DD2380 Artificial Intelligence

Lecture 9: Logic

Jana Tumova

Motivation: Reasoning

Let us assume that there are five houses of different colors next to each other on the same road. In each house lives a man of a different nationality. Every man has his favorite drink, his favorite brand of cigarettes, and keeps pets of a particular kind.
The Englishman lives in the red house. The Swede keeps dogs. The Dane drinks tea. The green house is just to the left of the white one. The owner of the green house drinks coffee. The Pall Mall smoker keeps birds. The owner of the yellow house smokes Dunhills. The man in the center house drinks milk. The Norwegian lives in the first house. The Blend smoker has a neighbor who keeps cats. The man who smokes Blue Masters drinks beer. The man who keeps horses lives next to the Dunhill smoker. The German smokes Prince. The Norwegian lives next to the blue house. The Blend smoker has a neighbor who drinks water.
Who keeps fish?

Challenges

- How do we *represent* the knowledge about the world?
- How do we *infer* new knowledge?
- How do we *use* the knowledge to deduce what to do?
    - Next lectures on planning

We look into AI as the process of reasoning operating on internal representation of knowledge: knowledge-based agents.

Knowledge-based Agent

Stores knowledge in a *knowledge base*

- A set of *sentences* in a *knowledge representation language*
- Adding new sentences to the knowledge base through TELL
- Asking the knowledge base what is known/what is the next action through ASK
- Both ASK and TELL can involve deriving new knowledge

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*))
  $t \leftarrow t + 1$
  return *action*

Today's Lecture

- *Logic*: a way to represent sentences in a knowledge base
- *Logical inference*: a way to derive new knowledge

Propositional logic

- Basic building block: proposition symbols that are either *true* or *false*
  - $Rainy, Sunny, Wet, P, Q, R \ldots$
- Logical operators: $\neg, \wedge, \vee, \Rightarrow, \iff$
- Sentences: also *true* or *false*
  - P
  - $P \vee \neg P$
  - $P \iff Q \vee R$
  - $Q \Rightarrow \neg R$
  - $(P \iff Q) \iff R, \ldots$

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, *m satisfies* the sentence, *m* is a *model* of the sentence.
- Is it true in all models?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.
- Is it true in all models?
  No, but it is true in some. It is *satisfiable*.

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.
- Is it true in all models?
  No, but it is true in some. It is *satisfiable*.
- How about $Rainy \lor \neg Rainy$?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.
- Is it true in all models?
  No, but it is true in some. It is *satisfiable*.
- How about $Rainy \vee \neg Rainy$?
  It is true in all models, it is *valid* (a *tautology*).

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.
- Is it true in all models?
  No, but it is true in some. It is *satisfiable*.
- How about $Rainy \lor \neg Rainy$?
  It is true in all models, it is *valid* (a *tautology*).
- How about $Wet \land \neg Wet$?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.
- Is it true in all models?
  No, but it is true in some. It is *satisfiable*.
- How about $Rainy \vee \neg Rainy$?
  It is true in all models, it is *valid* (a *tautology*).
- How about $Wet \wedge \neg Wet$?
  It is false in all models, it is *unsatisfiable* (a *contradiction*).

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $Rainy \iff Wet$ true or false?
  It is true, m *satisfies* the sentence, m is a *model* of the sentence.
- Is it true in all models?
  No, but it is true in some. It is *satisfiable*.
- How about $Rainy \vee \neg Rainy$?
  It is true in all models, it is *valid* (a *tautology*).
- How about $Wet \wedge \neg Wet$?
  It is false in all models, it is *unsatisfiable* (a *contradiction*).
- How about $(\neg P \vee Q) \Rightarrow (P \Rightarrow Q)$? For which models is it satisfied?

