First-Order Logic

Logic in Computer Science

1

Pros and Cons of Propositional Logic

- ☺ It is declarative
  - – solutions are specified by constraints, not by a procedure
- ☺ It allows partial/disjunctive/negated information
  - – (unlike most data structures and databases)
- ☺ It is compositional:
  - – meaning of $p \wedge q$ is derived from meanings of p and q
- ☺ Meaning in propositional logic is context-independent
  - – (unlike natural language, where meaning depends on context)
- ☹ It has very limited expressive power
  - – (unlike natural language)
  - – E.g., cannot say "everybody has a mother"

2

Problems with propositional logic

- No notion of objects
- No notion of relations among objects
- In Propositional Logic, we define A1 as "American sits at seat 1." The meaning of A1 is instructive **to us**, suggesting
  - there is an object we call American,
  - there is an object we call "seat 1",
  - there is a relationship "sit" between these two objects
- Formally, none of these are in Propositional Logic.

3

First-Order Logic

- First-order logic (FOL) models the world in terms of
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from other objects
  - **Relations** that hold among sets of objects
  - **Functions,** which map individuals in the domain to another in the domain.
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Properties: blue, oval, even, large, ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

4

A BNF for FOL

```
<Formula> ::= ⊥ | ⊤ | <Atom> |
              (<Formula> <Operator> <Formula>) |
              (<Quantifier> <Variable> <Formula>) |
              ¬ <Formula>
<Atom> ::= <Predicate>(<Term>, ...) | <Predicate>
<Term> ::= <Function>(<Term>, ...) |
           <Constant> | <Variable>
<Operator> ::= ∧ | ∨ | → | ↔
<Quantifier> ::= ∀ |
<Constant> ::= a | b | c | ... ∃
<Variable> ::= x | y | z | ...
<Predicate> ::= > | = | isRed | p | q | ...
<Function> ::= mother | head | ...
```

5

Precedence of Logical Operators

- As in arithmetic, an ordering is imposed on the use of logical operators in compound propositions

$\forall x\, p(x) \to q(x) \lor r(x)$ instead of $\forall x\, (p(x) \to (q(x) \lor r(x)))$

- To avoid unnecessary parenthesis, the following precedence order holds:
  1. Negation ($\neg$)
  2. Conjunction ($\land$)
  3. Disjunction ($\lor$)
  4. Implication ($\to$)
  5. Logical equal ($\leftrightarrow$), exclusive or ($\oplus$)
  6. Quantifiers ($\forall$, $\exists$)

6

Application related components

- **Constant symbols,** which represent objects in the world
  - Mary
  - green
- **Function symbols,** which map individuals to individuals
  - father(Mary) = John
  - color(Sky) = blue
- **Predicate symbols,** which map individuals to truth values
  - >(5, 3)
  - isGreen(Grass)
  - isColor(Grass, Green)

First-order language L = (*P*, *F*, *X*, *Op*)

7

Common Parts in FOL

First-order language L = (*P*, *F*, *X*, *Op*)

- **Variable symbols**
  - E.g., x, y, x1, …
- **Logical Operators**
  - Same as in PL: not (¬), and (∧), or (∨), implies (→), logic equal (biconditional ↔)
- **Quantifiers**
  - Universal ∀**x**
  - Existential ∃**x**

8

Formulas are built from terms and atoms

- A **term** (denoting a real-world individual) is either a constant symbol, a variable symbol, or an n-place function of n terms.
  - x and $f(t_1, ..., t_n)$ are terms, where each t_i is a term.

  A term with no variables is a **ground term**
- An **atom** (which has value true or false) is either
  - an n-place predicate of n terms, or a propositional variable
- A **formula** is either a Boolean constant, an atom, or an expression built from operators and quantifiers.
  - If P, Q are formulas, then $\forall$x P and $\exists$x P are formulas
  - $\neg$P, P$\vee$Q, P$\wedge$Q, P$\rightarrow$Q, P$\leftrightarrow$Q are formulas but not atoms

9

Quantifiers

- **Universal quantification**
  - $\forall$x P(x) means that P holds for **all** values of x in the domain associated with that variable
    - E.g., $\forall$x dolphin(x) $\rightarrow$ mammal(x)
- **Existential quantification**
  - means that P holds for **some** value of x in the domain associated with that variable
    - E.g., $\exists$x manmal(x) $\wedge$ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

10

Some Definitions

- The **scope** of x in ∀x P(x) or ∃x P(x) will be the whole formula P(x).
  - E.g., the scope of *x* in (∀ *x* p(*x*) ∨ q(*x*)) ∧ r(*x*) is p(*x*) ∨ q(*x*)
- If *x* appears in the scope of ∀*x* or ∃*x*, then *x* is bound; otherwise *x* is free.
- A **formula** is closed if it has no free variables; otherwise it's open.
  - ∀*x* p(*x*, *y*) is open because *x* is bound but *y* is free.
- A **formula** is ground if it has no variables, free or bound.
  - p(*a*, *f*(*b*)) is ground, but p(*x*, *a*) is not ground.

