Chapter 01
An Introduction to Artificial Intelligence

Instructor

LE Thanh Sach, Ph.D.

Instructor's Information

LE Thanh Sach, Ph.D.

Office:

Department of Computer Science,

Faculty of Computer Science and Engineering,

HoChiMinh City University of Technology.

Office Address:

268 LyThuongKiet Str., Dist. 10, HoChiMinh City, Vietnam.

E-mail: LTSACH@cse.hcmut.edu.vn

E-home: http://cse.hcmut.edu.vn/~ltsach/

Tel: (+84) 83-864-7256 (Ext: 5839)

Outline

- What is AI?
- The foundations of AI
- A brief history of AI
- The state of the art
- Introductory problems
- Tools and Programming Languages for AI

What is AI?

- Intelligence:
  - "ability to learn, understand and think" (Oxford dictionary)

- Artificial Intelligence:
  - "attempts to understand intelligent entities"
  - "strives to build intelligent entities" (Stuart Russel & Peter Norvig [1])

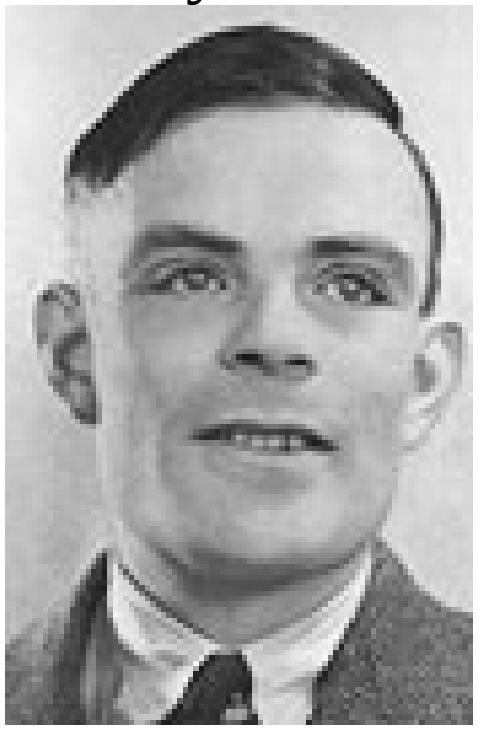
What is AI?

❖ Schools of Thought:

Thinking humanly	Thinking rationally
Acting humanly	Acting rationally

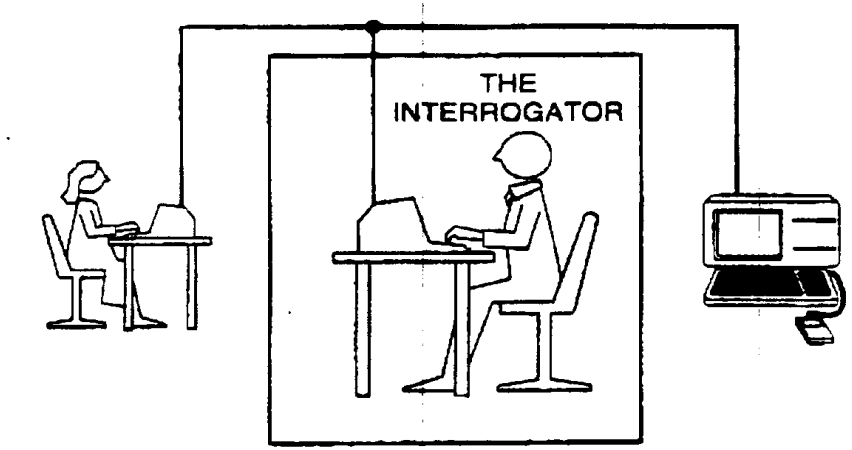
Acting Humanly: The Turing Test

- Alan Turing (1912-1954)
- "Computing Machinery and Intelligence" (1950)



Acting Humanly: The Turing Test

- Can machines think?" → "Can machines behave intelligently?"
- Operational test for intelligent behavior: the Imitation Game



Acting Humanly: The Turing Test

- Predicted that by 2000, a machine might have a 30% chance of fooling a lay person for 5 minutes.
- Anticipated all major arguments against AI in following 50 years.
- Suggested major components of AI: knowledge, reasoning, language, understanding, learning.

