Robot Learning

15-494 Cognitive Robotics
David S. Touretzky &
Ethan Tira-Thompson

Carnegie Mellon
Spring 2009

What Can Robots Learn?

- Parameter tuning, e.g., for a faster walk
- Perceptual learning: ALVINN driving the Navlab
- Map learning, e.g., SLAM algorithms
- Behavior learning; plans and macro-operators
  - Shakey the Robot (SRI)
  - Robo-Soar
- Learning from human teachers
  - Operant conditioning: Skinnerbots
  - Imitation learning

Lots of Work on Robot Learning

- IEEE Robotics and Automation Society
  - Technical Committee on Robot Learning
  - http://www.learning-robots.de
- Robot Learning Summer School
  - Lisbon, Portugal; July 20-24, 2009
- Workshops at major robotics conferences
  - ICRA 2009 workshop: Approachss to Sensorimotor Learning on Humanoid Robots
  - Kobe, Japan; May 17, 2009

Parameter Optimization

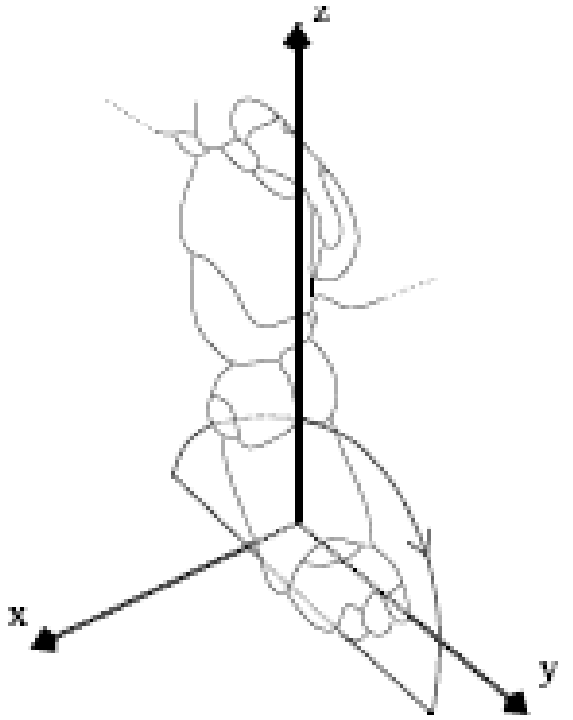
- How fast can an AIBO walk? Figures from Kohl & Stone, ICRA 2004, for the ERS-210 model:

– CMU (2002)	200 mm/s	Hand-tuned gaits
– German Team	230 mm/s	
– UT Austin Villa	245 mm/s	
– UNSW	254 mm/s	
– Hornsby (1999)	170 mm/s	Learned gaits
– UNSW	270 mm/s	
– UT Austin Villa	291 mm/s	

Walk Parameters

12 parameters to optimize:

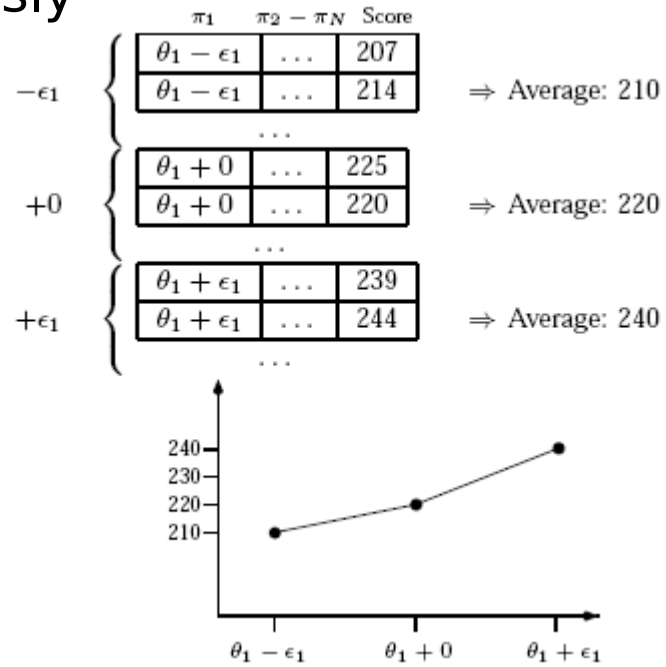
- Front locus (height, x pos, ypos)
- Rear locus (height, x pos, y pos)
- Locus length
- Locus skew multiplier
  (in the x-y plane, for turning)
- Height of front of body
- Height of rear of body
- Foot travel time
- Fraction of time foot is on ground



From Kohl & Stone (ICRA 2004)

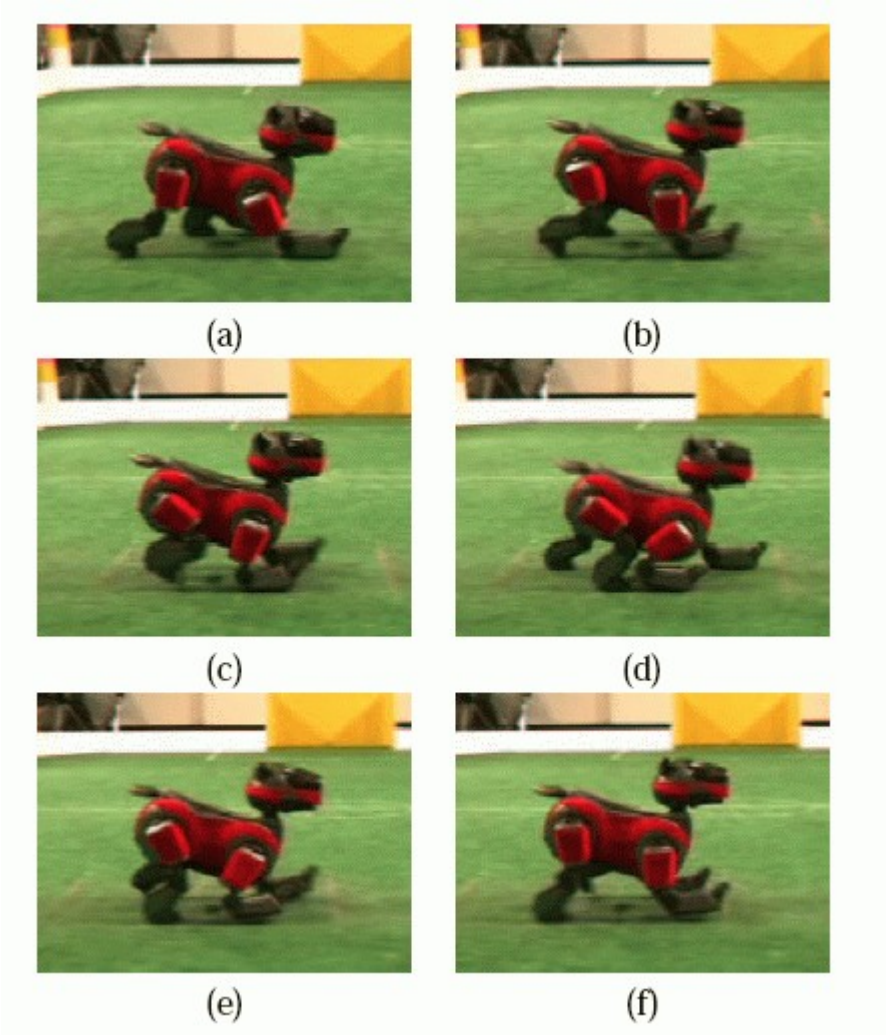
Optimization Strategy

- "Policy gradient reinforcement learning":
  - Walk parameter assignment = "policy"
  - Estimate the gradient along each dimension by trying combinations of slight perturbations in all parameters
  - Measure walking speed on the actual robot
  - Optimize all 12 parameters simultaneously
  - Adjust parameters according to the estimated gradient.



