Models—Examples.

Environment

Language (Syntax)

- Constants: a,b,c,e
- Functions:
  - No function
- Predicates:
  - on: binary
  - above: binary
  - clear: unary
  - ontatable: unary
Models—Examples.

Language (syntax)

- Constants: a, b, c, e
- Predicates:
  - on (binary)
  - above (binary)
  - clear (unary)
  - ontable (unary)

A possible Model I₁ (semantics)

- \( D = \{ A, B, C, E \} \)
- \( \Phi(a) = A, \Phi(b) = B, \Phi(c) = C, \Phi(e) = E. \)
- \( \Psi(\text{on}) = \{(A,B),(B,C)\} \)
- \( \Psi(\text{above})=\{(A,B),(B,C),(A,C)\} \)
- \( \Psi(\text{clear})=\{A,E\} \)
- \( \Psi(\text{ontable})=\{C,E\} \)
Models—Examples.

Model $I_1$

- $D = \{A, B, C, E\}$
- $\Phi(a) = A, \Phi(b) = B, \Phi(c) = C, \Phi(e) = E$.
- $\Psi(on) = \{(A,B),(B,C)\}$
- $\Psi(above) = \{(A,B),(B,C),(A,C)\}$
- $\Psi(clear) = \{A,E\}$
- $\Psi(ontable) = \{C,E\}$
Models—Formulas true or false?

Model $I_1$

- $D = \{A, B, C, E\}$
- $\Phi(a) = A, \Phi(b) = B, \Phi(c) = C, \Phi(e) = E.$
- $\Psi(on) = \{(A,B),(B,C)\}$
- $\Psi(above) = \{(A,B),(B,C),(A,C)\}$
- $\Psi(clear) = \{A,E\}$
- $\Psi(ontable) = \{C,E\}$

$\forall X,Y. \text{on}(X,Y) \rightarrow \text{above}(X,Y)$

- $X=A, Y=B$. True
- $X=C, Y=A$. True
- ... 

$\forall X,Y. \text{above}(X,Y) \rightarrow \text{on}(X,Y)$

- $X=A, Y=B$. True
- $X=A, Y=A$. True
- $X=A, Y=C$. False
Models—Examples.

Model $I_1$

- $D = \{A, B, C, E\}$
- $\Phi(a) = A$, $\Phi(b) = B$, $\Phi(c) = C$, $\Phi(e) = E$.
- $\Psi(on) = \{(A, B), (B, C)\}$
- $\Psi(above) = \{(A, B), (B, C), (A, C)\}$
- $\Psi(clear) = \{A, E\}$
- $\Psi(ontable) = \{C, E\}$

$\forall X \exists Y. (clear(X) \lor on(Y, X))$

- $X = A$
- $X = C$, $Y = B$
- …

$\exists Y \forall X. (clear(X) \lor on(Y, X))$

- $Y = A$ ? No! ($X = C$)
- $Y = C$ ? No! ($X = B$)
- $Y = E$ ? No! ($X = B$)
- $Y = B$ ? No! ($X = B$)
KB — many models

KB

1. on(b,c)
2. clear(e)
Models

• Let our Knowledge base KB, consist of a set of formulas.

• We say that $I$ is a **model** of KB or that $I$ **satisfies** KB
  - If, every formula $f \in KB$ is true under $I$

• We write $I \models KB$ if $I$ satisfies KB, and $I \models f$ if $f$ is true under $I$. 
What’s Special About Models?

• When we write KB, we intend that the real world (i.e. our set theoretic abstraction of it) is one of its models.

• This means that every statement in KB is true in the real world.

• Note however, that not every thing true in the real world need be contained in KB. We might have only incomplete knowledge.
Models support reasoning.

- Suppose formula $f$ is not mentioned in $KB$, but is true in every model of $KB$; i.e.,
  \[ I \models KB \rightarrow I \models f. \]

- Then we say that $f$ is a **logical consequence** of $KB$ or that $KB$ **entails** $f$.

- Since the real world is a model of $KB$, $f$ must be true in the real world.

- This means that entailment is a way of finding new true facts that were not explicitly mentioned in $KB$.

