Logic Programming
Query Examples

Michael Genesereth
Computer Science Department
Stanford University

Programme

Kinship

Blocks World

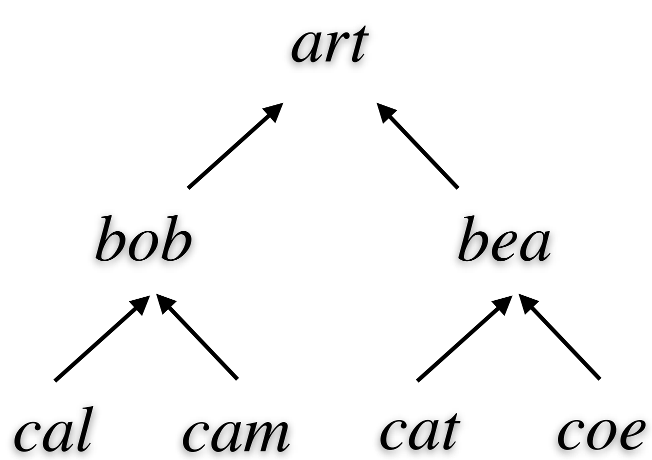
Food World

Map Coloring

Cryptarithmetic

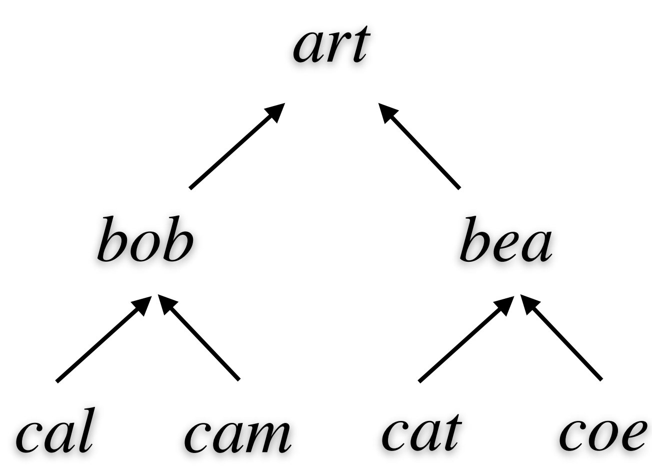
Kinship

Kinship



Dataset

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```



Grandparents

Query

```
goal(X,Z) :- parent(X,Y) & parent(Y,Z)
```

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```

Result:

```
goal(art,cal)
goal(art,cam)
goal(art,cat)
goal(art,coe)
```

People

Query:

```
goal(X) :- parent(X,Y)
goal(X) :- parent(Y,X)
```

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```

Result:

```
goal(art)
goal(bob)
goal(bea)
goal(cal)
goal(cam)
goal(cat)
goal(coe)
```

Siblings

Query

```
goal(Y,Z):-parent(X,Y) & parent(X,Z) & distinct(Y,Z)
```

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```

Result:

```
goal(bob,bea)
goal(bea,bob)
goal(cal,cam)
goal(cam,cal)
goal(cat,coe)
goal(coe,cat)
```

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)
```

Query: find every person with at least one child

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)
```

Query: find every person with at least one child

```
goal(X) :- parent(X,Y)
```

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)
```

Query: find every person with at least two children

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)
```

Query: find every person with at least two children

```
goal(X) :-
   parent(X,Y) & parent(X,Z) & distinct(Y,Z)
```

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)
```

Query: find every person with at least three children

```
goal(X) :-
   parent(X,Y) & parent(X,Z) & parent(X,W)
   mutex(Y,Z,W)
```

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)
```

