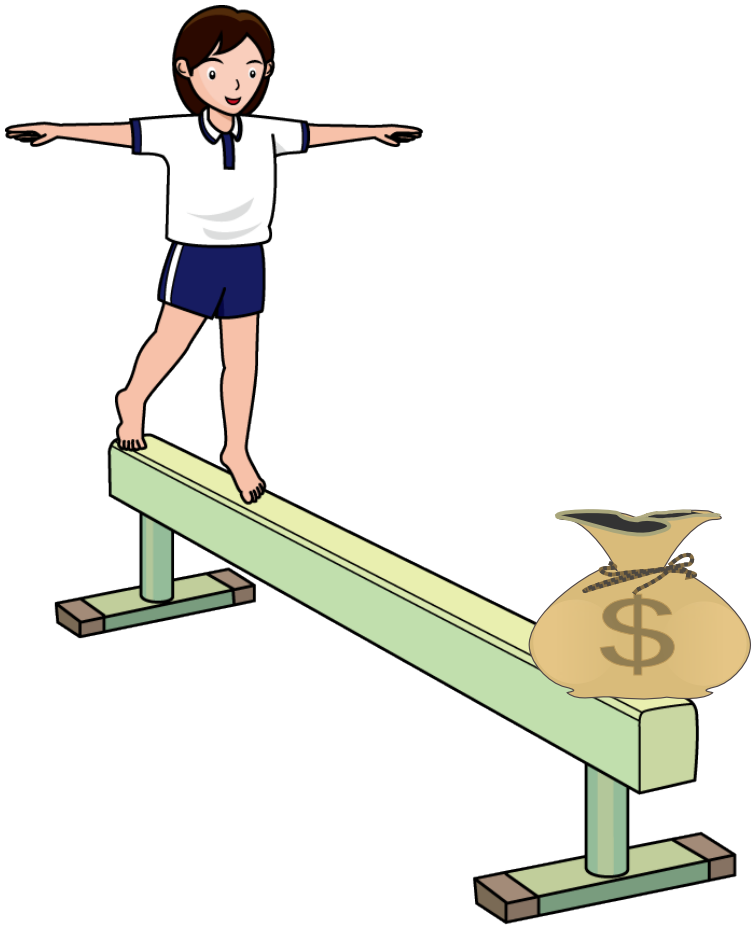
Ideas for Reinforcement Learning Algorithm that Generates Programs

Susumu Katayama

Motivation

Tractable RL algorithm
that can cope with *various unexpected cases*
and *environmental changes*
like humans

