

### A little history:

Hilbert's questions (1900)

Godel's theorem (1930)

*Entscheidungs* problem &  
Turing machines (1936)

Turing's later history

1

### Responses to questions

- In your opinion, is the Turing test (or "imitation game") a good way to answer the question "Can Machines Think?"? Why or why not? What weaknesses do you think it has (if any)?
- Give three questions you would ask if you were a judge in a Turing test.
- Why do you think it has been so easy for current-day chat boxes to fool judges at the Loebner Prize competitions?
- In class we discussed "Captchas" ("Computer Automated Public Turing test to tell Computers and Humans Apart") and how the current ones used on Yahoo and elsewhere may in danger of being solved soon by computer programs. Can you suggest a Captcha that might be more robust, but still easy and quick to generate?

2

### Assignments for next class (due Monday October 1):

- Read Searle paper (on class web page)
- Read *Vehicles*, pp. 1-19
- Answer reading questions
- Do exercises for "problem-solving as search"
- Think about AI system or topic area for presentation:
  - For undergrads: Choose broad topic area you are interested in (e.g., Speech recognition)
  - For grad students: what I mean by an "AI system". Let me know if you need suggestions on what to cover.
  - For all: On Monday I'll pass around sign-up sheet for presentation topics (or AI systems)

3

### A few comments on the Searle paper

- Who is Searle?
- Why did/do people pay so much attention to this article?
- What does he mean by "intentionality"?
- What does he mean by "physical symbol system" and "computational processes over formally defined elements"?

4

### *Vehicles*

by Valentino Braitenberg

- Building an intelligent system from scratch
- Understanding how intelligence can emerge from non-intelligent substrate

5

### Vehicle 1: *Getting around*

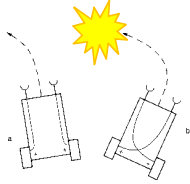
This and following slides adapted from  
"Notes on Braitenberg's Vehicles"  
([http://instruct.westvalley.edu/lafave/Vehicles\\_online.html](http://instruct.westvalley.edu/lafave/Vehicles_online.html))



- Components: Sensor and motor.
- Principle: The more there is of the quality (e.g., heat) to which the sensor is tuned, the faster the motor goes.
- Description: alive, restless, doesn't like "heat" (will slow down in cold regions, speed up in hot regions)

6

### Vehicle 2a: Fear

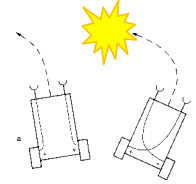


Vehicle 2a:

- Components: 2 sensors, 2 motors, each sensor connected to the motor on the same side ("uncrossed")
- Principle: The more there is of the quality to which the sensor is tuned, the faster the motors go ("excitatory").
- Description: Will run away from source to which the sensor is tuned; occasionally "attacks" it

7

### Vehicle 2b: Aggression

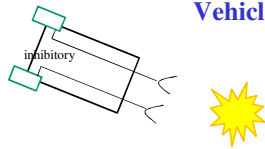


Vehicle 2a:

- Components: 2 sensors, 2 motors, each sensor connected to the motor on the opposite side ("crossed")
- Principle: The more there is of the quality to which the sensor is tuned, the faster the motors go ("excitatory").
- Description: dislikes source to which the sensor is tuned; "attacks" it

8

### Vehicle 3: Love



- Components: 2 sensors, 2 motors, each sensor connected to the motor on the same side ("uncrossed")
- Principle: The more there is of the quality to which the sensor is tuned, the *slower* the motors go ("inhibitory").
- Description: loves the source, wants to be near it, comes to rest facing it

9

### Vehicles Simulation

10

### Rod Brooks' subsumption architecture

Slides adapted from

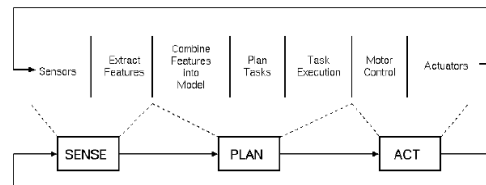
[http://www.inf.ed.ac.uk/teaching/courses/iar5/Notes/3\\_Subsum.pdf](http://www.inf.ed.ac.uk/teaching/courses/iar5/Notes/3_Subsum.pdf)