Kohl & Stone Results

- Used three robots
  - 23 iterations
  - 1000 total runs
  - elapsed time 3 hours
- Final speed 291 mm/s: faster than any other AIBO walk
- Videos: initial walk (clumsy 150 mm/s), final walk



Kohl & Stone Results

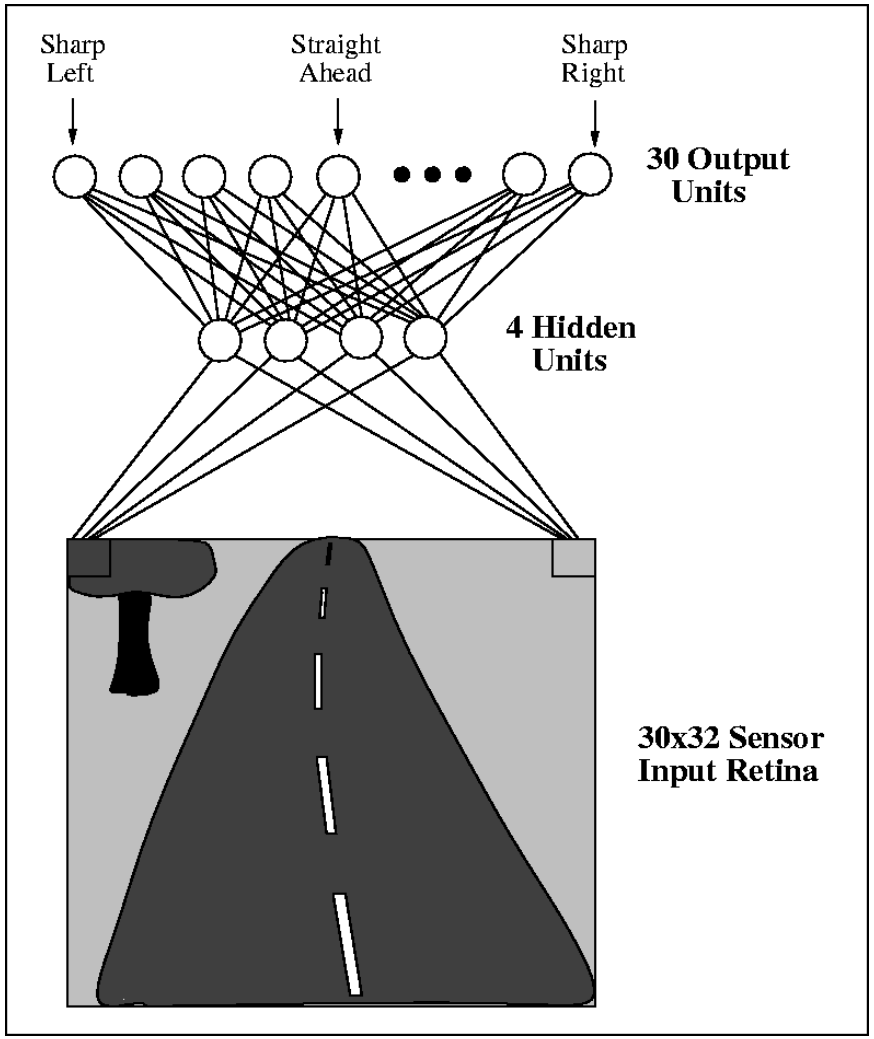
- Initial result was for optimizing walking speed alone.
- But stability is also important:
  - Bouncy walk makes vision hard
- Later experiments optimized for a combination of speed and stability.
- Also applied the technique to the ERS-7

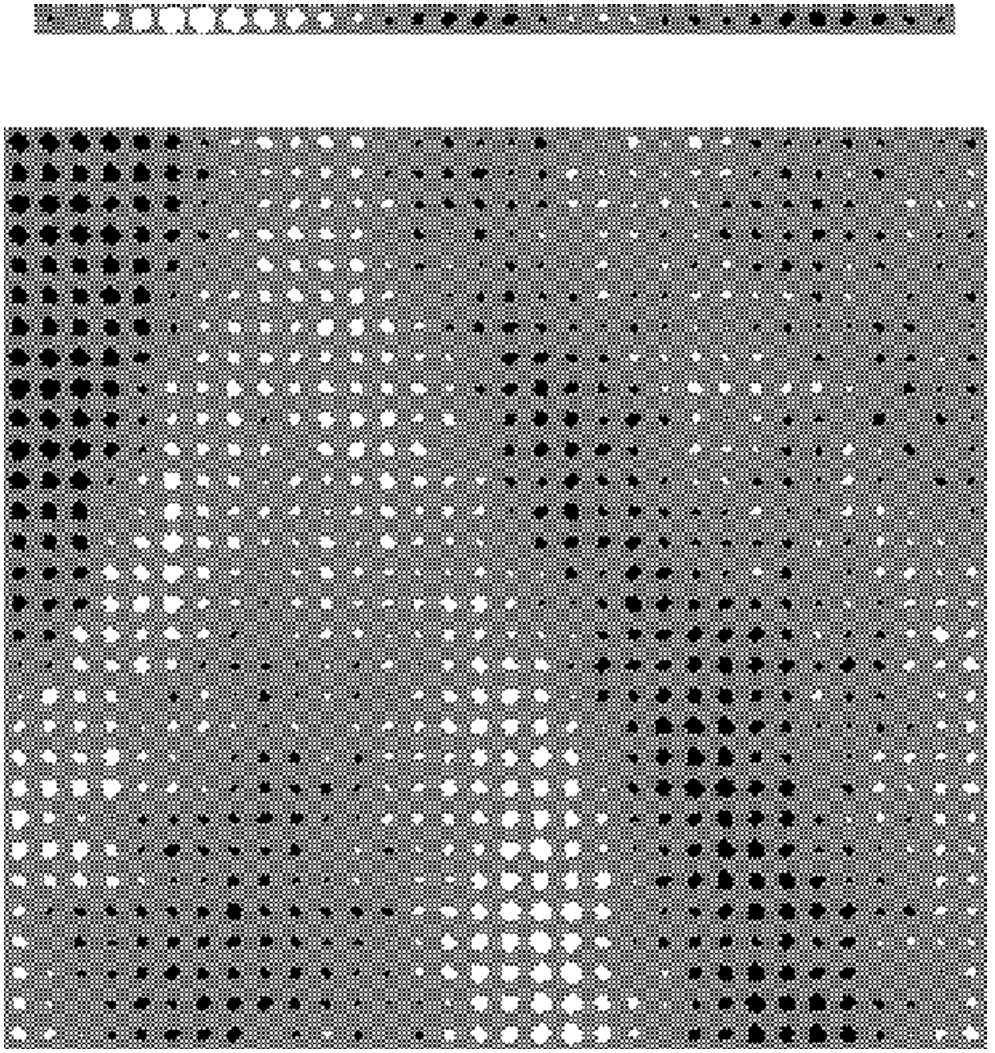
(videos)

