Chapter 5

Problem Solving before Programming

5.1 Problem-Solving Strategies
5.2 Algorithms as Paths Through “Problem Spaces”
5.3 Givens, Goals, and Resources
5.4 Analysing and Exploring Problem Spaces
5.5 From Blind Search to Informed Search
5.6 Analytical Reasoning
5.7 The Analogical Approach
5.8 References
5.1 Problem-Solving Strategies

Finding algorithmic solutions to programming problems is like solving mysteries in detective stories.

- **Analytical** problem-solving:
  - the “Hercule Poirot” approach
  - careful laying out of the evidence
  - logically putting together little clues

- **Analogical** approach:
  - the “Jane Marple” approach
  - finding solutions by analogy or comparison to already solved problems
• Trial and error, hacking:

This approach is, of course, less recommended — at least for beginners.
5.2 Algorithms as Paths Through “Problem Spaces”

**Algorithm:** a precise specification of a behaviour intended to solve a well-defined problem

**Coding:** the translation of an algorithm into a particular programming language

**Program:** a step-by-step execution plan that a computer can perform as a sequence of simple steps or instructions
Sequences can have complex underlying structures.

**Example:** sentences

- **Dagwood**
  - *Name*
  - *Subject*
- **likes**
  - *Verb*
  - *Predicate*
- **his**
  - *Pronoun*
- **fast**
  - *Adjective*
- **computer**
  - *Noun*
  - *Object*
- **Sentence**
5.3 Givens, Goals, and Resources

• **Givens:** initial conditions of the problem (the way things are)

• **Goals:** desired state(s) (the way things should be)

Any problem can be formulated at least as givens and goals.

• **Resources:** the means or methods that can transform a problem state into another, in order to move the solver from givens to goals in a step-by-step manner

Differences in givens, goals, and resources make different problems (example: different levels of programming environments).
Examples:

- **Game playing**
  - Given: initial game setup
  - Goal: win!
  - Resources: possible moves

- **Preparing a meal**
  - Givens: available ingredients and cooking facilities
  - Goal: end up with something to eat
  - Resources: recipe, the cook’s experience, ...

**Note**: Many of our daily problems are ill-defined: the givens, goals, resources, or combinations of the three, is **unclear**.

Example: client — application programmer — programming language — computer
5.4 Analysing and Exploring Problem Spaces

**Problem space:** a complete set of possible states, generated by exploring all the possible steps, or moves, which may or may not lead from a given state to a goal state.
What should we know for a preliminary analysis of a problem?

- What are the **givens**? Do we really have all of them?
  - Are the givens **specific** to a particular situation?
  - Can we **generalize**?
  - Is there a **notation** that represents the givens and other states succinctly?

- What is the **goal**?
  - Is there are single goal, or are there several?
  - If there is a single goal, can it be split into pieces?
  - If there are several goals or subgoals, are they independent or are they connected?
  - Are there obstacles to overcome to reach a goal? How can they be overcome?
  - Are there any constraints on developing a solution?
• What are the **resources** (moves, operators, procedures, rules, transformations)?
  - For each resource, are there constraints, or preconditions, to be addressed before applying it?
  - If so, are there other, possibly simpler, resources that meet the preconditions?
  - What are the variants? When you apply a resource, what changes?
  - What are the invariants? When you apply a resource, what stays the same?
  - Are there more powerful resources for this problem?
  - Would someone else know these resources?
  - ...
5.5 From Blind Search to Informed Search

Example: the FOX-CORN-GOOSE problem

Search tree (for two actions)
State space graph for the FOX-CORN-GOOSE problem

- P F G C /
- F G C / P
- G C / P F
- F C / P G
- F G / P C
- P G C / F
- P F C / G
- P F G / C
- C / P F G
- G / P F C
- F / P G C
- P C / F G
- P G / F C
- P F / G C
- / P F G C
5.6 Analytical Reasoning

Example: Planning a menu for a meal

I. **Preceding Courses:**

   A. **Soup course:**
      1. Cream of lettuce
      2. Light sherry

   B. **Fish course:**
      1. Filet of sole Veronique

II. **Main Course:**

   A. **Main dish:**
      1. Cornish game hen à l’orange
      2. Red wine

   B. **Accompanying dishes:**
      1. Creamed spinach
      2. Wild rice
      3. **Beverage:** dry white wine
III. **Following Courses:**

A. Salad course:
   1. Assorted green
   2. Vinaigrette dressing

B. Dessert:
   1. Mandarin orange tart
   2. Sweet, heavier white whine

Laying out an algorithm or a problem solution is like planning a menu.

The different parts of the problem solution are developed in a **top-down** fashion.

An alternative approach would be to develop **bottom-up** (e.g., starting from the ingredients).
## 5.7 The Analogical Approach

<table>
<thead>
<tr>
<th>Patient</th>
<th>Vomiting</th>
<th>Abdominal pain</th>
<th>Fever</th>
<th>Pulse</th>
<th>Drug successful</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>yes</td>
<td>yes</td>
<td>high</td>
<td>elevated</td>
<td>yes</td>
</tr>
<tr>
<td>B</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>normal</td>
<td>yes</td>
</tr>
<tr>
<td>C</td>
<td>yes</td>
<td>no</td>
<td>high</td>
<td>normal</td>
<td>no</td>
</tr>
<tr>
<td>D</td>
<td>no</td>
<td>yes</td>
<td>high</td>
<td>elevated</td>
<td>no</td>
</tr>
<tr>
<td>E</td>
<td>no</td>
<td>no</td>
<td>high</td>
<td>elevated</td>
<td>no</td>
</tr>
<tr>
<td>F</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>elevated</td>
<td>yes</td>
</tr>
<tr>
<td>New patient</td>
<td>yes</td>
<td>yes</td>
<td>low</td>
<td>normal</td>
<td>?</td>
</tr>
</tbody>
</table>
5.8 References