Building KBs in Prolog and Inference in FOPC

Deepak Kumar
November 2021

1

Knowledge Engineering in FOPC

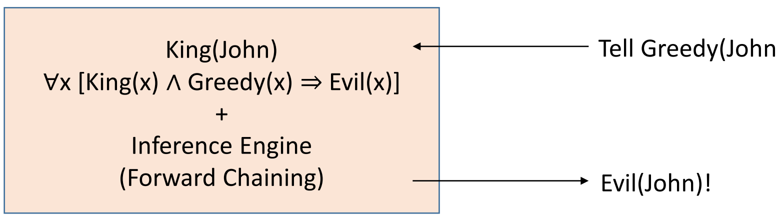
- Identify the task
- Assemble relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base

2

2

Forward Chaining Inference

- Tell-Ask Systems

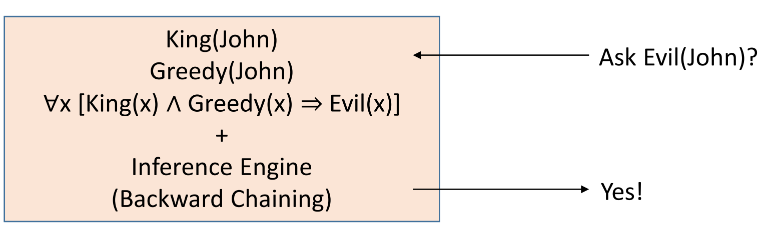


3

3

Backward Chaining Inference

- Tell-Ask Systems



4

4

Backward Chaining Inference

- Tell-Ask Systems

King(John)
Greedy(John)
∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]
+
Inference Engine
(Backward Chaining)

← Ask Evil(John)?

→ Yes!

- **Requires wffs to be in Definite Clause form!**

5

5

Definite Clauses & Horn Clauses

- **Clause:** A disjunction of literals

  e.g. ¬R ∨ ¬P ∨ ¬Q

- **Definite Clause:** A clause with exactly one positive literal

  e.g. ¬R ∨ P ∨ ¬Q

- **Horn Clause:** A clause with at most one positive literal

  e.g. ¬R ∨ P ∨ ¬Q

  ¬R ∨ ¬Q

  P

All definite clauses are Horn Clauses.

6

6

Definite Clauses in FOPC

- Disjunction of literals of which exactly one is positive

  i.e. $\neg \omega_1 \lor \neg \omega_2 \lor \ldots \lor \neg \omega_{n-1} \lor \omega_n$

7

7

Definite Clauses in FOPC

- Disjunction of literals of which exactly one is positive

  i.e. $\neg \omega_1 \lor \neg \omega_2 \lor \ldots \lor \neg \omega_{n-1} \lor \omega_n \equiv \omega_1 \land \omega_2 \land \ldots \land \omega_{n-1} \Rightarrow \omega_n$
- A definite clause is either a fact: American(JoeBiden)
- Or, an implication whose antecedent is a conjunction of positive literals
- Can have variables, but all must be universally quantified (∀)

  ∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]
- We can then rewrite the wff without the quantifier: King(x) ∧ Greedy(x) ⇒ Evil(x)

  ¬ King(x) ∨ ¬ Greedy(x) ∨ Evil(x)
- Most Knowledge bases can be converted to this form.

8

8

Logic Programming (Prolog)

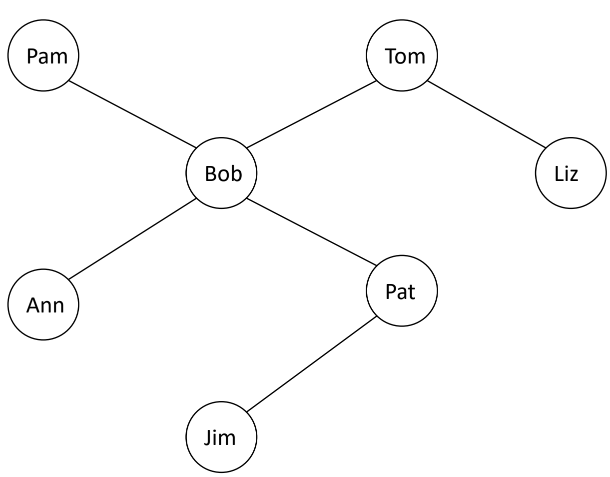
- A program is a set of definite clauses (facts, $\omega_1 \wedge \omega_2 \wedge \ldots \wedge \omega_{n-1} \Rightarrow \omega_n$)
- The Syntax is different from FOPC
  - Variables : written in uppercase
  - Constants : written in lowercase
  - Relations : written beginning with lowercase letter
  - Conjunction (∧) : comma (,)
  - Implications : written as Prolog rules

$\omega_1 \wedge \omega_2 \Rightarrow \omega_n$	:	C :- A, B.
∀x [Missile(x) ⇒ Weapon(x)]	:	weapon(X) :- missile(X).
∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]	:	evil(X) :- king(X), greedy(X).