Perceptual Learning

- ALVINN (Autonomous Land Vehicle in a Neural Network) learns to drive the Navlab

ALVINN


Sharp
Left
Straight
Ahead
Sharp
Right
30 Output
Units
4 Hidden
Units
30x32 Sensor
Input Retina



ALVINN Training

- Watched a human drive for a few minutes
- Used clever techniques to expand the training set
- Maintained a pool of 200 training images
- Trained on the fly

168

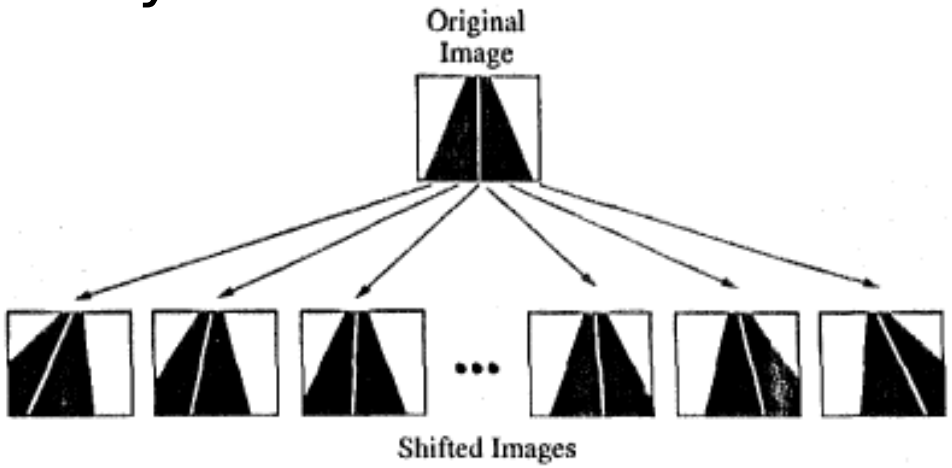
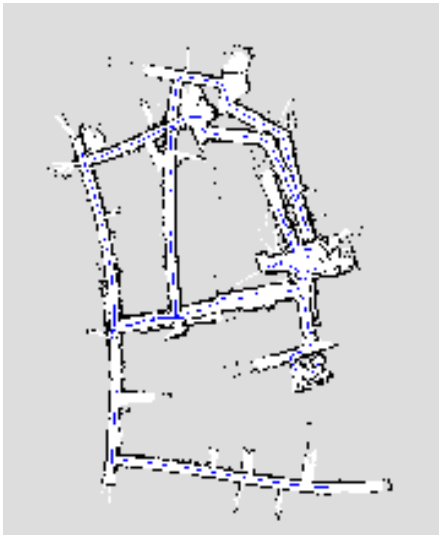


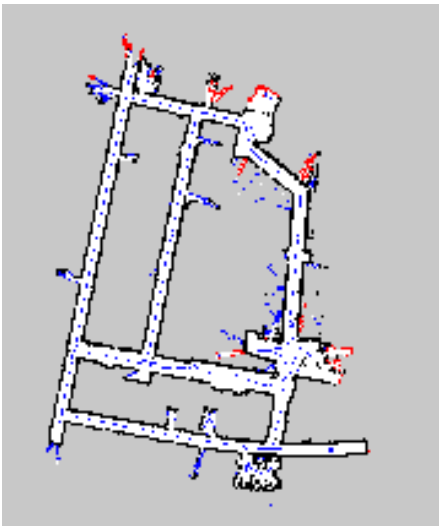
Figure 5.3.6
Shifted video images from a single original video image used to enrich the training set used to train ALVINN. (From D. A. Pomerleau, 1991, with permission of the MIT Press.)

Map Learning

- Lots of work on learning maps
  - from sonar data
  - from laser rangefinder data
  - using visual landmarks



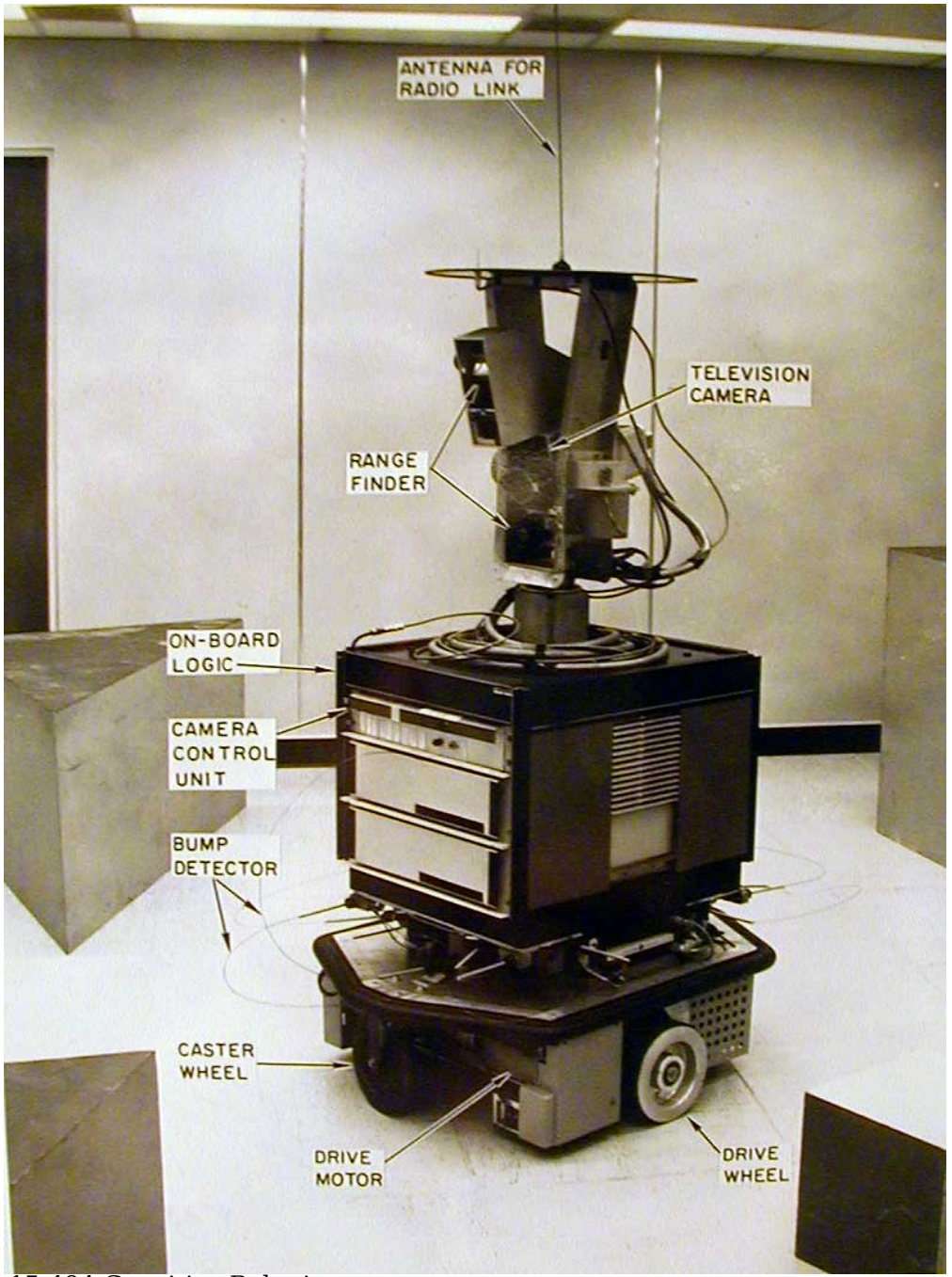
- Dieter Fox et al., particle filters for map learning