Propositional logic: Truth table

P	Q	$\neg P$	$\neg P \vee Q$	$P \Rightarrow Q$	$(\neg P \vee Q) \Rightarrow (P \Rightarrow Q)$
True	*True*	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*False*	*True*
False	*True*	*True*	*True*	*True*	*True*
False	*False*	*True*	*True*	*True*	*True*

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is α : $(Rainy \lor Wet) \Rightarrow Wet$ true or false?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is α : $(Rainy \vee Wet) \Rightarrow Wet$ true or false?
- What are the models of α?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is α : $(Rainy \vee Wet) \Rightarrow Wet$ true or false?
- What are the models of α?

$$\begin{aligned} M(\alpha) = \big\{ & \{Rainy = true, Wet = true\}, \\ & \{Rainy = false, Wet = true\}, \\ & \{Rainy = false, Wet = false\}\big\} \end{aligned}$$

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $\alpha : (Rainy \vee Wet) \Rightarrow Wet$ true or false?
- What are the models of α?

$$M(\alpha) = \big\{\{Rainy = true, Wet = true\},$$
$$\{Rainy = false, Wet = true\},$$
$$\{Rainy = false, Wet = false\}\big\}$$

- What are the models of $\beta : Rainy \wedge Wet$?

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is α : $(Rainy \vee Wet) \Rightarrow Wet$ true or false?
- What are the models of α?

$$M(\alpha) = \big\{\{Rainy = true, Wet = true\},\\ \{Rainy = false, Wet = true\},\\ \{Rainy = false, Wet = false\}\big\}$$

- What are the models of β : $Rainy \wedge Wet$?

$$M(\beta) = \big\{\{Rainy = true, Wet = true\}\big\}$$

Q

Model (a possible world) $m = \{Rainy = true, Wet = true\}$

- Is $\alpha : (Rainy \vee Wet) \Rightarrow Wet$ true or false?
- What are the models of α?

$$M(\alpha) = \big\{\{Rainy = true, Wet = true\},\\ \{Rainy = false, Wet = true\},\\ \{Rainy = false, Wet = false\}\big\}$$

- What are the models of $\beta : Rainy \wedge Wet$?

$$M(\beta) = \big\{\{Rainy = true, Wet = true\}\big\}$$

$M(\beta) \subseteq M(\alpha)$, α *logically follows* from β, β *entails* α,

$$\beta \models \alpha.$$

Q

- What are the models of $\alpha : \neg(\mathit{Rainy} \Rightarrow \mathit{Wet})$?

Q

- What are the models of $\alpha : \neg(\mathit{Rainy} \Rightarrow \mathit{Wet})$?

$$M(\alpha) = \big\{\{\mathit{Rainy} = \mathit{true}, \mathit{Wet} = \mathit{false}\}\big\}$$

Q

- What are the models of $\alpha : \neg(\mathit{Rainy} \Rightarrow \mathit{Wet})$?

$$M(\alpha) = \big\{\{\mathit{Rainy} = \mathit{true}, \mathit{Wet} = \mathit{false}\}\big\}$$

- What are the models of $\beta : \neg(\neg \mathit{Rainy} \vee \mathit{Wet})$?

Q

- What are the models of $\alpha : \neg(\mathit{Rainy} \Rightarrow \mathit{Wet})$?

$$M(\alpha) = \big\{\{\mathit{Rainy} = \mathit{true}, \mathit{Wet} = \mathit{false}\}\big\}$$

- What are the models of $\beta : \neg(\neg \mathit{Rainy} \vee \mathit{Wet})$?

$$M(\beta) = \big\{\{\mathit{Rainy} = \mathit{true}, \mathit{Wet} = \mathit{false}\}\big\}$$

Q

- What are the models of $\alpha : \neg(\mathit{Rainy} \Rightarrow \mathit{Wet})$?

$$M(\alpha) = \big\{\{\mathit{Rainy} = \mathit{true}, \mathit{Wet} = \mathit{false}\}\big\}$$

- What are the models of $\beta : \neg(\neg\mathit{Rainy} \vee \mathit{Wet})$?

$$M(\beta) = \big\{\{\mathit{Rainy} = \mathit{true}, \mathit{Wet} = \mathit{false}\}\big\}$$

$M(\alpha) = M(\beta)$, α and β are *logically equivalent*.

