

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

5

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

6

## Logical Representations

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

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

7

## 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).
- Generally **declarative** representations, which are easy to extend.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

8

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

9

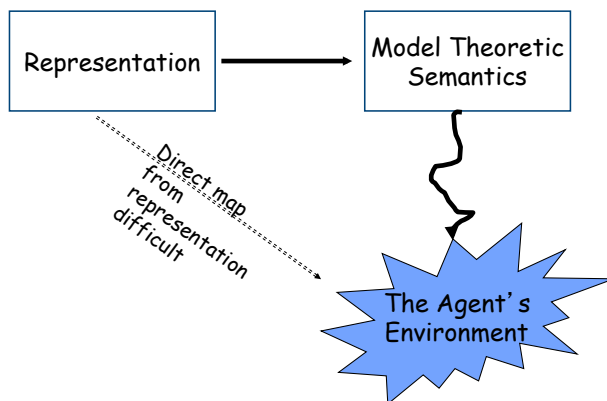
## Model theoretic semantics

- 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

10

## Model theoretic semantics



EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

11

## 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**.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

12

## 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:**
  - $\text{difficult}(\text{subject})$ ,  $\text{CSMajor}(\text{student})$ .
- **Relations:**
  - $\text{handedIn}(\text{student}, \text{assignment})$
- **Functions:**
  - $\text{Grade}(\text{student}, \text{assignment}) \rightarrow \text{number}$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

13

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

14

## 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).

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

15

## First Order Syntax—Building up.

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

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

16

## First Order Syntax—Building up.

- An *atom (or atomic formula)* is an
  - expression of the form  $p(t_1, \dots, t_k)$  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.

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

17

## 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)

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

18

## First Order Syntax—Building up.

- Atoms are formulas. (Atomic formulas).
- The negation (**NOT**) of a formula is a new formula
  - $\neg f$  ( $\neg f$ )Asserts that  $f$  is false.
- The conjunction (**AND**) of a set of formulas is a formula.
  - $f_1 \wedge f_2 \wedge \dots \wedge f_n$  where each  $f_i$  is formulaAsserts that each formula  $f_i$  is true.

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

19

## First Order Syntax—Building up.

- The disjunction (**OR**) of a set of formulas is a formula.
  - $f_1 \vee f_2 \vee \dots \vee f_n$  where each  $f_i$  is formulaAsserts that at least one formula  $f_i$  is true.
- Existential Quantification  $\exists$ .
  - $\exists X. f$  where  $X$  is a variable and  $f$  is a formula.Asserts there is some individual such that  $f$  under that binding will be true.
- Universal Quantification  $\forall$ .
  - $\forall X. f$  where  $X$  is a variable and  $f$  is a formula.Asserts that  $f$  is true for every individual.

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

20

## First Order Syntax—abbreviations.

- Implication “if ... then ...” :
  - $f1 \rightarrow f2$Take this to mean
  - $\neg f1 \vee f2$ .
- Double implication “if and only if”:
  - $F1 \leftrightarrow f2$Take this to mean
  - $(f1 \rightarrow f2) \wedge (f2 \rightarrow f1)$ .
- See text for connective precedence; use ( ) to override.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

21

## 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
  - $\forall X,Y. \text{elephant}(X) \wedge \text{teacup}(Y) \rightarrow \text{largerThan}(X,Y)$
  - $\forall X,Y. \text{teacup}(Y) \wedge \text{elephant}(X) \rightarrow \text{largerThan}(X,Y)$
- To capture this equivalence and to make sense of complex formulas we utilize the semantics.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

22

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

23

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

24

## Semantics—Formal Details

- An **interpretation** (model) is a tuple  $\langle D, \Phi, \Psi, v \rangle$ 
  - $D$  is a non-empty set (domain of individuals)
  - $\Phi$  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.
  - $\Psi$  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)

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

25

## Intuitions: Domain

- Domain  $D$ :  $d \in D$  is an *individual*
- E.g.,  $\{ \text{craig}, \text{jane}, \text{grandhotel}, \text{le-fleabag}, \text{rome}, \text{portofino}, \text{100}, \text{110}, \text{120} \dots \}$
- 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

