


Problem Formulation

CSE 415: Introduction to Artificial Intelligence
University of Washington
Spring, 2019


© S. Tanimoto and University of Washington, 2019



Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
 - Supporting Python constructs


CSE 415, Univ. of Wash.
Problem Formulation
2



Outline

- **Motivation**
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
 - Supporting Python constructs

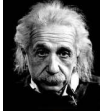
CSE 415, Univ. of Wash.
Problem Formulation
3




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



CSE 415, Univ. of Wash.
Problem Formulation
4




Motivation (cont.)

In AI, proper formulation supports:

- automatic solving
- computer-assisted manual solving
- problem-space analysis
- visualization
- use of heuristics
- reasoning about the problem

CSE 415, Univ. of Wash.
Problem Formulation
5



Outline

- Motivation
- **The 3 Phases of Problem Formulation**
- Case Study with the 8 Puzzle
 - Supporting Python constructs

CSE 415, Univ. of Wash.
Problem Formulation
6



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.

CSE 415, Univ. of Wash.

Problem Formulation



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.

(Preformulation)

CSE 415, Univ. of Wash.

Problem Formulation



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.

(Preformulation)

(Posing)

CSE 415, Univ. of Wash.

Problem Formulation



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.

(Preformulation)

(Posing)

(Coding the formulation)

CSE 415, Univ. of Wash.

Problem Formulation



Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
 - Supporting Python constructs

CSE 415, Univ. of Wash.

Problem Formulation

11



The Eight Puzzle

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



Fifteen Puzzle



Eight Puzzle

CSE 415, Univ. of Wash.

Problem Formulation

12



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..."

CSE 415, Univ. of Wash.

Problem Formulation

13



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 $\phi_i \in \Phi$ 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 $\phi_i \in \Phi$ zero or more times.

CSE 415, Univ. of Wash.

Problem Formulation

14



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.)

CSE 415, Univ. of Wash.

Problem Formulation

15



Eight Puzzle Formulation (cont).

- Note: These choices are not all forced.
- For example, a state COULD BE a string of characters, e.g., "ABCDEFGH".
- 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.

CSE 415, Univ. of Wash.

Problem Formulation

16



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. >>
```

CSE 415, Univ. of Wash.

Problem Formulation

17



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)
```

CSE 415, Univ. of Wash.

Problem Formulation

18

Pf
Problem
formulation

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__()
```

CSE 415, Univ. of Wash.

Problem Formulation

19

Pf
Problem
formulation

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))
```

CSE 415, Univ. of Wash.

Problem Formulation

20

Pf
Problem
formulation

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))
```

CSE 415, Univ. of Wash.

Problem Formulation

21

Pf
Problem
formulation

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
```

CSE 415, Univ. of Wash.

Problem Formulation

22

Pf
Problem
formulation

EightPuzzle.py (Code Excerpt 5)

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

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

CSE 415, Univ. of Wash.

Problem Formulation

23

Pf
Problem
formulation

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.

CSE 415, Univ. of Wash.

Problem Formulation

24



Outline

- Motivation
- The 3 Phases of Problem Formulation
- Case Study with the 8 Puzzle
 - Supporting Python constructs

CSE 415, Univ. of Wash.

Problem Formulation

25



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]
```

CSE 415, Univ. of Wash.

Problem Formulation

26



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)]
```

CSE 415, Univ. of Wash.

Problem Formulation

27



Lambda Expressions

The usual way to create a function is with def.

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

CSE 415, Univ. of Wash.

Problem Formulation

28



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]
```

CSE 415, Univ. of Wash.

Problem Formulation

29



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.

CSE 415, Univ. of Wash.

Problem Formulation

30