CMSC 373 Artificial Intelligence
Fall 2025
08-Logic in PROLOG

Deepak Kumar
Bryn Mawr College

1

Knowledge Engineering in FOPC

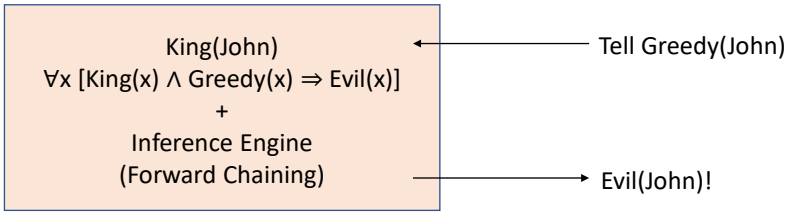
- Identify the task
- Assemble relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base

2

2

Forward Chaining Inference

- Tell-Ask Systems

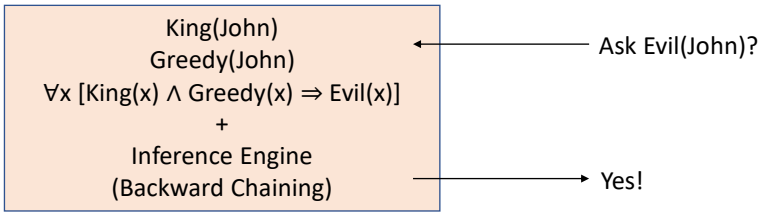


3

3

Backward Chaining Inference

- Tell-Ask Systems

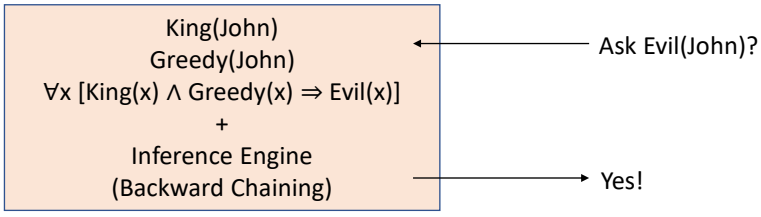


4

4

Backward Chaining Inference

- Tell-Ask Systems



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

5

5

Definite Clauses in FOPC

- Disjunction of literals of which exactly one is positive

  i.e. $\neg\omega_1 \vee \neg\, \omega_2 \vee \ldots \vee \neg\, \omega_{n-1} \vee \omega_n \quad \equiv \quad \omega_1 \wedge \omega_2 \wedge \ldots \wedge \omega_{n-1} \Rightarrow \omega_n$
- 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.

6

6

Logic Programming (Prolog)

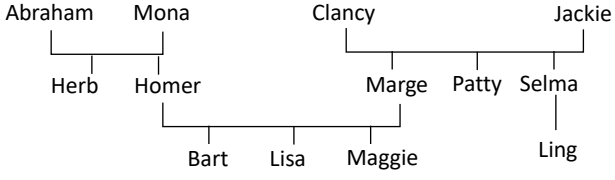
- A program is a set of definite clauses (facts, $\omega_1 \wedge \omega_2 \wedge \ldots \wedge \omega_{n-1} \Rightarrow \omega_n$)
- The Syntax is different from FOPC
  - Variables : written in uppercase
  - Constants : written in lowercase
  - Relations : written beginning with lowercase letter
  -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

$\omega_1 \wedge \omega_2 \Rightarrow \omega_n$	:	ω_n :- ω_1, ω_2.
$\forall x$ [Missile(x) $\Rightarrow$ Weapon(x)]	:	weapon(X) :- missile(X)
$\forall x$ [King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)]	:	evil(X) :- king(X), greedy(X).

7

7

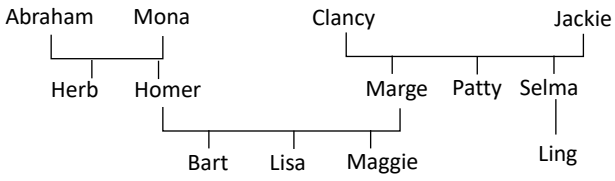
A Simple Prolog Program



8

8

A Simple Prolog Program

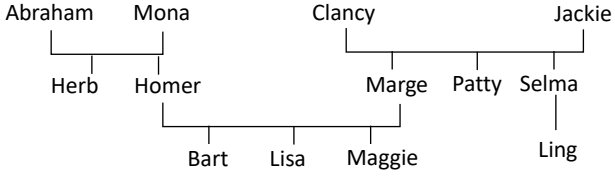



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(homer, marge).
true
```

9

9

A Simple Prolog Program

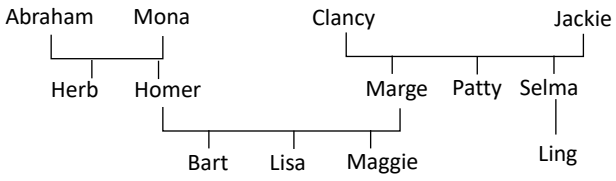



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(homer, marge).
true

?- married(homer, jackie).
false
```

