

Constraint Modeling with Albert Einstein

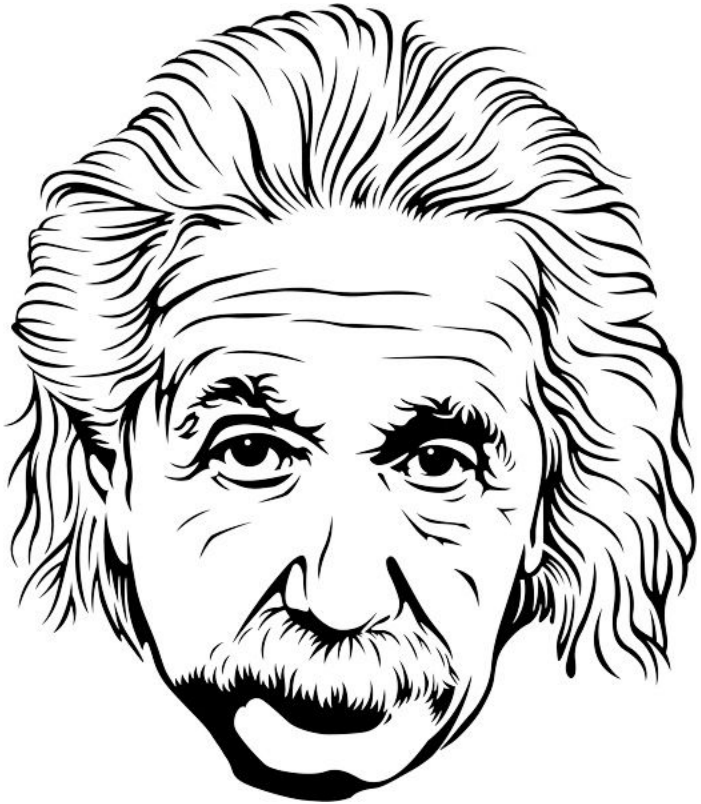
Einstein's Riddle in xtUML

Cortland Starrett - One Fact Inc



Outline

- Einstein's Riddle
- History
- Math
- Domain Model
- Discovery
- Future



Einstein's Riddle



There are 5 houses in five different colors.

In each house lives a person with a different nationality.

These five owners drink a certain type of beverage, smoke a certain brand of cigar and keep a certain pet.

No owners have the same pet, smoke the same brand of cigar or drink the same beverage.

...Who owns the fish?

Challenge

Only 2% of the population can solve the puzzle.



Hints



- The Brit lives in the red house.
- The Swede keeps dogs as pets.
- The Dane drinks tea.
- The green house is on the immediate left of the white house.
- The green house's owner drinks coffee.
- The owner who smokes Pall Mall rears birds.
- The owner of the yellow house smokes Dunhill.
- The owner living in the center house drinks milk.