Any Logic

- *Syntax*: says what are well-defined sentences
- *Semantics*: says what is the meaning of the sentences
- *Model*: a possible world, where a sentence α can be true or false
- *Satisfaction*: *m satisfies* α, *m* is a *model of* α
- *Set of all models of* α: *M*(α)
- *Entailment*: Sentence α *entails* β, β logically follows from α :

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

- The notion of model, satisfaction, entailment can be extended to a set of sentences (knowledge base) in the expected way.

Equivalence, validity, satisfiability, contradiction

- Validity: α is valid (a tautology) if it is true in *all* models.
- Unsatisfiability: α is unsatisfiable (a contradiction) if it is false in *all* models.
- Satisfiability: α is satisfiable if it is true in *some* models.
- Logical equivalence: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

Any Logic

Grounding issue: How do we make a connection betweel logical reasoning process and the real environment in which the agent exists? How do we know that KB is true in real world?

Propositional logic: Syntax

$$
\begin{aligned}
\mathit{Sentence} &\longrightarrow \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} &\longrightarrow \mathit{True} \mid \mathit{False} \mid A \mid B \mid P \mid Q \mid \mathit{Rain} \mid \mathit{Sun} \ldots \\
\mathit{ComplexSentence} &\longrightarrow (\mathit{Sentence}) \mid [\mathit{Sentence}] \\
&\quad \mid \neg \mathit{Sentence} \\
&\quad \mid \mathit{Sentence} \land \mathit{Sentence} \mid \mathit{Sentence} \lor \mathit{Sentence} \\
&\quad \mid \mathit{Sentence} \Rightarrow \mathit{Sentence} \mid \mathit{Sentence} \iff \mathit{Sentence}
\end{aligned}
$$

Operator precedence: $\neg, \land, \lor, \Rightarrow, \iff$

Propositional logic: Semantics

- *True* is true in every model and *False* is false in every model
- The truth value of any other proposition symbol must be specified in the model.

Consider a model m

- $\neg P$ is true in m iff P is false in m
- $P \wedge Q$ is true in m iff both P and Q are true in m
- $P \vee Q$ is true in m iff P or Q is true in m
- $P \Rightarrow Q$ is true in m unless P is true and Q is false in m
- $P \iff Q$ is true in m iff P and Q are both true or both false in m.

Prop. Logic: Some Interesting Equivalences

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

A cake example

Knowledge base KB:

- α_1 : $BakedCake$.
- α_2 : $\neg AteCake$.
- α_3 : $BakedCake \Rightarrow HaveCake \vee AteCake$.
- α_4 : $\neg HaveCake \Rightarrow AteCake \vee \neg BakedCake$.

- Can I conclude *HaveCake*? How?
- Can I add it to the knowledge base?

Model checking

- Make a big truth table and enumerate all truth values of all proposition symbols
- See on which lines all sentences from the knowledge base are true; these are models of the knowledge base
- See whether for all those models the query is true

- Complexity!

Logical Inference

- α_1 : *BakedCake*.
- α_2 : $\neg$*AteCake*.
- α_3 : *BakedCake* $\Rightarrow$ *HaveCake* $\vee$ *AteCake*.
- α_4 : $\neg$*HaveCake* $\Rightarrow$ *AteCake* $\vee$ $\neg$*BakedCake*.

- From α_4, we get $\neg$*AteCake* $\wedge$ *BakedCake* $\Rightarrow$ *HaveCake*.
- From that and from α_1 and α_2, we get *HaveCake*.

Inference rules

- *Modus Ponens*:

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

- *And-Elimination*:

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

- *All equivalences, bidirectionally*
    - $(\alpha \iff \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$

$$\frac{\alpha \iff \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

    - $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$
    - $\ldots$

Logical Inference

- α_1 : *BakedCake*.
- α_2 : $\neg$*AteCake*.
- α_3 : *BakedCake* $\Rightarrow$ *HaveCake* $\vee$ *AteCake*.
- α_4 : $\neg$*HaveCake* $\Rightarrow$ *AteCake* $\vee$ $\neg$*BakedCake*.

