CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Decision Analytics

Lecture 9: Logic Inference Example

The Zebra puzzle

- There are five houses.
- The Englishman lives in the red house.
- The Spaniard owns the dog.
- Coffee is drunk in the green house.
- The Ukrainian drinks tea.
- The green house is immediately to the right of the ivory house.
- The Old Gold smoker owns snails.
- Kools are smoked in the yellow house.
- Milk is drunk in the middle house.
- The Norwegian lives in the first house.
- The man who smokes Chesterfields lives in the house next to the man with the fox.
- Kools are smoked in the house next to the house where the horse is kept.
- The Lucky Strike smoker drinks orange juice.
- The Japanese smokes Parliaments.
- The Norwegian lives next to the blue house.
- **Now, who drinks water? Who owns the zebra?**

The Zebra puzzle

- First, we identify the predicates of the problem domain
  - $colour$
  - $nationality$
  - pet
  - $drink$
  - $cigarette$

- All of these operate on the same object domain (houses)
- The first clue reads “There are five houses.”, therefore
  - $D_O = \{House\ \#1, House\ \#2, House\ \#3, House\ \#4, House\ \#5\}$

```
houses = ["House #1", "House #2", "House #3", "House #4", "House #5"]
```

The Zebra puzzle

- The attribute domain for *colour* is

$$D_A = \{red, green, ivory, yellow, blue\}$$

```
colours = ["red", "green", "ivory", "yellow", "blue" ]
```

- We add the 5x5 Boolean variables corresponding to all combinations of objects and attributes

```
house_colour = {}
for house in houses:
    variables = {}
    for colour in colours:
        variables[colour] = model.NewBoolVar(house+colour)
    house_colour[house] = variables
```

The Zebra puzzle

- The attribute domain for $nationality$ is
  $$D_A = \{English, Spanish, Ukranian, Norwegian, Japanese\}$$

```
nationalities = ["English",
                 "Spanish",
                 "Ukrainian",
                 "Norwegian",
                 "Japanese"]
```

- Again, we add the 5x5 Boolean variables corresponding to all combinations of objects and attributes

```
house_nationality = {}
for house in houses:
    variables = {}
    for nationality in nationalities:
        variables[nationality] = model.NewBoolVar(house+nationality)
    house_nationality[house] = variables
```

The Zebra puzzle

- The attribute domain for pet is
$$D_A = \{dog, snails, fox, horse, zebra\}$$

```
pets = ["dog", "snails", "fox", "horse", "zebra"]
```

- We add the 5x5 Boolean variables corresponding to all combinations of objects and attributes

```
house_pet = {}
for house in houses:
    variables = {}
    for pet in pets:
        variables[pet] = model.NewBoolVar(house+pet)
    house_pet[house] = variables
```

The Zebra puzzle

- The attribute domain for $drink$ is

$$D_A = \{coffee, tea, milk, juice, water\}$$

```
drinks = ["coffee", "tea", "milk", "juice", "water"]
```

- We add the 5x5 Boolean variables corresponding to all combinations of objects and attributes

```
house_drink = {}
for house in houses:
    variables = {}
    for drink in drinks:
        variables[drink] = model.NewBoolVar(house+drink)
    house_drink[house] = variables
```

The Zebra puzzle

- The attribute domain for $cigarette$ is $D_A = \{OldGold, Chesterfields, Kools, LuckeyStrike, Parliaments\}$

```
cigarettes = ["Old Gold",
              "Chesterfields",
              "Kools",
              "Lucky Strike",
              "Parliaments"]
```

- Again, we add the 5x5 Boolean variables corresponding to all combinations of objects and attributes

```
house_cigarette = {}
for house in houses:
    variables = {}
    for cigarette in cigarettes:
        variables[cigarette] = model.NewBoolVar(house+cigarette)
    house_cigarette[house] = variables
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- “The Englishman lives in the red house.”

$$\forall x{:}\, nationality(x, English) \Rightarrow colour(x, red)$$

- We translate this into a conjunction over all houses

$$nationality(House\ \#1, English) \Rightarrow colour(House\ \#1, red)$$
$$nationality(House\ \#2, English) \Rightarrow colour(House\ \#2, red)$$
$$nationality(House\ \#3, English) \Rightarrow colour(House\ \#3, red)$$
$$nationality(House\ \#4, English) \Rightarrow colour(House\ \#4, red)$$
$$nationality(House\ \#5, English) \Rightarrow colour(House\ \#5, red)$$