- SLAM: Simultaneous Localization and Mapping
  - many algorithms, all based on probabilistic approaches like particle filters

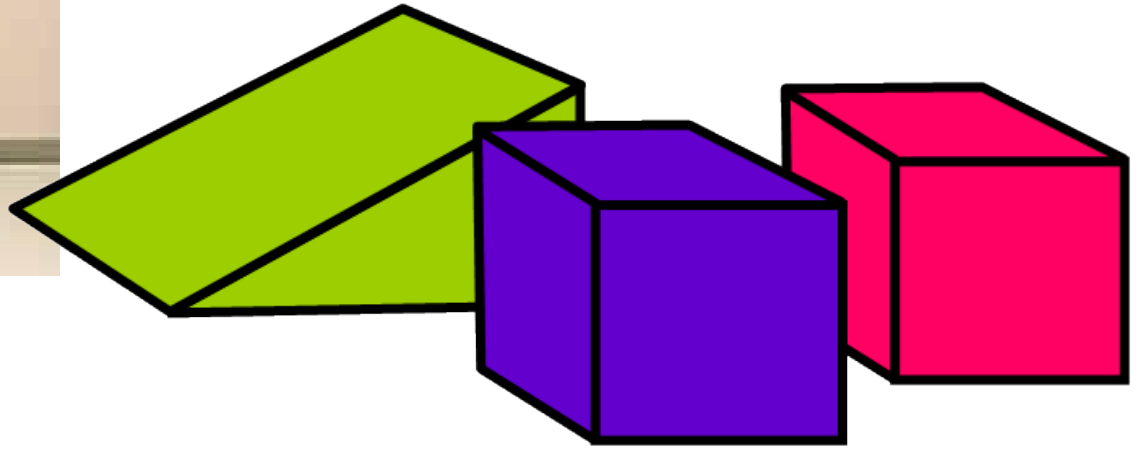
Shakey the Robot

- SRI International, 1968-1972
- Remote controlled by a PDP-10
- Programmed in Lisp and a theorem proving planner called STRIPS



Learning Blocks World Plans

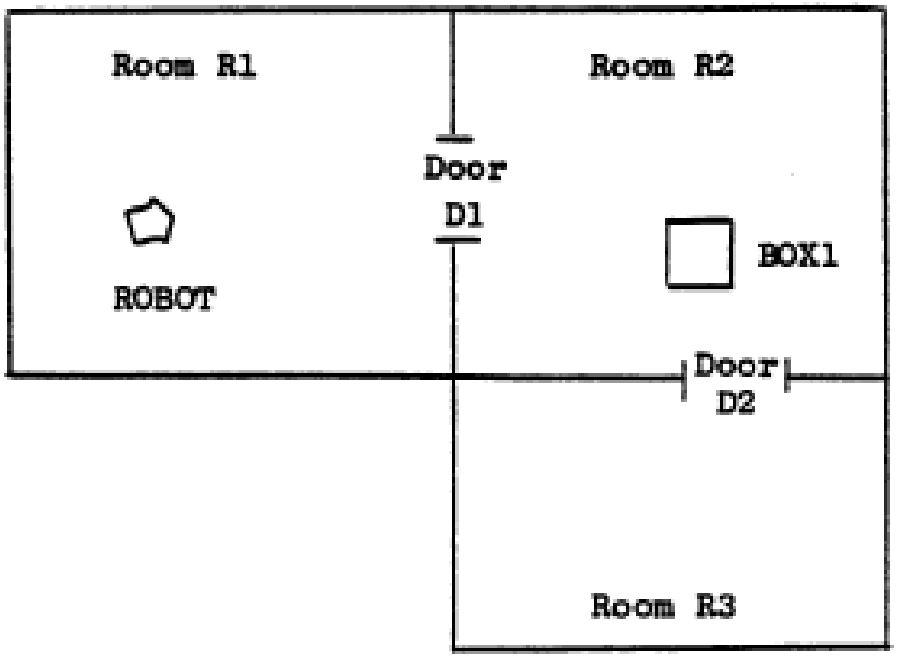
- Shakey learned plans for manipulating blocks by pushing them around.



Sample Problem

Initial model:

INROOM(robot,ROOM1)
CONNECTS(DOOR1,ROOM1,ROOM2)
CONNECTS(DOOR2,ROOM2,ROOM3)
BOX(BOX1)
INROOM(BOX1,ROOM2)
$(\forall x \forall y)$ [CONNECTS(d,x,y) $\rightarrow$ CONNECTS(d,y,x)]



Goal:

$(\exists x)$[BOX(x) & INROOM(x,ROOM1)]

Available operators:

GOTHRU(d,r1,r2)
PUSHTHRU(b,d,r1,r2)

GOTHRU Preconditions; Add and Delete Lists

- Operator: GOTHRU(d,r1,r2)
- Precondition:
  INROOM(ROBOT,r1) & CONNECTS(d,r1,r2)
- Delete list: INROOM(ROBOT, $)
- Add list: INROOM(ROBOT, r2)

PUSHTHRU Preconditions; Add and Delete Lists

- Operator: PUSHTHRU(b,d,r1,r2)
- Precondition:
  INROOM(b, r1) & INROOM(ROBOT,r1)
  & CONNECTS(d,r1,r2)
- Delete list: INROOM(ROBOT, $), INROOM(b, $)
- Add list: INROOM(ROBOT, r2), INROOM(b, r2)

