



# Expert Systems & Inference Methods

# Introduction

---

## □ What is Unification?

The process of making two logical expressions identical by finding a **substitution**.

•**Key Idea:**  $\text{Unify}(\text{Expression1}, \text{Expression2})$  returns a substitution  $\theta$  that makes them match.

If you have  $x + 2 = 5$ , the "substitution" that makes both sides equal is  $x = 3$ .

## Conditions for Success:

□ Predicate symbols must be the same.

# Introduction

---

## ✓ Example 1: CAN be Unified (Same Predicate Symbol)

- Expression 1: Loves(John, Mary)
- Expression 2: Loves(x, y)
- Predicate Symbol: Loves in both expressions.

## ✗ Example 2: CANNOT be Unified (Different Predicate Symbols)

- Expression 1: Loves(John, Mary)
- Expression 2: Hates(John, Mary)
- Predicate Symbols: Loves vs. Hates

# Introduction

---

- Number of arguments must be identical.

## ✓ Example 1: CAN be Unified (Same Number of Arguments)

- Expression 1: `Loves(John, Mary)` // 2 arguments: John, Mary
- Expression 2: `Loves(x, y)` // 2 arguments: x, y
- **Number of Arguments: 2 in both** expressions.

## ✗ Example 2: CANNOT be Unified (Different Number of Arguments)

- Expression 1: `Loves(John, Mary)` // 2 arguments
- Expression 2: `Loves(John, Mary, Cake)` // 3 arguments
- **Number of Arguments: 2 vs. 3**

# Introduction

---

- No conflicting variable assignments.

## ✓ Example 1: NO Conflict (Consistent Assignment)

- Expression 1: `Loves(x, Mary)`
- Expression 2: `Loves(John, y)`
- Substitution  $\theta$ : `{x/John, y/Mary}`

Variable  $x$  gets one value (John). Variable  $y$  gets one value (Mary). No variable is asked to be two different things.

**Unification SUCCEEDS.**

# Introduction

---

**A conflict** occurs when the process tries to assign two different values to the same variable.

## ✗ Example 2: CONFLICT (Variable vs. Different Constant)

- Expression 1: `Loves(x, x)` // This means "x loves themselves"
- Expression 2: `Loves(Ali, Adel)` // This means "Ali loves Adel"
- **Potential Substitution:** To match the first argument, we'd need `{x/Ali}`. To match the second argument, we'd need `{x/Adel}`.

x cannot be both Ali and Adel simultaneously.

**Unification FAILS.**



# The Inference Engine: Match-Fire Cycle

# Core of a Rule-Based Expert System

---

The core procedure of an inference engine where it repeatedly matches rules against the facts in a knowledge base, selects and resolves conflicts between matching rules, and then executes the selected rules..

## The Match-Fire Cycle

**1- Match:** The engine compares the "if" parts of its rules against the current facts in the knowledge base.

## Match Phase Example

A simple expert system for animal identification.

# Core of a Rule-Based Expert System

---

## Knowledge Base (Rules)









- Rule 1: IF it has fur AND says "meow" THEN it is a cat.
- Rule 2: IF it has feathers AND can fly THEN it is a bird.
- Rule 3: IF it is a cat THEN it is a mammal.

## Initial Facts in Working Memory

- Fact 1: HasFur
- Fact 2: SaysMeow

# Core of a Rule-Based Expert System

- ❑ The inference engine now compares each rule's **IF** part (the condition) against the facts.

Rule	IF (Condition)	Check Against Facts	Result of Match
Rule 1	HasFur AND SaysMeow	 Fact 1: HasFur is present.  Fact 2: SaysMeow is present.	 <b>CONDITION SATISFIED</b> The rule is <b>matched</b> and added to the "conflict set" (list of ready-to-fire rules).
Rule 2	HasFeathers AND CanFly	 Fact: HasFeathers is <b>not</b> present.  Fact: CanFly is <b>not</b> present.	 <b>CONDITION NOT SATISFIED</b> The rule is <b>not matched</b> . It is ignored in this cycle.
Rule 3	IsCat	 Fact: IsCat is <b>not</b> present.	 <b>CONDITION NOT SATISFIED</b> The rule is <b>not matched</b> . It is ignored in this cycle.

# Core of a Rule-Based Expert System

---

## The Match-Fire Cycle

**2- Fire (or Act/Resolve):** From the set of matched rules, the engine chooses one or more to "fire".

If multiple rules match, a conflict resolution strategy is used to select which rule(s) to execute.

### □ The Fire Process Step-by-Step

#### Step 1: Conflict Resolution

The engine sees that only **Rule 1** is in the conflict set.

# Core of a Rule-Based Expert System

---

## Step 2: Execution (The "Fire")

The engine **executes the THEN part** of Rule 1.

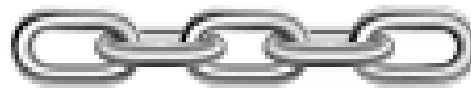
**Action:** Add "IsCat" to the working memory.

