Edited ver.

26 MAY 2025

Metamorphic Testing & Basic Description Logic

A FOUNDATION FOR METAPHORIC TESTING RESEARCH

PHANPHUM PRATHUMSUWAN | MAHIDOL UNIVERSITY

Table of Contents

Research Motivation

Large Language Models (LLMs), such as GPT-4o, are powerful but struggle with subtle logical consistency.

Description Logic (DL) is a foundation for knowledge systems like OWL ontologies.

Reasoners are essential for DL inference, but testing their correctness is challenging.

Metamorphic testing offers a way to check consistency under logical transformations.

Does the reasoner preserve expected class relationships and instance memberships
under valid ontology transformations?

What is Description Logic?

- Description Logic (DL) is **a formal logic-based language** used for describe **what things are** and **how they relate to one another**, in a way that allow computers to reason about them.
- It belongs to a **family of knowledge representation (KR) formalisms** that model the structure of an application domain by first defining the key **concepts** and **relationships** to form the domain's **terminology**.

Concepts: Classes	Roles: Binary relations	Individuals: Specific instances
Pizza	hasTopping	Hawaiian
Student	hasHomework	Nonraphan

Core uses: knowledge modeling, logical inference, and consistency checking

Structure of a Description Logic Knowledge Base

- A **knowledge base (KB)** consists of two components:
  - **TBox** (Terminology Box): defines the **concept** of the domain
  - **ABox** (Assertions Box): contains **facts about individuals** using the defined vocabulary
- **The vocaburaly consists of:**
  - **Concepts:** denotes sets of individuals (e.g., Student, Teacher, Course)
  - **Roles:** denotes binaray relationships (e.g., enrolledIn, teaches, hasHomework)
- **Example:**
  - **(TBox)**
    - $\text{ComputerScienceStudent} \equiv \text{Student} \sqcap \exists \text{enrolledIn.ComputerScienceCourse}$
    - $\text{ComputerScienceCourse} \sqsubseteq \text{Course}$
  - **(ABox)**
    - enrolledIn(Nonraphan, SpecialTopicforComputerScience101)
    - $\text{SpecialTopicforComputerScience101} \sqsubseteq \text{ComputerScienceCourse}$
  - **Therefore:** ComSciStudent(Nonraphan)

- **Example – Pizza Ontology**
  - **Concept:** MargheritaPizza
  - **Roles:** hasTopping(Tomato), hasTopping(Mozzarella)
  - DL Reasoning infers:
    - Margherita $\in$ MargheritaPizza
    - Margherita $\in$ VegetarianPizza
    - Margherita $\in$ Pizza

Reasoning in DL

- **Subsumption: Is concept A a subset of concept B? ($A \subseteq B$)**
- **Example:**
  - **Axiom:** Tomato $\subseteq$ RedVegetable
  - **Inference:** $\exists$hasTopping.Tomato $\subseteq$ $\exists$hasTopping.RedVegetable

- **Satisfiability: Is concept A logically consistent?**
- **Example:**
  - **Concept:** VegetarianPizza $\sqcap$ $\exists$hasTopping.Salami
    - If Salami $\subseteq$ MeatTopping and VegetarianPizza $\sqcap$ $\exists$hasTopping.MeatTopping $\sqsubseteq \bot$ (meaning VegetarianPizza cannot have meat toppings)
    - Then the concept is unsatisfiable, leading to a contradiction and cannot have any real instances.
  - **A satisfiable example:** Pizza $\sqcap$ $\exists$hasTopping.Tomato

Reasoning in DL

- **Equivalence: Are concepts A and B logically equivalent? (A ≡ B)**
- **Example:**
  - Two defined **concepts**:
    - VeganPizza ≡ Pizza ⊓ ∀hasTopping.VeganTopping
    - PlantBasedPizza ≡ Pizza ⊓ ∀hasTopping.VeganTopping
  - Since both are defined identically, therefore:
    - VeganPizza ≡ PlantBasedPizza

- **Instance checking: Does individual a belong to concept A? (a ∈ A)**
- **Example:**
  - **Individuals:**
    - margherita ∈ Pizza
    - hasTopping(margherita, Tomato)
  - **Concept:** ∃hasTopping.Tomato
    - Since margherita has a topping that is a Tomato, it can be infered that: margherita ∈ ∃hasTopping.Tomato

Metamorphic Testing

- A software testing method used when the correct output is unknown (oracle problem)
- Instead of checking correctness → check consistency under transformation
- Example (English to Spanish Translation):
  - Input: "Translate 'cat' to Spanish"
    → Output: 'gato'
  - Follow-up Input: "Translate 'cats' to Spanish"
    → Expected Output: 'gatos'

MMT4NL Framework (Racharak et al., 2024)

- MMT4NL uses metamorphic testing, which relies on relations between outputs of a program after specific input transformations, rather than needing a direct "test oracle" for every output.
- Introduces 9 perturbation types:
  - **Taxonomy:** Replaces a word in the input with its synonym; the output should remain consistent.
  - **NER (Named Entity Recognition):** Replaces pronouns with fictitious proper nouns; the output related to these nouns should remain unchanged.
  - **Negation Handling:** Transforms the input by adding negation cues, expecting the original sentiment or meaning to be appropriately adjusted or reversed.
  - **Vocab:** Introduces new or unknown words into the input to test if the LLM handles them gracefully or maintains the original intent.