- From α_4, we get $\neg$*AteCake* $\wedge$ *BakedCake* $\Rightarrow$ *HaveCake* (by contraposition equivalence $\frac{\alpha \Rightarrow \beta}{\neg\beta \Longrightarrow \neg\alpha}$ and De Morgan $\frac{\neg(\alpha \vee \beta)}{\neg\alpha \wedge \neg\beta}$ and double-negation elimination $\frac{\neg(\neg\alpha)}{\alpha}$).
- From that and from α_1 and α_2, we get *HaveCake* (by Modus Ponens $\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$).

Applying inference rules

- Monotonous: knowledge is increasing
- Sound (if a sentence is inferred, it is entailed by KB), but not complete (if a sentence is entailed, it might not be inferred)
    - Cure: resolution
- Can be automazed with search: the problem is defined as initial state, actions, result function, goal

Resolution

1. Transform knowledge base to CNF:
   - Literal: a proposition or its negation
   - Clause: a disjunction of literals
   - Sentence: a conjunction of clauses
2. Apply resolution rule:

$$\frac{\ell_1 \vee \ldots \vee \ell_k,\ \ m_1 \vee \ldots \ \vee m_n}{\ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n},$$

   where each ℓ, m is a literal, and ℓ_i and m_j are complementary literals (i.e. one is negation of the other).

A cake example

Knowledge base KB:

- α_1 : $\mathit{BakedCake}$.
- α_2 : $\neg \mathit{AteCake}$.
- ~~α_3 : $\mathit{BakedCake} \Rightarrow \mathit{HaveCake} \vee \mathit{AteCake}$.~~
- ~~α_4 : $\neg \mathit{HaveCake} \Rightarrow \mathit{AteCake} \vee \neg \mathit{BakedCake}$.~~

A cake example

Knowledge base KB:

- α_1 : $\mathit{BakedCake}$.
- α_2 : $\neg \mathit{AteCake}$.
- ~~α_3 : $\mathit{BakedCake} \Rightarrow \mathit{HaveCake} \vee \mathit{AteCake}$.~~
  α_3 : $\neg \mathit{BakedCake} \vee \mathit{HaveCake} \vee \mathit{AteCake}$.
- ~~α_4 : $\neg \mathit{HaveCake} \Rightarrow \mathit{AteCake} \vee \neg \mathit{BakedCake}$.~~

A cake example

Knowledge base KB:

- α_1 : $\mathit{BakedCake}$.
- α_2 : $\neg \mathit{AteCake}$.
- ~~α_3 : $\mathit{BakedCake} \Rightarrow \mathit{HaveCake} \vee \mathit{AteCake}$.~~

  α_3 : $\neg \mathit{BakedCake} \vee \mathit{HaveCake} \vee \mathit{AteCake}$.
- ~~α_4 : $\neg \mathit{HaveCake} \Rightarrow \mathit{AteCake} \vee \neg \mathit{BakedCake}$.~~

  α_4 : $\mathit{HaveCake} \vee \mathit{AteCake} \vee \neg \mathit{BakedCake}$.

A cake example

Knowledge base KB:

- α_1 : $BakedCake$.
- α_2 : $\neg AteCake$.
- $\cancel{\alpha_3 : BakedCake \Rightarrow HaveCake \vee AteCake.}$
  α_3 : $\neg BakedCake \vee HaveCake \vee AteCake$.
- $\cancel{\alpha_4 : \neg HaveCake \Rightarrow AteCake \vee \neg BakedCake.}$
  α_4 : $HaveCake \vee AteCake \vee \neg BakedCake$.

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \;\; m_1 \vee \ldots \;\; \vee m_n}{\ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n},$$

A cake example

Knowledge base KB:

- α_1 : $BakedCake$.
- α_2 : $\neg AteCake$.
- $\cancel{\alpha_3 : BakedCake \Rightarrow HaveCake \vee AteCake.}$
  α_3 : $\neg BakedCake \vee HaveCake \vee AteCake$.
- $\cancel{\alpha_4 : \neg HaveCake \Rightarrow AteCake \vee \neg BakedCake.}$
  α_4 : $HaveCake \vee AteCake \vee \neg BakedCake$.