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

26

## Intuitions: $\Phi$

- $\Phi(f) \rightarrow (D^k \rightarrow D)$   
Given  $k$ -ary function  $f$ ,  $k$  individuals, what individual does  $f(d_1, \dots, d_k)$  denote
- 0-ary functions (constants) are mapped to specific individuals in  $D$ .
  - $\Phi(\text{client17}) = \text{craig}$ ,  $\Phi(\text{hotel5}) = \text{le-fleabag}$ ,  $\Phi(\text{rome}) = \text{rome}$
- 1-ary functions are mapped to functions in  $D \rightarrow D$ 
  - $\Phi(\text{minquality}) = f_{\text{minquality}}$ :  
 $f_{\text{minquality}}(\text{craig}) = \text{3stars}$
  - $\Phi(\text{rating}) = f_{\text{rating}}$ :  
 $f_{\text{rating}}(\text{grandhotel}) = \text{5stars}$
- 2-ary functions are mapped to functions from  $D^2 \rightarrow D$ 
  - $\Phi(\text{distance}) = f_{\text{distance}}$ :  
 $f_{\text{distance}}(\text{toronto}, \text{sienna}) = \text{3256}$
- $n$ -ary functions are mapped similarly.

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

27

## Intuitions: $\Psi$

- $\Psi(p) \rightarrow (D^k \rightarrow \text{True/False})$ 
  - given  $k$ -ary predicate,  $k$  individuals, does the relation denoted by  $p$  hold of these?  $\Psi(p)(\langle d_1, \dots, d_k \rangle) = \text{true?}$
- 0-ary predicates are mapped to true or false.  
 $\Psi(\text{rainy}) = \text{True}$   $\Psi(\text{sunny}) = \text{False}$
- 1-ary predicates are mapped to indicator functions of subsets of  $D$ .
  - $\Psi(\text{satisfied}) = p_{\text{satisfied}}$ :  
 $p_{\text{satisfied}}(\text{craig}) = \text{True}$
  - $\Psi(\text{privatebeach}) = p_{\text{privatebeach}}$ :  
 $p_{\text{privatebeach}}(\text{le-fleabag}) = \text{False}$
- 2-ary predicates are mapped to indicator functions over  $D^2$ 
  - $\Psi(\text{location}) = p_{\text{location}}$ :  $p_{\text{location}}(\text{grandhotel}, \text{rome}) = \text{True}$   
 $p_{\text{location}}(\text{grandhotel}, \text{sienna}) = \text{False}$
  - $\Psi(\text{available}) = p_{\text{available}}$ :  
 $p_{\text{available}}(\text{grandhotel}, \text{week29}) = \text{True}$
- $n$ -ary predicates..

EECS 3401 Fall 2018 Fabien Bacchus & Yves Lesperance

28

## 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  $v$ :  
 $v(Y) = v[X/d](Y)$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

29

## Semantics—Building up

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

- Constant  $c$  (0-ary function) **denotes** an individual  
 $I(c) = \Phi(c) \in D$
- Variable  $X$  denotes an individual  
 $I(X) = v(X) \in D$  (variable assignment function).
- A complex term  $t = f(t_1, \dots, t_k)$  **denotes** an individual  
 $I(t) = \Phi(f)(I(t_1), \dots, I(t_k)) \in D$

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  $I$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

30

## Semantics—Building up

Formulas

- An atom  $a = p(t_1, \dots, t_k)$  has **truth value**  
 $I(a) = \Psi(p)(I(t_1), \dots, I(t_k)) \in \{ \text{True}, \text{False} \}$

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  $p$ .

