Problem Formulation

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2018

Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
  - Supporting Python constructs
- Wicked Problems
  - Rittel & Webber's 10 Criteria
  - The Climate Conundrum

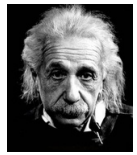
Outline

- **Motivation**
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
  - Supporting Python constructs
- Wicked Problems
  - Rittel & Webber's 10 Criteria
  - The Climate Conundrum

Motivation

"The formulation of the problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill."

--Albert Einstein



Motivation (cont.)

In AI, proper formulation supports:
- automatic solving
- computer-assisted manual solving
- problem-space analysis
- visualization
- use of heuristics
- reasoning about the problem

Outline

- Motivation
- **The 3 Phases of Problem Formulation**
- Case Study with the 8 Puzzle
  - Supporting Python constructs
- Wicked Problems
  - Rittel & Webber's 10 Criteria
  - The Climate Conundrum

Steps in Problem Formulation

- Describing a need
- Identifying resources
- Restriction and simplification
- Designing a state representation
- Designing a set of operators
- Listing constraints and desiderata
- Specifying in code the state representation, operators, constraints, evaluation criteria, and goal criterion.
- Specifying in code a state visualization method.
- If appropriate, providing for multiple roles within teams of solvers.

Steps in Problem Formulation

(Preformulation)

- Describing a need
- Identifying resources
- Restriction and simplification
- Designing a state representation
- Designing a set of operators
- Listing constraints and desiderata
- Specifying in code the state representation, operators, constraints, evaluation criteria, and goal criterion.
- Specifying in code a state visualization method.
- If appropriate, providing for multiple roles within teams of solvers.

Steps in Problem Formulation

- Describing a need
- Identifying resources

(Preformulation)

- Restriction and simplification
- Designing a state representation
- Designing a set of operators
- Listing constraints and desiderata

(Posing)

- Specifying in code the state representation, operators, constraints, evaluation criteria, and goal criterion.
- Specifying in code a state visualization method.
- If appropriate, providing for multiple roles within teams of solvers.

Steps in Problem Formulation

- Describing a need
- Identifying resources

(Preformulation)

- Restriction and simplification
- Designing a state representation
- Designing a set of operators
- Listing constraints and desiderata

(Posing)

- Specifying in code the state representation, operators, constraints, evaluation criteria, and goal criterion.
- Specifying in code a state visualization method.
- If appropriate, providing for multiple roles within teams of solvers.

(Coding the formulation)

Outline

- Motivation
- The 3 Phases of Problem Formulation
- **Case Study with the 8 Puzzle**
  - Supporting Python constructs
- Wicked Problems
  - Rittel & Webber's 10 Criteria
  - The Climate Conundrum

The Eight Puzzle

The Eight Puzzle is like the Fifteen Puzzle, but has only a 3x3 tray and eight tiles.

Fifteen Puzzle

Eight Puzzle

Why do I use puzzles?

Judea Pearl:

"The expository power of puzzles and games stems from their combined richness and simplicity. If we were to use examples taken from real-life problems, it would take more than a few pages just to lay the background..."

Definition (Revisited)

- A problem is a triple: (σ_0, Φ, Γ) where σ_0 is an initial state, Φ is a set of operators, and Γ is a set of goal states.
- Each $\varphi_i \in \Phi$ has a precondition, a state-transformation function, and an optional parameter list.
- These implicitly define Σ, the set of all states reachable from σ_0 by applying members $\varphi_i \in \Phi$ zero or more times.

Eight Puzzle Formulation

- State: 3x3 array containing 8 tiles
  - Each tile represented by a number in 1, 2, ..., 8
  - The blank represented by 0
- Initial State: A random state, except it must represent an even permutation.
- Goal State: [[0, 1, 2],[3, 4, 5],[6, 7, 8]]
- Operators: N,E,W,S
  - ("Move a tile North", etc.)

Eight Puzzle Formulation (cont).

- Note: These choices are not all forced.
- For example, a state COULD BE a string of characters, e.g., "ABCDEF_GH".
- An operator could be "Move the Void Left", or could be to interchange some pair of letters in the string representation.
- The specifications of a problem's states information content and structures and its operators are DESIGN DECISIONS.

