First-Order Logic

Chapters 8.1 – 8.3 and 9

(not responsible for Chapter 9 on the Final Exam)

General Logic

Logics are characterized by
what they commit to as "primitives"

Logic	What Exists in World	Knowledge States
Propositional	facts	true/false/unknown
First-Order	facts, objects, relations	true/false/unknown
Temporal	facts, objects, relations, times	true/false/unknown
Probability Theory	facts	degree of belief 0..1
Markov	facts, objects, relations	degree of belief 0..1

FOL Syntax: Basic

- A **term** is used to denote an object in the world
  - **constant**: `BobSmith`, `2`, `Madison`, `Green`, …
  - **variable**: `x`, `y`, `a`, `b`, `c`, …
  - **function(term$_1$, …, term$_n$)**:
    e.g., `Sqrt(9)`, `Distance(Madison, Milwaukee)`
    - is a relation for which there is one answer
    - maps one or more objects to another *single object*
    - can be used to refer to an unnamed object:
      e.g., `LeftLegOf(John)`
    - represents a user-defined *functional* relation
    - cannot be used with logical connectives
- A **ground term** is a term with no variables

FOL Syntax: Basic

- An **atom** is smallest expression
  to which a truth value can be assigned
  - **predicate(term$_1$, …, term$_n$)**:
    e.g., `Teacher(John, Deb)`, `≤(Sqrt(2), Sqrt(7))`
    - is a relation for which there may be more than one answer
    - maps one or more objects to a *truth value*
    - represents a user defined relation
  - **term$_1$ = term$_2$**:
    e.g., `Income(John) = 20K`, `1 = 2`
    - represents the ***equality*** relation when two terms refer to the same object
    - is a predicate in prefix form: **=(term$_1$, term$_2$)**

FOL Syntax: Basic

- A **sentence** represents a fact in the world that is assigned a truth value
  - atom
  - complex sentence using connectives: $\wedge \vee \neg \Rightarrow \Leftrightarrow$
    - e.g., `Friend(Deb,Jim)` $\Rightarrow$ `Friend(Jim,Deb)`
    - e.g., `>(11,22)` $\wedge$ `<(22,33)`
  - complex sentence using quantified variables: $\forall$ $\exists$

FOL Semantics: Assigning Truth

- The atom predicate($\text{term}_1, \ldots, \text{term}_n$) is true iff the objects referred to by $\text{term}_1, \ldots, \text{term}_n$ are in the relation referred to by the predicate
- What is the truth value for `F(D, J)`?
  - model:
    - **objects: Deb, Jim, Sue, Bob**
    - **relation: Friend {<Deb,Sue>,<Sue,Deb>}**
  - interpretation:
    - **`D` means Deb, `J` means Jim, `S` means Sue, `B` means Bob**
    - **`F(term`$_1$`,term`$_2$`)` means term_1 is friend of term_2**

FOL Syntax: Quantifiers

Universal quantifier: $\forall$`<variable> <sentence>`

- Means the sentence is true **for all** values of *x* in the domain of variable *x*
- Main connective typically $\Rightarrow$ forming if-then rules
  - ***All humans are mammals*** **becomes in FOL**
    - **$\forall$`x Human(x)` $\Rightarrow$ `Mammal(x)`**
    - i.e., for all *x*, if *x* is a human then *x* is a mammal
  - **Mammals must have fur becomes in FOL**
    - **$\forall$`x Mammal(x)` $\Rightarrow$ `HasFur(x)`**
    - for all *x*, if *x* is a mammal then *x* has fur

FOL Syntax: Quantifiers

$\forall x$ `(Human(`x`)` $\Rightarrow$ `Mammal(`x`))`

- Equivalent to the conjunction of instantiations of *x*:

```
(Human(Jim) ⇒ Mammal(Jim)) ∧
(Human(Deb) ⇒ Mammal(Deb)) ∧
(Human(22)  ⇒ Mammal(22) ) ∧ ...
```

FOL Syntax: Quantifiers

- Common mistake is to use ∧ as main connective
  - results in a blanket statement about *everything*
- For example: `∀x (Human(x) ∧ Mammal(x))`
  - **`(Human(Jim) ∧ Mammal(Jim)) ∧`**
    `(Human(Deb) ∧ Mammal(Deb)) ∧`
    `(Human(22)  ∧ Mammal(22) ) ∧ …`
  - means everything is human and a mammal

FOL Syntax: Quantifiers

Existential quantifier: `∃<variable> <sentence>`

- Means the sentence is true
  for some value of *x* in the domain of variable *x*
- Main connective is typically ∧
  - **Some humans are old** **becomes in FOL**
  - **`∃x Human(x) ∧ Old(x)`**
    there exist an *x* such that *x* is a human and *x* is old
  - **Mammals may have arms.** **becomes in FOL**
  - **`∃x Mammal(x) ∧ HasArms(x)`**
    there exist an x such that x is a mammal and x has arms

FOL Syntax: Quantifiers

`∃x (Human(x) ∧ Old(x))`

- Equivalent to the disjunction of instantiations of *x*:

```
(Human(Jim) ∧ Old(Jim)) ∨
(Human(Deb) ∧ Old(Deb)) ∨
(Human(22)  ∧ Old(22) ) ∨ …
```

FOL Syntax: Quantifiers

- Common mistake is to use ⇒ as main connective
  - results in a weak statement
- For example: `∃x (Human(x) ⇒ Old(x))`
  - **`(Human(Jim) ⇒ Old(Jim)) ∨`**
    `(Human(Deb) ⇒ Old(Deb)) ∨`
    `(Human(22)  ⇒ Old(22) ) ∨ …`
  - true if there is *anything* that *isn't* human

FOL Syntax: Quantifiers

- Properties of quantifiers:
  - ∀x ∀y is the same as ∀y ∀x
  - ∃x ∃y is the same as ∃y ∃x
  - note: ∃x ∃y can be written as ∃x,y likewise with ∀
- Examples
  - `∀x ∀y Likes(x,y)` is *active voice*:
    Everyone likes everyone.
  - `∀y ∀x Likes(x,y)` is *passive voice*:
    Everyone is liked by everyone.

