ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Intelligenza Artificiale

Knowledge representation and reasoning

Propositional logic

Ivan Heibi
Dipartimento di Filologia Classica e Italianistica (FICLIT)
Ivan.heibi2@unibo.it

Knowledge-based agents

Humans, it seems, know things; and what they know helps them do things.

Intelligence of humans is achieved - not (always) by purely reflex mechanisms but by processes of reasoning that operate on internal representation of knowledge.

In AI, **knowledge- based agents** use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take.

The central component of a knowledge-based agents is its **knowledge base**.

Knowledge base

*A knowledge base (KB) is a "**set of sentences**"* .

Each **sentence** is expressed in a language (i.e. the **knowledge representation language**) and represents some **assertion** about the world (called, **axioms**).

Operations:

- TELL (add a new sentence to the knowledge base)
- ASK (query the knowledge base)

Both operation may involve **inference**.
Inference is the process of deriving consequences from premises.

Agents with background knowledge

The background knowledge is the knowledge “initially” contained in the KB of an agent.

This may include knowledge about:

- Properties of the environments
- Properties of the objects
- Events might be useful for the task of the agents
- Other agents in the environment

....

Knowledge level and Symbol/Implementation Level

Each time the agent program is called does three things:

- TELLs the knowledge base what it perceives
- ASKs the knowledge base what action it should perform.
- TELLs the knowledge base which action was chosen, and the agent executes the action.

It is similar to the the agent program skeleton already seen but the behaviour of a knowledge based agents can be completely specified at **knowledge level**.
> ***We need specify only what agent knows and what its goals are.***

It is worth noticing that the description of the behaviour is **independent** of how the agent actually work how the operations are **implemented**. The implementation of the behaviour is done at **symbol/implementation level**

Knowledge Base Agent - Example

An automated taxi with:

- Goal of taking a passenger from Bologna to Milan
- He knows that the A1 is one possible way between the two locations
- He knows that the A1 is faster usually

We expect it to drive through A1 because it knows that will achieve its goal.

This analysis is **independent** of how the taxi works at the **implementation level** (it works on the **knowledge level).** such as:
> *“The geographical knowledge is implemented as linked lists or pixel maps?“*
> *it reasons by manipulating strings or by propagating noisy signals in a network of neurons?*
> … etc

Knowledge Base Agent – pseudocode

```
function KB-AGENT(percept) returns an action

   persistent: 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
```

Logic

Sentences of a knowledge base are expressed according to the **syntax** of the representation language. The **syntax** specifies all the sentences that are well formed
>Example: in arithmetics, X+Y=4 vs 4XY+=

A logic must also define the **semantics** (meaning) of sentences. The **semantics** defines the truth of each sentence with respect to each possible world (also called model)
> Example: X + Y = 4 is true in a world where X = 2 and Y=2, but also in a world where X = 3 and Y = 1. The sentence is false in a world where X = 1 and Y = 1

In standard logics, every sentence must be either true or false in each possible world (model).

Satisfaction and Entailment

The **possible models** are just all **possible assignments** of nonnegative integers to the variables x and y, which determine the truth of any sentence of arithmetic whose variables are x and y.

If a **sentence <u>α</u>** is true in model <u>**m**</u>, then we say:

- m <u>**satisfies**</u> α; or
- <u>**m is a model of α**</u>

We use the notation **M(α)** to mean the set of all models of α.

A sentence **α** <u>**entails**</u> a sentence **β**, if **β** logically follows **α**

$$\boldsymbol{\alpha \vDash \beta}$$

$\alpha \vDash \beta$ if and only if, in every model in which α is true, β is also true:

$$\alpha \vDash \beta \text{ if and only if } M(\alpha) \subseteq M(\beta)$$

Example

α: n divisible by 6

β: n divisible by 3

$\alpha \vDash \beta$

Inference

The definition of entailment can be applied to derive conclusions—that is, **to carry out logical inference**

if an inference algorithm ***i*** can derive **α** from **KB** we write:

$$KB \vdash_i \alpha$$

An inference algorithm that derives only entailed sentences is called **sound** or **truth-preserving**

An inference algorithm is **complete** if it can derive any sentence that is entailed

Inference – summary

Suppose that sentences **A, B and C are derivable from a KB**, but only A and B are **entailed** by the KB.

> An algorithm that **derives C** from the KB is **not sound**.

> An algorithm that **derives only A** from the KB is **not complete**.

> An algorithm that **derives only A and B** from the KB is **sound and complete**

Inference – example

An AI system (KB) designed for image recognition.

Sentences A, B, and C could represent different information deducible from KB:

- A: *"The object is a dog"*
- B: *"The object is brown"*
- C: *"The object is a domestic animal"*

Based on the information that is logically implied by the premises (KB), we might have only **A and B as entailed consequences**.