EightPuzzle.py Session Start

```
bash-4.2$ python3 ../Int_Solv_Client.py EightPuzzle
problem_name = EightPuzzle
Using default initial state list: [[8, 7, 6], [5, 4, 3], [2, 1, 0]]
 (To use a specific initial state, enter it on the command line, e.g.,
python3 ../Int_Solv_Client.py EightPuzzle '[[3, 1, 2], [0, 4, 5], [6, 7, 8]]'
Int_Solv_Client (Version 1)
Eight Puzzle; 0.1

Step 0, Depth 0
CURRENT_STATE =
[[8, 7, 6]
 [5, 4, 3]
 [2, 1, 0]]
1: Move a tile E into the void
3: Move a tile S into the void
Enter command: 0, 1, 2, etc. for operator; B-back; H-help; Q-quit. >>
```


Coding a Formulation

In CSE 415, we are using a particular format for problem formulations. Major sections are:

- **METADATA**
- **COMMON_DATA**
- **COMMON_CODE**
  - **State class**
  - **Supporting methods**
- **OPERATORS**
- **INITIAL_STATE**
- **GOAL_TEST**
- **STATE_VIS** (not required)

EightPuzzle.py (Code Excerpt 1)

```
#<COMMON_CODE>
class State:
  def __init__(self, list_of_lists):
    self.b = list_of_lists

  def __eq__(self,s2):
    for i in range(3):
      for j in range(3):
        if self.b[i][j] != s2.b[i][j]: return False
    return True

  def __str__(self):
    # Produces a textual description of a state.
    # Might not be needed in normal operation with GUIs.
    txt = "\n["
    for i in range(3):
      txt += str(self.b[i])+"\n "
    return txt[:-2]+"]"

  def __hash__(self):
    return (self.__str__()).__hash__()
```


EightPuzzle.py (Code Excerpt 2)

```
def copy(self):
    # Performs an appropriately deep copy of a state,
    # for use by operators in creating new states.
    news = State({})
    news.b = [row[:] for row in self.b]
    return news

 def find_void_location(self):
    '''Return the (vi, vj) coordinates of the void.
    vi is the row index of the void, and vj is its column index.'''
    for i in range(3):
      for j in range(3):
        if self.b[i][j]==0:
          return (i,j)
    raise Exception("No void location in state: "+str(self))
```

EightPuzzle.py (Code Excerpt 3)

```
# Needed in the operator precondition testing:

 def can_move(self,dir):
    '''Tests whether it's legal to move a tile that is next
       to the void in the direction given.'''
    (vi, vj) = self.find_void_location()
    if dir=='N': return vi<2
    if dir=='S': return vi>0
    if dir=='W': return vj<2
    if dir=='E': return vj>0
    raise Exception("Illegal direction in can_move: "+str(dir))
```


EightPuzzle.py (Code Excerpt 4)

```
# Needed in the operator state-transformation function:

 def move(self,dir):
   '''Assuming it's legal to make the move, this computes
      the new state resulting from moving a tile in the
      given direction, into the void.'''
  news = self.copy() # start with a deep copy.
  (vi, vj) = self.find_void_location()
  b = news.b
  if dir=='N':
    b[vi][vj] = b[vi+1][vj]
    b[vi+1][vj] = 0
  if dir=='S':
    b[vi][vj] = b[vi-1][vj]
    b[vi-1][vj] = 0
  if dir=='W':
    b[vi][vj] = b[vi][vj+1]
    b[vi][vj+1] = 0
  if dir=='E':
    b[vi][vj] = b[vi][vj-1]
    b[vi][vj-1] = 0
  return news # return new state
```


EightPuzzle.py (Code Excerpt 5)

```
# Operator class definition. (General, problem-independent)

class Operator:
  def __init__(self, name, precond, state_transf):
    self.name = name
    self.precond = precond
    self.state_transf = state_transf

  def is_applicable(self, s):
    return self.precond(s)

  def apply(self, s):
    return self.state_transf(s)
#</COMMON_CODE>
```


EightPuzzle.py (Code Excerpt 6)

```
#<OPERATORS>
directions = ['N','E','W','S']
OPERATORS = [Operator("Move a tile "+str(dir)+" into the void",
                      lambda s,dir1=dir: s.can_move(dir1),
                      # The default value construct is needed
                      # here to capture the value of dir
                      # in each iteration of the list comp. iteration.
                      lambda s,dir1=dir: s.move(dir1) )
             for dir in directions]
#</OPERATORS>
```

Here is a list comprehension, used to create a list of instances of the Operator class.

Each operator is expressed using a pair of lambda expressions.

The use of keyword parameters in the lambda expressions creates closures that remember the arguments.

Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
  - Supporting Python constructs
- Wicked Problems
  - Rittel & Webber's 10 Criteria
  - The Climate Conundrum

List Comprehensions

- Given one list (or some kind of iterator), we can construct another in a convenient way.
- For example:

```
>>> range(4)
range(0, 4)
>>> [x*x for x in range(4)]
[0, 1, 4, 9]
```


List Comprehensions (cont.)

```
>>> import math
>>> [math.pow(2, i) for i in [7, 5, 3]]
[128.0, 32.0, 8.0]
>>> # Note that no assignment was needed in the above.
>>> # Now for looping with a pair of indices:
>>> pairs = [(i,j) for i in ['a','b'] for j in [0,1]]
>>> pairs
[('a', 0), ('a', 1), ('b', 0), ('b', 1)]
```


Lambda Expressions

The usual way to create a function is with def.

```
>>> def foe(x):
...   return x + ' foo!'
...
>>> foe('computing')
'computing foo!'
```

However, we can have a function without naming it.

```
lambda x: x+' foo!'
```

It could be applied directly

```
>>> (lambda x: x+' foo!')('programming')
'programming foo!'
```

Lambda expressions as args.

```
>>> def first_ten(seq_function):
...     return [seq_function(n) for n in range(10)]
...
>>> first_ten(lambda x: 2*x+1)
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> first_ten(lambda x: x*x*x)
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
```


Closures

A closure is a function that encapsulates the value(s) of some variable(s) that existed when the closure was created. Lambda expressions are commonly used for this purpose, especially when many functions need to be created using alternative values of the same variables.

```
>>> def make_five_adders():
...     return [lambda n,m1=7*m: n+m1 for m in range(5)]
...
>>> adders = make_five_adders()
>>> adders[3](14)
35
```

In the third function in the list, adders[3], the number 21 has been "closed into" this third function, as the value of its local variable m1.
When this function is applied to 14, the 21 is added to it, and the function returns 35.

Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
  - Supporting Python constructs
- **Wicked Problems**
  - Rittel & Webber's 10 Criteria
  - The Climate Conundrum

What Kinds of Problems Can be Approached Using the Classical Theory?

- Puzzles (we've seen several examples)
- Optimization problems (e.g., Traveling Salesman Problem) ?
- Global challenge problems (e.g., ending homelessness) ?

What Kinds of Problems Can be Approached Using the Classical Theory?

- Puzzles (we've seen several examples)
- Optimization problems (e.g., Traveling Salesman Problem) ?
- Global challenge problems (e.g., ending homelessness) ?

Herb Simon (2-time Nobel Prize winner, and Turing Award winter), in his widely-read book "The Sciences of the Artificial", suggested that the classical theory IS applicable to all three categories.

The 3rd category is sometimes known as "wicked" problems.

Origin of "wicked problem"

attributed by Rittel and Webber to

C. West Churchman.

"Guest Editorial" of *Management Science* (Vol. 14, No. 4, December 1967)

Wicked Problems

Rittel and Webber:

1. There is no definitive formulation of a wicked problem
2. Wicked problems have no stopping rule
3. Solutions to wicked problems are not true-or-false, but good-or-bad
4. There is no immediate and no ultimate test of a solution to a wicked problem
5. Every solution to a wicked problem is a "one-shot operation"; because there is no opportunity to learn by trial-and-error, every attempt counts significantly

Wicked Problems (cont.)

Rittel and Webber:

6. Wicked problems do not have an enumerable (or an exhaustively describable) set of potential solutions, nor is there a well-described set of permissible operations that may be incorporated into the plan
7. Every wicked problem is essentially unique
8. Every wicked problem can be considered to be a symptom of another problem
9. The existence of a discrepancy representing a wicked problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution
10. The planner has no right to be wrong

Why is Problem Formulation Difficult?

- Requires several kinds of knowledge.
  - domain knowledge: about the problem
  - about the problem-solving process
  - about computer programming
- Some problems are "ill-structured".
  - wicked problems, for example.

Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
  - Supporting Python constructs
- Wicked Problems
  - Rittel & Webber's 10 Criteria
  - **The Climate Conundrum**

Case Study: Global Warming

Case Study: Global Warming

- "Sustainable development: development that meets the needs of the present without compromising the ability of future generations to meet their own needs."
- -- *Our Common Future* (1987 report of the World Commission on Environment and Development, United Nations)

Why is the Climate Change Problem Difficult?

