Knowledge Representation (II)

Paolo Turrini

Department of Computing, Imperial College London

Introduction to Artificial Intelligence

The main reference

Stuart Russell and Peter Norvig
Artificial Intelligence: a modern approach
Chapter 9

Today's class

- Start with a knowledge base and try to prove something interesting
- Various methods for doing so, with different computational properties

Artificial Intelligence and Law

The formalization of legislation and the development of computer systems to assist with legal problem solving provide a rich domain for developing and testing artificial-intelligence technology.

Marek Sergot, Fariba Sadri, Robert Kowalski, (and others)
The British Nationality Act as a logic program
Communications of the ACM, 1986

An informal knowledge base

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.

An informal knowledge base

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 Colonel West is a criminal

Example knowledge base contd.

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

Example knowledge base contd.

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

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

Example knowledge base contd.

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

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

Nono . . . has some missiles

Example knowledge base contd.

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

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

Nono ... has some missiles

$Owns(Nono, M_1)$ and $Missile(M_1)$

Example knowledge base contd.

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

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

Nono ... has some missiles

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

Example knowledge base contd.

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

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

Nono . . . has some missiles

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

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

Example knowledge base contd.

Missiles are weapons:

Example knowledge base contd.

Missiles are weapons:

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

Example knowledge base contd.

Missiles are weapons:

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

An enemy of America "counts as" hostile:

Example knowledge base contd.

Missiles are weapons:

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

An enemy of America "counts as" hostile:

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

Example knowledge base contd.

Missiles are weapons:

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

An enemy of America "counts as" hostile:

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

West, who is American.

Example knowledge base contd.

Missiles are weapons:

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

An enemy of America "counts as" hostile:

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

West, who is American.

$American(West)$

Example knowledge base contd.

Missiles are weapons:

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

An enemy of America "counts as" hostile:

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

West, who is American.

$American(West)$

The country Nono, an enemy of America ...

Example knowledge base contd.

Missiles are weapons:

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

An enemy of America "counts as" hostile:

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

West, who is American.

$American(West)$

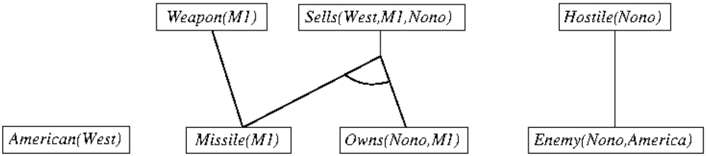
The country Nono, an enemy of America ...

$Enemy(Nono, America)$

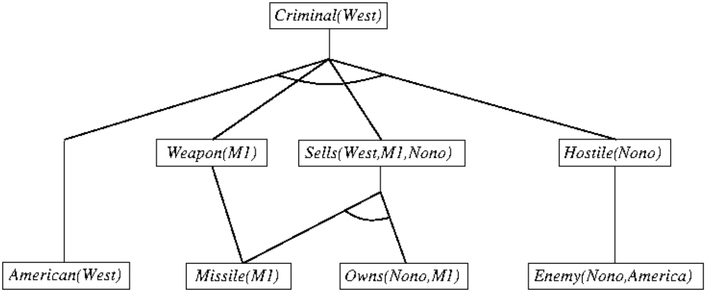
Forward chaining proof

$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono,M1)}$ $\boxed{Enemy(Nono,America)}$