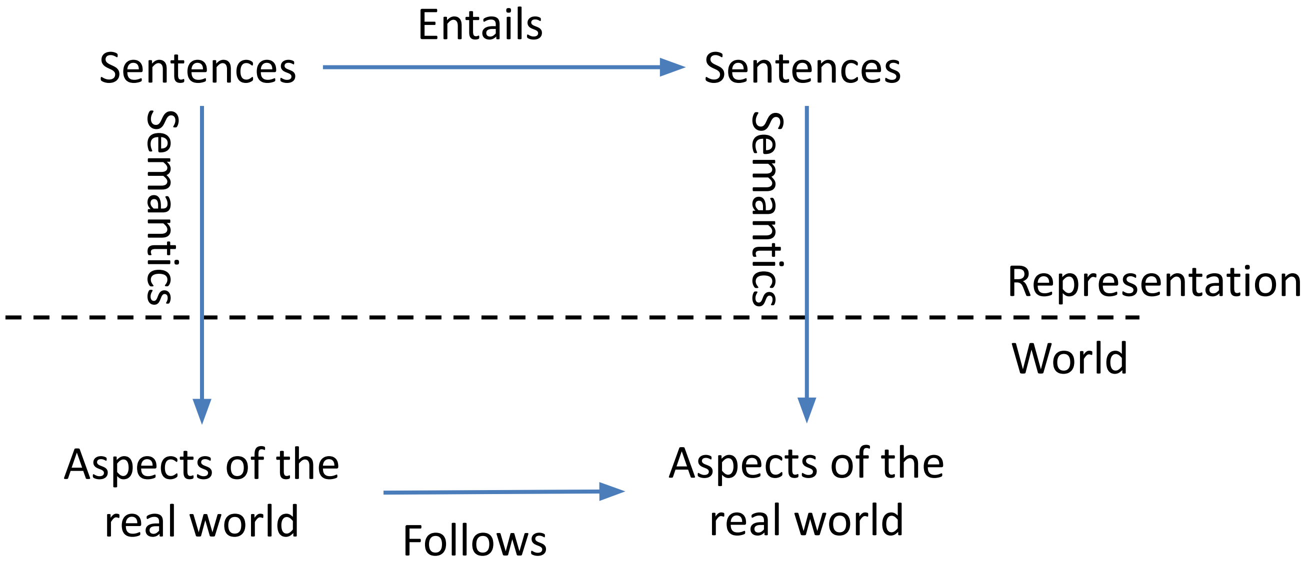
> We could say that "The object is a dog" (A) and "The object is brown" (B) are the only information that we can guarantee to be true (based on KB)

> The sentence C ("The object is a domestic animal") might be deducible, but it could depend on further considerations not explicitly stated in KB

Grounding

The **grounding** is the connection between the logical statements and the aspects of the real world where the agent exists.

> *how do we know that KB is true in the real world?*



Logical implication vs Logical consequence

Logical consequence (⊨) is a semantic relation between a set of premises and a conclusion, and it is more related to the notion of truth in models.

> **Logical consequence** is a more general and meta-mathematical concept;

Logical implication (→) is a syntactic operation between two formulas and is part of the formal language of propositional logic and predicate calculus.

> **Implication** is an element of formal mathematical logic.

Self assessment

https://forms.gle/TZqhW6TZFcinjq7t6

Propositional Logic

Propositional Logic: syntax

The **atomic sentences** consist of a single **propositional symbol**

Each **symbol** stands for a proposition that can be **true or false**

Symbols start with an **uppercase** and may contain other letters or subscripts (e.g. P, Q, Flag, StoreOpen, etc)

The symbol ***True*** is always true, ***False*** is always false

Complex sentences are constructed from simpler sentences, using parentheses an **logical connectives**

Propositional logic: Logical connectives

¬ (not). used to negate a sentence (e.g., ¬A)

∧ (and). used to conjunct two sentences (e.g., A ∧ B)

∨ (or). used to disjunct two sentences (e.g., A ∨ B)

⇒ (implies). used to assert implications (e.g., A ⇒ B, if A is *True* then B is *True*), Implications are also known as rules or if–then statements.

⇔ (if and only if). used to assert equivalences (e.g., A ⇔ B, A is *True* if and only if B is *True*)

S → AtomicSentence | ComplexSentence

AtomicSentence → *True*| *False*| P| Q| R| ...

ComplexSentence → | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

*operator precedence: ¬,∧,∨,⇒,⇔

Propositional logic: semantics

The **semantics** defines the rules for determining the truth of a sentence with respect to a particular model.

In propositional logic, a **model** simply fixes the **truth value** for every proposition symbol.
> For example a model could be m = {A = false, B = true}

Once the truth value is specified for every proposition symbol of the model, the semantics must specify how to compute the truth value of any sentence.

- $\neg A$ is true iff (if and only if) A is false in the model m
- $A \wedge B$ is true iff both A and B are true in the model m
- $A \vee B$ is true iff either A or B is true in the model m
- $A \Rightarrow B$ is true unless A is true and B is false
- $A \Leftrightarrow B$ is true iff are both true or both false in the model m.

Propositional logic: truth tables

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

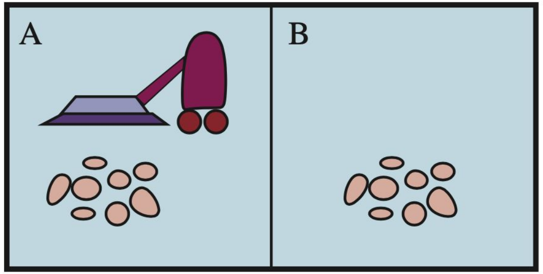
Propositional logic: implication truth table

if P then Q

	P	Q	$P \Rightarrow Q$
The value of Q is **not** relevant	*false*	*false*	*true*
	false	*true*	*true*
The value of Q is relevant	*true*	*false*	*false*
	true	*true*	*true*

Example: the vacuum cleaner

CleanA is *True* if position **A** is clean
CleanB is *True* if position **B** is clean
VacuumA is *True* if the vacuum position is **A**
VacuumB is *True* if the vacuum position is **B**

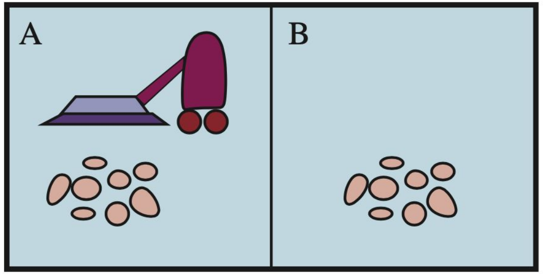


Example: the vacuum cleaner

CleanA is *True* if position **A** is clean
CleanB is *True* if position **B** is clean
VacuumA is *True* if the vacuum position is **A**
VacuumB is *True* if the vacuum position is **B**

