Facts?

- Hazlenuts?
- tomsRedCar?
- 2Ideas?
- Prolog?

Facts?

- Hazlenuts? No
- tomsRedCar? **Yes**
- 2Ideas? No
- Prolog? No

Cannot start of capital letter or a number

Given the database below…

blue_box.
red_box.
green_circle.
blue_circle.
orange_triangle

- green_circle?
- circle_green?
- red_triangle?
- red_box?
- orange_triangle?

Given the database below…

blue_box.
red_box.
green_circle.
blue_circle.
orange_triangle

- green_circle? **Yes**
- circle_green? **No**
- red_triangle? **No**
- red_box? **Yes**
- orange_triangle? **No**

Unification

- eats(fred,tomatoes).
- eats(Whom,What).

- eats(fred,Food).
- eats(Person,jim).

- cd(29,beatles,sgt_pepper).
- cd(A,B,help).

- f(X,a)
- f(a,X)

Unification

- eats(fred,tomatoes).
- eats(Whom,What).

- *Whom = fred*
- *What = tomatoes*

- eats(fred,Food).
- eats(Person,jim).

- cd(29,beatles,sgt_pepper).
- cd(A,B,help).

- f(X,a)
- f(a,X)

Unification

- eats(fred,tomatoes).
- eats(Whom,What).

- *Whom = fred*
- *What = tomatoes*

- eats(fred,Food).
- eats(Person,jim).

- *Person=fred*
- *Food=jim*

- cd(29,beatles,sgt_pepper).
- cd(A,B,help).

- f(X,a)
- f(a,X)

Unification

- eats(fred,tomatoes).
- eats(Whom,What).

- *Whom = fred*
- *What = tomatoes*

- eats(fred,Food).
- eats(Person,jim).

- *Person=fred*
- *Food=jim*

- cd(29,beatles,sgt_pepper).
- cd(A,B,help).

- *sgt_pepper and help do not unify*

- f(X,a)
- f(a,X)

Unification

- eats(fred,tomatoes).
- eats(Whom,What).

- *Whom = fred*
- *What = tomatoes*

- eats(fred,Food).
- eats(Person,jim).

- *Person=fred*
- *Food=jim*

- cd(29,beatles,sgt_pepper).
- cd(A,B,help).

- *sgt_pepper and help do not unify*

- f(X,a)
- f(a,X)

- *X = a*

Unification

- likes(jane,X)
- likes(X,jim)

- f(X,Y)
- f(P,P)

- f(foo,L)
- f(A1,A1)

Unification

- likes(jane,X)
- likes(X,jim)

- f(X,Y)
- f(P,P)

- f(foo,L)
- f(A1,A1)

- *X cannot bound to both jane and jim*

Unification

- likes(jane,X)
- likes(X,jim)

- *X cannot bound to both jane and jim*

- f(X,Y)
- f(P,P)

- *X = P and Y = P*
- *X = Y*

- f(foo,L)
- f(A1,A1)

Unification

- likes(jane,X)
- likes(X,jim)

- *X cannot bound to both jane and jim*

- f(X,Y)
- f(P,P)

- *X = P and Y = P*
- *X = Y*

- f(foo,L)
- f(A1,A1)

- *A1 = foo and A1 = L*
- *Hence, L = foo*

Rules?

- a :- b, c, d:- e f.
- happy(X):- a , b.
- happy(X):- hasmoney(X) & has_friends(X).
- fun(fish):- blue(betty), bike(yamaha).

Rules?

- a :- b, c, d:- e, f. **No**
- happy(X):- a , b. **Yes**
- happy(X):- hasmoney(X) & has_friends(X). **No**
- fun(fish):- blue(betty), bike(yamaha). **Yes**

Queries

likes(john,mary).
likes(john,trains).
likes(peter,fast_cars).

likes(Person1,Person2):-
hobby(Person1,Hobby),
hobby(Person2,Hobby).

hobby(john,trainspotting).
hobby(tim,sailing).
hobby(helen,trainspotting).
hobby(simon,sailing)

- likes(john,trains).
- likes(helen,john).
- likes(tim,helen).
- likes(john,helen).

Queries

```
likes(john,mary).
likes(john,trains).
likes(peter,fast_cars).

likes(Person1,Person2):-
    hobby(Person1,Hobby),
    hobby(Person2,Hobby).

hobby(john,trainspotting).
hobby(tim,sailing).
hobby(helen,trainspotting).
hobby(simon,sailing)
```

- likes(john,trains). **Yes**
- likes(helen,john). **Yes**
- likes(tim,helen). **No**
- likes(john,helen). **Yes**

Recursion

```
a(X):-
    b(X,Y),
    a(X).

go_home(no_12).
go_home(X):-
    get_next_house(X,Y),
    home(Y).

foo(X):- bar(X).

lonely(X):- no_friends(X).
no_friends(X):- totally_barmy(X).
```

```
has_flu(rebecca).
has_flu(john).
has_flu(X):- kisses(X,Y),has_flu(Y).
kisses(janet,john).

search(end).
search(X):- path(X,Y), search(Y).
```

Recursion

```
a(X):-
    b(X,Y),
    a(X).

go_home(no_12).
go_home(X):-
    get_next_house(X,Y),
    home(Y).

foo(X):- bar(X).

lonely(X):- no_friends(X).
no_friends(X):- totally_barmy(X).
```

```
has_flu(rebecca).
has_flu(john).
has_flu(X):-
    kisses(X,Y),has_flu(Y).
kisses(janet,john).

search(end).
search(X):- path(X,Y), search(Y).
```

Lists

- [a,d,z,c] and [H|T]
- [apple,pear,grape] and [A,pear|Rest]
- [a|Rest] and [a,b,c]
- [a,[]] and [A,B|Rest]

Lists

- [a,d,z,c] and [H|T]

  H = a and *T* = [d,z,c]
- [apple,pear,grape] and [A,pear|Rest]

  A = apple and *Rest* = [grape]
- [a|Rest] and [a,b,c]

  Rest = [b,c]
- [a,[]] and [A,B|Rest]

  A = a, *B* = [] and *Rest* = []

Lists

- [One] and [two|[]]
- [one] and [Two]
- [a,b,X] and [a,b,c,d]

Lists

- [One] and [two|[]]

  One = two
- [one] and [Two]

  Two = one
- [a,b,X] and [a,b,c,d]

  No *X* can not represent the two atoms c,d