Propositional Logic

Russell and Norvig:
Chapter 7, Sections 7.1–7.4

Slides by Jean-Claude Latombe, from an introductory AI course given at Stanford University Winter 2003
Used (and adapted) with permission.

Important Concepts in AI

- The Representation of Knowledge about the world
- The Reasoning Process to make use of it

Types of Agents

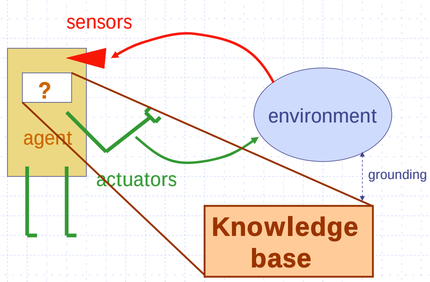
- Reflex Agent
  - Dumb luck
- Problem-solving Agent
  - Specific and inflexible
- Knowledge-based agent
  - General and flexible

Partially Seen Environments

- Knowledge-based Agents can combine
  - General Knowledge
  - Current Percepts

  To infer hidden aspects!

Knowledge-Based Agent



Types of Knowledge

- Procedural, e.g.: functions
  Such knowledge can only be used in one way -- by executing it

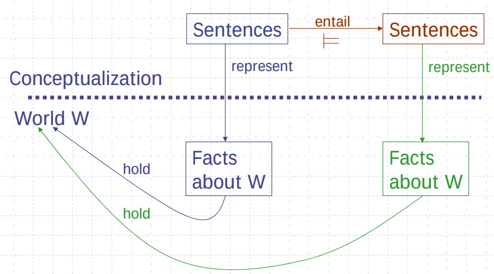
- Declarative, e.g.: constraints
  It can be used to perform many different sorts of inferences

Logic

Logic is a declarative language to:

- Assert sentences representing facts that hold in a world W
  (these sentences are given the value true)
- Deduce the true/false values to sentences representing other aspects of W

Connection World-Representation



Examples of Logics

- Propositional calculus
  A ∧ B ⇒ C
- First-order predicate calculus
  (∀x)(∃y) Mother(y,x)
- Logic of Belief
  B(John,Father(Zeus,Cronus))

Symbols of PL

- Connectives: ¬, ∧, ∨, ⇒
- Propositional symbols, e.g., P, Q, R, …
- *True*, *False*

Syntax of PL

- sentence → atomic sentence | complex sentence
- atomic sentence → Propositional symbol, *True*, *False*
- Complex sentence → ¬sentence
  | (sentence ∧ sentence)
  | (sentence ∨ sentence)
  | (sentence ⇒ sentence)

Syntax of PL

- sentence → atomic sentence | complex sentence
- atomic sentence → Propositional symbol, *True*, *False*
- Complex sentence → ¬sentence
  | (sentence ∧ sentence)
  | (sentence ∨ sentence)
  | (sentence ⇒ sentence)

- Examples:
  - ((P ∧ Q) ⇒ R)
  - (A ⇒ B) ∨ (¬C)
- Counter examples:
  - (A ∧ ⇒ R)
  - (A B) ∨ (¬C)

Order of Precedence

- $\neg \quad \wedge \quad \vee \quad \Rightarrow$
- Examples:
  - $\neg A \vee B \Rightarrow C$ is equivalent to $((\neg A)\vee B)\Rightarrow C$
  - $A \Rightarrow B \Rightarrow C$ is incorrect

    $(A \Rightarrow B) \Rightarrow C$
    $A \Rightarrow (B \Rightarrow C)$

Model

- Assignment of a truth value – true or false – to every atomic sentence
- Examples:
  - Let A, B, C, and D be the propositional symbols
  - m = {A=true, B=false, C=false, D=true} is a model
  - m' = {A=true, B=false, C=false} is not a model
- With n propositional symbols, one can define 2^n models

What Worlds Does a Model Represent?

A model represents any world in which a fact represented by a proposition A having the value *True* holds and a fact represented by a proposition B having the value *False* does not hold

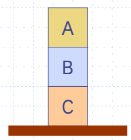
A model represents infinitely many worlds

Compare!

prop.symb.