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \;\; m_1 \vee \ldots \;\; \vee m_n}{\ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n},$$

$$\frac{\alpha_1 : BakedCake, \alpha_4 : HaveCake \vee AteCake \vee \neg BakedCake}{\alpha_5 : HaveCake \vee AteCake},$$

Logical Inference via Resolution

- Translation to CNF sound and complete
- Applying resolution rule sound and complete

The Einstein puzzle in propositional logic

- Proposition symbols:
  $E_{H_1}, E_{H_2}, \ldots, E_{green}, E_{blue} \ldots E_{dog}, E_{fish} \ldots E_{water},$
  $E_{tea} \ldots E_{dunhills}, E_{pallmall} \ldots, S_{H_1}, S_{H_2}, \ldots, S_{green}, S_{blue} \ldots S_{dog},$
  $S_{fish} \ldots S_{water}, S_{tea} \ldots S_{dunhills}, S_{pallmall} \ldots$
- Knowledge base:
  - E_{red}.
  - $(E_{pallmall} \wedge E_{birds}) \vee (S_{pallmall} \wedge S_{birds}) \vee (N_{pallmall} \wedge N_{birds}) \vee (G_{pallmall} \wedge G_{birds}) \vee (D_{pallmall} \wedge D_{birds})$.
  - $E_{pallmall} \Rightarrow \neg S_{pallmall} \wedge \neg B_{pallmall} \wedge \neg G_{pallmall} \wedge \neg D_{pallmall}, \ldots$.
  - $\ldots$

The Einstein puzzle in propositional logic

- Proposition symbols:
  $E_{H_1}, E_{H_2}, \ldots, E_{green}, E_{blue} \ldots E_{dog}, E_{fish} \ldots E_{water},$
  $E_{tea} \ldots E_{dunhills}, E_{pallmall} \ldots, S_{H_1}, S_{H_2}, \ldots, S_{green}, S_{blue} \ldots S_{dog},$
  $S_{fish} \ldots S_{water}, S_{tea} \ldots S_{dunhills}, S_{pallmall} \ldots$
- Knowledge base:
  - E_{red}.
  - $(E_{pallmall} \wedge E_{birds}) \vee (S_{pallmall} \wedge S_{birds}) \vee (N_{pallmall} \wedge N_{birds}) \vee (G_{pallmall} \wedge G_{birds}) \vee (D_{pallmall} \wedge D_{birds})$.
  - $E_{pallmall} \Rightarrow \neg S_{pallmall} \wedge \neg B_{pallmall} \wedge \neg G_{pallmall} \wedge \neg D_{pallmall}, \ldots$.
  - $\ldots$

A LOT of sentences in KB!

The Einstein puzzle in first order logic

- Constants: $E, S, \ldots, H_1, H_2, \ldots, dog, fish \ldots$
- Variables: $x, y, \ldots$
- Functions: $Color(H_1), Color(H_2), \ldots$
- Sentences: $Color(H_1) = red, Lives(E, red), \exists x. Has(x, dog) \ldots$
- Knowledge base:
  - $\cancel{E_{red}.}$
    $Lives(E, red).$
  - $\cancel{(E_{pallmall} \wedge E_{birds}) \vee (S_{pallmall} \wedge S_{birds}) \vee (N_{pallmall} \wedge N_{birds})}$
    $\cancel{\vee (G_{pallmall} \wedge G_{birds}) \vee (D_{pallmall} \wedge D_{birds}).}$
    $\forall x.\, Smokes(x, pallmall) \rightarrow Keeps(x, birds).$
  - $\cancel{E_{pallmall} \rightarrow \neg S_{pallmall} \wedge \neg B_{pallmall} \wedge \neg G_{pallmall} \wedge \neg D_{pallmall}, \ldots.}$
    $\forall x, y, z.\, Smokes(x, z) \wedge Smokes(y, z) \Rightarrow x = z.$
  - $\ldots$

First order logic

- Basic building blocks: terms (constants, variables, functions), predicates, atomic sentences
  - $E, S, dog, fish, x, y, Color(H_1), Plus(x, y) \ldots$
  - $Rainy, Loves, \ldots$
  - $Rainy, Loves(x, y), Loves(x, Mother(y)), Loves(Adam, y),$ $Plus(x, y) = 11, Plus(x, y) = z \ldots$
- Logical operators: $\neg, \wedge, \vee, \Rightarrow, \iff$
- Quantifiers: $\forall, \exists$
- Sentences: *true* or *false*
  - $Color(H_1) = red$
  - $\forall x. Color(x) = red \iff Lives(x, E)$
  - $\forall x. \exists y. Loves(x, y), \ldots$

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $Sibling(x, y)$ is an atomic sentence.