Case study: Balance beam

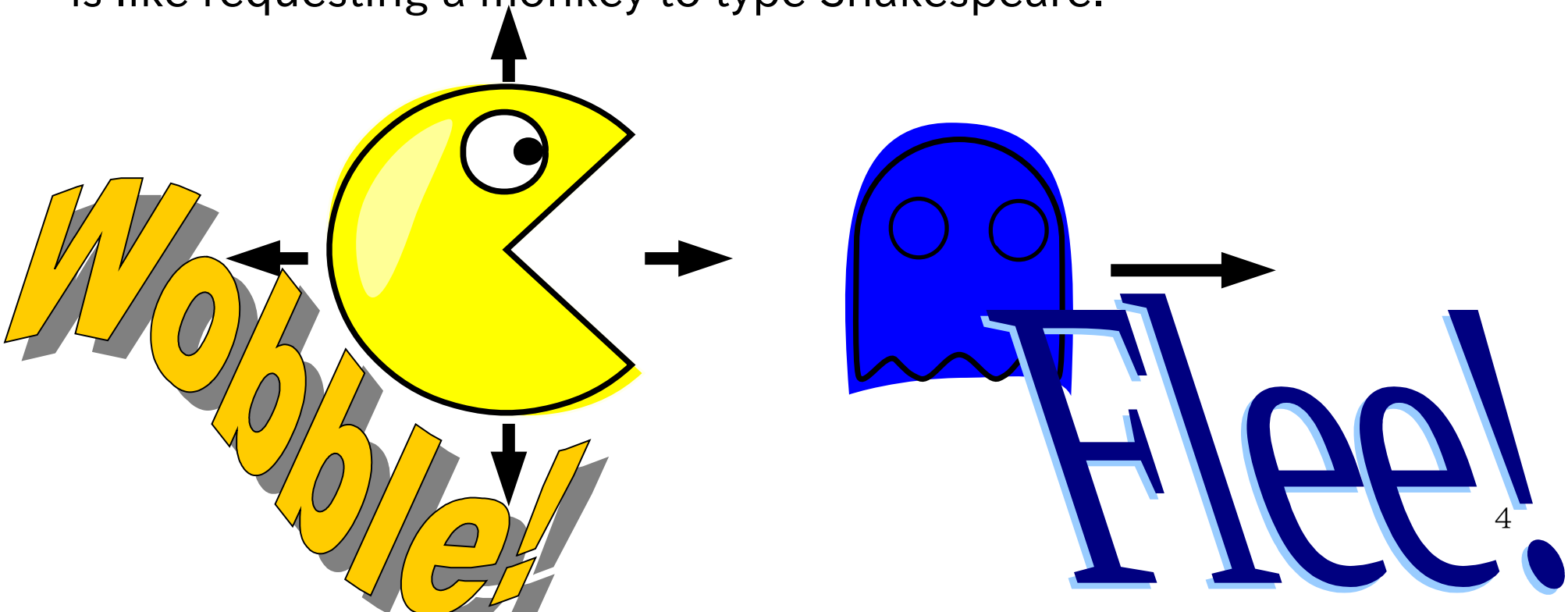


ε-greedy is not an option!

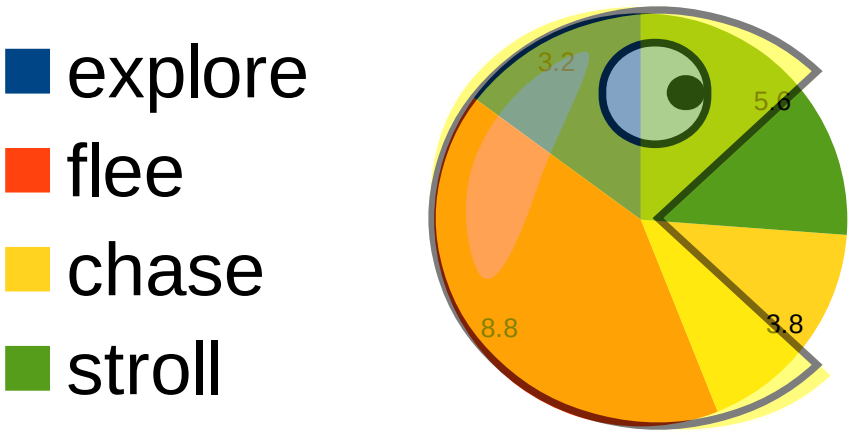
Case study: Chasing

With MC-AIXI (FAC-CTW) Pacman failed to learn to chase pale ghosts

- "this is a concern with AIXI in general"
  "It is the transposition of the exploration/exploitation dilemma."
  (Orseau 2010, proving non-optimality of AIXI)
- Even with random exploration, without any reward until capture, requesting to consistently chase down a fleeing ghost (even once) is like requesting a monkey to type Shakespeare.



Case study: Chasing (cont'd)



Possible solutions:

- randomized search over compound actions or programs
  (Policies should also be programs.)
- search by population
  (by enabling imitation between agents)

Case study: Problem solving by all means

Ability to plan/program is desired.

(Possible by some connectionist approaches in theory,
But they are not straightforward for doing this.)

Case study: Problem solving by all means

https://cryptome.org/eyeball/daiichi-npp2/daiichi-photos2.htm

Ability to plan/program is desired.

(Possible by some connectionist approaches in theory,
But they are not straightforward for doing this.)

Design policy

- Model the env. as dist. over programs
  (like AIXI unlike MC-AIXI(FAC_CTW),
  because generalization is essential.)
- Model the policy as dist. over programs
  (to search among compound functions,
  like policy gradient)
- Help search by population by imitation

Definitions

- $\mathscr{A}$: set of actions
- $\mathscr{X}$: set of inputs to the agent
- $\mathscr{H}$: set of histories, i.e. sequence of actions and inputs
- $\mathscr{C}$: set of computable functions with type $\mathscr{H} \rightarrow \mathscr{A}$
- $\mathscr{P}$: set of policies, which is subset of $\mathscr{C}$
- $V^{p\xi}(h)$: the expected return after history *h*
  assuming the agent will follow the policy *p*
  and the plausibility of the environment model
- $\hat{V}$: current estimation of V