## Step 3: Update Working Memory

### Updated Working Memory:

- Fact 1: HasFur
- Fact 2: SaysMeow
- Fact 3: IsCat ← *(The result of firing Rule 1)*

# Introduction to Chaining Methods



# Introduction to Chaining Methods

---

## Definition

• **Chaining** is the step-by-step process an expert system uses to apply rules to facts in order to reach a conclusion.

□ **Two main strategies for the inference engine:**

### **A- Forward Chaining:**

**Direction:** Data-Driven (Facts  $\rightarrow$  Goal)

**Think:** "What conclusions can I draw from what I already know?"

# Introduction to Chaining Methods

---

## Forward Chaining: The "Getting Ready for School" Example

### Rules (The "IF-THEN" instructions in your brain):

1. IF alarm is ringing THEN it's time to wake up
2. IF it's time to wake up THEN you should get out of bed
3. IF you are out of bed AND you feel hungry THEN you should eat breakfast
4. IF you ate breakfast THEN you should brush your teeth

# Introduction to Chaining Methods

---


## The Forward Chaining Process:

### Step 1: Initial Facts (What you observe when you wake up)

- Fact 1: Alarm is ringing
- Fact 2: Feel hungry

### Step 2: Apply Rules (Your brain thinking step-by-step)

#### Match & Fire Rule 1:

- IF alarm is ringing  (Fact 1 is true)
- THEN add new fact: It's time to wake up
- New Fact 3: It's time to wake up

# Introduction to Chaining Methods

---


## The Forward Chaining Process:

### Step 1: Initial Facts (What you observe when you wake up)

- Fact 1: Alarm is ringing
- Fact 2: Feel hungry

### Step 2: Apply Rules (Your brain thinking step-by-step)


#### Match & Fire Rule 1:

- IF alarm is ringing  (Fact 1 is true)
- THEN add new fact: It's time to wake up
- New Fact 3: It's time to wake up



# Introduction to Chaining Methods

---

## Match & Fire Rule 2:

- IF `it's time to wake up`  (Fact 3 is true)
- THEN add new fact: `You should get out of bed`
- New Fact 4: `You should get out of bed`


## Match & Fire Rule 3:

- IF `you are out of bed`  (Fact 4 is true) AND `you feel hungry`  (Fact 2 is true)
- THEN add new fact: `You should eat breakfast`
- New Fact 5: `You should eat breakfast`

# Introduction to Chaining Methods

---

## Match & Fire Rule 4:

- IF `you ate breakfast`  (Fact 5 is true)
- THEN add new fact: `You should brush your teeth`
- New Fact 6: `You should brush your teeth`

## Final Result:

Starting from: `Alarm is ringing` + `Feel hungry`

Ending with: `You should brush your teeth`

# Introduction to Chaining Methods

---

## **B- Backward Chaining:**

**Direction:** Goal-Driven (Goal  $\rightarrow$  Facts)

**Think:** "What facts do I need to prove this hypothesis?"

# Introduction to Chaining Methods

---

## Backward Chaining: The "Why Am I Late for School?" Example

Rules (The same "IF-THEN" instructions):

1. IF alarm is ringing THEN it's time to wake up
2. IF it's time to wake up THEN you should get out of bed
3. IF you are out of bed AND you feel hungry THEN you should eat breakfast
4. IF you ate breakfast THEN you should brush your teeth
5. IF you brushed teeth THEN you are ready for school

# Introduction to Chaining Methods

---

## The Backward Chaining Process:

Step 1: Start with the Goal (The problem)

- Hypothesis: I am ready for school ✗ (But I'm actually late! Let's prove why I'm NOT ready)

Step 2: Work Backward (Find what's missing)

Check Rule 5:

- To be ready for school, I must have brushed teeth
- Question: Did I brush my teeth? ✗ No
- New Sub-goal: Prove brushed teeth

# Introduction to Chaining Methods

---

## Check Rule 4:

- To have brushed teeth, I must have eaten breakfast
- Question: Did I eat breakfast? ✗ No
- New Sub-goal: Prove eaten breakfast

## Check Rule 3:

- To have eaten breakfast, I must be out of bed AND feel hungry
- Question 1: Am I out of bed? ✓ Yes
- Question 2: Do I feel hungry? ✓ Yes
- So why didn't I eat breakfast? Let's keep looking...

# Introduction to Chaining Methods

---

## Check Rule 2:

- To be out of bed, it must be time to wake up
- Question: Was it time to wake up? ☒ Yes (it's 8 AM!)

## Check Rule 1:

- For it to be time to wake up, the alarm must be ringing
- Question: Did the alarm ring? ☐ **FOUND THE PROBLEM!**

# Introduction to Chaining Methods

---

## Final Result:

Starting from: I am not ready for school (The observed problem)

Found the root cause: Alarm did not ring

# Introduction to Chaining Methods

---

## Comparison: Forward vs. Backward Chaining

Feature	Forward Chaining	Backward Chaining
<b>Direction</b>	Data-Driven (Bottom-Up)	Goal-Driven (Top-Down)
<b>Starts With</b>	Facts in the Database	Hypothesis (Goal) to Prove
<b>Explanation</b>	Difficult ("How did you get here?")	Easy ("Why do you ask that?")