A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
- Ex:

$\forall x$ At(x,CU) $\wedge$ Smart(x)
means "Everyone is at CU and everyone is smart"

Yet to say Everyone at CU is smart

$\forall x$ At(x,CU) $\Rightarrow$ Smart(x)

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \text{At}(x,\text{CU}) \Rightarrow \text{Smart}(x)$$

is true if there is anyone who is smart not at CU.

Yet to say: there exists someone in CU that is smart

$$\exists x \text{At}(x,\text{CU}) \wedge \text{Smart}(x)$$

Translating English to FOL

- Every gardener likes the sun.

  `(∀x) gardener(x) => likes(x,Sun)`

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

  `(∃x) person(x) ^ ((∀t) time(t)) => can-fool(x,t))`

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

  `(∀x) person(x) => ((∃t) time(t) ^ can-fool(x,t))`

- All purple mushrooms are poisonous.

  `(∀ x) (mushroom(x) ^ purple(x)) => poisonous(x)`

- No purple mushroom is poisonous.

  `~(∃x) purple(x) ^ mushroom(x) ^ poisonous(x)`
  or, equivalently,

  `(∀x) (mushroom(x) ^ purple(x)) => ~poisonous(x)`

Inference in FOL chapter 9 in Russel

- ***KB*** $\vdash_i$ α = sentence α can be derived from ***KB*** by procedure ***i***

i.e. deriving sentences from other sentences

- Soundness: ***i*** is sound if whenever ***KB*** $\vdash_i$ α, it is also true that ***KB*** $\models$ α

i.e. derivations produce only entailed sentences ***(no wrong inferences, but maybe not all inferences)***

- Completeness: ***i*** is complete if whenever ***KB*** $\models$ α, it is also true that ***KB*** $\vdash_i$ α

i.e. derivations can produce all entailed sentences (***all inferences can be made, but maybe some wrong extra ones as well)***

Validity and satisfiability

- A sentence is **valid** if it is true in **all models**,
- e.g., ***True***, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the following:
KB $\models \alpha$ if and only if (***KB*** $\Rightarrow \alpha$) is valid

A sentence is **satisfiable** if it is true in **some model**
e.g., $A \vee B$, C

A sentence is **unsatisfiable** if it is true in **no models**
e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
KB $\models \alpha$ if and only if (***KB*** $\wedge \neg \alpha$) is unsatisfiable
(there is no model for which KB=true and is false)

Proof Methods in FOL

Major Families:

- GMP
- Reduction
- Resolution
- Forward chaining
- Backward chaining

Some Other inference tools:

Entailment/ Unification/

Proof Methods in FOL

- GMP: Using the generalized form of Modus Ponense
- Reduction: Reduce all FOL sentences to propositional Calculus then use inference in propositional calculus
- Resolution – Refutation
  - Negate goal
  - Convert all pieces of knowledge into clausal form (disjunction of literals)
  - See if contradiction indicated by null clause $\square$ can be derived
- Forward chaining
  - Given P, $P \rightarrow Q$, to infer Q
  - P, match $L.H.S$ of
  - Assert Q from $R.H.S$
- Backward chaining
  - Q, Match $R.H.S$ of $P \rightarrow Q$
  - assert P
  - Check if P exists

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable v and ground term g

- E.g., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

Existential instantiation (EI)

- For any sentence α, variable ***v***, and constant symbol ***k*** that does not appear elsewhere in the knowledge base:

$$\frac{\exists v \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x\ Crown(x) \wedge OnHead(x, John)$ yields:

$$Crown(C_1) \wedge OnHead(C_1, John)$$

provided C_1 is a new constant symbol, called a Skolem constant

Unification

- $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ	
Knows(John,x)	Knows(John,Jane)	{x/Jane}}	
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}	
Knows(John,x)	Knows(y,Mother(y))		
Knows(John,x)	Knows(x,OJ)		

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,

  θ = {y/John, x/z } or θ = {y/John, x/John, z/John} or others…
- There are many possible unifiers for some atomic sentences. The first unifier is more general than the second.
- The UNIFY algorithm returns the most general unifier (MGU) that is unique up to renaming of variables. MGU makes the least commitment to variable values.

The Unification Algorithm

•In order to match sentences in the KB, we need a routine.
•UNIFY(p,q) takes two atomic sentences and returns a substitution that makes them equivalent.
UNIFY(p,q)= θ where SUBST(θ,p)=SUBST(θ,q) θ is called a unifier.

