

Foundations of AI

7. Propositional Logic

Rational Thinking, Logic, Resolution
Wolfram Burgard and Bernhard Nebel

Contents

- Agents that think rationally
- The wumpus world
- Propositional logic: syntax and semantics
- Logical entailment
- Logical derivation (resolution)

Agents that Think Rationally

- Until now, the focus has been on agents that **act rationally**.
- Often, however, rational action requires **rational** (logical) **thought** on the agent's part.
- To that purpose, portions of the world must be represented in a **knowledge base**, or **KB**.
 - A KB is composed of sentences in a language with a truth theory (logic), i.e. we (being external) can **interpret** sentences as **statements** about the world. (**semantics**)
 - Through their **form**, the sentences themselves have a **causal influence** on the agent's behaviour in a way that is correlated with the contents of the sentences. (**syntax**)
- Interaction with the KB through ASK and TELL (simplified):
ASK(KB, α) = yes exactly when α follows from the KB
TELL(KB, α) = KB' so that α follows from KB'
FORGET(KB, α) = KB' non-monotonic (will not be discussed)

3 Levels

In the context of knowledge representation, we can distinguish three levels [Newell 1990]:

Knowledge level: Most abstract level. Concerns the total knowledge contained in the KB. For example, the automated DB information system knows that a trip from Freiburg to Basel costs 18€.

Logical level: Encoding of knowledge in a formal language.

Price(Freiburg, Basel, 18.00)

Implementation level: The internal representation of the sentences, for example:

- As a string "Price(Freiburg, Basel, 18.00)"
- As a value in a matrix

When ASK and TELL are working correctly, it is possible to remain on the knowledge level. Advantage: very comfortable user interface. The user has his/her own mental model of the world (statements about the world) and communicates it to the agent (TELL).

A Knowledge-Based Agent

A knowledge-based agent uses its knowledge base to

- represent its background knowledge
- store its observations
- store its executed actions
- ... derive actions

```
function KB-AGENT(percept) returns an action  
  static: KB, a knowledge base  
           t, a counter, initially 0, indicating time  
  
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))  
  action ← ASK(KB, MAKE-ACTION-QUERY(t))  
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))  
  t ← t + 1  
  return action
```

The Wumpus World (1)

- A 4 x 4 grid
- In the square containing the **wumpus** and in the directly adjacent squares, the agent perceives a stench.
- In the squares adjacent to a **pit**, the agent perceives a breeze.
- In the square where the **gold** is, the agent perceives a glitter.
- When the agent walks into a **wall**, it perceives a bump.
- When the wumpus is **killed**, its scream is heard everywhere.
- Percepts are represented as a 5-tuple, e.g.,

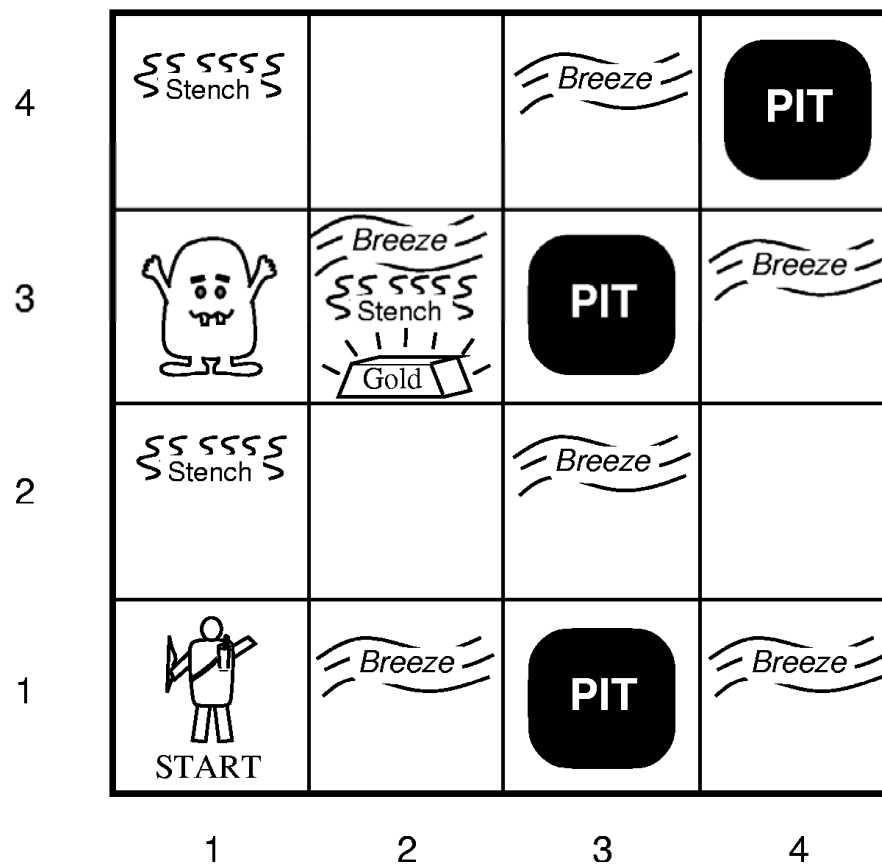
[Stench, Breeze, Glitter, None, None]