Thinking Humanly: Cognitive Modelling

- Not content to have a program correctly solving a problem.

  More concerned with comparing its reasoning steps to traces of human solving the same problem.

- Requires testable theories of the workings of the human mind: cognitive science.
  - How does a human think?

- Typical research:
  - General Problem Solver (GPS), Newell & Simon

Thinking Rationally: Laws of Thought

- Aristotle was one of the first to attempt to codify "right thinking", i.e., irrefutable reasoning processes.
  - Aristotle: ~ 430BC
  - Syllogism:
    - ✓ Major premise: **All men are mortal.**
    - ✓ Minor premise: **Socrates is a man.**
    - ✓ Conclusion: **Socrates is mortal.**

- Formal logic provides a precise notation and rules for representing and reasoning with all kinds of things in the world.
  - Example:
    - ✓ Major premise: **Men(X) $\Rightarrow$ Mortal(X)**
    - ✓ Minor premise: **Men(Socrates)**
    - ✓ Conclusion: **Mortal(Socrates)**

Thinking Rationally: Laws of Thought

- Obstacles:
  - Informal knowledge representation.
  - Computational complexity and resources.

Acting Rationally

- Acting so as to achieve one's goals, given one's beliefs.
- Does not necessarily involve thinking.
- Advantages:
  - More general than the "laws of thought" approach.
  - More amenable to scientific development than human-based approaches.

The Foundations of AI

- Philosophy (423 BC – present):
  - Logic, methods of reasoning.
  - Mind as a physical system.
  - Foundations of learning, language, and rationality.

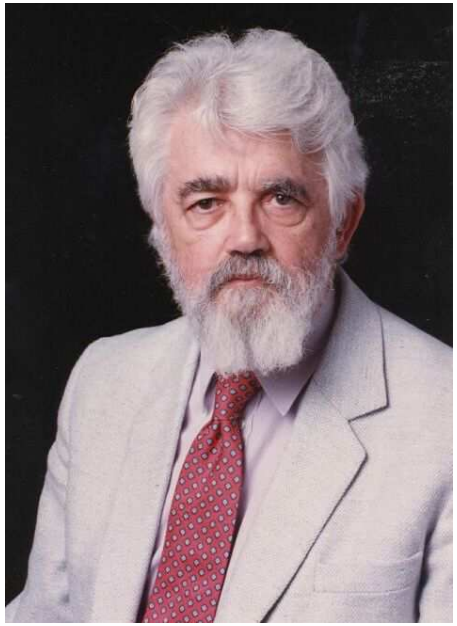
- Mathematics (c.800 – present):
  - Formal representation and proof.
  - Algorithms, computation, decidability, tractability.
  - Probability.

The Foundations of AI

- Psychology (1879 – present):
  - Adaptation.
  - Phenomena of perception and motor control.
  - Experimental techniques.
- Linguistics (1957 – present):
  - Knowledge representation.
  - Grammar.

A Brief History of AI

- The gestation of AI (1943 – 1956):
  - 1943: McCulloch & Pitts: Boolean circuit model of brain.
  - 1950: Turing's "Computing Machinery and Intelligence".
  - 1956: McCarthy's name "**Artificial Intelligence**" adopted.



A Brief History of AI

- Early enthusiasm, great expectations (1952 – 1969):
  - Early successful AI programs: Samuel's checkers, Newell & Simon's Logic Theorist, Gelernter's Geometry Theorem Prover.
  - Robinson's complete algorithm for logical reasoning.

A Brief History of AI

- A dose of reality (1966 – 1974):
  - AI discovered computational complexity.
  - Neural network research almost disappeared after Minsky & Papert's book in 1969.
- Knowledge-based systems (1969 – 1979):
  - 1969: DENDRAL by Buchanan et al..
  - 1976: MYCIN by Shortliffle.
  - 1979: PROSPECTOR by Duda et al..