FOL Syntax: Quantifiers

- Properties of quantifiers:
  - ∀x ∃y is ***not*** the same as ∃y ∀x
  - ∃x ∀y is ***not*** the same as ∀y ∃x
- Examples
  - `∀x ∃y Likes(x,y)` is *active voice*:
    Everyone likes someone.
  - `∃y ∀x Likes(x,y)` is *passive voice*:
    Someone is liked by everyone.

FOL Syntax: Quantifiers

- Properties of quantifiers:
  - `∀x P(x)` is the same as `¬∃x ¬P(x)`
  - `∃x P(x)` is the same as `¬∀x ¬P(x)`
- Examples
  - `∀x Likes(x,IceCream)`
    Everyone likes ice cream.
  - `¬∃x ¬Likes(x,IceCream)`
    No one doesn't like ice cream.
    It's a double negative!

FOL Syntax: Quantifiers

- Properties of quantifiers:
  - `∀x P(x)` when negated is `∃x ¬P(x)`
  - `∃x P(x)` when negated is `∀x ¬P(x)`
- Examples
  - `∀x Likes(x,IceCream)`
    Everyone likes ice cream.
  - `∃x ¬Likes(x,IceCream)`
    Someone doesn't like ice cream.
  - This is from the application of de Morgan's law to the fully instantiated sentence

FOL Syntax: Basics

- A **free variable** is a variable that isn't bound by a quantifier
  - `∃y Likes(x,y)`

    x is free, **y** is bound
- A **well-formed formula** is a sentence where all variables are quantified

Summary so Far

• Constants:	Bob, 2, Madison, …
• Functions:	Income, Address, Sqrt, …
• Predicates:	Sister, Teacher, <=, …
• Variables:	x, y, a, b, c, …
• Connectives:	$\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
• Equality:	$=$
• Quantifiers:	$\forall$ $\exists$

Summary so Far

- **Term**: Constant, variable, or Function(term_1, …, term_n)

  denotes an object in the world

  Ground Term has no variables
- **Atom**: Predicate(term_1, …, term_n), $\text{term}_1 = \text{term}_2$

  is smallest expression assigned a truth value
- **Sentence**: atom, quantified sentence with variables or complex sentence using connectives is assigned a truth value
- **Well-Formed Formula** (wff):

  sentence where all variables are quantified

Fun with Sentences

Convert the following English sentences into FOL

- **Bob is a fish.**
  - What are the objects?

    Bob
  - What are the relations?

    is a fish

  Answer: `Fish(Bob)` **a unary relation or property**
- **Deb and Sue are women.** **we'll be casual about plurals**
- **Deb or Sue isn't a plant.** **ambiguous?**
- **Deb and Sue are friends.** **use a function? predicate?**

Fun with Sentences

Convert the following English sentences into FOL

- **America bought Alaska from Russia.**
  - What are the objects?
    America, Alaska, Russia
  - What are the relations?
    bought(who, what, from) – an *n*-ary relation where *n* is 3

  Answer: `Bought(America, Alaska, Russia)`
- **Warm is between cold and hot.**
- **Deb, Lynn, Jim, and Steve went together to APT.**

Fun with Sentences

Convert the following English sentences into FOL

- **Jim collects everything.**
  - What are the variables?
    everything ***x***
  - How are they quantified?
    all, universal

  Answer: $\forall$`x Collects(Jim,x)`

  `Collects(Jim,Pencil)` $\wedge$ `Collects(Jim,Deb)` $\wedge$ …
- **Jim collects something.**
- **Somebody collects Jim.** **How do you handle "body"?**

Fun with Sentences

When to restrict the domain, e.g., to people:

- All: $\forall$x (Person(x) $\wedge$ ...) $\Rightarrow$ ...
  - **things**: anything, everything, whatever
  - **people**: anybody, anyone, everybody, everyone, whoever
- Some (at least one): $\exists$x Person(x) $\wedge$ ... $\wedge$ ...
  - **things**: something
  - **people**: somebody, someone
- None: $\neg\exists$x Person(x) $\wedge$ ... $\wedge$ ...
  - **things**: nothing
  - **people**: nobody, no one

Fun with Sentences

Convert the following English sentences into FOL

- **Somebody collects something.**
  - What are the variables?
    somebody **x** and something **y**
  - How are they quantified?
    at least one, existential

  Answer: $\exists$`x,y Person(x)` $\wedge$ `Collects(x,y)`
- **Everybody collects everything.**
- **Everybody collects something.**
- **Something is collected by everybody.**

Fun with Sentences

Convert the following English sentences into FOL

- **Nothing collects anything.**
  - What are the variables and quantifiers?
    nothing **x** and anything **y**
    not one (i.e., not existential) and all (universal)
    Answer: `¬∃x ∀y Collects(x,y)`
  - Equivalent?
    Everything does not collect anything.
    Answer: `∀x,y ¬Collects(x,y)`
- **Everything collects nothing.**

Fun with Sentences

Convert the following English sentences into FOL

- **All hoarders collect everything.**
  - How are ideas connected?
    being a hoarder implies collecting everything
  Answer: `∀x,y Horder(x) ⇒ Collects(x,y)`
- **Hoarders collect valuable things.** **Ambiguous:**
  - **All hoarders collect all valuable things.**
  - **All hoarders collect some valuable things.**
  - **Some hoarders collect all valuable things.**
  - **Some hoarders collect some valuable things.**

Fun with Sentences

Convert the following English sentences into FOL

- **All stinky shoes are allowed.**
  - How are ideas connected?
    being a shoe and stinky implies it is allowed
  Answer: `∀x (Shoe(x) ∧ Stinky(x)) ⇒ Allowed(x)`
- **No stinky shoes are allowed.** **Is this negative of above?**
  Answer: `¬∃x Shoe(x) ∧ Stinky(x) ∧ Allowed(x)`
  - Equivalent (carry negation through)?
    (All) Stinky shoes are not allowed.
    Answer: `∀x (Shoe(x) ∧ Stinky(x)) ⇒ ¬Allowed(x)`