9

9

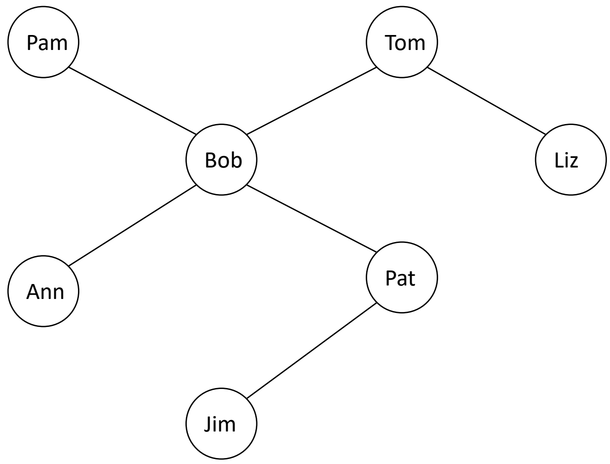
A Simple Prolog Program



10

10

A Simple Prolog Program

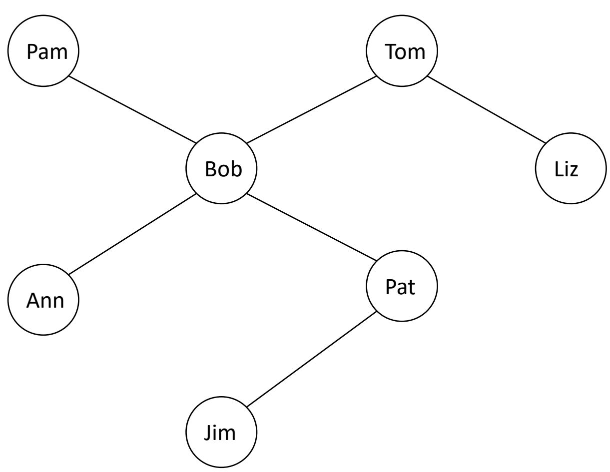



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(bob, pat).
true
```

11

11

A Simple Prolog Program

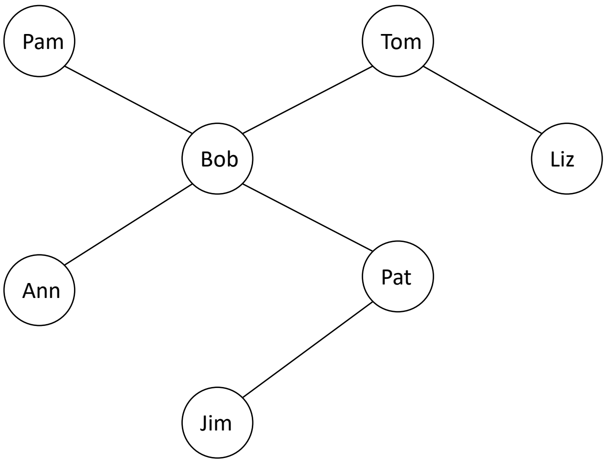



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(bob, pat).
true

?- parent(liz, pat).
false
```

12

12

A Simple Prolog Program

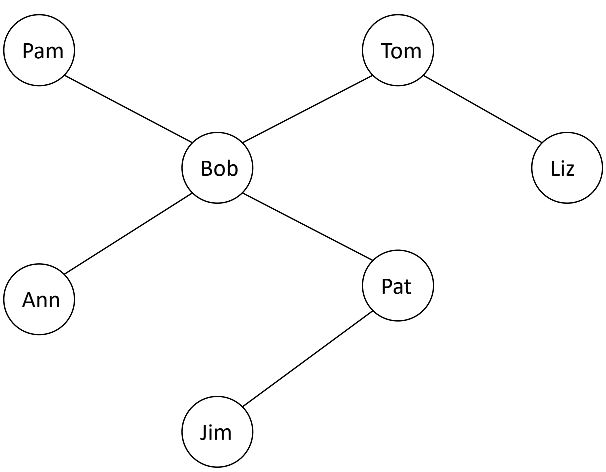



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(bob, pat).
true

?- parent(liz, pat).
false

?- parent(tom, deepak).
false
```

13

13

A Simple Prolog Program

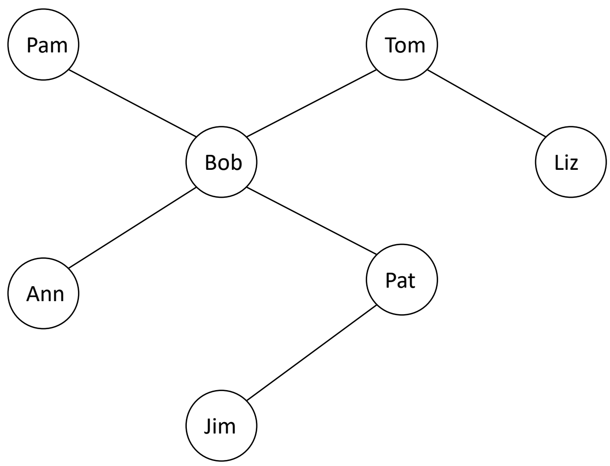



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom
```

14

14

A Simple Prolog Program

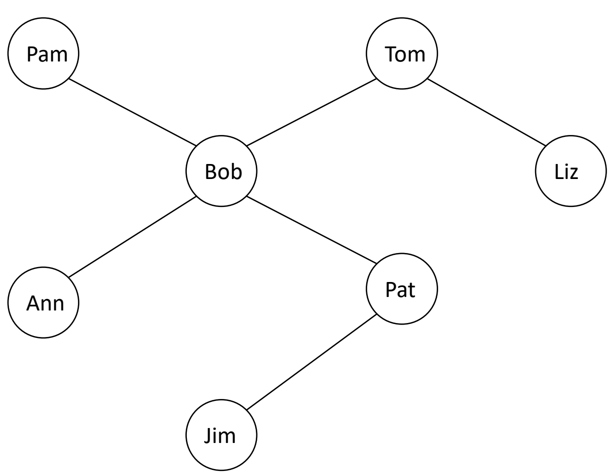



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann
```

15

15

A Simple Prolog Program

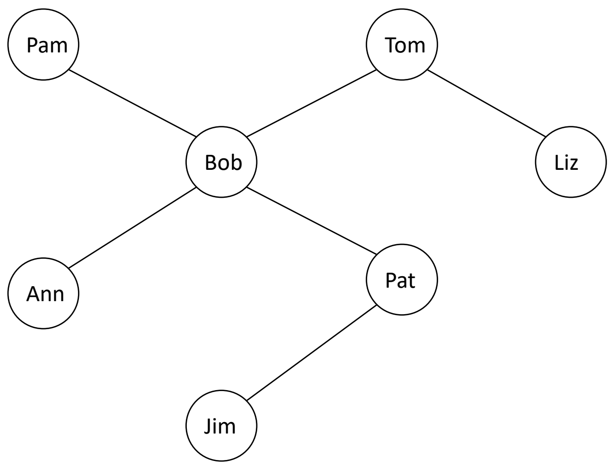



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann              What about Pat???
```