A Brief History of AI

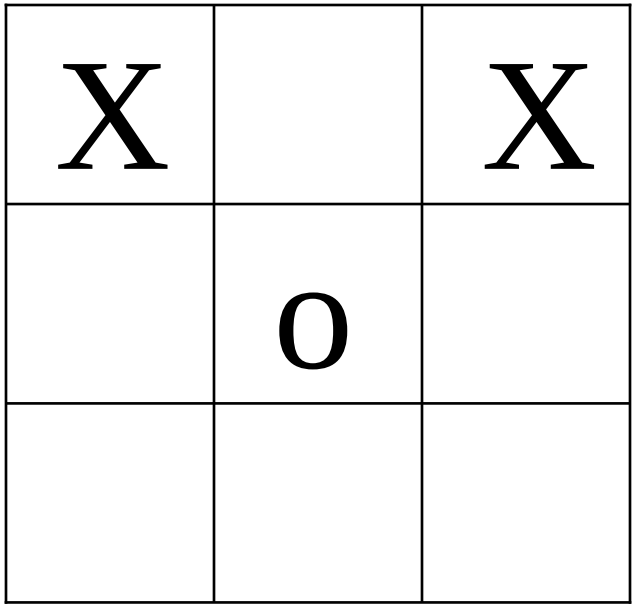
- AI becomes an industry (1980 – 1988):
  - Expert systems industry booms.
  - 1981: Japan's 10-year Fifth Generation project.

- The return of NNs and novel AI (1986 – present):
  - Mid 80's: Back-propagation learning algorithm reinvented.
  - Expert systems industry busts.
  - 1988: Resurgence of probability.
  - 1988: Novel AI (ALife, GAs, Soft Computing, …).
  - 1995: Agents everywhere.
  - 2003: Human-level AI back on the agenda.

The State of the Art

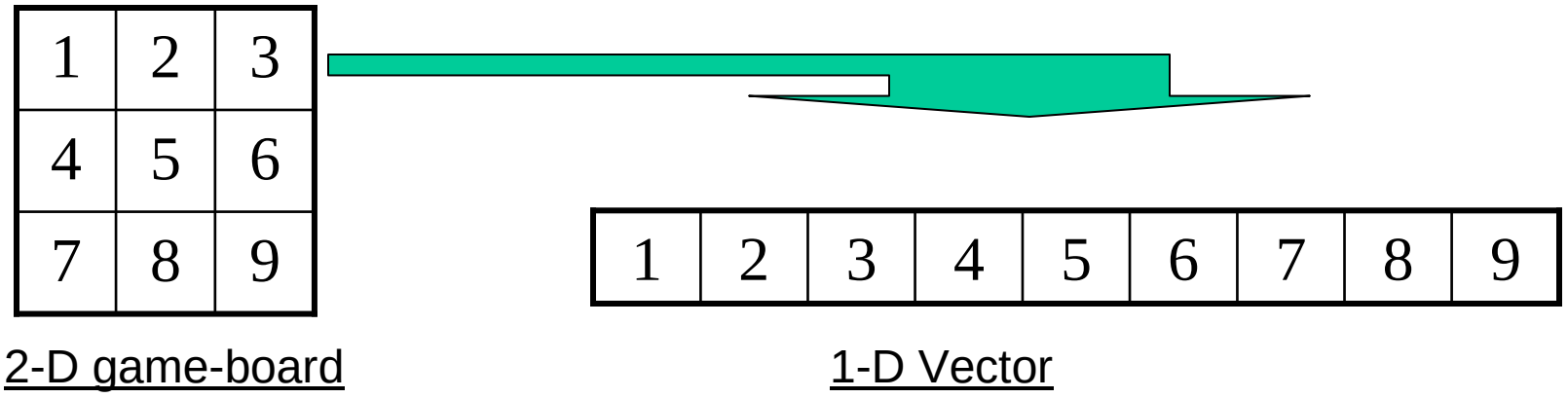
- Computer beats human in a chess game.
- Computer-human conversation using speech recognition.
- Computer program can chat with human
- Expert system controls a spacecraft.
- Robot can walk on stairs and hold a cup of water.
- Language translation for webpages.
- Home appliances use fuzzy logic.
- .....