Fun with Sentences

Convert the following English sentences into FOL

- **Any good amateur can beat some professional.**
  1. $\forall x$ [(x is a good amateur) $\Rightarrow$
     (x can beat some professional)]
  2. (x can beat some professional) becomes
     $\exists y$ [(y is a professional) $\wedge$ (x can beat y)]

  Answer: `∀x [(Amateur(x) ∧ GoodPlayer(x)) ⇒`
  `∃y (Professional(y) ∧ Beat(x,y))]`
- **Some professionals can beat all amateurs.**

Fun with Sentences

Convert the following English sentences into FOL

- **There is exactly one shoe.**
  Answer: $\exists x\ Shoe(x) \wedge \forall y(Shoe(y) \Rightarrow (x=y))$

- **There are exactly two shoes.**
  - Are quantities specified?
  - Are equalities implied?

  Answer: $\exists x,y\ Shoe(x) \wedge Shoe(y) \wedge \neg(x=y) \wedge$
  $\forall z\ (Shoe(z) \Rightarrow (x=z) \vee (y=z))$

Fun with Sentences

- Interesting words: ***always***, ***sometimes***, ***never***
  - Good people always have friends.
    could mean: All good people have friends.
    $\forall x\ Person(x) \wedge Good(x) \Rightarrow \exists y(Friend(x,y))$
  - Busy people sometimes have friends.
    could mean: Some busy people have friends.
    $\exists x\ Person(x) \wedge Busy(x) \wedge \exists y(Friend(x,y))$
  - Bad people never have friends.
    could mean: Bad people have no friends.
    $\forall x\ Person(x) \wedge Bad(x) \Rightarrow \neg\exists y(Friend(x,y))$
    or equivalently: No bad people have friends.
    $\neg\exists x\ Person(x) \wedge Bad(x) \wedge \exists y(Friend(x,y))$

Fun with Sentences

- The "interesting words" are ambiguous in that they can quantify a variable as in the last slide, or they may refer to units of time

- Temporal indicators: while, when, whenever, …
  - are used to determine when "interesting words" refer to time
  - Jo always writes home when she is away from home.
  - $\forall x\ (Time(x) \wedge Away(Jo,Home,x)) \Rightarrow Writes(Jo,Home,x)$

Fun with Sentences

Convert the following English sentences into FOL

- **X is above Y if X is directly on the top of Y or else there is a pile of one or more other objects directly on top of another starting with X and ending with Y.**
  - Answer: $\forall x\ \forall y\ Above(x,y) \Leftrightarrow [OnTop(x,y) \vee$
    $\exists z(OnTop(x,z) \wedge Above(z,y))]$

- **Lincoln: "You can fool some of the people all of the time, and all of the people some of the time, but you cannot fool all of the people all of the time.“**
  - Answer: $(\exists x\ \forall t\ (person(x) \wedge time(t)) \Rightarrow CanFool(x,t)) \wedge$ …

Inference Rules for FOL

may be many ways to do this!

- **Universal Elimination (UE)**
  variable substituted with ground term
  ∀*x* ***Eats*(*Jim*, *x*)** infer ***Eats*(*Jim*, *Cake*)**

$$\frac{\forall v\ \alpha}{\mathrm{SUBST}(\{v/g\},\ \alpha)}$$

- **Existential Elimination (EE)**
  variable substituted with a new constant
  ∃*x* ***Eats*(*Jim*, *x*)** infer ***Eats*(*Jim*, *NewFood*)**

$$\frac{\exists v\ \alpha}{\mathrm{SUBST}(\{v/k\},\ \alpha)}$$

k is a new term

- **These two inference rules enable the knowledge base to be propositionalized**
- **Then natural deduction can be done using inference rules for PL**

A Simple FOL Proof using Natural Deduction

- Jim is a turtle.
  `1. Turtle(Jim)`
- Deb is a rabbit.
  `2. Rabbit(Deb)`
- Turtles outlast Rabbits.
  `3. ∀x,y (Turtle(x) ∧ Rabbit(y)) ⇒ Outlast(x,y)`
- Query: Jim outlasts Deb.
  – `Outlast(Jim,Deb)`

  - Treat predicates like propositional symbols

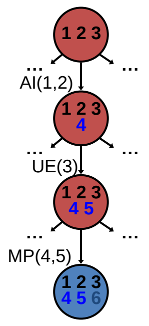
A Simple FOL Proof using Natural Deduction

- And Introduction: AI(1, 2)
  `4. Turtle(Jim) ∧ Rabbit(Deb)`
- Universal Elimination: UE(3, {x/Jim, y/Deb})
  `5. Turtle(Jim) ∧ Rabbit(Deb) ⇒ Outlast(Jim,Deb)`
- Modus Ponens: MP(4, 5)
  `6. Outlast(Jim,Deb)`
- AI, UE, MP is a common inference pattern
- Automated inference is harder with FOL than PL

  Variables can take on a potentially ***infinite*** number of possible values from their domain and thus UE can be applied in a potentially infinite number of ways to KB

Proof as Search using Inference Rules

- Operators are inference rules
- States are the KB
- Goal test checks if query in KB
- **Problem:**
  huge branching factor, especially for UE
- **Idea:**
  find a substitution that makes the rule premise match known facts
- Make a single powerful inference rule



Generalized Modus Ponens (GMP)

- "Unify" rule premises with known facts and apply unifier to conclusion
- Rule: `∀x,y (Turtle(x) ∧ Rabbit(y)) ⇒ Outlast(x,y)`

  Known facts: `Turtle(Jim), Rabbit(Deb)`

  Unifier: `{x/Jim, y/Deb}`
- Apply unifier to conclusion: `Outlast(Jim,Deb)`

Generalized Modus Ponens (GMP)

- Combines AI, UE, and MP into a single rule:

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

where $\mathrm{SUBST}(\theta, p_i') = \mathrm{SUBST}(\theta, p_i)$ for all i

- **$\mathrm{SUBST}(\theta, \alpha)$ means apply substitutions in θ to α**
- **Substitution list $\theta = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$ means**
  - replace all occurrences of variable v_i with term t_i
  - substitutions are made in left to right order

Generalized Modus Ponens (GMP)

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

where $\mathrm{SUBST}(\theta, p_i') = \mathrm{SUBST}(\theta, p_i)$ for all i

- **All variables *assumed* to be *universally* quantified**
- **Used with a KB in Horn normal form (HNF):**

  definite clause: disjunction of literals with exactly 1 positive literal
  - **fact**: single positive literal $\quad$ `P1(x), P2(x)`
  - **rule**: conjunction of atoms ⇒ atom $\quad$ `(P1(x) ∧ P2(x)) ⇒ Q(x)`

    has only one positive literal $\quad$ `¬P1(x) ∨ ¬P2(x) ∨ Q(x)`

Generalized Modus Ponens (GMP)

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

where $\mathrm{SUBST}(\theta, p_i') = \mathrm{SUBST}(\theta, p_i)$ for all i

Example:

p_1'	= `Smarter(Deb, Bob)`
p_2'	= `Smarter(Bob, Joe)`
$(p_1 \wedge p_2) \Rightarrow q$	= `(Smarter(x,y) ∧ Smarter(y,z)) ⇒ Smarter(x,z)`
θ	= $\{x/\mathrm{Deb}, y/\mathrm{Bob}, z/\mathrm{Joe}\}$
$\mathrm{SUBST}(\theta, q)$	= `Smarter(Deb, Joe)`

Unification

- Substitution θ **unifies** p_1' and p_1
  if $\text{SUBST}(\theta, p_1') = \text{SUBST}(\theta, p_1)$

p_1'	p_1	θ
`Turtle(y)`	`Turtle(Jim)`	{y/Jim}
`Hears(Deb,x)`	`Hears(Deb,Sue)`	{x/Sue}
`Hears(Deb,x)`	`Hears(`~~x~~`,Jim)`	{y/Deb, x/Jim}

- **Variables must be standardized apart!**

  I.e., if the same variable(s) is found in *both* p_1' and p_1
  then rename variable(s) so none are shared

Unification

- Substitution θ **unifies** p_1' and p_1
  if $\text{SUBST}(\theta, p_1') = \text{SUBST}(\theta, p_1)$

p_1'	p_1	θ
`Turtle(y)`	`Turtle(Jim)`	{y/Jim}
`Hears(Deb,x)`	`Hears(Deb,Sue)`	{x/Sue}
`Hears(Deb,x)`	`Hears(y,Jim)`	{y/Deb, x/Jim}
`Hears(Deb,x)`	`Hears(z,Mother(z))`	{z/Deb, x/Mother(Deb)}
`Eats(y,y)`	`Eats(z,Fish)`	{y/z, z/Fish}
`Sees(Jo,x,y)`	`Sees(z,Jim,At(z))`	{z/Jo, x/Jim, y/At(Jo)}
`Sees(x, ID(x),At(Jo))`	`Sees(Jim, ID(y),At(y))`	*failure*, assuming At(Jo) $\neq$ At(Jim)

Unification Algorithm

- Unify returns θ, a most general unifier (MGU)
  - shortest length substitution list to make a match
  - in general, more than one MGU
- AIMA algorithm recursively explores the two expressions and simultaneously builds θ
- Occurs-Check prevents a variable from replacing a term that contains that variable, e.g., prevents $\{x/F(x)\}$
  - this slows down the unify algorithm
- Unify with the occurs-check has a time complexity $O(n^2)$ where n is the size of the expressions

Completeness of Automated Inference

- **Truth table enumeration**: **incomplete for FOL**
  table may be infinite in size for infinite domain
- **Natural Deduction**: **complete for FOL**
  impractical since branching factor too large
- **GMP**: **incomplete for FOL**
  not every sentence can be converted to Horn form
- **GMP**: **complete for FOL KB in HNF (definite clauses)**
  - forward chaining: move from KB to query
  - backward chaining: move from query to KB

Forward Chaining (FC) with GMP

- Move "forward" from KB to query
- Simplified FC Algorithm (see Figure 9.3):
  Given: query *q* is asked of KB

```
repeat until no new sentences are inferred
    initialize NEW to empty
    for each rule that can have all of its premises satisfied
        apply composed substitution to the conclusion
        add the new conclusion to NEW if it's not just a renaming
        done if the new conclusion unifies with the query
    add sentences in NEW to KB
return false since q never concluded
```

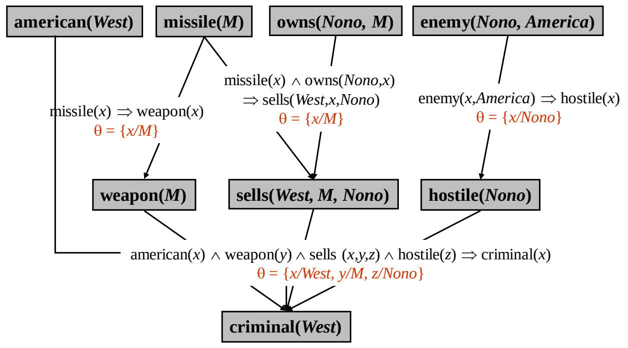
FOL Inference Example

"The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is an American."

KB:
1. american(x) $\wedge$ weapon(y) $\wedge$ sells (x,y,z) $\wedge$ hostile(z) $\Rightarrow$ criminal(x)
2. enemy(*Nono,America*)
$\exists x$ owns(*Nono*, x) $\wedge$ missile(x) ← *Must be in HNF!*
Use EE and generate 2 sentences
3. owns(*Nono, M*)
4. missile(*M*)
5. missile(x) $\wedge$ owns(*Nono*, x) $\Rightarrow$ sells(*West*, x, *Nono*)
6. american(*West*)
7. enemy(x, *America*) $\Rightarrow$ hostile(x)
8. missile(x) $\Rightarrow$ weapon(x)

Query: **criminal(*West*)** ?

