Early AI & Robotics

Earlier robotic attempts in late 1960s were rooted in the logic-oriented approach of the formative years of Artificial Intelligence.

Logic-based AI (GOFAI-Good Old Fashioned AI)

Emphasis on symbol-manipulation problem solving techniques inspired by pioneering works in theoretical computer science by John von Neumann, Alan Touring and Claude Shannon. McCarthy's LISP language.
Applications: Symbolic algebra, theorem proving, mathematical discovery, diagnosis.

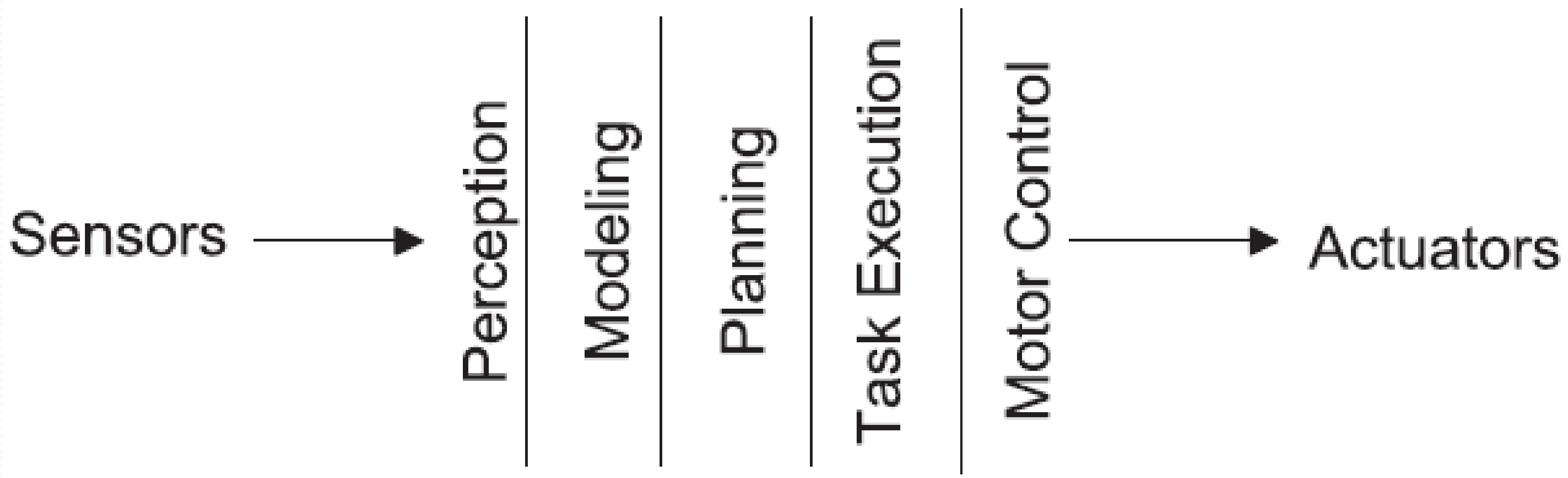
The Computational Theory of Mind (CTM)

Hilary Putnam, then Jerry Fodor: human cognition is implemented through a computational logic apparatus which represent mental states as set of symbols hence equating mental processes to inference mechanisms carried out by some form of computation upon these symbols

Why do we need this type of cognition in robots?

Sense-Model-Plan-Act

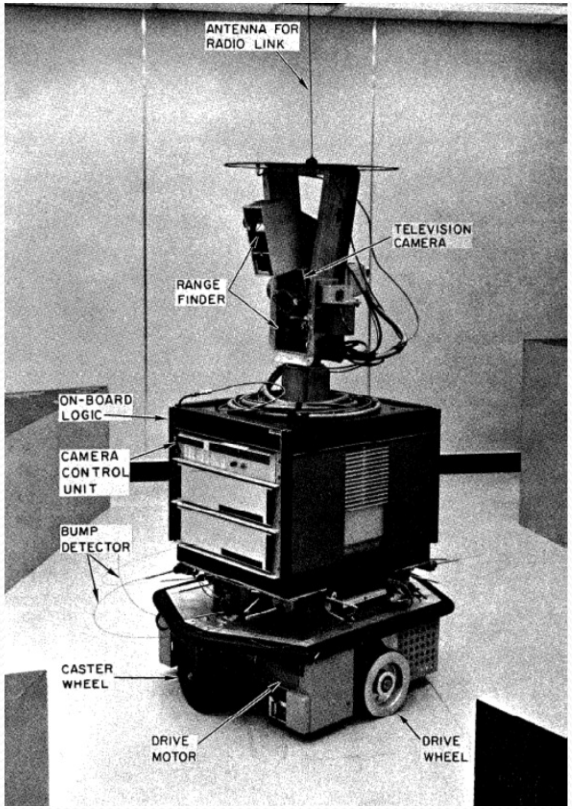
Top-Down Control Architectures



Example: Shakey the Robot

- Built at SRI, Late 1960's
- http://www.ai.sri.com/shakey/
- For robotics, the equivalent of Xerox PARC's Alto computer
  - Alto – mouse, GUI, network, laser printer, WYSIWYG, multiplayer computer game
  - Shakey – mobile, wireless, path-planning, Hough transform, camera vision, logical reasoning, English commands