KB:

$$\text{VacuumA} \Rightarrow \text{CleanA}$$
$$\text{VacuumB} \Rightarrow \text{CleanB}$$
$$\text{VacuumA} \wedge \text{VacuumB} \Leftrightarrow \textit{False}$$
$$\text{VacuumA} \vee \text{VacuumB} \Leftrightarrow \textit{True}$$



Example: the vacuum cleaner

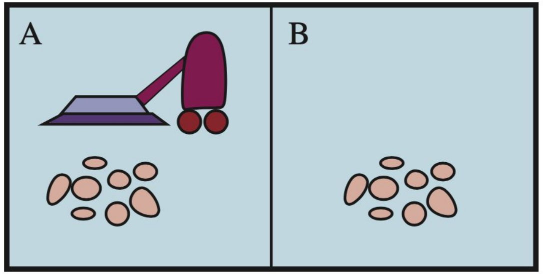
CleanA is *True* if position **A** is clean
CleanB is *True* if position **B** is clean
VacuumA is *True* if the vacuum position is **A**
VacuumB is *True* if the vacuum position is **B**

KB:

$$\text{VacuumA} \Rightarrow \text{CleanA}$$
$$\text{VacuumB} \Rightarrow \text{CleanB}$$
$$\text{VacuumA} \land \text{VacuumB} \Leftrightarrow \mathit{False}$$
$$\text{VacuumA} \lor \text{VacuumB} \Leftrightarrow \mathit{True}$$

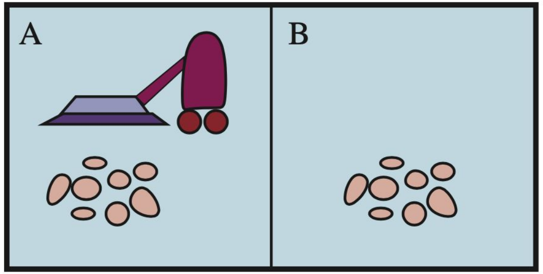
Goal:

$$\text{CleanA} \land \text{CleanB}$$



Example: the vacuum cleaner

CleanA is *True* if position **A** is clean
CleanB is *True* if position **B** is clean
VacuumA is *True* if the vacuum position is **A**
VacuumB is *True* if the vacuum position is **B**



KB:

$$VacuumA \Rightarrow CleanA$$
$$VacuumB \Rightarrow CleanB$$
$$VacuumA \land VacuumB \Leftrightarrow \mathit{False}$$
$$VacuumA \lor VacuumB \Leftrightarrow \mathit{True}$$

Goal:

$$CleanA \land CleanB$$

Percept:

$$(VacuumA,\ \neg CleanA)$$

Example: the vacuum cleaner

In which model the goal (i.e. a sentence) is satisfied?

A **greedy algorithm** would enumerate all possible worlds and check that the goal is true in very model in which the KB is true.

The KB is true in:

- m1 = {VacuumA = F, VacuumB = T, CleanA = T, CleanB = T}
- m2 = {VacuumA = T, VacuumB= F, CleanA = T, CleanB = T}
- ...

There are finitely models (in particular 2n) => *Time complexity O(2n)*

Finitely many is not always the same as "few"

A simple inference procedure – truth table (1)

$$KB \vDash CleanA \land CleanB \quad iff \quad M(KB) \subseteq M(CleanA \land CleanB)\ ?$$

VA	VB	CA	CB	VA ⇒ CA	VB ⇒ CB	VA ∧ VB ⇔ False	VA ∨ VB ⇔ True	CA ∧ CB
F	F	F	F	T	T	T	F	F
F	F	F	T	T	T	T	F	F
F	F	T	F	T	T	T	F	F
F	F	T	T	T	T	T	F	T
F	T	F	F	T	F	T	T	F
F	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**F**
F	T	T	F	T	F	T	T	F
F	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	F	F	F	F	T	T	T	F
T	F	F	T	F	T	T	T	F
T	**F**	**T**	**F**	**T**	**T**	**T**	**T**	**F**
T	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	T	F	F	F	F	F	T	F
T	T	F	T	F	T	F	T	F
T	T	T	F	T	F	F	T	F
T	T	T	T	T	T	F	T	T

CA = CleanA
CB = CleanB
VA = VacuumA
VB = VacuumB

A simple inference procedure – truth table (2)

$$KB \models CleanA \lor CleanB \quad iff \quad M(KB) \subseteq M(CleanA \lor CleanB)\ ?$$

VA	VB	CA	CB	VA ⇒ CA	VB ⇒ CB	VA ∧ VB ⇔ False	VA ∨ VB ⇔ True	CA ∨ CB
F	F	F	F	T	T	T	F	F
F	F	F	T	T	T	T	F	F
F	F	T	F	T	T	T	F	F
F	F	T	T	T	T	T	F	T
F	T	F	F	T	F	T	T	F
F	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**
F	T	T	F	T	F	T	T	F
F	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	F	F	F	F	T	T	T	F
T	F	F	T	F	T	T	T	F
T	**F**	**T**	**F**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	T	F	F	F	F	F	T	F
T	T	F	T	F	T	F	T	F
T	T	T	F	T	F	F	T	F
T	T	T	T	T	T	F	T	T

CA = CleanA
CB = CleanB
VA = VacuumA
VB = VacuumB

Theorem proving: Logical equivalence

Theorem proving applies rules of inference directly to the sentences in our KB in order to construct a proof of the desired sentences without consulting models

Theorem provers use **logical equivalence**: two sentences are logically equivalent if they are true in the same set of models

$$\alpha \equiv \beta$$

For instance:

$$A \land B \equiv B \land A$$

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

Theorem proving: Logical equivalence

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

Theorem proving: Validity

A sentence is **valid** if its **true** in <u>all models</u>

For example, this is a valid sentence:

$A \lor \neg A$

V**alid** sentences are also called <u>**tautologies**</u>

What about this one $\neg(A \land \neg A)$?