means that it stinks, there is a breeze and a glitter, but no bump and no scream. The agent *cannot* perceive its own location!

The Wumpus World (2)

- Actions: Go forward, turn right by 90°, turn left by 90°, pick up an object in the same square (grab), shoot (there is only one arrow), leave the cave (only works in square [1,1]).
- The agent dies if it falls down a pit or meets a live wumpus.
- Initial situation: The agent is in square [1,1] facing east. Somewhere exists a wumpus, a pile of gold and 3 pits.
- Goal: Find the gold and leave the cave.

The Wumpus World (3): A Sample Configuration



The Wumpus World (4)

[1,2] and [2,1] are safe:

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

(a)

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

(b)

The Wumpus World (5)

The wumpus is in [1,3]!

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(a)

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(b)

Declarative Languages

Before a system that is capable of learning, thinking, planning, explaining, ... can be built, one must find a way to **express** knowledge.

We need a precise, declarative language.

- **Declarative**: System believes P iff it considers P to be **true** (one cannot believe P without an idea of what it means for the world to fulfill P).
- **Precise**: We must know,
 - which symbols represent sentences,
 - what it means for a sentence to be true, and
 - when a sentence follows from other sentences.

One possibility: **Propositional Logic**

Basics of Propositional Logic (1)

Propositions: The building blocks of propositional logic are indivisible, atomic **statements** (atomic propositions), e.g.,

- "The block is red"
- "The wumpus is in [1,3]"

and the logical connectives "and", "or" and "not", which we can use to build **formulae**.

Basics of Propositional Logic (2)

We are interested in knowing the following:

- When is a proposition **true**?
- When does a proposition **follow** from a knowledge base (KB)?
- Symbolically: $KB \models \varphi$
- Can we (syntactically) define the concept of *derivation*,
- Symbolically: $KB \vdash \varphi$
such that it is equivalent to the concept of logical implication conclusion?

→ Meaning and implementation of ASK

Syntax of Propositional Logic

Countable alphabet Σ of **atomic propositions**: P, Q, R, \dots

Logical formulae:	$P \in \Sigma$	atomic formula
	\perp	falsehood
	\top	truth
	$\neg\varphi$	negation
	$\varphi \wedge \psi$	conjunction
	$\varphi \vee \psi$	disjunction
	$\varphi \Rightarrow \psi$	implication
	$\varphi \Leftrightarrow \psi$	equivalence

Operator precedence: $\neg > \wedge > \vee > \Rightarrow = \Leftrightarrow$. (use brackets when necessary)

Atom: atomic formula

Literal: (possibly negated) atomic formula

Clause: disjunction of literals

Semantics: Intuition

Atomic propositions can be *true* (T) or *false* (F).

The truth of a formula follows from the truth of its atomic propositions (*truth assignment* or *interpretation*) and the connectives.

Example:

$$(P \vee Q) \wedge R$$

- If P and Q are *false* and R is *true*, the formula is *false*
- If P and R are *true*, the formula is *true* regardless of what Q is.

Semantics: Formally

A **truth assignment** of the atoms in Σ , or an **interpretation** over Σ , is a function

$$I : \Sigma \rightarrow \{T, F\}$$

Interpretation $I(\varphi)$ or φ^I of a formula φ :

$$I \models \top$$

$$I \not\models \perp$$

$$I \models P \quad \text{iff} \quad P^I = T$$

$$I \not\models \neg\varphi \quad \text{iff} \quad I \models \varphi$$

$$I \models \varphi \wedge \psi \quad \text{iff} \quad I \models \varphi \text{ and } I \models \psi$$

$$I \models \varphi \vee \psi \quad \text{iff} \quad I \models \varphi \text{ or } I \models \psi$$

$$I \models \varphi \Rightarrow \psi \quad \text{iff} \quad \text{if } I \models \varphi, \text{ then } I \models \psi$$

$$I \models \varphi \Leftrightarrow \psi \quad \text{iff} \quad \text{if } I \models \varphi \text{ if and only if } I \models \psi$$

I **satisfies** φ ($I \models \varphi$) or φ is **true** under I , when $I(\varphi) = T$.