11

Universal Quantifiers

- Universal quantifiers are often used with "implies" to form "conditional rules":
  - ∀x student(x) → smart(x)
  - It means "All students are smart"
- Universal quantification is *rarely* used to make blanket statements about every individual in the world:
  - ∀x student(x) ∧ smart(x)
  - It means "Everyone in the world is a student and is smart"

12

Existential Quantifiers

- Existential quantifiers are usually used with "and" to specify a list of properties about an individual:
  - $\exists x$ student(x) $\wedge$ smart(x)
  - It means "There is a student who is smart"
- A common mistake is to represent this English sentence as the FOL sentence:
  - $\exists x$ student(x) $\rightarrow$ smart(x)
  - But what happens when there is a person who is *not* a student?

13

Translating English to FOL

- **All zombies eat brains.**
  - **This assumption can be re**presented in propositional logic by the propositional variable *p*.
  - In first order logic, the expression can be broken down
    - $\forall x\ zombie(x) \rightarrow eatsbrains(x)$
    - $\forall x\ zombie(x) \rightarrow \text{eats}(x, brains)$
    - $\forall x\ \exists y\ zombie(x) \rightarrow eats(x, y) \wedge brains(y)$
- **Some people with no heartbeats are zombies.**
  - $\exists x\ \text{person}(x) \wedge \text{zombie}(x) \wedge \neg\, heartbeat(x)$

14

Translating English to FOL

You can fool some of the people all of the time.

$\exists x \forall t$ person(x) $\wedge$ time(t) $\rightarrow$ foolAt(x, t)

You can fool all of the people some of the time.

$\forall x \exists t$ person(x) $\wedge$ time(t) $\rightarrow$ foolAt(x, t)

You cannot fool all of the people all of the time.

$\neg\forall x \forall t$ person(x) $\wedge$ time(t) $\rightarrow$ foolAt(x, t)

All purple mushrooms are poisonous.

$\forall x$ mushroom(x) $\wedge$ purple(x) $\rightarrow$ poisonous(x)

No purple mushroom is poisonous.

$\neg\exists x$ purple(x) $\wedge$ mushroom(x) $\wedge$ poisonous(x)

$\forall x$ mushroom(x) $\wedge$ purple(x) $\rightarrow \neg$ poisonous(x)

15

Translating English to FOL

There are exactly two purple mushrooms.

$\exists x \exists y$ mushroom(x) $\wedge$ purple(x) $\wedge$ mushroom(y) $\wedge$ purple(y) $\wedge \neg(x=y) \wedge \forall z$ mushroom(z) $\wedge$ purple(z) $\rightarrow ((x=z) \vee (y=z))$

Clinton is not tall.

$\neg$ tall(Clinton)

X is above Y if X is directly on top of Y or there is another object Z directly on top of Y and X is above Z.

$\forall x \forall y$ above(x, y) $\leftrightarrow$ (on(x,y) $\vee \exists z$ on(z, y) $\wedge$ above(x, z))

16

Formalizing English Sentences

. Some rules that usually work for English sentences are:

- $\forall x$ quantifies a conditional.
- $\exists x$ quantifies a conjunction.
- Use $\forall x$ with conditional for "all," "every," and "only."
- Use $\exists x$ with conjunction for "some," "there is," and "not all."
- Use $\forall x$ with conditional or $\neg \exists x$ with conjunction for "no *A* is *B*."
- Use $\exists x$ with conjunction or $\neg \forall x$ with conditional for "not all *A's* are *B*."

18

Semantics of FOL

- Define an **interpretation I** in which a formula is true or false
- **Interpretation I** includes
  - **Domain D:** the set of all objects in the world (of interest)
    - Assign each constant to an object in **D**
    - Assign each function of n arguments to a function D^n => **D**
    - Assign each predicate of n arguments to a relation D^n => {1, 0}
  - Therefore, every ground formula will have a truth value
  - In general there is an infinite number of interpretations because **D** may be infinite
- **Define logical operators** as in Propositional Logic
- **Define semantics of $\forall$x and $\exists$x**
  - $\forall$x P(x) is true iff P(x) is true under all substitutions for x in **D**.
  - $\exists$x P(x) is true iff P(x) is true under some substitutions for x in **D**.

19

Truth in first-order logic

First-order language L = (P, F, X, Op)

- Formulas are true with respect to an interpretation

$$I = (D, R, G)$$

- I defines D as the set of objects (domain of elements)
- I assigns relations R to predicate symbols P:

$$p \in P \quad \text{iff} \quad p^I \in R$$

- I assigns functions G over D to function symbols (including constant symbols) F:

$$f \in F \quad \text{iff} \quad f^I \in G$$

20

Truth in first-order logic

L = (P, F, X, Op)
I = (D, R, G)

- **Example**: Interpretations of $\forall x\, p(x, f(x))$
- Interpretation I_1:
  D^{I_1} = people, $p^{I_1} = love$, $f^{I_1} = child\_of$
  $\forall x\, p(x, f(x))$ = "everybody loves his/her child"

- Interpretation I_2:
  $D^{I_2} = integers$, $p^{I_2} = less$, $f^{I_2} = successor$
  $\forall x\, p(x, f(x))$ = "every integer is less than its successor"

- Interpretation I_3:
  $D^{I_3} = sets$, $p^{I_3} = disjoint$, $f^{I_3} = complement$
  $\forall x\, p(x, f(x))$ = "any set is disjoint from its complement"

21

Truth in first-order logic

L = (P, F, X, Op)
I = (D, R, G)