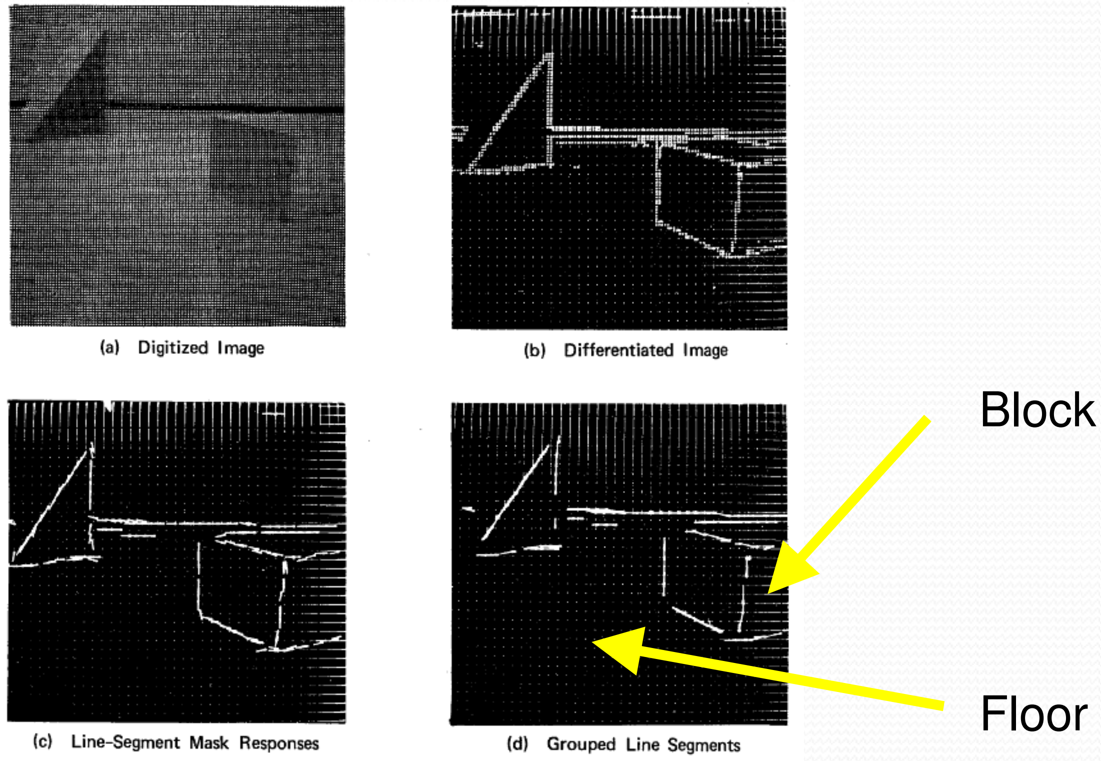
    (*put two square boxes together, put triangle box near square box*)



Nilsson, N.J. **A Mobile Automaton: An Application of Artificial Intelligence Techniques**, Technical Note 40. AI Center, SRI International, CA 94025, Mar 1969.

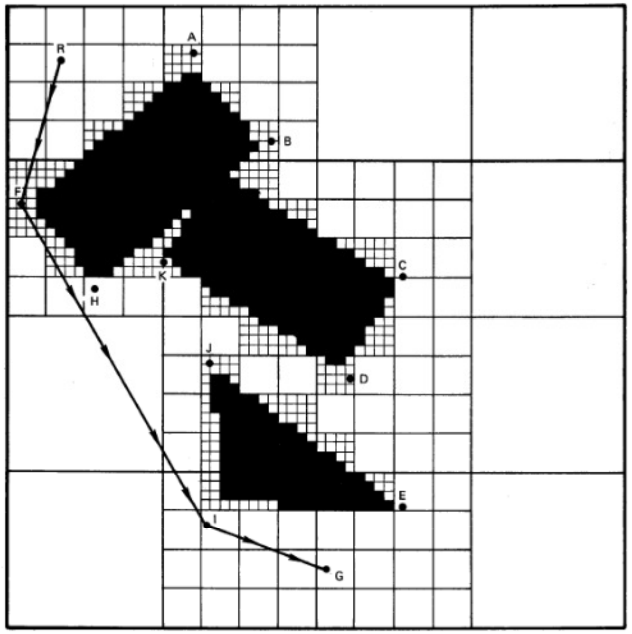
Fikes, R.E. and Nilsson, N.J. **STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving**, Technical Note 43r. AI Center, SRI International,May 1971.

Sensing & Perception



L. G. Roberts, “Machine Perception of Three-Dimensional Solids”, Optical and Electro-Optical Information Processing (MIT Press, 1965)

Modelling



Grid Map used to represent obstacles and free or unknown space
Cell(i,j)ϵ{empty, full, unknown)

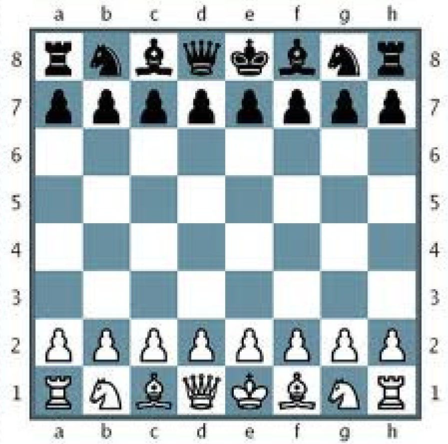
World Model

AT(BlockA, pos1, s)
AT(BlockB, pos2, s)
AT(BlockC, pos3, s)
Shape(BlockA, rectangle)
Shape(BlockB, rectangle)
Shape(BlockC, triangle)

Predicate Logic used to summarise the perceived situation

Planning

Simon and Amarel, Herbert Simon, 1966:



- Given:
  - A way to describe the world
  - An initial state of the world
  - A goal description
  - A set of possible actions to change the world

- Find:
  - *sequence of admissible actions that successively transform the world into a desired world-state where the goal is satisfied*

Applications

- **Mobile robots**
  - An initial motivator, and still being developed
- Simulated environments
  - Goal-directed agents for training or games
- Web and grid environments
  - Composing queries or services
  - Workflows on a computational grid
- Managing crisis situations
  - E.g. oil-spill, forest fires, urban evacuation, in factories, …
- And many more
  - Factory automation, flying autonomous spacecraft, playing bridge, military planning, …

