Intelligence by Design

Principles of Modularity and Coordination for Engineering Complex Adaptive Agents

Joanna J. Bryson

MIT Artificial Intelligence Laboratory
joanna@ai.mit.edu
http://www.ai.mit.edu/∼joanna

Goal

The aim of this thesis: to make it easier for engineers to build complex agents which can succesfully behave, learn and plan.

- Artifacts with 'personality' (e.g. autonomous robots, virtual reality characters.)
- Anything with potentially conflicting goals or behaviors.

Working systems in talk: Standard ALife comparison platform, mobile robot, model of primate learning.

Outline

- Introduction
  - Combinatorics and Search
  - Modularity
  - Behavior Oriented Design
- Components of Agent Intelligence
- Design Methodology
- Related Work
- Future Work
- Conclusions and Contributions

The Problem

- Combinatorics is the problem. Search is the solution.
  - Planning
  - Learning
  - Design
- The task of intelligence is to bias (focus) search.
  - Develop good search techniques.
  - Limit search space to likely solutions.
- Engineering is the primary source of bias in AI.

Modularity

- Modularity simplifies design.
  - Decomposes the problem into simpler units.
  - Focuses search using locally optimal representations.
- It also generates design issues.
  - Decomposition
  - Coordination
  - Learning

Behavior Oriented Design

BOD exploits modularity to limit search while addressing modularity's problems:

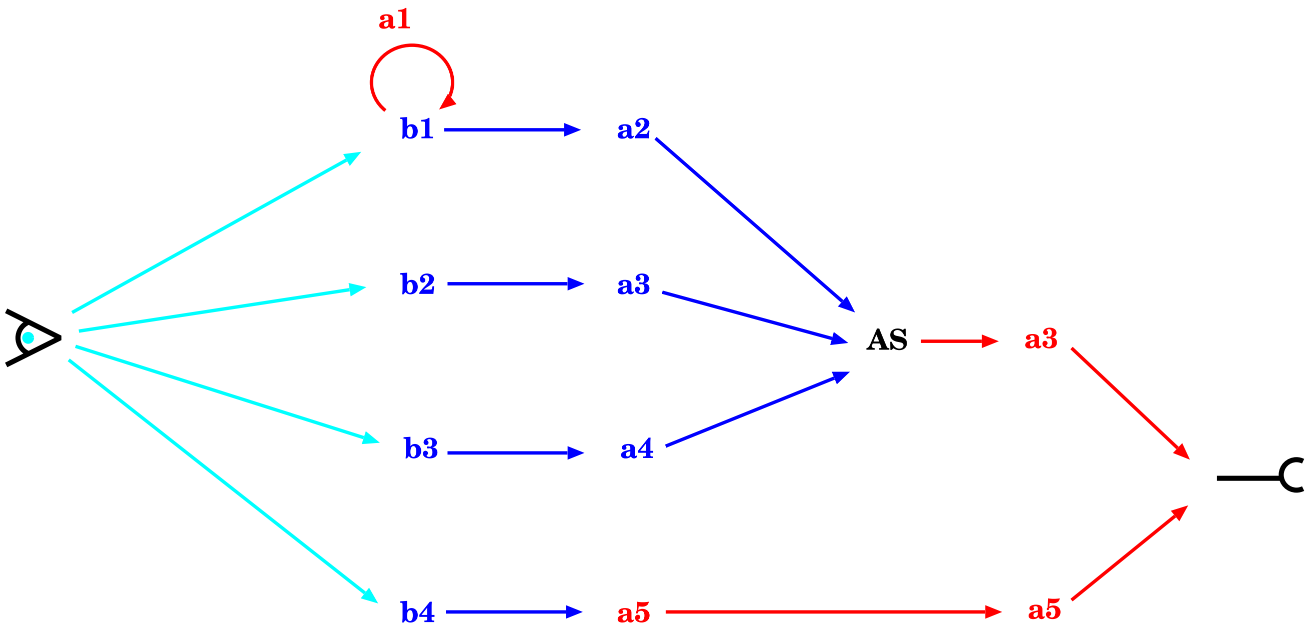
- Learning is done within modules.
- Modular decomposition is dictated by variable state.
- Coordination between modules is done by hierarchical reactive plans.

Outline

- Introduction
- Components of Agent Intelligence
  - Behaviors
  - Reactive Plans
  - Examples
- Design Methodology
- Related Work
- Future Work
- Conclusions and Contributions

Components: What Every Agent Wants

1. Modularity
2. Hierarchical Reactive Plans
3. Environment Monitoring / Alarm System



What is a Behavior? (in BOD)

- A module in an agent.
- Control for agent's actions (expressed and/or internal).
- Perception required for that control.
- Variable state required for perception or control.
- Not fully encapsulated.