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound
          y, a variable, constant, list, or compound
          θ, the substitution built up so far

  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
      return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
  else if LIST?(x) and LIST?(y) then
      return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure
```

The Unification Algorithm

```
function UNIFY-VAR(var, x, θ) returns a substitution
  inputs: var, a variable
          x, any expression
          θ, the substitution built up so far

  if {var/val} ∈ θ then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

Inference Rules for Quantifiers

- **Universal Elimination:** "$\forall v \alpha$ |- SUBST({v/g}, α)"

  for any sentence, α, variable, *v, and ground term, g*

- $\forall$ x Study(x, AI) |- Study(Mary, AI)

- **Existential Elimination:** "$\exists v$ α |- SUBST({v/k},a)"

  for any sentence, α, variable, *v, and constant symbol, k,* that doesn't occur elsewhere in the KB (Skolem constant)

- $\exists$x (Owns(Mary,x) $\wedge$ Cat(x)) |- Owns(Mary,Jusy)$\wedge$ Cat(Jusy)

- **Existential Introduction:** "α |- $\exists$ v SUBST({*g/v*}, α)"

  for any sentence, α, variable, *v, that does not occur in* α, and ground term, *g, that does occur in* α

- Study(Mary, AI) |- $\exists$ x Study(x, AI)

Proof Example

1) $\forall$ x,y(Parent(x,y) $\wedge$ Male(x) $\Rightarrow$ Father(x,y))

2) Parent(Tom,John)

3) Male(Tom) Using Universal Elimination from 1)

4) $\forall$ y(Parent(Tom,y) $\wedge$ Male(Tom) $\Rightarrow$ Father(Tom,y))
Using Universal Elimination from 4)

5) Parent(Tom,John) $\wedge$ Male(Tom) $\Rightarrow$ Father(Tom,John)
Using And Introduction from 2) and 3)

6) Parent(Tom,John) $\wedge$ Male(Tom)
Using Modes Ponens from 5) and 6)

7) Father(Tom,John)

Generalized Modus Ponens (GMP)

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

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

Ex.:

1) $\forall x,y(Parent(x,y) \wedge Male(x) \Rightarrow Father(x,y))$

2) Parent(Tom,John)

3) Male(Tom)

q={x/Tom, y/John)

4) Father(Tom,John)

Generalized Modus Ponens (GMP)

- In order to Apply generalized Modus Ponens, all sentences in the KB must be in the form of **Horn Clauses:**
- where a clause is a disjunction of literals, because they can be rewritten as disjunctions with at most one non-negated literal.

$\forall v_1, v_2, \ldots, v_n (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$ can be expressed as

$\forall v_1, v_2, \ldots, v_n \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q$

- If we have exactly one definite clause, the sentence is called a **definite clause**
- Quantifiers can be dropped since all variables can beassumed to be universally quantified by default.
- Many sentences can be transformed into Horn clauses, but not all (e.g. $P(x) \vee Q(x)$, and $\neg P(x)$)

Resolution in PC

- **Propositional version.**

$\{a \vee b, \neg b \vee g\} \vdash a \vee g$ **OR** $\{\neg a \Rightarrow b, b \Rightarrow g\} \vdash \neg a \Rightarrow g$

- Reasoning by cases **OR** transitivity of implication
- **First-order form**
- For two literals p_k and q_l in two clauses

$p_1 \vee p_2 \vee \ldots \vee p_n$

$q_1 \vee q_2 \vee \ldots \vee q_n$

such that $\theta = \text{UNIFY}(p_k, \neg q_l)$, derives

$\text{SUBST}(\theta, p_1 \vee p_2 \vee \ldots p_{k-1} \vee p_{k+1} \vee \ldots \vee p_n \vee q_1 \vee q_2 \vee \ldots q_{l-1} \vee q_{l+1} \vee \ldots \vee q_n)$

- For resolution to apply, all sentences must be in **conjunctive normal form,**

Conjunction Normal Form (CNF) in PC

We like to prove: $KB \models \alpha$ *equivalent to* : $KB \wedge \neg\alpha$ *unsatifiable*

We first rewrite $KB \wedge \neg\alpha$ into conjunctive normal form (CNF).

A "conjunction of disjunctions" — literals

$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Clause Clause

In theory

• Any KB can be converted into CNF.

• In fact, any KB can be converted into CNF-3, i.e. using clauses with at most 3 literals.

Example: Conversion to CNF in PC

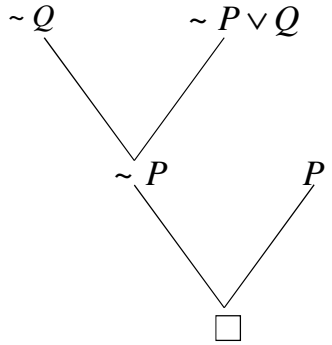
$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. Move $\neg$ inwards using de Morgan's rules and double-negation: $\neg(\alpha \vee \beta) = \neg\alpha \wedge \neg\beta$
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Apply distributive law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution in PC

1. P
2. $P \rightarrow Q$ converted to $\sim P \vee Q$
3. $\sim Q$

Draw the resolution tree (actually an inverted tree). Every node is a clausal form and branches are intermediate inference steps.



Resolution Algorithm in PC

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- Generate all new sentences from KB and the query.
- One of two things can happen:

1. We find $P \wedge \neg P$ which is unsatisfiable. i.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \wedge \neg\alpha$ (non-trivial) and hence we cannot entail the query.