First Order Logic

Artificial Intelligence

Slides are mostly adapted from AIMA and MIT Open Courseware,

and Milos Hauskrecht (U. Pittsburgh)

Pros and cons of propositional logic

- ☺ Propositional logic is declarative
- ☺ Propositional logic allows partial/disjunctive/negated information
  - (unlike most data structures and databases)
- ☺ Propositional logic is compositional:
  - meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- ☺ Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- ☹ Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "pits cause breezes in adjacent squares“
    - except by writing one sentence for each square

Logic Puzzles

- Gilderoy, Minerva, Pomona and Horace each belong to a different one of the four houses: Gryffindor, Hufflepuff, Ravenclaw, and Slytherin House.
- Gilderoy belongs to Gryffindor or Ravenclaw.
- Pomona does not belong in Slytherin.
- Minerva belongs to Gryffindor.

Slide credit : HarvardX CS50AICS50's Introduction to Artificial Intelligence with Python, David J. Malan and Brian Yu

Logic Puzzles

Propositional Symbols

GilderoyGryffindor	*MinervaGryffindor*
GilderoyHufflepuff	*MinervaHufflepuff*
GilderoyRavenclaw	*MinervaRavenclaw*
GilderoySlytherin	*MinervaSlytherin*
PomonaGryffindor	*HoraceGryffindor*
PomonaHufflepuff	*HoraceHufflepuff*
PomonaRavenclaw	*HoraceRavenclaw*
PomonaSlytherin	*HoraceSlytherin*

Logic Puzzles

$$(PomonaSlytherin \rightarrow \neg PomonaHufflepuff)$$

$$(MinervaRavenclaw \rightarrow \neg GilderoyRavenclaw)$$

$$(GilderoyGryffindor \vee GilderoyRavenclaw)$$

Slide credit : HarvardX CS50AICS50's Introduction to Artificial Intelligence with Python, David J. Malan and Brian Yu

First Order Logic

- Propositional logic only deals with "facts", statements that may or may not be true of the world, e.g., "It is raining". But, one cannot have variables that stand for books or tables.
- In first-order logic, variables refer to things in the world and, furthermore, you can quantify over them: talk about all of them or some of them without having to name them explicitly.

First-order logic

- First-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus,
    - (relations in which there is only one value for a given input)

FOL Motivation

Statements that cannot be made in propositional logic but can be made in FOL

- When you paint a block with green paint, it becomes green.
  - In propositional logic, one would need a statement about every single block, one cannot make the general statement about all blocks.
- When you sterilize a jar, all the bacteria are dead.
  - In FOL, we can talk about all the bacteria without naming them explicitly.
- A person is allowed access to this Web site if they have been formally authorized or they are known to someone who has access.

Syntax of FOL: Basic elements

- Constants : KingJohn, 2, ...
- Predicates : Brother, >,...
- Functions : Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality $=$
- Quantifiers $\forall, \exists$

Atomic sentences

Term = *function* ($term_1$,...,$term_n$)
or *constant*
or *variable*

Atomic sentence = *predicate* ($term_1$,...,$term_n$)
or $term_1 = term_2$

- E.g., *Brother(KingJohn,RichardTheLionheart)*
- *> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S,\ S_1 \wedge S_2,\ S_1 \vee S_2,\ S_1 \Rightarrow S_2,\ S_1 \Leftrightarrow S_2,$$

E.g.

$Sibling(KingJohn,Richard) \Rightarrow Sibling(Richard,KingJohn)$

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Quantifiers

Universal quantification, $\forall$ (pronounced as "For all")

$\forall$ x Cat(x) $\Rightarrow$ Mammal(x)

All cats are mammals

Existential quantification, $\exists$ (pronounced as "There exists")

$\exists$ x Sister (x, Spot) $\wedge$ Cat(x)

Spot has a sister who is a cat

- $\forall$x *P* is true in a model *m*
  iff *P* is true with *x* being each possible object in the model
- $\exists x$ *P* is true in a model *m*
  iff *P* is true with *x* being some possible object in the model

First-Order Logic

Constant Symbol	Predicate Symbol
Minerva	*Person*
Pomona	*House*
Horace	*BelongsTo*
Gilderoy	
Gryffindor	
Hufflepuff	
Ravenclaw	
Slytherin	

Slide credit : HarvardX CS50AICS50's Introduction to Artificial Intelligence with Python, David J. Malan and Brian Yu

First-Order Logic

$Person(Minerva)$ Minerva is a person.

$House(Gryffindor)$ Gryffindor is a house.

$\neg House(Minerva)$ Minerva is not a house.

$BelongsTo(Minerva, Gryffindor)$

Minerva belongs to Gryffindor.

Slide credit : HarvardX CS50AICS50's Introduction to Artificial Intelligence with Python, David J. Malan and Brian Yu

Universal Quantification

$$\forall x.\ \mathit{BelongsTo}(x, \mathit{Gryffindor}) \rightarrow \neg \mathit{BelongsTo}(x, \mathit{Hufflepuff})$$

For all objects x, if x belongs to Gryffindor, then x does not belong to Hufflepuff.

Anyone in Gryffindor is not in Hufflepuff.

Slide credit : HarvardX CS50AICS50's Introduction to Artificial Intelligence with Python, David J. Malan and Brian Yu

Existential Quantification

$$\exists x.\ House(x) \land BelongsTo(Minerva, x)$$

There exists an object x such that
x is a house and Minerva belongs to x.

Minerva belongs to a house.

Slide credit : HarvardX CS50AICS50's Introduction to Artificial Intelligence with Python,
David J. Malan and Brian Yu

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- An atomic sentence *predicate(term*$_1$*,...,term*$_n$*)* is true
  iff the objects referred to by *term*$_1$*,...,term*$_n$
  are in the relation referred to by *predicate*
- Interpretation specifies referents for
  constant symbols → objects
  predicate symbols → relations
  function symbols → functional relations

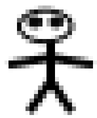
Interpretation

An interpretation *I* is defined by a **mapping** to the **domain of discourse D or relations on D**

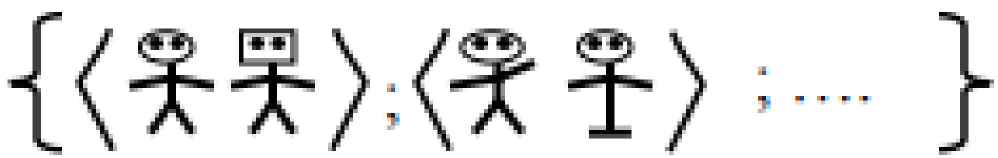
- **domain of discourse:** a set of objects in the world we represent and refer to;

An interpretation *I* maps:

- Constant symbols to objects in D

  I(*John*) = 

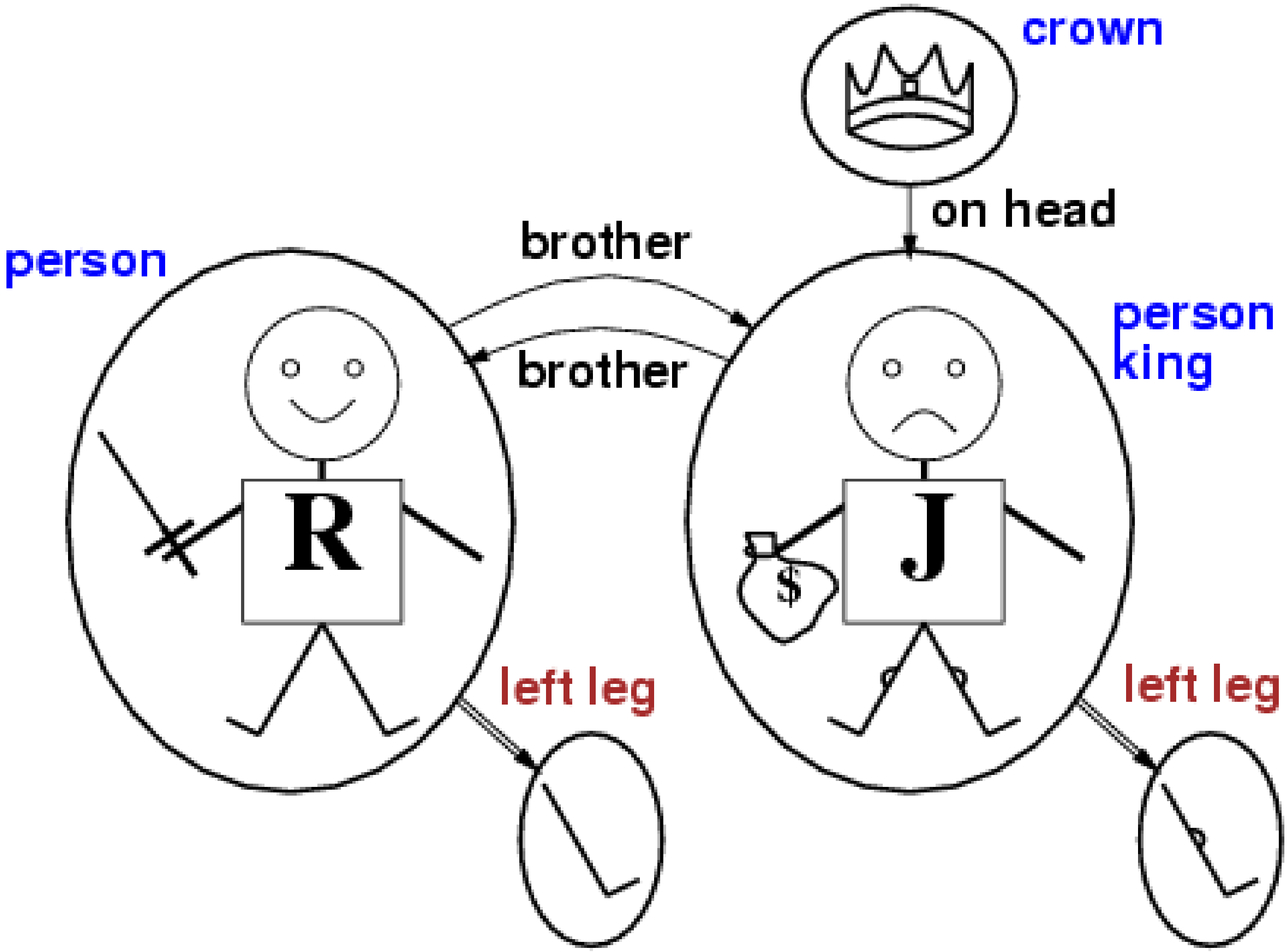
- Predicate symbols to relations, properties on D