- **substitution**: θ: $X \rightarrow D$
  a function which assigns an object to a variable.
- **proc** *eval*(A, I, θ) // evaluate A to an object or a truth value
  if $A = 0$ or $A = 1$ **return** A; // A is a Boolean constant
  if $A \in X$ **return** $\theta(A)$; // A is a free variable, return an object
  if $A = f(t_1, t_2, ..., t_k)$ **return** $f^I(a_1, a_2, ..., a_k)$, where $a_i = eval(t_i, I, \theta)$;
  if $A = p(t_1, t_2, ..., t_k)$ **return** $p^I(a_1, a_2, ..., a_k)$, where $a_i = eval(t_i, I, \theta)$;
  if $A = \neg B$ **return** $\neg\, eval(B, I, \theta)$; // the top symbol of A is "$\neg$"
  if $A = (B\ op\ C)$ **return** $eval(B, I, \theta)\ op\ eval(C, I, \theta)$,
  // the top symbol of A is "op", i.e., $\wedge, \vee, \rightarrow, \leftrightarrow$
  if $A = (\forall x\ B)$ **return** $allInD(B, I, \theta, x, D^I)$; // top sym. of A is "$\forall$"
  if $A = (\exists x\ B)$ **return** $someInD(B, I, \theta, x, D^I)$; // top sym. of A is "$\exists$"

22

Truth in first-order logic

L = (P, F, X, Op)
I = (D, R, G)

- **substitution**: θ: $X \rightarrow D$
  a function which assigns an object to a variable.

proc $allInD(B, I, \theta, x, S)$; // = eval($\forall x\ B, I, \theta$), $S = D^I$
  if $S = \varnothing$ **return** 1;
  pick $d \in S$;
  if $eval(B, I, \theta \cup \{x{=}d\}) = 0$ **return** 0;
  return $allInD(B, I, \theta, x, S - \{ d \})$;

$$\text{eval}(\forall x\ B, I, \theta) = \bigwedge_{d \in D^I} eval(B, I, \theta \cup \{x{=}d\})$$

23

Truth in first-order logic

L = (*P*, *F*, *X*, *Op*)
I = (*D*, *R*, *G*)

- **substitution**: $\theta: X \rightarrow D$
  a function which assigns an object to a variable.

proc *someInD*(*B*, *I*, *θ*, *x*, *S*) // = eval(∃*x B*, *I*, *θ*), $S = D^I$
  if $S = \varnothing$ **return** 0;
  pick $d \in S$;
  if *eval*(*B*, *I*, $\theta \cup \{x=d\}$) = 1 **return** 1;
  return *someInD*(*B*, *I*, *θ*, *x*, $S - \{ d \}$);

$$\text{eval}(\exists x B, I, \theta) = \bigvee_{d \in D^I} eval(B, I, \theta \cup \{x=d\})$$

24

Truth in first-order logic

L = (*P*, *F*, *X*, *Op*)
I = (*D*, *R*, *G*)

- **substitution**: $\theta: X \rightarrow D$
  a function which assigns an object to a variable.

$$\text{eval}(\exists x B, I, \theta) = \bigvee_{d \in D^I} eval(B, I, \theta \cup \{x=d\})$$

- **Example:** L = ({*p*}, {*a*}, {*x*}, *Op*)
  I = (*D*, {*r*}, {*a*}), where *D*={*a*, *b*, *c*}, *r* = {(*a*, *b*), (*b*, *c*), (*c*, *a*)}

- $\text{eval}(\exists x\, p(a, x), I, \varnothing) = \bigvee_{d \in D} eval(B, I, \varnothing \cup \{x=d\}) =$

$\text{eval}(p(a,x), I, \{x=a\}) \vee \text{eval}(p(a,x), I, \{x=b\}) \vee \text{eval}(p(a,x), I, \{x=c\}) =$
$r(a, \text{eval}(x, I, \{x=a\})) \vee r(a, \text{eval}(x, I, \{x=b\})) \vee r(a, \text{eval}(x, I, \{x=c\})) =$
$r(a, a) \vee r(a, b) \vee r(a, c) = 0 \vee 1 \vee 0 = 1$

25

More Definitions

L = (*P*, *F*, *X*, *Op*)
I = (*D*, *R*, *G*)

- **Definitions**: Let A be a closed formula.
- An interpretation I is a model of A if eval(A, I, ∅) = 1.
- If A has model, A is said to be satisfiable.
- For any formula A, **M**(A) is the set of all models of A, i.e., all interpretations that satisfy A.
- A is valid if **M**(A) contains every interpretation, denoted by |= A.
- A is unsatisfiable if **M**(A) is empty.
- Given formula B, A entails B if **M**(A) ⊆ **M**(B), denoted by A |= B. We say "B is a logical consequence of A."
- A and B are equivalent, A ≡ B, if **M**(A) = **M**(B).