...more hints



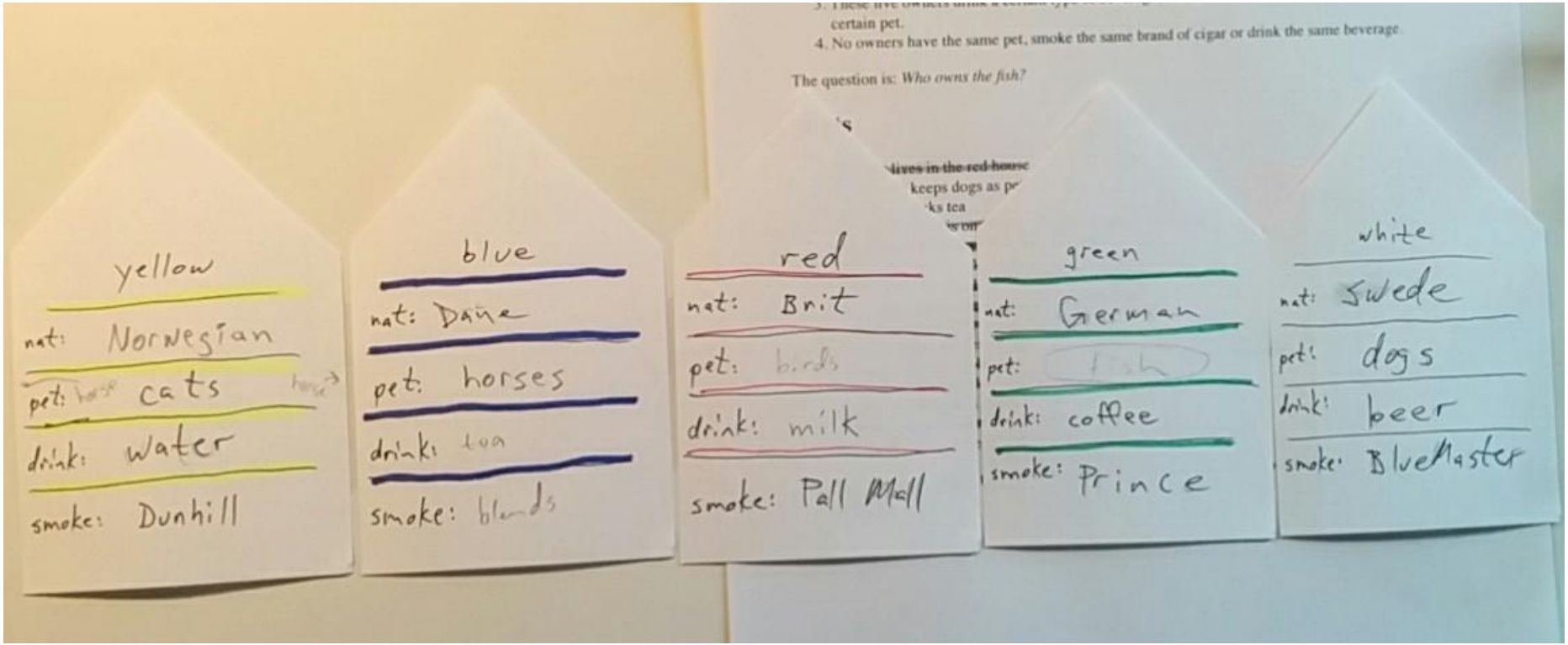
- The Norwegian lives in the first house.
- The owner who smokes Blends lives next to the one who keeps cats.
- The owner who keeps the horse lives next to the one who smokes Dunhill.
- The owner who smokes Bluemasters drinks beer.
- The German smokes Prince.
- The Norwegian lives next to the blue house.
- The owner who smokes Blends lives next to the one who drinks water.

All 15 Hints



- The Brit lives in the red house.
- The Swede keeps dogs as pets.
- The Dane drinks tea.
- The green house is on the immediate left of the white house.
- The green house's owner drinks coffee.
- The owner who smokes Pall Mall rears birds.
- The owner of the yellow house smokes Dunhill.
- The owner living in the center house drinks milk.
- The Norwegian lives in the first house.
- The owner who smokes Blends lives next to the one who keeps cats.
- The owner who keeps the horse lives next to the one who smokes Dunhill.
- The owner who smokes Bluemasters drinks beer.
- The German smokes Prince.
- The Norwegian lives next to the blue house.
- The owner who smokes Blends lives next to the one who drinks water.

First Attempt



Attempt by Someone Else

History of the 'Zebra Puzzle'



- The Zebra puzzle is often called Einstein's Riddle because it is said to have been invented by Albert Einstein as a boy.
- The Zebra puzzle has been used as a benchmark in the evaluation of computer algorithms for solving constraint satisfaction problems.
- Published in *Life International* magazine on December 1962. The March 1963 issue of *Life* contained the solution.