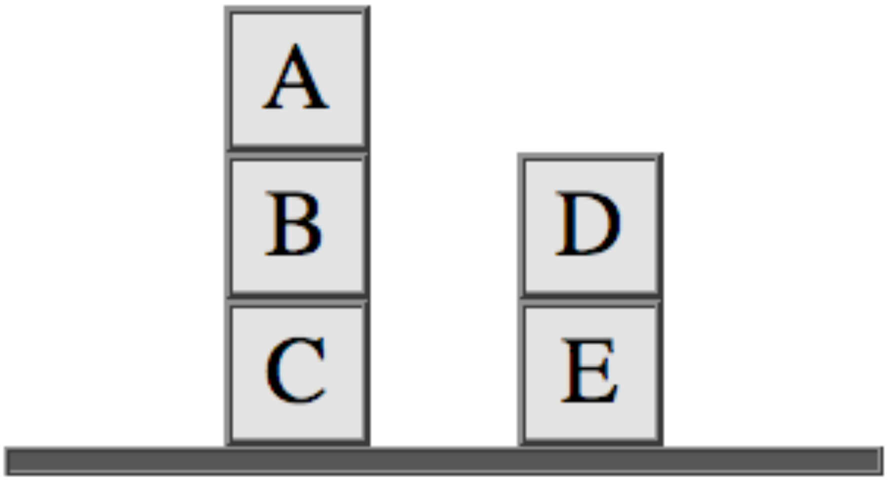
Query: find every person with *exactly* three children

```
goal(X) :-
    parent(X,Y) &
    evaluate(countofall(Z,parent(X,Z)),3)
```

See unit on views to see how to do this without `countofall`.

Blocks World

Blocks World



Vocabulary

Symbols: `a`, `b`, `c`, `d`, `e`

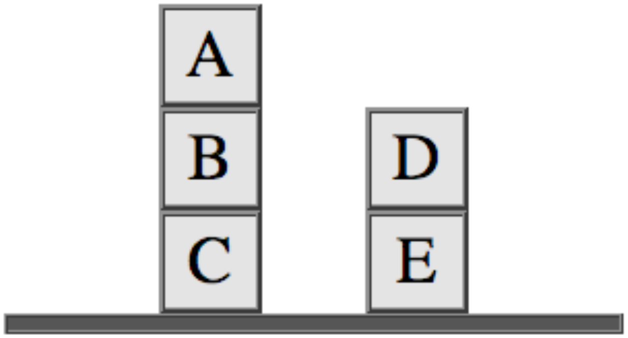
Unary Predicate:

`block`

Binary Predicate:

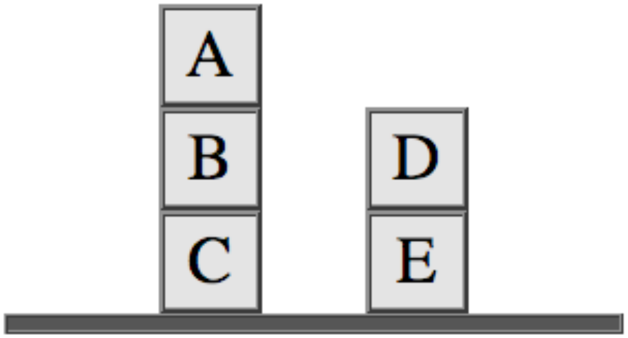
`on` - pairs of blocks in which first is on the second

Data




```
block(a)
block(b)        on(a,b)
block(c)        on(b,c)
block(d)        on(d,e)
block(e)
```

Blocks World - cluttered

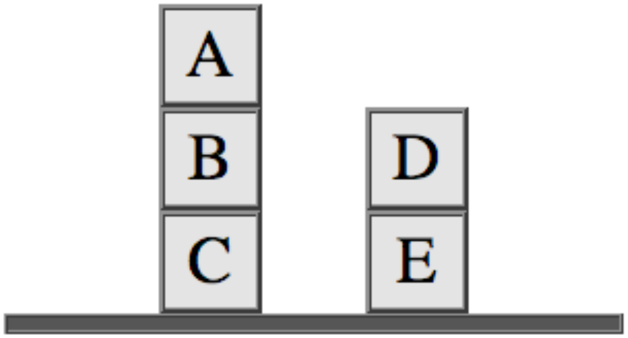



```
block(a)
block(b)      on(a,b)
block(c)      on(b,c)
block(d)      on(d,e)
block(e)
```

```
goal(Y) :- on(X,Y)
```

```
goal(b)
goal(c)
goal(e)
```

Blocks World - clear

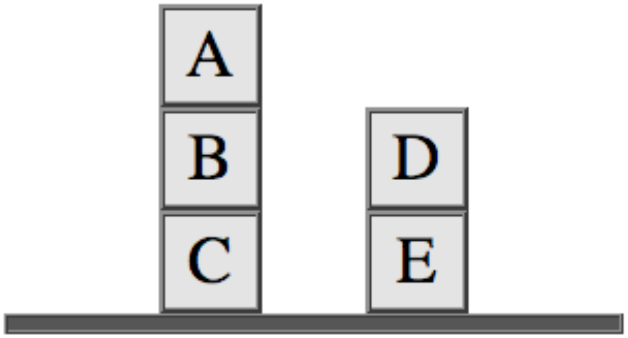



```
block(a)
block(b)      on(a,b)
block(c)      on(b,c)
block(d)      on(d,e)
block(e)
```

```
goal(Y) :-
  block(Y) &
  evaluate(countofall(X,on(X,Y)),0)
```

```
goal(a)
goal(d)
```

