Modular ("agent-agnostic") Human-in-the-loop RL

Owain Evans
University of Oxford

Collaborators

David Abel
(Brown)

Andreas Stuhlmueller
(Stanford)

John Salvatier
(Oxford)

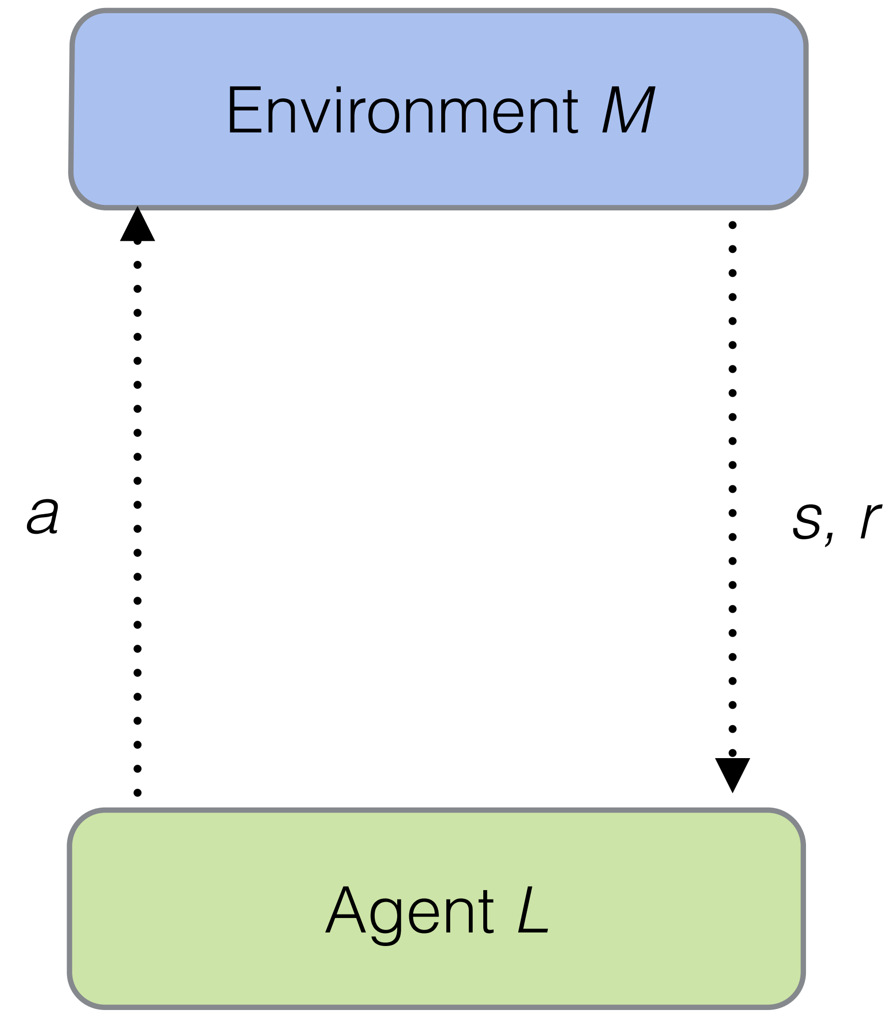
Overview

1. Autonomous vs. human-controlled / interactive RL
2. Framework for interactive RL
3. Applications of framework: reward shaping and simulations.
4. Case study: prevent catastrophes without side-effects.

Overview

1. **Autonomous vs. human-controlled / interactive RL**
2. Framework for interactive RL
3. Applications of our framework: reward shaping and simulations.
4. Case study: prevent catastrophes without side-effects.

