LLMs** with **ASP**. In this system, LLMs effectively translate user intents into a structured format, and ASP computes all possible solutions for matching services to clusters.

- The experiments show that **even** smaller, **lightweight LLMs** provide a **good** balance between **performance** and computational efficiency. This makes them a practical choice for real-world applications without requiring massive computing resources.

- Future work involve **expanding** the system's **symbolic reasoning** to manage more **complex deployment** rules, such as **multi-objective optimization** and temporal constraints. We also plan to **integrate advanced decision-making** techniques like **reinforcement learning** and improve support for real-time orchestration in resource-limited environments.

Thank you for your attention!

Lorenzo Colombi

PhD student @ **University of Ferrara**

https://ds.unife.it/team/lorenzo-colombi

lorenzo.colombi@unife.it