```
for house in houses:
    model.AddBoolAnd([house_colour[house]["red"]]).OnlyEnforceIf(house_nationality[house]["English"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "The Spaniard owns the dog."

$$\forall x{:}\, nationality(x, Spanish) \Rightarrow pet(x, dog)$$

- We translate this into a conjunction over all houses

$$nationality(House\ \#1, Spanish) \Rightarrow pet(House\ \#1, dog)$$
$$nationality(House\ \#2, Spanish) \Rightarrow pet(House\ \#2, dog)$$
$$nationality(House\ \#3, Spanish) \Rightarrow pet(House\ \#3, dog)$$
$$nationality(House\ \#4, Spanish) \Rightarrow pet(House\ \#4, dog)$$
$$nationality(House\ \#5, Spanish) \Rightarrow pet(House\ \#5, dog)$$

```
for house in houses:
    model.AddBoolAnd([house_pet[house]["dog"]]).OnlyEnforceIf(house_nationality[house]["Spanish"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "Coffee is drunk in the green house."

$$\forall x: colour(x, green) \Rightarrow drink(x, coffee)$$

- We translate this into a conjunction over all houses

$$colour(House\ \#1, green) \Rightarrow drink(House\ \#1, coffee)$$
$$colour(House\ \#2, green) \Rightarrow drink(House\ \#2, coffee)$$
$$colour(House\ \#3, green) \Rightarrow drink(House\ \#3, coffee)$$
$$colour(House\ \#4, green) \Rightarrow drink(House\ \#4, coffee)$$
$$colour(House\ \#5, green) \Rightarrow drink(House\ \#5, coffee)$$

```
for house in houses:
    model.AddBoolAnd([house_drink[house]["coffee"]]).OnlyEnforceIf(house_colour[house]["green"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "The Ukrainian drinks tea."

$$\forall x: nationality(x, Ukrainian) \Rightarrow drink(x, tea)$$

- We translate this into a conjunction over all houses

$$nationality(House\ \#1, Ukrainian) \Rightarrow drink(House\ \#1, tea)$$
$$nationality(House\ \#2, Ukrainian) \Rightarrow drink(House\ \#2, tea)$$
$$nationality(House\ \#3, Ukrainian) \Rightarrow drink(House\ \#3, tea)$$
$$nationality(House\ \#4, Ukrainian) \Rightarrow drink(House\ \#4, tea)$$
$$nationality(House\ \#5, Ukrainian) \Rightarrow drink(House\ \#5, tea)$$

```
for house in houses:
    model.AddBoolAnd([house_drink[house]["tea"]]).OnlyEnforceIf(house_nationality[house]["Ukrainian"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "The green house is immediately to the right of the ivory house."

$$colour(House\ \#1, ivory) \Rightarrow colour(House\ \#2, green)$$
$$colour(House\ \#2, ivory) \Rightarrow colour(House\ \#3, green)$$
$$colour(House\ \#3, ivory) \Rightarrow colour(House\ \#4, green)$$
$$colour(House\ \#4, ivory) \Rightarrow colour(House\ \#5, green)$$
$$\neg colour(House\ \#5, ivory)$$
$$\neg colour(House\ \#1, green)$$

```
for i in range(4):
    model.AddBoolAnd([house_colour[houses[i+1]]["green"]]).OnlyEnforceIf(house_colour[houses[i]]["ivory"])
model.AddBoolAnd([house_colour[houses[4]]["ivory"].Not()])
model.AddBoolAnd([house_colour[houses[0]]["green"].Not()])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- “The Old Gold smoker owns snails.”