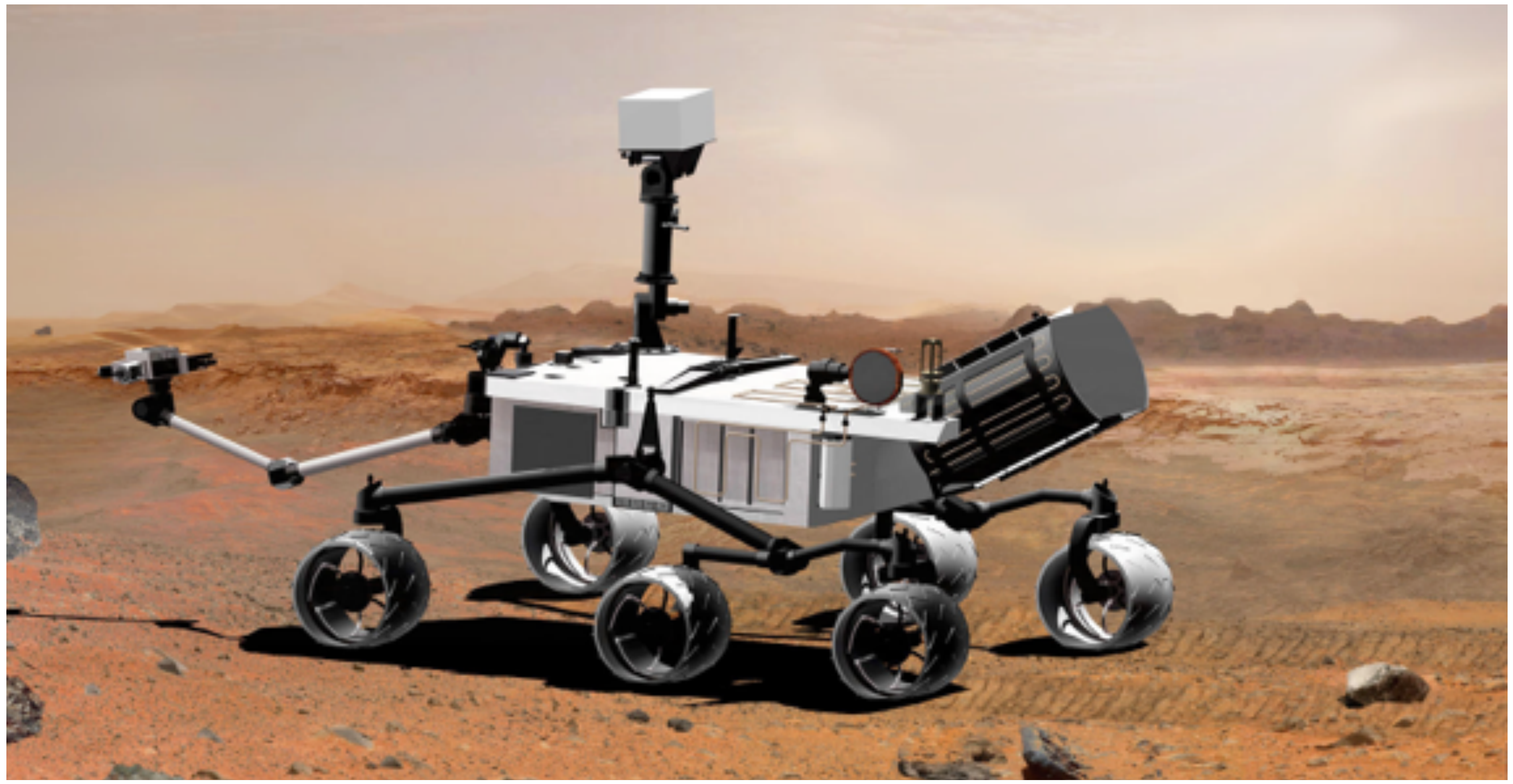
Standard RL picture

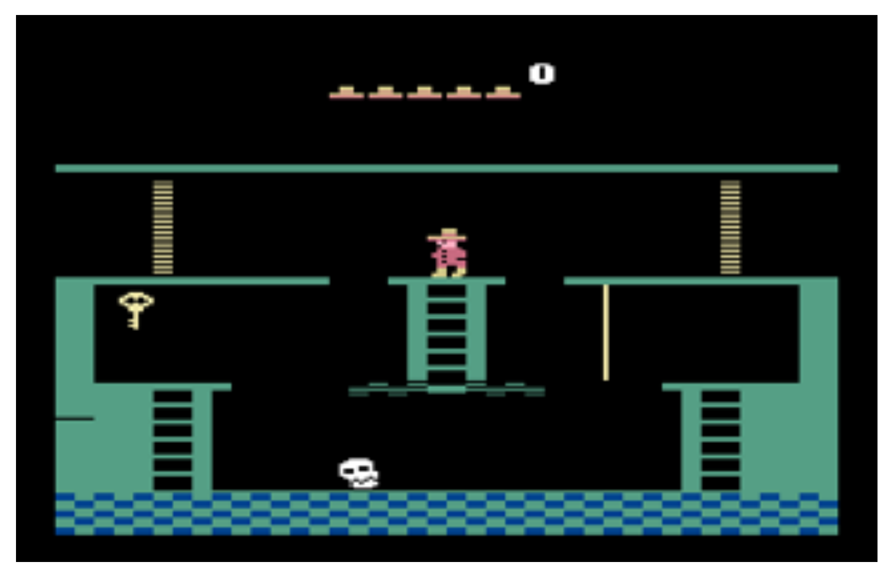


Two contrasting research programs for RL:

A. Autonomous RL
B. Interactive RL

Picture A: Autonomous RL (Deepmind et al.)