26

Properties of Quantifiers

- Changing quantified variables names *does not* change the meaning:
  - (∀x P(x)) ≡ (∀y P(y))
  - (∃x P(x)) ≡ (∃y P(y))

- Proof: For any $I = (D, R, G)$,
- *eval*(∀x P(x), *I*, ∅) =

$$\bigwedge_{d \in D} eval(P(x), I, \{x=d\}) =$$

$$\bigwedge_{d \in D} eval(P(y), I, \{y=d\}) =$$

eval(∀y P(y), *I*, ∅)

eval(∀*x B*, *I*, *θ*) =

$$\bigwedge_{d \in D^I} eval(B, I, \theta \cup \{x=d\})$$

27

Properties of Quantifiers

- Switching the order of universal quantifiers *does not* change the meaning:
  - $(\forall x \forall y P(x,y)) \equiv (\forall y \forall x\ P(x,y))$
- Proof: For any $I = (D, R, G)$.
- eval($\forall x\ \forall y$ P(x,y), I, $\emptyset$) =

$$\bigwedge_{d \in D} eval(\forall y\ P(x,y), I, \{x=d\}) =$$

$$\bigwedge_{d \in D} \bigwedge_{d' \in D} eval(P(x,y), I, \{x=d, y=d'\}) =$$

$$\bigwedge_{d' \in D} \bigwedge_{d \in D} eval(P(x,y), I, \{x=d, y=d'\}) =$$

$$\bigwedge_{d' \in D} eval(\forall x\ P(x,y), I, \{y=d'\}) =$$

eval($\forall y\ \forall x$ P(x,y), I, $\emptyset$)

- We may write $\forall x,y$ P(x,y)) for $\forall x \forall y$ P(x,y))

eval($\forall x\ B$, I, θ) =

$$\bigwedge_{d \in D^I} eval(B, I, \theta \cup \{x=d\})$$

28

Properties of Quantifiers

- We can also switch the order of existential quantifiers:
  - $(\exists x\ \exists y\ P(x,y)) \equiv (\exists y\ \exists x\ P(x,y))$
  - We may write $\exists x,y$ P(x,y)) for $\exists x \exists y$ P(x,y))

$$eval(\exists x B, I, \theta) = \bigvee_{d \in D^I} eval(B, I, \theta \cup \{x=d\})$$

- Switching the order of universal and existential quantifiers *does* change meaning:
  - Everyone likes someone: $\forall x\ \exists y$ likes(x, y)
  - Someone is liked by everyone: $\exists y \forall x$ likes(x,y)
  - Everyone has a mother: $\forall x\ \exists y$ mother(x, y)
  - Someone is everyone's mother: $\exists y \forall x$ mother(x,y)

29

Properties of Quantifiers

- $(\forall x\ \exists y\ p(x, y)) \to \exists y \forall x\ p(x, y)$ is not valid.
- **Proof**: Consider I = (N, { < }, { }), then $(\forall x\ \exists y\ p(x, y))$ states "for each integer x, there exists an integer y, such that x < y", which is true. $(\exists y \forall x\ p(x, y))$ states that "there exists a (maximal) integer y, such that x < y for any integer x", which is false. So $(\forall x\ \exists y\ p(x, y)) \to \exists y \forall x\ p(x, y)$ is false in I.
- $\models (\exists y \forall x\ p(x, y)) \to (\forall x\ \exists y\ p(x, y))$.
- **Proof**: For any I = (D, {p}, ∅), if $eval(\exists y \forall x\ p(x, y), I, \emptyset) = 1$, then $eval(\forall x\ p(x, y), I, \{y=d\}) = 1$ for some $d \in D$.
- So $eval(\forall x\ p(x, y), I, \{y=d\}) = \bigwedge_{a \in D} eval(p(x,y), I, \{y=d, x=a\})$
- $= \bigwedge_{a \in D} eval(\exists y\ p(x, y), I, \{x=a\}) = eval(\forall x\ \exists y\ p(x, y), I, \emptyset) = 1$
- Hence, I is a model of $\forall x\ \exists y\ p(x, y)$.

30

Connections between All and Exists

∀ and ∃'s De Morgan's laws:

1. $(\forall x\ \neg P(x)) \equiv \neg(\exists x\ P(x))$
2. $(\forall x\ Q(x)) \equiv \neg(\exists x\ \neg Q(x))$
3. $(\exists x\ \neg P(x)) \equiv \neg(\forall x\ P(x))$
4. $(\exists x\ Q(x)) \equiv \neg(\forall x)\ \neg Q(x))$

Proof: 1. $eval(\forall x\ \neg P(x), I, \emptyset) = \bigwedge_{d \in D^I} eval(\neg P(x), I, \{x=d\}) =$

$\bigwedge_{d \in D^I} \neg eval(P(x), I, \{x=d\}) = \neg \bigvee_{d \in D^I} eval(P(x), I, \{x=d\}) =$

$= eval(\neg(\exists x\ P(x)), I, \emptyset)$

2. From 1. letting $Q(x) = \neg P(x)$.
3. From 2. letting $Q(x) = P(x)$, and negating both sides.
4. From 3. letting $Q(x) = \neg P(x)$.