  I(*brother*) = 

- Function symbols to functional relations on D

  I(*father-of*) = 

Models for FOL: Example

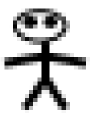
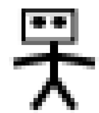


Semantics

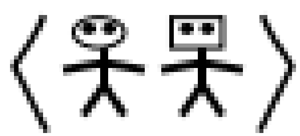
Meaning (evaluation) function:

$$V : \text{sentence} \times \text{interpretation} \rightarrow \{True, False\}$$

A **predicate** *predicate(term-1, term-2, term-3, term-n)* is true for the interpretation I, iff the objects referred to by *term-1, term-2, term-3, term-n* are in the relation referred to by *predicate*

I(*John*) =  I(*Paul*) = 

I(*brother*) = 

brother(John, Paul) =  in I(*brother*)

V(*brother(John, Paul), I*) = ***True***

Semantics

- **Equality** $V(term\text{-}1 = term\text{-}2, I) = \textbf{\textit{True}}$

  Iff $I(term\text{-}1) = I(term\text{-}2)$

- **Boolean expressions: standard**

  E.g. $V(sentence\text{-}1 \vee sentence\text{-}2, I) = \textbf{\textit{True}}$

  Iff $V(sentence\text{-}1, I) = \textbf{\textit{True}}$ or $V(sentence\text{-}2, I) = \textbf{\textit{True}}$

- **Quantifications**

  $V(\forall x\ \phi, I) = \textbf{\textit{True}}$

  Iff for all $d \in D$ $V(\phi, \overbrace{I[x/d]}^{\text{substitution of } x \text{ with } d}) = \textbf{\textit{True}}$

  $V(\exists x\ \phi, I) = \textbf{\textit{True}}$

  Iff there is a $d \in D$, s.t. $V(\phi, I[x/d]) = \textbf{\textit{True}}$

Universal quantification

$\forall$ *<variables>* *<sentence>*

All Kings are persons:

$\forall x\ King(x) \Rightarrow Person(x)$

$\forall x\ P$ is true in a model m

iff P is true with x being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

Richard the Lionheart is a king $\Rightarrow$ Richard the Lionheart is a person

$\land$ King John is a king $\Rightarrow$ King John is a person

$\land$ Richard's left leg is a king $\Rightarrow$ Richard's left leg is a person

$\land$ John's left leg is a king $\Rightarrow$ John's left leg is a person

$\land$ The crown is a king $\Rightarrow$ The crown is a person

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$

  $\forall x\ King(x) \Rightarrow Person(x)$

- Common mistake: using $\wedge$ as the main connective with $\forall$:

  $\forall x\ King(x) \wedge Person(x)$

  means "Everyone is a king and everyone is a person"

  Richard the Lionheart is a king $\wedge$ Richard the Lionheart is a person

  $\wedge$ King John is a king $\wedge$ King John is a person

  $\wedge$ Richard's left leg is a king $\wedge$ Richard's left leg is a person

  $\wedge$ John's left leg is a king $\wedge$ John's left leg is a person

  $\wedge$ The crown is a king $\wedge$ The crown is a person

Existential quantification

$\exists$<*variables*> <*sentence*>

$\exists x$ Crown(x) $\wedge$ OnHead(x,John)

$\exists x\ P$ is true in a model m

iff P is true with x being some possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

  The crown is a crown $\wedge$ the crown is on John's head

$\vee$ Richard the Lionheart is a crown $\wedge$ Richard the Lionheart is on John's head

$\vee$ King John is a crown $\wedge$ King John is on John's head

$\vee$ ...

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- $\exists x$ Crown(x) $\wedge$ OnHead(x,John)

- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \text{ Crown}(x) \Rightarrow \text{OnHead}(x,\text{John})$$

is true even if there is anything which is not a crown

The crown is a crown $\Rightarrow$ the crown is on John's head
$\vee$ Richard the Lionheart is a crown $\Rightarrow$ Richard the Lionheart is on John's head
$\vee$ King John is a crown $\Rightarrow$ King John is on John's head

Properties of quantifiers

$\forall x\ \forall y$ is the same as $\forall y\ \forall x$, and can be written as $\forall x,y$
$\exists x\ \exists y$ is the same as $\exists y\ \exists x$, and can be written as $\exists x,y$

$\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
$\forall y\ \exists x$ Loves(x,y)

– "Everyone in the world is loved by at least one person"

$\exists x\ \forall y$ Loves(x,y)

– "There is a person who loves everyone in the world"

$\forall x\ \exists y$ P(x,y) : every object in the universe has a particular property, given by P
$\exists x\ \forall y$ P(x,y) : there is some object in the world that has a particular property

Rule: the variable belongs to the innermost quantifier that mentions it
$\forall x$ [Cat(x) V ($\exists x$ Brother(Richard,x))]
$\forall x$ [Cat(x) V ($\exists z$ Brother(Richard,z))]

Properties of quantifiers

- Quantifier duality: each can be expressed using the other

$\exists x$ Likes(x,Broccoli) $= \neg\forall x \neg$Likes(x,Broccoli)
$\forall x$ Likes(x,IceCream) $= \neg\exists x \neg$Likes(x,IceCream)

- De Morgan's rules for quantifiers:

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

Equality

- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

- E.g., definition of *Sibling* in terms of *Parent*:

  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \wedge\ \exists m,f\ \neg\ (m = f) \wedge$
  $Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge\ Parent(f,y)]$

Writing FOL

- Cats are mammals [Cat^1, $Mammal^1$]
  - $\forall$ x. Cat(x) $\rightarrow$ Mammal(x)
- Jane is a tall surveyor [$Tall^1$, $Surveyor^1$, Jane]
  - Tall(Jane) $\wedge$ Surveyor(Jane)
- A nephew is a sibling's son [$Nephew^2$, $Sibling^2$, Son^2]
  - $\forall$xy. [Nephew(x,y) $\leftrightarrow$ $\exists$z . [Sibling(y,z) $\wedge$ Son(x,z)]]
- A maternal grandmother is a mother's mother [functions: mgm, mother-of]
  - $\forall$xy. x=mgm(y) $\leftrightarrow$
    $\exists$z. x=mother-of(z) $\wedge$ z=mother-of(y)

Writing FOL

- Nobody loves Jane
  - $\forall x. \neg Loves(x,Jane)$
  - $\neg\exists x. Loves(x,Jane)$
- Everybody has a father
  - $\forall x. \exists y. Father(y,x)$
- Everybody has a father and a mother
  - $\forall x. \exists yz. Father(y,x) \wedge Mother(z,x)$
- Whoever has a father, has a mother
  - $\forall x.[[\exists y. Father(y,x)] \rightarrow [\exists y. Mother(y,x)]]$

Using FOL

The kinship domain:

- Brothers are siblings

  $\forall$x,y *Brother(x,y)* $\Leftrightarrow$ *Sibling(x,y)*

- One's mother is one's female parent

  $\forall$m,c *Mother(c)* = m $\Leftrightarrow$ *(Female(m)* $\wedge$ *Parent(m,c))*

- “Sibling” is symmetric

  $\forall$x,y *Sibling(x,y)* $\Leftrightarrow$ *Sibling(y,x)*

- One's husband is one's male spouse

  $\forall$w,h *Husband(h,w)* $\Leftrightarrow$ *(Male(m)* $\wedge$ *Spouse(h,w))*

- Sibling is another child of one’s parents

  $\forall$*x,y Sibling(x,y)* $\Leftrightarrow$ [$\neg$(x = y) $\wedge$ $\exists$m,f $\neg$ (m = f) $\wedge$ *Parent(m,x)* $\wedge$ *Parent(f,x)* $\wedge$ *Parent(m,y)* $\wedge$ *Parent(f,y)]*

Inference in First Order Logic

Artificial Intelligence

Slides are mostly adapted from AIMA and MIT Open Courseware,

Milos Hauskrecht (U. Pittsburgh)

and Max Welling (UC Irvine)

Logical Inference

Logical inference problem:

- Given a knowledge base KB (a set of sentences) and a sentence α , does the KB semantically entail α ?

$$KB \models \alpha \quad ?$$

In other words: In all interpretations in which sentences in the KB are true, is also α true?

Inference in Propositional Logic

Computational procedures that answer:

$$KB \models \alpha \ ?$$

Three approaches:

- **Truth-table approach**
- **Inference rules**
- **Conversion to the inverse SAT problem**
  - **Resolution-refutation**

Inference in FOL : Truth Table Approach

- Is the Truth-table approach a viable approach for the FOL?
  ?
- **NO!**
- Why?
- It would require us to enumerate and list all possible interpretations **I**
- I = (assignments of symbols to objects, predicates to relations and functions to relational mappings)
- Simply there are too many interpretations

Inference Rules

- **Inference rules from the propositional logic:**
  - Modus ponens

$$\frac{A \Rightarrow B, \quad A}{B}$$

  - Resolution

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

  - and others: And-introduction, And-elimination, Or-introduction, Negation elimination
- **Additional inference rules are needed for sentences with quantifiers and variables**
  - **Rules must involve variable substitutions**

Sentences with variables

First-order logic sentences can include variables.

- **Variable** is:
  - **Bound** – if it is in the scope of some quantifier

$$\forall x\ P(x)$$

  - **Free** – if it is not bound.

$$\exists x\ P(y) \wedge Q(x) \quad y \text{ is free}$$

Examples:

$$\forall x\ \exists y\ Likes\ (x, y)$$

- Bound

$$\forall x\ (Likes\ (x, y) \wedge \exists y\ Likes\ (y, Raymond\ ))$$

- Free

Sentences with variables

First-order logic sentences can include variables.