Introductory Problem: Tic-Tac-Toe



Tic-Tac-Toe: Solution 1

- Data Structure:



Tic-Tac-Toe: Solution 1

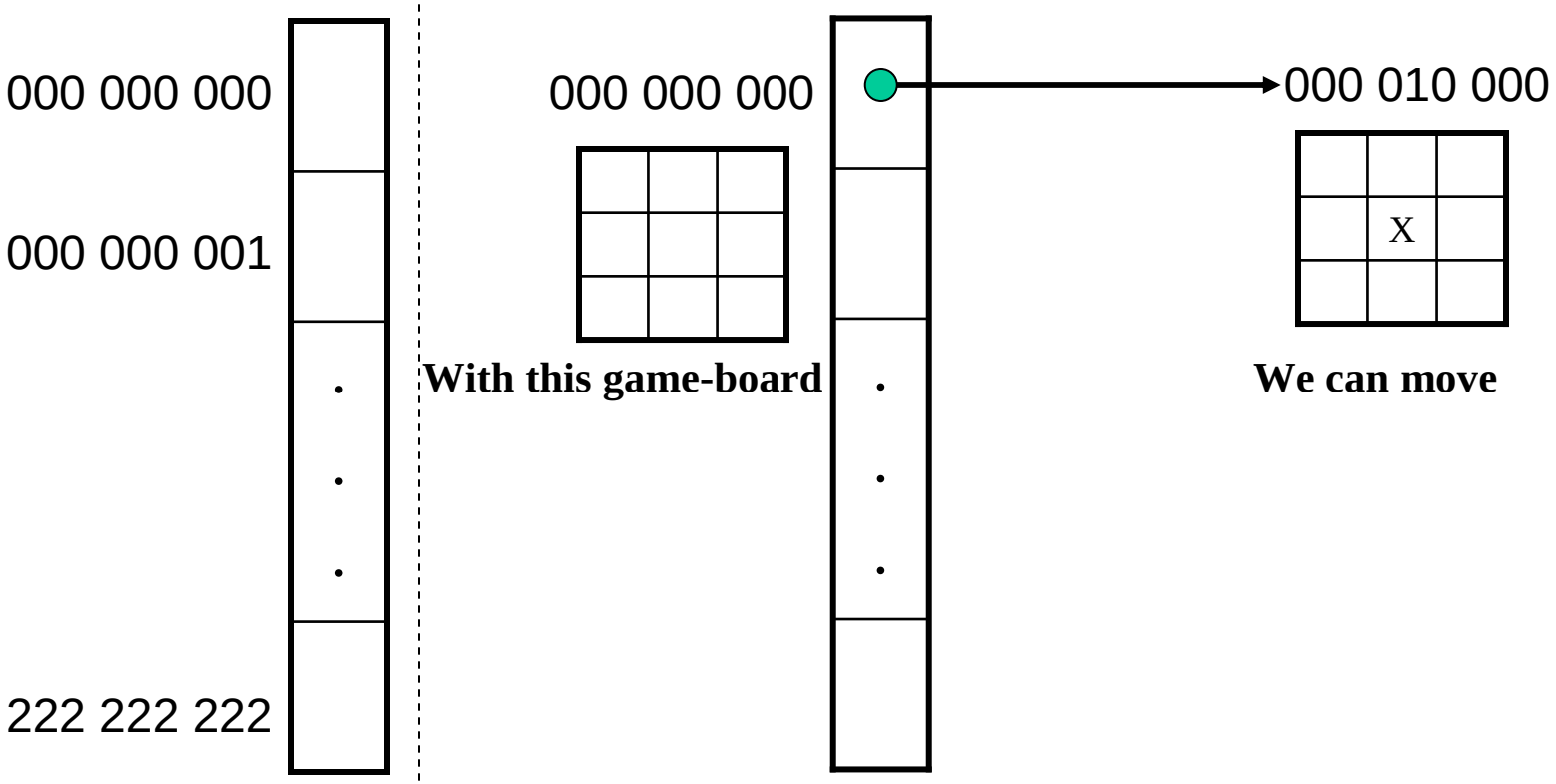
- Elements of vector:
  - ✓ 0 : Empty
  - ✓ 1 : X
  - ✓ 2 : O

⇒ The vector is a ternary number

- Store inside the program a move-table (lookup-table):
  - ✓ #Elements in the table: 19683 ($=3^9$)
  - ✓ Element = A vector which describes the most suitable move from the current game-board

Tic-Tac-Toe: Solution 1

❖ Data Structure:



Tic-Tac-Toe: Solution 1

- Algorithm:
  1. View the vector as a ternary number. Convert it to a decimal number.
  2. Use the computed number as an index into Move-Table and access the vector stored there.
  3. Set the new board to that vector.