16

16

A Simple Prolog Program

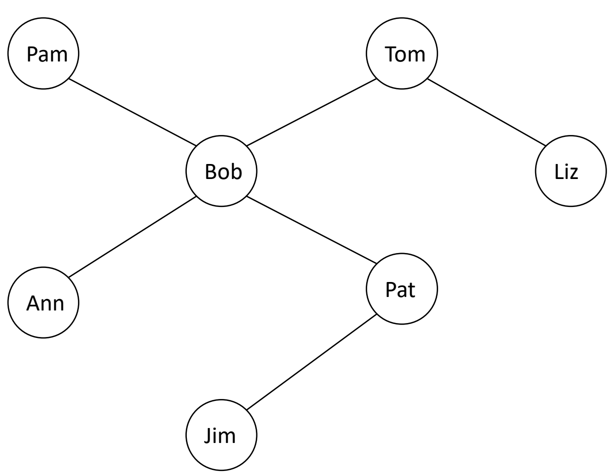



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann;            What about Pat???
X = pat
```

17

17

A Simple Prolog Program

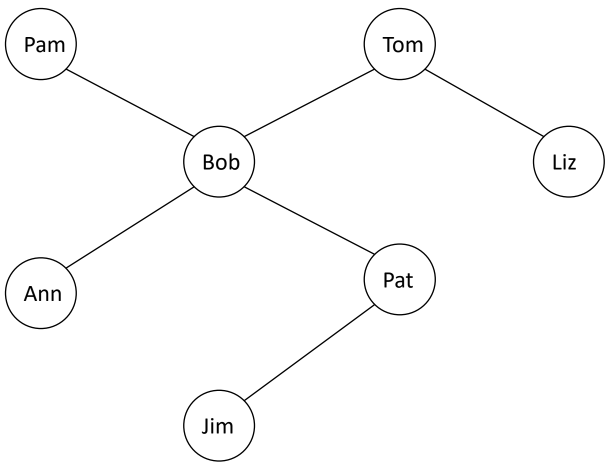



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann;            What about Pat???
X = pat.
```

18

18

A Simple Prolog Program

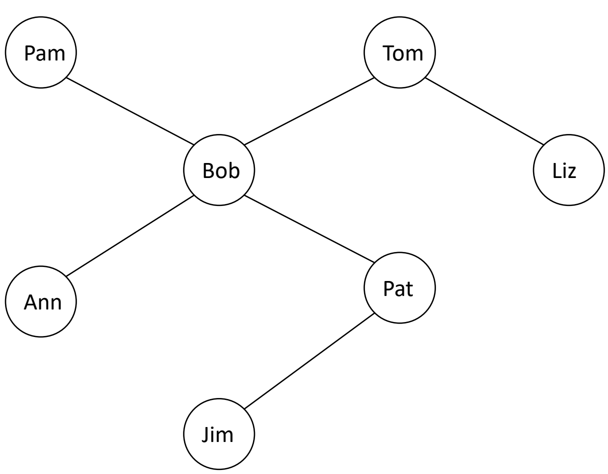



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, Y).
X = pam
Y = bob
```

19

19

A Simple Prolog Program

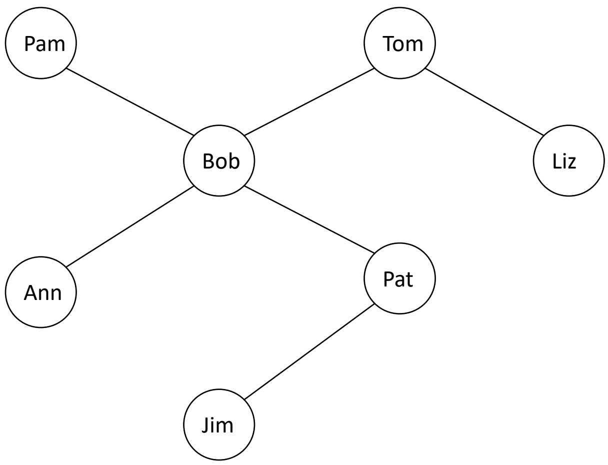



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, Y).
X = pam
Y = bob;
```

20

20

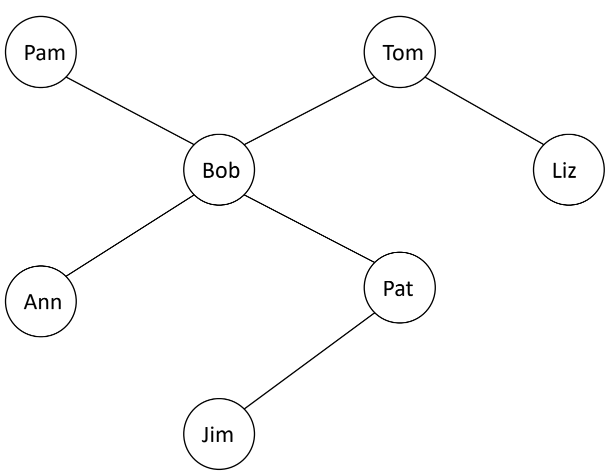
A Simple Prolog Program




```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, Y).
X = pam
Y = bob;

X = tom
Y = bob
X = tom,
Y = liz ;
X = bob,
Y = ann ;
...
```