Theorem proving: Deduction theorem

Why do we need valid sentences?

For any pair of sentences α and *β* we have that:

$\alpha \models \beta$ if and only if $\alpha \Rightarrow \beta$ is valid

The theorem establishes a relationship between proofs in formal logic systems and conditional statements

Deduction theorem: valid example

α = "It is raining and there are clouds" ($P \land N$)
β = "There are clouds" (N)

Question: Is **α ⊨ β**?
Yes

if "It is raining and there are clouds" is **true**,
then "There are clouds" must necessarily be **true**.

The implication $\mathbf{(P \land N) \Rightarrow N}$ is always true, making it a tautology.
Therefore, $\alpha \models \beta$.

Deduction theorem: not valid example

α = "It is raining" (P)
β = "There are clouds" (N)

Question: Is **α** ⊨ **β**?
No

it can rain even without visible clouds (e.g., artificial rain).

The implication **P ⇒ N** is <u>not always true</u>, so it is not a tautology.
Therefore, α ⊭ β

Theorem proving: Satisfiability

A sentence is **satisfiable** if it is true in, or **satisfied by, some model**
> can be checked by enumerating the possible models until one is found that satisfies the sentence.

The problem of determining the satisfiability of sentences in propositional logic is called **SAT problem**.

For example:

$(P \lor \neg Q) \land (\neg P \lor R)$ is **satisfiable**, since at least one model makes it true,

For instance:

- P = *True*,
- Q=*False*,
- R=*True*

Validity and Satisfiability

Validity and satisfiability are of course connected!

- α is valid if and only if $\neg\alpha$ is unsatisfiable;
- α is satisfiable if and only if $\neg\alpha$ is not valid.

Another important result is:

$\alpha \models \beta$ *if and only if the sentence ($\alpha \land \neg\beta$) is unsatisfiable*

> Proof it by **contradiction:** One assumes a sentence β to be false and shows that this leads to a contradiction with known axioms α.

Example sentence generation

$(P \lor \neg Q) \land$
$(\neg P \lor R)$

S → AtomicSentence | ComplexSentence

AtomicSentence → *True*| *False*| P| Q| R| ...

ComplexSentence → | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

S

Example sentence generation

$(P \lor \neg Q) \land$
$(\neg P \lor R)$

S → AtomicSentence | ComplexSentence

AtomicSentence → *True*| *False*| P| Q| R| ...

ComplexSentence → | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

S

ComplexSentence

S∧S

Example sentence generation

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

S → AtomicSentence | ComplexSentence

AtomicSentence → *True*| *False*| P| Q| R| ...

ComplexSentence → | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

S∧S

ComplexSentence	ComplexSentence
(S)	(S)

Example sentence generation

$(P \lor \neg Q) \land$
$(\neg P \lor R)$

S → AtomicSentence | ComplexSentence

AtomicSentence → *True*| *False*| P| Q| R| ...

ComplexSentence → | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

(S) ∧ (S)

ComplexSentence	ComplexSentence
S∨S	S∨S

Example sentence generation

$(P \lor \neg Q) \land$
$(\neg P \lor R)$

$S \rightarrow$ AtomicSentence | ComplexSentence

AtomicSentence $\rightarrow$ *True*| *False*| P| Q| R| ...

ComplexSentence $\rightarrow$ | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

$(S \lor S) \land (S \lor S)$

AtomicSentence	ComplexSentence	ComplexSentence	AtomicSentence
P	¬S	¬S	R

Example sentence generation

$(P \lor \neg Q) \land$
$(\neg P \lor R)$

$S \rightarrow$ AtomicSentence | ComplexSentence

AtomicSentence $\rightarrow$ *True*| *False*| P| Q| R| ...

ComplexSentence $\rightarrow$ | (S) | ¬S | S∧S | S∨S | S⇒S | S⇔S

$(P \lor \neg S) \land (\neg S \lor R)$

AtomicSentence	AtomicSentence
Q	R

Example sentence generation

$(P \lor \neg Q) \land$
$(\neg P \lor R)$

S → AtomicSentence | ComplexSentence

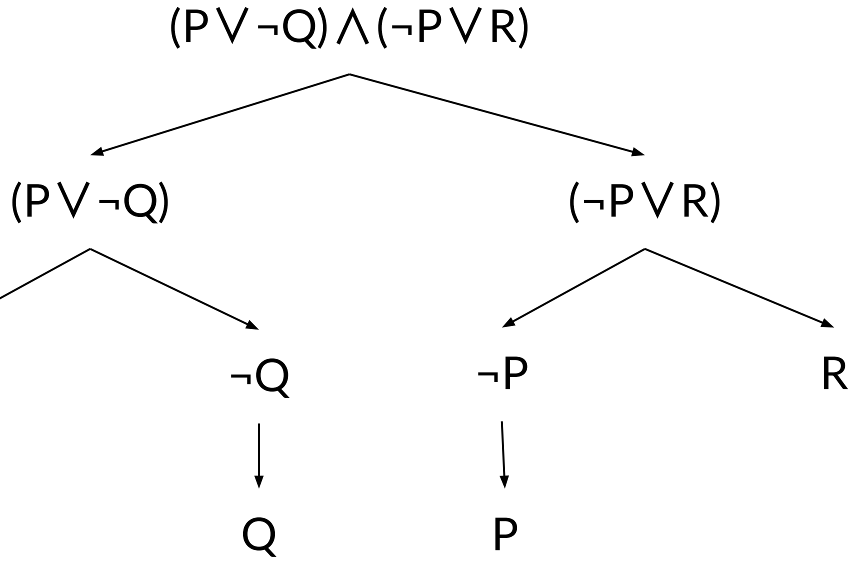
AtomicSentence → *True*| *False*| P| Q| R| ...