- **Sentence** (formula) is:
  - **Closed** – if it has no free variables

    $$\forall y \exists x \; P(y) \Rightarrow Q(x)$$

  - **Open** – if it is not closed

    $\exists x \; P(y) \wedge Q(x)$ $\quad$ y is free

  - **Ground** – if it does not have any variables

    $$Likes(John, Jane)$$

Variable Substitutions

- Variables in the sentences can be substituted with terms.
  (terms = constants, variables, functions)
- **Substitution:**
  - Is represented by a mapping from variables to terms

$$\theta = \{x_1 / t_1, x_2 / t_2, \ldots\} \qquad \mathrm{SUBST}(\theta, \alpha)$$

  - Application of the substitution to sentences

$$SUBST(\{x / Sam, y / Pam\}, Likes(x, y)) = Likes(Sam, Pam)$$

$$SUBST(\{x / z, y / fatherof(John)\}, Likes(x, y)) =$$
$$Likes(z, fatherof(John))$$

Universal elimination

- 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:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$, $\{x/John\}$

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$, $\{x/Richard\}$

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

$\{x/Father(John)\}$

Example:

$$\forall x\ Likes(x, IceCream)$$

$$\Big\downarrow \quad \{x/Ben\}$$

$$Likes(Ben, IceCream)$$

$$\frac{\forall x\ \phi(x)}{\phi(a)}$$

Existential elimination

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

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

- 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

$$\exists x\ Kill(x, Victim) \longrightarrow Kill(Murderer, Victim) \qquad \frac{\exists x\ \phi(x)}{\phi(a)}$$

Special constant called a **Skolem** constant

$$\exists x\ Crown(x) \wedge OnHead(x, John) \longrightarrow$$

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

Inference rules for quantifiers

- **Universal instantiation (introduction)**

$$\frac{\phi}{\forall x\ \phi} \qquad x \text{ – is not free in } \phi$$

  – Introduces a universal variable which does not affect ϕ or its assumptions

$$Sister(Amy, Jane) \qquad \forall x\ Sister(Amy, Jane)$$

- **Existential instantiation (introduction)**

$$\frac{\phi(a)}{\exists x\, \phi(x)} \qquad \begin{array}{l} a \text{ – is a ground term in } \phi \\ x \text{ – is not free in } \phi \end{array} \qquad \frac{\alpha}{\exists v\ \mathrm{Subst}(\{g/v\}, \alpha)}$$

  – Substitutes a ground term in the sentence with a variable and an existential statement

$$Likes(Ben, IceCream) \qquad \exists x\ Likes(x, IceCream)$$

Example Proof

- 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 American.

- Prove that Col. West is a criminal

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$\forall x,y,z\ American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Nation(z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono … has some missiles, i.e.,

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

… all of its missiles were sold to it by Colonel West

$\forall x\ Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Missiles are weapons:

$\forall\ x\ Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile“:

$\forall x\ Enemy(x,America) \Rightarrow Hostile(x)$

West, who is American …

$American(West)$

The country Nono

$Nation(Nono)$

Nono, an enemy of America …

$Enemy(Nono,America)$, $Nation(America)$

Example knowledge base contd.

1. $\forall$ x,y,z *American(x)* $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Nation(z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*
2. $\exists$x Owns(Nono,x) $\wedge$ Missile(x):
3. $\forall$x *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
4. $\forall$ x *Missile(x)* $\Rightarrow$ *Weapon(x)*
5. $\forall$x *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*
6. *American(West)*
7. *Nation(Nono)*
8. *Enemy(Nono,America)*
9. *Nation(America)*
10. *Owns(Nono,M_1) and Missile(M_1)* *Existential elimination 2*
11. *Owns(Nono,M_1)* *And elimination 10*
12. *Missile(M_1)* *And elimination 10*
13. *Missile(M1)* $\Rightarrow$ *Weapon(M1)* *Universal elimination 4*
14. *Weapon(M1)* *Modus Ponens, 12, 13*
15. *Missile(M1)* $\wedge$ *Owns(Nono,M1)* $\Rightarrow$ *Sells(West,M1,Nono)* *Universal Elimination 3*
16. *Sells(West,M1,Nono)* *Modus Ponens 10,15*
17. *American(West)* $\wedge$ *Weapon(M1)* $\wedge$ *Sells(West,M1,Nono)* $\wedge$ *Nation(Nono)* $\wedge$ *Hostile(Nono)* $\Rightarrow$ *Criminal(Nono)* *Universal elimination, three times 1*
18. *Enemy(Nono,America)* $\Rightarrow$ *Hostile(Nono)* *Universal Elimination 5*
19. *Hostile(Nono)* *Modus Ponens 8, 18*
20. *American(West)* $\wedge$ *Weapon(M1)* $\wedge$ *Sells(West,M1,Nono)* $\wedge$ *Nation(Nono)* $\wedge$ *Hostile(Nono)* *And Introduction 6,7,14,16,19*
21. *Criminal(West)* *Modus Ponens 17, 20*

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

King(John)

Greedy(John)

Brother(Richard,John)

- Instantiating the universal sentence in all possible ways (there are only two ground terms: John and Richard) , we have:

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

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

  King(John)

  Greedy(John)

  Brother(Richard,John)

- The new KB is propositionalized: proposition symbols are

  King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction contd.

- Every FOL KB can be propositionalized so as to preserve entailment
  - A ground sentence is entailed by new KB iff entailed by original KB
- Idea for doing inference in FOL:
  - propositionalize KB and query
  - apply inference
  - return result
- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*))), etc

Reduction contd.

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do

create a propositional KB by instantiating with depth-n terms

see if α is entailed by this KB

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936)

Entailment for FOL is semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from:

  $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

  King(John)

  $\forall y$ Greedy(y)

  Brother(Richard,John)

- it seems obvious that *Evil*(*John*) is entailed, but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.
- Lets see if we can do inference directly with FOL sentences

Generalized Modus Ponens (GMP)

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

where we can unify p_i‘ and p_i for all i
i.e. $p_i'\theta = p_i\,\theta$ for all *i*

Example: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

p_1' is *King*(*John*) p_1 is *King*(*x*)

p_2' is *Greedy*(*y*) p_2 is *Greedy*(*x*)

θ is {x/John, y/John} q is *Evil*(*x*)

Subst(θ,q) is *Evil*(*John*)

Example: $\forall x\ Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

p_1' is *Missile*(*M1*) p_1 is *Missile*(*x*)

p_2' is *Owns*(*y, M1*) p_2 is Owns(*Nono*,*x*)

θ is {x/M1, y/Nono} q is *Sells*(West, Nono, *x*)

Subst(θ,q) is *Sells*(*West*, *Nono*, *M1*)

- Implicit assumption that all variables universally quantified
  GMP used with KB of definite clauses (exactly one positive literal)

Soundness and completeness of GMP

GMP is sound

Only derives sentences that are logically entailed

- Need to show that $p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$
  provided that $p_i'\theta = p_i\theta$ for all *I*
- Lemma: For any sentence *p*, we have $p \models p\theta$ by UI

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1', \backslash; \ldots, \backslash;p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

GMP is complete for a KB consisting of definite clauses

- Complete: derives all sentences that entailed
- OR…answers every query whose answers are entailed by such a KB
-
- Definite clause: disjunction of literals of which exactly 1 is positive,

  e.g., King(x) AND Greedy(x) -> Evil(x)

  NOT(King(x)) OR NOT(Greedy(x)) OR Evil(x)

Generalized Modus Ponens (GMP)

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

Substitution that satisfies the generalized inference rule can be build via *unification process*

Advantage of the generalized rules: **they are focused**

- only substitutions that allow the inferences to proceed are tried

Use substitutions that let us make inferences !!!!

Convert each sentence into cannonical form prior to inference: Either an atomic sentence or an implication with a conjunction of atomic sentences on the left hand side and a single atom on the right (Horn clauses)

Unification

- **Problem in inference:** Universal elimination gives us many opportunities for substituting variables with ground terms

$$\frac{\forall x\ \phi(x)}{\phi(a)} \qquad a \text{ - is a constant symbol}$$

- **Solution: make only substitutions that may help**
  - Use substitutions of "similar" sentences in KB

- **Unification** – takes two similar sentences and computes the substitution that **makes them look the same**, if it exists

$$UNIFY\ (p,q) = \sigma \ \text{ s.t. } SUBST(\sigma, p) = SUBST\ (\sigma, q)$$

Unification

- **Unification:**

$$UNIFY\,(p,q) = \sigma \text{ s.t. } SUBST(\sigma, p) = SUBST\,(\sigma, q)$$

- **Examples:**

$$UNIFY\,(Knows\,(John, x), Knows\,(John, Jane)) = \{x \,/\, Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = \{x \,/\, Ann, y \,/\, John\}$$

$$UNIFY\,(Knows\,(John, x), Knows\,(y, MotherOf\,(y))) \\ = \{x \,/\, MotherOf\,(John), y \,/\, John\}$$

$$UNIFY\,(Knows\,(John, x), Knows\,(x, Elizabeth)) = fail$$

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, Elizabeth)	{x/ Elizabeth,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x, Elizabeth)	{fail}

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

Unification

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

  θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

- The first unifier is more general than the second.
- Most general unifier is the substitution that makes the least commitment about the bindings of the variables
- There is a single most general unifier (MGU) that is unique up to renaming of variables.