Blocks World - supported

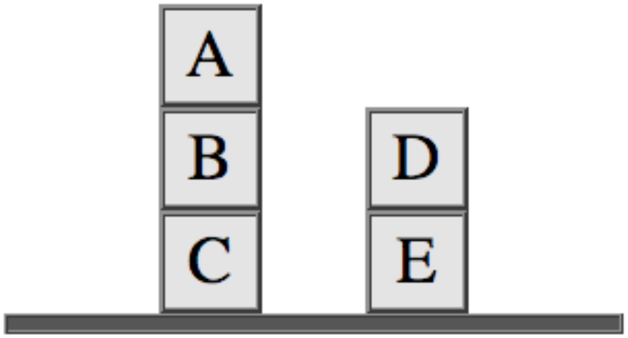



```
block(a)
block(b)      on(a,b)
block(c)      on(b,c)
block(d)      on(d,e)
block(e)
```

???

```
goal(a)
goal(b)
goal(d)
```

Blocks World - supported

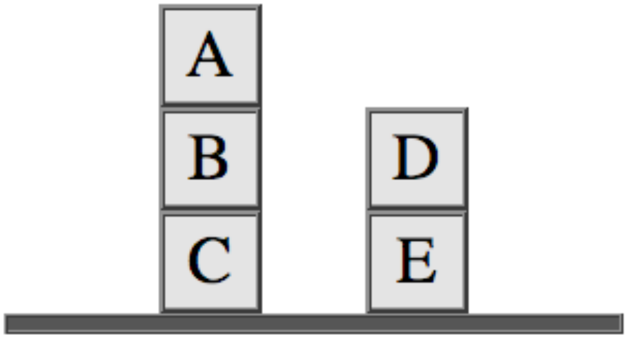



```
block(a)
block(b)      on(a,b)
block(c)      on(b,c)
block(d)      on(d,e)
block(e)
```

```
goal(X) :- on(X,Y)
```

```
goal(a)
goal(b)
goal(d)
```

Blocks World - table

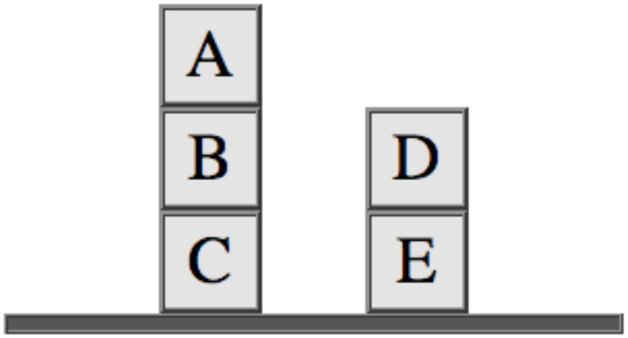



```
block(a)
block(b)      on(a,b)
block(c)      on(b,c)
block(d)      on(d,e)
block(e)
```

```
???
```

```
goal(c)
goal(e)
```

Blocks World - table

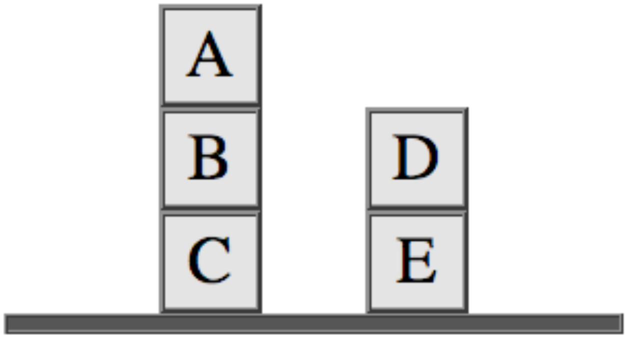



```
block(a)
block(b)        on(a,b)
block(c)        on(b,c)
block(d)        on(d,e)
block(e)
```

```
goal(X) :- block(X) & countofall(Y,on(X,Y),0)
```

```
goal(c)
goal(e)
```