31

Properties of Quantifiers

- $(\exists x\ p(x) \rightarrow q(x)) \equiv (\forall x\ p(x)) \rightarrow (\exists x\ q(x))$

- **Proof**: X
  $\exists x\ p(x) \rightarrow q(x) \equiv$
  $\exists x\ \neg p(x) \vee q(x) \equiv$
  $(\exists x\ \neg p(x)) \vee (\exists x\ q(x)) \equiv$
  $\neg(\forall x\ p(x)) \vee (\exists x\ q(x)) \equiv$
  $(\forall x\ p(x)) \rightarrow (\exists x\ q(x))$

32

Properties of Quantifiers

If x is not a free variable in B, then

1. $(\forall x\ A(x) \vee B) \equiv (\forall x\ A(x)) \vee B$
2. $(\forall x\ A(x) \wedge B) \equiv (\forall x\ A(x)) \wedge B$
3. $(\exists x\ A(x) \vee B) \equiv (\exists x\ A(x)) \vee B$
4. $(\exists x\ A(x) \wedge B) \equiv (\exists x\ A(x)) \wedge B$

Proof: 1. $eval(\forall x\ A(x) \vee B, I, \emptyset) = \bigwedge_{d \in D^I} eval(A(x) \vee B, I, \{x=d\}) =$

$\bigwedge_{d \in D^I} \big(eval(A(x), I, \{x=d\}) \vee eval(B, I, \{x=d\})\big) =$

$(\bigwedge_{d \in D^I} eval(A(x), I, \{x=d\})) \vee eval(B, I, \emptyset) =$

$(eval(\forall x\ A(x), I, \{\})) \vee eval(B, I, \emptyset) =$

$eval((\forall x\ A(x)) \vee B, I, \emptyset)$

33

Properties of Quantifiers

We know that $\exists x\ \forall y\ p(x, y) \rightarrow \forall y\ \exists x\ p(x, y)$ is valid, but the converse is not valid. So we can't interchange $\exists x$ and $\forall y$. But for predicates that take single arguments, the two quantifiers can be interchanged:

$$\exists x\ \forall y\ (p(x) \rightarrow q(y)) \equiv \forall y\ \exists x\ (p(x) \rightarrow q(y)).$$

Proof: $\exists x\ \forall y\ (p(x) \rightarrow q(y)) \equiv \exists x\ \forall y\ (\neg p(x) \vee q(y))$

$\equiv \exists x\ (\neg\, p(x) \vee \forall y\ q(y))$

$\equiv (\exists\, x \neg\, p(x)) \vee \forall y\ q(y)$

$\equiv \forall y\ (\exists\, x \neg\, p(x)) \vee q(y)$

$\equiv \forall y\ \exists x\ (\neg\, p(x) \vee q(y))$

$\equiv \forall y\ \exists x\ (p(x) \rightarrow q(y))$

$(\forall x\ \ A(x) \vee B) \equiv (\forall x\ A(x)) \vee B$
$(\exists x\ \ A(x) \vee B) \equiv (\exists x\ A(x)) \vee B$

34

Properties of Quantifiers

- $A_1 = \forall x\ p(x, a) \rightarrow p(a, a)$ is satisfiable, but not valid.
- **Proof**: Consider I = ({a, b}, {p}, {a}), where p = {(a,a)}. *eval*(p(a,a), I, {}) = 1, so I is a model of A_1.
- Let I' = ({a, b}, {p}, {a}), where p = {(b,a)}. *eval*(p(a,a), I', $\emptyset$) = 0, eval(p(b,a), I', $\emptyset$) = 1, hence *eval*(p(b,a) $\rightarrow$ p(a, a) , I', $\emptyset$) = 0, so A_1 is false in I'.
- $A_2 = (\forall x\ p(x, a)) \rightarrow p(a, a)$ is valid.
- **Proof**: For any I = (D, {p}, {a}),
- If *eval*(p(a, a), I, $\emptyset$) = 1, then A_2 is true in I.
- If *eval*(p(a, a), I, $\emptyset$) = 0, then *eval*($\forall$x p(x, a), I, $\emptyset$) = 0.
- Thus, eval(A_2, I, $\emptyset$) = 1 in both cases for any interpretation I.
- Note: $A_2 \equiv \neg\forall x\ p(x, a) \vee p(a, a) \equiv \exists x\ p(x, a) \rightarrow p(a, a)$

35

Properties of Quantifiers

- A = (∀x p(x) ∨ q(x)) → (∀x p(x)) ∨ (∃x q(x)) is valid.
- **Proof**: For any I = (D, R, G),
- if eval((∀x p(x)) ∨ (∃x q(x)), I, ∅) = 1, then A is true in I.
- If eval((∀x p(x)) ∨ (∃x p(x)), I, ∅) = 0,
- then eval(∀x p(x), I, ∅) = 0 and eval(∃x q(x), I, ∅) = 0.
- From eval(∀x p(x), I, ∅) = 0, eval(p(x), I, {x = c}) = 0 for some c in D,
- From eval(∃x q(x), I, ∅) = 0, eval(q(x), I, {x = c}) = 0
- So eval(p(x) ∨ q(x), I, {x = c}) = 0
- So eval(∀x p(x) ∨ q(x), I, ∅) = 0
- So A is true in I and I is a model of A.
- Since I is arbitrary, A is true in every interpretation.