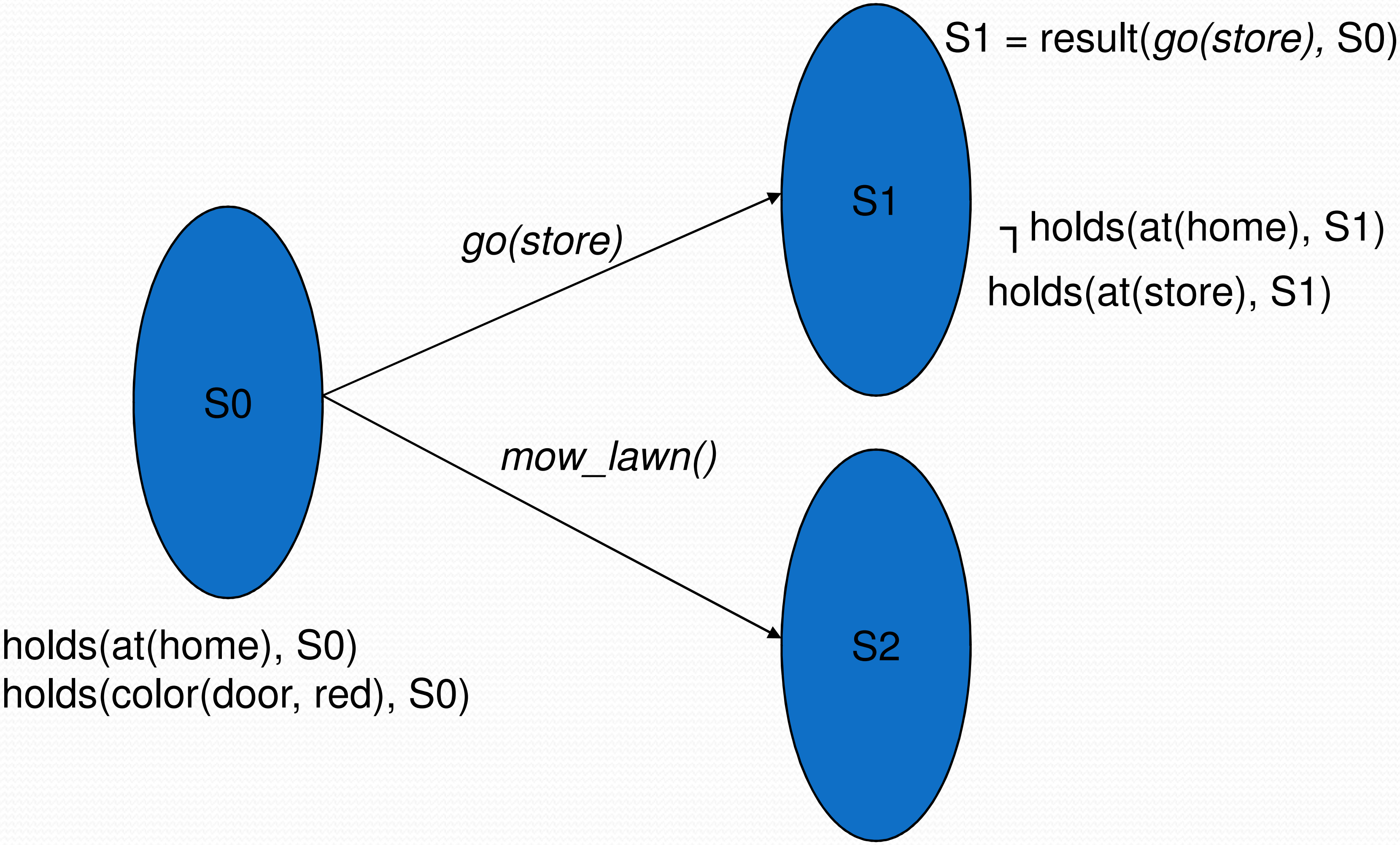
Representing change

- As actions change the world OR we consider possible actions, we want to:
  - Know how an action will alter the world
  - Keep track of the history of world states (have we been here before?)
  - Answer questions about potential world states (what would happen if..?)

The situation calculus (McCarthy 63)

- Key idea: represent a snapshot of the world, called a 'situation' explicitly.
- 'Fluents' are statements that are true or false in any given situation, e.g. 'I am at home'
- Actions map situations to situations.

Example


S1 = result(*go(store),* S0)
S1
¬ holds(at(home), S1)
holds(at(store), S1)
go(store)
S0
mow_lawn()
S2
holds(at(home), S0)
holds(color(door, red), S0)

Frame problem

- I go from home to the store, creating a new situation S’. In S’:
  - My friend is still at home
  - The store still sells chips
  - My age is still the same
  - Los Angeles is still the largest city in California…
- How can we efficiently represent everything that hasn’t changed?

Successor state axioms

- Normally, things stay true from one state to the next -- unless an action changes them:

holds(at(X),result(A,S)) iff A = go(X)
or [holds(at(X),S) and A != go(Y)]

- We need one or more of these for every fluent.

- Now we can use theorem proving to deduce a plan.

Well, not quite..

- Theorem proving can be really inefficient for planning
- How do we handle concurrent events? uncertainty? metric time? preferences about plans? …

Strips (Fikes and Nilsson 71)

- For efficiency, separates theorem-proving within a world state from searching the space of possible states

- Highly influential representation for actions:
  - Preconditions (list of propositions to be true)
  - Delete list (list of propositions that will *become* false)
  - Add list (list of propositions that will *become* true)

Example problem:

Initial state: at(home), ¬ have(beer), ¬ have(chips)

Goal: have(beer), have(chips), at(home)

Actions:

Buy (X):

Pre: at(store)

Add: have(X)

Go (X, Y):

Pre: at(X)

Del: at(X)

Add: at(Y)

Frame problem (again)

- I go from home to the store, creating a new situation S’. In S’:
  - The store still sells chips
  - My age is still the same
  - Dublin is still the largest city in Ireland…
- How can we efficiently represent everything that hasn’t changed?

Ramification problem

- I go from home to the store, creating a new situation S'. In S':
  - I am now in Ranelagh
  - The number of people in the store went up by 1
  - The contents of my pockets are now in the store..
- Do we want to say all that in the action definition?