  MGU = { y/John, x/z }

The unification algorithm

```
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 $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Example knowledge base revisited

1. $\forall$ x,y,z *American(x)* $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Nation(z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*
2. $\exists$x Owns(Nono,x) $\wedge$ Missile(x):
3. $\forall$x *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
4. $\forall$ x *Missile(x)* $\Rightarrow$ *Weapon(x)*
5. $\forall$x *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*
6. *American(West)*
7. *Nation(Nono)*
8. *Enemy(Nono,America)*
9. *Nation(America)*

Convert the sentences into Horn form

1. *American(x)* $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Nation(z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*
2. *Owns*(*Nono*,M_1)
3. *Missile*(M_1)
4. *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
5. *Missile(x)* $\Rightarrow$ *Weapon(x)*
6. *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*
7. *American(West)*
8. *Nation(Nono)*
9. *Enemy(Nono,America)*
10. *Nation(America)*
11. *Proof*
12. *Weapon(M1)*
13. *Hostile(Nono)*
14. *Sells(West,M1,Nono)*
15. ***Criminal(West)***

Inference appoaches in FOL

- Forward-chaining
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses

- Backward-chaining
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference

- Note that these methods are generalizations of their propositional equivalents

Forward chaining algorithm

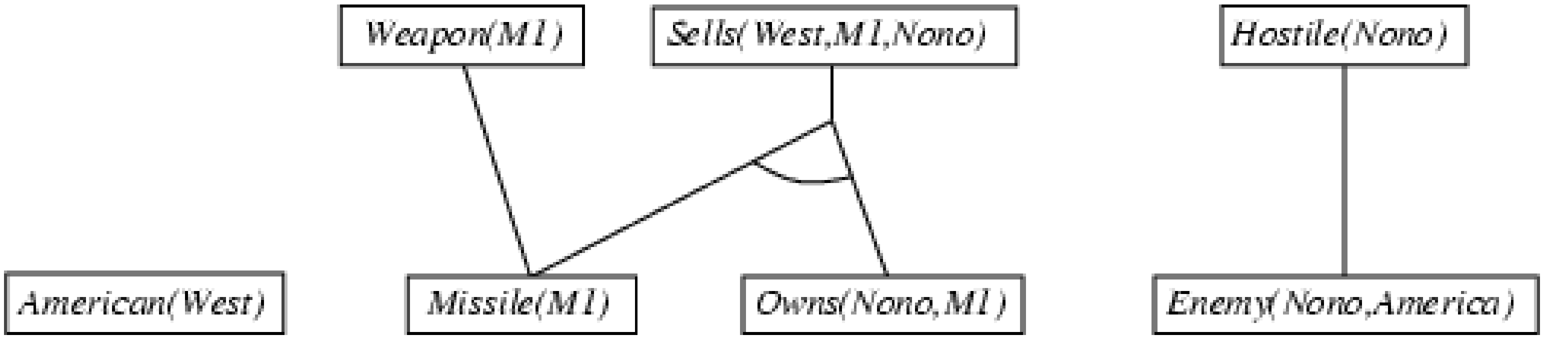
function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

  repeat until *new* is empty
    new $\leftarrow \{\,\}$
    for each sentence r **in** KB **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
      for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
          for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' is not a renaming of a sentence already in KB or *new* **then do**
          add q' to *new*
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not *fail* **then return** ϕ
    add *new* to KB
  return *false*

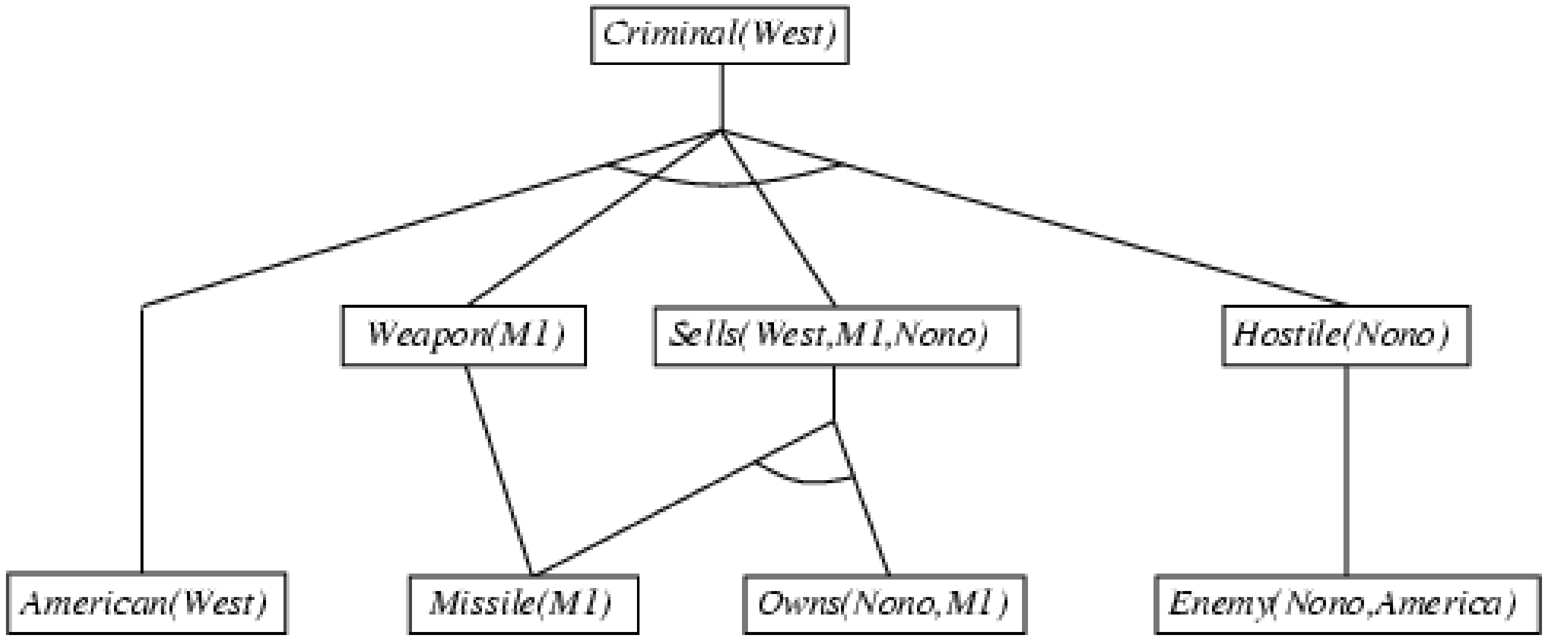
Forward chaining proof

American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

Forward chaining proof



Forward chaining proof



Properties of forward chaining

- Sound and complete for first-order definite clauses

- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations

- May not terminate in general if α is not entailed

- This is unavoidable: entailment with definite clauses is semidecidable

Efficiency of forward chaining

Incremental forward chaining: no need to match a rule on iteration k if a premise wasn't added on iteration $k\text{-}1$

$\Rightarrow$ match each rule whose premise contains a newly added positive literal

Matching itself can be expensive:

Database indexing allows O(1) retrieval of known facts

- e.g., query $Missile(x)$ retrieves $Missile(M_1)$

Forward chaining is widely used in deductive databases

Backward chaining algorithm

function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: ans, a set of substitutions, initially empty