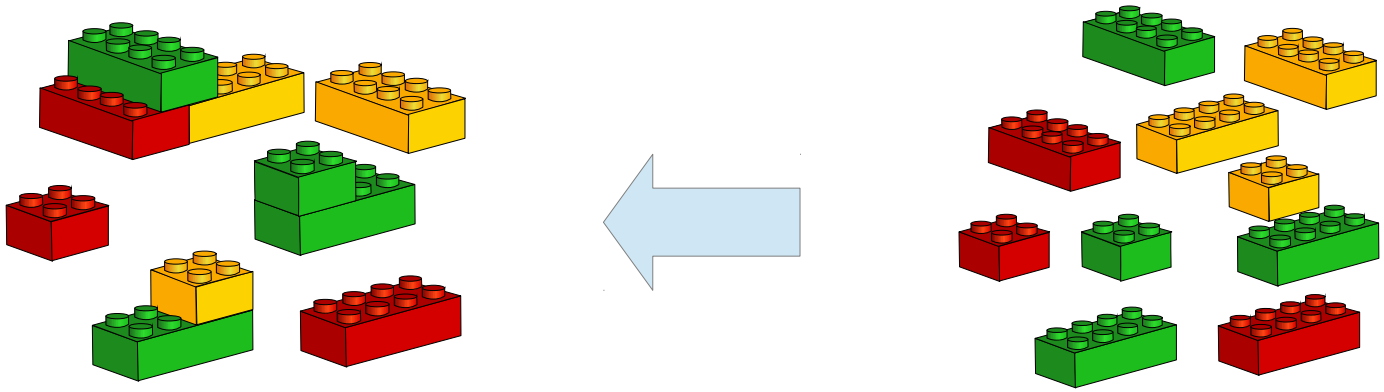
The environment model

Idea: Use MagicHaskeller to generate environment program candidates

- Keeps all program candidates,
  (like Solomonoff induction and AIXI)
- Saves space by removing semantically equivalent programs
  (So it is efficient!)

* MagicHaskeller generates Haskell programs instead of Turing machines.