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $Sibling(x, y)$ is an atomic sentence.
- $\forall x.\, Loves(x, Raymond)$
- $\forall x \exists y.\, Loves(x, y)$

Q

What do the following say?

- $\forall x, y. \mathit{Sibling}(x, y) \iff \mathit{Sibling}(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $\mathit{Sibling}(x, y)$ is an atomic sentence.
- $\forall x. \mathit{Loves}(x, \mathit{Raymond})$ says that everybody loves Raymond.
- $\forall x \exists y. \mathit{Loves}(x, y)$ says that everybody loves somebody.

Q

What do the following say?

- $\forall x, y.\, \mathit{Sibling}(x, y) \iff \mathit{Sibling}(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $\mathit{Sibling}(x, y)$ is an atomic sentence.
- $\forall x.\, \mathit{Loves}(x, \mathit{Raymond})$ says that everybody loves Raymond.
- $\forall x \exists y.\, \mathit{Loves}(x, y)$ says that everybody loves somebody.
- $\forall x \exists y.\, \mathit{Loves}(x, \mathit{Mother}(y))$

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $Sibling(x, y)$ is an atomic sentence.
- $\forall x.\, Loves(x, Raymond)$ says that everybody loves Raymond.
- $\forall x \exists y.\, Loves(x, y)$ says that everybody loves somebody.
- $\forall x \exists y.\, Loves(x, Mother(y))$ says that everybody loves somebody's mother. *Loves* is a predicate, *Mother* is a function.

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $Sibling(x, y)$ is an atomic sentence.
- $\forall x.\, Loves(x, Raymond)$ says that everybody loves Raymond.
- $\forall x \exists y.\, Loves(x, y)$ says that everybody loves somebody.
- $\forall x \exists y.\, Loves(x, Mother(y))$ says that everybody loves somebody's mother. *Loves* is a predicate, *Mother* is a function.
- How do we say that everybody loves their own mother?

Q

What do the following say?

- $\forall x, y.\, \mathit{Sibling}(x, y) \iff \mathit{Sibling}(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $\mathit{Sibling}(x, y)$ is an atomic sentence.
- $\forall x.\, \mathit{Loves}(x, \mathit{Raymond})$ says that everybody loves Raymond.
- $\forall x \exists y.\, \mathit{Loves}(x, y)$ says that everybody loves somebody.
- $\forall x \exists y.\, \mathit{Loves}(x, \mathit{Mother}(y))$ says that everybody loves somebody's mother. *Loves* is a predicate, *Mother* is a function.
- How do we say that everybody loves their own mother? $\forall x.\, \mathit{Loves}(x, \mathit{Mother}(x))$.

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $Sibling(x, y)$ is an atomic sentence.
- $\forall x.\, Loves(x, Raymond)$ says that everybody loves Raymond.
- $\forall x \exists y.\, Loves(x, y)$ says that everybody loves somebody.
- $\forall x \exists y.\, Loves(x, Mother(y))$ says that everybody loves somebody's mother. *Loves* is a predicate, *Mother* is a function.
- How do we say that everybody loves their own mother? $\forall x.\, Loves(x, Mother(x))$.
- $\forall x \forall y.\, Plus(x, y) = Plus(y, x)$

Q

What do the following say?

- $\forall x, y.\, Sibling(x, y) \iff Sibling(y, x)$ says that siblinghood is a symmetric relationship. *Sibling* is a predicate, $Sibling(x, y)$ is an atomic sentence.
- $\forall x.\, Loves(x, Raymond)$ says that everybody loves Raymond.
- $\forall x \exists y.\, Loves(x, y)$ says that everybody loves somebody.
- $\forall x \exists y.\, Loves(x, Mother(y))$ says that everybody loves somebody's mother. *Loves* is a predicate, *Mother* is a function.
- How do we say that everybody loves their own mother? $\forall x.\, Loves(x, Mother(x))$.
- $\forall x \forall y.\, Plus(x, y) = Plus(y, x)$ says that addition of two numbers is commutative. *Plus* is a function.