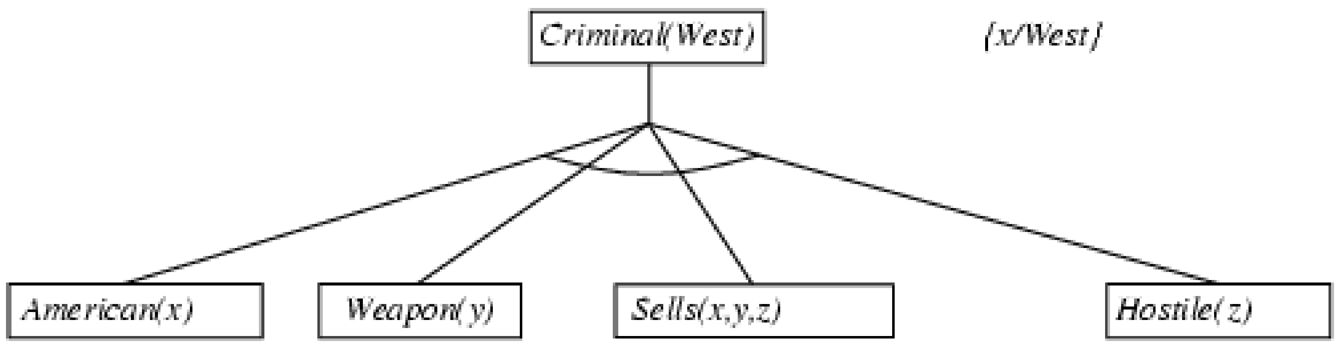
  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \textsc{Subst}(\theta, \textsc{First}(goals))$
  for each r **in** KB where $\textsc{Standardize-Apart}(r) = (\, p_1 \land \ldots \land p_n \Rightarrow q)$
    and $\theta' \leftarrow \textsc{Unify}(q, q')$ succeeds
   $ans \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \ldots, p_n | \textsc{Rest}(goals)], \textsc{Compose}(\theta, \theta')) \cup ans$
  return ans

SUBST(COMPOSE(θ_1, θ_2), p) =
 SUBST(θ_2, SUBST(θ_1, p))

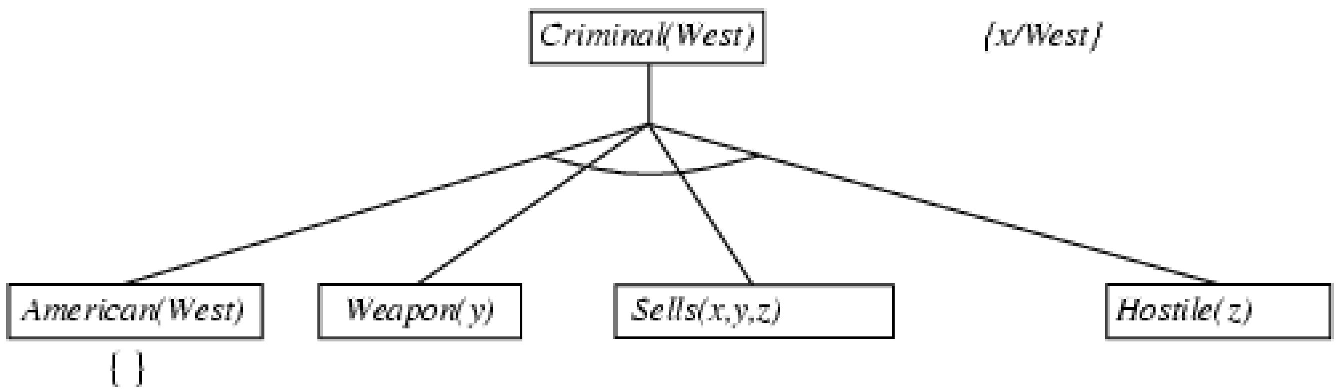
Backward chaining example

$\boxed{Criminal(West)}$

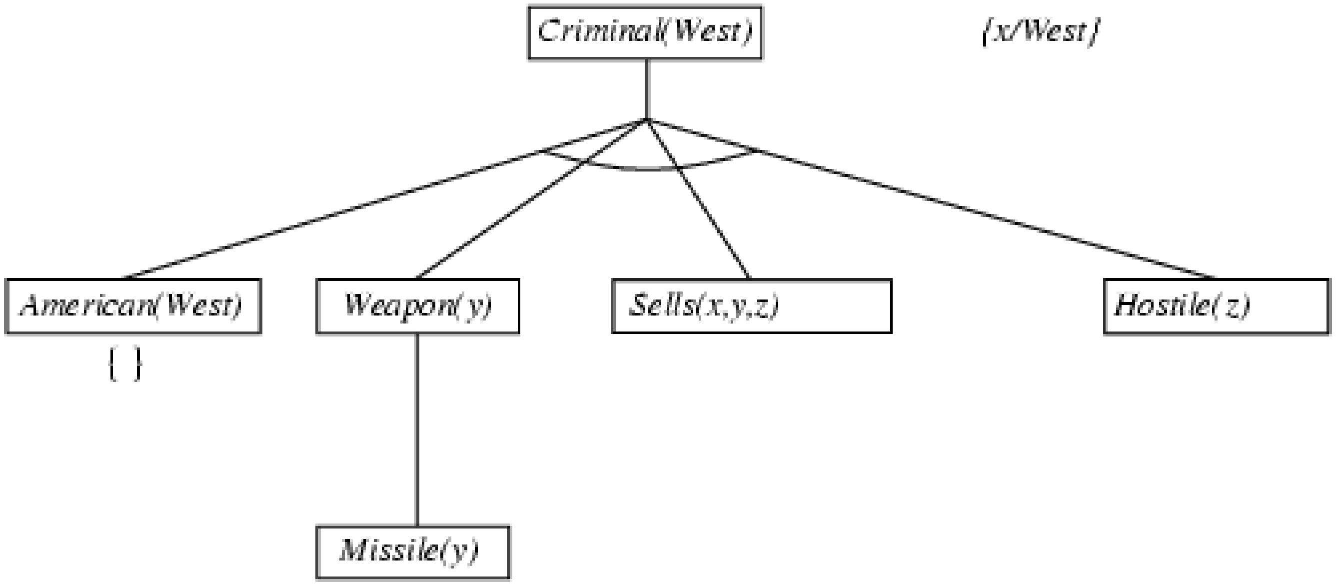
Backward chaining example



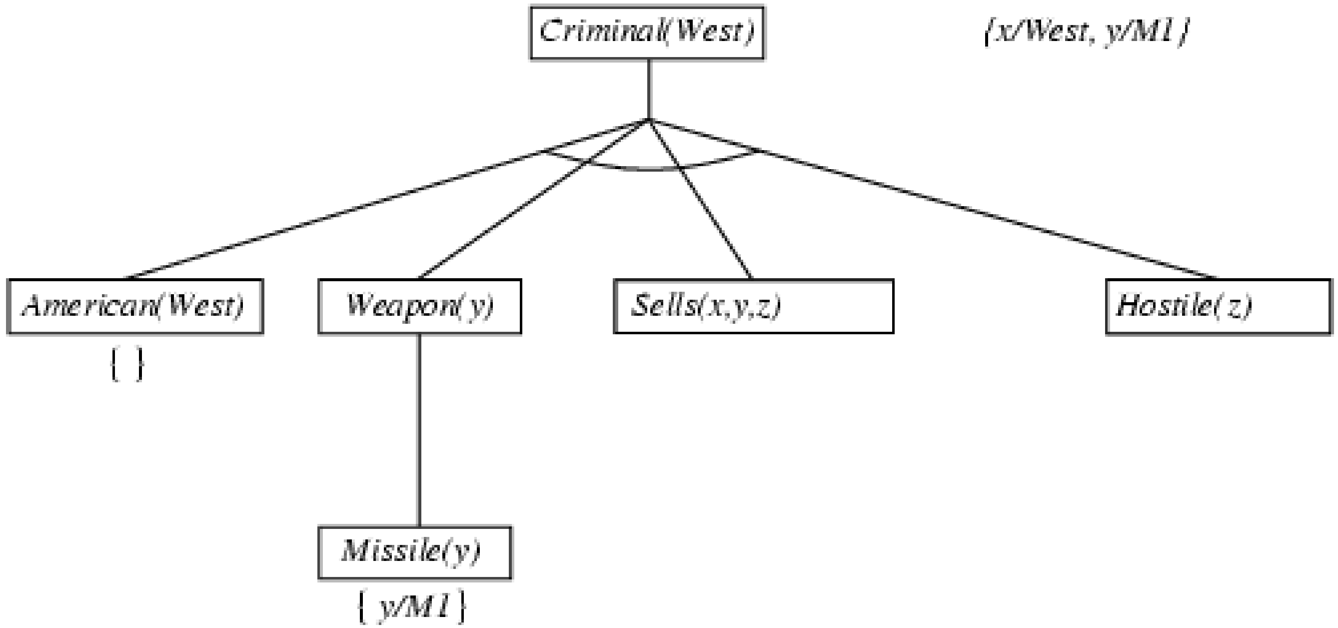
Backward chaining example



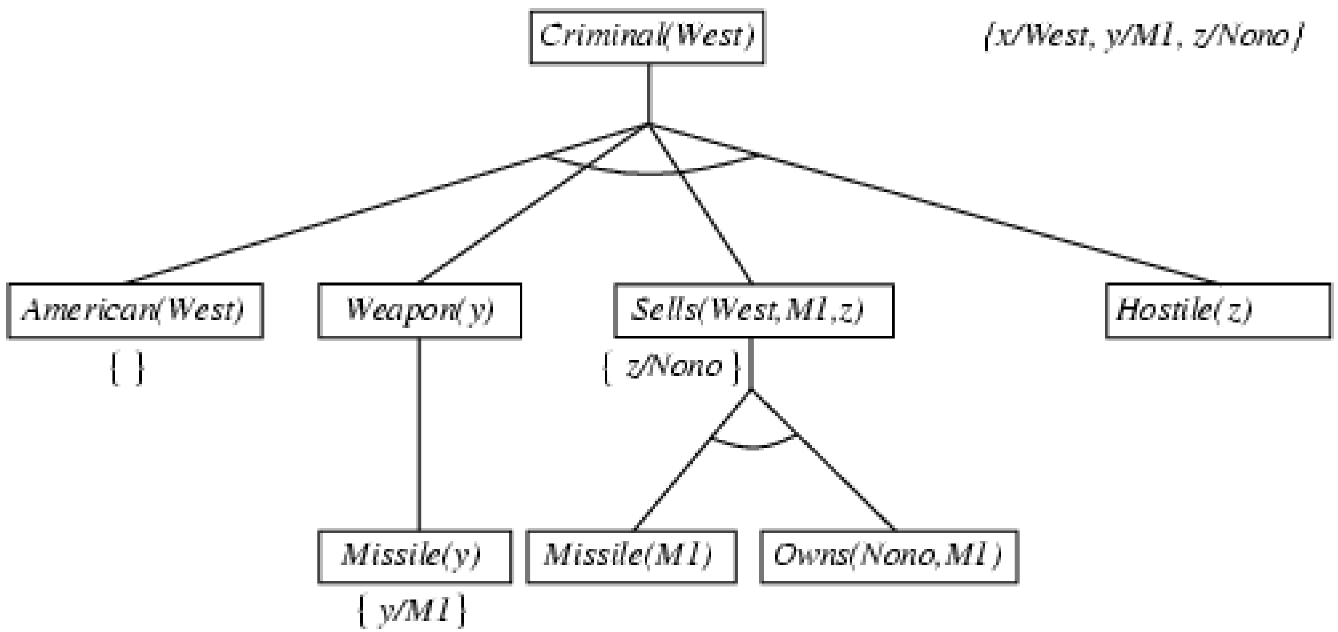
Backward chaining example



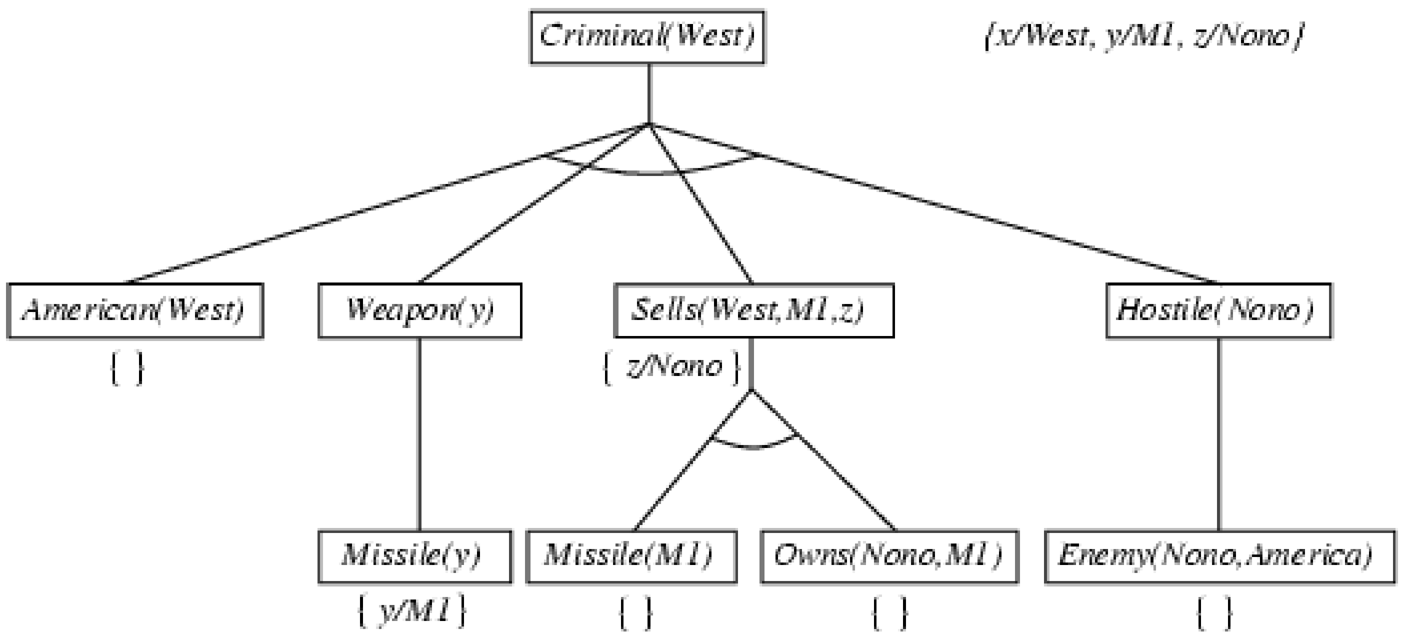
Backward chaining example



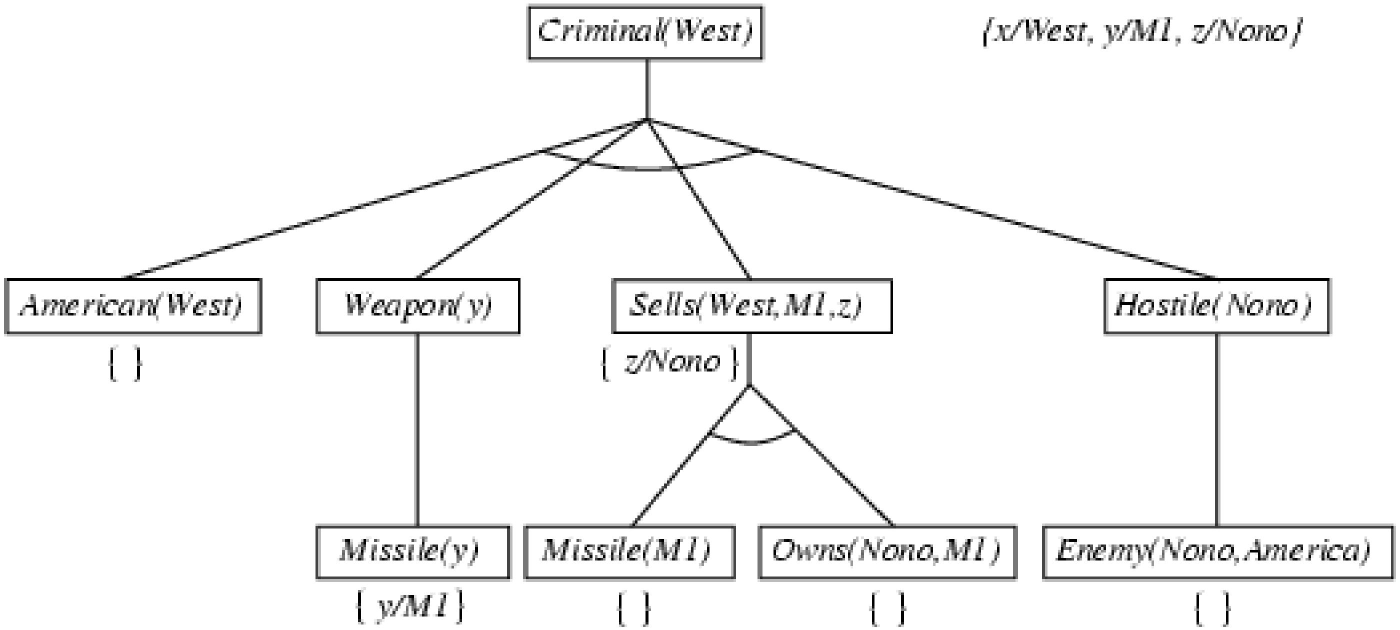
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - $\Rightarrow$ fix using caching of previous results (extra space)
- Widely used for logic programming

Logic programming: Prolog

- Algorithm = Logic + Control
- Basis: backward chaining with Horn clauses + bells & whistles
- Program = set of clauses = `head :- literal`$_1$`, … literal`$_n$`.`

  `criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).`
- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
- Built-in predicates that have side effects (e.g., input and output
- predicates, assert/retract predicates)
- Closed-world assumption ("negation as failure")
  - e.g., given `alive(X) :- not dead(X).`
  - `alive(joe)` succeeds if `dead(joe)` fails

Resolution in First Order Logic

Artificial Intelligence

Slides are mostly adapted from AIMA and MIT Open Courseware

and Milos Hauskrecht (U. Pittsburgh)

Resolution Inference Rule

- **Recall:** Resolution inference rule is sound and complete (refutation-complete) for the **propositional logic** and CNF

$$\frac{A \vee B, \quad \neg A \vee C}{B \vee C}$$

- **Generalized resolution rule is sound and refutation complete** for the first-order logic and CNF w/o equalities (if unsatisfiable the resolution will find the contradiction)

$$\sigma = UNIFY\ (\phi_i, \neg \psi_j) \neq fail$$

$$\frac{\phi_1 \vee \phi_2 \ldots \vee \phi_k, \quad \psi_1 \vee \psi_2 \vee \ldots \psi_n}{SUBST(\sigma, \phi_1 \vee \ldots \vee \phi_{i-1} \vee \phi_{i+1} \ldots \vee \phi_k \vee \psi_1 \vee \ldots \vee \psi_{j-1} \vee \psi_{j+1} \ldots \psi_n)}$$

Example: $$\frac{P(x) \vee Q(x), \quad \neg Q(John) \vee S(y)}{P(John) \vee S(y)}$$

First Order Resolution

$$\frac{\forall x.\ P(x) \rightarrow Q(x) \qquad P(A)}{Q(A)}$$

Syllogism:

All men are mortal
Socrates is a man
Socrates is mortal

uppercase letters: constants

lowercase letters: variables

$$\frac{\forall x.\ \neg P(x) \vee Q(x) \qquad P(A)}{Q(A)}$$

Equivalent by definition of implication

Two new things:

- converting FOL to clausal form
- resolution with variable substitution

$$\frac{\neg P(A) \vee Q(A) \qquad P(A)}{Q(A)}$$

Substitute A for x, still true

then