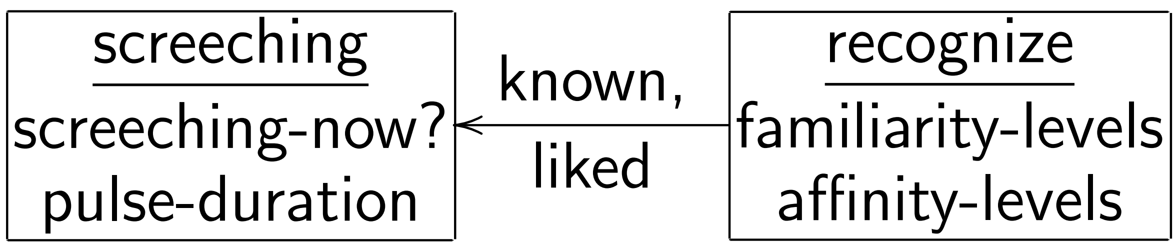
A Simple Behavior

screeching

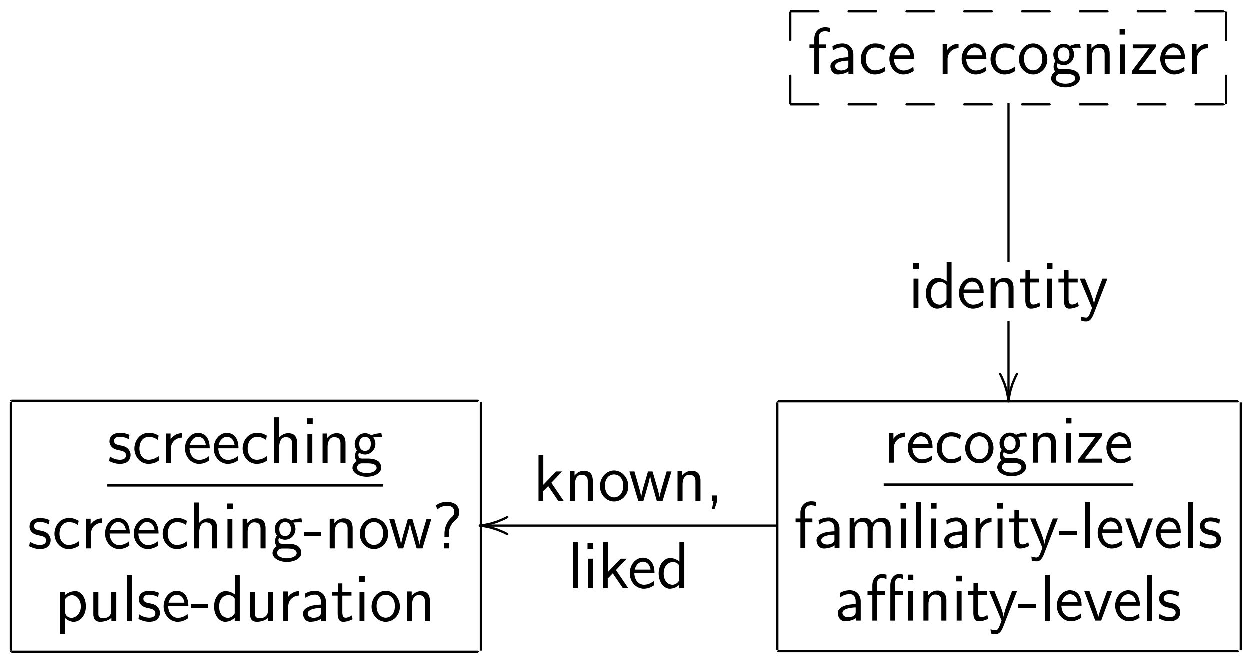
A Behavior with State

screeching
screeching-now?
pulse-duration

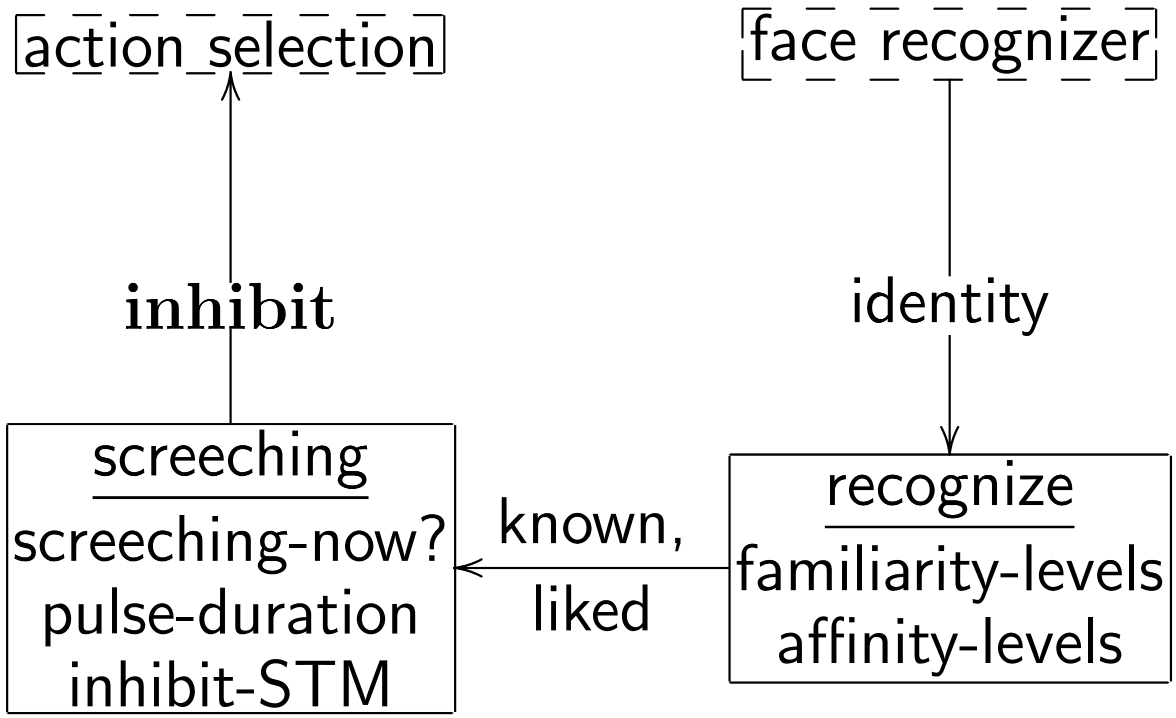
Behaviors with Perception



Behaviors that Aren't Objects



Behaviors with Processes and/or Triggers



Outline

- Introduction
- Components of Agent Intelligence
  - Behaviors
  - Reactive Plans
  - Examples
- Design Methodology
- Related Work
- Future Work
- Conclusions and Contributions

What is a Reactive Plan?

- Modularity leads to coordination problems.
  - Behavior arbitration
  - Multi-agent coordination
  - Action selection
- Reactive plans are an engineered solution.
  - Planning
  - Reactive Planning
  - Reactive Plans

Reactive Plans in BOD

- Use hierarchy (modularity) to limit search.
- Take advantage of what engineers are good at: (currently?)
  - Describing sequences of events.
  - Ordering priorities.
- Support three types of action selection problems:
  - Some things need to be checked at all times.
  - Some only need considering in particular contexts.
  - Some things reliably follow from others.

Some Things Follow: Action Patterns

$\langle$get a banana $\longrightarrow$ peel a banana $\longrightarrow$ eat a banana$\rangle$

Are Production Rules Better than Sequences?

$$\begin{aligned}
\text{(have hunger)} &\Rightarrow \text{get a banana} \\
\text{(have a banana)} &\Rightarrow \text{peel a banana} \\
\text{(have a peeled banana)} &\Rightarrow \text{eat a banana}
\end{aligned}$$

Are Production Rules Better than Sequences?

$$
\begin{aligned}
\text{(have hunger)} &\Rightarrow \text{get a banana} \\
\text{(have a banana)} &\Rightarrow \text{peel a banana} \\
\text{(have a peeled banana)} &\Rightarrow \text{eat a banana}
\end{aligned}
$$

No — A Sequence is State

$\langle$get a banana from left $\longrightarrow$ pass a banana to right$\rangle$

(left neighbor offers banana) $\Rightarrow$ get a banana from left

(have a banana) $\Rightarrow$ pass a banana to right

Basic Reactive Plans: State + Flexibility

$$
(\text{hungry}) \Rightarrow \Bigg\uparrow \left\langle \begin{array}{rl} (\text{full}) \Rightarrow & \textit{goal} \\ (\text{have a peeled banana}) \Rightarrow & \text{eat a banana} \\ (\text{have a banana}) \Rightarrow & \text{peel a banana} \\ \Rightarrow & \text{get a banana} \end{array} \right\rangle
$$