First order logic: Syntax

$$
\begin{aligned}
\mathit{Term} &\longrightarrow \mathit{Constant} \mid \mathit{Variable} \mid \mathit{Function}(\mathit{Term}) \\
\mathit{Constant} &\longrightarrow A \mid X_1 \mid \mathit{ScroogeMcDuck} \mid \ldots \\
\mathit{Variable} &\longrightarrow a \mid x \mid \ldots \\
\mathit{Function} &\longrightarrow \mathit{Mother} \mid \mathit{Plus} \mid \ldots \\
\mathit{AtomicSentence} &\longrightarrow \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{Predicate} &\longrightarrow \mathit{True} \mid \mathit{False} \mid \mathit{Uncle} \mid \mathit{Member} \mid \mathit{After} \mid \ldots \\
\mathit{Sentence} &\longrightarrow \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{ComplexSentence} &\longrightarrow (\mathit{Sentence}) \mid [\mathit{Sentence}] \\
&\quad \mid \neg \mathit{Sentence} \mid \mathit{Sentence} \wedge \mathit{Sentence} \mid \mathit{Sentence} \vee \mathit{Sentence} \\
&\quad \mid \mathit{Sentence} \Rightarrow \mathit{Sentence} \mid \mathit{Sentence} \iff \mathit{Sentence} \\
&\quad \mid \mathit{Quantifier}\ \mathit{Variable}, \ldots .\ \mathit{Sentence} \\
\mathit{Quantifier} &\longrightarrow \forall \mid \exists
\end{aligned}
$$

Operator precedence: $\neg, \wedge, \vee, \Rightarrow, \iff$

First order logic: Semantics

- Domain: set of objects a model contains
- Interpretation: specifies which objects, relations, and functions are referred to by the constant, predicate, and function symbols
- Ground term: does not contain variables

Recap: Any Logic

- *Model*: a possible world, where a sentence α can be true or false
- *Satisfaction*: *m satisfies* α, *m* is a *model of* α
- *Set of all models of* α: $M(\alpha)$
- *Entailment*: Sentence α *entails* β, β logically follows from α :

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

- *Validity*: α is valid (a tautology) if it is true in *all* models.
- *Unsatisfiability*: α is unsatisfiable (a contradiction) if it is false in *all* models.
- *Satisfiability*: α is satisfiable if it is true in *some* models.
- *Logical equivalence*: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

Logical inference in first order logic

- Analogous as for the propositional logic
- Translation to CNF for first order logic still works (sound and complete)
- Resolution still works (sound and complete)

Substitution

- $SUBST(\{v/g\}, \alpha)$ is a result of substituting each occurence of variable v in sentence α by a *term* g.

- Q: What is $SUBST(\{x/2, y/3\}, Plus(x, y) = Plus(y, x))$?

Substitution

- $SUBST(\{v/g\}, \alpha)$ is a result of substituting each occurence of variable v in sentence α by a *term* g.

- Q: What is $SUBST(\{x/2, y/3\}, Plus(x, y) = Plus(y, x))$?

$$Plus(2, 3) = Plus(3, 2).$$

Generalized Modus Ponens

- Existentially instantiate to obtain only universally quantified sentences
- Forget the universal quantifiers

$$\frac{p_1', p_2', \ldots, p_n',\ (p_1 \wedge p_2 \wedge \ldots p_n \Rightarrow q)}{SUBST(\theta, q)},$$

where θ is a substitution such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$, and p_i, p_i' are atomic sentences, for all i.

Q

$$\frac{p'_1, p'_2, \ldots, p'_n,\ (p_1 \wedge p_2 \wedge \ldots p_n \Rightarrow q)}{SUBST(\theta, q)},$$

where θ is a substitution such that $SUBST(\theta, p'_i) = SUBST(\theta, p_i)$, and p_i, p'_i are atomic sentences, for all i.