Propositional resolution

Clausal Form

- like CNF in outer structure
- no quantifiers

$$\forall x.\ \exists y.\ P(x) \rightarrow R(x,y)$$

$$\Downarrow$$

$$\neg P(x) \vee R(x, F(x))$$

Converting to Clausal Form

1. Eliminate arrows

$$\alpha \leftrightarrow \beta \Rightarrow (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$$\alpha \rightarrow \beta \Rightarrow \neg\alpha \vee \beta$$

2. Drive in negation

$$\neg(\alpha \vee \beta) \Rightarrow \neg\alpha \wedge \neg\beta$$

$$\neg(\alpha \wedge \beta) \Rightarrow \neg\alpha \vee \neg\beta$$

$$\neg\neg\alpha \Rightarrow \alpha$$

$$\neg\forall x.\ \alpha \Rightarrow \exists x.\ \neg\alpha$$

$$\neg\exists x.\ \alpha \Rightarrow \forall x.\ \neg\alpha$$

3. Rename variables apart

$$\forall x.\ \exists y.\ (\neg P(x) \vee \exists x.\ Q(x,y)) \Rightarrow$$

$$\forall x_1.\ \exists y_2.\ (\neg P(x_1) \vee \exists x_3.\ Q(x_3,y_2))$$

Also move all quantifiers left

$$(\forall x\ P(x)) \vee (\exists y\ Q(y)) \rightarrow \forall x\ \exists y\ P(x) \vee Q(y)$$

Converting to Clausal Form - Skolemization

Skolemization (removal of existential quantifiers through elimination)

If no universal quantifier occurs before the **existential quantifier**, replace the **variable with a new constant symbol**

also called Skolem constant

$$\exists y\ P(A) \vee Q(y) \rightarrow P(A) \vee Q(B)$$

If a universal quantifier precedes the existential quantifier replace the variable with a function of the “universal” variable

$$\forall x\, \exists y\ P(x) \vee Q(y) \rightarrow \forall x\ \ P(x) \vee Q(F(x))$$

$F(x)$ - **a special function**

- **called Skolem function**

Converting to Clausal Form - Skolemization

4. Skolemize
   - substitute new name for each existential var

$$\exists x.\, P(x) \Rightarrow P(\text{Fred})$$

$$\exists x, y.\, R(x, y) \Rightarrow R(\text{Thing1}, \text{Thing2})$$

$$\exists x.\, P(x) \wedge Q(x) \Rightarrow P(\text{Fleep}) \wedge Q(\text{Fleep})$$

$$\exists x.\, P(x) \wedge \exists x.\, Q(x) \Rightarrow P(\text{Frog}) \wedge Q(\text{Grog})$$

$$\exists y.\, \forall x.\, \text{Loves}(x, y) \Rightarrow \forall x.\, \text{Loves}(x, \text{Englebert})$$

   - substitute new function of all universal vars in outer scopes

$$\forall x.\, \exists y.\, \text{Loves}(x, y) \Rightarrow \forall x.\, \text{Loves}(x, \text{Beloved}(x))$$

$$\forall x.\, \exists y.\, \forall z.\, \exists w.\, P(x, y, z) \wedge R(y, z, w) \Rightarrow P(x, F(x), z) \wedge R(F(x), z, G(x, z))$$

Converting to Clausal Form

5. Drop universal quantifiers

$$\forall x.\ \text{Loves}(x, \text{Beloved}(x)) \Rightarrow \text{Loves}(x, \text{Beloved}(x))$$

6. Distribute or over and; return clauses

$$\text{P}(z) \vee (\text{Q}(z,w) \wedge \text{R}(w,z)) \Rightarrow \{\{\text{P}(z), \text{Q}(z,w)\}, \{\text{P}(z), \text{R}(w,z)\}\}$$

7. Rename the variables in each clause

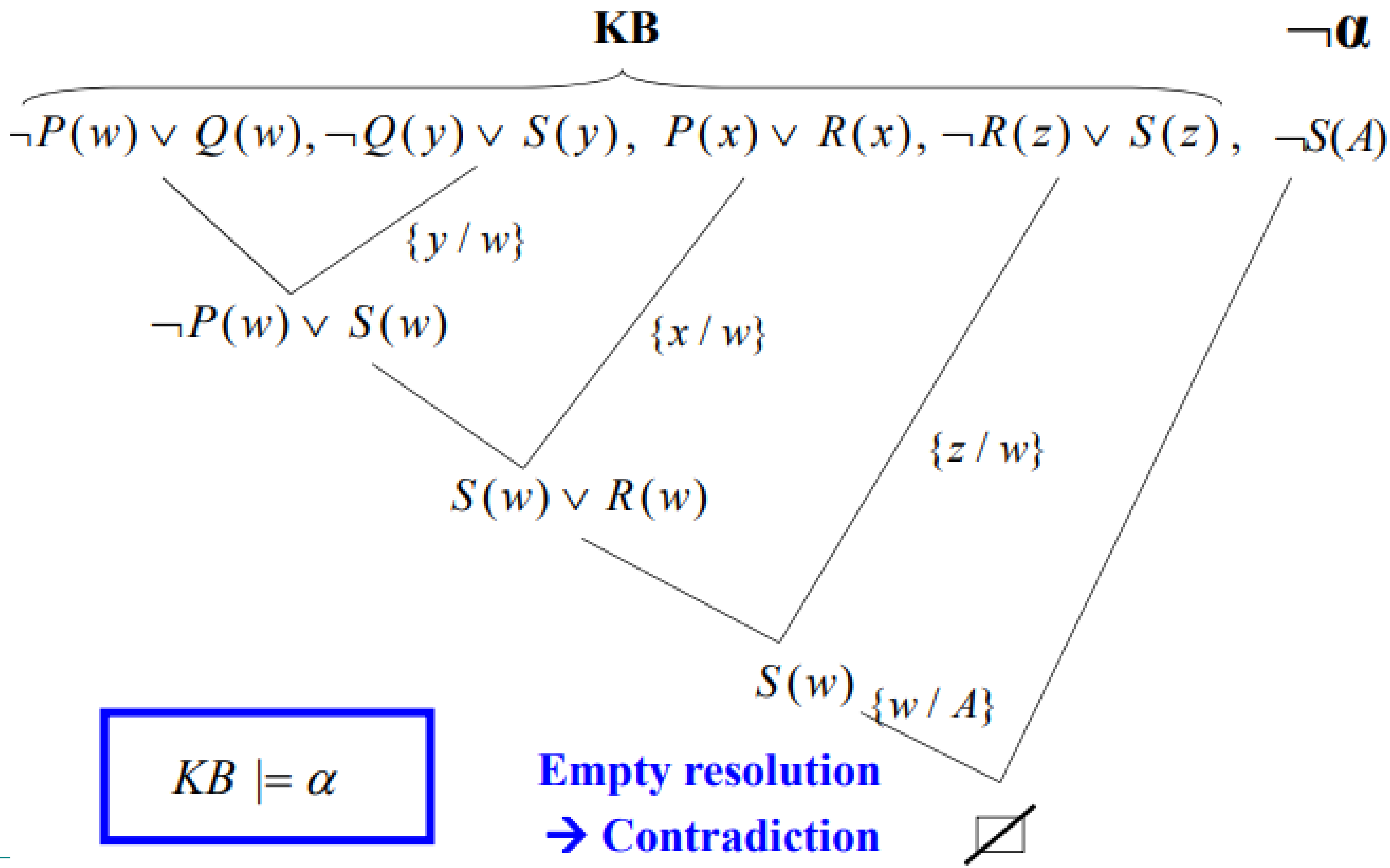
$$\{\{\text{P}(z), \text{Q}(z,w)\},\ \{\text{P}(z), \text{R}(w,z)\}\} \Rightarrow \{\{\text{P}(z_1), \text{Q}(z_1,w_1)\}, \{\text{P}(z_2), \text{R}(w_2,z_2)\}\}$$

Inference with resolution rule

- **Proof by refutation:**