Solutions to the ramification problem

- In Strips, some facts are inferred within a world state,
  - e.g. the number of people in the store
- ‘primitive’ facts, e.g. at(home) persist between states unless changed. ‘inferred’ facts are not carried over and must be re-inferred.
  - Avoids making mistakes, perhaps inefficient.

Questions about Strips

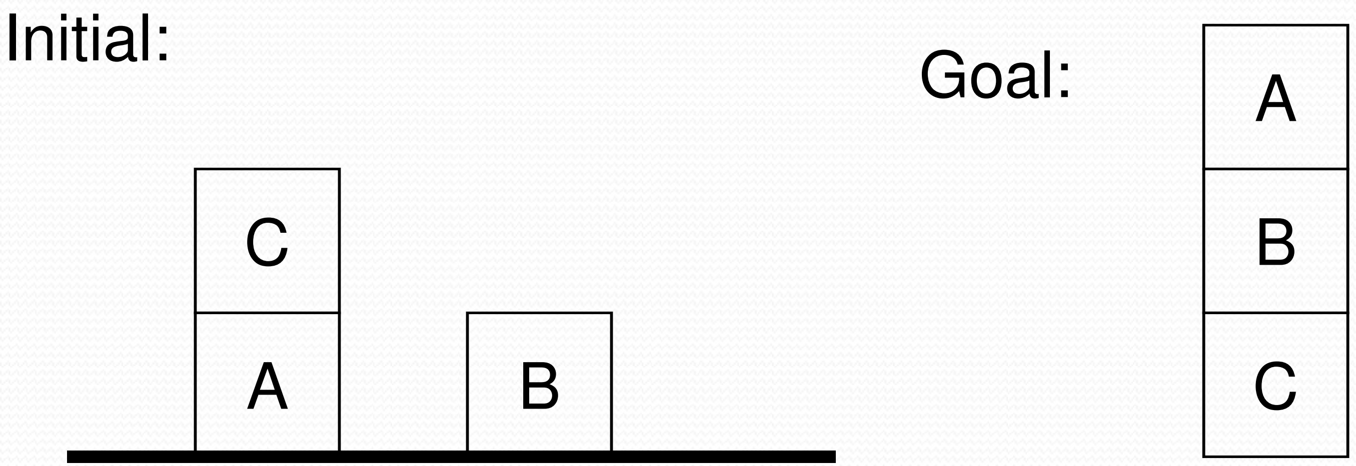
- What would happen if the order of goals was
  at(home), have(beer), have(chips) ?

- When Strips returns a plan, is it always correct? efficient?[]

- Can Strips always find a plan if there is one?

- Deciding the existence of a plan for a propositional STRIPS instance is PSPACE-complete. Various restrictions can be enforced on the instances to make the problem NP-complete[1

Example: blocks world



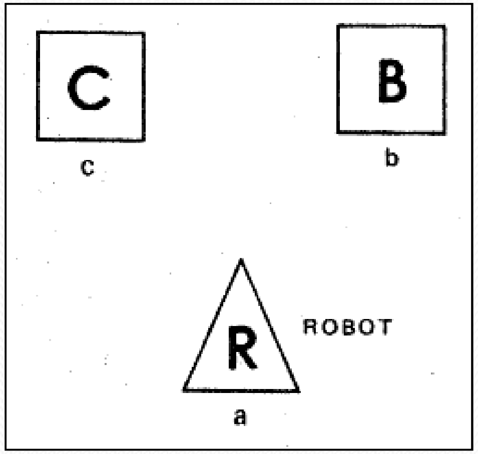
S_0 : onTable(A), on(C, A), onTable(B), clear(B), clear(C)

G = on(A, B) Λ on(B, C), G1= on(A, B), G2 =on(B, C)

STRIPS is a **non-interleaved planner:** when given two subgoals G_1 and G_2, produces either a plan for G_1 concatenated with a plan for G_2, or vice versa.

-> run into problems when sub-goals are not independent

Strips in Shakey



Initial Situation:

$\mathbf{S_0}$: RobotAT(a, S_0), AT(C, c), AT(B,b)

Vu,x,y,s (AT(u,x,s) /\ x≠y) → ~AT(u,y,s))

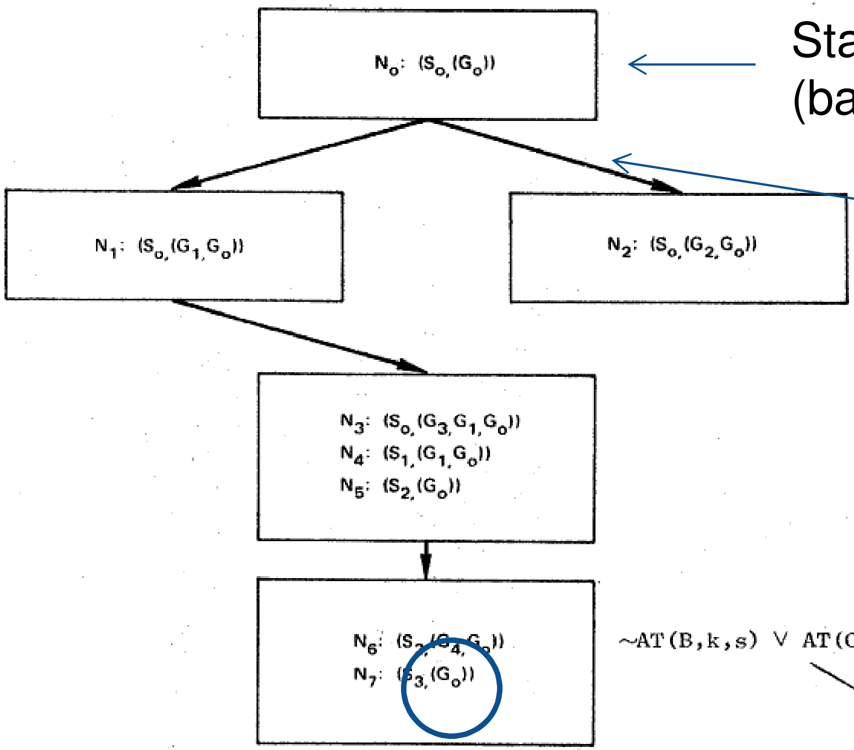
Problem: achieve a configuration in which *object B is at place k and in which object C is not in place c*, G_0 :∃s [AT(B,k,s) Λ~AT(C,c,s)]