Tic-Tac-Toe: Solution 1

❖ Comments:

1. A lot of space to store the Move-Table.

2. A lot of work to specify all the entries in the Move-Table.

3. Difficult to extend.

Tic-Tac-Toe: Solution 2

- Data Structure:
  - Use vector, called board, as Solution 1
  - However, elements of the vector:
    - 2 : Empty
    - 3 : X
    - 5 : O
  - Turn of move: indexed by integer
    - 1,2,3, etc.

Tic-Tac-Toe: Solution 2

❖ Function Library:

1. Make2:

- Return a location on a game-board.

1	2	3
4	5	6
7	8	9

```
IF (board[5] = 2)
   RETURN 5; //the center cell.
ELSE
   RETURN any cell that is not at the board's corner;
   // (cell: 2,4,6,8)
```

Tic-Tac-Toe: Solution 2

- Function Library:
  - Let P represent for X or O
  - can_win(P) :
    - ✓ P has filled already at least two cells on a straight line (horizontal, vertical, or diagonal)
  - cannot_win(P) = NOT(can_win(P))

Tic-Tac-Toe: Solution 2

- Function Library:

  2. Posswin(P):

```
IF (cannot_win(P))
   RETURN 0;
ELSE
   RETURN index to the empty cell on the line of
   can_win(P)
```

Tic-Tac-Toe: Solution 2

- Function Library:
  - Let odd numbers are turns of X
  - Let even numbers are turns of O

3. Go(n): make a move.

```
IF odd(turn) THEN          // for X
   Board[n] = 3
ELSE                       // for O
   Board[n] = 5
turn = turn + 1
```

Tic-Tac-Toe: Solution 2

❖ Algorithm:

1. Turn = 1: (X moves)

```
Go(1)  //make a move at the left-top cell
```

2. Turn = 2: (O moves)

```
IF board[5] is empty THEN
        Go(5)
ELSE
        Go(1)
```

Tic-Tac-Toe: Solution 2

❖ Algorithm:

3. Turn = 3: (X moves)

```
IF board[9] is empty THEN
        Go(9)
ELSE
        Go(3).
```

4. Turn = 4: (O moves)

```
IF Posswin(X) <> 0 THEN
    Go(Posswin(X))
    //Prevent the opponent to win
ELSE Go(Make2)
```

Tic-Tac-Toe: Solution 2

❖ Algorithm:

5. Turn = 5: (X moves)

```
IF Posswin(X) <> 0 THEN
        Go(Posswin(X))
        //Win for X.
ELSE IF Posswin(O) <> THEN
        Go(Posswin(O))
        //Prevent the opponent to win
ELSE IF board[7] is empty THEN
        Go(7)
ELSE Go(3).
```

Tic-Tac-Toe: Solution 2

❖ Comments:

1. Not efficient in time, as it has to check several conditions before making each move.

2. Easier to understand the program's strategy.

3. Hard to generalize.

Tic-Tac-Toe: Solution 2

8	3	4
1	5	9
6	7	2

15 – (8 + 5)

Tic-Tac-Toe: Solution 2

❖ Comments:

1. Checking for a possible win is quicker.

2. Human finds the row-scan approach easier, while computer finds the number-counting approach more efficient.

Tic-Tac-Toe: Solution 3

- Data structure:

1. Game-board: Use vector as described for the above program
2. List:
   - ✓ Contains possible game-boards generated from the current game-board
   - ✓ Each the game-board is augmented a score indicating the possibility of victory of the current turn

Tic-Tac-Toe: Solution 3

❖ Algorithm:

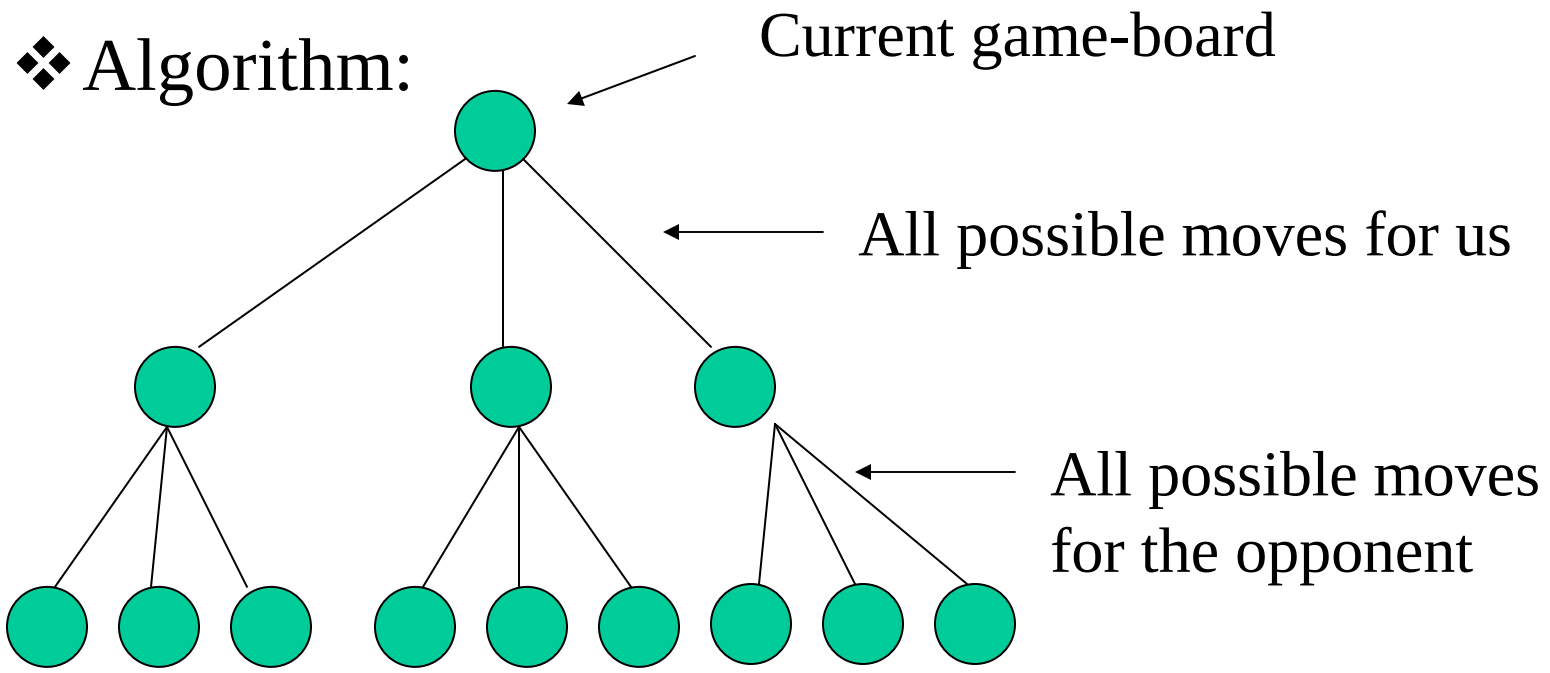
1. If it is a win, give it the highest rating.

2. Otherwise, consider all the moves the opponent could make next. Assume the opponent will make the move that is worst for us. Assign the rating of that move to the current node.

3. The best node is then the one with the highest rating.

Tic-Tac-Toe: Solution 3

❖ Algorithm:



Please see more detail in Chapter: Game Theory

Tic-Tac-Toe: Solution 3

❖ Comments:

1. Require much more time to consider all possible moves.

2. Could be extended to handle more complicated games.

Introductory Problem: Question Answering

"Mary went shopping for a new coat. She found a red one she really liked. When she got it home, she discovered that it went perfectly with her favourite dress".

Q1: What did Mary go shopping for?

Q2: What did Mary find that she liked?

Q3: Did Mary buy anything?

Question Answering: Solution 1

- Data structure:
  - Question Patterns:
    - ✓ Set of predefined TEMPLATEs.
      Given TEMLATEs, generate PATTERNs to match with input text. For example,

TEMPLATE: **"Who did X Y"**

PATTERN: **"X Y Z"**

ANSWER: **Z**

Question Answering: Solution 1

- Algorithm:

1. MATCH(TEMPLATE, QUESTION) ➔ PATTERNs.

2. Extend the generated set of PATTERNs by adding patterns that change the verb in the original pattern. For example, from “go”, we can add patterns for “went”, “goes”, etc.