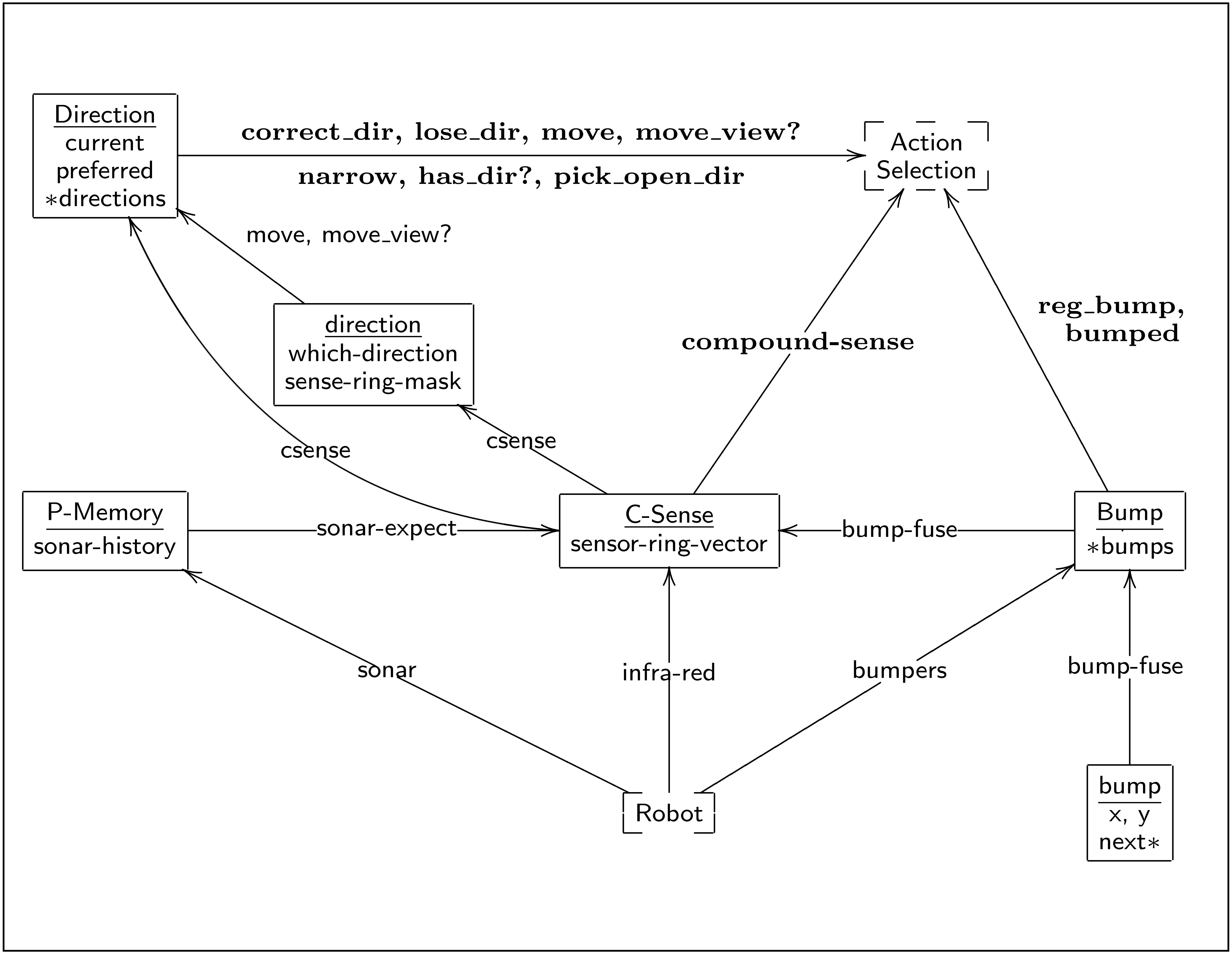
Many different *expressed* plans (sequences of behavior) are determined by one *reactive* plan.

Parallel-rooted, Ordered Slip-stack Hierarchical (POSH) Action Selection

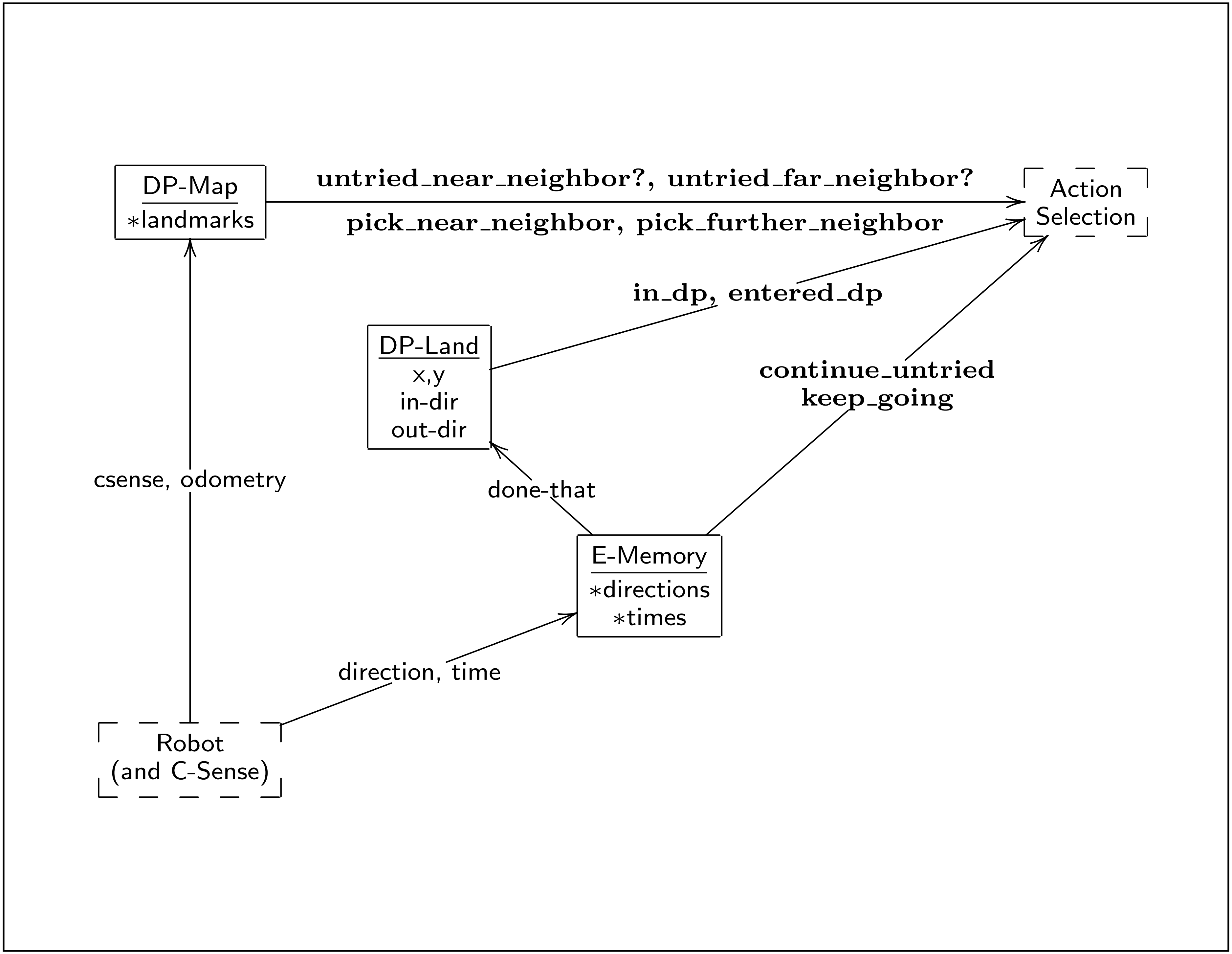
- *Action Pattern*: $\iota_1, \iota_2, \ldots \iota_n$
- Basic Reactive Plans: set of steps $\{\langle \pi_i, \rho_i, \alpha_i \rangle *\}$
  - *Competence:* competence step $\langle \pi,\ \rho,\ \alpha,\ \eta \rangle$
  - *Drive Collection:* drive $\langle \pi, \rho, \alpha, A, \nu \rangle$
    - No stack (3,000Hz on a 486)
    - Action scheduler (256Hz on a PentiumII)