How do we apply Generalized Modus Ponens on KB with

$Rich(Uncle(Louie)), Happy(Louie)$

$Rich(y) \wedge Happy(x) \Rightarrow Happy(y)$?

$$\frac{Rich(Uncle(Louie)), Happy(Louie) \quad (Rich(y) \wedge Happy(x) \Rightarrow Happy(y))}{Happy(Uncle(Louie))},$$

where $\theta = \{x/Louie, y/Uncle(Louie)\}$.

Resolution

Resolution rule:

$$\frac{\ell_1 \vee \ldots \vee \ell_k,\ \ m_1 \vee \ldots\ \vee m_n}{SUBST(\theta, \ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)},$$

where $UNIFY(l_i, \neg m_j) = \theta, SUBST(\theta, l_i) = SUBST(\theta, \neg m_j)$.

Q

Resolution rule:

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \;\; m_1 \vee \ldots \;\vee m_n}{SUBST(\theta, \ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)},$$

where $UNIFY(l_i, \neg m_j) = \theta, SUBST(\theta, l_i) = SUBST(\theta, \neg m_j)$.

How do we resolve
$Brothers(Louie, x) \vee Brothers(x, Dewey) \vee Brothers(x, Huey)$,
$\neg Brothers(Louie, Louie) \vee \neg Brothers(Dewey, Dewey)$?

Q

Resolution rule:

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \;\; m_1 \vee \ldots \;\; \vee m_n}{\mathit{SUBST}(\theta, \ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)},$$

where $\mathit{UNIFY}(l_i, \neg m_j) = \theta, \mathit{SUBST}(\theta, l_i) = \mathit{SUBST}(\theta, \neg m_j)$.

How do we resolve
$\mathit{Brothers}(\mathit{Louie}, x) \vee \mathit{Brothers}(x, \mathit{Dewey}) \vee \mathit{Brothers}(x, \mathit{Huey})$,
$\neg \mathit{Brothers}(\mathit{Louie}, \mathit{Louie}) \vee \neg \mathit{Brothers}(\mathit{Dewey}, \mathit{Dewey})$?

- Look at $\ell_1 \equiv \mathit{Brothers}(\mathit{Louie}, x)$ and $m_1 \equiv \neg \mathit{Brothers}(\mathit{Louie}, \mathit{Louie})$.
  $\mathit{UNIFY}(\ell_1, \neg m_1) =$
  $\mathit{UNIFY}(\mathit{Brothers}(\mathit{Louie}, x), \mathit{Brothers}(\mathit{Louie}, \mathit{Louie})) = \{x/\mathit{Louie}\}$.
- ℓ_1 and m_1 disappear, we need to do $\mathit{SUBST}(\{x/\mathit{Louie}\}, \ell_2 \vee \ell_3 \vee m_2)$.
- We get $\mathit{Brothers}(\mathit{Louie}, \mathit{Dewey}) \vee \mathit{Brothers}(\mathit{Louie}, \mathit{Huey}) \vee \neg \mathit{Brothers}(\mathit{Dewey}, \mathit{Dewey})$

Prolog

Try it out: `https://swish.swi-prolog.org`

Knowledge Engineering

1. Identify the task
   - Will the KB need to choose actions or just answer questions about the content of the environment? What are we able to sense?
2. Asssemble the relevant knowledge
   - What are the rules?
3. Decide on vocabulary of predicates, functions, and constants
   - Should some feature be a function or a predicate?
4. Encode general knowledge about the domain
   - Write down the axioms
5. Encode a description of a problem instance
   - Write down atomic sentences that are known to hold
6. Pose queries and get answers
   - Let the KB infer facts that we are interested in knowing
7. Debug, debug, debug

Logic Summary

- Support for representing knowledge and deriving new knowledge
- Propositional logic: simple, but unable to represent complex knowledge in a concise way
- First-order logic: more powerful logic, but designing a knowledge base is not a straightforward process
- Both: inference procedures allowing (automatic) deduction
- What we have not talked about: more exotic logic, logic programming,...