MMT4NL Framework (Racharak et al., 2024)

  - **Fairness:** Changes demographic attributes (e.g., gender, race) in the input; the output should remain the same if these changes don't affect the task.
  - **Robustness:** Introduces minor errors like spelling mistakes or typos into the input; the inference is expected to remain unchanged if the error is minor.
  - **Temporal:** Modifies time-based information in the input; the model's output should remain consistent.
  - **SRL (Semantic Role Labeling):** Rephrases the input while preserving its meaning to ensure predicates and semantic roles are consistently interpreted.
  - **Coreference:** Restructures questions to include explicit pronoun references, creating referential distance to test if the model maintains logical connections
- The paper evaluates LLMs (specifically GPT-4o and Gemini-2.0-Flash) on sentiment analysis and question-answering tasks using these perturbations.

Applying Metamorphic Testing to Description Logic Reasoning

Idea:

- Verifying complex DL reasoners is tough due to the "test oracle problem".
- Proposed Approach: Use Metamorphic Testing.
  - Transform input ontologies in logically sound ways.
  - Check if reasoner outputs (inferences) maintain expected logical consistency or change predictably, focusing on relationships between different outputs.

Goal:

- Develop an metamorphic testing framework to systematically test DL reasoner consistency and robustness.
- Identify bugs or deviations in reasoners and benchmark their performance under logical transformations.
- Ultimately, enhance the trustworthiness and reliability of DL reasoning systems.

Sample Test Cases

Original Expression/Assertion	Metamorphic Variant	Expected Result
∃hasTopping.Tomato	∃hasTopping.RedVegetable	Equivalent
∃hasChild.Male	∃hasChild.¬Female	Equivalent
∃hasIngredient.Cheese	∃hasIngredient.DairyProduct	Similar
VegetarianPizza(Margherita)	∃hasTopping.Tomato(Margherita) and ∃hasTopping.Mozzarella(Margherita)	Inferred as VegetarianPizza

Current Implementation Plan

Step 1: Understand the original ontology (Establish Ground Truth)

Goal: Figure out what the ontology logically says before making any changes

1. **Choose expressions to test**
    - Pick class definitions and facts (e.g., "Tomato is a RedVegetable", "Pizza hasTopping Tomato").
2. **Use a reasoner to infer knowledge**
    - Let a DL reasoner generate logical conclusions:
        - What classes are subclasses of others?
        - What individuals belong to which classes?
3. **Record the ground truth**
    - Save these reasoning results (baseline knowledge/ ground truth before transformation)

Current Implementation Plan

Step 2: Apply changes and see what happens

Goal: Change the input slightly, then check if the reasoner still behaves correctly.

1. **Make a controlled change (Perturbation)**
   - Modify the ontology slightly.
   - Example:
     - Replace “Tomato” with “RedVegetable”
     - Add a typo: “Cheese” → “Chese”
     - Reword a sentence without changing its meaning
2. **Re-run the reasoner on the changed ontology**
   - Ask the same reasoner to infer knowledge again, then record the new results.
3. **Compare before and after**
   - If the change shouldn’t affect the meaning, the inferences should stay the same.
   - If the change does affect the meaning, the change in inference should make sense.

Real-World Ontologies

Domain	Ontology Examples	Purpose
Healthcare	SNOMED CT, FMA, UMLS	Standardized medical concepts and anatomy
Education	LOM, EDM, OntoEdu	Representing courses, learners, and competencies
Social Web	FOAF (Friend of a Friend), SIOC	Modeling people, profiles, and connections
E-commerce	GoodRelations, schema.org	Product data, pricing, business terms
Biology	Gene Ontology, BioPAX	Genes, proteins, biological pathways

Conclusion

- **Description Logic (DL)** is essential for reasoning in knowledge-based systems.
- **Metamorphic Testing** provides a scalable and systematic method for evaluating the logical consistency and robustness of DL reasoners.
- **Ontologies** serve as structured, logic-rich test inputs that can be transformed to examine reasoner behavior.

Next Steps:

- Expand test coverage using diverse, ontology-aware metamorphic transformations.
- Quantify logical consistency with pass/fail metrics across perturbed inference results.
- Explain and visualize reasoning outcomes from Sample Test Cases table, and possibly some other examples, step-by-step using DL semantics.

References

- Baader, F., & Nutt, W. (2003). Basic Description Logics. In: Baader et al. (Eds.), The Description Logic Handbook. Cambridge University Press.
- Racharak, T., et al. (2024). Test It Before You Trust It: Applying Software Testing for Trustworthy In-Context Learning. arXiv:2402.14771.
- MMT4NL Project. (2024). GitHub Repository
- Rector, A., et al. (2004). OWL Pizzas: Common Errors and Common Patterns from Practical Experience of Teaching OWL-DL. European Knowledge Acquisition Workshop (EKAW 2004). Retrieved from OWL Pizza Ontology. Link (Manchester OWL Repository)