??? *If $KB$ doesn’t entail $f$, is $f$ false in the real world?*
Logical Consequence Example

• \textsf{elephant}(clyde)
  • the individual denoted by the symbol \textit{clyde} in the set denoted by \textit{elephant} (has the property that it is an \textit{elephant}).

• \textsf{teacup}(cup)
  • \textit{cup} is a teacup.

• Note that in both cases a unary predicate specifies a set of individuals. Asserting a unary predicate to be true of a term means that the individual denoted by that term is in the specified set.
Logical Consequence Example

• $\forall X, Y. \text{elephant}(X) \land \text{teacup}(Y) \rightarrow \text{largerThan}(X, Y)$

  • For all pairs of individuals if the first is an elephant and the second is a teacup, then the pair of objects are related to each other by the $\text{largerThan}$ relation.

  • For pairs of individuals who are not elephants and teacups, the formula is immediately true.
Logical Consequence Example

- $\forall X, Y. \text{largerThan}(X, Y) \rightarrow \neg \text{fitsIn}(X, Y)$

- For all pairs of individuals if $X$ is larger than $Y$ (the pair is in the largerThan relation) then we cannot have that $X$ fits in $Y$ (the pair cannot be in the fitsIn relation).

- (The relation largerThan has an empty intersection with the fitsIn relation).
Logical Consequences

• \( \neg \text{fitsIn}(clyde, cup) \)

• We know \text{largerThan}(clyde, teacup) from the first implication. Thus we know this from the second implication.
Logical Consequences

fitsIn

\neg fitsIn

\text{largerThan}

\text{Elephants} \times \text{teacups} \\
\text{(clyde , cup)}
Logical Consequence Example

• If an interpretation satisfies KB, then the set of pairs elephant X teacup must be a subset of largerThan, which is disjoint from fitsIn.

• Therefore, the pair (clyde,cup) must be in the complement of the set fitsIn.

• Hence, ¬fitsIn(clyde,cup) must be true in every interpretation that satisfies KB.

• ¬fitsIn(clyde,cup) is a logical consequence of KB.
Models Graphically

Set of All Interpretations

Models of KB

a, b, c, and d are atomic formulas

Consequences? $a, c \rightarrow b$, $b \rightarrow c$, $d \rightarrow b$, $\neg b \rightarrow \neg c$
Models and Interpretations

• the more sentences in KB, the fewer models (satisfying interpretations) there are.

• The more you write down (as long as it’s all true!), the “closer” you get to the “real world”! Because Each sentence in KB rules out certain unintended interpretations.

• This is called axiomatizing the domain
Computing logical consequences

• We want procedures for computing logical consequences that can be implemented in our programs.

• This would allow us to reason with our knowledge
  • Represent the knowledge as logical formulas
  • Apply procedures for generating logical consequences

• These procedures are called proof procedures.
Proof Procedures

• Interesting, proof procedures work by simply manipulating formulas. They do not know or care anything about interpretations.

• Nevertheless they respect the semantics of interpretations!

• We will develop a proof procedure for first-order logic called resolution.
  • Resolution is the mechanism used by PROLOG
Properties of Proof Procedures

• Before presenting the details of resolution, we want to look at properties we would like to have in a (any) proof procedure.

• We write $\text{KB} \vdash f$ to indicate that $f$ can be proved from $\text{KB}$ (the proof procedure used is implicit).
Properties of Proof Procedures

• Soundness
  • $\text{KB} \vdash f \rightarrow \text{KB} \models f$
    i.e. all conclusions arrived at via the proof procedure are correct: they are logical consequences.

• Completeness
  • $\text{KB} \models f \rightarrow \text{KB} \vdash f$
    i.e. every logical consequence can be generated by the proof procedure.