10

10

A Simple Prolog Program

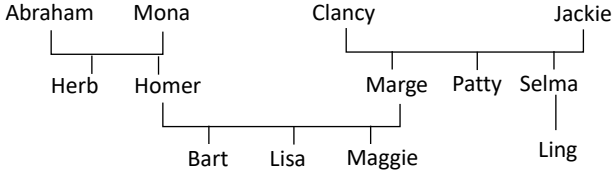



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(homer, marge).
true

?- married(homer, jackie).
false

?- married(deepak, jackie).
false
```

11

11

A Simple Prolog Program

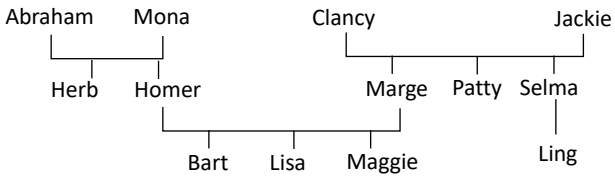



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(X, homer).
X = marge
```

12

12

A Simple Prolog Program

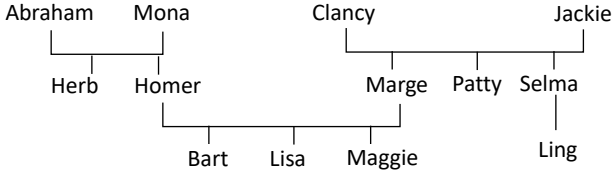



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(X, homer).
X = marge

?- married(homer, X).
X = marge
```

13

13

A Simple Prolog Program

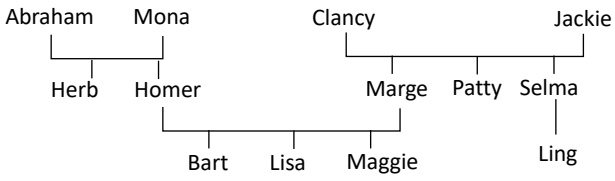



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(X, Y).
X = Abraham
Y = mona
```

14

14

A Simple Prolog Program

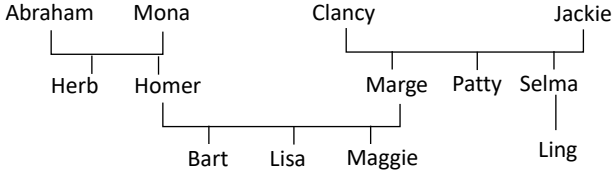



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(X, Y).
X = abraham
Y = mona;
X = mona
Y = abraham
```

15

15

A Simple Prolog Program

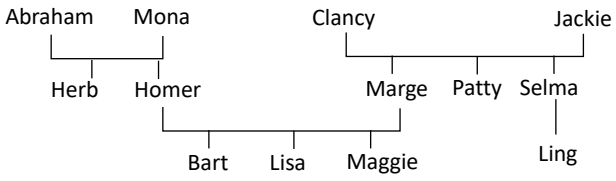



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(X, Y).
X = abraham
Y = mona;
X = mona
Y = Abraham
X = clancy,
Y = jackie ;
X = jackie,
Y = clancy ;
X = homer,
Y = marge ;
X = marge,
Y = homer
```

16

16

A Simple Prolog Program

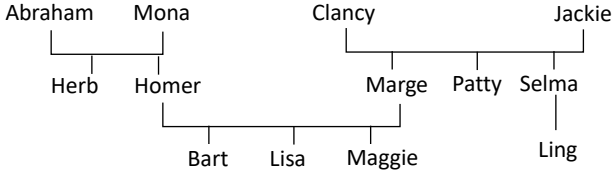



```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

?- married(X, Y).
X = abraham
Y = mona;
X = mona
Y = Abraham
X = clancy,
Y = jackie .
```