Blocks World - stack

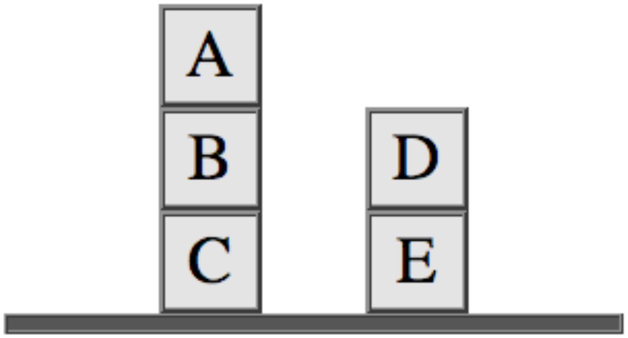



```
block(a)
block(b)        on(a,b)
block(c)        on(b,c)
block(d)        on(d,e)
block(e)
```

```
goal(X,Y,Z) :- on(X,Y) & on(Y,Z)
```

```
goal(a,b,c)
```

Blocks World - above




```
block(a)
block(b)        on(a,b)
block(c)        on(b,c)
block(d)        on(d,e)
block(e)
```

```
goal(X,Y) :- on(X,Y)
goal(X,Z) :- on(X,Y) & on(Y,Z)
goal(X,W) :- on(X,Y) & on(Y,Z) & on(Z,W)
...
```

```
goal(a,b)
goal(b,c)
goal(a,c)
goal(d,e)
```

See unit on views to see how to do this using just two rules.

Food World


YUM!
tea
Eat
FOOD
café
bakery
coffee
kitchen
cucina

Vocabulary

Symbols:

`calamari`, `greek`, `caesar`, `green`
`puree`, `consomme`, `vichyssoise`
`beef`, `lamb`, `chicken`, `trout`
`baklava`, `icecream`, `shortcake`, `souffle`, `tiramisu`
`mon`, `tue`, `wed`, `thu`, `fri`

Constructors:

`three/3`, `four/4`, `five/5` - meals of different sizes

Predicates:

`food/1` - type predicate
`day/1` - type predicate
`menu/2` - day and meal

Menu

Dataset

```
menu(mon,three(calamari,beef,shortcake))
menu(mon,three(puree,beef,icecream))
menu(tue,three(puree,beef,icecream))
menu(tue,four(consomme,greek,lamb,baklava))
menu(wed,four(consomme,greek,lamb,baklava))
menu(thu,five(vichyssoise,caesar,trout,chicken,tiramisu))
menu(fri,five(vichyssoise,green,trout,beef,souffle))
```

Meals Served on Monday

Dataset

```
menu(mon,three(calamari,beef,shortcake))
menu(mon,three(puree,beef,icecream))
menu(tue,three(puree,beef,icecream))
menu(tue,four(consomme,greek,lamb,baklava))
menu(wed,four(consomme,greek,lamb,baklava))
menu(thu,five(vichyssoise,caesar,trout,chicken,tiramisu))
menu(fri,five(vichyssoise,green,trout,beef,souffle))
```

Query

```
goal(M) :- menu(mon,M)
```

Result

```
goal(three(calamari,beef,shortcake))
goal(three(puree,beef,icecream))
```

Days with Three Course Meals

Dataset

```
menu(mon,three(calamari,beef,shortcake))
menu(mon,three(puree,beef,icecream))
menu(tue,three(puree,beef,icecream))
menu(tue,four(consomme,greek,lamb,baklava))
menu(wed,four(consomme,greek,lamb,baklava))
menu(thu,five(vichyssoise,caesar,trout,chicken,tiramisu))
menu(fri,five(vichyssoise,green,trout,beef,souffle))
```

Query

```
goal(D) :- menu(D,three(X,Y,Z))
```

Result

```
goal(mon)
goal(tue)
```

Dietary Versions of Five Course Meals

Dataset

```
menu(mon,three(calamari,beef,shortcake))
menu(mon,three(puree,beef,icecream))
menu(tue,three(puree,beef,icecream))
menu(tue,four(consomme,greek,lamb,baklava))
menu(wed,four(consomme,greek,lamb,baklava))
menu(thu,five(vichyssoise,caesar,trout,chicken,tiramisu))
menu(fri,five(vichyssoise,green,trout,beef,souffle))
```