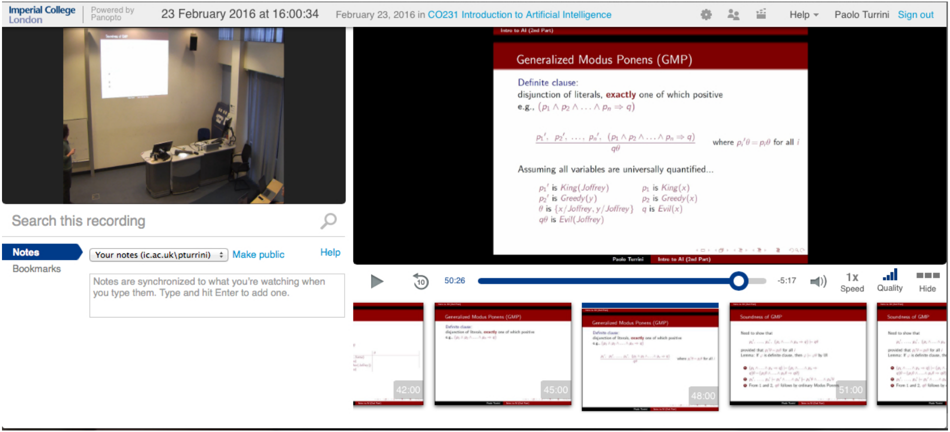
Forward chaining proof



Forward chaining proof



Footage from Tuesday



GMP: recall...

Definite clause:
disjunction of literals, **exactly** one of which positive
e.g., $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

$$\frac{p_1{}',\;\; p_2{}',\; \ldots,\; p_n{}',\;\; (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \qquad \text{where } p_i{}'\theta = p_i\theta \text{ for all } i$$

Assuming all variables are universally quantified...

$p_1{}'$ is $King(Joffrey)$	p_1 is $King(x)$
$p_2{}'$ is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/Joffrey, y/Joffrey\}$	q is $Evil(x)$
$q\theta$ is $Evil(Joffrey)$	

Forward chaining algorithm

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*
inputs: *KB*, the knowledge base, a set of definite clauses
α, the (atomic) query
local variables: *new*, the new sentences inferred at each iteration

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* **do**
      $q' \leftarrow q\theta$
      if q' does not already unify in *KB* or *new* **then**
        add q' to *new*
        $\phi \leftarrow$ UNIFY(q', α)
        if ϕ is not *fail* **then return** ϕ
  add *new* to *KB*
return *false*

First iteration

On the first iteration...

$$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$$

First iteration

On the first iteration...

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

has unsatisfied premises

First iteration

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

First iteration

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

is satisfied with $\{x/M_1\}$ and $Sells(West, M_1, Nono)$ is added

First iteration

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

First iteration

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

is satisfied with $\{x/M_1\}$ and $Weapon(M_1)$ is added

First iteration

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

First iteration

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

is satisfied with $\{x/Nono\}$ and $Hostile(Nono)$ is added

First iteration

So in total we have added...

First iteration

So in total we have added...

$Sells(West, M_1, Nono)$

First iteration

So in total we have added...

$Sells(West, M_1, Nono)$

$Weapon(M_1)$

First iteration

So in total we have added...

$Sells(West, M_1, Nono)$

$Weapon(M_1)$

$Hostile(Nono)$

Second iteration

On the second iteration...

Second iteration

On the second iteration...

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

Second iteration

On the second iteration...

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

is satisfied with $\{x/West, y/M_1, z/Nono\}$
and $Criminal(West)$ is added

Second iteration

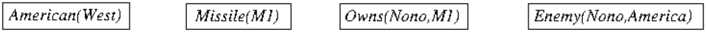
On the second iteration...

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

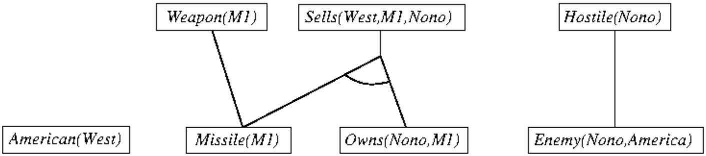
is satisfied with $\{x/West, y/M_1, z/Nono\}$
and $Criminal(West)$ is added

Yay! :)

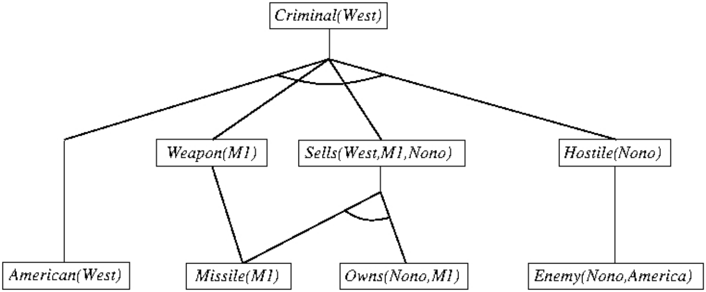
Forward chaining proof



Forward chaining proof



Forward chaining proof



Properties of forward chaining

- Sound (because of soundness of GMP)