Forward Chaining with GMP



american(West)
missile(M)
owns(Nono, M)
enemy(Nono, America)
missile(x) ⇒ weapon(x)
θ = {x/M}
missile(x) ∧ owns(Nono,x) ⇒ sells(West,x,Nono)
θ = {x/M}
enemy(x,America) ⇒ hostile(x)
θ = {x/Nono}
weapon(M)
sells(West, M, Nono)
hostile(Nono)
american(x) ∧ weapon(y) ∧ sells (x,y,z) ∧ hostile(z) ⇒ criminal(x)
θ = {x/West, y/M, z/Nono}
criminal(West)

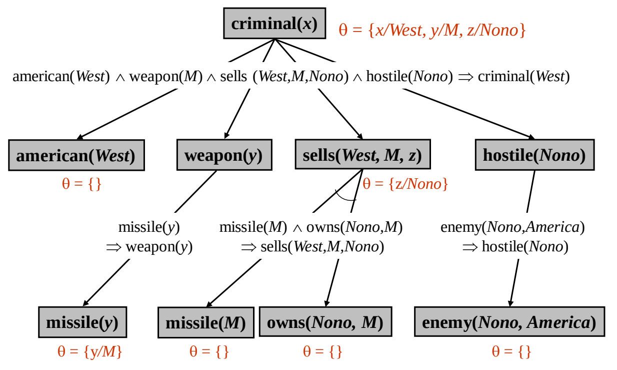
Backward Chaining (BC) with GMP

- Move "backwards" from query to KB
- Simplified BC Algorithm (see Figure 9.6):
  Given: query *q* is asked of KB

```
if a matching fact q' is known then return unifier        (base case)
for each rule whose consequent q' matches q               (recursive cases)
    attempt to prove each premise of the rule by BC        (depth first)
```

(Complication added to keep track of unifiers, i.e., composition)
(Further complications help avoid infinite loops)

Backward Chaining with GMP



Backward Chaining (BC) with GMP

- BC is depth-first search
- BC versions
  - find any solution
  - find all solutions
- BC is basis for logic programming, e.g. Prolog:
  - a program is a set of logic sentences in HNF, which is called the database
  - it is executed by specifying a query to be proved

Prolog Examples

- grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
- brother_in_law_of(B, P) :- married(P, Spouse), brother_of(B, Spouse).
- brother_in_law_of(B, P) :- sister_of(Sister, P), husband_of(B, Sister).
- brother_in_law_of(B, P) :- married(P, Spouse), sister_of(Sister, Spouse), husband_of(B, Sister).

?- brother_in_law_of(John, Mary).

Completeness of General FOL

- FC and BC are complete for Horn KBs but are incomplete for general FOL KBs:

```
PhD(x)    ⇒ HighlyQualified(x)
¬PhD(x)   ⇒ EarlyEarnings(x)
HighlyQualified(x)  ⇒ Rich(x)
EarlyEarnings(x)    ⇒ Rich(x)
Query: Rich(Me)
```

- Can't prove query with FC or BC. Why?
- Does a complete algorithm for FOL exist?

Resolution Proofs

- Entailment in FOL is only semidecidable:
  - can prove α if **KB** $\models \alpha$
  - **cannot** always prove that **KB** doesn't $\models \alpha$ (halting)
- Resolution is a refutation technique:
  - to prove **KB** $\models \alpha$ show that **KB** $\wedge \neg\alpha$ is unsatisfiable
- Resolution uses KB and $\neg\alpha$ in CNF:
  - conjunction of clauses that are disjunction of literals
- Resolution *repeatedly combines two clauses to make a new one until the empty clause is derived*
  - the new clauses are called resolvents
  - empty clause: False, unsatisfiable, a contradiction

Resolution Inference Rule

- **Resolution Rule in PL** **equivalently:**

$$\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma} \qquad \frac{\neg\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

- **Resolution Rule in FOL (RR):**
  where l_i and m_i are literals for all i
  where $\text{UNIFY}(l_j, m_k) = \theta$, and m_k is the negation of l_j

$$\frac{l_1 \vee \ldots\ l_j \vee \ldots \vee l_m,\quad m_1 \vee \ldots\ m_k \vee \ldots \vee m_n}{\text{SUBST}(\theta,\ l_1 \vee \ldots\ l_{j-1} \vee l_{j+1} \ldots \vee l_m \vee m_1 \vee \ldots\ m_{k-1} \vee m_{k+1} \ldots \vee m_n)}$$

- **RR can equivalently be written as implications**

GMP Example of Resolution Rule

$$\frac{\texttt{Fulfilled(Me)},\ \neg\texttt{Fulfilled(x)} \vee \texttt{Happy(x)}}{\text{SUBST}(\theta,\ \texttt{Happy(x)})}$$

- l_j is **Fulfilled(Me)**
  m_k is $\neg$**Fulfilled(x)**
- $\text{UNIFY}(l_j, m_k)$ results in $\theta = \{x/\text{Me}\}$
  SUBST(θ, **Happy(x)**) results in **Happy(Me)**

  Inferred sentence: **Happy(Me)**

- GMP is special case of resolution

Resolution Refutation Example

To prove `KB` $\models$ `Rich(Me)`:

1. negate query: $\neg$`Rich(Me)`
2. convert query to CNF: $\neg$`Rich(Me)`
3. add query to CNF KB:
   (how do we convert sentences below into CNF?)
   PhD(x) $\Rightarrow$ **HighlyQualified(x)**
   $\neg$**PhD(x)** $\Rightarrow$ **EarlyEarnings(x)**
   HighlyQualified(x) $\Rightarrow$ **Rich(x)**
   EarlyEarnings(x) $\Rightarrow$ **Rich(x)**
4. infer contradiction, i.e., False