Outline

- Introduction
- Components of Agent Intelligence
  - Behaviors
  - Reactive Plans
  - Examples (on other projector)
- Design Methodology
- Related Work
- Future Work
- Conclusions and Contributions



Direction
current
preferred
*directions
correct_dir, lose_dir, move, move_view?
narrow, has_dir?, pick_open_dir
Action
Selection
move, move_view?
direction
which-direction
sense-ring-mask
compound-sense
reg_bump,
bumped
csense
csense
P-Memory
sonar-history
sonar-expect
C-Sense
sensor-ring-vector
bump-fuse
Bump
*bumps
sonar
infra-red
bumpers
bump-fuse
Robot
bump
x, y
next*


DP-Map
*landmarks
untried_near_neighbor?, untried_far_neighbor?
pick_near_neighbor, pick_further_neighbor
Action Selection
in_dp, entered_dp
DP-Land
x,y
in-dir
out-dir
continue_untried
keep_going
csense, odometry
done-that
E-Memory
*directions
*times
direction, time
Robot
(and C-Sense)

Outline

- Introduction
- Components of Agent Intelligence
- Behavior Oriented Design
  - Initial Decomposition
  - Cyclic Development
  - Example
- Related Work
- Future Work
- Conclusions and Contributions

Initial Decomposition

1. Specify (high-level) what the agent will do.
2. Describe activities as sequences of actions. reactive plans
3. Identify sensory and action primitives from these sequences.
4. Identify the state necessary to enable the primitives, cluster primitives by shared state. behaviors
5. Identify and prioritize goals or drives. drive collection
6. Select a first behavior to implement.

Cyclic Development

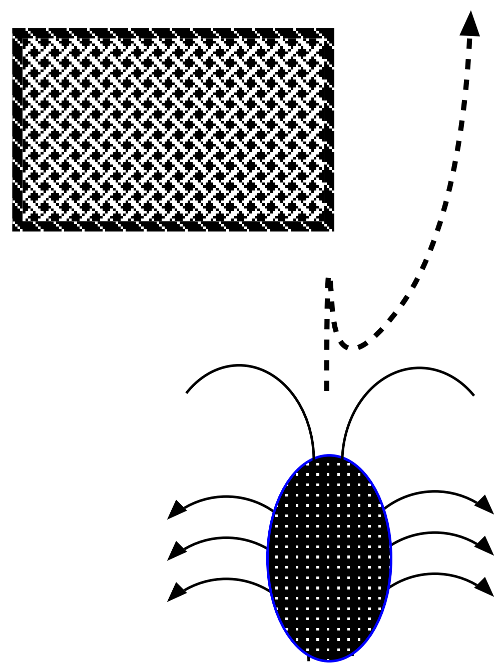
- Scale the system.
  - Code behaviors and / or plans.
  - Test and debug code.
- Simplify the design.
  - Revise the specifications.

Simplifying the Design

Exploit trade-offs between representations.

- Behavior Modules
- Reactive Plans

Example



Control State Only

$$\textbf{walk} \Rightarrow \uparrow \left\langle \begin{array}{rl} \text{(left-feeler-hit)} \Rightarrow & \textbf{avoid-obstacle-left} \\ \text{(right-feeler-hit)} \Rightarrow & \textbf{avoid-obstacle-right} \\ \Rightarrow & \text{walk-straight} \end{array} \right\rangle$$

$$\textbf{avoid-obstacle-left} \Rightarrow \langle \text{walk backwards} \rightarrow \text{walk right} \rightarrow \text{walk left} \rangle$$

$$\textbf{avoid-obstacle-right} \Rightarrow \langle \text{walk backwards} \rightarrow \text{walk left} \rightarrow \text{walk right} \rangle$$

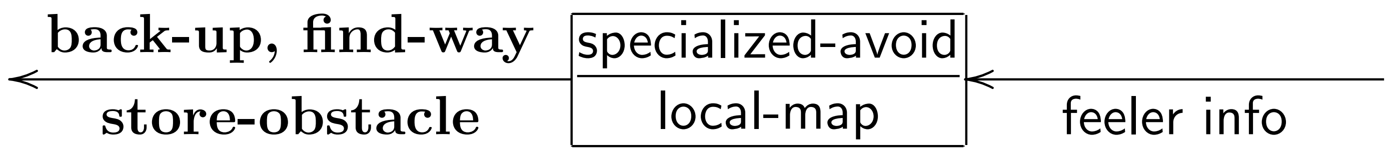
Deictic State as Well

$$\xleftarrow[\textbf{compensate-avoid}]{\textbf{avoid-hit, feeler-hit,}} \boxed{\frac{\text{deictic-avoid}}{\text{hit-left}}} \xleftarrow[\text{feeler info}]{}$$

$$\textbf{walk} \Rightarrow \Big\uparrow \left\langle \begin{array}{l} \text{(feeler-hit)} \Rightarrow \textbf{avoid-obstacle} \\ \qquad\qquad\quad\;\; \Rightarrow \text{walk-straight} \end{array} \right\rangle$$

$$\textbf{avoid-obstacle} \Rightarrow \langle \text{walk backwards} \rightarrow \text{avoid hit} \rightarrow \text{compensate avoid} \rangle$$

Specialized State (rather than Deictic)



$$\textbf{walk} \Rightarrow \Big\uparrow \left\langle \begin{array}{l} \text{(feeler-hit)} \Rightarrow \text{store-obstacle back-up} \\ \qquad\qquad\quad\; \Rightarrow \text{find-way} \end{array} \right\rangle$$

Revising the Specification: State

- Prefer the simplest.
  1. Control State
  2. Deictic State
  3. Specialized State (learning)
  4. Meta-State (learning to learn)
- Exceptions:
  - Eliminate Plan Redundancy
  - Reduce Plan Complexity

Revising the Specification: Control

- Prefer the simplest.
  - Single Primitive > Sequence
  - Sequence > BRP
  - Control State > Variable State
- Exceptions:
  - Want part of primitive ⇒ sequence.
  - Sequence elements repeated, skipped ⇒ BRP.
  - Use variable state to:
    * Replace lots of triggers.
    * Generalize control state.