Solving the Problem

- Goal G0: (∃x) [BOX(x) & INROOM(x,ROOM1)]
  - Not satisfied in current model
  - But PUSHTHRU could make INROOM(BOX1,ROOM1) true
- Precondition for PUSHTHRU gives subgoal G1:
  INROOM(BOX1,r1) & INROOM(ROBOT,r1)
  & CONNECTS(d,r1,ROOM1)
  - Not satisfied in current model
  - But if r1=ROOM2 and d=DOOR1, the operator could apply
  - Need INROOM(ROBOT,ROOM2)
- Precondition for GOTHRU gives subgoal G2:
  INROOM(ROBOT,r1) & CONNECTS(d,r1,ROOM2)

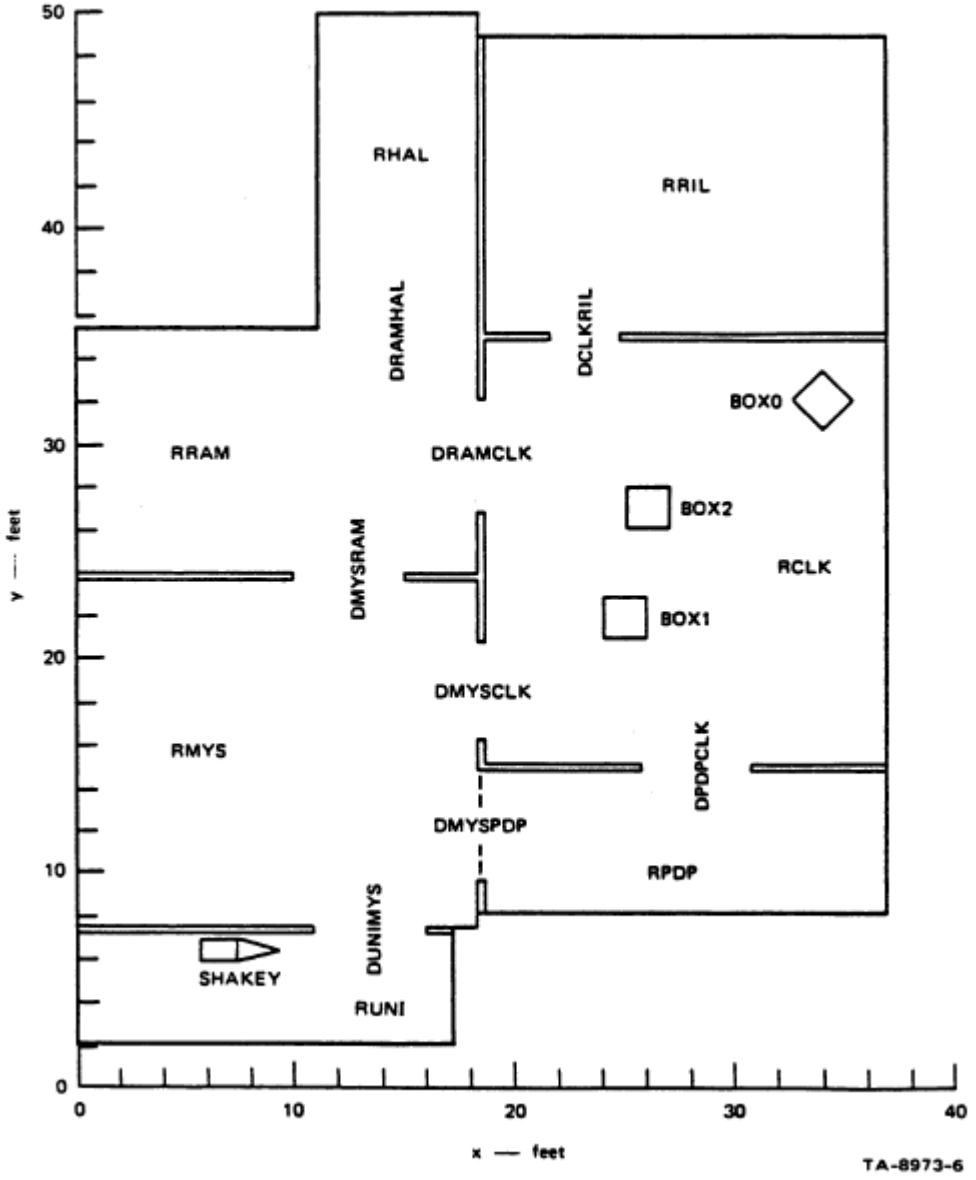
Solving the Problem

- Apply operator GOTHRU(DOOR1,ROOM1,ROOM2)
- New model M1:

```
INROOM(ROBOT,ROOM2)
CONNECTS(DOOR1,ROOM1,ROOM2)
CONNECTS(DOOR2,ROOM2,ROOM3)
BOX(BOX1)
INROOM(BOX1,ROOM2)
```

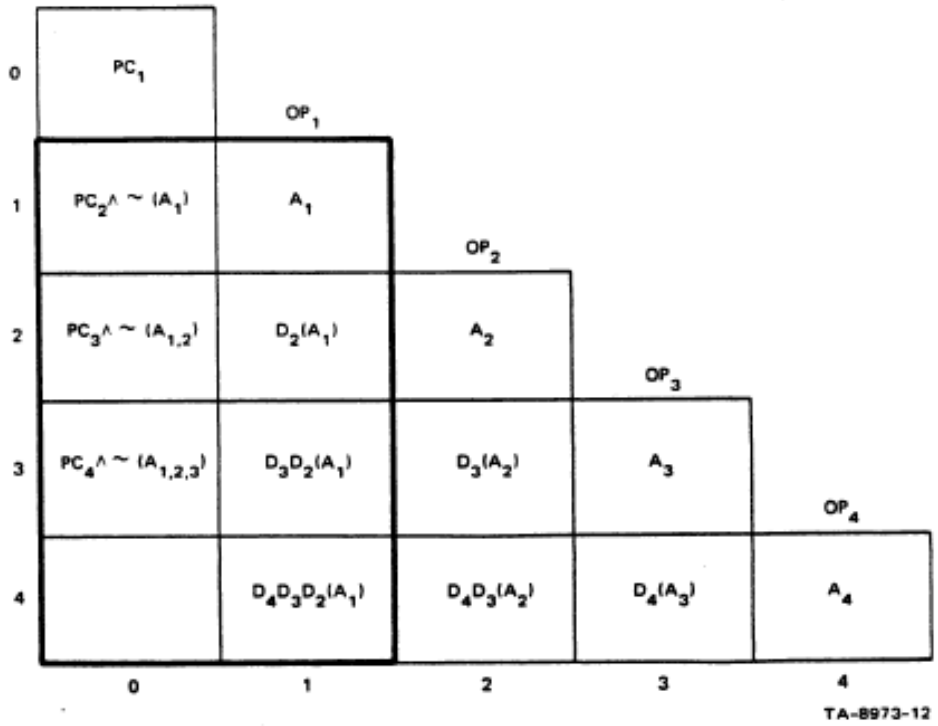
- Apply operator PUSHTHRU(BOX1,DOOR1,ROOM2,ROOM1)
- Goal G0 has now been achieved.