$$\forall x: cigarette(x, Old\ Gold) \Rightarrow pet(x, snails)$$

- We translate this into a conjunction over all houses

$$cigarette(House\ \#1, Old\ Gold) \Rightarrow pet(House\ \#1, snails)$$
$$cigarette(House\ \#2, Old\ Gold) \Rightarrow pet(House\ \#2, snails)$$
$$cigarette(House\ \#3, Old\ Gold) \Rightarrow pet(House\ \#3, snails)$$
$$cigarette(House\ \#4, Old\ Gold) \Rightarrow pet(House\ \#4, snails)$$
$$cigarette(House\ \#5, Old\ Gold) \Rightarrow pet(House\ \#5, snails)$$

```
for house in houses:
    model.AddBoolAnd([house_pet[house]["snails"]]).OnlyEnforceIf(house_cigarette[house]["Old Gold"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "Kools are smoked in the yellow house."

$$\forall x: cigarette(x, Kools) \Rightarrow colour(x, yellow)$$

- We translate this into a conjunction over all houses

$$cigarette(House\ \#1, Kools) \Rightarrow colour(House\ \#1, yellow)$$
$$cigarette(House\ \#2, Kools) \Rightarrow colour(House\ \#2, yellow)$$
$$cigarette(House\ \#3, Kools) \Rightarrow colour(House\ \#3, yellow)$$
$$cigarette(House\ \#4, Kools) \Rightarrow colour(House\ \#4, yellow)$$
$$cigarette(House\ \#5, Kools) \Rightarrow colour(House\ \#5, yellow)$$

```
for house in houses:
    model.AddBoolAnd([house_colour[house]["yellow"]]).OnlyEnforceIf(house_cigarette[house]["Kools"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- “Milk is drunk in the middle house.”

$$drink(House\ \#3, milk)$$

```
model.AddBoolAnd([house_drink["House #3"]["milk"]])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- “The Norwegian lives in the first house.”

$$nationality(House\ \#1, Norwegian)$$

```
model.AddBoolAnd([house_nationality["House #1"]["Norwegian"]])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "The man who smokes Chesterfields lives in the house next to the man with the fox."

$$cigarette[House\ \#1, \text{Chesterfields}] \Rightarrow pet(House\ \#2, fox)$$
$$cigarette[House\ \#2, \text{Chesterfields}] \Rightarrow pet(House\ \#1, \text{fox}) \vee pet(House\ \#3, fox)$$
$$cigarette[House\ \#3, \text{Chesterfields}] \Rightarrow pet(House\ \#2, \text{fox}) \vee pet(House\ \#4, fox)$$
$$cigarette[House\ \#4, \text{Chesterfields}] \Rightarrow pet(House\ \#3, \text{fox}) \vee pet(House\ \#5\ fox)$$
$$cigarette[House\ \#5, \text{Chesterfields}] \Rightarrow pet(House\ \#4, fox)$$

```
for i in range(1,4):
    model.AddBoolOr([
            house_pet[houses[i+1]]["fox"],
            house_pet[houses[i-1]]["fox"]]).OnlyEnforceIf(house_cigarette[houses[i]]["Chesterfields"])
model.AddBoolOr([house_pet["House #2"]["fox"]]).OnlyEnforceIf(house_cigarette["House #1"]["Chesterfields"])
model.AddBoolOr([house_pet["House #4"]["fox"]]).OnlyEnforceIf(house_cigarette["House #5"]["Chesterfields"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- “Kools are smoked in the house next to the house where the horse is kept.”

$$cigarette[House\ \#1, \text{Kools}] \Rightarrow pet(House\ \#2, horse)$$
$$cigarette[House\ \#2, \text{Kools}] \Rightarrow pet(House\ \#1, \text{horse}) \vee pet(House\ \#3, horse)$$
$$cigarette[House\ \#3, \text{Kools}] \Rightarrow pet(House\ \#2, \text{horse}) \vee pet(House\ \#4, horse)$$
$$cigarette[House\ \#4, \text{Kools}] \Rightarrow pet(House\ \#3, \text{horse}) \vee pet(House\ \#5\ horse)$$
$$cigarette[House\ \#5, \text{Kools}] \Rightarrow pet(House\ \#4, horse)$$

```
for i in range(1,4):
    model.AddBoolOr([
               house_pet[houses[i+1]][“horse"],
               house_pet[houses[i-1]]["horse"]]).OnlyEnforceIf(house_cigarette[houses[i]][“Kools"])
model.AddBoolOr([house_pet["House #2"]["horse"]]).OnlyEnforceIf(house_cigarette["House #1"][“Kools"])
model.AddBoolOr([house_pet["House #4"]["horse"]]).OnlyEnforceIf(house_cigarette["House #5"][“Kools"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "The Lucky Strike smoker drinks orange juice."