- Global scale
- Complexity
- Disinformation campaign
- Politics
- High costs & sacrifices required

IPCC Report of 2013

- "Warming of the climate system is unequivocal, and since the 1950s, many of the observed changes are unprecedented over decades to millennia. The atmosphere and ocean have warmed, the amounts of snow and ice have diminished, sea level has risen, and the concentrations of greenhouse gases have increased."

IPCC Report of 2013 (cont)

- "Continued emissions of greenhouse gases will cause further warming and changes in all components of the climate system. Limiting climate change will require substantial and sustained reductions of greenhouse gas emissions."

How to Begin Understanding?

- A daunting problem, tending to overwhelm potential solvers.

- How to begin? Select key features...
- --physical model (e.g., avg. temp. aggregated over space, time)
- --a few social/artificial constructs (e.g., government programs, expected effects, costs)

Conceptual Design of a Problem Template

Design interaction affordances:

1. visualization
   --graphic
   --text
2. operators

   Presentation of the operators in natural language and figures.

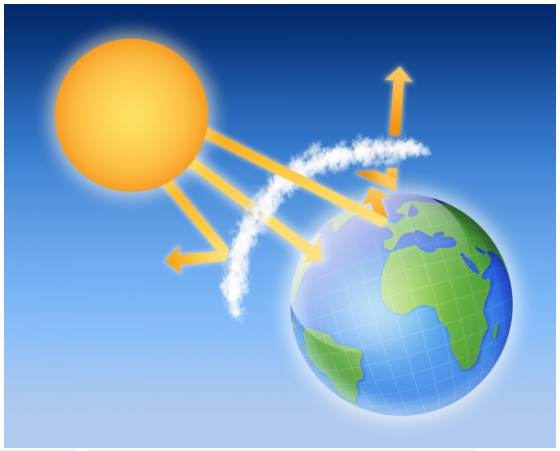
Human Value of Simplified Formulations

Counter the perception that
"This is a problem that most people just cannot get any grasp of, because it is too complex. We have no idea of what we can do that might have any effect."

Like a literary work of historical fiction, we communicate in a narrative that people can understand, that offers "cognitive handles" for grasping the key concepts involved.

Our posed problem may be a form of science fiction, but it is a step towards understanding reality because it highlights, and uses in context, the ideas needed to understand reality.

Basic Physical Model



Basic Physical Model

- Blackbody radiation equilibrium equation:
- energy absorbed = energy re-radiated

- $(1 - a)\, S\, \pi\, r^2 = 4\, \pi\, r^2\, \varepsilon\, \sigma\, T^4$
- S: solar constant
- a: earth albedo
- ε: earth emissivity
- σ: Stefan-Bolzmann constant
- r: radius of the earth
- T: earth temperature, degrees Kelvin

Basic Physical Model

Solve for T:

$$T = \sqrt[4]{(1 - a)\, S / 4\, \varepsilon\, \sigma}$$

Earth Albedo: a

Fraction of irradiance that is reflected:

$0 \le a \le 1$

Average value:

$a \approx 0.3$

Sample albedos

Surface	Typical albedo
Fresh asphalt	0.04[2]
Worn asphalt	0.12[2]
Conifer forest (Summer)	0.08,[3] 0.09 to 0.15[4]
Deciduous trees	0.15 to 0.18[4]
Bare soil	0.17[5]
Green grass	0.25[5]
Desert sand	0.40[6]
New concrete	0.55[5]
Ocean ice	0.5–0.7[5]
Fresh snow	0.80–0.90[5]

Earth Emissivity

Emissivity is the fraction of available radiation that escapes from a body

$0 \le \varepsilon \le 1$

Average earth $\varepsilon \approx 0.6$

Increasing greenhouse gases lower it, trapping more heat.

Clouds

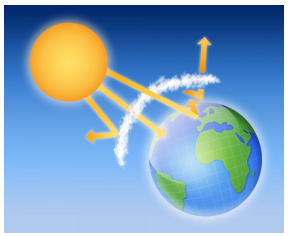
Clouds tend to block radiation:

increased albedo

decreased emissivity

Cooler days

Warmer nights



Contrails

Artificial clouds in jet wakes.

Just after 11 September 2001, flights to/from N. America stopped.

Daytime highs in N.A. were higher,

nighttime temps were lower.