- BLOCK(A), BLOCK(B), BLOCK(C)
- ON(A,B), ON(B,C), ONTABLE(C)
- ON(A,B) $\wedge$ ON(B,C) $\Rightarrow$ ABOVE(A,C)

→ ABOVE(A,C)



- HUMAN(A), HUMAN(B), HUMAN(C)
- CHILD(A,B), CHILD(B,C), BLOND(C)
- CHILD(A,B) $\wedge$ CHILD(B,C) $\Rightarrow$ GRAND-CHILD(A,C)

→ GRAND-CHILD(A,C)

Semantics of PL

- It specifies how to determine the truth value of any sentence in a model m
- The truth value of *True* is *True*
- The truth value of *False* is *False*
- The truth value of each atomic sentence is given by m
- The truth value of every other sentence is obtained recursively by using truth tables

Truth Tables

A	B	$\neg$ A	A $\wedge$ B	A $\vee$ B	A $\Rightarrow$ B
True	*True*	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*

Truth Tables

A	B	¬ A	A ∧ B	A ∨ B	A ⇒ B
True	*True*	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*

Truth Tables

A	B	¬ A	A ∧ B	A ∨ B	A ⇒ B
True	*True*	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*

About ⇒

- ODD(5) ⇒ CAPITAL(Japan,Tokyo)
- EVEN(5) ⇒ SMART(Sam)
- Read A ⇒ B as:
  "If A IS *True*, then I claim that B is *True*, otherwise I make no claim."

Example

Model: A=*True*, B=*False*, C=*False*, D=*True*

$(\neg A \lor B \Rightarrow C) \Rightarrow D \land A$

F

F

T   T

T

Definition: If a sentence s is true in a model m, then m is said to be a model of s

A Small Knowledge Base

1. Battery-OK $\land$ Bulbs-OK $\Rightarrow$ Headlights-Work
2. Battery-OK $\land$ Starter-OK $\land$ $\neg$Empty-Gas-Tank $\Rightarrow$ Engine-Starts
3. Engine-Starts $\land$ $\neg$Flat-Tire $\Rightarrow$ Car-OK
4. Headlights-Work
5. $\neg$Car-OK

Sentences 1, 2, and 3 → Background knowledge

Sentences 4 and 5 → Observed knowledge

Model of a KB

- Let KB be a set of sentences
- A model m is a model of KB iff it is a model of all sentences in KB, that is, all sentences in KB are true in m

Satisfiability of a KB

A KB is satisfiable iff it admits at least one model; otherwise it is unsatisfiable

KB1 = {P, ¬Q∧R} is satisfiable

KB2 = {¬P∨P} is satisfiable

valid sentence or tautology

KB3 = {P, ¬P} is unsatisfiable

Logical Entailment

- KB : set of sentences
- α : arbitrary sentence
- KB entails α – written KB ⊨ α – iff every model of KB is also a model of α

Logical Entailment

- KB : set of sentences
- α : arbitrary sentence
- KB entails α – written KB ⊨ α – iff every model of KB is also a model of α
- Alternatively, KB ⊨ α iff
  - {KB,¬α} is unsatisfiable
  - KB ⇒ α is valid

Logical Equivalence

- Two sentences α and β are logically equivalent – written $\alpha \equiv \beta$ -- iff they have the same models, i.e.:

  $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

Logical Equivalence

- Two sentences α and β are logically equivalent – written $\alpha \equiv \beta$ -- iff they have the same models, i.e.:

  $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
- Examples:
  - $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$
  - $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
  - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
  - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$

Logical Equivalence

- Two sentences α and β are logically equivalent – written $\alpha \equiv \beta$ -- iff they have the same models, i.e.:

  $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
- Examples:
  - $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$
  - $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
  - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
  - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
- One can always replace a sentence by an equivalent one in a KB

Inference Rule

- An inference rule $\{\xi, \psi\} \vdash \varphi$ consists of 2 sentence patterns ξ and ψ called the conditions and one sentence pattern φ called the conclusion