$$\forall x: cigarette(x, LuckyStrike) \Rightarrow drink(x, juice)$$

- We translate this into a conjunction over all houses

$$cigarette(House\ \#1, LuckyStrike) \Rightarrow drink(House\ \#1, juice)$$
$$cigarette(House\ \#2, LuckyStrike) \Rightarrow drink(House\ \#2, juice)$$
$$cigarette(House\ \#3, LuckyStrike) \Rightarrow drink(House\ \#3, juice)$$
$$cigarette(House\ \#4, LuckyStrike) \Rightarrow drink(House\ \#4, juice)$$
$$cigarette(House\ \#5, LuckyStrike) \Rightarrow drink(House\ \#5, juice)$$

```
for house in houses:
  model.AddBoolAnd([house_drink[house]["juice"]]).OnlyEnforceIf(house_cigarette[house]["Lucky Strike"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- "The Japanese smokes Parliaments."

$$\forall x\colon nationality(x, Japanese) \Rightarrow cigarette(x, Parliaments)$$

- We translate this into a conjunction over all houses

$$nationality(House\ \#1, Japanese) \Rightarrow cigarette(House\ \#1, Parliaments)$$
$$nationality(House\ \#2, Japanese) \Rightarrow cigarette(House\ \#2, Parliaments)$$
$$nationality(House\ \#3, Japanese) \Rightarrow cigarette(House\ \#3, Parliaments)$$
$$nationality(House\ \#4, Japanese) \Rightarrow cigarette(House\ \#4, Parliaments)$$
$$nationality(House\ \#5, Japanese) \Rightarrow cigarette(House\ \#5, Parliaments)$$

```
for house in houses:
    model.AddBoolAnd([house_cigarette[house]["Parliaments"]]).OnlyEnforceIf(house_nationality[house]["Japanese"])
```

The Zebra puzzle

- Now we go sentence by sentence and re-formulate in first-order logic
- “The Norwegian lives next to the blue house.

$$colour[House\ \#1, \text{blue}] \Rightarrow nationality(House\ \#2, Norwegian)$$
$$colour[House\ \#2, blue] \Rightarrow nationality(House\ \#1, \text{Norwegian}) \vee nationality(House\ \#3, Norwegian)$$
$$colour[House\ \#3, blue] \Rightarrow nationality(House\ \#2, \text{Norwegian}) \vee nationality(House\ \#4\ Norwegian)$$
$$colour[House\ \#4, blue] \Rightarrow nationality(House\ \#3, \text{Norwegian}) \vee nationality(House\ \#4, Norwegian)$$
$$colour[House\ \#5, \text{blue}] \Rightarrow nationality(House\ \#4, Norwegian)$$

```
for i in range(1,4):
    model.AddBoolOr([
                 house_nationality[houses[i+1]]["Norwegian"],
                 house_nationality[houses[i-1]]["Norwegian"]]).OnlyEnforceIf(house_colour[houses[i]]["blue"]
model.AddBoolOr([house_nationality[houses[1]]["Norwegian"]]).OnlyEnforceIf(house_colour[houses[0]]["blue"])
model.AddBoolOr([house_nationality[houses[3]]["Norwegian"]]).OnlyEnforceIf(house_colour[houses[4]]["blue"])
```

The Zebra puzzle

- That is all sentences of the zebra puzzle encoded, but is this all?

- There are some **implicit assumptions** that are not explicitly stated in the text
  - Every house has a colour/nationality/pet/drink/cigarette
  - Every house has no more than one colour/nationality/pet/drink/cigarette
  - All houses have a different colour/nationality/pet/drink/cigarette

The Zebra puzzle

- Every house has a colour

$$\forall x \exists y: colour(x, y)$$

- That translates into

$$\begin{aligned}
&(colour(House\ \#1, red) \lor colour(House\ \#1, green) \lor colour(House\ \#1, ivory) \lor colour(House\ \#1, yellow) \lor colour(House\ \#1, blue)) \\
&\land (colour(House\ \#2, red) \lor colour(House\ \#2, green) \lor colour(House\ \#2, ivory) \lor colour(House\ \#2, yellow) \lor colour(House\ \#2, blue)) \\
&\land (colour(House\ \#3, red) \lor colour(House\ \#3, green) \lor colour(House\ \#3, ivory) \lor colour(House\ \#3, yellow) \lor colour(House\ \#3, blue)) \\
&\land (colour(House\ \#4, red) \lor colour(House\ \#4, green) \lor colour(House\ \#4, ivory) \lor colour(House\ \#4, yellow) \lor colour(House\ \#4, blue)) \\
&\land (colour(House\ \#5, red) \lor colour(House\ \#5, green) \lor colour(House\ \#5, ivory) \lor colour(House\ \#5, yellow) \lor colour(House\ \#5, blue))
\end{aligned}$$