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

31

## Semantics—Building up

Formulas

- Negated formulas  $\neg f$  has truth value  
 $I(\neg f) = \text{True}$  if  $I(f) = \text{False}$   
 $I(\neg f) = \text{False}$  if  $I(f) = \text{True}$
- And formulas  $f_1 \wedge f_2 \wedge \dots \wedge f_n$  have truth value  
 $I(f_1 \wedge f_2 \wedge \dots \wedge f_n) = \text{True}$  if every  $I(f_i) = \text{True}$ .  
 $I(f_1 \wedge f_2 \wedge \dots \wedge f_n) = \text{False}$  otherwise.
- Or formulas  $f_1 \vee f_2 \vee \dots \vee f_n$  have truth value  
 $I(f_1 \vee f_2 \vee \dots \vee f_n) = \text{True}$  if any  $I(f_i) = \text{True}$ .  
 $I(f_1 \vee f_2 \vee \dots \vee f_n) = \text{False}$  otherwise.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

32



## Semantics—Building up

### Formulas

- e) Existential formulas  $\exists X. f$  have truth value  
 $I(\exists X. f) = \text{True}$  if **there exists** a  $d \in D$  such that  
 $I'(f) = \text{True}$   
 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

33

## Semantics—Building up

### Formulas

- f) Universal formulas  $\forall X. f$  have truth value  
 $I(\forall X. f) = \text{True}$  if **for all**  $d \in D$   
 $I'(f) = \text{True}$   
 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  $I$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

34

## Example

$D = \{\text{bob}, \text{jack}, \text{fred}\}$   
 $I(\forall X. \text{happy}(X))$

- $\Psi(\text{happy})(v[X/\text{bob}](X)) = \Psi(\text{happy})(\text{bob}) = \text{True}$
- $\Psi(\text{happy})(v[X/\text{jack}](X)) = \Psi(\text{happy})(\text{jack}) = \text{True}$
- $\Psi(\text{happy})(v[X/\text{fred}](X)) = \Psi(\text{happy})(\text{fred}) = \text{True}$

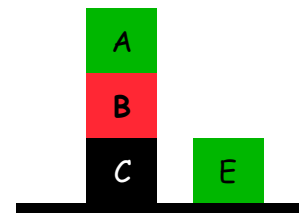
Therefore  $I(\forall X. \text{happy}(X)) = \text{True}$ .

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

35

## Models—Examples.

### Environment



### Language (Syntax)

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

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

36

## Models—Examples.

Language (syntax)

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

A possible Model  $I_1$  (semantics)

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

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

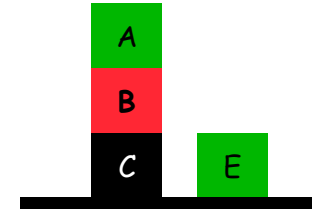
37

## Models—Examples.

Model  $I_1$

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

Environment



EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

38

## Models—Formulas true or false?

Model  $I_1$

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

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

- ✓  $X = \underline{A}, Y = \underline{B}$
- ✓  $X = \underline{C}, Y = \underline{A}$
- ✓ ...

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

- ✓  $X = \underline{A}, Y = \underline{B}$
- ✗  $X = \underline{A}, Y = \underline{C}$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

39

## Models—Examples.

Model  $I_1$

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

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

- ✓  $X = \underline{A}$
- ✓  $X = \underline{C}, Y = \underline{B}$
- ✓ ...

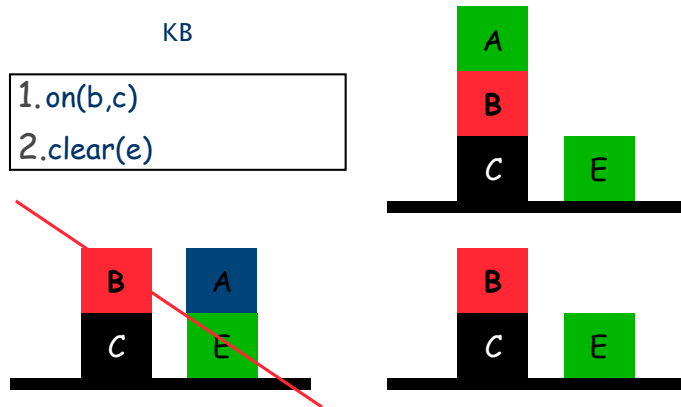
$\exists Y \forall X. (\text{clear}(X) \vee \text{on}(Y, X))$

- ✗  $Y = \underline{A} ? \text{No! } (X = \underline{C})$
- ✗  $Y = \underline{C} ? \text{No! } (X = \underline{B})$
- ✗  $Y = \underline{E} ? \text{No! } (X = \underline{B})$
- ✗  $Y = \underline{B} ? \text{No! } (X = \underline{B})$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