Operators:

Goto(x,y):
Pre-condition: ∃ x,s: RobotAT(x,s)
Delete-List : RobotAT(x,s)
Add-List : RobotAT(y, goto'(x,s))

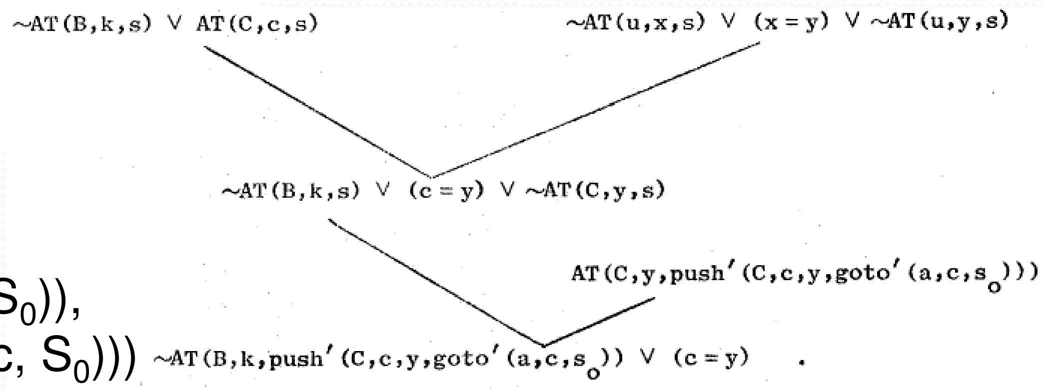
Push(u,x,y):
Pre-condition: ∃u,x,s [AT(u,x,s)Λ~RobotAT(x,s)]
Delete-List : RobotAT(x,s), At(u,x,s)
Add-List : RobotAT(y, push'(u,x,y,s)), AT(u, y, push'(u, x,y,s))

Strips in Shakey



Start from goal as root (backward chaining)

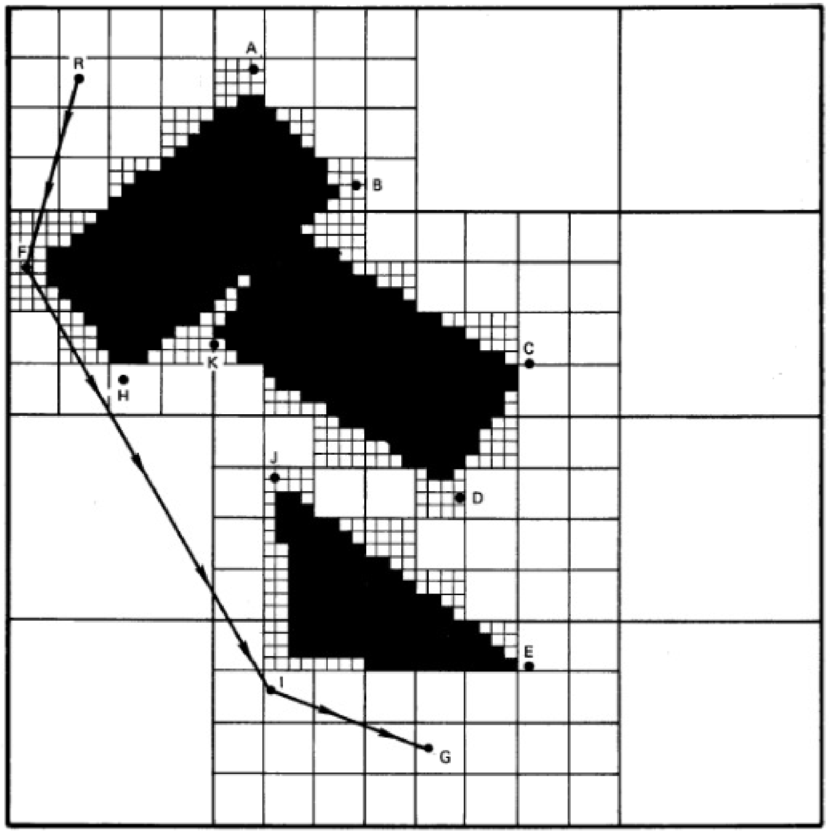
Graph Search,
Node expansion & Resolution



Plan: goto'(a,c, S_0),
push(B,c,a, goto'(a,c, S_0)),
goto(a,b,push(goto'(a,c, S_0)))
...