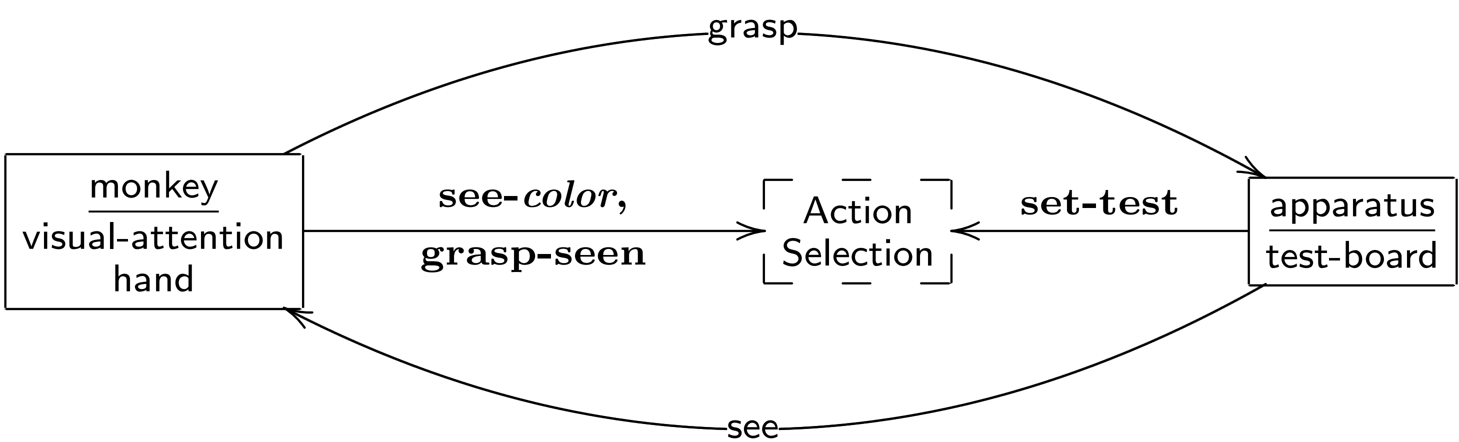
Outline

- Introduction
- Components of Agent Intelligence
- Behavior Oriented Design
  - Initial Decomposition
  - Cyclic Development
  - Example
- Related Work
- Future Work
- Conclusions and Contributions

Transitive Inference

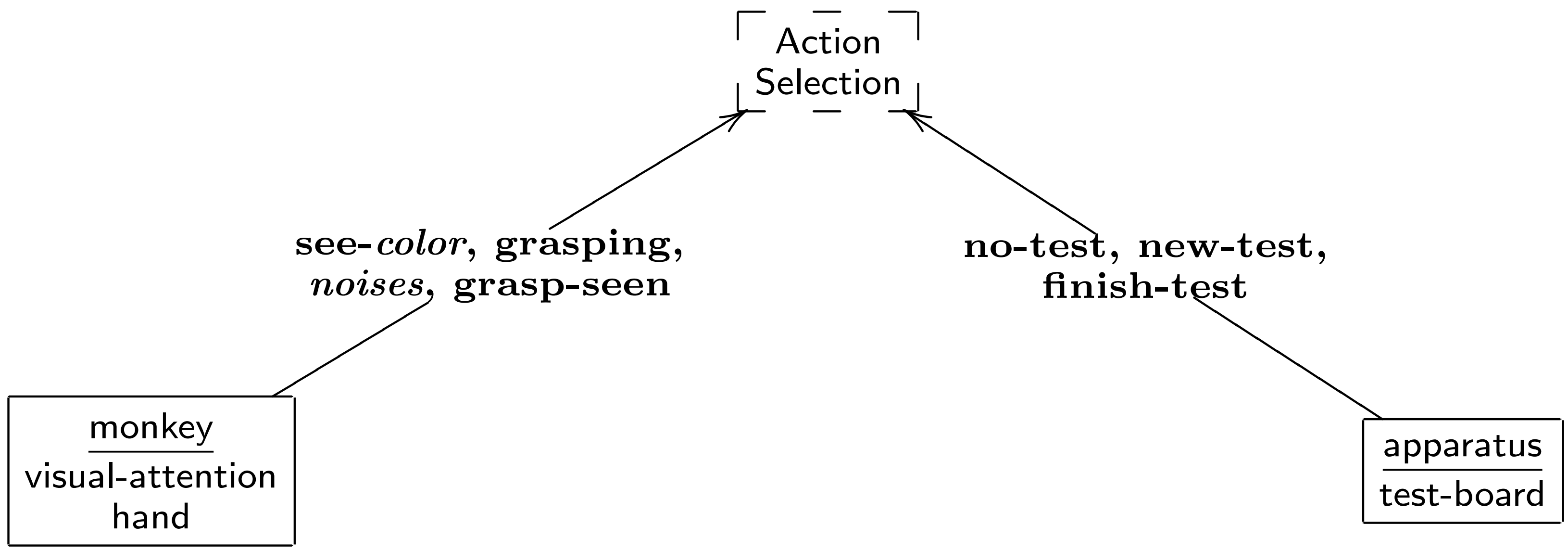
- $A > B$ and $B > C$ implies $A > C$.
  - Not about logic or concrete operational thought.
- McGonigle & Chalmers (1977) show:
  - monkeys can do it for 5 items, and
  - not as good at triads (neither are childeren).
- Harris & McGonigle (1994) demonstrate:
  - model with production rule stack, and
  - learning ordering of rules, not of blocks.
- Many neural network models (Wynne 1998).
  - show learning but not learning rules.