and

<http://www.cs.uml.edu/~holly/91.451/Spring2005/architecture-lecture.pdf>

11

### Traditional robotics: Hierarchical Organization is "Horizontal"

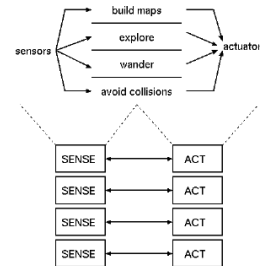


From Murphy 2000

Horizontal Behaviors: Accomplish Steps Sequentially

13

## Biology and Reactive Architectures are more “Vertical”



From Murphy 2000

Vertical Behaviors: Many Active at Once

15

### Task Decomposition: Advantages of vertical approach

- Don't have to build all parts before testing
- Immediate appreciation of effects of embodiment and situatedness
- Multiple goals pursued in parallel, late decision
- Multiple sensors without requiring fusion
- Each layer adds competence to already working robot – graceful degradation if higher level fails
- Can map onto hardware e.g. new processors for each new level of behaviour - additivity

16

### Argument from evolution (Brooks, 1990)

- |   |   |   |
|---|---|---|
| <ul style="list-style-type: none"> <li>• 3.5 billion years, single cells</li> <li>• 550 million years, vertebrates</li> <li>• 450 million years, insects</li> <li>• 250 million years, mammals</li> <li>• 120 million years, primates</li> <li>• 18 million years, ape predecessors</li> <li>• 2.5 million years, humans</li> </ul> | } | <p>The hard bit:<br/>Surviving and reproducing in dynamic environment</p> |
| <ul style="list-style-type: none"> <li>• 19,000 years, agriculture</li> <li>• 5000 years, writing</li> <li>• 500 years, 'expert' knowledge</li> </ul>   | } | <p>The easy bit:<br/>Language, problem solving, reasoning...</p>          |

## The behaviour-based approach

- Always build a complete agent
- Start with simple tasks rather than simplified environments – work in real world
- Build a robust system and then increment
- Exploit physics when possible, using interaction with environment for emergent behaviours
- Avoid unnecessary representations, eliminate external calibration

18

## The subsumption architecture

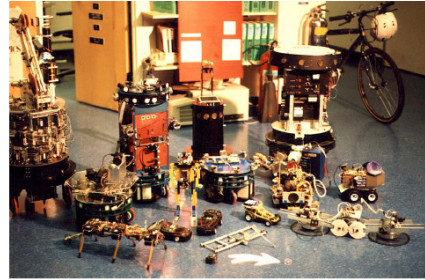
- Decompose problem into layers of competence
- Each layer uses sensors, actuators and control
- Build and debug lowest layers first
- Add new layers without changing lower ones
- New layers will ‘subsume’ the output of lower layers

E.g. ‘Polly’ –  
Horswill, 1989

Provide Tour  
Recognise Person  
Navigate  
Avoid hazards

19

## Subsumption-based Robots



20

[Brooks video](#)

21

## Applications

- [iRobot Corp.](#)
- [Roomba ad/demo](#)

22

## Conclusions

- **Produced some very elegant and successful robots:**
  - Still widely used in robot and agent approaches
  - But did not see continuous evolution to higher capabilities (e.g., “Cog” project)
- **Wide influence across AI and related fields:**
  - Importance of embodiment and situatedness
  - Possibilities for low-level sensorimotor coupling, exploiting environments, emergent behaviours
  - Use of world rather than internal representations
  - Solving problems with physics and hardware as well as software

## More on “projects”

- Presentations will be during last week of classes (and possibly during finals week).
- Final paper due by the end of the term.
- List of general topic areas
- Form teams
- Short project description from each team by Wednesday Oct. 8.

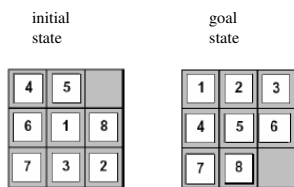
24

## Problem-solving as search

## Some classic AI search problems

- “Toy problems”
  - 15-puzzle
  - Missionaries and cannibals
- Solving algebraic equations
- Proving theorems
- Robotic path planning
- Game-playing

## 8-puzzle



- Can use search to:
- find any solution
  - find solution with minimal number of moves

## What is size of state space for 8-puzzle?

**Assume that from any possible configuration of the 8-Puzzle we can get to any other possible configuration.**

## What is size of state space for 8-puzzle?

**Assume that from any possible configuration of the 8-Puzzle we can get to any other possible configuration.**

Size of state space  $\approx 9! = 181,440$

Size of 15-puzzle state space?