WIKIPEDIA
The Free Encyclopedia

Math



- CSP (Constraint Satisfaction Problem)
- permutation
  - In mathematics, the notion of permutation relates to the act of arranging all the members of a set into some sequence or order.
  - The number of permutations of n distinct objects is n factorial, n!.
- example: 5 owners of 5 pets, 5x4x3x2x1 = 5! = 120
- Permute the 'pairings'... which maps to a 1:1 association link.
- (5!)^5 = 24,883,200,000 ~ 25 billion permutations

xtUML Domain Model



- class diagram
  - "class blitz"
  - 'address' is its own class so that its links can be permuted
  - populate the instances (which remain fixed)

xtUML Domain Model



- class diagram
  - "class blitz"
  - 'address' is its own class so that its links can be permuted
  - populate the instances (which remain fixed)
- mapping to association links
  - links are constraints and variant; instances remain fixed
  - check assertions on link traversal

xtUML Domain Model



- class diagram
  - "class blitz"
  - 'address' is its own class so that its links can be permuted
  - populate the instances (which remain fixed)
- mapping to association links
  - links are constraints and variant; instances remain fixed
  - check assertions on link traversal
- exhaustive search

xtUML Domain Model

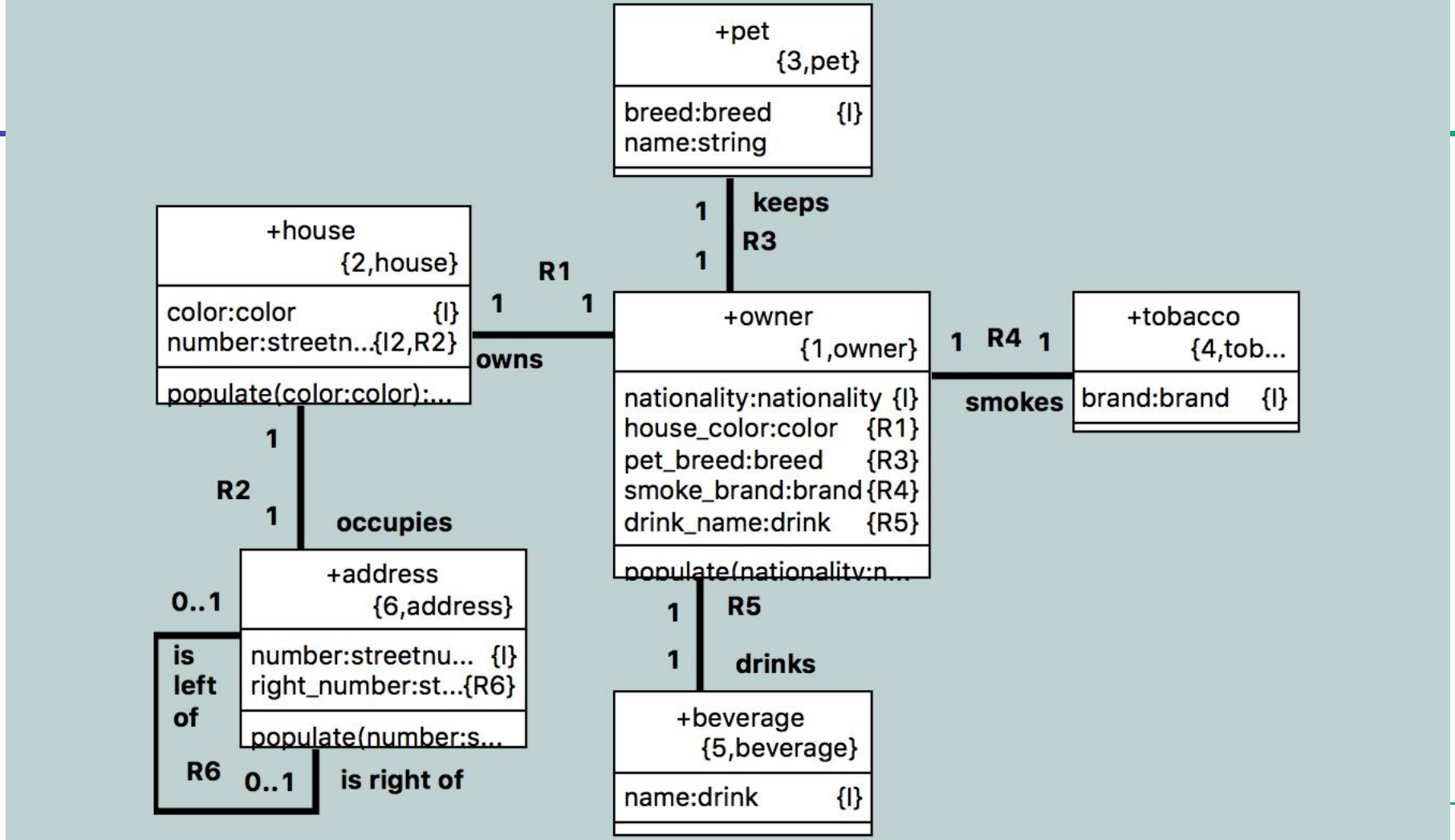


- class diagram
  - "class blitz"
  - 'address' is its own class so that its links can be permuted
  - populate the instances (which remain fixed)
- mapping to association links
  - links are constraints and variant; instances remain fixed
  - check assertions on link traversal
- exhaustive search
- interpretive execution

xtUML Domain Model



- class diagram
  - "class blitz"
  - 'address' is its own class so that its links can be permuted
  - populate the instances (which remain fixed)
- mapping to association links
  - links are constraints and variant; instances remain fixed
  - check assertions on link traversal
- exhaustive search
- interpretive execution
- compiled execution


+pet
{3,pet}
breed:breed {I}
name:string
1
keeps
R3
1
+house
{2,house}
color:color {I}
number:streetn...{I2,R2}
populate(color:color):...
R1
1
1
owns
+owner
{1,owner}
nationality:nationality {I}
house_color:color {R1}
pet_breed:breed {R3}
smoke_brand:brand {R4}
drink_name:drink {R5}
populate(nationality:n...
1
R4
1
smokes
+tobacco
{4,tob...
brand:brand {I}
1
R2
1
occupies
0..1
+address
{6,address}
number:streetnu... {I}
right_number:st...{R6}
populate(number:s...
is left of
R6
0..1
is right of
1
R5
1
drinks
+beverage
{5,beverage}
name:drink {I}