Properties of forward chaining

- Sound (because of soundness of GMP)
- Complete for first-order (entailed!) definite clauses

Properties of forward chaining

- Sound (because of soundness of GMP)
- Complete for first-order (entailed!) definite clauses
- May not terminate in general if α is not entailed

Properties of forward chaining

- Sound (because of soundness of GMP)
- Complete for first-order (entailed!) definite clauses
- May not terminate in general if α is not entailed
- It's inefficient:

Properties of forward chaining

- Sound (because of soundness of GMP)
- Complete for first-order (entailed!) definite clauses
- May not terminate in general if α is not entailed
- It's inefficient:
    - but there can be improvements

Properties of forward chaining

- Sound (because of soundness of GMP)
- Complete for first-order (entailed!) definite clauses
- May not terminate in general if α is not entailed
- It's inefficient:
  - but there can be improvements
  - well... matching conjunctive premises against known facts is NP-hard

Properties of forward chaining

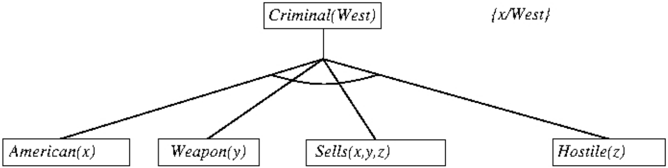
- Sound (because of soundness of GMP)
- Complete for first-order (entailed!) definite clauses
- May not terminate in general if α is not entailed
- It's inefficient:
  - but there can be improvements
  - well... matching conjunctive premises against known facts is NP-hard