Size of 24-puzzle state space?

Can't do exhaustive search!

## Approximate number of states

- Tic-Tac-Toe:  $3^9$
- Checkers:  $10^{40}$
- Rubik's cube:  $10^{19}$
- Chess:  $10^{120}$

## How to solve a problem by searching

1. Define search space
  - Initial, goal, and intermediate states
2. Define operators for expanding a given state into its possible successor states
3. Define  $cost(n)$ : Cost to get from initial state to node  $n$ .
4. Apply search algorithm to find path from initial to goal state, while avoiding repeating a state during the search.

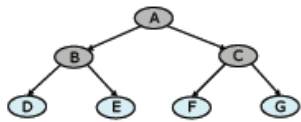
Examples...

## Search methods

- **Uninformed search:**
  - Breadth-first
  - Depth-first
  - Depth-limited
  - Iterative deepening depth-first
  - Bidirectional
- **Informed (or heuristic) search:**
  - Greedy best-first
  - A\*
  - Anytime A\*
- **Adversarial search**
  - Minimax with alpha-beta pruning

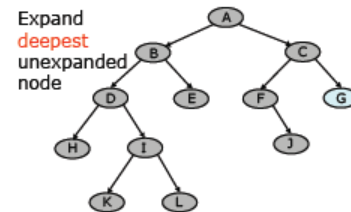
## Breadth-first search

- Expand all nodes at depth  $d$  before proceeding to depth  $d+1$



## Depth-first search

- Expand deepest unexpanded node



## Depth-limited search

Depth-first search with depth limit  $l$ .

Nodes at depth  $l$  have no successors.

Problem knowledge can be used.

Solves the “infinite path” problem

## Iterative deepening

A strategy to find best depth limit  $l$ .

Depth-Limited Search to depth 1, 2, ...

Expands from the root each time.

Combines benefits of DFS and BFS.

### Iterative deepening search $l = 0$

Limit = 0

### Iterative deepening search $l = 1$

Limit = 1

### Iterative deepening search $l = 2$

Limit = 2

### Iterative deepening search $l = 3$

Limit = 3

### Bi-directional search

- In some search problems we want to find the path from the initial state to the **unique goal state** (e.g. traveler problem)
- Bi-directional search idea:**

Initial state

Goal state

- Search both from the initial state and the goal state;
- Use inverse operators for the goal-initiated search.

CS 1571 Intro to AI M. Hauskrecht

From: <http://www.cs.pitt.edu/~milos/courses/cs1571-Fall06/lectures/Class6.pdf>

### Bi-directional search

Why bidirectional search? What is the benefit? Assume BFS.

- Cut the depth of the search space by half

Initial state

$d/2$

Goal state

$d/2$

$O(b^{d/2})$  Time and memory complexity

CS 1571 Intro to AI M. Hauskrecht

From: <http://www.cs.pitt.edu/~milos/courses/cs1571-Fall06/lectures/Class6.pdf>

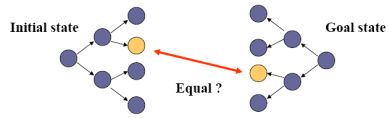
### Bi-directional search

Why bidirectional search? Assume BFS.

- It cuts the depth of the search tree by half.

What is necessary?

- Merge the solutions.



- How? The hash structure remembers the side of the tree the state was expanded first time. If the same state is reached from other side we have a solution.

CS 1571 Intro to AI

M. Hauskrecht

From: <http://www.cs.pitt.edu/~milos/courses/cs1571-Fall06/lectures/Class6.pdf>

Examples on the 8-puzzle

44

### Comparisons

- Breadth first:
  - Complete (will always find solution if it exists)
  - Optimal (will always find a least-cost solution)
  - High space complexity
- Depth-first:
  - Not complete
  - Not optimal
  - Space efficient
- Iterative deepening
  - Complete
  - Asymptotically optimal
  - Space efficient (depending on max depth)