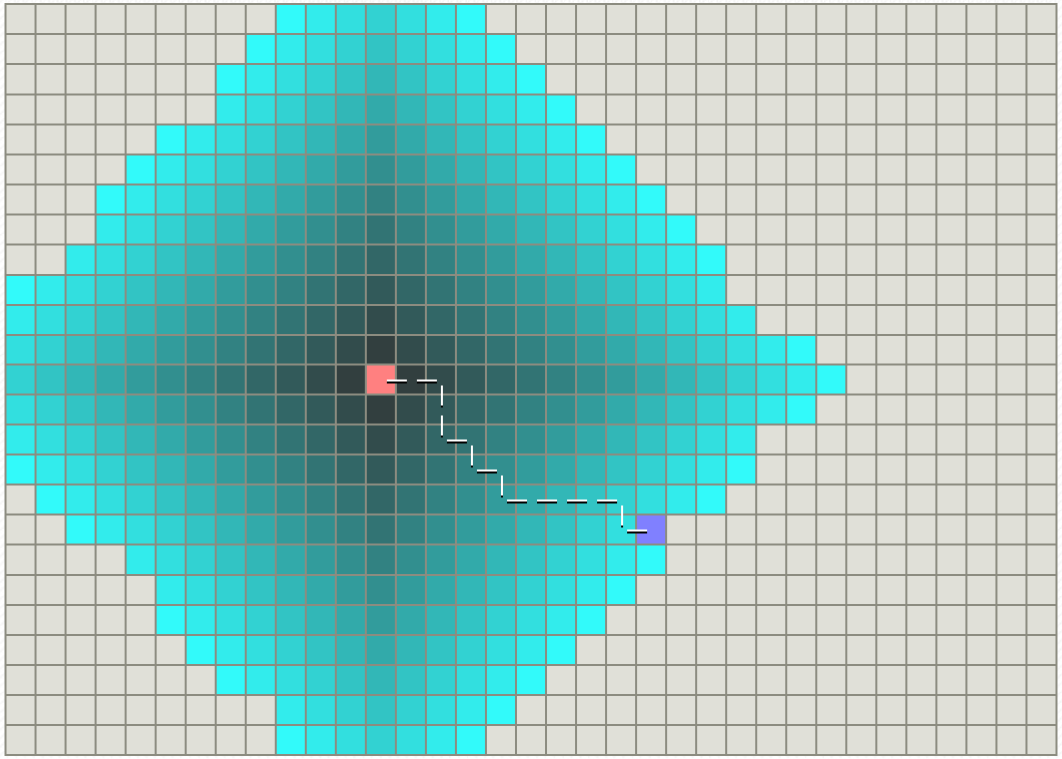
Path planning

- Compute nodes at corners of objects
- Find shortest path through nodes – A*
- Execute paths by using low-level operators, i.e.. *move-forward*, *turn…*



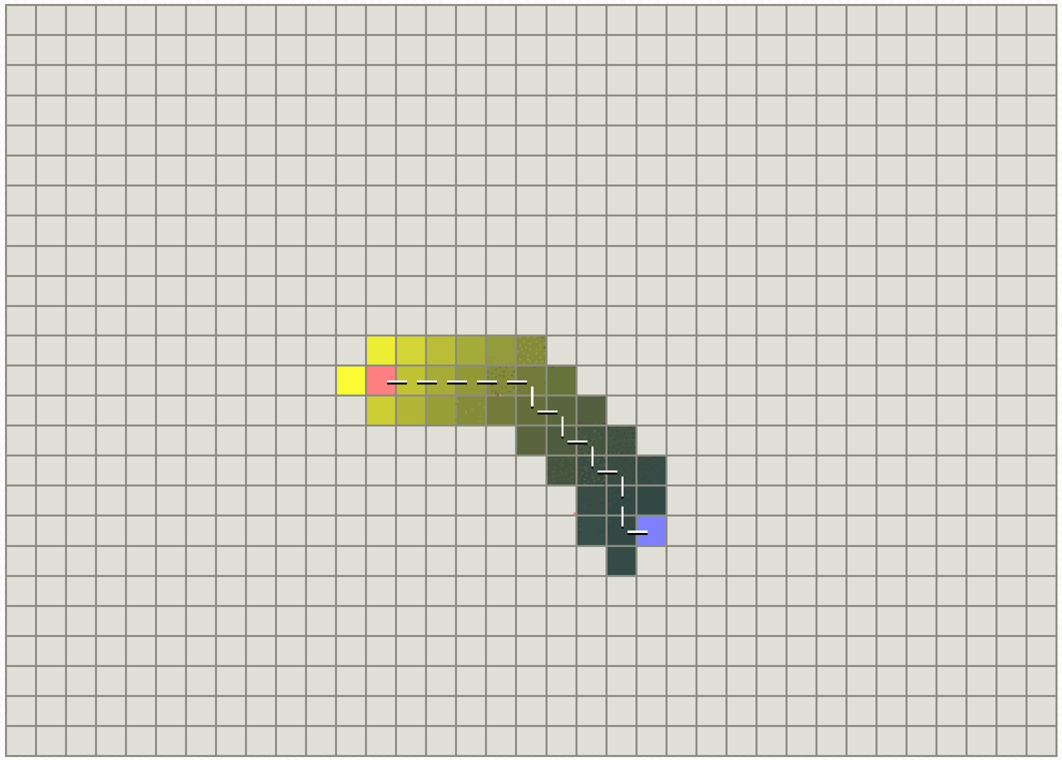
Finding paths

- Dijkstra's Algorithm
  - Breadth-first
  - Will find a shortest path
  - May have to examine many nodes!