36

Rules for Semantic Tableau

- α-rules (AND-rules)

$$\frac{U \cup \{\alpha\}}{U \cup \{\alpha_1, \alpha_2\}}$$

- β-rules (OR-rules)

$$\frac{U \cup \{\beta\}}{U \cup \{\beta_1\} \qquad U \cup \{\beta_2\}}$$

37

α-rules

- α-rule produces a conjunction of formulas and creates one successor node ($\wedge$ is replaced by ",")
  - $A \wedge B \equiv A, B$ // "," for $\wedge$
  - $A \downarrow B \equiv \neg A, \neg B$
  - $A \oplus B \equiv (A \vee B), (\neg A \vee \neg B)$
  - $A \leftrightarrow B \equiv (\neg A \vee B), (A \vee \neg B)$
  - $\neg (A \vee B) \equiv \neg A, \neg B$
  - $\neg (A \rightarrow B) \equiv A, \neg B$
  - $\neg (A \uparrow B) \equiv A, B$
  - $\neg \neg p \equiv p$

38

β-rules

- β-rule produces a disjunction of formulas and creates two successor nodes (for each disjunct)
  - $A \vee B \equiv A \mid B$ // "|" for $\vee$, two branches
  - $A \rightarrow B \equiv \neg A \mid B$
  - $A \uparrow B \equiv \neg A \mid \neg B$
  - $\neg (A \wedge B) \equiv \neg A \mid \neg B$
  - $\neg (A \downarrow B) \equiv A \mid B$
  - $\neg (A \oplus B) \equiv (A \wedge B) \mid (\neg A \wedge \neg B)$
  - $\neg (A \leftrightarrow B) \equiv (\neg A \wedge B) \mid (A \wedge \neg B)$

39

Rules for Semantic Tableau

- α-rules (AND-rules)

$$\frac{U \cup \{\alpha\}}{U \cup \{\alpha_1, \alpha_2\}}$$

- ∀-rules (Forall-rule)

$$\frac{U \cup \{\forall x\, A(x)\}}{U \cup \{\forall x\, A(x), A(c)\}} \qquad \frac{U \cup \{\neg\exists x\, A(x)\}}{U \cup \{\neg\exists x\, A(x), \neg A(c)\}}$$

c is an existing constant

- β-rules (OR-rules)

$$\frac{U \cup \{\beta\}}{U \cup \{\beta_1\} \qquad U \cup \{\beta_2\}}$$

- ∃-rules (Exist-rule)

$$\frac{U \cup \{\exists x\, A(x)\}}{U \cup \{A(c)\}} \qquad \frac{U \cup \{\neg\forall x\, A(x)\}}{U \cup \{\neg A(c)\}}$$

c is a new constant

40

Example of Semantic Tableau

• $\neg ((\forall x\, p(x) \vee q(x)) \rightarrow (\forall x\, p(x)) \vee (\exists x\, q(x)))$	$\alpha\ \neg\rightarrow$
• 1: $(\forall x\, p(x) \vee q(x)), \neg ((\forall x\, p(x)) \vee (\exists x\, q(x)))$	$\alpha\ \neg \vee$
• 11: $(\forall x\, p(x) \vee q(x)), \neg\forall x\, p(x), \neg\exists x\, q(x)$	$\neg\forall$
• 111: $(\forall x\, p(x) \vee q(x)), \neg p(c), \neg\exists x\, q(x)$	$\neg\exists$
• 1111: $(\forall x\, p(x) \vee q(x)), \neg p(c), \neg\exists x\, q(x), \neg q(c)$	$\forall$
• 11111: (1), $p(c) \vee q(c), \neg p(c), \neg\exists x\, q(x), \neg q(c)$	$\beta\ \vee$
• 111111: (1), $p(c), \neg p(c), \neg\exists x\, q(x), \neg q(c)$	closed
• 111112: (1), $q(c), \neg p(c), \neg\exists x\, q(x), \neg q(c)$	closed

- $\neg ((\forall x\, p(x) \vee q(x)) \rightarrow (\forall x\, p(x)) \vee (\exists x\, q(x)))$ is unsatisfiable
- $(\forall x\, p(x) \vee q(x)) \rightarrow (\forall x\, p(x)) \vee (\exists x\, q(x))$ is valid

41

Example of Semantic Tableau

(a)$\forall x \exists y$ p(x,y), (b) $\forall x,y,z$, p(x,y)$\wedge$p(y,z) $\rightarrow$ p(x,z), (c) $\forall x$ $\neg$p(x,x)
What positive literals can be derived from (a), (b), (c) ?

(a), (b), (c)

- 1: (a), (b), (c), $\exists y$ p(c0, y),
- 11: (a), (b), (c), p(c0, c1)
- 111: (a), (b), (c), p(c0, c1), $\exists y$ p(c1, y)
- 1111: (a), (b), (c), p(c0, c1), p(c1, c2)
- 11111: (a), (b), (c), p(c0, c1), p(c1, c2), $\exists y$ p(c2, y)
- …[by (a)]
- 111…1: (a), (b), (c), p(c0, c1), p(c1, c2), …, p(c_{k-1}, c_k)
- …[by (b)]
- 1111…1: (a), (b), (c), p(c0, c1), p(c1, c2), …, p(c0, c2), p(c0, c_k), p(c1, c3), …, p(c_{k-1}, c_k), p(c_{k-2}, c_k)

No finite models available. Only infinite models

42

Infinite Models

- A model I = (D, R, G) is infinite if D is infinite.

(a)$\forall x \exists y$ p(x,y), (b) $\forall x,y,z$, p(x,y)$\wedge$p(y,z) $\rightarrow$ p(x,z), (c) $\forall x$ $\neg$p(x,x)

- The above three formulas have an infinite model:
- I = (N, { < }, {})
- (a) For every x, there exists y, x < y;
- (b) “<” is transitive;
- (c) “<” is irreflexive, i.e., (x < x) is false for any x.