Shakey's Full Environment



Macro Operators (MACROPs)

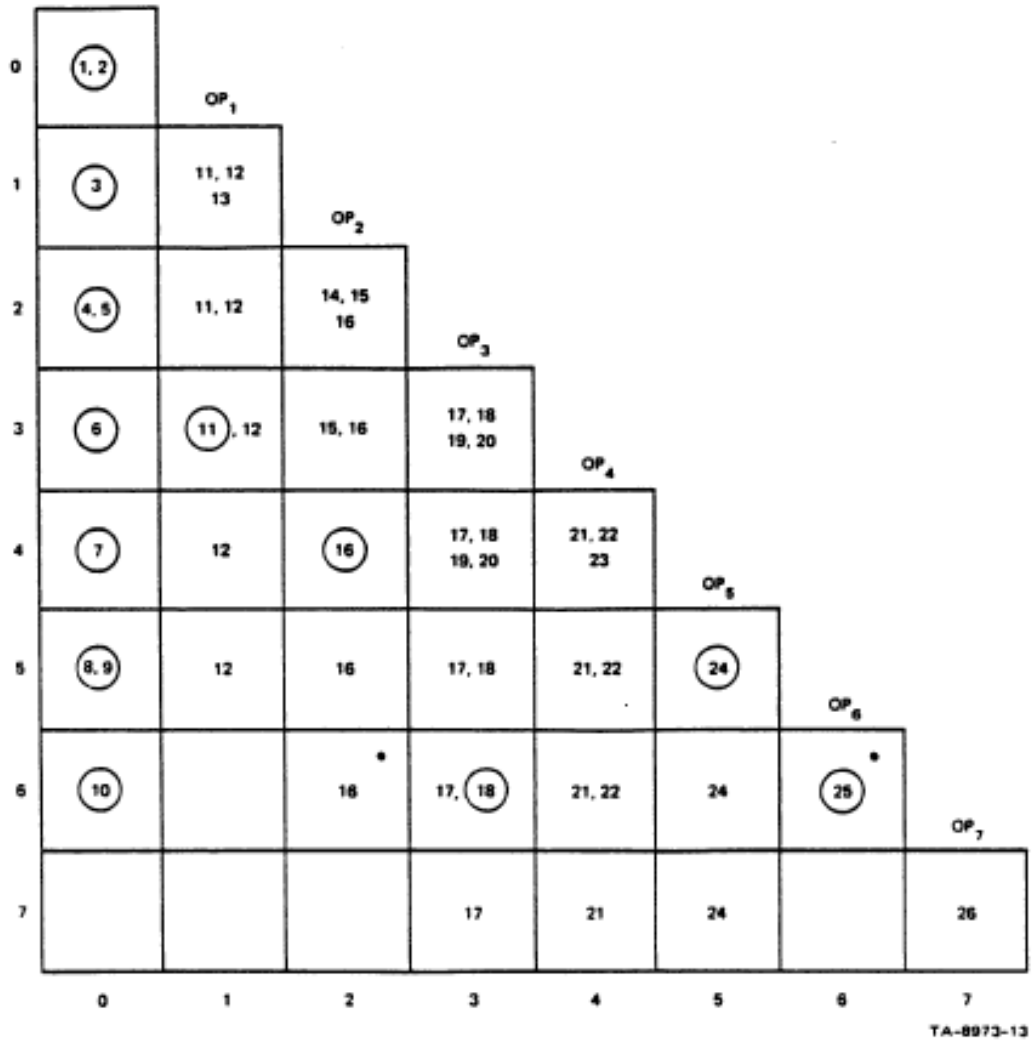
- Idea: extract general plans from solutions to specific problems.
- Reuse those plans in novel contexts by replacing constants with variables and/or deleting irrelevant steps.
- "Triangle table" defines additions/deletions for each operator in a plan.
- Reasoning algorithm determines which clauses are relevant to the new plan.



Marked Clauses

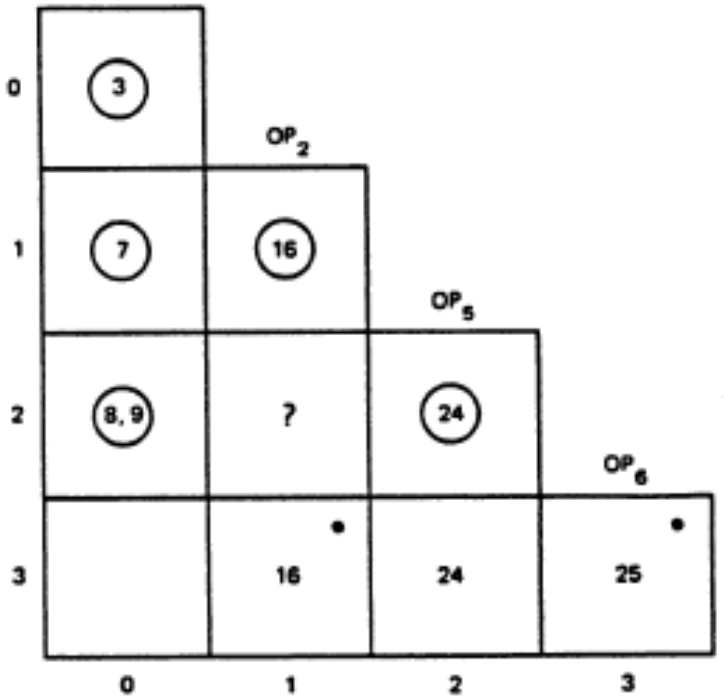
- Clauses are “marked” if needed to prove the precondition of the operator in that row.
- Unneeded clauses are deleted.

Operator	Precondition Support Supplied By	Precondition Support Supplied To
OP_1	I	OP_4
OP_2	I	OP_5
OP_3	I	OP_7, F
OP_4	I,OP_1	F
OP_5	I,OP_2	OP_6, F
OP_6	I,OP_5	OP_7
OP_7	I,OP_3,OP_6	F