ComplexSentence → | (S) | ¬S | S ∧ S | S ∨ S | S ⇒ S |
S ⇔ S

$(P \lor \neg Q) \land (\neg P \lor R)$

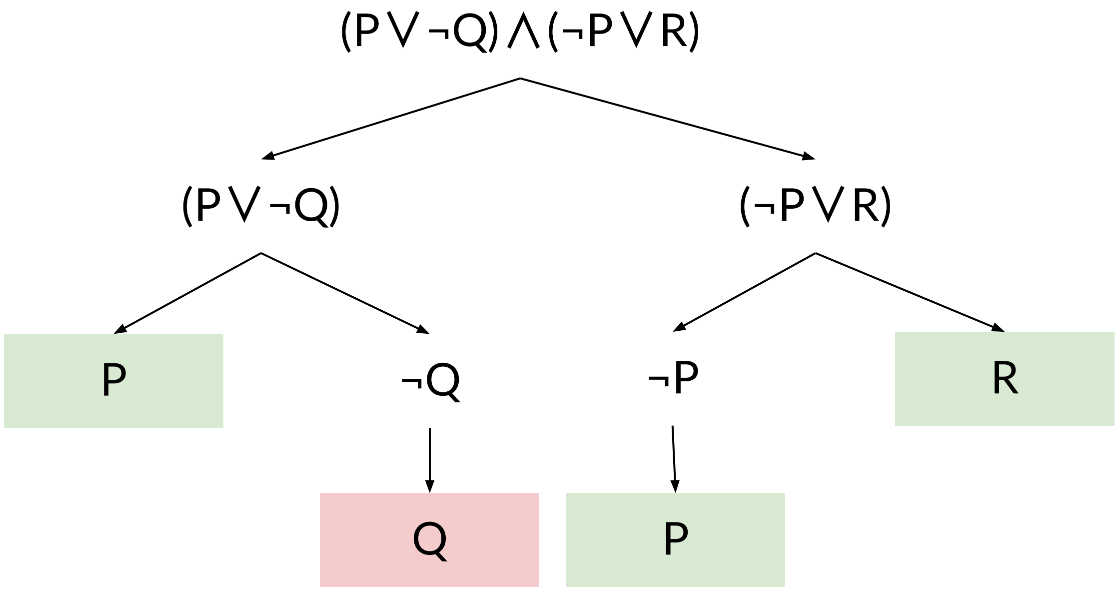
Example sentence satisfiable check

$(P \lor \neg Q) \land (\neg P \lor R)$ is **satisfiable,** when:
P = *True*; Q=*False*; R=*True*



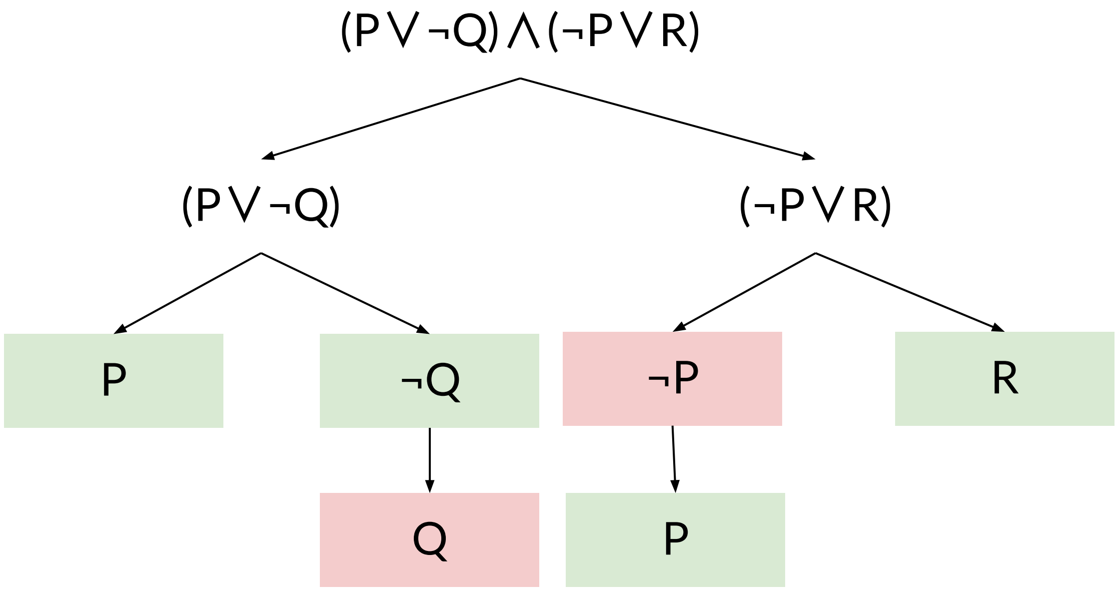
Example sentence satisfiable check

$(P \lor \neg Q) \land (\neg P \lor R)$ is **satisfiable,** when:
P = *True*; Q=*False*; R=*True*