43

Prenex Normal Form

- A formula containing no quantifiers at all, or
- A formula of the form

$$Q_1x_1\ Q_2x_2 \ldots Q_nx_n\ P$$

where Q_i are either the universal or existential quantifier, x_i are variables and P is free of quantifiers.

e.g., $\exists x\ \forall y\ (p(x) \rightarrow q(y))$.

51

Conversion to Prenex Normal Form

1. Replace implications, biconditionals, etc., by and-or-negation. E.g., $(A \rightarrow B)$ by $(\neg A \vee B)$
2. Move $\neg$ "inwards" until there are no quantifiers in the scope of a negation, by deMorgan's laws.
3. Rename variables so each variable following a quantifier has a unique name.
4. Move quantifiers to the front of the sentence, without changing their order.

- Prenex normal forms are not unique

52

Example of Prenex NF

$\forall x\, ((C(x) \wedge \exists y\, (T(y) \wedge L(x, y))) \rightarrow \exists y\, (D(y) \wedge B(x, y)))$

$\forall x\, (\neg(C(x) \wedge \exists y\, (T(y) \wedge L(x, y))) \vee \exists z\, (D(z) \wedge B(x, z)))$

$\forall x\, (\neg \exists y\, (C(x) \wedge T(y) \wedge L(x, y)) \vee \exists z\, (D(z) \wedge B(x, z)))$

$\forall x \forall y\, (\neg(C(x) \wedge T(y) \wedge L(x, y)) \vee \exists z\, (D(z) \wedge B(x, z)))$

$\forall x \forall y \exists z\, (\neg(C(x) \wedge T(y) \wedge L(x, y)) \vee (D(z) \wedge B(x, z)))$

If you want to restore the implication:

$\forall x \forall y \exists z\, (C(x) \wedge T(y) \wedge L(x, y)) \rightarrow (D(z) \wedge B(x, z))$

Another prenex normal form is:

$\forall x \exists z \forall y\, (C(x) \wedge T(y) \wedge L(x, y)) \rightarrow (D(z) \wedge B(x, z))$

53

Skolemization: Removal of Quantifiers

1. Obtain a prenex NF $B = Q_1x_1\, Q_2x_2 \dots Q_nx_n\, P$
2. For j := 1 to n do
3. If (Q_j is $\forall$) remove $Q_j\, x_j$ from B
4. If (Q_j is $\forall\exists$) remove $Q_j\, x_j$ and replace x_j by f(V), where V is the set of free variables in B

Example: $A = \forall x \exists z \forall y\, (C(x) \wedge T(y) \wedge L(x,y)) \rightarrow (D(z) \wedge B(x,z))$

B := A

1. $B := \exists z \forall y\, (C(x) \wedge T(y) \wedge L(x, y)) \rightarrow (D(z) \wedge B(x, z))$
2. $B := \forall y\, (C(x) \wedge T(y) \wedge L(x, y)) \rightarrow (D(f(x)) \wedge B(x, f(x)))$
3. $B := (C(x) \wedge T(y) \wedge L(x, y)) \rightarrow (D(f(x)) \wedge B(x, f(x)))$

- **Theorem**: $A \approx B$, i.e., A and B are equally satisfiable.

54

CNF: Conjunction Normal Forms

1. Obtain a PNF of A: B = $Q_1x_1\ Q_2x_2 \ldots Q_nx_n$ P
2. Remove quantifiers by Skolemization
3. Convert the formal into CNF as in PL

Example:

- A =$\forall x \exists z \forall y$ (C(x) $\wedge$ T(y) $\wedge$ L(x, y)) $\rightarrow$ (D(z) $\wedge$ B(x, z))
- B = (C(x) $\wedge$ T(y) $\wedge$ L(x, y)) $\rightarrow$ (D(f(x)) $\wedge$ B(x, f(x)))
- C = { (-C(x) | -T(y) | -L(x, y) | D(f(x)),
  (-C(x) | -T(y) | -L(x, y) | B(x, f(x)) }

- **Theorem**: A $\approx$ C, i.e., A and C are equally satisfiable.

55

Converting formulas to CNF

- No need to convert to PNF

1. Obtain NNF (negation normal form) A
   a. Get rid of $\leftrightarrow$ or $\oplus$
   b. Get rid of $\rightarrow$
   c. Push $\neg$ downward
2. Remove quantifiers by Skolemization to get B
   a. Rename quantified variables
   b. Replace existentially quantified variables by Skolem constants/functions.
   c. Discard all universal quantifiers
3. Convert B into clause set C
   a. Convert B into CNF
   b. Convert CNF into clause set
   c. Standardize the variables in clauses

