CSE 3401: Intro to AI & LP
Knowledge Representation
& First-Order Logic

- Required Readings: Chapter 8
- Optional: If you need more background:
  - 7.1-7.3 Motivation for logic, basic introduction to semantics.
  - 7.4-7.5 propositional logic (good background for first-order logic).
  - 7.7 useful insights into applications of propositional logic.

Why Knowledge Representation?

- Consider the task of understanding a simple story.
- How do we test understanding?
- Not easy, but understanding at least entails some ability to answer simple questions about the story.

Example.

- Three little pigs



Example.

- Three little pigs



Example

- Why couldn't the wolf blow down the house made of bricks?
- What background knowledge are we applying to come to that conclusion?
  - Brick structures are stronger than straw and stick structures.
  - Objects, like the wolf, have physical limitations. The wolf can only blow so hard.

Why Knowledge Representation?

- Large amounts of knowledge are used to understand the world around us, and to communicate with others.
- We also have to be able to reason with that knowledge.
  - Our knowledge won't be about the blowing ability of wolfs in particular, it is about physical limits of objects in general.
  - We have to employ reasoning to make conclusions about the wolf.
  - More generally, reasoning provides an exponential or more compression in the knowledge we need to store. I.e., without reasoning we would have to store a infeasible amount of information: e.g., Elephants can't fit into teacups.

Logical Representations

- AI typically employs logical representations of knowledge.
- Logical representations useful for a number of reasons:

Logical Representations

- They are mathematically precise, thus we can analyze their limitations, their properties, the complexity of inference etc.
- They are formal languages, thus computer programs can manipulate sentences in the language.
- They come with both a formal syntax and a formal semantics.
- Typically, have well developed proof theories: formal procedures for reasoning (achieved by manipulating sentences).

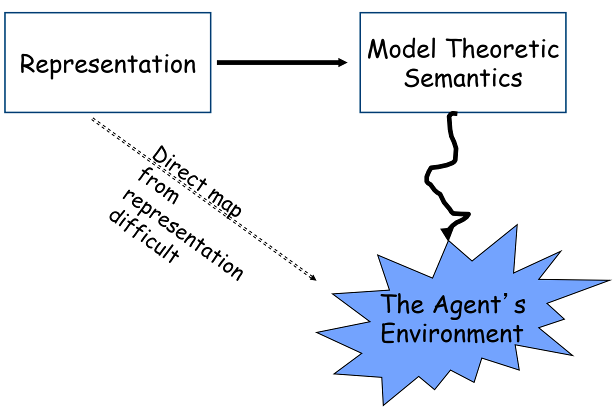
Model theoretic semantics

- Suppose our knowledge is represented in our program by some collection of data structures. We can think of these as a collection of strings (sentences).
- We want a clear mapping from this set of sentences to features of the environment. What are sentences asserting about environment?
  - In other words, we want to be able to provide an intuitive interpretation of any piece of our representation.
  - Similar in spirit to having an intuitive understanding of what individual statements in a program mean. It does not mean that it is easy to understand the whole, but it provides the means to understand the whole by understanding the parts.

Model theoretic semantics

- Model theoretic semantics facilitates both goals.
  - It is a formal characterization (in terms of sets), and it can be used to prove a wide range of properties of the representation.
  - It maps arbitrarily complex sentences of the logic down into intuitive assertions about the real world.
  - It is based on notions that are very close to how we think about the real world. Thus it provides the bridge from the syntax to an intuitive understanding of what is being asserted.

Model theoretic semantics



Semantics Formal Details

- A set of objects. These are objects in the environment that are important for your application.
- Distinguished subsets of objects. Properties.
- Distinguished sets of tuples of objects. Relations.
- Distinguished functions mapping tuples of objects to objects. Functions.

Example

- Teaching CSE 3401, want to represent knowledge that would be useful for making the course a successful learning experience.

- Objects:
  - students, subjects, assignments, numbers.
- Predicates:
  - difficult(*subject*), CSMajor(*student).*
- Relations:
  - handedIn(*student, assignment)*
- Functions:
  - Grade(*student, assignment*) → *number*

First Order Logic

1. **Syntax**: A grammar specifying what are legal syntactic constructs of the representation.

2. **Semantics**: A formal mapping from syntactic constructs to set theoretic assertions.

First Order Syntax

Start with a set of primitive symbols.

1. *constant* symbols.
2. *function* symbols.
3. *predicate* symbols (for predicates and relations).
4. *variables.*

- Each function and predicate symbol has a specific arity (determines the number of arguments it takes).

First Order Syntax—Building up.

- A *term* is either:
  - a variable
  - a constant
  - an expression of the form $f(t_1, \ldots t_k)$ where
    - (a) f is a function symbol;
    - (b) k is its arity;
    - (c) each t_i is a term

First Order Syntax—Building up.