21

A Simple Prolog Program

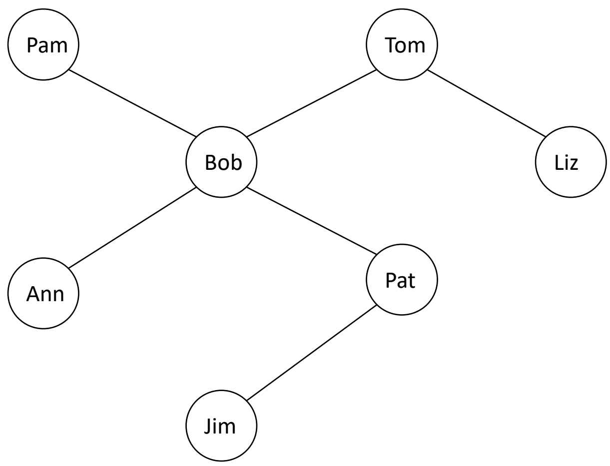



```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
female(pam).
female(liz).
female(ann).
female(pat).
male(tom).
male(bob).
male(jim).
```

22

22

A Simple Prolog Program




```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
female(pam).
female(liz).
female(ann).
female(pat).
male(tom).
male(bob).
male(jim).
mother(X, Y) :- parent(X, Y), female(X).
```

23

23

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
$\forall x \forall y \forall z\ [Parent(x, y) \wedge Parent(y, z) \Rightarrow GrandParent(x, z)]$

Sibling(x, y): x is a sibling of y
$\forall x \forall y \forall z \forall w\ [\ father(z, x) \wedge father(z, y) \wedge mother(w, x) \wedge mother(w, y) \wedge (x \neq y) \Rightarrow Sibling(x, y)\ ]$

AuntOrUncle(x, y): x is an aunt or uncle of y
$\forall x \forall y \forall w\ [Sibling(x, y) \wedge Parent(y, w) \Rightarrow AuntOrUncle(x, w)]$

$\forall w \forall x \forall y \forall w\ [Married(x, y) \wedge Sibling(y, w) \wedge Parent(w, z) \Rightarrow AuntOrUncle(x, z)]$

Aunt(x, y): x is an aunt of y
$\forall x \forall y\ [Female(x) \wedge AuntOrUncle(x, y) \Rightarrow Aunt(x, y)]$

Ancester(x, y): x is an ancestor
$\forall x \forall y\ [Parent(x, y) \Rightarrow Ancester(x, y)]$

$\forall x \forall y \forall z\ [Parent(x, z) \wedge Ancestor(z, y) \Rightarrow Ancester(x, y)]$

24

24

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y): x is a sibling of y
∀x ∀y ∀z ∀w [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y)]

AuntOrUncle(x, y): x is an aunt or uncle of y
∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

∀w ∀x ∀y ∀w [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]

Aunt(x, y): x is an aunt of y
∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancester(x, y): x is an ancestor
∀x ∀y [Parent(x, y) ⇒ Ancester(x, y)]

∀x ∀y ∀z [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancester(x, y)]

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
```

25

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y): x is a sibling of y
∀x ∀y ∀z ∀w [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y)]

AuntOrUncle(x, y): x is an aunt or uncle of y
∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

∀w ∀x ∀y ∀w [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]

Aunt(x, y): x is an aunt of y
∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancester(x, y): x is an ancestor
∀x ∀y [Parent(x, y) ⇒ Ancester(x, y)]

∀x ∀y ∀z [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancester(x, y)]

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
                 mother(W, X), mother(W, Y), not(X = Y).
```

26

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
$\forall x \forall y \forall z\ [Parent(x, y) \land Parent(y, z) \Rightarrow GrandParent(x, z)]$

Sibling(x, y): x is a sibling of y
$\forall x \forall y \forall z \forall w\ [\ father(z, x) \land father(z, y) \land mother(w, x) \land mother(w, y) \land (x \neq y) \Rightarrow Sibling(x, y)\ ]$

AuntOrUncle(x, y): x is an aunt or uncle of y
$\forall x \forall y \forall w\ [Sibling(x, y) \land Parent(y, w) \Rightarrow AuntOrUncle(x, w)]$

$\forall w \forall x \forall y \forall w\ [Married(x, y) \land Sibling(y, w) \land Parent(w, z) \Rightarrow AuntOrUncle(x, z)]$

Aunt(x, y): x is an aunt of y
$\forall x \forall y\ [Female(x) \land AuntOrUncle(x, y) \Rightarrow Aunt(x, y)]$

Ancester(x, y): x is an ancestor
$\forall x \forall y\ [Parent(x, y) \Rightarrow Ancester(x, y)]$

$\forall x \forall y \forall z\ [Parent(x, z) \land Ancestor(z, y) \Rightarrow Ancester(x, y)]$

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
            mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).
```

27

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
$\forall x \forall y \forall z\ [Parent(x, y) \land Parent(y, z) \Rightarrow GrandParent(x, z)]$

Sibling(x, y): x is a sibling of y
$\forall x \forall y \forall z \forall w\ [\ father(z, x) \land father(z, y) \land mother(w, x) \land mother(w, y) \land (x \neq y) \Rightarrow Sibling(x, y)\ ]$

AuntOrUncle(x, y): x is an aunt or uncle of y
$\forall x \forall y \forall w\ [Sibling(x, y) \land Parent(y, w) \Rightarrow AuntOrUncle(x, w)]$

$\forall w \forall x \forall y \forall w\ [Married(x, y) \land Sibling(y, w) \land Parent(w, z) \Rightarrow AuntOrUncle(x, z)]$

Aunt(x, y): x is an aunt of y
$\forall x \forall y\ [Female(x) \land AuntOrUncle(x, y) \Rightarrow Aunt(x, y)]$

Ancester(x, y): x is an ancestor
$\forall x \forall y\ [Parent(x, y) \Rightarrow Ancester(x, y)]$

$\forall x \forall y \forall z\ [Parent(x, z) \land Ancestor(z, y) \Rightarrow Ancester(x, y)]$

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
            mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
```

28

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
$\forall x \forall y \forall z$ [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y): x is a sibling of y
$\forall x \forall y \forall z \forall w$ [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y)]