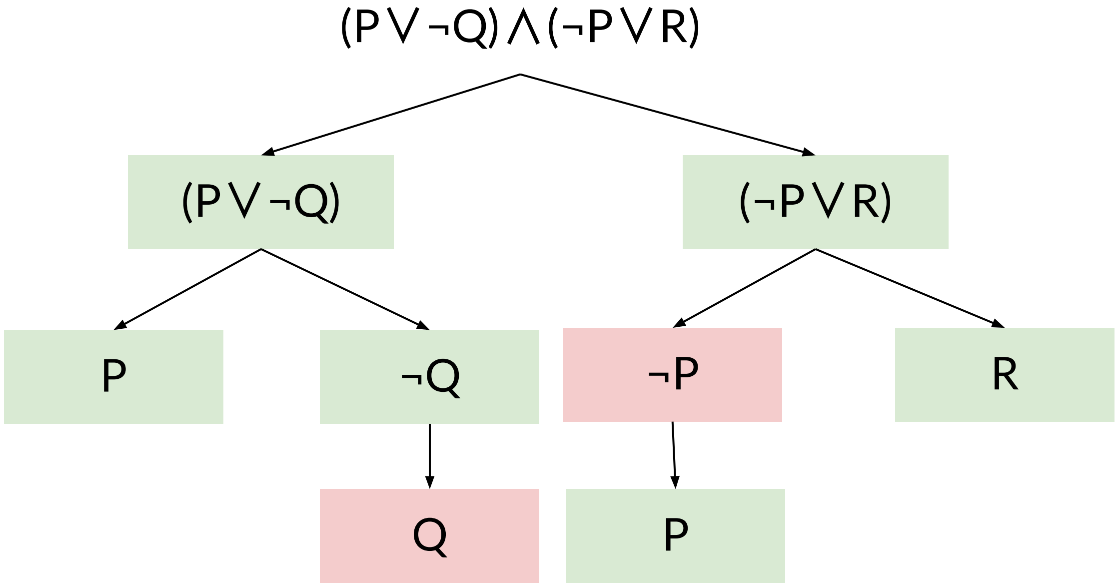
Example sentence satisfiable check

$(P \lor \neg Q) \land (\neg P \lor R)$ is **satisfiable,** when:
P = *True*; Q=*False*; R=*True*



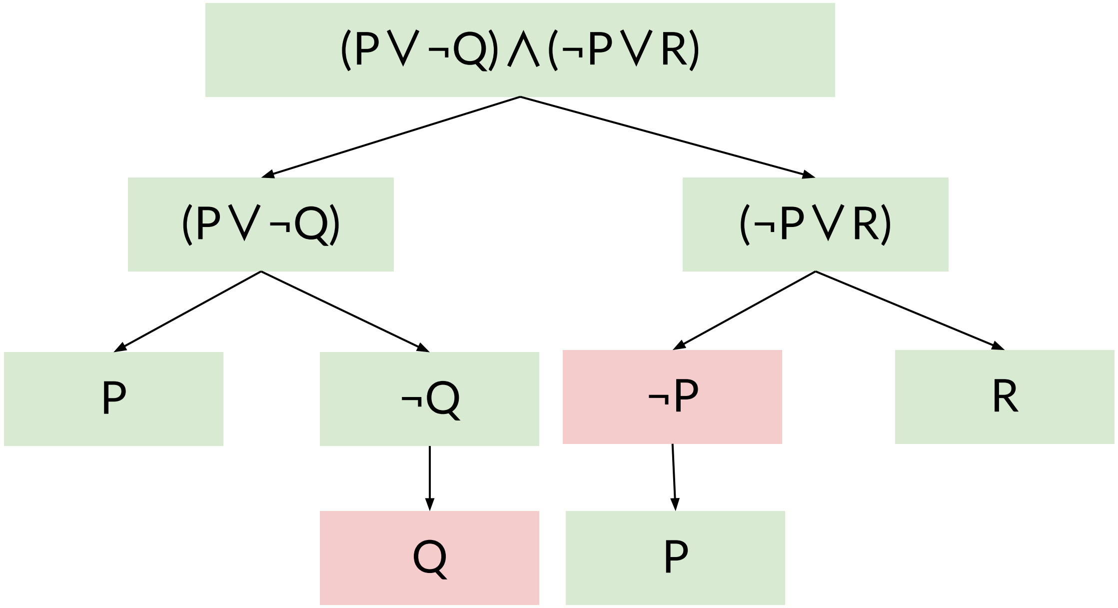
Example sentence satisfiable check

(P ∨ ¬Q) ∧ (¬P ∨ R) is **satisfiable,** when:
P = *True*; Q=*False*; R=*True*



Example sentence satisfiable check

$(P \lor \neg Q) \land (\neg P \lor R)$ is **satisfiable,** when:
P = *True*; Q=*False*; R=*True*



Monotonicity

One final property of logical systems is **monotonicity**, which says that the set of entailed sentences can only increase as information is added to the KB

For any sentences α and β, if KB ⊨ α then KB ∧ β ⊨ α.

For example (Vacuum example):

If the KB contains the additional assertion β stating that Cell A is clean in the world, helps the agent draw additional conclusions, but it cannot invalidate any conclusion α already inferred—such as the conclusion that Cell B is dirty

Inference and proofs: Modus ponens

Inference rules can be applied to derive a proof – a chain of conclusions that leads to the desired goal.

The best-known rule is called Modus Ponens!

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

The notation means that, whenever any sentences of the form $\alpha \Rightarrow \beta$ and α are given, then the sentence β can be inferred.

Inference and proofs: Modus ponens – example

Inference rules can be applied to derive a proof – a chain of conclusions that leads to the desired goal.

The best-known rule is called Modus Ponens!

$$\frac{\boxed{\alpha \Rightarrow \beta,} \quad \boxed{\alpha}}{\boxed{\beta}}$$

If it is raining (α), then the streets are wet (β). (α ⇒ β)

It is raining (α is true).