Picture A: Autonomous RL (Deepmind et al.)

1. ML researcher designs generic RL agent

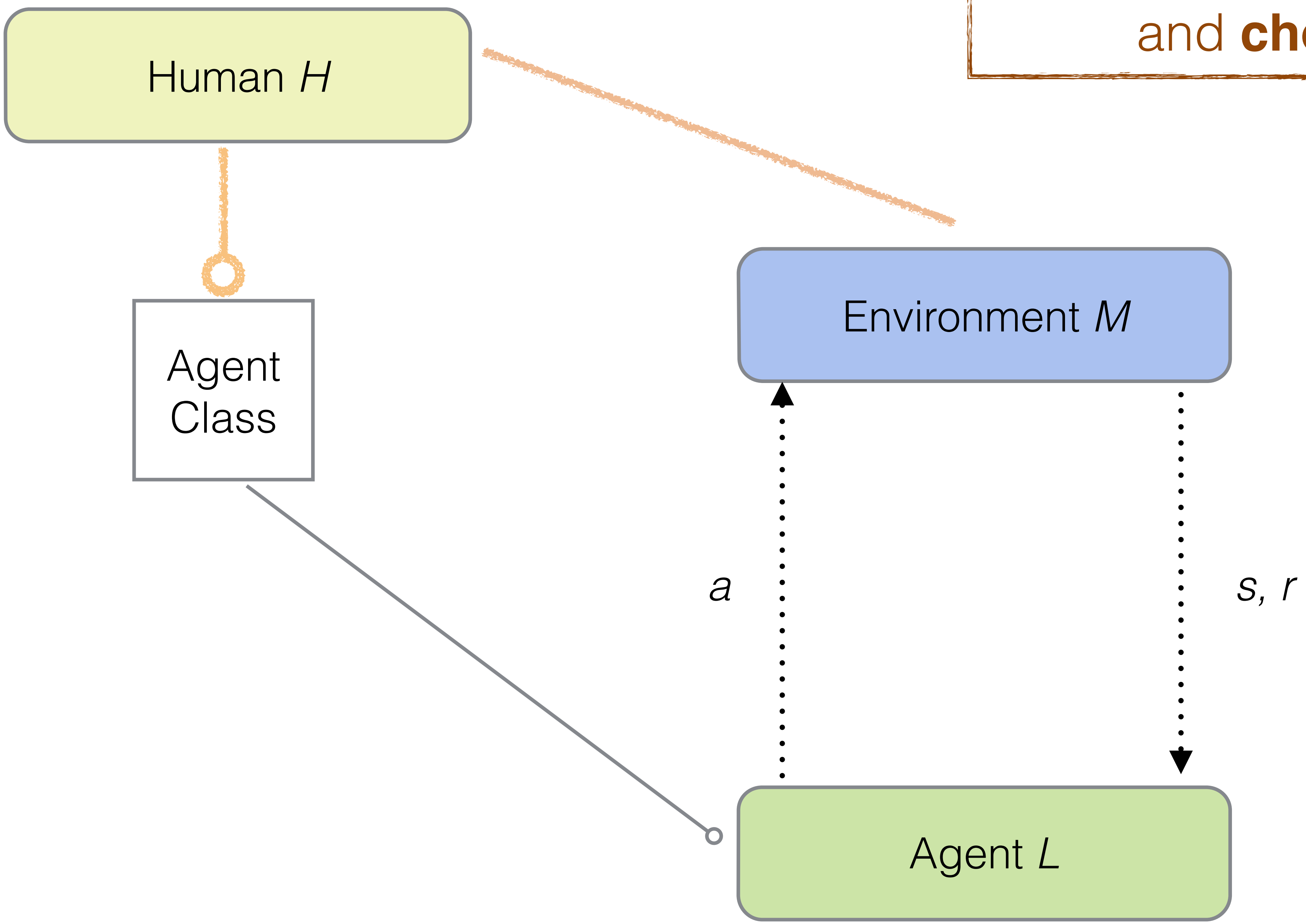
2. Real-world environment and sparse rewards