Example

$$I : \begin{cases} P \rightarrow T \\ Q \rightarrow T \\ R \rightarrow F \\ S \rightarrow T \\ \dots \end{cases}$$

$$\varphi = ((P \vee Q) \Leftrightarrow (R \vee S)) \wedge (\neg(P \wedge Q) \wedge (R \wedge \neg S)).$$

Question: $I \models \varphi$?

Terminology

An interpretation I is called a **model** of φ if $I \models \varphi$.

An interpretation is a **model** of a **set of formulae** if it fulfils all formulae of the set.

A formula φ is

- **satisfiable** if there exists I that satisfies φ ,
- **unsatisfiable** if φ is not satisfiable,
- **falsifiable** if there exists I that doesn't satisfy φ , and
- **valid** (a **tautology**) if $I \models \varphi$ holds for all I .

Two formulae are

- **logically equivalent** ($\varphi \equiv \psi$) if $I \models \varphi$ iff $I \models \psi$ holds for all I .

The Truth Table Method

How can we decide if a formula is **satisfiable**, **valid**, etc.?

→Generate a **truth table**

Example: Is $\varphi = ((P \vee H) \wedge \neg H) \Rightarrow P$ valid?

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$(P \vee H) \wedge \neg H \Rightarrow P$
F	F	F	F	T
F	T	T	F	T
T	F	T	T	T
T	T	T	F	T

Since the formula is true for all possible combinations of truth values (satisfied under all interpretations), φ is **valid**.

Satisfiability, falsifiability, unsatisfiability likewise.

Normal Forms

- A formula is in **conjunctive normal form** (CNF) if it consists of a conjunction of disjunctions of literals $l_{i,j}$, i.e., if it has the following form:

$$\bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} l_{i,j} \right)$$

- A formula is in **disjunctive normal form** (DNF) if it consists of a disjunction of conjunctions of literals:

$$\bigvee_{i=1}^n \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right)$$

- For every formula, there exists at least one equivalent formula in CNF and one in DNF.
- A formula in DNF is satisfiable iff one disjunct is satisfiable.
- A formula in CNF is valid iff every conjunct is valid.

Producing CNF

1. Eliminate \Rightarrow and \Leftrightarrow : $\alpha \Rightarrow \beta \rightarrow (\neg\alpha \vee \beta)$ etc.
2. Move \neg inwards: $\neg(\alpha \wedge \beta) \rightarrow (\neg\alpha \vee \neg\beta)$ etc.
3. Distribute \vee over \wedge : $((\alpha \wedge \beta) \vee \gamma) \rightarrow ((\alpha \vee \gamma) \wedge (\beta \vee \gamma))$
4. Simplify: $\alpha \vee \alpha \rightarrow \alpha$ etc.

The result is a conjunction of disjunctions of literals

An analogous process converts any formula to an equivalent formula in DNF.

- During conversion, formulae can expand *exponentially*.
- Note: Conversion to CNF formula can be done *polynomially* if only satisfiability should be preserved

Logical Implication: Intuition

A set of formulae (a KB) usually provides an incomplete description of the world, i.e., leaves the truth values of a proposition open.

Example: $KB = \{P \vee Q, R \vee \neg P, S\}$

is definitive with respect to S , but leaves P, Q, R open (although they cannot take on arbitrary values).

Models of the KB:

P	Q	R	S
F	T	F	T
F	T	T	T
T	F	T	T
T	T	T	T

In all models of the KB, $Q \vee R$ is true, i.e., $Q \vee R$ follows logically from KB.

Logical Implication: Formal

The formula φ follows logically from the KB if φ is true in all models of the KB (symbolically $KB \models \varphi$):

$$KB \models \varphi \text{ iff } I \models \varphi \text{ for all models } I \text{ of KB}$$

Note: The \models symbol is a *meta-symbol*

Some properties of logical implication relationships:

- **Deduction theorem:** $KB \cup \{\varphi\} \models \psi$ iff $KB \models \varphi \Rightarrow \psi$
- **Contraposition theorem:** $KB \cup \{\varphi\} \models \neg\psi$ iff $KB \cup \{\psi\} \models \neg\varphi$
- **Contradiction theorem:** $KB \cup \{\varphi\}$ is unsatisfiable iff $KB \models \neg\varphi$

Question: Can we determine $KB \models \varphi$ without considering all interpretations (the truth table method)?