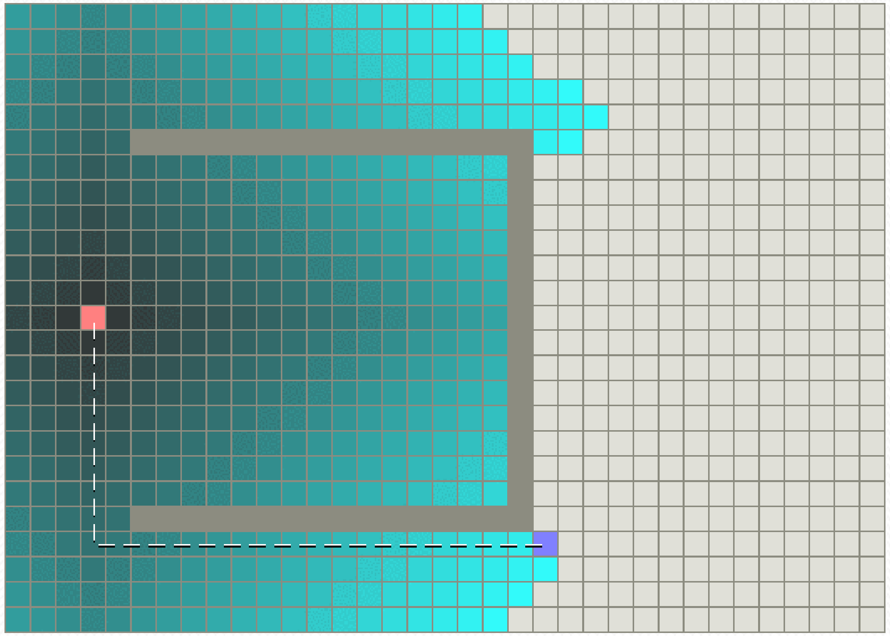


Best-first

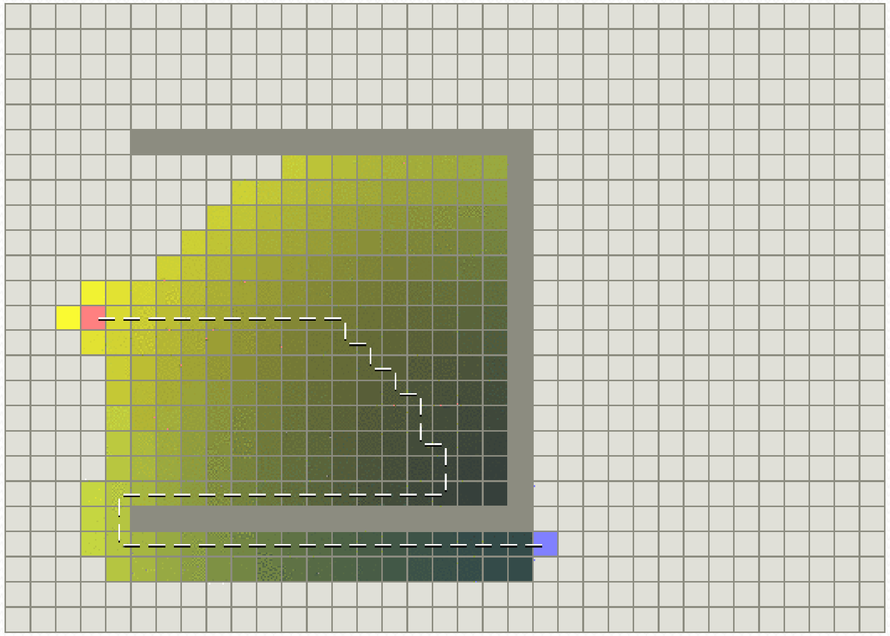
- Uses a heuristic for cost to goal
- Greedy algorithm
  - May not find a shortest path



What about obstacles?



Breadth-first

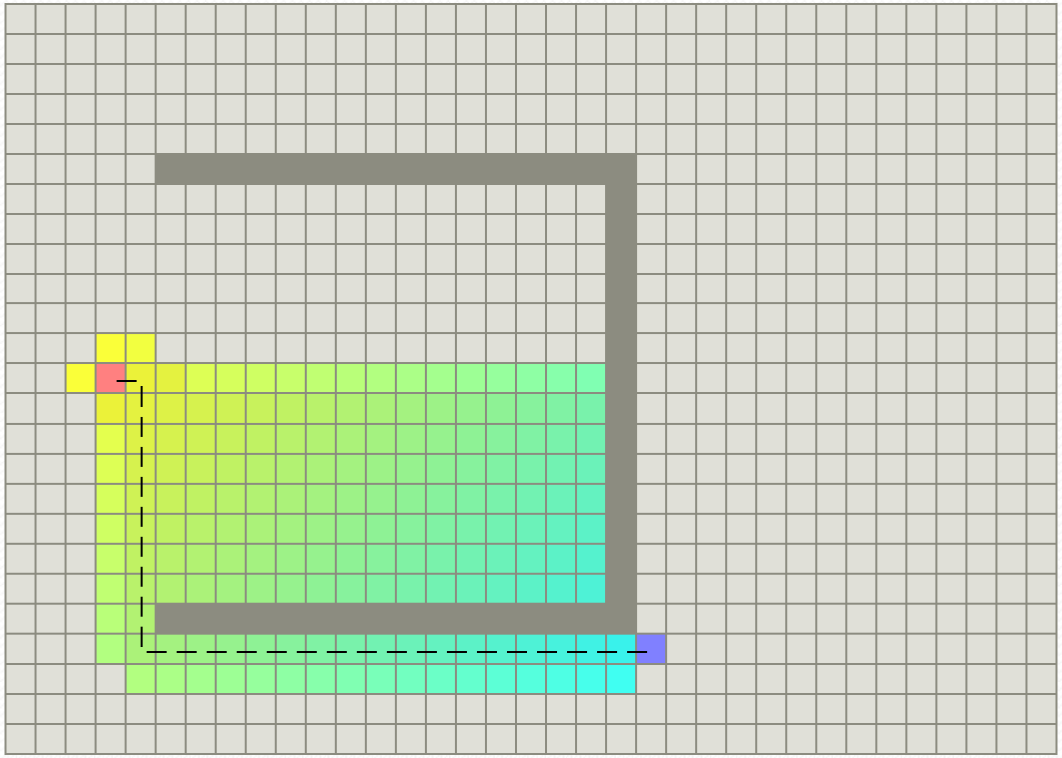


Best-first

A*

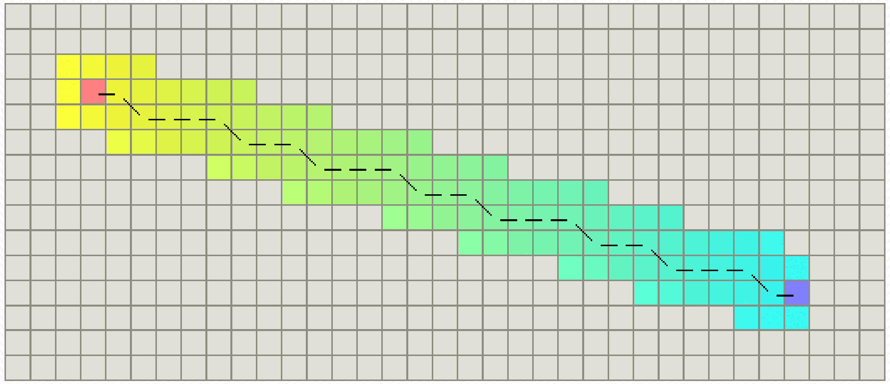
- A* combines the properties of each
- G(n) = cost of path from start to n
- H(n) = estimated cost from n to goal
    - Might just be straight-line distance, nothing magic