Simplified Resolution Refutation Algorithm

```
//returns true if KB |- q, false otherwise
//KB is a set of consistent, true FOL sentences
//q is the query to be proved
//requires KB and q are in CNF
boolean resolutionRefutation(CNFlist KB, CNFsentence q) {
  KB = unionOf(KB, logicalNegationOf(q));
  while (false not in KB) {
    pick two sentences s1 and s2 in KB with literals that unify
    if (none) return false;   //failure
    CNFsentence resolvent = resolutionRule(s1, s2);
    KB = unionOf(KB, resolvent);
  }
  return true;  //success
}
```

Conjunctive Normal Form (CNF)

- KB and query are conjunctions of CNF clauses
- CNF clause: a disjunction of literals

  e.g., **Hot(x) ∨ Warn(x) ∨ Cold(x)**
- Literal: an atom
  either positive (unnegated) or negative (negated)

  e.g., **¬Happy(Sally), Rich(x)**
- Any FOL KB can be converted into CNF

Simple Example of Converting FOL Sentence to CNF

- Replace implications:

  PhD(x) ⇒ HighlyQualified(x) **becomes**
  ¬PhD(x) ∨ HighlyQualified(x)
  ¬PhD(x) ⇒ EarlyEarnings(x) **becomes**
  PhD(x) ∨ EarlyEarnings(x)
  HighlyQualified(x) ⇒ Rich(x) **becomes**
  ¬HighlyQualified(x) ∨ Rich(x)
  EarlyEarnings(x) ⇒ Rich(x) **becomes**
  ¬EarlyEarnings(x) ∨ Rich(x)

- In-class Resolution Refutation example

Converting FOL Sentences Conjunctive Normal Form (CNF)

1. Replace ⇔ with equivalent (added):
   convert **P⇔Q** to **P⇒Q∧Q⇒P**

Example:

```
∀x,y {Above(x,y) ⇔ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)})}
```

becomes:

```
∀x,y {
   [ Above(x,y) ⇒ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ]
   ∧
   [ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ⇒ Above(x,y) ]
}
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

2. Replace ⇒ with equivalent:
   convert P ⇒ Q to ¬P ∨ Q

```
∀x,y {
    [ Above(x,y) ⇒ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ]
    ∧
    [ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ⇒ Above(x,y) ] }
```

becomes:

```
∀x,y {
    [ ¬Above(x,y) ∨ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ]
    ∧
    [ ¬(OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ∨ Above(x,y) ] }
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

3. Reduce scope of ¬ to single literals:

convert	¬¬P	to	P	(DNE)
convert	¬(P ∨ Q)	to	¬P ∧ ¬Q	(de Morgan's)
convert	¬(P ∧ Q)	to	¬P ∨ ¬Q	(de Morgan's)
convert	¬∀x P	to	∃x ¬P	
convert	¬∃x P	to	∀x ¬P	

```
∀x,y {
    [ ¬Above(x,y) ∨ (OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ] ∧
    [ ¬(OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ∨ Above(x,y) ] }
```

highlighted part becomes in stages:

```
(¬OnTop(x,y) ∧ ¬∃z{OnTop(x,z) ∧ Above(z,y)})      (de Morgan's)
∀z ¬{OnTop(x,z) ∧ Above(z,y)}                   (¬∃x P to ∀x ¬P)
{¬ OnTop(x,z) ∨ ¬ Above(z,y)}                    (de Morgan's)
(¬OnTop(x,y) ∧ ∀z{¬ OnTop(x,z) ∨ ¬ Above(z,y)})  (result)
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

4. Standardize variables apart:
   each quantifier must have a unique variable name
   avoids confusion in steps 5 and 6
   e.g. convert ∀x P ∨ ∃x Q to ∀x P ∨ ∃y Q

```
∀x,y {
    [ ¬Above(x,y)∨(OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ] ∧
    [ (¬OnTop(x,y) ∧∀z{¬OnTop(x,z) ∨ ¬Above(z,y)})∨Above(x,y) ] }
```

becomes:

```
∀x,y {
    [ ¬Above(x,y)∨(OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)}) ] ∧
    [ (¬OnTop(x,y) ∧∀w{¬OnTop(x,w) ∨ ¬Above(w,y)})∨Above(x,y) ] }
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

5. Eliminate existential quantifiers (Skolemize):
   - convert **∃x P(x)** to **P(C)** (EE)
     C must be a new constant (Skolem constant)
   - convert **∀x,y ∃z P(x,y,z)** to **∀x,y P(x,y,F(x,y))**
     F() must be a new function (Skolem function) with arguments that are all enclosing universally quantified variables

   e.g. Everyone has a name.
   ∀x Person(x)⇒ ∃y Name(y)∧ Has(x,y)
   wrong: ∀x Person(x)⇒ Name(K)∧ Has(x,K)
   Everyone has the same name K.
   Want everyone to have a name based on who they are.
   right: ∀x Person(x)⇒ Name(F(x))∧ Has(x,F(x))

Converting FOL Sentences Conjunctive Normal Form (CNF)

5. Eliminate existential quantifiers (Skolemize):
   - convert **∀x,y ∃z P(x,y,z)** to **∀x,y P(x,y,F(x,y))**
     F() must be a new function (Skolem function) with arguments that are all enclosing universally quantified variables

```
∀x,y {
    [ ¬Above(x,y)∨(OnTop(x,y) ∨∃z{OnTop(x,z) ∧ Above(z,y)})  ]
    ∧
    [  (¬OnTop(x,y) ∧∀w{¬OnTop(x,w) ∨ ¬Above(w,y)})∨Above(x,y)  ]  }
```

becomes:

```
∀x,y {
    [¬Above(x,y)∨(OnTop(x,y)∨(OnTop(x,F(x,y))∧Above(F(x,y),y)))]
    ∧
    [(¬OnTop(x,y) ∧∀w{¬OnTop(x,w) ∨ ¬Above(w,y)})∨Above(x,y)]  }
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

6. Drop quantifiers:
   all variables are only universally quantified after step 5
   e.g. convert ∀x P(x) ∨∀y Q(y) to P(x) ∨Q(y)
   all variables in KB will be assumed to be universally quantified