AuntOrUncle(x, y): x is an aunt or uncle of y
$\forall x \forall y \forall w$ [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

$\forall w \forall x \forall y \forall w$ [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]

Aunt(x, y): x is an aunt of y
$\forall x \forall y$ [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancester(x, y): x is an ancestor
$\forall x \forall y$ [Parent(x, y) ⇒ Ancester(x, y)]

$\forall x \forall y \forall z$ [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancester(x, y)]

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
        mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).
```

29

From FOPC To Prolog

- **Knowledge Base – General statements about family relationships**

GrandParent(x, y) : x is a grand parent of y
$\forall x \forall y \forall z$ [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y): x is a sibling of y
$\forall x \forall y \forall z \forall w$ [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y)]

AuntOrUncle(x, y): x is an aunt or uncle of y
$\forall x \forall y \forall w$ [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

$\forall w \forall x \forall y \forall w$ [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]

Aunt(x, y): x is an aunt of y
$\forall x \forall y$ [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancester(x, y): x is an ancestor
$\forall x \forall y$ [Parent(x, y) ⇒ Ancester(x, y)]

$\forall x \forall y \forall z$ [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancester(x, y)]

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
        mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).
ancester(X, Y) :- parent(X, Y).
ancester(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

30

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base

31

31

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.

32

32

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.

- Decide on a vocabulary of predicates, functions, and constants

33

33

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.

- Decide on a vocabulary of **predicates**, functions, and constants

American(x)	: x is an American
Enemy(x, y)	: x is an enemy of y
Hostile(x)	: x is hostile
Criminal(x)	: x is a criminal
Missile(x)	: x is a missile
Weapon(x)	: x is a weapon
Owns(x, y)	: x owns y
Sells(x, y, z)	: x sells y to z

34

34

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

*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.*

- Decide on a vocabulary of predicates, functions, and **constants**

American(x)	: x is an American	America
Enemy(x, y)	: x is an enemy of y	Nono
Hostile(x)	: x is hostile	CWest
Criminal(x)	: x is a criminal	
Missile(x)	: x is a missile	
Weapon(x)	: x is a weapon	
Owns(x, y)	: x owns y	
Sells(x, y, z)	: x sells y to z	