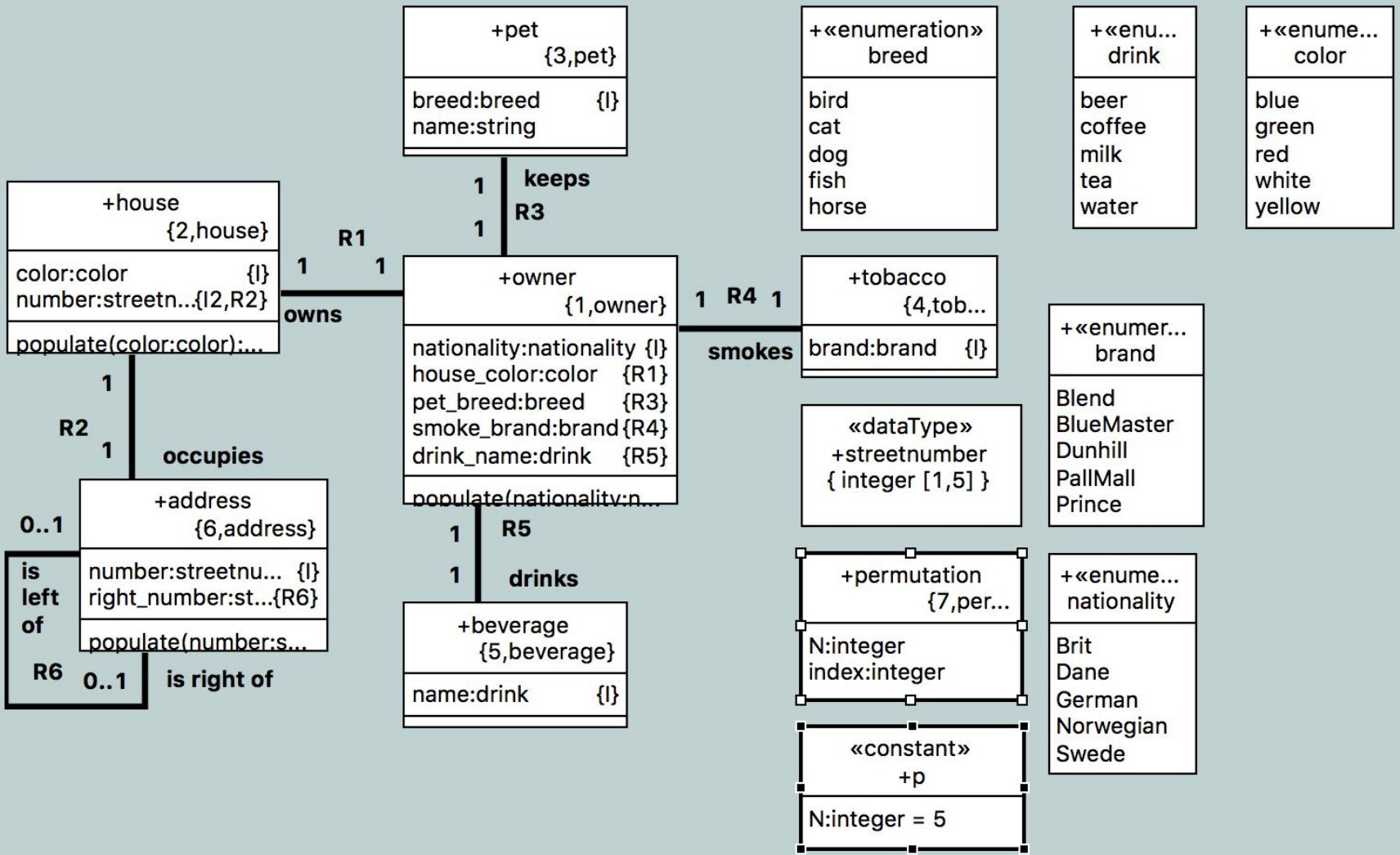
Creation



```
// owners
brit = owner::populate( nationality:nationality::Brit );
dane = owner::populate( nationality:nationality::Dane );
german = owner::populate( nationality:nationality::German );
norwegian = owner::populate( nationality:nationality::Norwegian );
swede = owner::populate( nationality:nationality::Swede );

// pets
bird = pet::populate( breed:breed::bird );
cat = pet::populate( breed:breed::cat );
dog = pet::populate( breed:breed::dog );
fish = pet::populate( breed:breed::fish );
horse = pet::populate( breed:breed::horse );

// drinks
beer = beverage::populate( name:drink::beer );
```


+pet
{3,pet}
breed:breed {I}
name:string
+«enumeration»
breed
bird
cat
dog
fish
horse
+«enu...
drink
beer
coffee
milk
tea
water
+«enume...
color
blue
green
red
white
yellow
1
keeps
R3
1
+house
{2,house}
color:color {I}
number:streetn...{I2,R2}
populate(color:color):...
R1
1
1
owns
+owner
{1,owner}
nationality:nationality {I}
house_color:color {R1}
pet_breed:breed {R3}
smoke_brand:brand {R4}
drink_name:drink {R5}
populate(nationality:n...
1
R4
1
smokes
+tobacco
{4,tob...
brand:brand {I}
+«enumer...
brand
Blend
BlueMaster
Dunhill
PallMall
Prince
1
R2
1
occupies
«dataType»
+streetnumber
{ integer [1,5] }
0..1
+address
{6,address}
number:streetnu... {I}
right_number:st...{R6}
populate(number:s...
is left of
R6
0..1
is right of
1
R5
1
drinks
+beverage
{5,beverage}
name:drink {I}
+permutation
{7,per...
N:integer
index:integer
+«enume...
nationality
Brit
Dane
German
Norwegian
Swede
«constant»
+p
N:integer = 5