3. Autonomous learning (no human intervention)

4. Motivation: pragmatic (hand-engineering doesn't scale), biological (animals can learn autonomously).

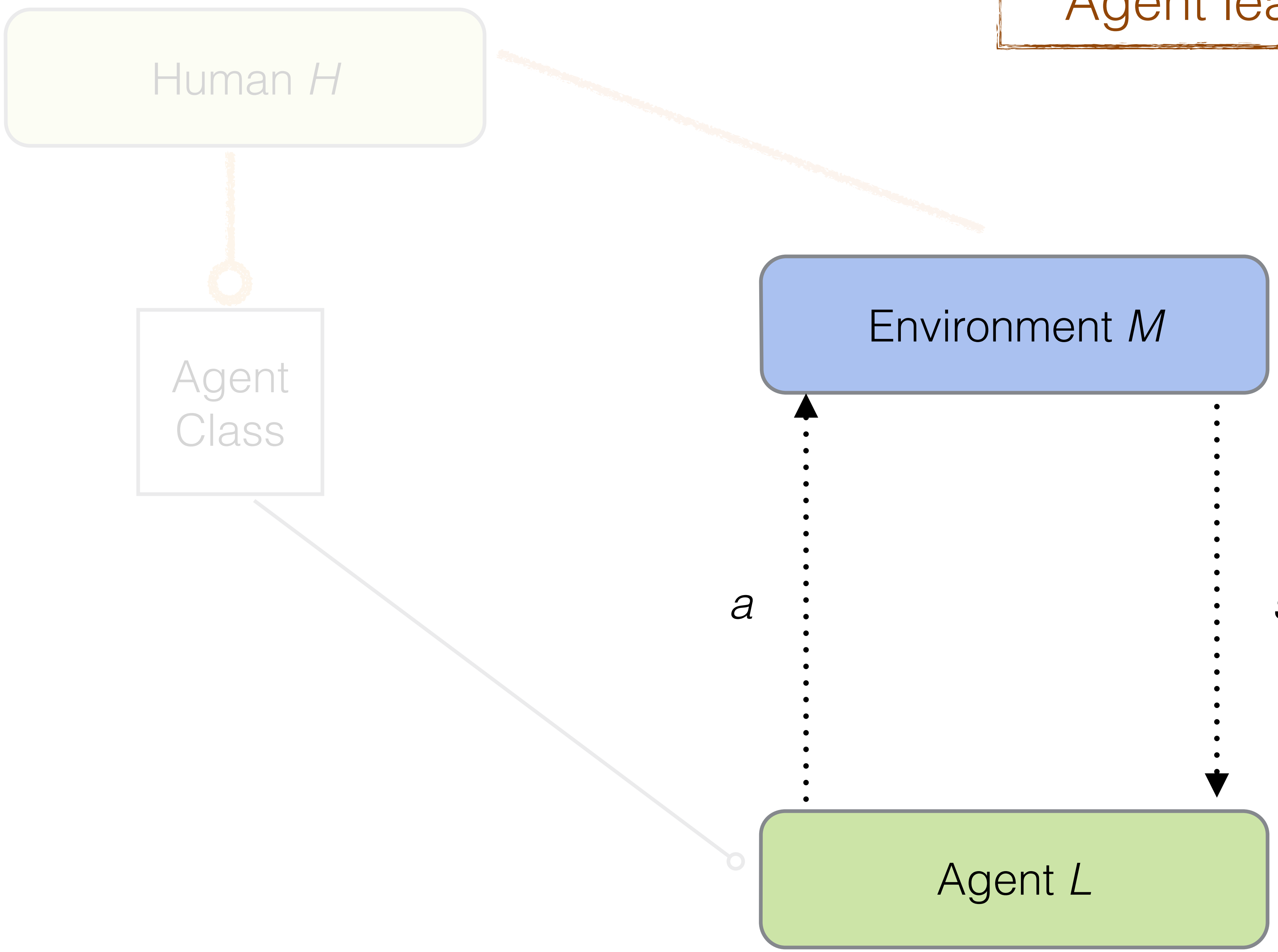
A: Autonomous RL (Deepmind et al.)

Human **designs** Agent Class
and **chooses** Env.



