

Logical Inference Using Backward Chaining (First-Order Logic)

Overview

This tutorial describes the design and implementation of a backward chaining inference engine for First-Order Logic (FOL). Backward chaining is a goal-driven reasoning approach that starts with a hypothesis and works backward to determine what facts and rules must be satisfied for the hypothesis to hold true.

The system supports variable unification, recursion, and the construction of AND/OR goal trees to represent multiple logical paths that may justify a conclusion.

Core Algorithm

The central function of the system is:

```
backchain_to_goal_tree(rules, hypothesis)
```

This function:

- Accepts a hypothesis expressed as a logical statement
- Searches IF-THEN rules whose consequents match the hypothesis
- Applies unification to bind variables
- Recursively expands subgoals
- Builds an AND/OR tree representing all possible proofs

Leaf nodes represent atomic facts, while internal nodes represent logical AND or OR relationships.

Backward Chaining Concept

Backward chaining follows a top-down reasoning process:

1. Start with a hypothesis (e.g., "opus is a penguin")
2. Identify rules that could produce the hypothesis
3. Replace the hypothesis with the rule antecedents
4. Repeat until only basic facts remain

Unlike forward chaining, which derives conclusions from known facts, backward chaining focuses on answering specific queries by working backward from the goal.

Testing with Concrete Hypotheses

The system was tested using a grounded hypothesis:

```
"opus is a penguin"
```

This confirms correct rule expansion and goal tree construction when constants are used.

Testing with Variable-Based Hypotheses

To validate First-Order Logic support, the following query was tested:

"(?x) is a penguin"

This confirms that the system:

- Correctly unifies variables
- Handles symbolic reasoning
- Produces a general goal tree applicable to any entity

Example Goal Tree

A simplified goal tree demonstrates that proving an entity is a penguin requires satisfying multiple conditions, including proving that the entity is a bird and that it meets all defining penguin properties.

Command-Line Usage

Run the system:

```
python3 run_task7.py
```

Interactive testing:

```
python3
from data import zookeeper_rules
from lab1 import backchain_to_goal_tree
print(backchain_to_goal_tree(zookeeper_rules, "opus is a penguin"))
```

Variable-based testing:

```
python3
from data import zookeeper_rules
from lab1 import backchain_to_goal_tree
from run_task7 import pretty
print(pretty(backchain_to_goal_tree(zookeeper_rules, "(?x) is a penguin")))
```

Conclusion

This project presents a backward chaining inference engine for First-Order Logic. It supports variable unification, recursive goal expansion, and AND/OR goal tree construction, allowing the system to generate clear and meaningful reasoning structures.