- An *atom* is an
  - expression of the form $p(t_1, \ldots t_k)$ where
  - (a) p is a predicate symbol;
  - (b) k is its arity;
  - (c) each t_i is a term
- Note:
  - constants are the same as functions taking zero arguments.
  - Use UPPER CASE for variables, lower case for function/constant/predicate symbols.

Semantic Intuition (formalized later)

- Terms denote individuals:
  - constants denote specific individuals;
  - functions map tuples of individuals to other individuals
    - bill, jane, father(jane), father(father(jane))
    - X, father(X), hotel7, rating(hotel7), cost(hotel7)
- Atoms denote facts that can be true or false about the world
  - father_of(jane, bill), female(jane), system_down()
  - satisfied(client15), satisfied(C)
  - desires(client15,rome,week29), desires(X,Y,Z)
  - rating(hotel7, 4), cost(hotel7, 125)

First Order Syntax—Building up.

- Atoms are formulas. (Atomic formulas).
- The negation (NOT) of a formula is a new formula
  - $\neg f$ (–f)

  Asserts that f is false.
- The conjunction (AND) of a set of formulas is a formula.
  - $f_1 \wedge f_2 \wedge \ldots \wedge f_n$ where each f_i is formula

  Asserts that each formula fi is true.

First Order Syntax—Building up.

- The disjunction (OR) of a set of formulas is a formula.
  - $f_1 \vee f_2 \vee \ldots \vee f_n$ where each f_i is formula

  Asserts that at least one formula fi is true.
- Existential Quantification ∃.
  - $\exists X.\ f$ where X is a variable and f is a formula.

  Asserts there is some individual such that f under than binding will be true.
- Universal Quantification ∀.
  - $\forall X.f$ where X is a variable and f is a formula.

  Asserts that f is true for every individual.

First Order Syntax—abbreviations.

- Implication "if … then …" :
  - f1 → f2

  Take this to mean
  - ¬f1 ∨ f2.
- Double implication "if and only if":
  - F1 ↔ f2

  Take this to mean
  - (f1 → f2) ∧ (f2 → f1).
- See text for connective precedence; use ( ) to override.

Semantics.

- Formulas (syntax) can be built up recursively, and can become arbitrarily complex.
- Intuitively, there are various distinct formulas (viewed as strings) that really are asserting the same thing
  - ∀X,Y. elephant(X) ∧ teacup(Y) → largerThan(X,Y)
  - ∀X,Y. teacup(Y) ∧ elephant(X) → largerThan(X,Y)
- To capture this equivalence and to make sense of complex formulas we utilize the semantics.

Semantics.

- A formal mapping from formulas to semantic entities (individuals, sets and relations over individuals, functions over individuals).
- The mapping mirrors the recursive structure of the syntax, so we can give any formula, no matter how complex, a mapping to semantic entities.

Semantics—Formal Details

- First, we must fix the particular first-order language we are going to provide semantics for. The primitive symbols included in the syntax defines the particular language.
  L(F,P,V)
- *F = set of function (and constant symbols)*
  - *Each symbol f in F has a particular arity.*
- *P = set of predicate and relation symbols.*
  - *Each symbol p in P has a particular arity.*
- *V = an infinite set of variables.*

Semantics—Formal Details

- An interpretation (model) is a tuple
  ⟨D, Φ, Ψ,v⟩
  - D is a non-empty set (domain of individuals)
  - Φ is a mapping: $\Phi(f) \rightarrow (D^k \rightarrow D)$
    - maps k-ary function symbol f, to a function from k-ary tuples of individuals to individuals.
  - Ψ is a mapping: $\Psi(p) \rightarrow (D^k \rightarrow \text{True/False})$
    - maps k-ary predicate symbol p, to an indicator function over k-ary tuples of individuals (a subset of D^k)
  - v is a variable assignment function. $v(X) = d \in D$ (it maps every variable to some individual)

Intuitions: Domain

- Domain D: $d \in D$ is an *individual*
- E.g., *{ craig, jane, grandhotel, le-fleabag, rome, portofino, 100, 110, 120 …}*
- Underlined symbols denote domain individuals (as opposed to symbols of the first-order language)
- Domains often infinite, but we' ll use finite models to prime our intuitions

Intuitions: Φ

- $\Phi(f) \rightarrow (D^k \rightarrow D)$

Given *k*-ary function f, *k* individuals, what individual does *f(d1, …, dk)* denote

- 0-ary functions (constants) are mapped to specific individuals in D.
  - *Φ(client17) = craig, Φ(hotel5) = le-fleabag, Φ (rome) = rome*
- 1-ary functions are mapped to functions in D → *D*
  - *Φ(minquality)=f_minquality: f_minquality(craig) = 3stars*
  - *Φ(rating)=f_rating: f_rating(grandhotel) = 5stars*
- 2-ary functions are mapped to functions from $D^2 \rightarrow D$
  - *Φ(distance)=f_distance: f_distance(toronto, sienna) = 3256*
- n-ary functions are mapped similarly.