A: Autonomous RL (Deepmind et al.)

Agent learns autonomously



Picture B: human-controlled, human-shaped RL

Picture B: human-controlled, human-shaped RL

Motivation for Interactive RL: useful RL systems should be **safe**, **value-aligned**, **interpretable**

1. Fine-grained **rewards**: reward function or by demonstration (IRL or Apprenticeship).

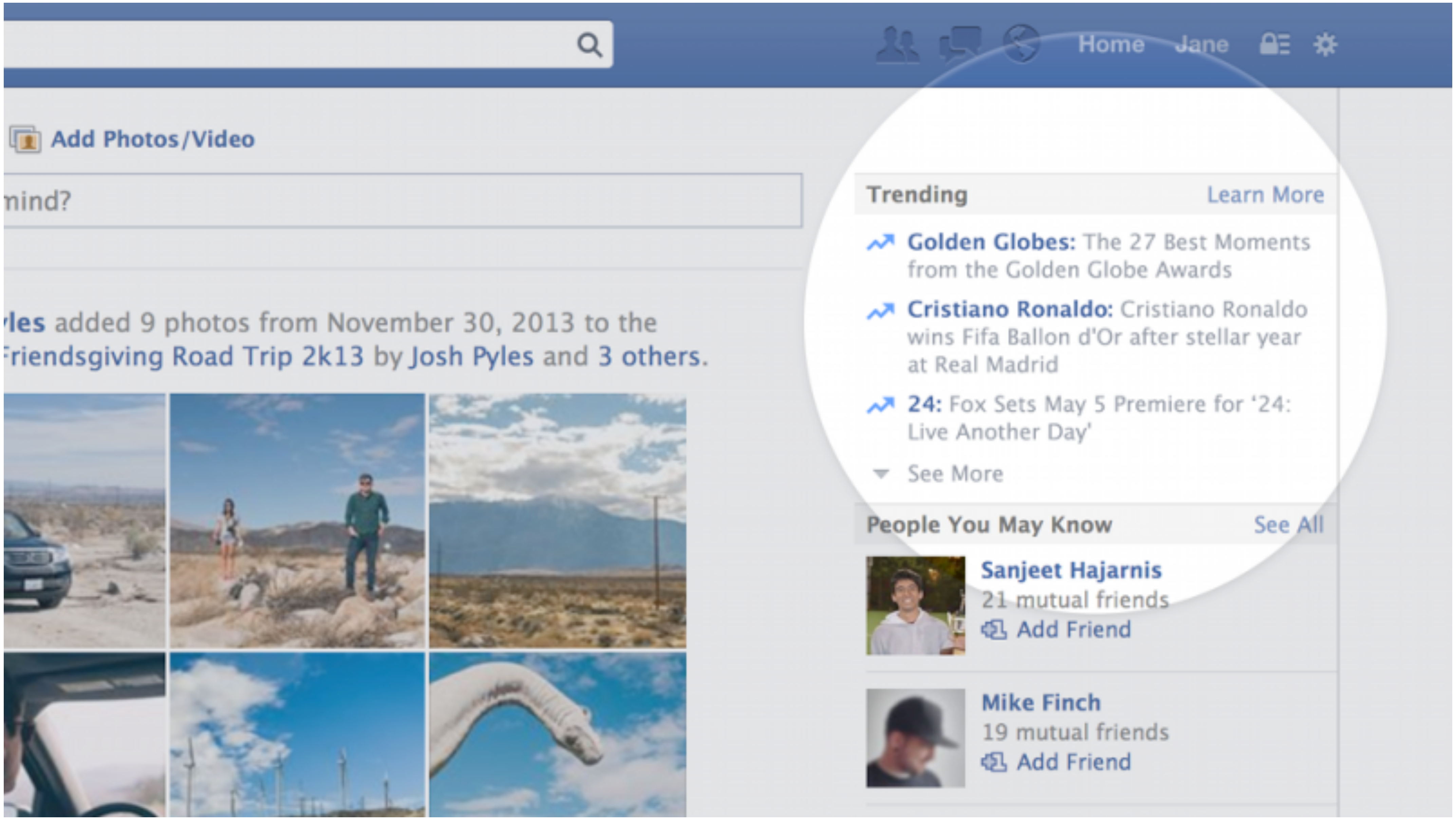
2. Human designs **curriculum**: simulations, practice environment, sequence of real-world environments.