Query

```
goal(three(V,Y,Z)) :- menu(D,five(U,V,X,Y,Z))
```

Result

```
goal(three(caesar,chicken,tiramisu))
goal(three(green,beef,souffle))
```

Days When Beef Is Served

Dataset

```
menu(mon,three(calamari,beef,shortcake))
menu(mon,three(puree,beef,icecream))
menu(tue,three(puree,beef,icecream))
menu(tue,four(consomme,greek,lamb,baklava))
menu(wed,four(consomme,greek,lamb,baklava))
menu(thu,five(vichyssoise,caesar,trout,chicken,tiramisu))
menu(fri,five(vichyssoise,green,trout,beef,souffle))
```

Query

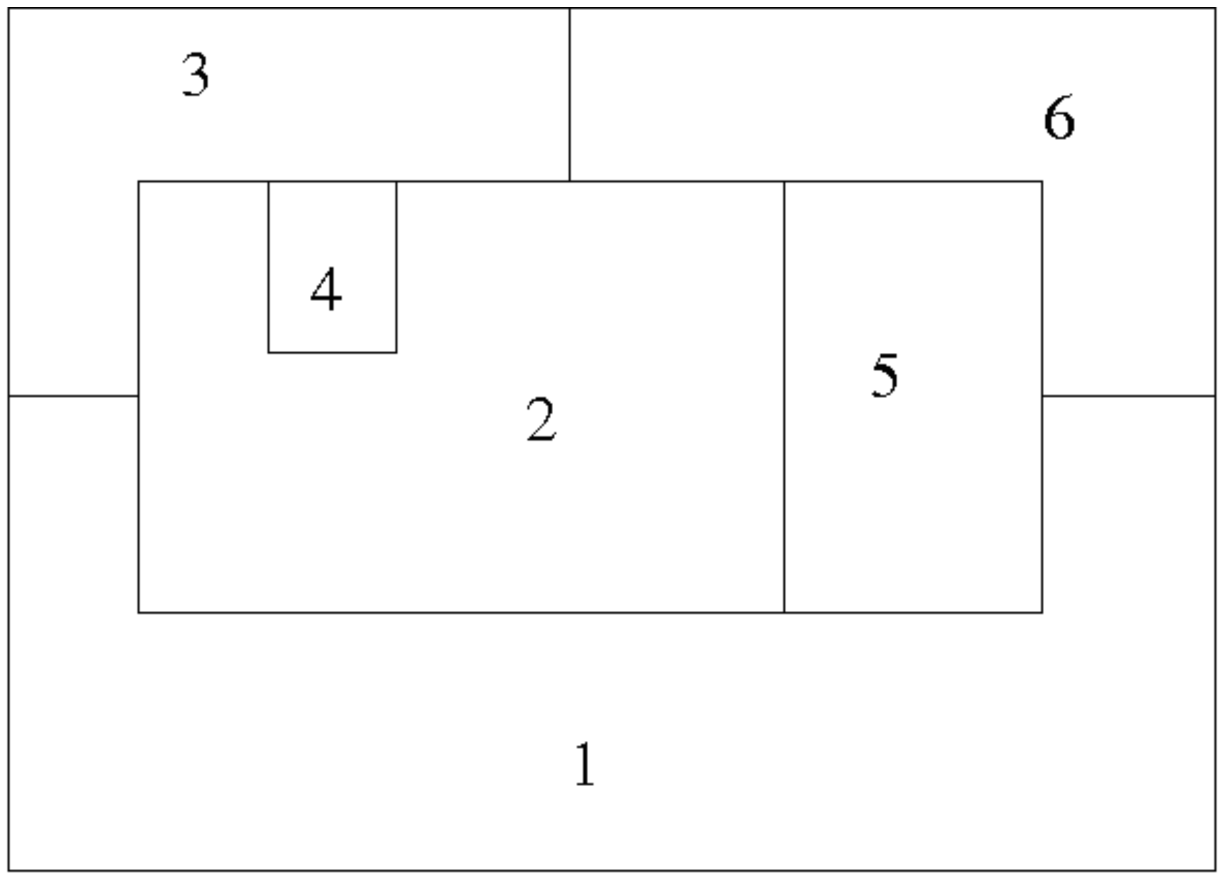
```
goal(D) :- menu(D,three(X,beef,Z))
goal(D) :- menu(D,four(X,Y,beef,Z))
goal(D) :- menu(D,five(X,Y,Z,beef,W))
```