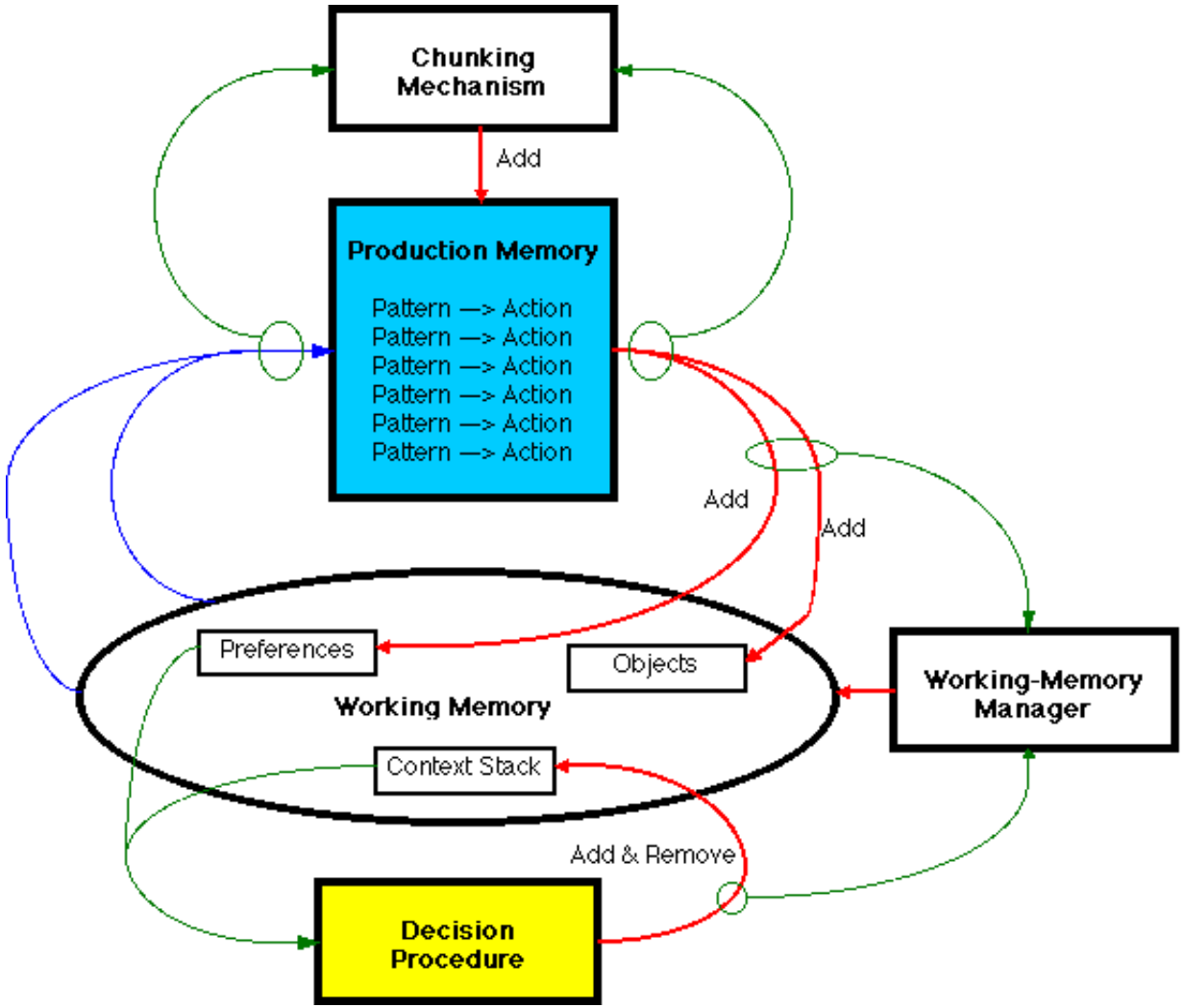
Resulting MACROP

Operator	Precondition Support Supplied By	Precondition Support Supplied To
OP_2	I	OP_5,F
OP_5	I,OP_2	OP_6
OP_6	I,OP_5	–



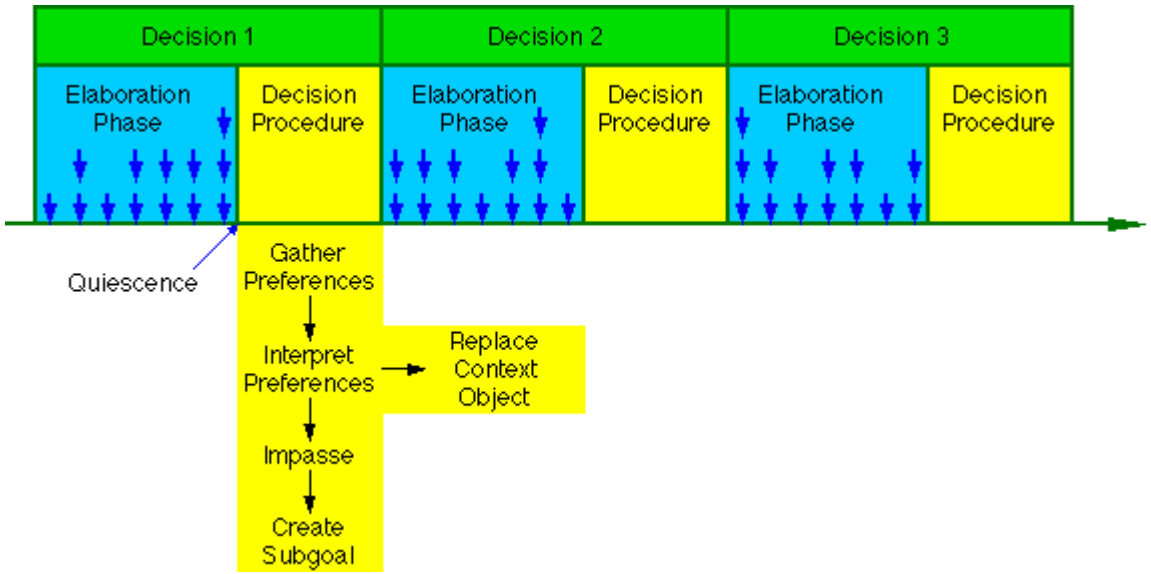
SOAR

- SOAR is a cognitive modeling architecture originally developed by Allen Newell and his students at CMU.



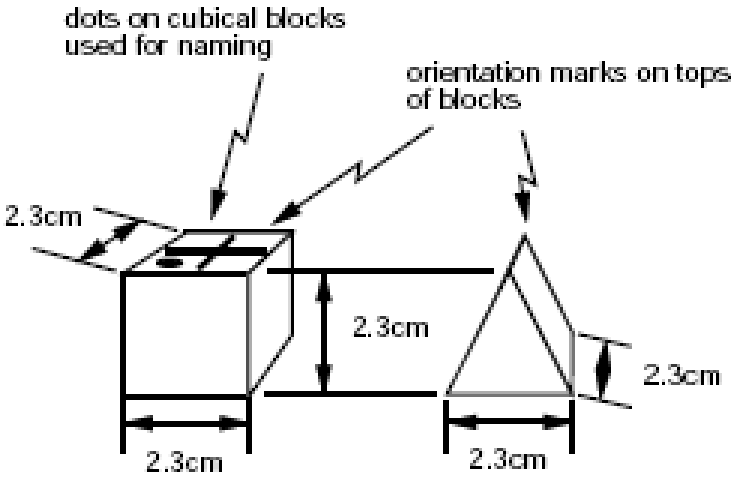
SOAR

- SOAR uses *production rules* to match items in working memory, update the memory, and initiate actions.
- An impasse (failure to match) leads to subgoal creation.
- Chunking is used to abstract and remember production sequences.