Forward chaining is widely used in deductive databases

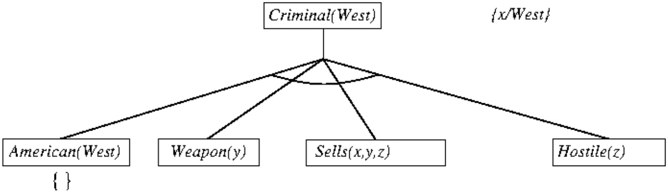
Backward chaining example

$\boxed{Criminal(West)}$

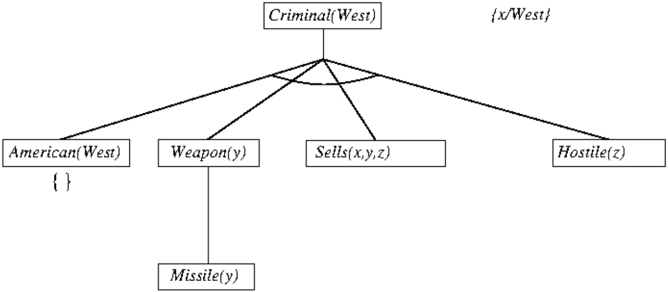
Backward chaining example



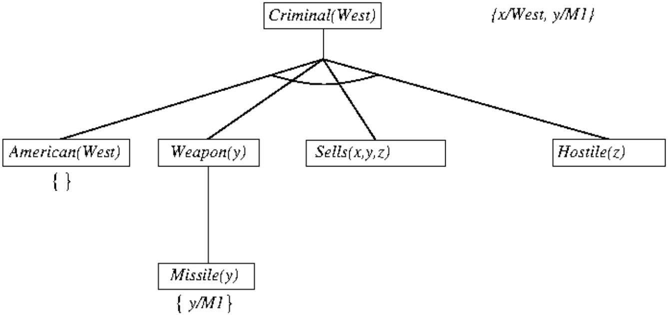
Backward chaining example



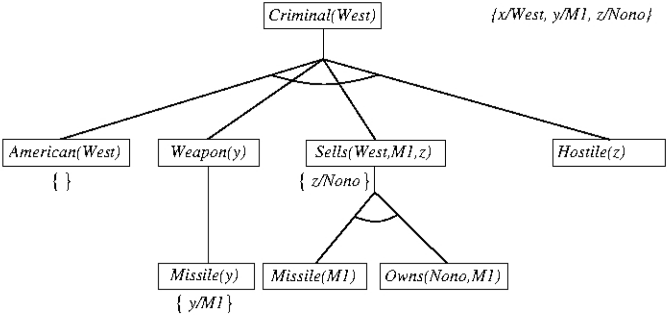
Backward chaining example



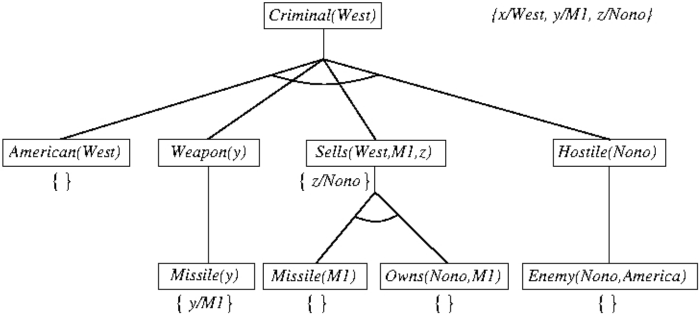
Backward chaining example



Backward chaining example



Backward chaining example



Backward chaining algorithm

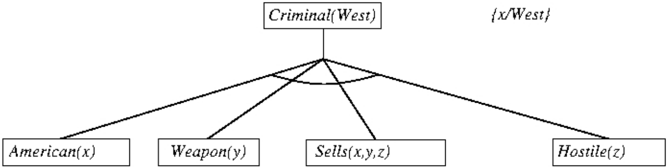
function FOL-BC-ASK($KB, goals, \theta$) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query (θ already applied)
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: $answers$, a set of substitutions,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