40

## KB—many models



EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

41

## 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  $I$ .

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

42

## 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.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

43

## 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?

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

44

## 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).

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

45

## Logical Consequence Example

- $\forall X,Y. \text{elephant}(X) \wedge \text{teacup}(Y) \rightarrow \text{largerThan}(X,Y)$ 
  - 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 *largerThan* relation.
  - For pairs of individuals who are not elephants and teacups, the formula is immediately true.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

46

## 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 a empty intersection with the *fitsIn* relation).

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

47

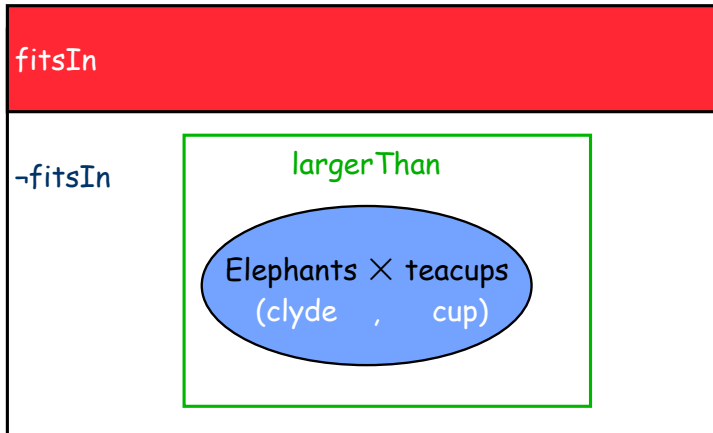
## Logical Consequences

- $\neg \text{fitsIn}(\text{clyde}, \text{cup})$
- We know *largerThan*(clyde,teacup) from the first implication. Thus we know this from the second implication.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

48

## Logical Consequences



EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

49

## Logical Consequence Example

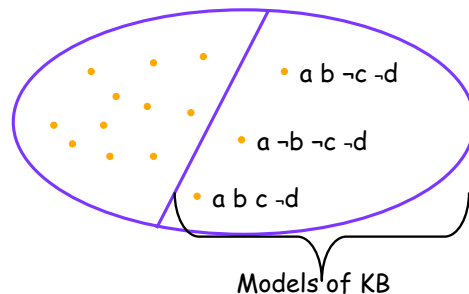
- If an interpretation satisfies KB, then the set of pairs *elephant X teacup* must be a subset of *largerThan*, which is disjoint from *fitsIn*.
- Therefore, the pair *(clyde, cup)* must be in the complement of the set *fitsIn*.
- Hence,  $\neg \text{fitsIn}(\text{clyde}, \text{cup})$  must be true in every interpretation that satisfies KB.
- $\neg \text{fitsIn}(\text{clyde}, \text{cup})$  is a logical consequence of KB.

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

50

## Models Graphically

### Set of All Interpretations



Consequences?  $a, c \rightarrow b, b \rightarrow c, d \rightarrow b, \neg b \rightarrow \neg c$

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

51

## 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**

EECS 3401 Fall 2018 Fahiem Bacchus & Yves Lesperance

52

## Knowledge Representation

- **Knowledge representation** is the area of AI that focuses on how to represent information about an application domain.
- An important aspect is specifying the concepts and relations of interest and how they relate, i.e. the domain's **ontology**.
- Many large ontologies have been developed, e.g.
  - Cyc general-purpose ontology, more than 1 million terms
  - Dbpedia which represents the content of Wikipedia formally
  - SNOMED CT, for medical terms
- Related to Semantic Web, which seeks to make information on the web machine processable
- **Important application is Ontology-Based Data Access**
- **Tools for building ontologies, e.g. Protégé**

## Knowledge Representation Formalisms

- These are formal languages for representing information
- *FOL is quite expressive, but checking entailment is undecidable*
- *Description logics are essentially decidable fragments of FOL for knowledge representation and reasoning; concepts/classes are organized in a hierarchy*
- *OWL (Web Ontology Language) is a family of such representation languages standardized by the W3C*
- *Reasoners have been developed for such formalisms, e.g. RACER, FACT++*