3. MATCH(PATTERNs, TEXT) ➔ ANSWER.
4. Return Answers.

Question Answering: Solution 1

- Algorithm:

Example:

- ✓ TEMPATE: "What did X Y"
- ✓ QUESTION: "What did Mary go shopping for"
- ➔ PATTERN:

Mary go shopping for Z

- ➔ PATTERN^{+}:

Mary go shopping for Z ;

Mary goes shopping for Z;

Mary went shopping for Z

Question Answering: Solution 1

- Q1: What did Mary go shopping for?
  (Original text: Mary went shopping for a new coat)
- PATTERN:

  Mary go shopping for Z ;

  Mary goes shopping for Z;

  Mary went shopping for Z
- INPUT TEXT:

  Mary went shopping for a new coat

- ANSWER:

  Z = a new coat.

Question Answering: Solution 1

- Q2: What did Mary find that she liked?
  (Original text: She found a red one she really liked)

  - Need further processing:
    - ✓ Inserting new word, like “really”.
    - ✓ Relating noun to pronoun, i.e., Mary → She.

  - Answers:
    “a red one”
    “a red coat”

Question Answering: Solution 1

- Q3: Did Mary buy anything?

  (Original text: When she got it home, she discovered that it went perfectly with her favourite dress)

  - Cannot answer, because there is no word "buy" in the input text.

Question Answering: Solution 1

- Comments:
  - Heavily depend on syntax analysis (using template, patterns).
  - Difficult to realize in the real world.
  - Similar to a well-known program called ELIZA.

Question Answering: Solution 2

- Data structure:
  - Should have an efficient representation for input text, input questions.

Question Answering: Solution 2

Event2:	
Instance:	Finding
Tense:	Past
Agent:	Mary
Object:	THING1

THING1:	
Instance:	Coat
Color:	Red

Event2:	
Instance:	Liking
Tense:	Past
Agent:	Mary
Object:	THING1

Concepts in the current problem:

Mary.
Coat.
Liking.
Finding.

Question Answering: Solution 2

- Algorithm:

1. Convert the input text to the internal representation, referred to as structured text.

2. Convert the input question to the internal representation, referred to as structured question. Focus on the parts which will be returned from the questions. For example,
   - With representation as Slot-Filler: focus on the fields that will be returned
   - With logic representation: focus on the value of some variable.

Question Answering: Solution 2

❖ Algorithm:

3. MATCH(structured text, structured question).

4. Return the parts which marked as results for the questions.

- For example: with the three questions above.

Q1: "a new coat"

Q2: "a red coat"

Q3: Cannot answer.