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

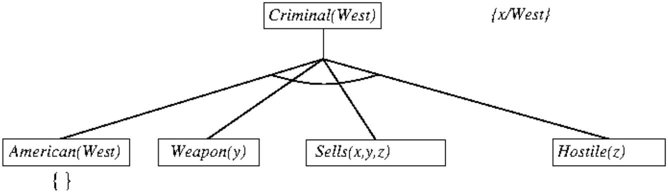
Backward chaining example

$$\boxed{Criminal(West)}$$

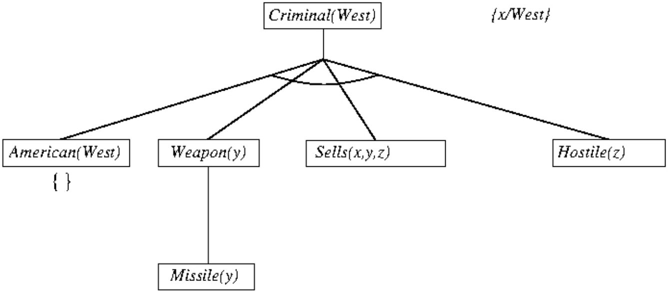
Backward chaining example



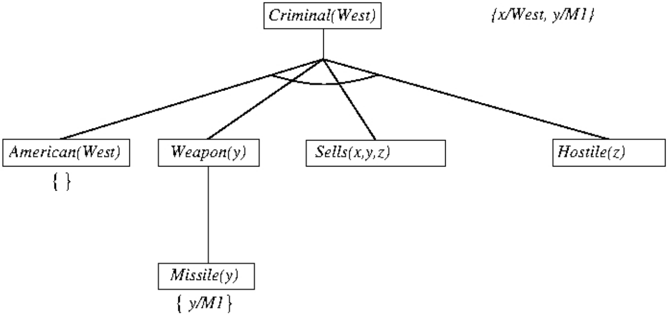
Backward chaining example



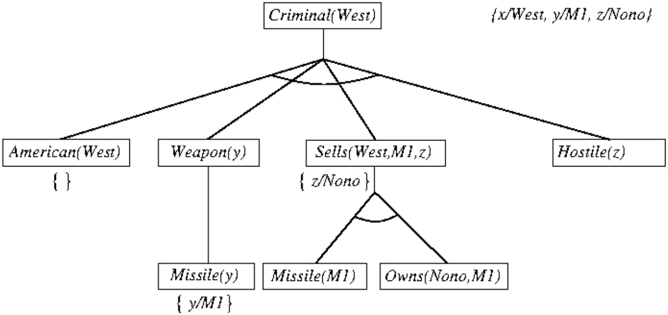
Backward chaining example



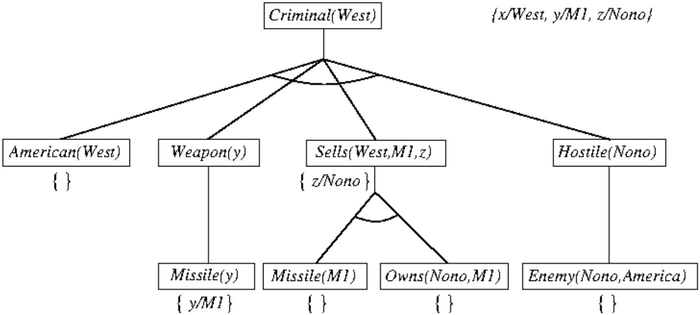
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

Depth-first recursive proof search

Properties of backward chaining

Depth-first recursive proof search

- space is linear in size of proof

Properties of backward chaining

Depth-first recursive proof search

- space is linear in size of proof
- Incomplete due to infinite loops

Properties of backward chaining

Depth-first recursive proof search

- space is linear in size of proof
- Incomplete due to infinite loops

Widely used for logic programming

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\neg Rich(x) \vee Unhappy(x)$$

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\text{UNIFY}(\ell_i, \neg m_j) = \theta$.

For example,

$$\neg Rich(x) \vee Unhappy(x)$$
$$Rich(Berlusconi)$$

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Berlusconi)\end{array}}{Unhappy(Berlusconi)}$$

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x)\\ Rich(Berlusconi)\end{array}}{Unhappy(Berlusconi)}$$

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Berlusconi)\end{array}}{Unhappy(Berlusconi)}$$

with $\theta = \{x/Berlusconi\}$

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Berlusconi)\end{array}}{Unhappy(Berlusconi)}$$

with $\theta = \{x/Berlusconi\}$

Apply resolution steps to $CNF(KB \wedge \neg\alpha)$

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x \ [\forall y \ \ Pizza(y) \Rightarrow Dislikes(x, y)] \Rightarrow [\exists y \ \ Dislikes(y, x)]$$

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x\ [\forall y\ \ Pizza(y) \Rightarrow Dislikes(x,y)] \Rightarrow [\exists y\ \ Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x\ [\forall y\ \ Pizza(y) \Rightarrow Dislikes(x,y)] \Rightarrow [\exists y\ \ Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x\ [\neg\forall y\ \ \neg Pizza(y) \vee Dislikes(x,y)] \vee [\exists y\ \ Dislikes(y,x)]$$

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x\ [\forall y\ \ Pizza(y) \Rightarrow Dislikes(x,y)] \Rightarrow [\exists y\ \ Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x\ [\neg\forall y\ \ \neg Pizza(y) \vee Dislikes(x,y)] \vee [\exists y\ \ Dislikes(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x\varphi \ \equiv \exists x\ \neg\varphi$, $\ \neg\exists x\varphi \ \equiv \forall x\ \neg\varphi$:

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x \ [\forall y \ \ Pizza(y) \Rightarrow Dislikes(x,y)] \Rightarrow [\exists y \ \ Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x \ [\neg \forall y \ \ \neg Pizza(y) \vee Dislikes(x,y)] \vee [\exists y \ \ Dislikes(y,x)]$$

2. Move $\neg$ inwards: $\neg \forall x \varphi \ \equiv \exists x \ \neg \varphi$, $\ \neg \exists x \varphi \ \equiv \forall x \ \neg \varphi$:

$$\forall x \ [\exists y \ \ \neg(\neg Pizza(y) \vee Dislikes(x,y))] \vee [\exists y \ \ Dislikes(y,x)]$$