Robo-SOAR

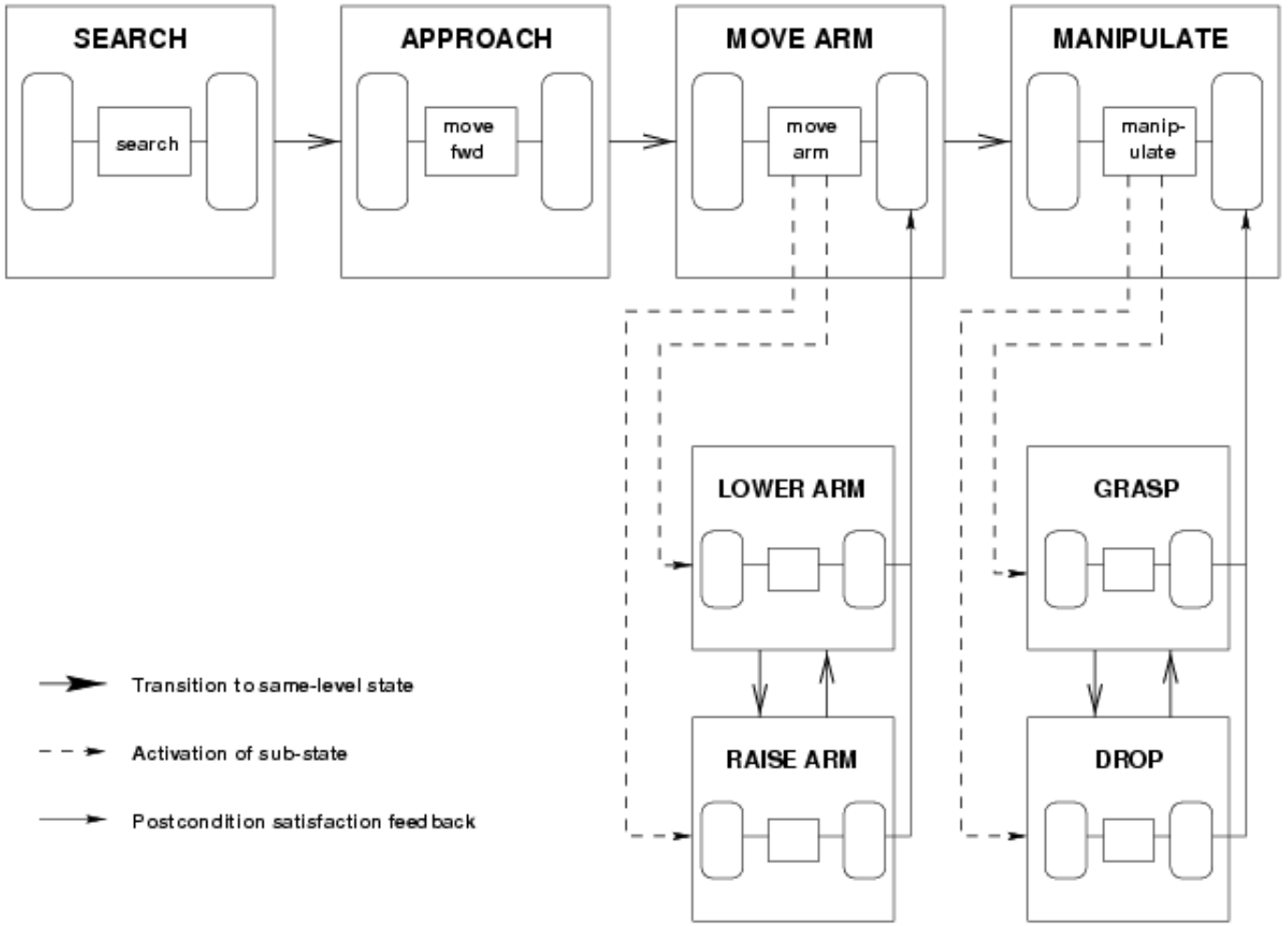
- Laird, Yager, and Hucka (1991)
- Extended SOAR to allow control of a Puma 560 arm
- Solved simple blocks world problems
- Two block types: cubes and pyramids
- Initial plan to pick up a pyramid fails when gripper not oriented correctly.



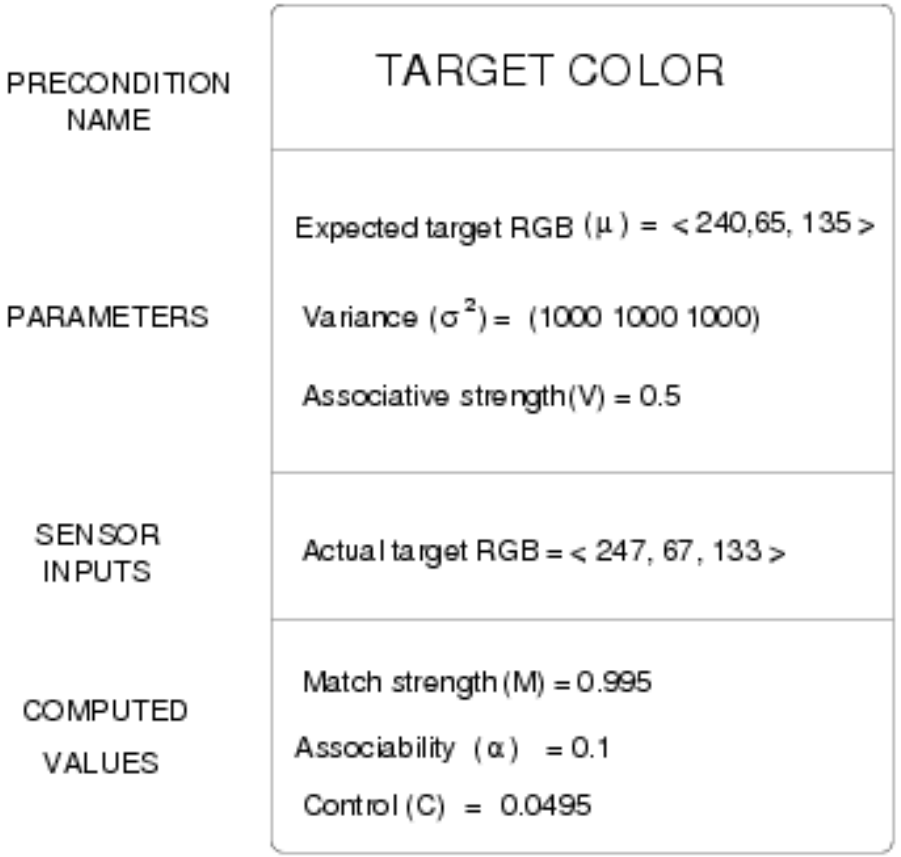
Skinnerbots (Touretzky & Saksida)

- Can we apply Skinnerian (operant) conditioning to robots?
  - Represent behaviors as schemas with modifiable preconditions
  - Use reward signal to train the robot
  - Try to infer preconditions from context + reward

Schema Representation



Example of a Precondition



Machine Learning Algorithms Applied to the AIBO

- Temporal Difference (TD) learning for classical conditioning

- Two-armed bandit learning problem

Potential for Learning in Tekkotsu

- Currently there is map learning (MapBuilder)
- Cognitive robotics student project (2006): interface to ACT-R
- New ideas:
  - Provide a way to supply reward signals
  - Provide some persistence of memory across reboots via a facility for storing variables on the memory stick.