56

Converting formulas to CNF

1a. Eliminate all $\leftrightarrow$ connectives

$(P \leftrightarrow Q) \Rightarrow ((P \rightarrow Q) \wedge (Q \rightarrow P))$

1b. Eliminate all $\rightarrow$ connectives

$(P \rightarrow Q) \Rightarrow (\neg P \vee Q)$

1c. Reduce the scope of each negation symbol to a single predicate

$\neg\neg P \Rightarrow P$

$\neg(P \vee Q) \Rightarrow \neg P \wedge \neg Q$

$\neg(P \wedge Q) \Rightarrow \neg P \vee \neg Q$

$\neg\forall x\, P \Rightarrow \exists x\, \neg P$

$\neg\, \exists x\, P \Rightarrow \forall x\, \neg P$

57

Converting formulas to clausal form
Skolem constants and functions

2a. Standardize variables: rename all variables so that each quantifier has its own unique variable name

2b. Eliminate existential quantification by introducing Skolem constants/functions

$\exists x\, P(x) \Rightarrow P(C)$

C is a Skolem constant (a brand-new constant symbol that is not used in any other sentence)

$\forall x\, \exists y\, P(x,y) \Rightarrow \forall x\, P(x, f(x))$

since $\exists$ is within scope of a universally quantified variable, use a **Skolem function f** to construct a new value that **depends on** the universally quantified variable.

f must be a brand-new function name not occurring anywhere

58

Converting formulas to clausal form

2c. Remove universal quantifiers by (1) moving them all to the left end; (2) making the scope of each the entire sentence; and (3) dropping the "prefix" part

Ex: $\forall x\ P(x) \Rightarrow P(x)$

3a. Put into conjunctive normal form (conjunction of disjunctions) using distributive and associative laws

$(P \wedge Q) \vee R \Rightarrow (P \vee R) \wedge (Q \vee R)$

$(P \vee Q) \vee R \Rightarrow (P \vee Q \vee R)$

3b. Split conjuncts into separate clauses

3c. Standardize variables so each clause contains only variable names that do not occur in any other clause

59

An example

$\forall \mathbf{x}\ (\mathbf{P(x)} \rightarrow ((\forall \mathbf{y})(\mathbf{P(y)} \rightarrow \mathbf{P(f(x,y))}) \wedge \neg(\forall \mathbf{y})(\mathbf{Q(x,y)} \rightarrow \mathbf{P(y)})))$

1a. Eliminate $\leftrightarrow$

1b. Eliminate $\rightarrow$

$\forall x\ (\neg P(x) \vee (\forall y\ (\neg P(y) \vee P(f(x, y))) \wedge \neg\forall y\ (\neg Q(x, y) \vee P(y))))$

1c Reduce scope of negation

$\forall x\ (\neg P(x) \vee (\forall y\ (\neg P(y) \vee P(f(x, y))) \wedge \exists y\ (Q(x, y) \wedge \neg P(y))))$

2a. Standardize variables

$\forall x\ (\neg P(x) \vee (\forall y\ (\neg P(y) \vee P(f(x, y))) \wedge \exists z\ (Q(x, z) \wedge \neg P(z))))$

2b. Eliminate existential quantification

$\forall x\ (\neg P(x) \vee(\forall y\ (\neg P(y) \vee P(f(x, y))) \wedge(Q(x, g(x)) \wedge \neg P(g(x)))))$

2c. Drop universal quantification symbols

$(\neg P(x) \vee ((\neg P(y) \vee P(f(x, y))) \wedge (Q(x, g(x)) \wedge \neg P(g(x)))))$

60

An Example (continued)

3a. Convert to conjunction of disjunctions

$(\neg P(x) \mid \neg P(y) \mid P(f(x,y))) \wedge (\neg P(x) \mid Q(x,g(x))) \wedge$
$(\neg P(x) \mid \neg P(g(x)))$

3b. Create separate clauses

$(\neg P(x) \mid \neg P(y) \mid P(f(x,y)))$
$(\neg P(x) \mid Q(x, g(x)))$
$(\neg P(x) \mid \neg P(g(x)))$

3c. Standardize variables

$(\neg P(x) \mid \neg P(y) \mid P(f(x,y)))$
$(\neg P(z) \mid Q(z, g(z)))$
$(\neg P(w) \mid \neg P(g(w)))$

61

Colonel West is a criminal

1. It is a crime for an American to sell weapons to a hostile country.
2. The country Nono has some missiles.
3. All of its missiles were sold to it by Colonel West.
4. Nono is an enemy of USA.
5. Colonel West is an American.

65

Modeling with Horn Clauses: at most one positive literal

$(\neg A_1 \mid \neg A_2 \mid \neg A_3 \mid \neg A_4 \mid B)$ as $A_1 \wedge A_2 \wedge A_3 \wedge A_4 \rightarrow B$

1. It is a crime for an American to sell weapons to a hostile country.

   American(x) $\wedge$ Weapons(y) $\wedge$ Hostile(z) $\wedge$ Sell(x,y,z) $\rightarrow$ Criminal (x)

2. The country Nono has some missiles.

   // $\exists$ x Owns(Nono, x) $\wedge$ Missile(x)

   Missile(M1) // Skolem Constant introduction

   Owns(Nono, M1)

66

Modeling with Horn Clauses: at most one positive literal

3. All of its missiles were sold to it by Colonel West.

   Missile(x) $\wedge$ Owns(Nono,x) $\rightarrow$ Sells(West,x,Nono).

4. Nono is an enemy of USA.

   Enemy(Nono,American).

5. Colonel West is an American.

   American(West).

// common sense

Missile(x) $\rightarrow$ Weapon(x)

Enemy(x, America) $\rightarrow$ Hostile(x)

67