  - Prove that $KB, \neg \alpha$ is **unsatisfiable**
  - resolution is **refutation-complete**

- **Main procedure (steps):**

1. Convert $KB, \neg \alpha$ to CNF with ground terms and universal variables only
2. Apply repeatedly the resolution rule while keeping track and consistency of substitutions
3. Stop when empty set (contradiction) is derived or no more new resolvents (conclusions) follow


KB
¬α
¬P(w) ∨ Q(w), ¬Q(y) ∨ S(y), P(x) ∨ R(x), ¬R(z) ∨ S(z), ¬S(A)
{y / w}
¬P(w) ∨ S(w)
{x / w}
S(w) ∨ R(w)
{z / w}
S(w)
{w / A}
KB |= α
Empty resolution
→ Contradiction

Dealing with Equality

- Resolution works for the first-order logic without equalities
- To incorporate equalities we need an additional inference rule
- **Demodulation rule**

$\sigma = UNIFY\ (z_i, t_1) \neq fail$ where z_i occurs in ϕ_i

$$\frac{\phi_1 \vee \phi_2 \ldots \vee \phi_k, \quad t_1 = t_2}{SUB(SUBST(\sigma, t_1), SUBST(\sigma, t_2), \phi_1 \vee \phi_2 \ldots \vee \phi_k)}$$

- **Example:** $$\frac{P(f(a)), f(x) = x}{P(a)}$$
- **Paramodulation rule: more powerful**
- **Resolution+paramodulation give a refutation-complete proof theory for FOL**

Example

a. John owns a dog
$\exists x.\ D(x) \wedge O(J,x)$
$D(Fido) \wedge O(J, Fido)$

b. Anyone who owns a dog is a lover-of-animals
$\forall x.\ (\exists y.\ D(y) \wedge O(x,y)) \rightarrow L(x)$
$\forall x.\ (\neg\exists y.\ (D(y) \wedge O(x,y)) \vee L(x)$
$\forall x.\ \forall y.\ \neg(D(y) \wedge O(x,y)) \vee L(x)$
$\forall x.\ \forall y.\ \neg D(y) \vee \neg O(x,y) \vee L(x)$
$\neg D(y) \vee \neg O(x,y) \vee L(x)$

c. Lovers-of-animals do not kill animals
$\forall x.\ L(x) \rightarrow (\forall y.\ A(y) \rightarrow \neg K(x,y))$
$\forall x.\ \neg L(x) \vee (\forall y.\ A(y) \rightarrow \neg K(x,y))$
$\forall x.\ \neg L(x) \vee (\forall y.\ \neg A(y) \vee \neg K(x,y))$
$\neg L(x) \vee \neg A(y) \vee \neg K(x,y)$

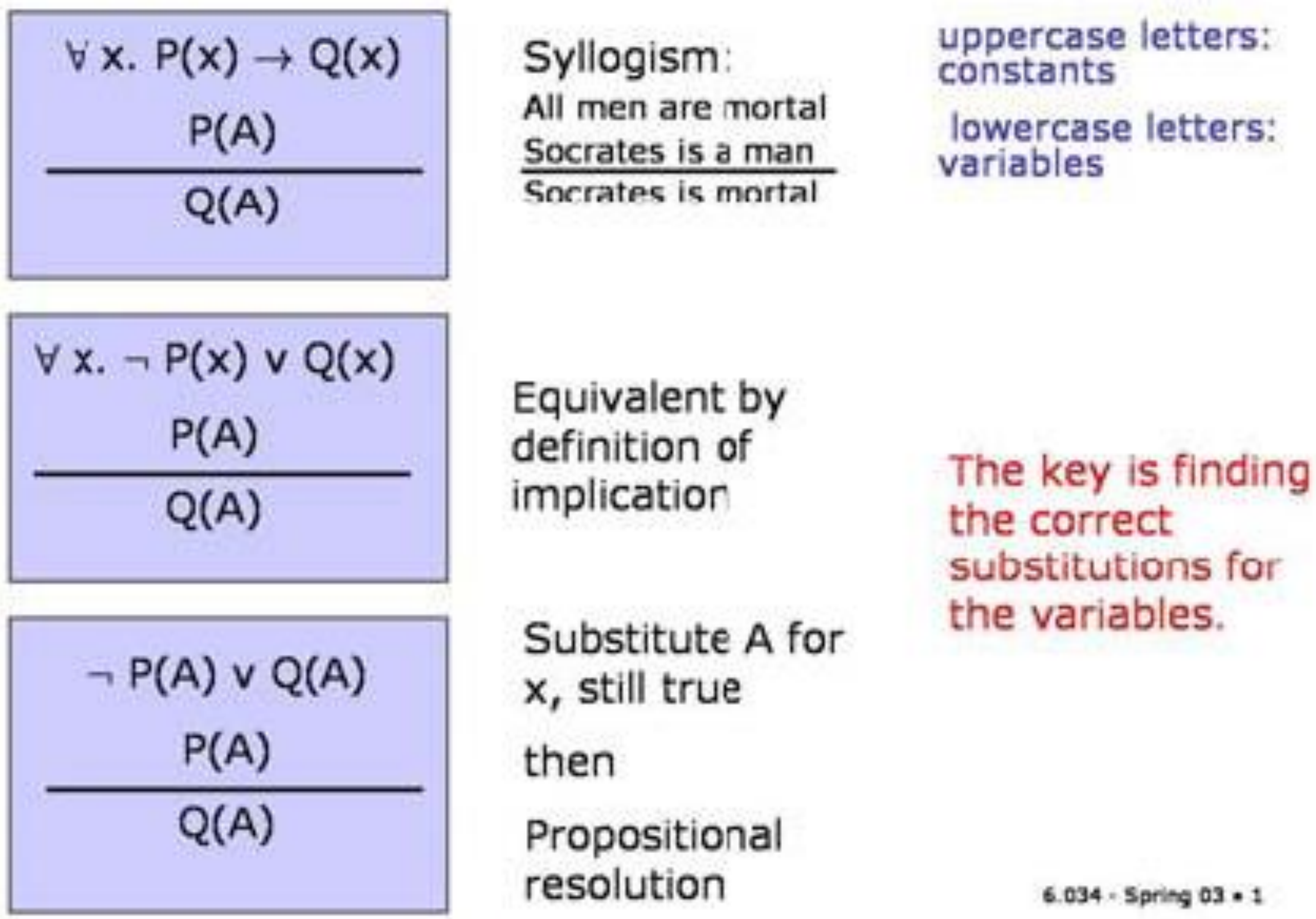
More examples

d. Either Jack killed Tuna or curiosity killed Tuna
K(J,T) v K(C,T)

e. Tuna is a cat
C(T)

f. All cats are animals
¬ C(x) v A(x)

First Order Resolution



Substitutions

$P(x, f(y), B)$: an atomic sentence

Substitution instances	Substitution $\{v_1/t_1, \ldots, v_n/t_n\}$	Comment
$P(z, f(w), B)$	$\{x/z, y/w\}$	Alphabetic variant
$P(x, f(A), B)$	$\{y/A\}$	
$P(g(z), f(A), B)$	$\{x/g(z), y/A\}$	
$P(C, f(A), B)$	$\{x/C, y/A\}$	Ground instance

Applying a substitution:

$P(x, f(y), B)\ \{y/A\} = P(x,f(A),B)$

$P(x, f(y), B)\ \{y/A, x/y\} = P(A, f(A), B)$

Unification

- Expressions ω_1 and ω_2 are unifiable iff there exists a substitution s such that ω_1 s = ω_2 s
- Let ω_1 = x and ω_2 = y, the following are unifiers

s	ω_1 s	ω_2 s
{y/x}	x	x
{x/y}	y	y
{x/f(f(A)), y/f(f(A))}	f(f(A))	f(f(A))
{x/A, y/A}	A	A

Most General Unifier

g is a most general unifier of ω_1 and ω_2 iff for all unifiers s, there exists s′ such that ω_1 s = (ω_1 g) s′ and ω_2 s = (ω_2 g) s′

ω_1	ω_2	MGU
P(x)	P(A)	{x/A}
P(f(x), y, g(x))	P(f(x), x, g(x))	{y/x} or {x/y}
P(f(x), y, g(y))	P(f(x), z, g(x))	{y/x, z/x}
P(x, B, B)	P(A, y, z)	{x/A, y/B, z/B}
P(g(f(v)), g(u))	P(x, x)	{x/g(f(v)), u/f(v)}
P(x, f(x))	P(x, x)	No MGU!

Unification Algorithm