Intuitions: Ψ

- $\Psi(p) \rightarrow (D^k \rightarrow \text{True/False})$
  - given *k*-ary predicate, *k* individuals, does the relation denoted by p hold of these? Ψ(p)(*<d1, … dk>) = true?*
- 0-ary predicates are mapped to true or false.
  Ψ(rainy) = True Ψ(sunny) = False
- 1-ary predicates are mapped indicator functions of subsets of D.
  - *Ψ(satisfied) = p_satisfied: p_satisfied(craig) = True*
  - *Ψ(privatebeach) = p_privatebeach: p_privatebeack(le-fleabag) = False*
- 2-ary predicates are mapped to indicator functions over D^2
  - *Ψ(location) = p_location: p_location(grandhotel, rome) = True*
    p_location(grandhotel, sienna) = False
  - *Ψ(available) = p_available: p_available(grandhotel, week29) = True*
- n-ary predicates..

Intuitions: v

- *v* exists to take care of quantification. As we will see the exact mapping it specifies will not matter.

- Notation: v[X/d] is a new variable assignment function.
  - Exactly like v, except that it maps the variable X to the individual d.
  - Maps every other variable exactly like v:
    v(Y) = v[X/d](Y)

Semantics—Building up

Given language L(F,P,V), and an interpretation
$I = \langle D, \Phi, \Psi, v\rangle$

a) Constant c (0-ary function) denotes an individual
$I(c) = \Phi(c) \in D$

b) Variable X denotes an individual
$I(X) = v(X) \in D$ (variable assignment function).

c) Ground term $t = f(t_1,\ldots, t_k)$ denotes an individual
$I(t) = \Phi(f)(I(t_1),\ldots I(t_k)) \in D$

We recursively find the denotation of each term, then we apply the function denoted by f to get a new individual.

Hence terms always denote individuals under an interpretation I

Semantics—Building up

Formulas

a) Ground atom $a = p(t_1,\ldots t_k)$ has truth value

$I(a) = \Psi(p)(I(t_1), \ldots, I(t_k)) \in \{ \text{True}, \text{False} \}$

We recursively find the individuals denoted by the t_i, then we check to see if this tuple of individuals is in the relation denoted by p.

Semantics—Building up

Formulas

b) Negated formulas $\neg f$ has truth value

$I(\neg f)$ = True if $I(f)$ = False
$I(\neg f)$ = False if $I(f)$ = True

c) And formulas $f_1 \land f_2 \land \ldots \land f_n$ have truth value

$I(f_1 \land f_2 \land \ldots \land f_n)$ = True if every $I(f_i)$ = True.
$I(f_1 \land f_2 \land \ldots \land f_n)$ = False otherwise.

d) Or formulas $f_1 \lor f_2 \lor \ldots \lor f_n$ have truth value

$I(f_1 \lor f_2 \lor \ldots \lor f_n)$ = True if any $I(f_i)$ = True.
$I(f_1 \lor f_2 \lor \ldots \lor f_n)$ = False otherwise.

Semantics—Building up

Formulas

e) Existential formulas $\exists X.\ f$ have truth value

$I(\exists X.\ f)$ = True if there exists a $d \in D$ such that

$I'(f)$ = True

where $I' = \langle D, \Phi, \Psi, v[X/d] \rangle$

False otherwise.

I' is just like I except that its variable assignment function now maps X to d. "d" is the individual of which "f" is true.

Semantics—Building up

Formulas

f) Universal formulas $\forall X.f$ have truth value

$I(\forall X.f)$ = True if for all $d \in D$

$I'(f)$ = True

where $I' = \langle D, \Phi, \Psi, v[X/d] \rangle$

False otherwise.

Now "f" must be true of every individual "d".

Hence formulas are always either True or False under an interpretation I

Example

D = {bob, jack, fred}

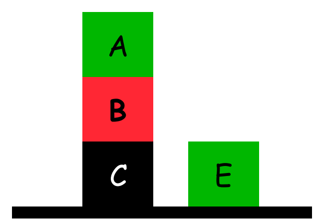
$I(\forall X.happy(X))$

1. $\Psi(happy)(v[X/bob](X)) = \Psi(happy)(bob)$ = True
2. $\Psi(happy)(v[X/jack](X)) = \Psi(happy)(jack)$ = True
3. $\Psi(happy)(v[X/fred](X)) = \Psi(happy)(fred)$ = True

Therefore $I(\forall X.happy(X))$ = True.