<u>Therefore, the streets are wet (β is true)</u>

Inference and proofs: AND elimination/introduction

From a conjunction, any of the conjuncts can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

Given two sentences α and β a conjunction could be asserted:

$$\frac{\alpha, \beta}{\alpha \wedge \beta}$$

Inference and proofs: OR elimination/introduction

Given a disjunction between α and β, and the negation of one of the disjuncts, the other could be inferred:

$$\frac{\alpha \lor \beta, \neg\alpha}{\beta}$$

Given α, then a disjunction with any β could be formed:

$$\frac{\alpha}{\alpha \lor \beta}$$

Exercises

Exercises (1)

Which of the following propositional logic sentences is well-formed:

1.	$P \land (Q \lor \neg R)$
2.	$\neg(P \Rightarrow Q)$
3.	$P \land Q \Rightarrow$
4.	$(P \lor Q) \land (\neg P \lor R)$
5.	$\neg(P \land Q \lor \neg)$
6.	$P \Leftrightarrow (Q \land \neg R)$

Exercises (1) – Solution

Which of the following propositional logic sentences is well-formed:

1.	$P \wedge (Q \vee \neg R)$	Yes
2.	$\neg(P \Rightarrow Q)$	Yes
3.	$P \wedge Q \Rightarrow$	No
4.	$(P \vee Q) \wedge$ $(\neg P \vee R)$	Yes
5.	$\neg(P \wedge Q \vee \neg)$	No
6.	$P \Leftrightarrow (Q \wedge \neg R)$	Yes

Exercises (2)

Compute the truth table of $(P \land Q) \lor \neg R$

P	Q	R	¬R	$(P \land Q)$	$(P \land Q) \lor \neg R$
F	F	F			
F	F	T			
F	T	F			
F	T	T			
T	F	F			
T	F	T			
T	T	F			
T	T	T			

Exercises (2) – Solution

Compute the truth table of $(P \land Q) \lor \neg R$

P	Q	R	¬R	$(P \land Q)$	$(P \land Q) \lor \neg R$
F	F	F	**T**	**F**	**T**
F	F	T	**F**	**F**	**F**
F	T	F	**T**	**F**	**T**
F	T	T	**F**	**F**	**F**
T	F	F	**T**	**F**	**T**
T	F	T	**F**	**F**	**F**
T	T	F	**T**	**T**	**T**
T	T	T	**F**	**T**	**T**

Exercises (3)

Compute the truth table of $(P \lor Q) \land (\neg P \land R)$

P	Q	R	¬P	(¬P∧R)	(P∨Q)	(P∨Q) ∧ (¬P∧R)
F	F	F				
F	F	T				
F	T	F				
F	T	T				
T	F	F				
T	F	T				
T	T	F				
T	T	T				

Exercises (3) – Solution

Compute the truth table of $(P \lor Q) \land (\neg P \land R)$

P	Q	R	¬P	(¬P∧R)	(P∨Q)	(P∨Q) ∧ (¬P∧R)
F	F	F	**T**	**F**	**F**	**F**
F	F	T	**T**	**T**	**F**	**F**
F	T	F	**T**	**F**	**T**	**F**
F	T	T	**T**	**T**	**T**	**T**
T	F	F	**F**	**F**	**T**	**F**
T	F	T	**F**	**F**	**T**	**F**
T	T	F	**F**	**F**	**T**	**F**
T	T	T	**F**	**F**	**T**	**F**

Exercises (4)

Let's consider a propositional language where

- H means "Paul is happy"
- P means "Paul paints a picture"
- R means "Rachel is happy"

Formalize the following sentences in propositional logic:

1.	If Paul is happy and paints, then Rachel is happy
2.	If Paul is happy, then he paints a picture
3.	If Rachel isn't happy Paul is happy

Exercises (4) – Solution

Let's consider a propositional language where

- H means "Paul is happy"
- P means "Paul paints a picture"
- R means "Rachel is happy"

Formalize the following sentences in propositional logic:

1.	If Paul is happy and paints, then Rachel is happy	$(H \wedge P) \Rightarrow R$
2.	If Paul is happy, then he paints a picture	$H \Rightarrow P$
3.	If Rachel isn't happy Paul is happy	$\neg R \Rightarrow H$

Exercises (5)

Let's consider the same example of Paul and Rachel.

Premise:

- If Paul is happy and paints a picture ($H \wedge P$), then Rachel is happy (R)
- If Paul is happy (H) then he paints a picture (P)

Given Information: Paul is happy (H)

Inference: Conclude whether Rachel is happy (R) based on the given information.

Exercises (5) – Solution

Let's consider the same example of Paul and Rachel.

Premise:

- If Paul is happy and paints a picture ($H \wedge P$), then Rachel is happy (R)
- If Paul is happy (H) then he paints a picture (P)

Given Information: Paul is happy (H)

Inference: Conclude whether Rachel is happy (R) based on the given information.

1) The premise (1): $(H \wedge P) \Rightarrow R$
2) The premise (2): $H \Rightarrow P$
3) Given: H

$\boxed{(H \wedge P) \Rightarrow R}$ $\boxed{H \Rightarrow P}$ $\boxed{H}$

Exercises (5) – Solution

Let's consider the same example of Paul and Rachel.

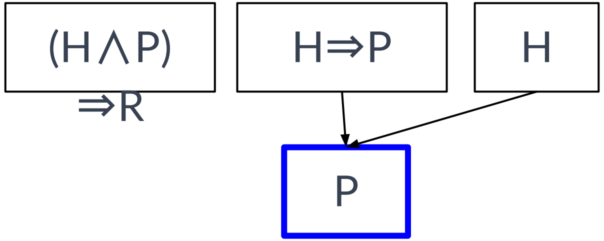
Premise:

- If Paul is happy and paints a picture ($H \land P$), then Rachel is happy (R)
- If Paul is happy (H) then he paints a picture (P)

Given Information: Paul is happy (H)

Inference: Conclude whether Rachel is happy (R) based on the given information.

1) The premise (1): $(H \land P) \Rightarrow R$
2) The premise (2): $H \Rightarrow P$
3) Given: H
4) Modus Ponens (2) and (3): **P**



$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

Exercises (5) – Solution

Let's consider the same example of Paul and Rachel.