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x \ \ [\forall y \ \ Pizza(y) \Rightarrow Dislikes(x,y)] \Rightarrow [\exists y \ \ Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x \ \ [\neg\forall y \ \ \neg Pizza(y) \vee Dislikes(x,y)] \vee [\exists y \ \ Dislikes(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x\varphi \ \equiv \exists x \ \neg\varphi$, $\ \neg\exists x\varphi \ \equiv \forall x \ \neg\varphi$:

$$\forall x \ \ [\exists y \ \ \neg(\neg Pizza(y) \vee Dislikes(x,y))] \vee [\exists y \ \ Dislikes(y,x)]$$
$$\forall x \ \ [\exists y \ \ \neg\neg Pizza(y) \wedge \neg Dislikes(x,y)] \vee [\exists y \ \ Dislikes(y,x)]$$

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x \; [\forall y \;\; Pizza(y) \; \Rightarrow \; Dislikes(x,y)] \; \Rightarrow \; [\exists y \;\; Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x \;\; [\neg\forall y \;\; \neg Pizza(y) \vee Dislikes(x,y)] \vee [\exists y \;\; Dislikes(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x \varphi \;\; \equiv \exists x \;\; \neg\varphi$, $\;\; \neg\exists x \varphi \;\; \equiv \forall x \;\; \neg\varphi$:

$$\forall x \;\; [\exists y \;\; \neg(\neg Pizza(y) \vee Dislikes(x,y))] \vee [\exists y \;\; Dislikes(y,x)]$$
$$\forall x \;\; [\exists y \;\; \neg\neg Pizza(y) \wedge \neg Dislikes(x,y)] \vee [\exists y \;\; Dislikes(y,x)]$$
$$\forall x \;\; [\exists y \;\; Pizza(y) \wedge \neg Dislikes(x,y)] \vee [\exists y \;\; Dislikes(y,x)]$$

Conversion to CNF

Everyone who dislikes pizzas is disliked by someone:

$$\forall x\ [\forall y\ \ Pizza(y) \Rightarrow Dislikes(x,y)] \Rightarrow [\exists y\ \ Dislikes(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x\ [\neg \forall y\ \ \neg Pizza(y) \lor Dislikes(x,y)] \lor [\exists y\ \ Dislikes(y,x)]$$

2. Move $\neg$ inwards: $\neg \forall x \varphi \ \equiv \exists x\ \neg \varphi$, $\neg \exists x \varphi \ \equiv \forall x\ \neg \varphi$:

$$\forall x\ [\exists y\ \ \neg(\neg Pizza(y) \lor Dislikes(x,y))] \lor [\exists y\ \ Dislikes(y,x)]$$
$$\forall x\ [\exists y\ \ \neg\neg Pizza(y) \land \neg Dislikes(x,y)] \lor [\exists y\ \ Dislikes(y,x)]$$
$$\forall x\ [\exists y\ \ Pizza(y) \land \neg Dislikes(x,y)] \lor [\exists y\ \ Dislikes(y,x)]$$

Conversion to CNF contd.

3. Standardise variables: each quantifier should use a different one

Conversion to CNF contd.

3. Standardise variables: each quantifier should use a different one

$$\forall x \;\; [\exists y \;\; Pizza(y) \land \neg Dislikes(x, y)] \lor [\exists z \;\; Dislikes(z, x)]$$

Conversion to CNF contd.

3. Standardise variables: each quantifier should use a different one

$$\forall x \;\; [\exists y \;\; Pizza(y) \wedge \neg Dislikes(x, y)] \vee [\exists z \;\; Dislikes(z, x)]$$

4. Skolemize: a more general form of existential instantiation.

Conversion to CNF contd.

3. Standardise variables: each quantifier should use a different one

$$\forall x \;\; [\exists y \;\; Pizza(y) \land \neg Dislikes(x, y)] \lor [\exists z \;\; Dislikes(z, x)]$$

4. Skolemize: a more general form of existential instantiation.
Each existential variable is replaced by a Skolem function

Conversion to CNF contd.

3. Standardise variables: each quantifier should use a different one

$$\forall x \; [\exists y \; \; Pizza(y) \wedge \neg Dislikes(x, y)] \vee [\exists z \; \; Dislikes(z, x)]$$

4. Skolemize: a more general form of existential instantiation.
Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

Conversion to CNF contd.

3. Standardise variables: each quantifier should use a different one

$$\forall x \;\; [\exists y \;\; Pizza(y) \wedge \neg Dislikes(x, y)] \vee [\exists z \;\; Dislikes(z, x)]$$

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$$\forall x \;\; [Pizza(F(x)) \wedge \neg Dislikes(x, F(x))] \vee Dislikes(G(x), x)$$

Conversion to CNF contd.