Constraint Assertions



```
// the Brit lives in the red house
select any owner from instances of owner where ( selected.nationality == nationality::Brit );
select one house related by owner->house[R1.'owns'] where ( selected.color == color::red );
if ( empty house )
  distance = distance + 1;
end if;

// the Swede keeps dogs as pets
select any owner from instances of owner where ( selected.nationality == nationality::Swede );
select one pet related by owner->pet[R3.'keeps'] where ( selected.breed == breed::dog );
if ( empty pet )
  distance = distance + 1;
end if;
```

Constraint Assertions



```
// the man who smokes blends lives next to the one who keeps cats
select any tobacco from instances of tobacco where ( selected.brand == brand::Blend );
select one left_pet related by tobacco->owner[R4]->house[R1.'owns']->address[R2]->address[R6.'is right of']->house[R2]->owner[R1]->pet[R3.'keeps']
  where ( selected.breed == breed::cat );
select one right_pet related by tobacco->owner[R4]->house[R1.'owns']->address[R2]->address[R6.'is left of']->house[R2]->owner[R1]->pet[R3.'keeps']
  where ( selected.breed == breed::cat );
if ( empty left_pet and empty right_pet )
  distance = distance + 1;
end if;

// the man who keeps horses lives next to the man who smokes Dunhill
select any pet from instances of pet where ( selected.breed == breed::horse );
select one left_tobacco related by pet->owner[R3]->house[R1.'owns']->address[R2]->address[R6.'is right of']->house[R2]->owner[R1]->tobacco[R4.'smokes']
  where ( selected.brand == brand::Dunhill );
select one right_tobacco related by pet->owner[R3]->house[R1.'owns']->address[R2]->address[R6.'is left of']->house[R2]->owner[R1]->tobacco[R4.'smokes']
  where ( selected.brand == brand::Dunhill );
if ( empty left_tobacco and empty right_tobacco )
  distance = distance + 1;
end if;

// the owner who smokes BlueMaster drinks beer
select any tobacco from instances of tobacco where ( selected.brand == brand::BlueMaster );
select one beverage related by tobacco->owner[R4]->beverage[R5.'drinks'] where ( selected.name == drink::beer );
if ( empty beverage )
  distance = distance + 1;
end if;
```

Constraint Assertions



```
// the green house's owner drinks coffee
select any house from instances of house where ( selected.color == color::green );
select one beverage related by house->owner[R1]->beverage[R5.'drinks'] where ( selected.name == drink::coffee );
if ( empty beverage )
  distance = distance + 1;
end if;

// the person who smokes Pall Mall rears birds
select any tobacco from instances of tobacco where ( selected.brand == brand::PallMall );
select one pet related by tobacco->owner[R4]->pet[R3.'keeps'] where ( selected.breed == breed::bird );
if ( empty pet )
  distance = distance + 1;
end if;

// the owner of the yellow house smokes Dunhill
select any house from instances of house where ( selected.color == color::yellow );
select one tobacco related by house->owner[R1]->tobacco[R4.'smokes'] where ( selected.brand == brand::Dunhill );
if ( empty tobacco )
  distance = distance + 1;
end if;
```

Discovery



- There is exactly one solution (no surprise)
  - proved by exhaustive search
  - counting solutions in entire space
  - commenting out assertions 1 by 1 (Why?)

Discovery



- There is exactly one solution (no surprise)
  - proved by exhaustive search
  - counting solutions in entire space
  - commenting out assertions 1 by 1 (Why?)
- found superfluous constraint (surprise!)
  - removed constraints and found multiple solutions except for one
  - Did the original author know?
  - Is it needed to help humans solve?

Discovery



- There is exactly one solution (no surprise)
  - proved by exhaustive search
  - counting solutions in entire space
  - commenting out assertions 1 by 1 (Why?)
- found superfluous constraint (surprise!)
  - removed constraints and found multiple solutions except for one
  - Did the original author know?
  - Is it needed to help humans solve?
- inspired student Python solver

Future



- applying and establishing constraints starting with simple and extending to complex
- read-only links
- meta-links between links (link pairings)