17

17

A Simple Prolog Program



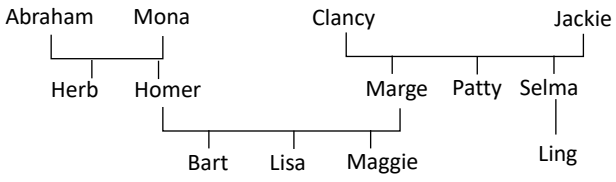

```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

% males and females
% female(x) : x is female
% male(x)   : x is male
male(abraham).
male(clancy).
male(herb).
male(homer).
male(bart).
female(mona).
female(jackie).
female(marge).
female(patty).
female(selma).
female(ling).
```

```
% parents
% parent(x, y) : x is a parent of y
parent(abraham, herb).
parent(mona, herb).
parent(abraham, homer).
parent(mona, homer).
parent(clancy, marge).
parent(jackie, marge).
parent(clancy, patty).
parent(jackie, patty).
parent(clancy, selma).
parent(jackie, selma).
parent(homer, bart).
parent(marge, bart).
parent(homer, lisa).
parent(marge, lisa).
parent(homer, maggie).
parent(marge, maggie).
parent(selma, ling).
```

18

18

A Simple Prolog Program



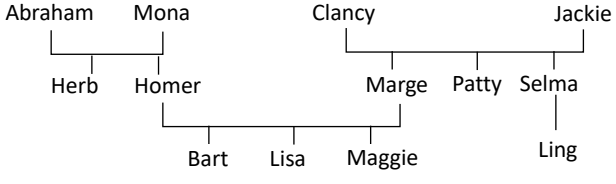

```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

% males and females
% female(x) : x is female
% male(x)   : x is male
male(abraham).
male(clancy).
male(herb).
male(homer).
male(bart).
female(mona).
female(jackie).
female(marge).
female(patty).
female(selma).
female(ling).
```

```
% parents
% parent(x, y) : x is a parent of y
parent(abraham, herb).
parent(mona, herb).
parent(abraham, homer).
parent(mona, homer).
parent(clancy, marge).
parent(jackie, marge).
parent(clancy, patty).
parent(jackie, patty).
parent(clancy, selma).
parent(jackie, selma).
parent(homer, bart).
parent(marge, bart).
parent(homer, lisa).
parent(marge, lisa).
parent(homer, maggie).
parent(marge, maggie).
parent(selma, ling).

%who is Marge the mother of?
?- parent(marge, X), female(marge).
X = bart ;
X = lisa ;
X = maggie.
```

19

19

A Simple Prolog Program




```
% married(x, y) : x is married to y
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

% males and females
% female(x) : x is female
% male(x)   : x is male
male(abraham).
male(clancy).
male(herb).
male(homer).
male(bart).
female(mona).
female(jackie).
female(marge).
female(patty).
female(selma).
female(ling).
```

```
% parents
% parent(x, y) : x is a parent of y
parent(abraham, herb).
parent(mona, herb).
parent(abraham, homer).
parent(mona, homer).
parent(clancy, marge).
parent(jackie, marge).
parent(clancy, patty).
parent(jackie, patty).
parent(clancy, selma).
parent(jackie, selma).
parent(homer, bart).
parent(marge, bart).
parent(homer, lisa).
parent(marge, lisa).
parent(homer, maggie).
parent(marge, maggie).
parent(selma, ling).

% Define mother(x, y) rule
?- mother(X, Y) :-  parent(X, Y),
                    female(X).
```

20

20

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)]

21

21

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
$\forall x \forall y \forall z$ [parent(z, x) $\wedge$ parent(z, 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)]

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

22

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
$\forall x \forall y \forall z \forall w$ [parent(z, x) $\wedge$ parent(z, 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)]

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

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
$\forall x \forall y \forall z$ [parent(z, x) $\wedge$ parent(z, 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)]

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

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 [parent(z, x) ∧ parent(z, 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, Y) :- parent(Z, X), parent(Z, 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).
```

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 [parent(z, x) ∧ parent(z, 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, Y) :- parent(Z, X), parent(Z, 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).
```

26

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 [parent(z, x) ∧ parent(z, 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, Y) :- parent(Z, X), parent(Z, 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), ancester(Z, Y).
```

27

Vocabulary

Knowledge Engineering
FOPC
Knowledge Base
Tell-Ask Systems
Forward Chaining Inference
Backward Chaining Inference
Definite Clauses
Logic Programming
PROLOG

28

28

References

- M. Wooldridge: *A Brief History of Artificial Intelligence*. Flatiron Books, 2020.
- Michael Genesereth & Nils Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kauffman, 1987.
- Ivan Bratko, *PROLOG Programming for Artificial Intelligence,* Fourth Edition, Addison-Wesley, 1990.
- Hector Levesque, *Thinking as Computation*, MIT Press, 2012.

29