35

35

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

*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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

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

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

36

36

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

∃x [Owns(Nono, x) ∧ Missile(x)]

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

37

37

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

∃x [Owns(Nono, x) ∧ Missile(x)]

∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

38

38

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

  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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

39

39

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

  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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

  ∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

40

40

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

  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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

  ∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

41

41

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

  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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

  ∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

- Encode a description of specific problem instance

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

42

42

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

  *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.***

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

  ∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

- Encode a description of specific problem instance

  American(Cwest)

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

43

43

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

  *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.*

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

  ∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

- Encode a description of specific problem instance

  American(Cwest)

  Enemy(Nono, America)

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

44

44

Knowledge Engineering in FOPC

- **Knowledge Base**

  ∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]

  ∃x [Owns(Nono, x) ∧ Missile(x)]

  ∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

  ∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

  ∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

  American(CWest)

  Enemy(Nono, America)

- **Pose queries to the inference procedure to get answers**

  Is Colonel West a criminal?

America
Nono
Cwest

American(x)
Enemy(x, y)
Hostile(x)
Criminal(x)
Missile(x)
Weapon(x)
Owns(x, y)
Sells(x, y, z)

45

45

FOPC – Inference Rules

- Inference rules for Quantifiers (∀, ∃)
  - Universal Instantiation
  - Existential Instantiation
- Generalized Modus Ponens
  - Unification
- Forward & Backward Chaining
  - Definite Clauses
  - Logic Programming in Prolog
- Resolution
  - Reductio ad Absurdum

46

46

Inference Rules for Quantifiers (∀, ∃)

- Universal Instantiation Rule

  Can infer any sentence obtained by substituting a ground term (a term without variables) for a universally quantified variable (∀)

  e.g.

  $\forall x\ [King(x) \land Greedy(x) \Rightarrow Evil(x)]$

  We can replace/substitute John (a ground term) for x in that wff:

  $King(John) \land Greedy(John) \Rightarrow Evil(John)$ when {x = John}

47

47

Substitutions ({x = John}) formally

A substitution is written as

$\theta = \{ v/x \}$ - replace v by x

Given a wff, α

SUBST(θ, α) - apply substitution θ to α

e.g. $\theta = \{ x/John \}$ $\alpha = \forall x\ [King(x) \land Greedy(x) \Rightarrow Evil(x)]$

then $SUBST(\theta, \alpha) = King(John) \land Greedy(John) \Rightarrow Evil(John)$

48

48

Inference Rules for Quantifiers (∀, ∃)

- Universal Instantiation Rule, formally

$$\frac{\forall v\ [\alpha]}{\text{SUBST}(\{v/g\}, \alpha)}$$ for any variable, v and ground term, g.

Can infer any sentence obtained by substituting a ground term (a term without variables) for a universally quantified variable (∀)

e.g.

$\forall x\ [King(x) \land Greedy(x) \Rightarrow Evil(x)]$

$King(John) \land Greedy(John) \Rightarrow Evil(John)$ when $\{x/John\}$
$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$ when $\{x/Richard\}$
...

49

49

Inference Rules for Quantifiers (∀, ∃)

- Existential Instantiation Rule, formally

$$\frac{\exists v\ [\alpha]}{\text{SUBST}(\{v/k\}, \alpha)}$$ for any variable, v and a constant symbol, k.

Can infer a sentence obtained by substituting a constant symbol for a existentially quantified variable (∃)

e.g.

$\alpha = \exists x\ [Owns(Nono, x) \land Missile(x)]$

Let constant be M1, then SUBST({x/M1}, α) gives

$Owns(Nono, M1) \land Missile(M1)$

50

50

Inference Rules for Quantifiers (∀, ∃)

- Existential Instantiation Rule, formally

$$\frac{\exists v\ [\alpha]}{\text{SUBST}(\{v/k\}, \alpha)}$$ for any variable, v and a constant symbol, k.

Can infer a sentence obtained by substituting a constant symbol for a existentially quantified variable (∃)

e.g.

α = ∃x [Owns(Nono, x) ∧ Missile(x)]

Let constant be M1, then SUBST({x/M1}, α) gives

Owns(Nono, M1) ∧ Missile(M1)

M1 is called a Skolem Constant

Generalized Modus Ponens

For atomic sentences p_i, p_i', and q

where there is a substitution θ such that

SUBST(θ, p_i') = SUBST(θ, p_i), for all i

p_1'
p_2'
…
p_n'