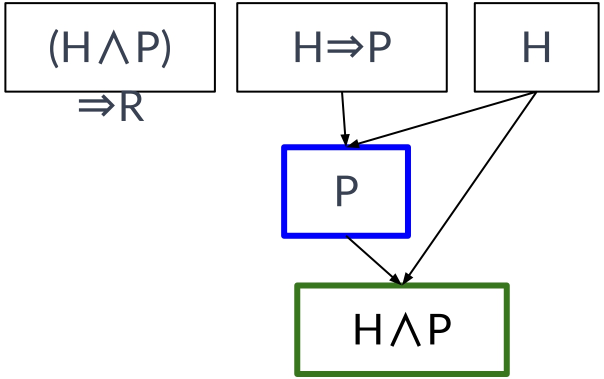
Premise:

- If Paul is happy and paints a picture (H∧P), then Rachel is happy (R)
- If Paul is happy (H) then he paints a picture (P)

Given Information: Paul is happy (H)

Inference: Conclude whether Rachel is happy (R) based on the given information.

1) The premise (1): (H∧P)⇒R
2) The premise (2): H⇒P
3) Given: H
4) Modus Ponens (2) and (3): **P**
5) And-introduction (3) and (4): **H∧P**



$$\frac{\alpha, \beta}{\alpha \wedge \beta}$$

Exercises (5) – Solution

Let’s consider the same example of Paul and Rachel.

Premise:

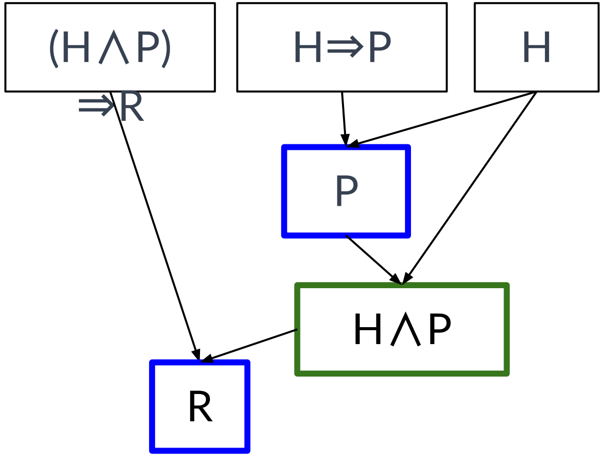
- If Paul is happy and paints a picture (H∧P), then Rachel is happy (R)
- If Paul is happy (H) then he paints a picture (P)

Given Information: Paul is happy (H)

Inference: Conclude whether Rachel is happy (R) based on the given information.

1) The premise (1): (H∧P)⇒R
2) The premise (2): H⇒P
3) Given: H
4) Modus Ponens (2) and (3): **P**
5) And-introduction (3) and (4): **H∧P**
6) Modus Ponens (1) and (5): **R**

> Rachel is Happy!



$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

Proofs forwards and backwards

Forwards Proof:

- Starts from premises and uses logical rules of inference to derive a conclusion.
- Shows that if certain conditions hold, then a particular conclusion can be derived

Backwards Proof:

- Starts from the conclusion and works backward, showing that the conclusion follows logically from certain assumptions.
- Shows that the conclusion is a logical consequence of certain conditions.

Proofs forwards and backwards - backward example

Prove that **$\neg R$** follows from **$P \land S$**, **$Q \Rightarrow \neg R$**, and **$\neg S \lor Q$**

1. $P \land S$ (given)
2. $Q \Rightarrow \neg R$ (given)
3. $\neg S \lor Q$ (given)
4.
5.
6.
7. $\neg R$

Proofs forwards and backwards - backward example

Prove that **¬R** follows from **P∧ S**, **Q ⇒¬R**, and **¬S** ∨ Q

1. P∧ S (given)
2. Q ⇒¬R (given)
3. ¬S ∨ Q (given)
4.
5.
6. Q
7. ¬R **(modus ponens: 6, 2)**

$$\boxed{\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}}$$

Proofs forwards and backwards - backward example

Prove that **<u>¬R</u>** follows from **<u>P∧ S</u>**, **<u>Q ⇒¬R</u>**, and **<u>¬S</u>** <u>∨ Q</u>

1. P∧ S (given)
2. Q ⇒¬R (given)
3. ¬S ∨ Q (given)
4.
5. ¬¬S
6. Q (or-elimination: 3,5)
7. ¬R **(modus ponens: 6, 2)**

$$\boxed{\frac{\alpha \vee \beta, \neg\alpha}{\beta}}$$

Proofs forwards and backwards - backward example

Prove that **¬R** follows from **P∧ S**, **Q ⇒¬R**, and **¬S** ∨ Q

1. P∧ S (given)
2. Q ⇒¬R (given)
3. ¬S ∨ Q (given)
4. S
5. ¬¬S (double negation: 4)
6. Q (or-elimination: 3,5)
7. ¬R **(modus ponens: 6, 2)**

$$\boxed{\frac{\neg\neg\alpha}{\alpha}}$$

Proofs forwards and backwards - backward example

Prove that **$\neg R$** follows from **$P \wedge S$**, **$Q \Rightarrow \neg R$**, and **$\neg S \vee Q$**

1. $P \wedge S$ (given)
2. $Q \Rightarrow \neg R$ (given)
3. $\neg S \vee Q$ (given)
4. S (and-elimination: 1)
5. $\neg\neg S$ (double negation: 4)
6. Q (or-elimination: 3,5)
7. $\neg R$ **(modus ponens: 6, 2)**

$$\boxed{\frac{\alpha \wedge \beta}{\alpha}}$$

Usually, if the primary goal is to establish the truth of a specific known conclusion, a backward proof might be the most effective choice.

Useful links

Truth Table Generator (by the university of Stanford): https://web.stanford.edu/class/cs103/tools/truth-table-tool/

Propositional logic test on Wolfram demonstration projects (free online resource): https://demonstrations.wolfram.com/PropositionalLogicTest/