Learning the environment incrementally

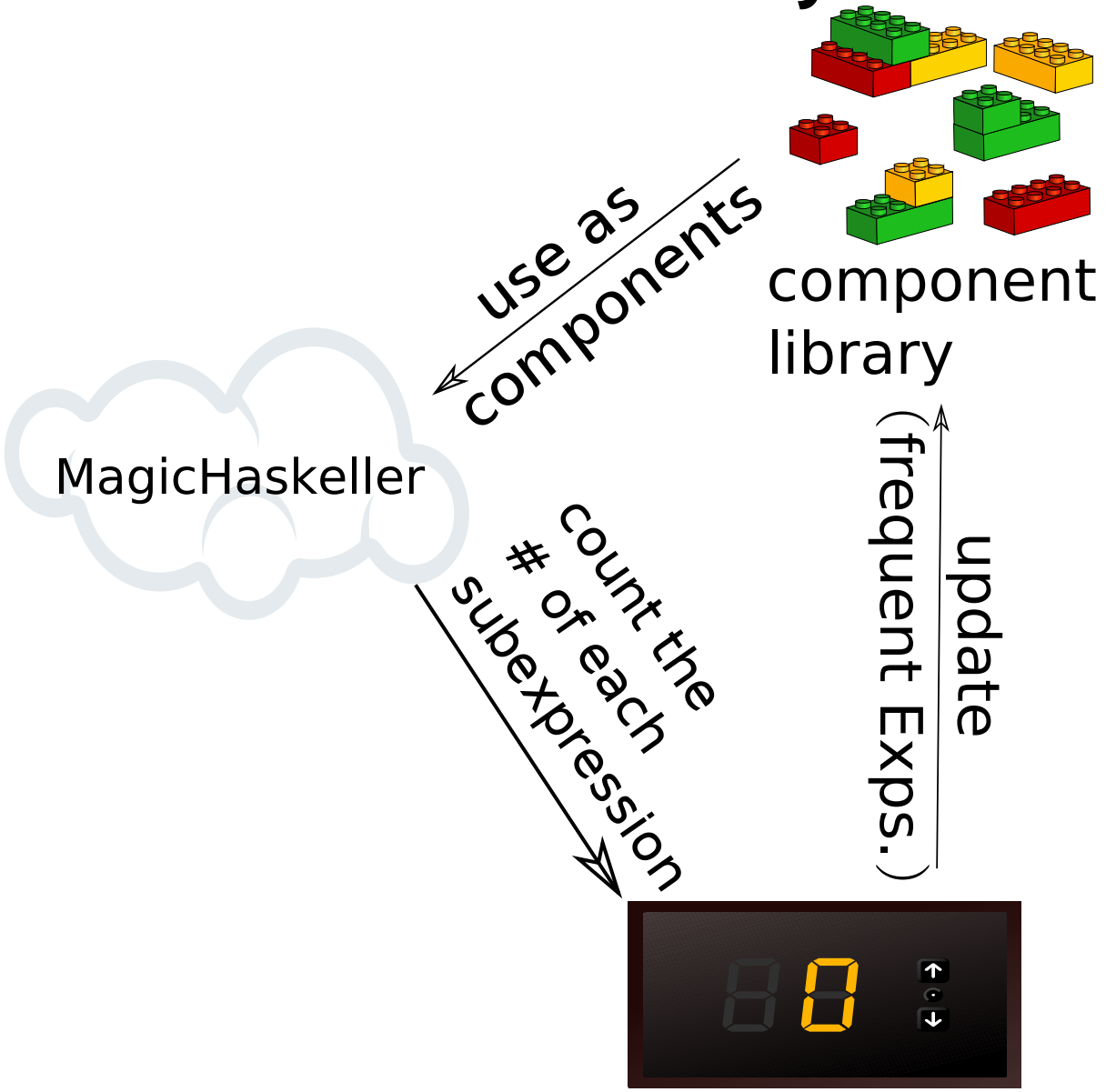


Synthesized environment model

Component Library

Bigger and bigger programs can be learned by updating the component library based on usage frequency.

Learning the environment incrementally



Policy as programs: A Straightforward Approach

Recompute the policy at each time step *k*?