Models—Examples.

Environment



Language (Syntax)

- Constants: a,b,c,e
- Functions:
  - No function
- Predicates:
  - on: binary
  - above: binary
  - clear: unary
  - ontable: unary

Models—Examples.

Language (syntax)

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

A possible Model I_1 (semantics)

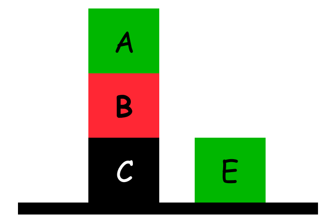
- $D = \{A, B, C, E\}$
- $\Phi(a) = A$, $\Phi(b) = B$, $\Phi(c) = C$, $\Phi(e) = E$.
- $\Psi(on) = \{(A,B),(B,C)\}$
- $\Psi(above) = \{(A,B),(B,C),(A,C)\}$
- $\Psi(clear)=\{A,E\}$
- $\Psi(ontable)=\{C,E\}$

Models—Examples.

Model I_1

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

Environment



Models—Formulas true or false?

Model I_1

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

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

- ✓ X=A, Y=B
- ✓ X=C, Y=A
- ✓ …

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

- ✓ X=A, Y=B
- × X=A, Y=C

Models—Examples.

Model I_1

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

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

- ✓ X=A
- ✓ X=C, Y=B
- ✓ …

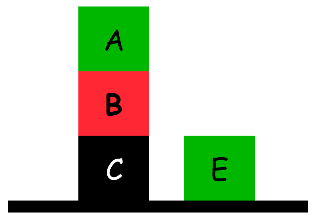
$\exists Y \forall X.(clear(X) \vee on(Y,X))$

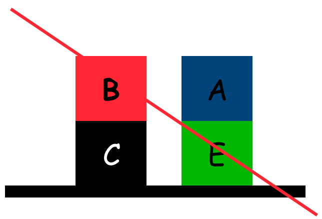
- × Y=A ? No! (X=C)
- × Y=C? No! (X=B)
- × Y=E? No! (X=B)
- × Y=B ? No! (X=B)

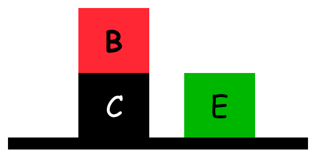
KB—many models

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 → 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

- elephant(clyde)
  - the individual denoted by the symbol *clyde* in the set denoted by *elephant* (has the property that it is an *elephant*).
- teacup(cup)
  - *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.
  - Formally, we map individuals to TRUE/FALSE (this is an indicator function for the set).

Logical Consequence Example

- ∀X,Y.elephant(X) ∧ teacup(Y) →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 *largerThan* relation.
  - For pairs of individuals who are not elephants and teacups, the formula is immediately true.

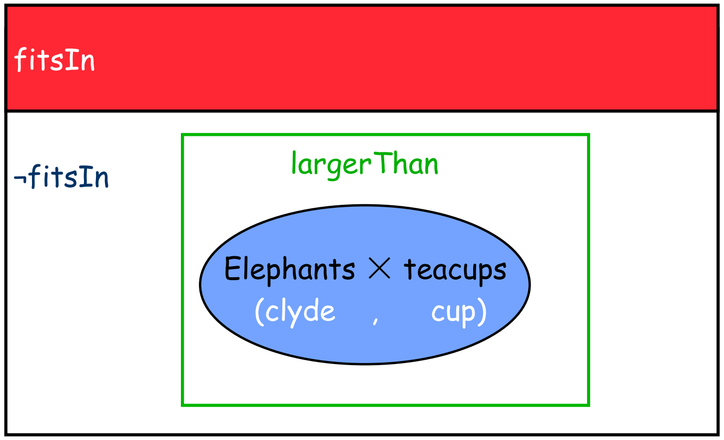
Logical Consequence Example

- ∀X,Y.largerThan(X,Y) → ¬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 a empty intersection with the fitsIn relation).

Logical Consequences

- ¬fitsIn(clyde,cup)
- We know largerThan(clyde,teacup) from the first implication. Thus we know this from the second implication.

Logical Consequences

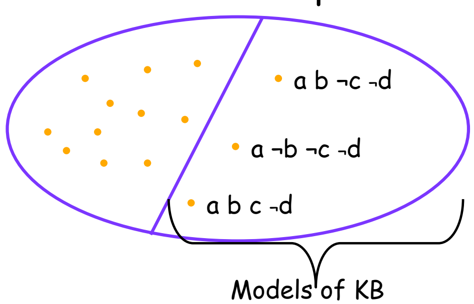


Logical Consequence Example

- 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



Consequences? a, c → b, b → c, d → b, ¬b → ¬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