binary-test

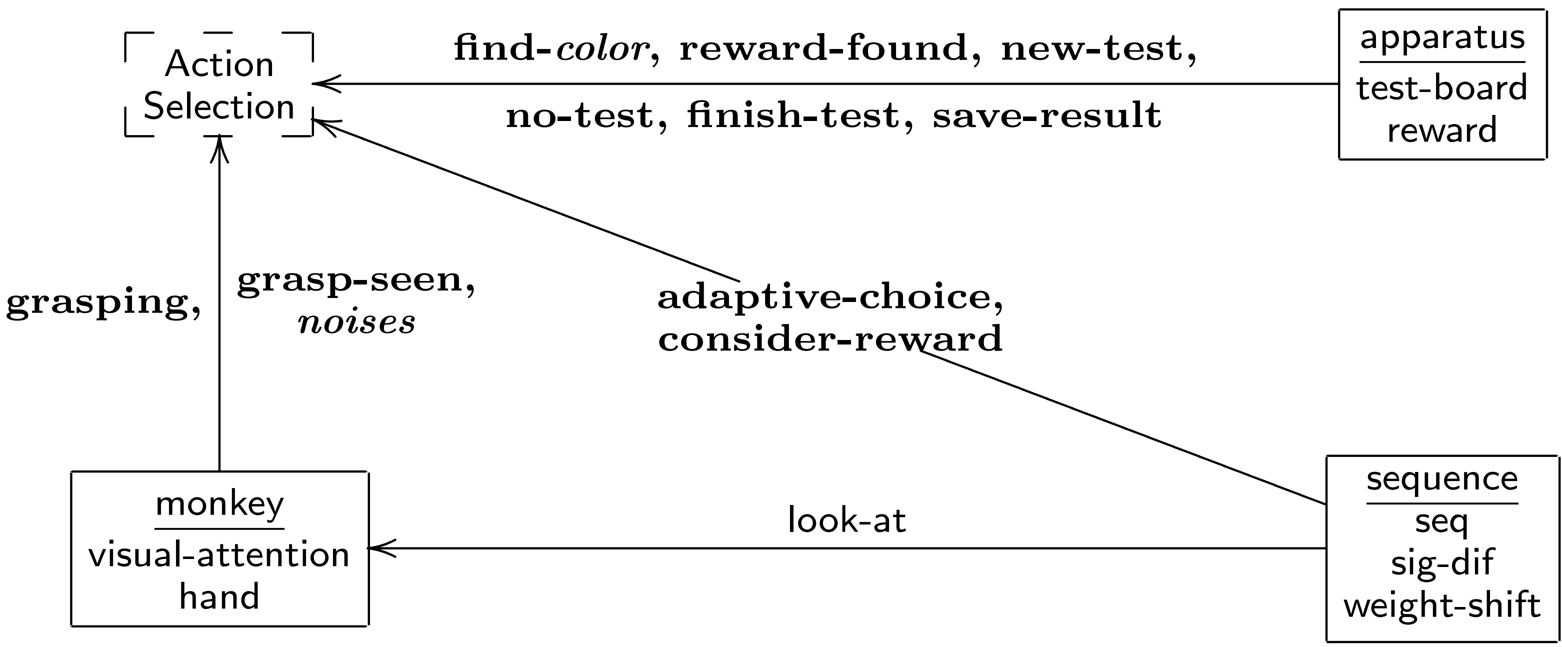


grasp
monkey
visual-attention
hand
see-*color*,
grasp-seen
Action
Selection
set-test
apparatus
test-board
see