5. Drop universal quantifiers:

$$[Pizza(F(x)) \wedge \neg Dislikes(x, F(x))] \vee Dislikes(G(x), x)$$

Conversion to CNF contd.

5. Drop universal quantifiers:

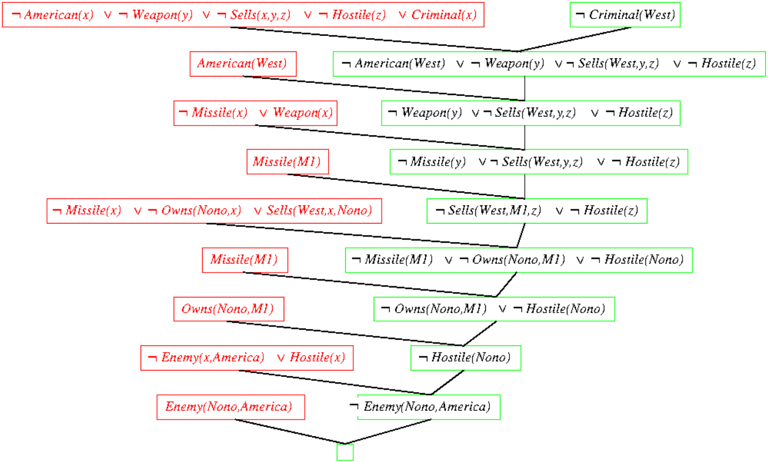
$$[Pizza(F(x)) \wedge \neg Dislikes(x, F(x))] \vee Dislikes(G(x), x)$$

6. Distribute $\wedge$ over $\vee$:

$$[Pizza(F(x)) \vee Dislikes(G(x), x)] \wedge$$

$$\wedge [\neg Dislikes(x, F(x)) \vee Dislikes(G(x), x)]$$

Resolution proof: definite clauses



What we have seen

Inference methods with different structure and properties

- Forward-chaining (deductive databases)
- Backward-chaining (logic programming)
- Resolution (full-first order logic)

Coming next

- Knowledge and uncertainty: what if we don't know exactly?
- How to handle uncertain information:
  - representation
  - reasoning

Appendix: Universal Instantiation

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

$$\frac{\forall v \;\; \alpha}{\alpha(\{v/g\})}$$

for any variable v and ground term g

Appendix: Universal Instantiation

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

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

for any variable v and ground term g
E.g., $\forall x \ \ King(x) \land Greedy(x) \ \Rightarrow \ Evil(x)$ yields

Appendix: Universal Instantiation

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

$$\frac{\forall v \;\; \alpha}{\alpha(\{v/g\})}$$

for any variable v and ground term g
E.g., $\forall x \;\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields

$$\begin{array}{l} King(Joffrey) \wedge Greedy(Joffrey) \Rightarrow Evil(Joffrey) \\ King(Aerys) \wedge Greedy(Aerys) \Rightarrow Evil(Aerys) \\ King(Father(Joffrey)) \wedge Greedy(Father(Joffrey)) \Rightarrow Evil(Father(Joffrey)) \\ \quad \vdots \end{array}$$

Appendix: Existential instantiation (EI)

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

$$\frac{\exists v \quad \alpha}{\alpha(\{v/g\})}$$

Appendix: Existential instantiation (EI)

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

$$\frac{\exists v \;\; \alpha}{\alpha(\{v/g\})}$$

E.g., $\exists x \;\; Crown(x) \land OnHead(x, John)$ yields

Appendix: Existential instantiation (EI)

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

$$\frac{\exists v \ \ \alpha}{\alpha(\{v/g\})}$$

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

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

Appendix: Existential instantiation (EI)

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

$$\frac{\exists v \quad \alpha}{\alpha(\{v/g\})}$$

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

Appendix: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

Appendix: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence;

Appendix: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence;

the new KB is **not** equivalent to the old,

Appendix: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence;

the new KB is **not** equivalent to the old,

but is satisfiable iff the old KB was satisfiable