Proof of the Deduction Theorem

" \Rightarrow " Assumption: $KB \cup \{\varphi\} \models \psi$, i.e., every model of $KB \cup \{\varphi\}$ is also a model of ψ .

Let I be any model of KB . If I is also a model of φ , then it follows that I is also a model of ψ .

This means that I is also a model of $\varphi \Rightarrow \psi$, i.e., $KB \models \varphi \Rightarrow \psi$.

" \Leftarrow " Assumption: $KB \models \varphi \Rightarrow \psi$. Let I be any model of KB that is also a model of φ , i.e., $I \models KB \cup \{\varphi\}$.

From the assumption, I is also a model of $\varphi \Rightarrow \psi$ and thereby also of ψ , i.e., $KB \cup \{\varphi\} \models \psi$.

Proof of the Contraposition Theorem

$$KB \cup \{\varphi\} \models \neg\psi$$

$$\text{iff } KB \models \varphi \Rightarrow \neg\psi \quad (1)$$

$$\text{iff } KB \models (\neg\varphi \vee \neg\psi)$$

$$\text{iff } KB \models (\neg\psi \vee \neg\varphi)$$

$$\text{iff } KB \models \psi \Rightarrow \neg\varphi$$

$$\text{iff } KB \cup \{\psi\} \models \neg\varphi \quad (2)$$

Note:

(1) and (2) are applications of the deduction theorem.

Inference Rules, Calculi and Proofs

We can often **derive** new formulae from formulae in the KB. These new formulae should **follow logically** from the syntactical structure of the KB formulae.

Example: If the KB is $\{\dots, (\varphi \Rightarrow \psi), \dots, \varphi, \dots\}$, then ψ is a logical consequence of KB

→ **Inference rules**, e.g.,
$$\frac{\varphi, \varphi \Rightarrow \psi}{\psi}$$

Calculus: Set of inference rules (potentially including so-called logical axioms)

Proof step: Application of an inference rule on a set of formulae.

Proof: Sequence of proof steps where every newly-derived formula is added, and in the last step, the **goal formula** is produced.

Soundness and Completeness

In the case where in the calculus C there is a proof for a formula φ , we write

$$KB \vdash_C \varphi$$

(optionally without subscript C).

A calculus C is **sound** (or **correct**) if all formulae that are derivable from a KB actually follow logically.

$$KB \vdash_C \varphi \text{ implies } KB \models \varphi$$

This normally follows from the soundness of the inference rules and the logical axioms.

A calculus is **complete** if every formula that follows logically from the KB is also derivable with C from the KB:

$$KB \models \varphi \text{ implies } KB \vdash_C \varphi$$

Resolution: Idea

We want a way to **derive** new formulae that does not depend on testing every interpretation.

Idea: We attempt to show that a set of formulae is unsatisfiable.

Condition: All formulae must be in CNF.

But: In most cases, the formulae are close to CNF (and there exists a fast satisfiability-preserving transformation – Theoretical Computer Science course).

Nevertheless: In the **worst case**, this derivation process requires an exponential amount of time (this is, however, probably unavoidable).

Resolution: Representation

Assumption: All formulae in the KB are in CNF.

Equivalently, we can assume that the KB is a *set of clauses*.

Due to commutativity, associativity, and idempotence of \vee , **clauses** can also be understood as **sets of literals**. The **empty set of literals** is denoted by \square .

Set of clauses: Δ

Set of literals: C, D

Literal: l

Negation of a literal: \bar{l}

An interpretation I satisfies C iff there exists $l \in C$ such that $I \models l$. I satisfies Δ if for all $C \in \Delta : I \models C$, i.e., $I \not\models \square, I \not\models \{\square\}, I \models \{\}$, for all I .

The Resolution Rule

$$\frac{C_1 \cup \{l\}, C_2 \cup \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ are called **resolvents** of the parent clauses $C_1 \cup \{l\}$ and $C_1 \cup \{\bar{l}\}$. l and \bar{l} are the **resolution literals**.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not equivalent to the parent clauses, but it follows from them!

Notation: $R(\Delta) = \Delta \cup \{C \mid C \text{ is a resolvent of two clauses from } \Delta\}$

Derivations

We say D can be **derived** from Δ using resolution, i.e.,

$$\Delta \vdash D,$$

if there exist $C_1, C_2, C_3, \dots, C_n = D$ such that

$$C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\}), \text{ for } 1 \leq i \leq n.$$

Lemma (soundness) If $\Delta \vdash D$, then $\Delta \models D$.

Proof idea: Since all $D \in R(\Delta)$ follow logically from Δ , the lemma results through induction over the length of the derivation.