$p_1' \wedge p_2' \wedge \ldots \wedge p_n' \Rightarrow q$

SUBST(θ, q)

e.g.

King(John)
Greedy(John)
∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]
θ is {x/John}
SUBST(θ, Evil(x)) is Evil(John)

Unification - UNIFY(p, q) = θ

How to find substitutions that make different logical expressions look identical?

For wffs p and q (sentences with universally quantified variables)

UNIFY(p, q) = θ

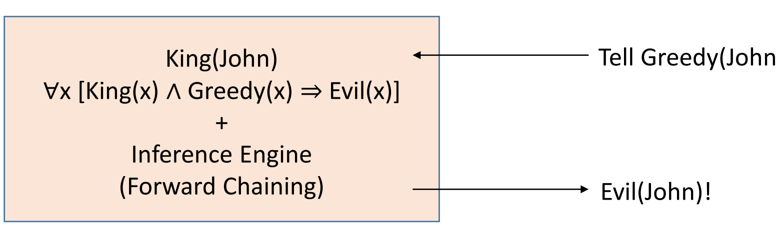
e.g.	p = Knows(John, x) q = Knows(John, Mary)	θ = {x/Mary}
	p = Knows(John, x) q = Knows(y, Bill)	θ = {y/John, x/Bill}
	p = Knows(John, x) q = Knows(x, Elizabeth)	θ = { } i.e. Fail! (no unification)
	p = Knows(John, x_1) q = Knows(x_2, Elizabeth)	θ = {x_1/Elizabeth, x_2/John}

53

53

Forward Chaining Inference

- Tell-Ask Systems



54

54

Backward Chaining Inference

- Tell-Ask Systems

King(John)
Greedy(John)
$\forall x\ [King(x) \land Greedy(x) \Rightarrow Evil(x)]$
+
Inference Engine
(Backward Chaining)

← Ask Evil(John)?

→ Yes!

55

55

Backward Chaining Inference

- Tell-Ask Systems

King(John)
Greedy(John)
$\forall x\ [King(x) \land Greedy(x) \Rightarrow Evil(x)]$
+
Inference Engine
(Backward Chaining)

← Ask Evil(John)?

→ Yes!

- **Requires wffs to be in Definite Clause form!**

56

56

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

American(Cwest) Missile(M1) Owns(Nono, M1) Enemy(Nono, America)

57

57

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

Weapon(M1)

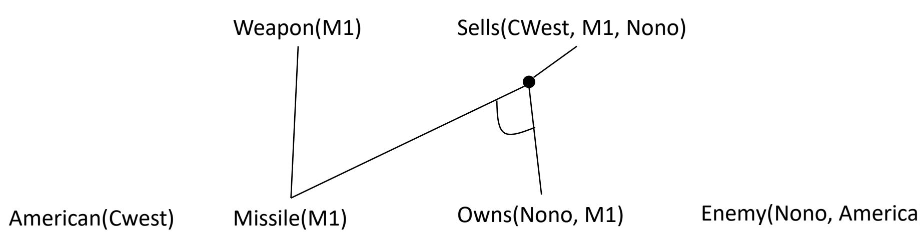
American(Cwest) Missile(M1) Owns(Nono, M1) Enemy(Nono, America)

58

58

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

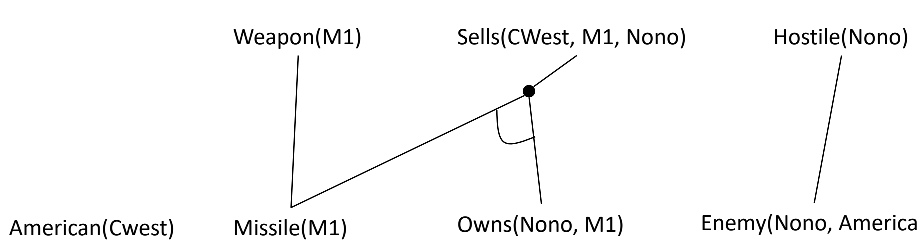


59

59

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

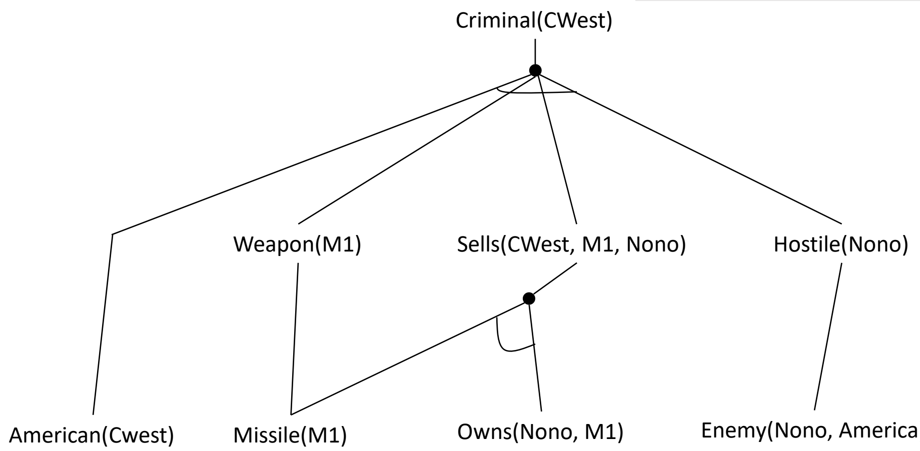


60

60

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)



61

61

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

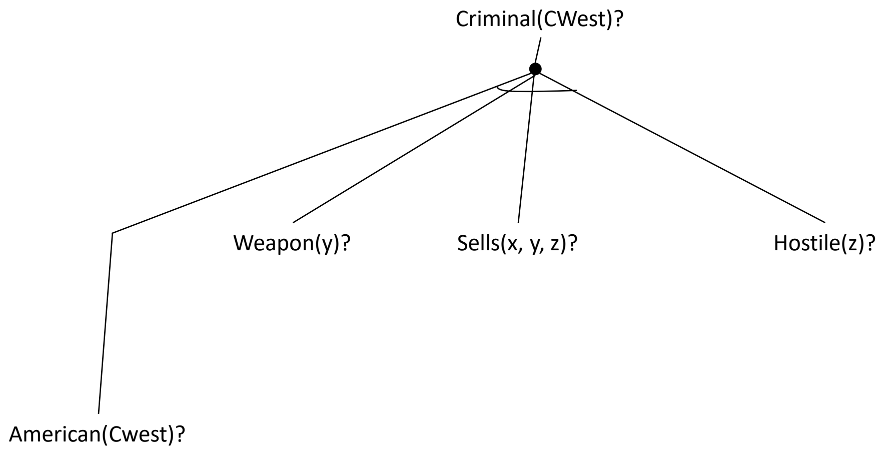
Criminal(CWest)?

62

62

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

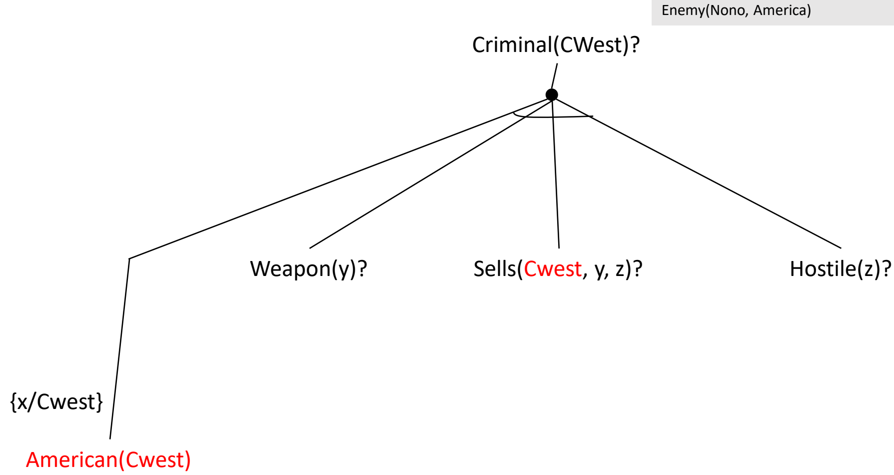


63

63

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

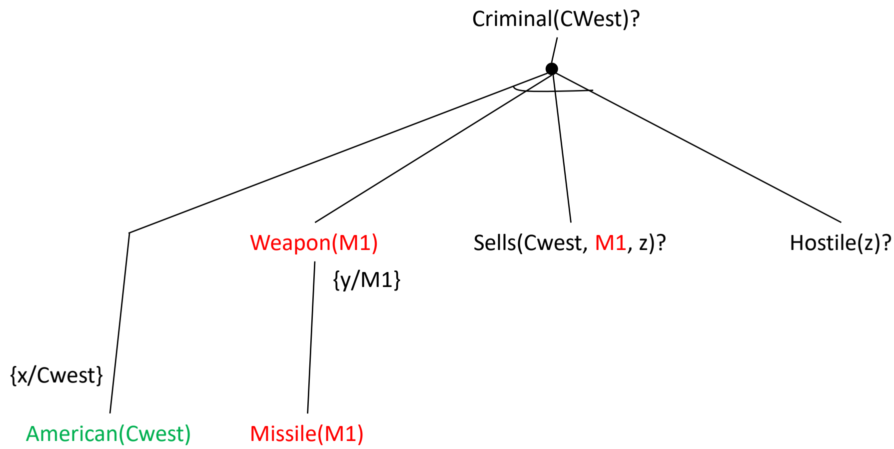


64

64

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

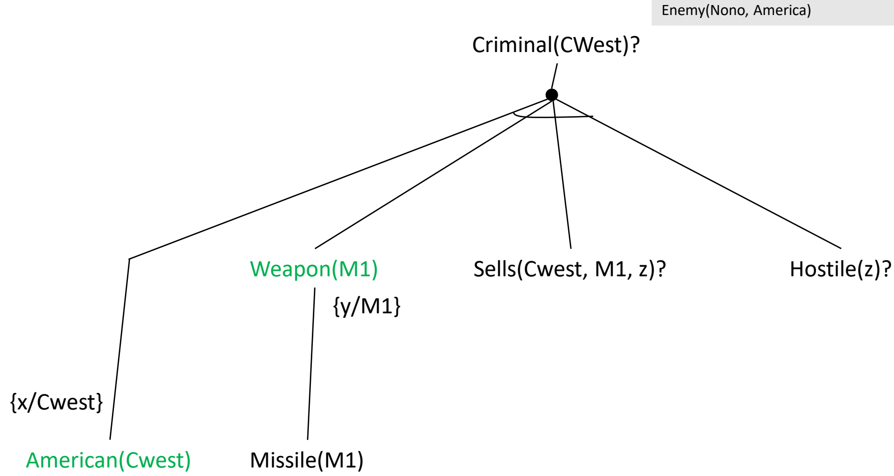


65

65

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

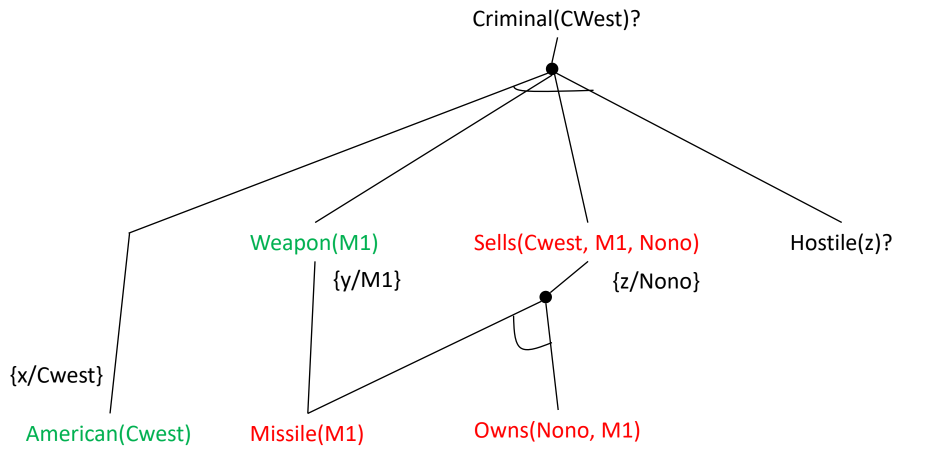


66

66

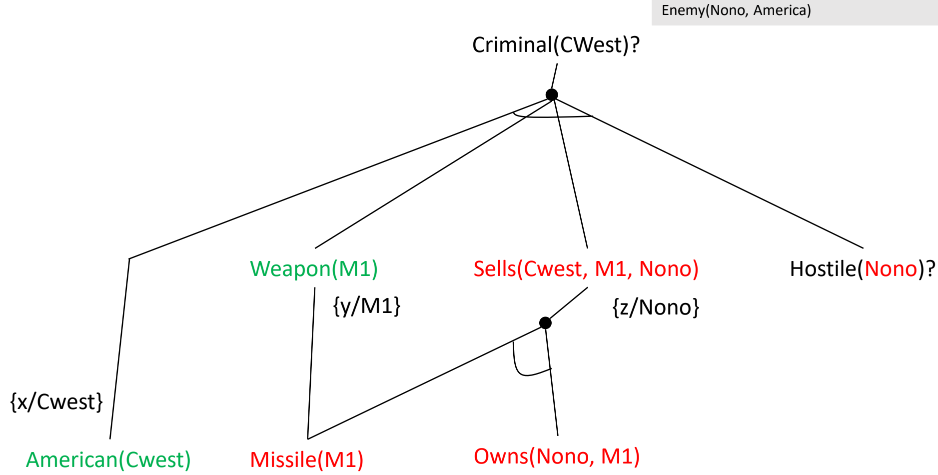
Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)



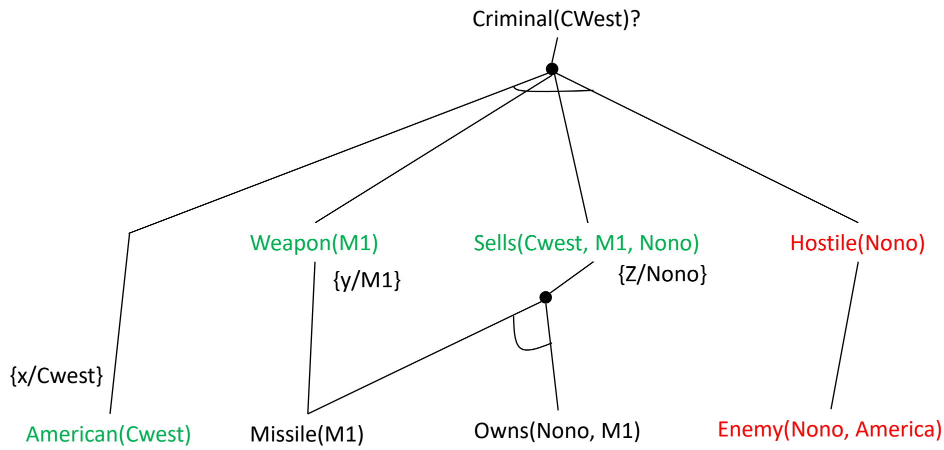
Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)



Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

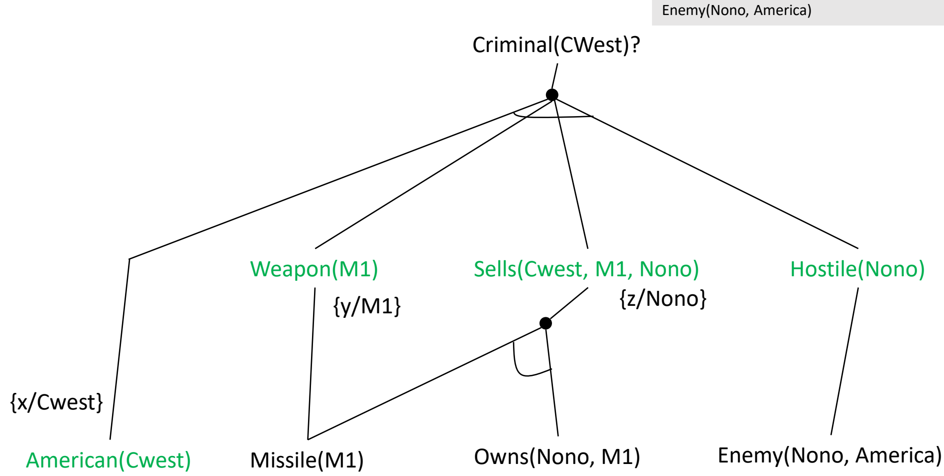


69

69

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)



70

70

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)