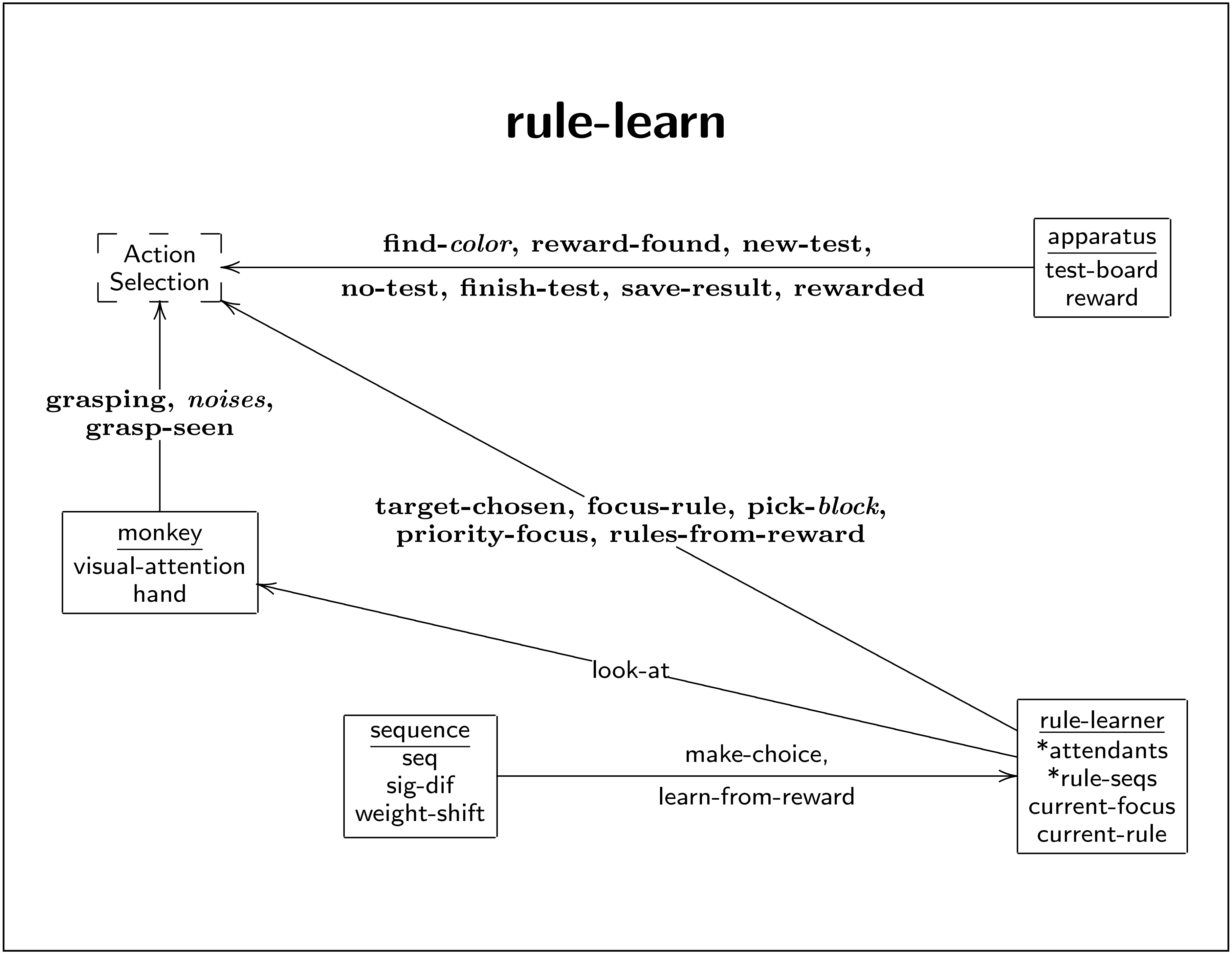
driven-b-test



prior-learn



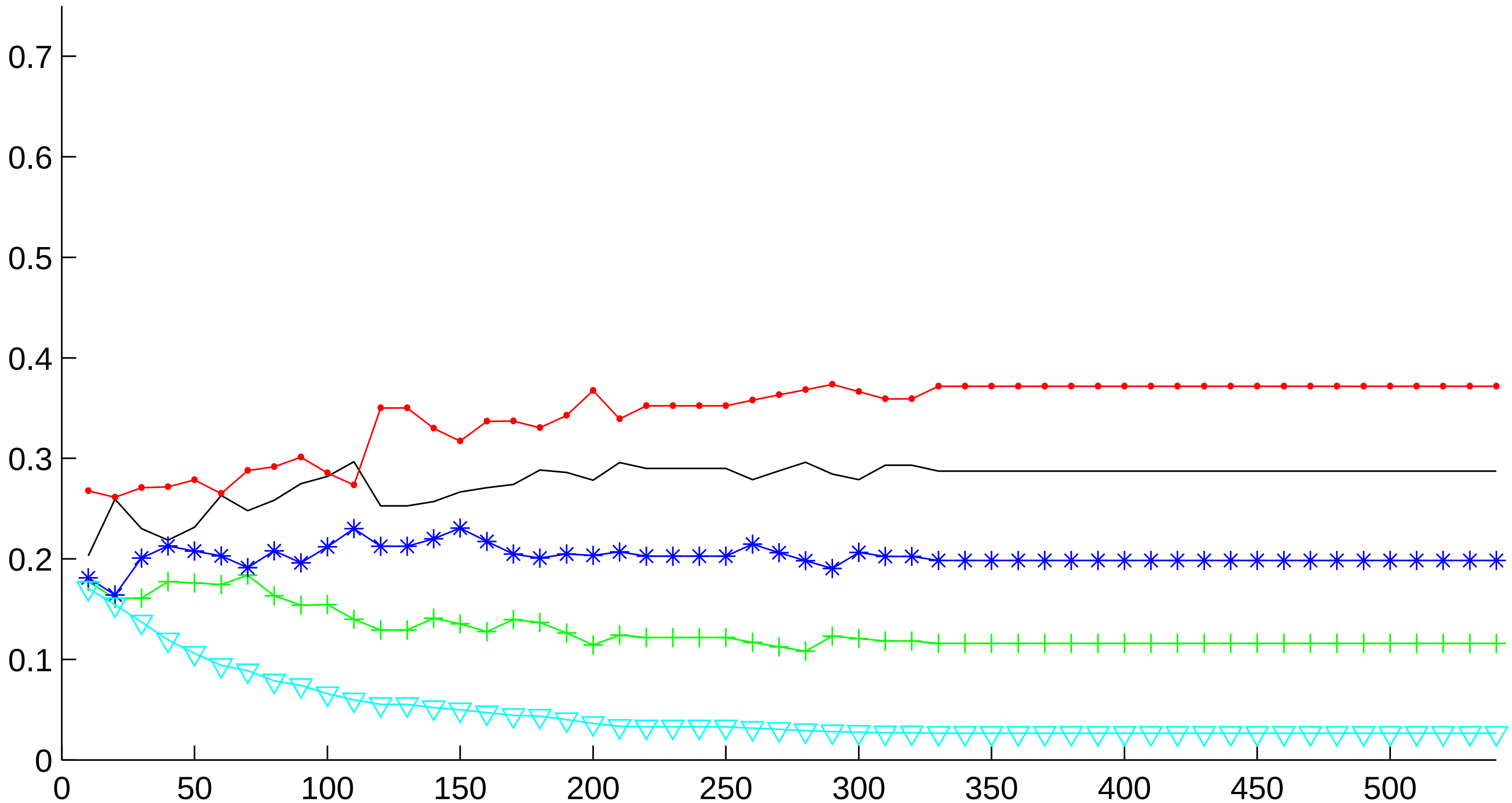
Action Selection
find-*color*, reward-found, new-test,
no-test, finish-test, save-result
apparatus
test-board
reward
grasping,
grasp-seen,
noises
adaptive-choice,
consider-reward
monkey
visual-attention
hand
look-at
sequence
seq
sig-dif
weight-shift



rule-learn
Action Selection
find-*color*, reward-found, new-test,
no-test, finish-test, save-result, rewarded
apparatus
test-board
reward
grasping, *noises*,
grasp-seen
target-chosen, focus-rule, pick-*block*,
priority-focus, rules-from-reward
monkey
visual-attention
hand
look-at
sequence
seq
sig-dif
weight-shift
make-choice,
learn-from-reward
rule-learner
*attendants
*rule-seqs
current-focus
current-rule

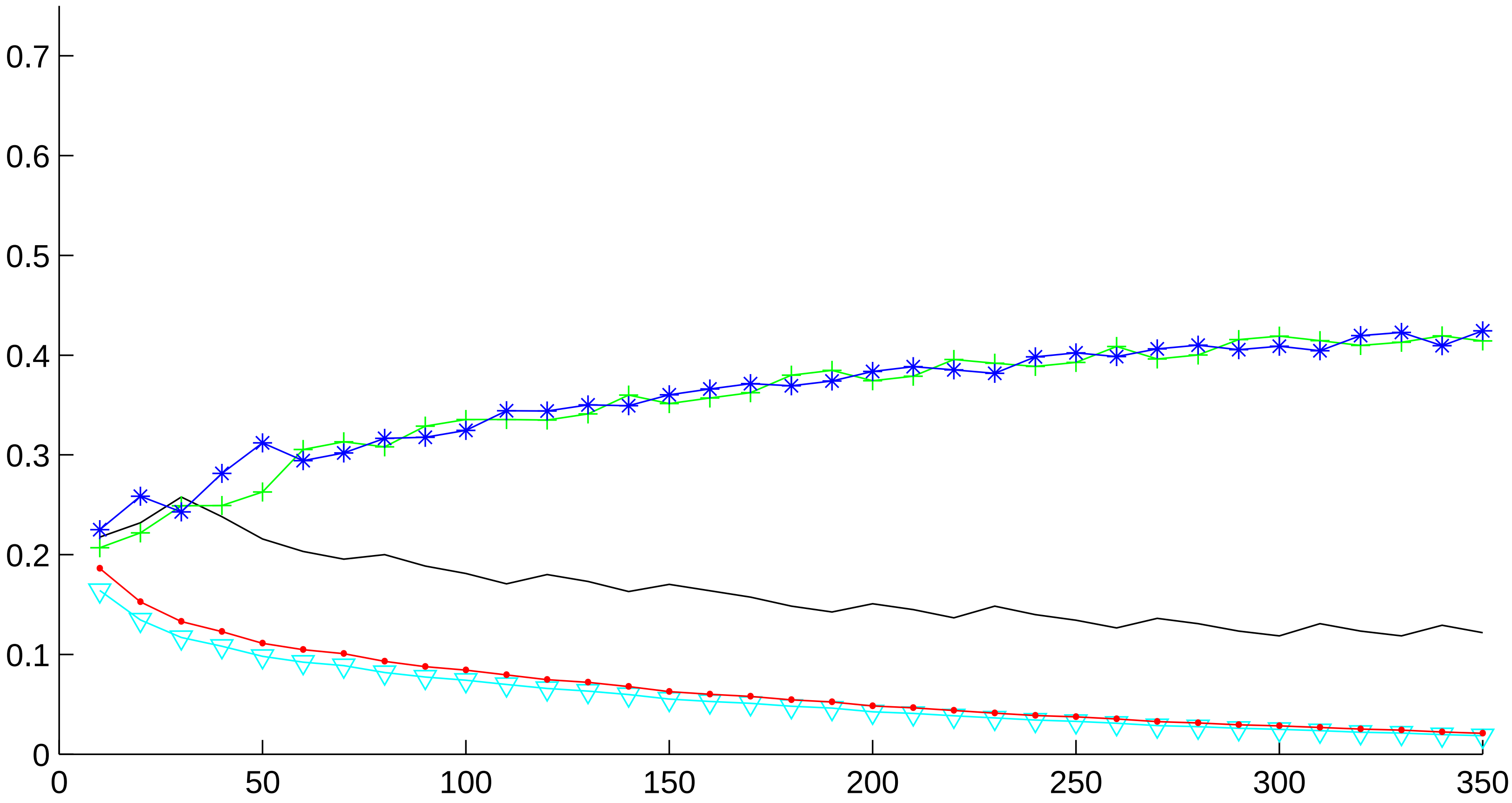
Prior-learn without regimented training

Select 1^{st}, Select 2^{nd}, Select 3^{rd} (correct)