Completeness?

Is resolution also complete? I.e. is

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi$$

valid? Only for clauses. Consider:

$$\{\{a,b\}, \{\neg b,c\}\} \models \{a,b,c\} \not\vdash \{a,b,c\}$$

But it can be shown that resolution is **refutation-complete**:

$$\Delta \text{ is unsatisfiable implies } \Delta \vdash \square,$$

Theorem: Δ is unsatisfiable iff $\Delta \vdash \square$

With the help of the contradiction theorem, we can show that $KB \models \varphi$.

Resolution: Overview

- Resolution is a refutation-complete proof process. There are others (Davis-Putnam Procedure, Tableaux Procedure, ...).
- In order to implement the process, a **strategy** must be developed to determine which resolution steps will be executed and when.
- In the worst case, a resolution proof can take exponential time. This, however, very probably holds for all other proof procedures.
- For CNF formulae in propositional logic, the Davis-Putnam Procedure (backtracking over all truth values) is probably (in practice) the fastest complete process that can also be taken as a type of resolution process.

Where is the Wumpus?

The Situation

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

- A** = Agent
- B** = Breeze
- G** = Glitter, Gold
- OK** = Safe square
- P** = Pit
- S** = Stench
- V** = Visited
- W** = Wumpus

Where is the Wumpus? Knowledge of the Situation

B = Breeze, S = Stench, $B_{i,j}$ = there is a breeze in (i,j)

$$\begin{array}{ll} \neg S_{1,1} & \neg B_{1,1} \\ \neg S_{2,1} & B_{2,1} \\ S_{1,2} & \neg B_{1,2} \end{array}$$

Knowledge about the wumpus and smell:

$$R_1: \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$$

$$R_2: \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$$

$$R_3: \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$$

$$R_4: S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1} \dots$$

To show: $KB \models W_{1,3}$

Clausal Representation of the Wumpus World

Situational knowledge:

$\neg S_{1,1}, \neg S_{2,1}, \neg S_{1,2}, \dots$

Knowledge of rules:

Knowledge about the wumpus and smell:

$R_1: S_{1,1} \vee \neg W_{1,1}, S_{1,1} \vee \neg W_{1,2}, S_{1,1} \vee \neg W_{2,1}$

$R_2: \dots, S_{2,1} \vee \neg W_{2,2}, \dots$

$R_3: \dots$

$R_4: \neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

\dots

Negated goal formula: $\neg W_{1,3}$

Resolution Proof for the Wumpus World

Resolution:

$$\neg W_{1,3}, \neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$$

$$\rightarrow \neg S_{1,2} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$$

$$S_{1,2}, \neg S_{1,2} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$$

$$\rightarrow W_{1,2} \vee W_{2,2} \vee W_{1,1}$$

$$\neg S_{1,1}, S_{1,1} \vee \neg W_{1,1}$$

$$\rightarrow \neg W_{1,1}$$

$$\neg W_{1,1}, W_{1,2} \vee W_{2,2} \vee W_{1,1}$$

$$\rightarrow W_{1,2} \vee W_{2,2}$$

...

$$\neg W_{2,2}, W_{2,2}$$

$$\rightarrow \square$$

From Knowledge to Action

We can now infer new facts, but how do we translate knowledge into action?

Negative selection: Excludes any provably dangerous actions.

$$A_{1,1} \wedge \text{East}_A \wedge W_{2,1} \Rightarrow \neg \text{Forward}$$

Positive selection: Only suggests actions that are provably safe.

$$A_{1,1} \wedge \text{East}_A \wedge \neg W_{2,1} \Rightarrow \text{Forward}$$

Differences?

From the suggestions, we must still select an action.

Problems with Propositional Logic

Although propositional logic suffices to represent the wumpus world, it is rather involved.

1. **Rules** must be set up for each square.

$$R_1: \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$$

$$R_2: \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$$

$$R_3: \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$$

...

We need a time index for each proposition to represent the validity of the proposition over time → further expansion of the rules.

→ More powerful logics exist, in which we can use object variables.

→ First-Order Predicate Logic

Summary

- Rational agents require **knowledge** of their world in order to make rational decisions.
- With the help of a **declarative** (knowledge-representation) language, this knowledge is represented and stored in a **knowledge base**.
- We use **propositional logic** for this (for the time being).
- Formulae of propositional logic can be **valid**, **satisfiable** or **unsatisfiable**.
- The concept of **logical implication** is important.
- Logical implication can be mechanized by using an **inference calculus** → **resolution**.
- Propositional logic quickly becomes impractical when the world becomes too large (or infinite).