A* and obstacles

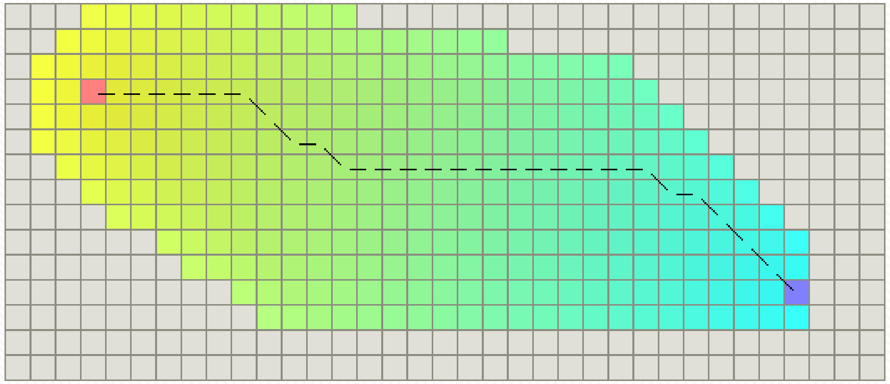


How to choose a heuristic

- H(n) = 0
  - Reverts to breadth-first search
- If H(n) < true cost
  - A* will find a shortest path
- If H(n) >> true cost
  - Acts like best-first search



H(n) = diagonal distance



H(n) = Euclidean Distance

Shakey used many good ideas

- Plan Monitor and Plan Repair (Planex-STRIPS)
- Grid Representation
- A*
- Putting sub-goals on corners of vertices
  - This has been generalized into the idea of visibility graphs.
  - …

Problems of SMPA

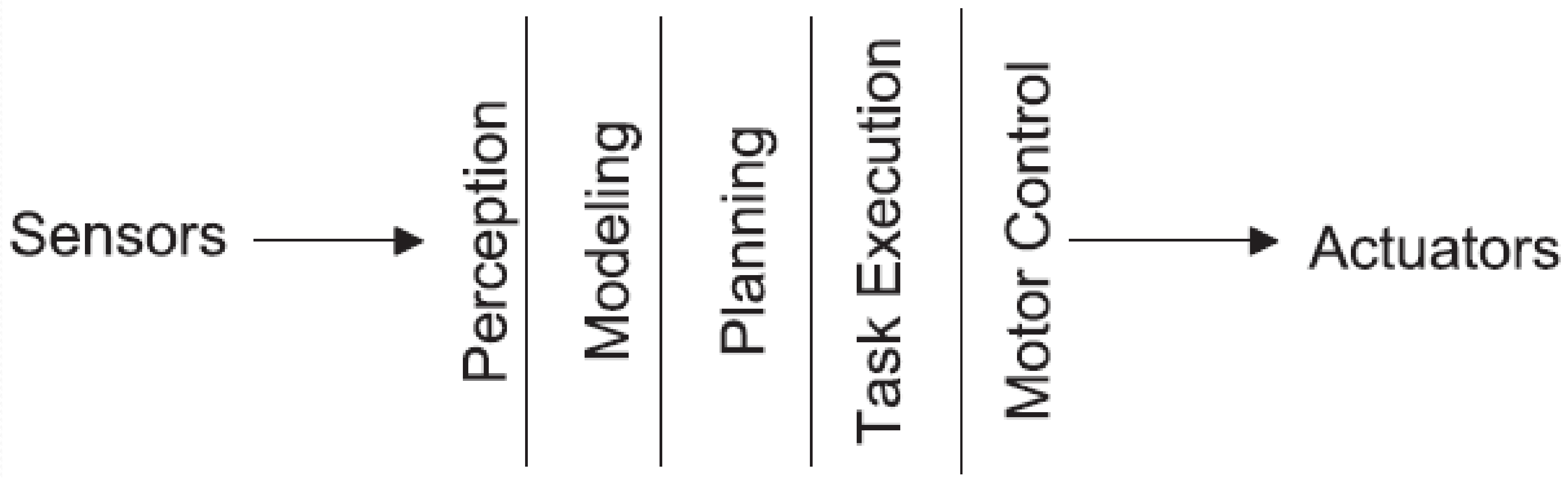
- Perception
  - Inadequacies of robotic sensors and processing techniques
  - Inherent difficulties in the physical sensing process – uncertainty and noise in sensor data, limited view
- Modelling
  - World Model must be maintained in synch with the real World!
- Planning
  - Slow! What if the world has changed?
  - Hard to consider other agents, exogenous events, and undeterministic domains

Problems of SMPA

- Perception
  - Inadequacies of robotic sensors and processing techniques
  - Inherent difficulties in the physical sensing process – uncertainty and noise in sensor data, limited view
- Modelling
  - World Model must be maintained in synch with the real World!
- Planning
  - Slow! What if the world has changed?
  - Hard to consider other agents, exogenous events, and undeterministic domains

Not good for Dynamic Environments !!

How about modularity?



Readings

Rodney Brooks
http://people.csail.mit.edu/brooks/publications.html

- "New Approaches to Robotics", Science (253), September 1991, pp. 1227–1232.
- "Intelligence Without Representation", Artificial Intelligence Journal (47), 1991, pp. 139–159
- "Elephants Don't Play Chess", Robotics and Autonomous Systems (6), 1990, pp. 3–15.

We'll discuss these papers on our next lecture, which will cover Behaviour Based Control Architectures