Result

```
goal(mon)
goal(tue)
goal(fri)
```

Map Coloring

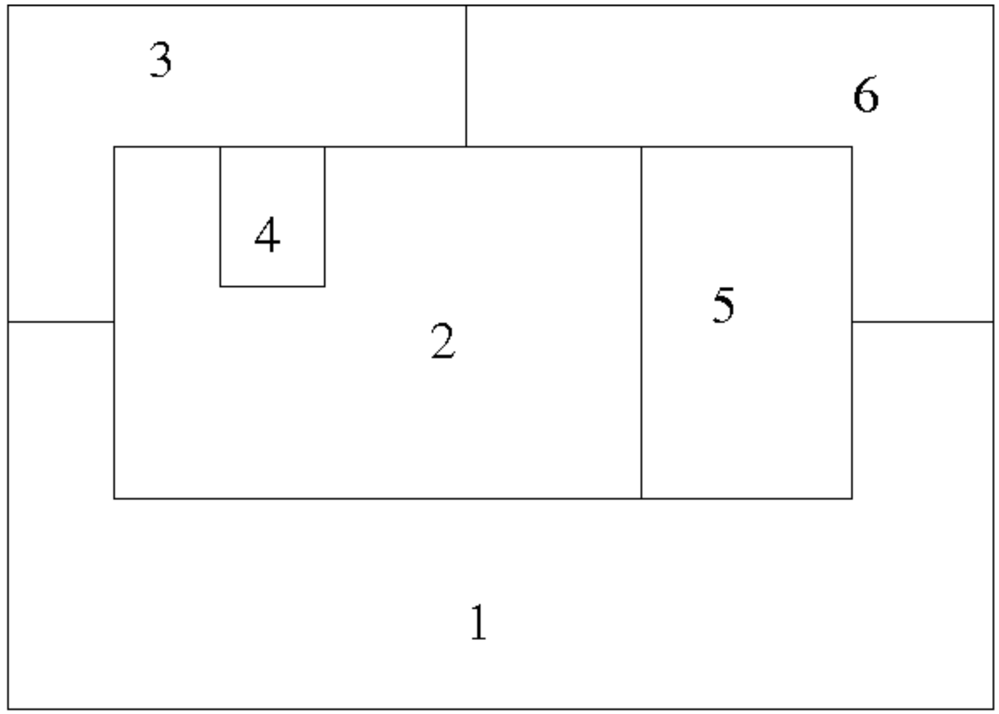
Map



Dataset

```
hue(red)
hue(green)
hue(blue)
hue(purple)
```

Query




```
goal(C1,C2,C3,C4,C5,C6) :-
  hue(C1) & hue(C2) & hue(C3) & hue(C4) & hue(C5) & hue(C6) &
  distinct(C1,C2) & distinct(C1,C3) & distinct(C1,C5) &
  distinct(C1,C6) & distinct(C2,C3) & distinct(C2,C4) &
  distinct(C2,C5) & distinct(C2,C6) & distinct(C3,C4) &
  distinct(C3,C6) & distinct(C5,C6)
```

Example

Example

```
 SEND
+MORE
-----
MONEY
```

One Solution

Data

```
digit(1)      digit(6)
digit(2)      digit(7)
digit(3)      digit(8)
digit(4)      digit(9)
digit(5)      digit(0)
```

Query

```
goal(S,E,N,D,M,O,R,Y) :-
  digit(S) & digit(E) & digit(N) & digit(D) &
  digit(M) & digit(O) & digit(R) & digit(Y) &
  mutex(S,E,N,D,M,O,R,Y) &
  distinct(S,0) & distinct(M,0) &
  evaluate(S*1000+E*100+N*10+D,U) &
  evaluate(M*1000+O*100+R*10+E,V) &
  evaluate(M*10000+O*1000+N*100+E*10+Y,W) &
  evaluate(plus(U,V),W)
```

Computational Analysis

Data

```
digit(1)      digit(6)
digit(2)      digit(7)
digit(3)      digit(8)
digit(4)      digit(9)
digit(5)      digit(0)
```

Query

```
goal(S,E,N,D,M,O,R,Y) :-
  digit(S) & digit(E) & digit(N) & digit(D) &
  digit(M) & digit(O) & digit(R) & digit(Y) & ...
```

Analysis

10x10x10x10x10x10x10x10 = 10^8 = 100,000,000 instances

Running time ~ minutes

See next week to see how to do in less than a second.