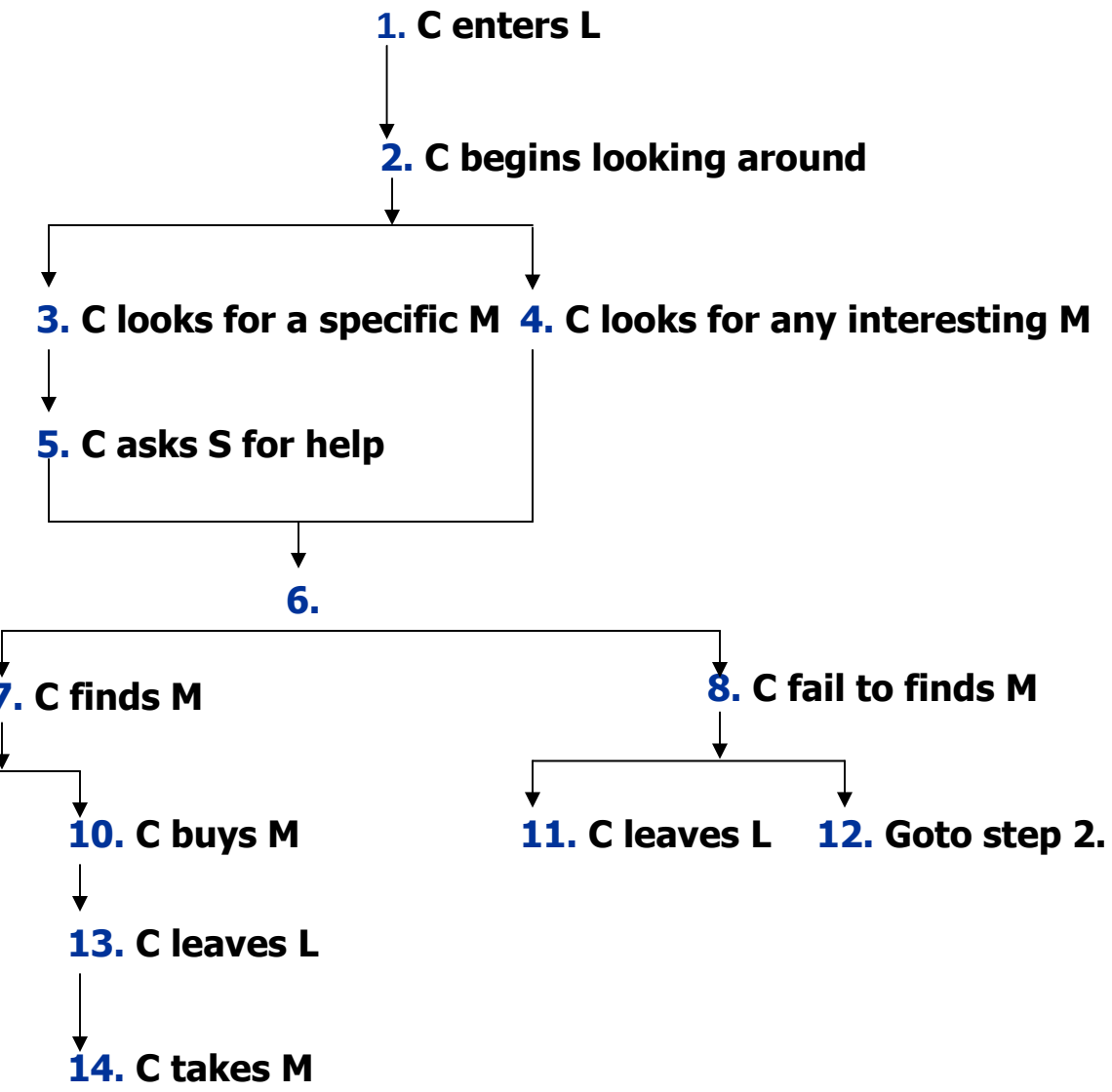
Question Answering: Solution 2

- Comments:
  - More efficient than Solution 1.
  - Using semantic and knowledge inside.
  - Time-consuming when doing "match"
  - Knowledge representation, in general, is a difficult problem.

Question Answering: Solution 3

- Data structure:
  - World Model:
    - ✓ An internal representation for the world.
    - ✓ Including: objects, actions, situation, see the following chart.
  - IntegratedText:
    - ✓ An internal representation of the input text that has the relation to the world model.
  - Other knowledge: like Solution 2.

Example: about a shopping tour.



- Roles:
  - C = customer.
  - S = salesperson.
  - M: merchandise
  - D: dollars
  - L = a store

“Mary went shopping for a new coat. She found a red one she really liked. When she got it home, she discovered that it went perfectly with her favourite dress”.

Q1: What did Mary go shopping for?

Q2: What did Mary find that she liked?

Q3: Did Mary buy anything?

Question Answering: Solution 3

- Algorithm:

1. Convert the input text to an internal representation.

2. Convert the input question to an internal representation.

3. MATCH(result of Step 1, result of Step 2).

4. Return answer.

Question Answering: Solution 3

- Algorithm:
  - Example: With the three questions above

Q1: like Solution 2

Q2: like Solution 2

Q3: Can answer:

- ✓ Run script from 1 to 14.
- ✓ Assign: C= Mary, M = a red coat.
- ✓ Step 10: “Mary buys a red coat”
- ✓ Hence, Answer is “She bought a red coat”

Question Answering: Solution 3

- Comments
  - Outperform the two previous solutions.
  - Need additional reasoning to answer more complex questions.

Tools and Programming Languages for AI

- Programming Languages:
  - Prolog
  - Java
  - C++
- Tools:
  - For building expert systems or rule-based systems:
    - ✓ CLIPS: C Language Integrated Production System
    - ✓ JESS: Java Expert System Shell
  - For processing semantics:
    - ✓ RDF, RDFS, OIL

Homework

1. Read “Computing Machinery and Intelligence” (1950).

2. Read documents related to
   - Expert System.
   - Knowledge Representation and languages.