```
for house in houses:
    variables = []
    for colour in colours:
        variables.append(house_colour[house][colour])
    model.AddBoolOr(variables)
```

The Zebra puzzle

- Every house has an occupant

$$\forall x \exists y : nationality(x, y)$$

```
for house in houses:
    variables = []
    for nationality in nationalities:
        variables.append(house_nationality[house][nationality])
    model.AddBoolOr(variables)
```

The Zebra puzzle

- Every house has a pet

$$\forall x \exists y: pet(x, y)$$

```
for house in houses:
    variables = []
    for pet in pets:
        variables.append(house_pet[house][pet])
    model.AddBoolOr(variables)
```

The Zebra puzzle

- Every house has a drink

$$\forall x \exists y: drink(x, y)$$

```
for house in houses:
   variables = []
   for drink in drinks:
      variables.append(house_drink[house][drink])
   model.AddBoolOr(variables)
```

The Zebra puzzle

- Every house has a cigarette

$$\forall x \exists y: cigarette(x, y)$$

```
for house in houses:
    variables = []
    for cigarette in cigarettes:
        variables.append(house_cigarette[house][cigarette])
    model.AddBoolOr(variables)
```

The Zebra puzzle

- Every house has no more than one colour

$$\forall x \forall y \forall z: y \neq z \Rightarrow \neg(colour(x,y) \wedge colour(x,z))$$

This is equivalent to

$$\forall x \forall y \forall z: y \neq z \Rightarrow (\neg colour(x,y) \vee \neg colour(x,z))$$

```
for house in houses:
    for i in range(5):
        for j in range(i+1,5):
            model.AddBoolOr([
                house_colour[house][colours[i]].Not(),
                house_colour[house][colours[j]].Not()])
```

The Zebra puzzle

- Every house has no more than one occupant

$$\forall x \forall y \forall z: y \neq z \Rightarrow \neg(nationality(x,y) \wedge nationality(x,z))$$

This is equivalent to

$$\forall x \forall y \forall z: y \neq z \Rightarrow \ (\neg nationality(x,y) \vee \neg nationality(x,z))$$

```
for house in houses:
    for i in range(5):
        for j in range(i+1,5):
            model.AddBoolOr([
                house_nationality[house][nationalities[i]].Not(),
                house_nationality[house][nationalities[j]].Not()])
```

The Zebra puzzle

- Every house has no more than one pet

$$\forall x \forall y \forall z: y \neq z \Rightarrow \neg(pet(x,y) \wedge pet(x,z))$$

This is equivalent to

$$\forall x \forall y \forall z: y \neq z \Rightarrow (\neg pet(x,y) \vee \neg pet(x,z))$$

```
for house in houses:
    for i in range(5):
        for j in range(i+1,5):
            model.AddBoolOr([
                house_pet[house][pets[i]].Not(),
                house_pet[house][pets[j]].Not()])
```

The Zebra puzzle

- Every house has no more than one drink

$$\forall x \forall y \forall z: y \neq z \Rightarrow \neg(drink(x,y) \wedge drink(x,z))$$

This is equivalent to

$$\forall x \forall y \forall z: y \neq z \Rightarrow \ (\neg drink(x,y) \vee \neg drink(x,z))$$