I.e.,

$$p_k = \mathrm{argmax}_{p \in \mathscr{P}} V^{p\xi}(h_k)$$

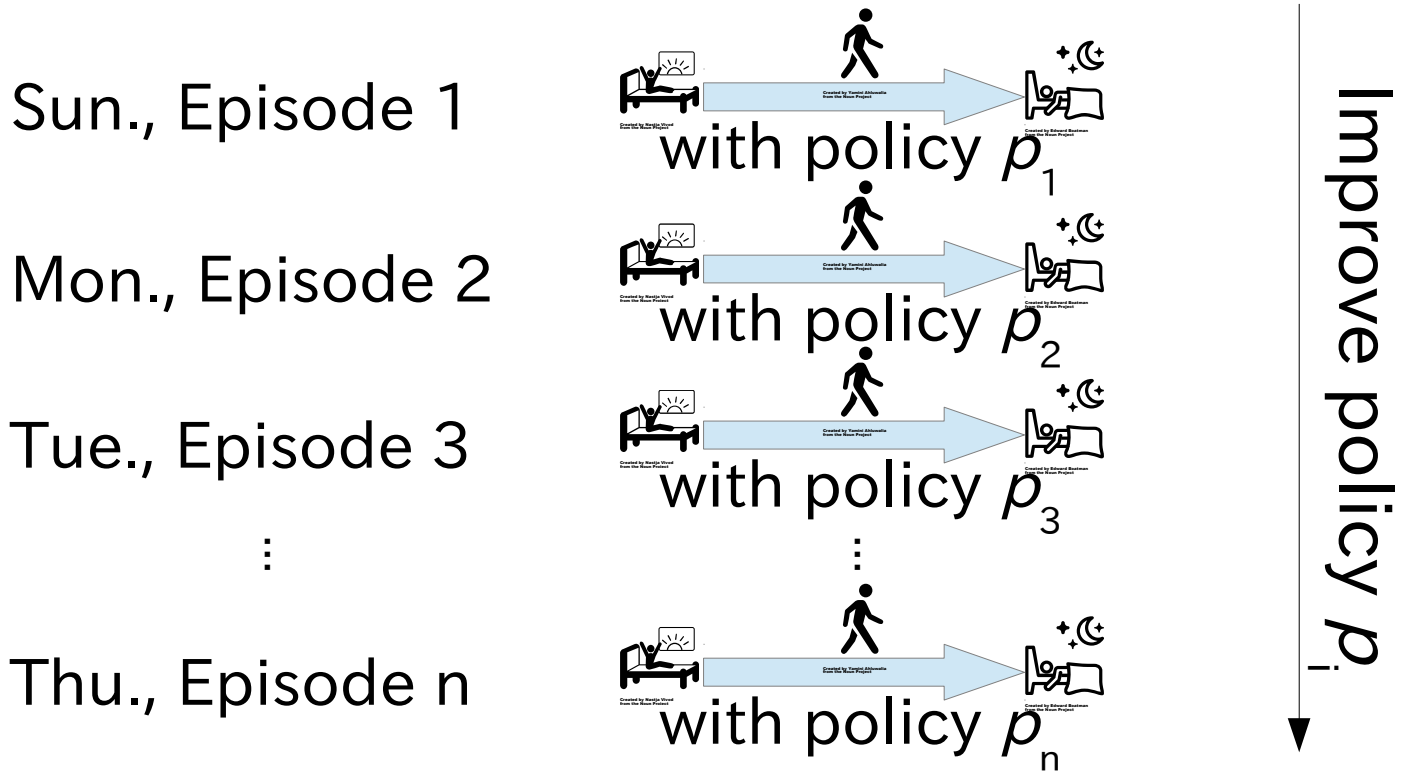
$$\dot{a}_k = p_k(h_k)$$

This is BAD, because ...

- $\mathrm{argmax}_{p \in \mathscr{P}} \ldots$ at each step can be costly.
- p is undecidable because it has many candidates, because there is no restriction (other than $\mathscr{P}$) on $p(h)$ for unknown h's.

Episodic environment assumption

Same, consistent program (as policy) within each episode, which improves for each episode.



NB: No learning within each episode.
This is OK. We may not learn a *skill* within a day, but our policy exploits a solution when found.

Conjecture on the Best Policy

For all history $h \in \mathscr{H}$ and set of policies $\mathscr{P} \subset \mathscr{C}$ there exists $p \in \mathscr{C}$ s.t.

$$V^{p\xi}(h) = \max_{p' \in \mathscr{P}} V^{p'\xi}(h)$$

Proof sketch:

p can case over *h* and return the maximal *p*'(*h*)

Selecting the policies stochastically

- V can be learned asymptotically.
- While learning V, policies should be first exploratory then deterministic.

  This can (hopefully) be achieved by roulette selection over $\frac{1}{d(V,\hat{V})^2}$

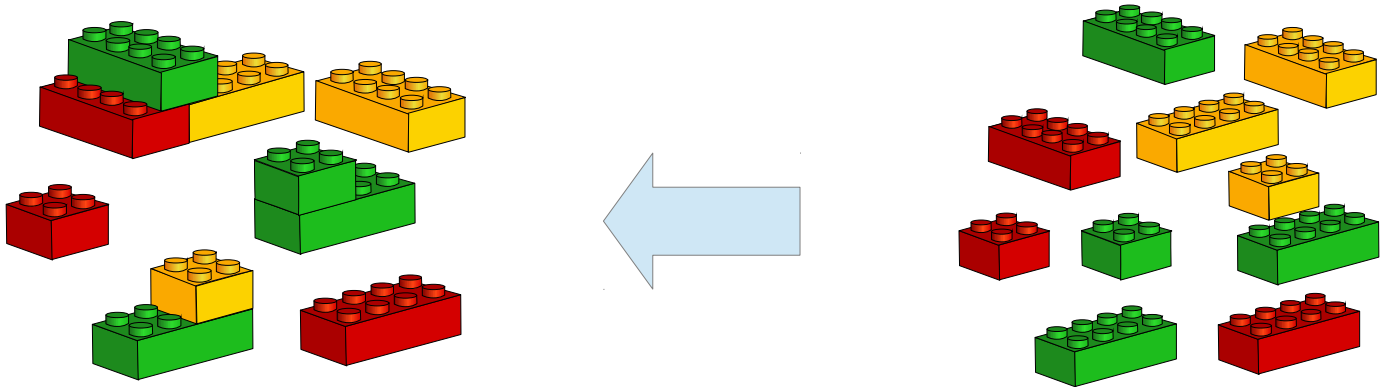
  where d is the L^2 distance
- Unknown policies should not be tried until time $\frac{1}{M(p)}$

  where $M(p)$ is its universal prior of p

  (defined based on the Haskell grammar).

Imitation

Reuse the incrementally-learned component library for the environment model.



Synthesized policies

Component library
for the environment model

Then, the agent's action will consist of what it sees!