```
unify(Expr x, Expr y, Subst s){
  if s = fail, return fail
  else if x = y, return s
  else if x is a variable, return unify-var(x, y, s)
  else if y is a variable, return unify-var(y, x, s)
  else if x is a predicate or function application,
       if y has the same operator,
             return unify(args(x), args(y), s)
       else  return fail
  else                          ; x and y have to be lists
       return unify(rest(x), rest(y),
                    unify(first(x), first(y), s))
}
```

Unify-var subroutine

Substitute in for var and x as long as possible, then add new binding

```
unify-var(Variable var, Expr x, Subst s){
  if var is bound to val in s,
        return unify(val, x, s)
  else if x is bound to val in s,
        return unify-var(var, val, s)
  else if var occurs anywhere in (x s), return fail
  else return add({var/x}, s)
}
```

Examples

ω_1	ω_2	MGU
A(B, C)	A(x, y)	{x/B, y/C}
A(x, f(D,x))	A(E, f(D,y))	{x/E, y/E}
A(x, y)	A(f(C,y), z)	{x/f(C,y),y/z}
P(A, x, f(g(y)))	P(y, f(z), f(z))	{y/A,x/f(z),z/g(y)}
P(x, g(f(A)), f(x))	P(f(y), z, y)	none
P(x, f(y))	P(z, g(w))	none

Resolution with Variables

$$\frac{\begin{array}{c}\alpha \vee \varphi \\ \neg\varphi \vee \beta\end{array}}{(\alpha \vee \beta)\theta} \quad \mathrm{MGU}(\varphi, \psi) = \theta$$

$$\begin{array}{rl} \forall x, y. & P(x) \vee Q(x, y) \\ \forall x. & \neg P(A) \vee R(B, x) \\ \hline \end{array}$$

$$\begin{array}{rl} \forall x, y. & P(x) \vee Q(x, y) \\ \forall z. & \neg P(A) \vee R(B, z) \\ \hline & (Q(x, y) \vee R(B, z))\theta \\ & Q(A, y) \vee R(B, z) \end{array}$$

$\theta = \{x/A\}$

$$\begin{array}{c} P(x_1) \vee Q(x_1, y_1) \\ \neg P(A) \vee R(B, x_2) \\ \hline (Q(x_1, y_1) \vee R(B, x_2))\theta \\ Q(A, y_1) \vee R(B, x_2) \end{array}$$

$\theta = \{x_1/A\}$

Curiosity Killed the Cat

1	D(Fido)	a
2	O(J,Fido)	a
3	¬ D(y) v ¬ O(x,y) v L(x)	b
4	¬ L(x) v ¬ A(y) v ¬ K(x,y)	c
5	K(J,T) v K(C,T)	d
6	C(T)	e
7	¬ C(x) v A(x)	f
8	¬ K(C,T)	Neg
9	K(J,T)	5,8
10	A(T)	6,7 {x/T}
11	¬ L(J) v ¬ A(T)	4,9 {x/J, y/T}
12	¬ L(J)	10,11
13	¬ D(y) v ¬ O(J,y)	3,12 {x/J}
14	¬ D(Fido)	13,2 {y/Fido}
15	•	14,1

Proving Validity

- How do we use resolution refutation to prove something is valid?
- Normally, we prove a sentence is entailed by the set of axioms
- Valid sentences are entailed by the empty set of sentences
- To prove validity by refutation, negate the sentence and try to derive contradiction.

Example

- Syllogism

$$\left(\forall x. P(x) \rightarrow Q(x)\right) \wedge P(A) \rightarrow Q(A)$$

- Negate and convert to clausal form

$$\neg\left(\left(\forall x. P(x) \rightarrow Q(x)\right) \wedge P(A) \rightarrow Q(A)\right)$$

$$\neg\left(\neg\left(\forall x. \neg P(x) \vee Q(x)\right) \vee \neg P(A) \vee Q(A)\right)$$

$$\left(\forall x. \neg P(x) \vee Q(x)\right) \wedge P(A) \wedge \neg Q(A)$$

$$\left(\neg P(x) \vee Q(x)\right) \wedge P(A) \wedge \neg Q(A)$$

Example

- Do proof

1.	$\neg P(x) \vee Q(x)$	
2.	$P(A)$	
3.	$\neg Q(A)$	
4.	$Q(A)$	1,2
5.	$\blacksquare$	3,4

Green's Trick

- Use resolution to get answers to existential queries

$$\exists x.\ \text{Mortal}(x)$$

1.	$\neg \text{Man}(x) \vee \text{Mortal}(x)$	
2.	$\text{Man}(\mathit{Socrates})$	
3.	$\neg \text{Mortal}(x) \vee \text{Answer}(x)$	
4.	$\text{Mortal}(\mathit{Socrates})$	1,2
5.	$\text{Answer}(\mathit{Socrates})$	3,5

Equality

- Special predicate in syntax and semantics; need to add something to our proof system
- Could add another special inference rule called paramodulation
- Instead, we will axiomatize equality as an equivalence relation

$$\forall x. \mathrm{Eq}(x, x)$$
$$\forall x, y. \mathrm{Eq}(x, y) \rightarrow \mathrm{Eq}(y, x)$$
$$\forall x, y, z. \mathrm{Eq}(x, y) \wedge \mathrm{Eq}(y, z) \rightarrow \mathrm{Eq}(x, z)$$

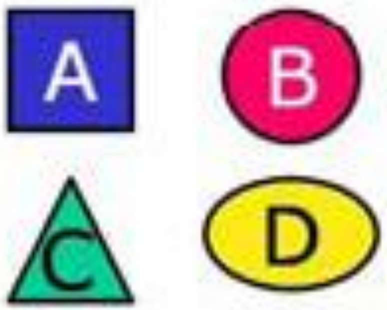
- For every predicate, allow substitutions

$$\forall x, y. \mathrm{Eq}(x, y) \rightarrow (\mathrm{P}(x) \rightarrow \mathrm{P}(y))$$

Proof Example

- Let's go back to our old geometry domain and try to prove what the hat of A is
- Axioms in FOL (plus equality axioms)

$$\text{Above}(A, C)$$
$$\text{Above}(B, D)$$
$$\neg\exists x.\ \text{Above}(x, A)$$
$$\neg\exists x.\ \text{Above}(x, B)$$
$$\forall x, y.\ \text{Above}(x, y) \rightarrow \text{hat}(y) = x$$
$$\forall x.\ (\neg\exists y.\ \text{Above}(y, x)) \rightarrow \text{hat}(x) = x$$



- Desired conclusion: $\exists x.\ \text{hat}(A) = x$
- Use Green's trick to get the binding of x

The Clauses

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.		
11.		
12.		

The Query

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.	~Eq(hat(A), x) v Answer(x)	

The Proof

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.	~Eq(hat(A), x) v Answer(x)	conclusion
11.	Above(sk(A), A) v Answer(A)	6, 10 {x/A}
12.	Answer(A)	11, 3 {x/sk(A)}

Hat of D

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.	~Eq(hat(D), x) v Answer(x)	conclusion
11.	~Above(x,D) v Answer(x)	5, 10 {x1/x}
12.	Answer(B)	11, 2 {x/B}

Who is Jane’s Lower

- Jane’s lover drives a red car
- Fred is the only person who drives a red car
- Who is Jane’s lover?

1.	Drives(lover(Jane))	
2.	~Drives(x) v Eq(x,Frec)	
3.	~Eq(lover(Jane),x) v Answer(x)	
4.	Eq(lover(Jane), Fred)	1,2 {x/lover(Jane)}
5.	Answer(Fred)	3,4 {x/Fred}