```
for house in houses:
    for i in range(5):
        for j in range(i+1,5):
            model.AddBoolOr([
                house_drink[house][drinks[i]].Not(),
                house_drink[house][drinks[j]].Not()])
```

The Zebra puzzle

- Every house has no more than one cigarette

$$\forall x \forall y \forall z : y \neq z \Rightarrow \neg(cigarette(x,y) \land cigarette(x,z))$$

This is equivalent to

$$\forall x \forall y \forall z : y \neq z \Rightarrow (\neg cigarette(x,y) \lor \neg cigarette(x,z))$$

```
for house in houses:
    for i in range(5):
        for j in range(i+1,5):
            model.AddBoolOr([
                house_cigarette[house][cigarettes[i]].Not(),
                house_cigarette[house][cigarettes[j]].Not()])
```

The Zebra puzzle

- Every house has a different colour

$$\forall x \forall y \forall z: x \neq y \Rightarrow \neg(colour(x,z) \wedge colour(y,z))$$

This is equivalent to

$$\forall x \forall y \forall z: x \neq y \Rightarrow (\neg colour(x,z) \vee \neg colour(y,z))$$

```
for i in range(5):
    for j in range(i+1,5):
        for k in range(5):
            model.AddBoolOr([
                house_colour[houses[i]][colours[k]].Not(),
                house_colour[houses[j]][colours[k]].Not()])
```

The Zebra puzzle

- Every house has a different occupant

$$\forall x \forall y \forall z: x \neq y \Rightarrow \neg(nationality(x,z) \wedge nationality(y,z))$$

This is equivalent to

$$\forall x \forall y \forall z: x \neq y \Rightarrow (\neg nationality(x,z) \vee \neg nationality(y,z))$$

```
for i in range(5):
    for j in range(i+1,5):
        for k in range(5):
            model.AddBoolOr([
                    house_nationality[houses[i]][nationalities[k]].Not(),
                    house_nationality[houses[j]][nationalities[k]].Not()])
```

The Zebra puzzle

- Every house has a different pet

$$\forall x \forall y \forall z: x \neq y \Rightarrow \neg(pet(x,z) \wedge pet(y,z))$$

This is equivalent to

$$\forall x \forall y \forall z: x \neq y \Rightarrow (\neg pet(x,z) \vee \neg pet(y,z))$$

```
for i in range(5):
    for j in range(i+1,5):
        for k in range(5):
            model.AddBoolOr([
                house_pet[houses[i]][pets[k]].Not(),
                house_pet[houses[j]][pets[k]].Not()])
```

The Zebra puzzle

- Every house has a different drink

$$\forall x \forall y \forall z: x \neq y \Rightarrow \neg(drink(x,z) \wedge drink(y,z))$$

This is equivalent to

$$\forall x \forall y \forall z: x \neq y \Rightarrow (\neg drink(x,z) \vee \neg drink(y,z))$$

```
for i in range(5):
    for j in range(i+1,5):
        for k in range(5):
            model.AddBoolOr([
                house_drink[houses[i]][drinks[k]].Not(),
                house_drink[houses[j]][drinks[k]].Not()])
```

The Zebra puzzle

- Every house has a different cigarette

$$\forall x \forall y \forall z: x \neq y \Rightarrow \neg(cigarette(x,z) \wedge cigarette(y,z))$$

This is equivalent to

$$\forall x \forall y \forall z: x \neq y \Rightarrow (\neg cigarette(x,z) \vee \neg cigarette(y,z))$$

```
for i in range(5):
    for j in range(i+1,5):
        for k in range(5):
            model.AddBoolOr([
                    house_cigarette[houses[i]][cigarettes[k]].Not(),
                    house_cigarette[houses[j]][cigarettes[k]].Not()])
```

The Zebra puzzle

- That is all constraints, both explicit and implicit, encoded

- Who drinks water? The Norwegian.
- Who owns the zebra? The Japanese.

```
for house in houses:
    if solver.Value(house_drink[house]["water"]):
        for nationality in nationalities:
            if solver.Value(house_nationality[house][nationality]):
                print("The "+nationality+" drinks water.")
    if solver.Value(house_pet[house]["zebra"]):
        for nationality in nationalities:
            if solver.Value(house_nationality[house][nationality]):
                print("The "+nationality+" owns the zebra.")
```

Thank you for your attention!