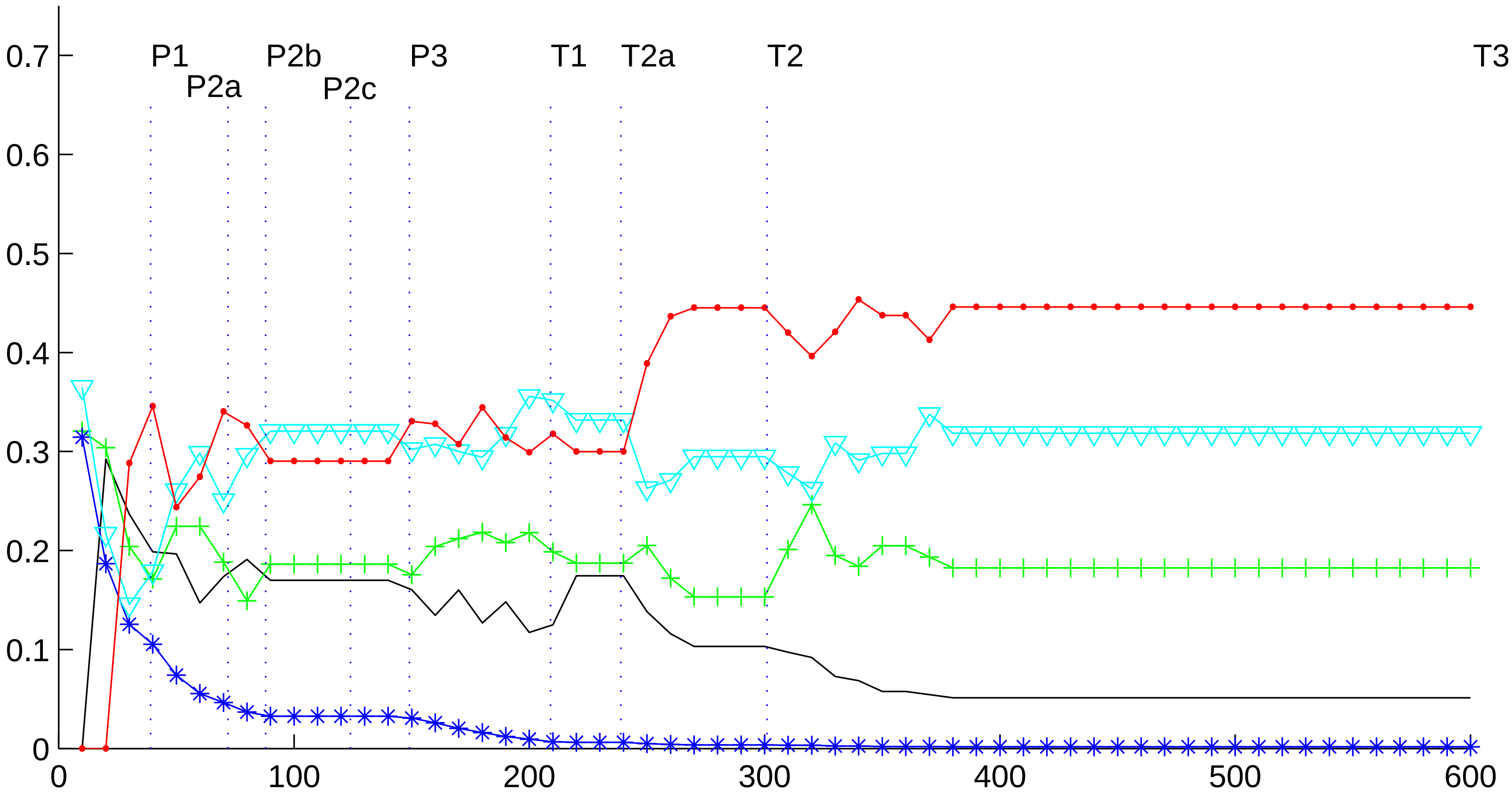
Rule-learn without regimented training

Select 4^{th}, Avoid 3^{rd}, Avoid 2^{nd} (confuses only 3^{rd} with 4^{th})



Rule-learn with regimented training

Select 1^{st}, Avoid 5^{th}, Avoid 4^{th} (correct!)



Outline

- Introduction
- Components of Agent Intelligence
- Design Methodology
- Related Work
  - Reactive Control
  - Behavior Modules
- Future Work
- Conclusions and Contributions

Related Work: Reactive Control

- Behavior-Based and Production Rule systems, e.g. Subsumption (Brooks 1986), ANA (Maes 1992), Soar (Newell 1990), specialize in emergencies.
- Plan-based hybrids, e.g. PRS (Georgeff & Lansky 1987), JAM (Huber 1999), 3T / RAPs (Bonasso, Firby et. al 1997), specialize in order.
- Only Teleo-Reactive (Nilsson 1994) has BRPs and a user base. No user base: (Fikes 1972) (Correia and Steiger-Garção 1995) (me).

Related Work: Behaviors, Modularity and Learning

- Behavior-Based AI has modularity and specialized learning, but overly diffuse control. (e.g. Brooks 1991, Horswill 1993)
- Hybrid systems have reactive plans, but reduce behaviors to mere primitives, have overly monolithic representations.

Outline

- Introduction
- Components of Agent Intelligence
- Design Methodology
- Related Work
- Future Work
  - Learning
  - Tools
  - Applications
- Conclusions and Contributions

Learning

- Learning in Behaviors
- Learning of Plans
  - Search
  - Evolution
  - Imitation
- Learning Behaviors
  - Existing work is one behavior in BOD.
  - Learning dynamical models (e.g. Hogg, Brand)

Tools

- BOD has been applied in a variety of architectures.
  - Support object-level coding.
  - Implement POSH Action Selection.
  - PRS (Meyer 1996), JAM (Huber 1999), Ymir (Thórisson 1996)
- Tools support methodology across architectures.
  - Construction
  - Debugging
- Customized tools for users in one domain.

Applications

- Virtual Reality Characters
- Simplifying "Big AI" Systems
  - Dialog Systems
  - Intelligent Environments
- Cognitive Modeling

Outline

- Introduction
- Components of Agent Intelligence
- Design Methodology
- Related Work
- Future Work
- Conclusions and Contributions

Conclusions

- Engineering is key to AI.
- Modularity supports specialized representations for focussed tasks.
  - This makes learning (and planning) tractable.
- Coordination in time is a critical module.
  - Represented via explicit hierarchy and sequence.
- Optimizing for simplicity should be an integral part of the development cycle.

Contributions

- In this talk:
  - Behavior Oriented Design.
  - Details of POSH Action Selection.
  - Models of primate transitive inference / learning BRPs.
- Read the thesis:
  - Two POSH architectures (C++ and CLOS).
  - Relation to other architectures.
  - Relation to the brain.
  - MAS model of monkey social behavior.

[Talk Boundary]

Drive Collections: BRPs for Environment Monitoring

$$life \Rightarrow \left\langle \left\langle \begin{array}{r} \text{(something looming)} \Rightarrow \text{avoid} \\ \text{(something loud)} \Rightarrow \text{attend to threat} \\ \text{(hungry)} \Rightarrow \text{forage} \\ \Rightarrow \text{lounge around} \end{array} \right\rangle \right\rangle$$

Revising the Specification – BRPs

- A BRP is a worst-case scenerio sequence backwards.
- A BRP should only have 3-7 elements.
- Too many elements or triggers:
  - Two ways to do same goal $\Rightarrow$ make sibling BRPs
  - Multi-step subgoal $\Rightarrow$ make child sequence or BRP.
- Be careful of termination.
  - Converge to goal.
  - Fail if goal is impossible (habituate).
  - Manage chaining.

Learning Behaviors