```
∀x,y {
    [¬Above(x,y)∨(OnTop(x,y)∨(OnTop(x,F(x,y))∧Above(F(x,y),y)))]
    ∧
    [(¬OnTop(x,y) ∧∀w{¬OnTop(x,w) ∨ ¬Above(w,y)})∨Above(x,y)]  }
```

becomes:

```
    [¬Above(x,y)∨(OnTop(x,y)∨(OnTop(x,F(x,y))∧Above(F(x,y),y)))]
    ∧
    [(¬OnTop(x,y) ∧ (¬OnTop(x,w) ∨ ¬Above(w,y)))∨ Above(x,y)]
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

7. Distribute ∧ over ∨ to get conjunction of disjunctions :
   convert (P ∧ Q)∨R to (P ∨ R)∧(Q ∨ R)

```
    [ ¬Above(x,y)∨(OnTop(x,y)∨(OnTop(x,F(x,y))∧Above(F(x,y),y)))  ]
    ∧
    [  (¬OnTop(x,y) ∧ (¬OnTop(x,w) ∨ ¬Above(w,y)))∨ Above(x,y)  ]
```

highlighted part becomes in steps:

```
given        A∨ (B∨ (C∧D))
converts to  A∨ ((B∨C) ∧ (B∨D))
converts to  (A∨ (B∨C)) ∧ (A∨ (B∨D))
```

highlighted part result:

```
    [    (¬Above(x,y) ∨ (OnTop(x,y) ∨ OnTop(x,F(x,y))))  ∧
         (¬Above(x,y) ∨ (OnTop(x,y) ∨ Above(F(x,y),y)))      ]
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

7. Distribute ∧ over ∨ to get conjunction of disjunctions :
   convert (P ∧ Q)∨R to (P ∨ R)∧(Q ∨ R)

```
    [ ¬Above(x,y)∨(OnTop(x,y)∨(OnTop(x,F(x,y))∧Above(F(x,y),y)))  ]
    ∧
    [(¬OnTop(x,y) ∧ (¬OnTop(x,w) ∨ ¬Above(w,y))) ∨ Above(x,y)]
```

highlighted part becomes in steps:

```
given        (A∧B) ∨C
converts to  (A∨C) ∧ (B∨C)
```

highlighted part result:

```
    [    (¬OnTop(x,y)                  ∨ Above(x,y)  ) ∧
         ((¬OnTop(x,w) ∨ ¬Above(w,y))  ∨ Above(x,y)  )     ]
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

8. Flatten nested conjunctions and disjunctions:
   convert (P ∧ Q) ∧ R to (P ∧ Q ∧ R)
   convert (P ∨ Q) ∨ R to (P ∨ Q ∨ R)

```
[   (¬Above(x,y) ∨ (OnTop(x,y) ∨ OnTop(x,F(x,y))))  ∧
    (¬Above(x,y) ∨ (OnTop(x,y) ∨ Above(F(x,y),y)))    ] ∧
[    (¬OnTop(x,y) ∨                 Above(x,y))     ∧
    ((¬OnTop(x,w) ∨ ¬Above(w,y)) ∨  Above(x,y))       ]
```

becomes:

```
(¬Above(x,y) ∨ OnTop(x,y) ∨ OnTop(x,F(x,y))) ∧
(¬Above(x,y) ∨ OnTop(x,y) ∨ Above(F(x,y),y))
∧
(¬OnTop(x,y) ∨ Above(x,y)) ∧
(¬OnTop(x,w) ∨ ¬Above(w,y) ∨ Above(x,y))
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

9. Separate each conjunct (added)
   spllit at ∧'s so each conjunct is now a CNF clause

```
(¬Above(x,y) ∨ OnTop(x,y) ∨ OnTop(x,F(x,y))) ∧
(¬Above(x,y) ∨ OnTop(x,y) ∨ Above(F(x,y),y))
∧
(¬OnTop(x,y) ∨ Above(x,y)) ∧
(¬OnTop(x,w) ∨ ¬Above(w,y) ∨ Above(x,y))
```

becomes:

```
¬Above(x,y)∨OnTop(x,y)∨OnTop(x,F(x,y))
¬Above(x,y)∨OnTop(x,y)∨Above(F(x,y),y)
¬OnTop(x,y)∨ Above(x,y)
¬OnTop(x,w)∨¬Above(w,y)∨Above(x,y)
```

Converting FOL Sentences Conjunctive Normal Form (CNF)

10. Standardize variables apart in each clause (added)
   – each clause in KB must contain unique variable names
   – now during unification the standardize apart step need only be done on deduced clauses (i.e. resolvents)

```
¬Above(x,y)∨OnTop(x,y)∨OnTop(x,F(x,y))
¬Above(x,y)∨OnTop(x,y)∨Above(F(x,y),y)
¬OnTop(x,y)∨ Above(x,y)
¬OnTop(x,w)∨¬Above(w,y)∨Above(x,y)
```

becomes:

```
¬Above(a,b)∨OnTop(a,b)∨OnTop(a,F(a,b))
¬Above(c,d)∨OnTop(c,d)∨Above(F(c,d),d)
¬OnTop(e,f)∨Above(e,f)
¬OnTop(g,h)∨¬Above(h,i)∨Above(g,i)
```

Dealing with Equality

- Limitation of unification:
  – can't unify different terms that refer to same object
  – uses syntactic matching
  – doesn't do semantic test of sameness
- Equational Unification axiomizes properties of =:
  – **reflexivity:** $\forall x \quad x = x$
  – **symmetricity:** $\forall x,y \quad x = y \Rightarrow y = x$
  – **transitivity:** $\forall x,y,z \quad x = y \wedge y = z \Rightarrow x = z$
  – **for all** P_i $\quad \forall x,y \quad x = y \Rightarrow P_i(x) \Leftrightarrow P_i(z)$
  – **etc…**

  Terms are unifiable if they're provably equal under some substitution

Dealing with Equality

- **Another approach is to use a special inference rule: Paramodulation:**

$$\frac{l_1 \vee \ldots \vee l_k \vee x = y,\ m_1 \vee \ldots \vee m_n[z]}{SUBST(\theta,\ l_1 \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_n[y])}$$

  - where l_i and m_i are literals for all i, and $m_n[z]$ is a literal containing term z
  - for any terms x, y, and z, where $UNIFY(x, z) = \theta$
  - Put simpler term on the right of equality to do simplification, since term on left is always replaced with term on right of =

- **Demodulation is a special case where there are no l_i literals**

Paramodulation Example

$$\frac{\mathtt{L(v)} \vee \mathtt{F(H,v)} = \mathtt{F(J,v)},\quad \mathtt{M(J)} \vee \mathtt{N(F(H,K))}}{SUBST(\theta,\ \mathtt{L(v)} \vee \mathtt{M(J)} \vee \mathtt{N(F(J,v))})}$$

Predicates: **L,M,N** Function: **F**
Variable: **v** Constants: **H,J,K**

- $m_n[z]$ is `N(F(H,K))` and z is `F(H,K)`
  $x = y$ is where x is `F(H,v)` and y is `F(J,v)`
- **UNIFY**(x, z) result in $\theta = \{\mathtt{v/K}\}$
- SUBST(θ, …) **results in inferred sentence:**
  `L(K) ∨ M(J) ∨ N(F(J,K))`

Resolution Strategies

- Resolution refutation proofs can be thought of as search:
  - reversed construction of search tree (leaves to root)
  - leaves are KB clauses and ¬query
  - resolvent is new node with arcs to parent clauses
  - root is a clause containing False

Resolution Strategies

- A search is complete if it guarantees the empty clause can be derived whenever `KB` ⊨ `q`
- Goal is to design a complete search that efficiently finds a contradiction (i.e., empty clause, False)
- Rather than just choosing any two clauses to be resolved, instead reduce the choices to be from some subset of clauses. The different resolution strategies specify what that subset is.

Resolution Strategies

- Breadth-First
  - level 0 clauses: KB clauses and ¬query
  - level k clauses: resolvents computed from 2 clauses:
    - one of which must be from level k-1
    - other from any earlier level
  - compute all possible level 1 clauses, then all possible level 2 clauses, etc.
  - **complete but very inefficient**

Resolution Strategies

- Unit Preference
  - prefer to do resolutions where 1 sentence is a single literal, a unit clause
  - goal is to produce the empty clause, focus search by producing resolvents that are shorter
  - **complete but too slow for medium sized problems**
- Unit Resolution
  - requires at least 1 to be a unit clause
  - resembles FC
  - **complete for FOL KB in HNF**

Resolution Strategies

- Set-of-Support (SoS)
  - identify some subset of sentences, called SoS
  - P and Q can be resolved if one if from SoS
  - resolvent is added to the SoS
  - common approach:
    - ¬query is the initial SoS, resolvents are added
    - assumes KB is true (i.e., consistent, jointly satisfiable)
  - **complete if KB-SoS is jointly satisfiable**

Resolution Strategies

- Input Resolution
  - P and Q can be resolved if at least one is from the set of original clauses, i.e. KB and ¬query
  - proof trees have a single "spine" (see Fig. 9.11)
  - MP is a form of input resolution since each step a rule (input) is used to generate a new fact
  - **complete for FOL KB in HNF**

Resolution Strategies

- Linear Resolution
  - a slight generalization of input resolution
  - P and Q can be resolved if:
    - at least 1 is from the set of original clauses
    - or P must be an ancestor of Q in the proof tree
  - **complete**

Reference: Converting FOL Sentences to CNF

1. Replace $\Leftrightarrow$ with equivalent (added):
   - convert $P \Leftrightarrow Q$ to $P \Rightarrow Q \land Q \Rightarrow P$
2. Replace $\Rightarrow$ with equivalent: convert $P \Rightarrow Q$ to $\neg P \lor Q$
3. Reduce scope of $\neg$ to single literals:
   - convert $\neg\neg P$ to P (DNE)
   - convert $\neg(P \lor Q)$ to $\neg P \land \neg Q$ (de Morgan's)
   - convert $\neg(P \land Q)$ to $\neg P \lor \neg Q$ (de Morgan's)
   - convert $\neg\forall x\, P$ to $\exists x\, \neg P$
   - convert $\neg\exists x\, P$ to $\forall x\, \neg P$
4. Standardize variables apart:
   - each quantifier must have a unique variable name
   - avoids confusion in steps 5 and 6
   - e.g. convert $\forall x\, P \lor \exists x\, Q$ to $\forall x\, P \lor \exists y\, Q$

Reference: Converting FOL Sentences to CNF

5. Eliminate existential quantifiers (Skolemize):
   - convert $\exists x\, P(x)$ to $P(C)$ (EE)
     C must be a new constant (Skolem constant)
   - convert $\forall x,y\, \exists z\, P(x,y,z)$ to $\forall x,y\, P(x,y,F(x,y))$
     $F()$ must be a new function (Skolem function) with arguments that are all enclosing universally quantified variables
6. Drop quantifiers:
   - all variables are only universally quantified after step 5
   - e.g. convert $\forall x\, P(x) \lor \forall y\, Q(y)$ to $P(x) \lor Q(y)$
   - all variables in KB will be assumed to be universally quantified
7. Distribute $\land$ over $\lor$ to get conjunction of disjunctions :
   - convert $(P \land Q) \lor R$ to $(P \lor R) \land (Q \lor R)$

Reference: Converting FOL Sentences to CNF

8. Flatten nested conjunctions and disjunctions:
   - convert $(P \land Q) \land R$ to $(P \land Q \land R)$
   - convert $(P \lor Q) \lor R$ to $(P \lor Q \lor R)$
9. Separate each conjunct (added)
   - spllit at $\land$'s so each conjunct is now a CNF clause
10. Standardize variables apart in each clause (added)
    - each clause in KB must contain unique variable names
    - now during unification the standardize apart step need only be done on deduced clauses (i.e. resolvents)