Gamification of Global Warming

- Creating a challenge: survive until 2065
- Design of game operators with constraints:
  - Limited funds, interventions cost money
  - Engaging narrative

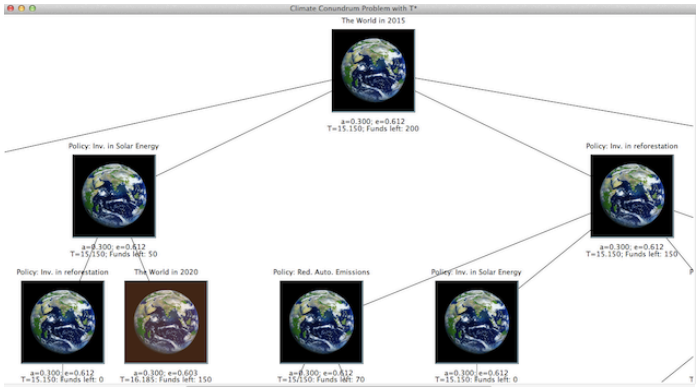
Modeling of Operator Effects

- This is perhaps the weakest scientific link in the formulation, but
- the most important in terms of engaging solvers.
- We seek:
  - SIMPLICITY (for user understanding),
  - NARRATIVE INTEGRITY (for user engagement)
  - STRUCTURAL INTEGRITY (leading to an interesting and comprehensible state space)

CLIMATE CONUNDRUM PUZZLE in TStar

- Key state variables:
  - albedo
  - emissivity
  - rate of change of emissivity
  - funds remaining
- Operators:
  - Invest in solar power
  - Invest in reducing automobile emissions
  - Invest in reforestation
  - Implement selected policies for 5 years

Portion of State Space Tree



BBC Climate Challenge

- Public Broadcasting Sponsored Formulation:
- Implementation in Flash
- Main game mechanic: Selection of policy cards.

- Impact: Led to development of Fate of the World.

"Fate of the World"

- Commercial Production and Marketed successor to BBC Climate Challenge
- Simulation of earth climate, economy, politics, with regional decomposition.
- Several alternative "missions" available.
- Very difficult to win (i.e., realistic)

Pf Problem formulation

Climate Model for FotW

- Attributed to Oxford Univ. researcher.
- Game has thousands of parameters.
- Opaque to players.
- Parameters available via posted mods.
- Design notes available in special edition.

Pf Problem formulation

Some Model Parameters

```
attr={
        agriculture_coal_per_robot=0,
        agriculture_emissions_per_coal=0,
        agriculture_emissions_per_energy=0,
        agriculture_emissions_per_fuelOil=0.3,
        agriculture_emissions_per_gas=0.2,
        agriculture_emissions_per_transport_fright_intl=0,
        agriculture_emissions_per_transport_fright_local=0,
        agriculture_emissions_per_transport_pers_intl=0,
        agriculture_emissions_per_transport_pers_local=0,
        agriculture_emissions_per_uranium=0,
        agriculture_emissions_per_water=0,
        agriculture_empr=4.26e-07,
        agriculture_empw=0,
        agriculture_energy_per_robot=6.8e-07,
        agriculture_fuelOil_per_robot=2.06e-07,
        agriculture_gas_per_robot=4.68e-07,
        agriculture_gdppw_base=1100,
        agriculture_gdppw_temp_product=1,
        agriculture_land_gdppw_mix_frac=0.55,
        agriculture_land_gdppw_weighting=0.0034,
        agriculture_livestock_gdppw_mix_frac=0.21,
        agriculture_livestock_gdppw_weighting=0.0177,
        agriculture_max_robotspw=8.0005,
        agriculture_meat_gdppw_mix_frac=0.23,
        agriculture_meat_gdppw_weighting=0.0177,
        agriculture_min_robots=6.46655e+07,
        agriculture_min_robotspw=0,
        agriculture_ringfencing_limit=1,
        agriculture_robot_growth_ceil=1.03,
        agriculture_robot_reduction=1,
        agriculture_robots=6.46655e+08,
        agriculture_robots_green_frac=0,
        agriculture_sea_gdppw_mix_frac=0.01,
        agriculture_sea_gdppw_weighting=0,
        agriculture_size_ratio=0.35,
        agriculture_toxicity_per_ton=0,
        agriculture_toxicity_total=0,
```


Pf Problem formulation

Summary of Problem Formulation for Global Warming

- Recognize its wickedness
- Determine most important factors
- Find a simple, reliable model for some factors
- Build elements of a formulation to balance scientific accuracy, cognitive requirements, computational/design tractability.
- Allow for multiple formulations, and evolutionary development.