Inference Rule

- An inference rule $\{\xi, \psi\} \vdash \varphi$ consists of 2 sentence patterns ξ and ψ called the conditions and one sentence pattern φ called the conclusion
- If ξ and ψ match two sentences of KB then the corresponding φ can be inferred according to the rule

Example: Modus Ponens

$$\{\, \alpha \Rightarrow \beta \,,\, \alpha \,\} \ \vdash \ \beta$$

$$\{\xi, \psi\} \vdash \varphi$$

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

$$\{\xi, \psi\} \vdash \varphi$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

$$\{\xi, \psi\} \vdash \varphi$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

$$\{\xi, \psi\} \vdash \varphi$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK

Example: Modus Ponens

$$\{\ \alpha \Rightarrow \beta\ ,\ \alpha\ \}\ \vdash\ \beta$$

$$\{\xi,\ \psi\} \vdash \varphi$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK
Headlights-Work

Example: Modus Tolens

$$\{\ \alpha \Rightarrow \beta\ ,\ \neg\beta\ \}\ \vdash\ \neg\alpha$$

Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
¬Car-OK

Example: Modus Tolens

$$\{\ \alpha \Rightarrow \beta\ ,\ \neg\beta\ \}\ \vdash\ \neg\alpha$$

Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
¬Car-OK
¬(Engine-Starts ∧ ¬Flat-Tire)

Example: Modus Tolens

$\{ \alpha \Rightarrow \beta , \neg\beta \} \vdash \neg\alpha$

Engine-Starts $\wedge$ $\neg$Flat-Tire $\Rightarrow$ Car-OK
$\neg$Car-OK
$\neg$(Engine-Starts $\wedge$ $\neg$Flat-Tire) $\equiv$ $\neg$Engine-Starts $\vee$ Flat-Tire

Other Examples

- $\{\alpha,\beta\} \vdash \alpha \wedge \beta$
- $\{\alpha \wedge \beta,.\} \vdash \alpha$
- $\{\alpha \wedge \beta,.\} \vdash \beta$
- Etc ...

Inference

- I: Set of inference rules
- KB: Set of sentences
- Inference is the process of applying successive inference rules from I to KB, each rule adding its conclusion to KB

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. ¬Engine-Starts ∨ Flat-Tire ← (3+8)

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. ¬Engine-Starts ∨ Flat-Tire ← (3+8) ≡ Engine-Starts ⇒ Flat-Tire

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. Engine-Starts ⇒ Flat-Tire ← (3+8)

Example

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. Engine-Starts ⇒ Flat-Tire ← (3+8)
13. Flat-Tire ← (11+12)

Soundness

- An inference rule is sound if it generates only entailed sentences

Soundness

- An inference rule is sound if it generates only entailed sentences
- All inference rules previously given are sound, e.g.:
  modus ponens: $\{\alpha \Rightarrow \beta , \alpha\} \vdash \beta$

Soundness

- An inference rule is sound if it generates only entailed sentences
- All inference rules previously given are sound, e.g.:
  modus ponens: $\{\alpha \Rightarrow \beta , \alpha\} \vdash \beta$
- The following rule:
  $\{\alpha \vee \beta , .\} \vdash \neg\alpha \vee \neg\beta$
  is unsound, which does not mean it is useless

Completeness

- A set of inference rules is complete if every entailed sentences can be obtained by applying some finite succession of these rules
- Modus ponens alone is not complete, e.g.:
  from $A \Rightarrow B$ and $\neg B$, we cannot get $\neg A$
  (needed Modus Tolens for that)

Proof

The proof of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound inference rules

Proof

The proof of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound inference rules

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. Engine-Starts ⇒ Flat-Tire ← (3+8)
13. Flat-Tire ← (11+12)

⇒ Connective symbol (implication)

⊨ Logical entailment

KB ⊨ α iff KB ⇒ α is valid

⊢ Inference

⊢ ~ ⊨ if ⊢ sound and complete

Summary

- Knowledge representation
- Propositional Logic
- Truth tables
- Model of a KB
- Satisfiability of a KB
- Logical entailment
- Inference rules
- Proof