• Note proof procedures for FOL have very high complexity in the worst case. So completeness is not necessarily achievable in practice.
Resolution

• **Clausal form.**
  • Resolution works with formulas expressed in clausal form.
  • A **literal** is an atomic formula or the negation of an atomic formula. dog(fido), ¬cat(fido)
  • A **clause** is a disjunction of literals:
    • ¬owns(fido,fred) ∨ ¬dog(fido) ∨ person(fred)
    • We write
      (¬owns(fido,fred), ¬dog(fido), person(fred))
  • A **clausal theory** is a conjunction of clauses.
Resolution Rule for Ground Clauses

- The resolution proof procedure consists of only one simple rule:
  - **From the two clauses**
    - \((P, Q_1, Q_2, \ldots, Q_k)\)
    - \((\neg P, R_1, R_2, \ldots, R_n)\)
  - **We infer the new clause**
    - \((Q_1, Q_2, \ldots, Q_k, R_1, R_2, \ldots, R_n)\)
  - **Example:**
    - \((\neg \text{largerThan(clyde,cup)}, \neg \text{fitsIn(clyde,cup)})\)
    - \((\text{fitsIn(clyde,cup)})\)
    - \(\Rightarrow \neg \text{largerThan(clyde,cup)}\)
Resolution Proof: Forward chaining

• Logical consequences can be generated from the resolution rule in two ways:

  1. Forward Chaining inference.
     • If we have a sequence of clauses C₁, C₂, …, Cₖ
     • Such that each Ci is either in KB or is the result of a resolution step involving two prior clauses in the sequence.
     • We then have that KB ⊢ Cₖ.

   Forward chaining is sound so we also have KB ⊨ Cₖ
Resolution Proof: Refutation proofs

2. Refutation proofs.
   - We determine if $\text{KB} \vdash f$ by showing that a contradiction can be generated from $\text{KB} \land \neg f$.
   - In this case a contradiction is an empty clause ($\emptyset$).
   - We employ resolution to construct a sequence of clauses $C_1, C_2, \ldots, C_m$ such that
     - $C_i$ is in $\text{KB} \land \neg f$, or is the result of resolving two previous clauses in the sequence.
     - $C_m = \emptyset$ i.e. its the empty clause.
Resolution Proof: Refutation proofs

• If we can find such a sequence C1, C2, …, Cm=(), we have that
  • $\text{KB} \vdash f$.
  • Furthermore, this procedure is sound so
    • $\text{KB} \models f$

• And the procedure is also complete so it is capable of finding a proof of any $f$ that is a logical consequence of $\text{KB}$. I.e.
  • If $\text{KB} \models f$ then we can generate a refutation from $\text{KB} \land \neg f$
Resolution Proofs Example

Want to prove \( \text{likes(clyde,peanuts)} \) from:
1. \( (\text{elephant(clyde)}, \text{giraffe(clyde)}) \)
2. \( (\neg\text{elephant(clyde)}, \text{likes(clyde,peanuts)}) \)
3. \( (\neg\text{giraffe(clyde)}, \text{likes(clyde,leaves)}) \)
4. \( \neg\text{likes(clyde,leaves)} \)

Forward Chaining Proof:
• 3&4 \( \rightarrow \neg\text{giraffe(clyde)} \) [5.]
• 5&1 \( \rightarrow \text{elephant(clyde)} \) [6.]
• 6&2 \( \rightarrow \text{likes(clyde,peanuts)} \) [7.] ✓
Resolution Proofs Example

1. \((\text{elephant}(\text{clyde}), \text{giraffe}(\text{clyde}))\)
2. \((\neg\text{elephant}(\text{clyde}), \text{likes}(\text{clyde},\text{peanuts}))\)
3. \((\neg\text{giraffe}(\text{clyde}), \text{likes}(\text{clyde},\text{leaves}))\)
4. \(\neg\text{likes}(\text{clyde},\text{leaves})\)

Refutation Proof:
- \(\neg\text{likes}(\text{clyde},\text{peanuts})\) [5.]
- \(5 \& 2 \rightarrow \neg\text{elephant}(\text{clyde})\) [6.]
- \(6 \& 1 \rightarrow \text{giraffe}(\text{clyde})\) [7.]
- \(7 \& 3 \rightarrow \text{likes}(\text{clyde},\text{leaves})\) [8.]
- \(8 \& 4 \rightarrow ()\) ✓
Resolution Proofs

• Proofs by refutation have the advantage that they are easier to find.
  • They are more focused to the particular conclusion we are trying to reach.

• To develop a complete resolution proof procedure for First-Order logic we need:
  1. A way of converting KB and f (the query) into clausal form.
  2. A way of doing resolution even when we have variables (unification).