3. Human can intervene during **learning** (human-in-loop)

Picture B: human-controlled, human-shaped RL

Picture B: human-controlled, human-shaped RL

Picture B: human-controlled, human-shaped RL



Overview

1. Autonomous vs. human-controlled / interactive RL

2. **Framework for interactive RL**

3. Applications of our framework: reward shaping and simulations.

4. Case study: prevent catastrophes without side-effects.

Framework for interactive RL

Lots of techniques for integrating human into RL system

- reward design/shaping as in TAMER, Active Reward Learning (Knox and Stone 2008, Daniel et al. 2014)
- avoid catastrophes by biasing training distribution (Frank et al. 2008, Paul et al. 2016)
- provide online advice about Q-values, policy (Thomaz et al 2016, Torrey et al 2013, Loftin et al 2014)

Framework for interactive RL

Current work: Lots of techniques for integrating human into RL system

GOAL: specify existing techniques in common/unified framework

Benefit: Easier to **analyse**, to **generalize** and to **compose** techniques.
(AI Safety: interested in abstract properties of techniques.)

Framework for interactive RL

Current work: Lots of techniques for integrating human into RL system

GOAL: specify existing techniques in common/unified framework

SUB-GOAL: framework should abstract away details of agent's algorithm (modular or "agent-agnostic")

Overview

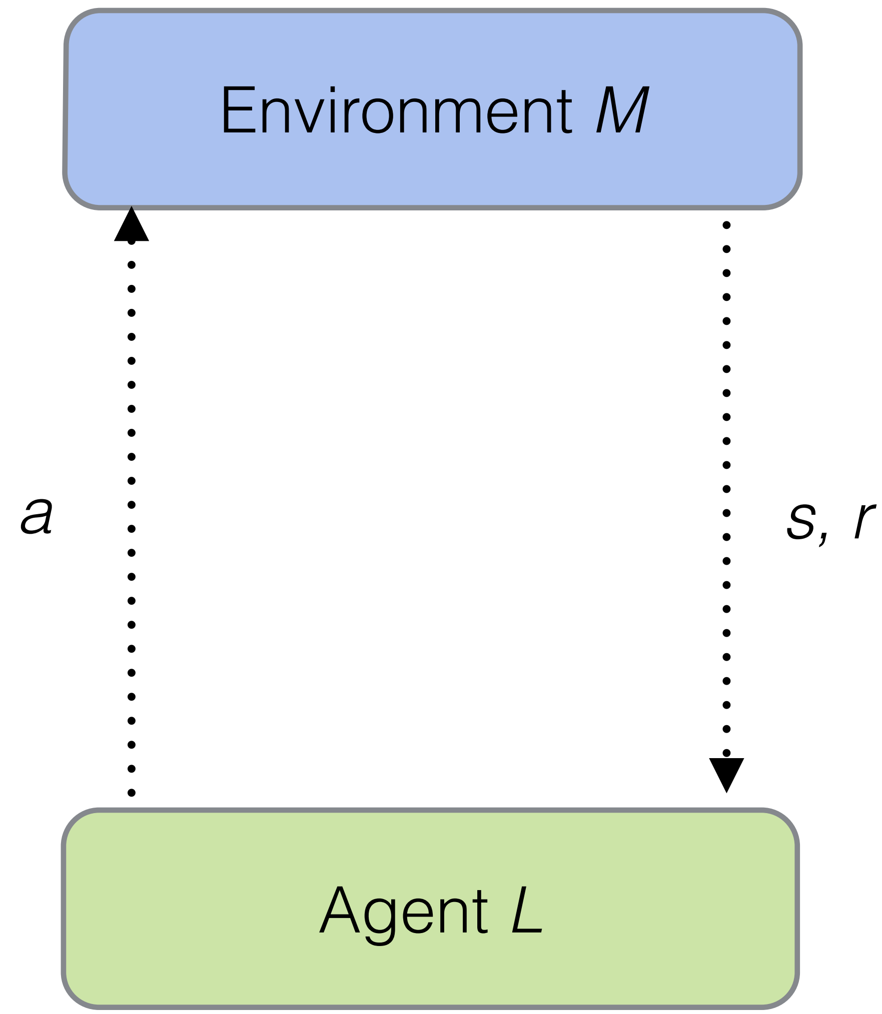
1. Autonomous vs. human-controlled / interactive RL

2. Framework for interactive RL

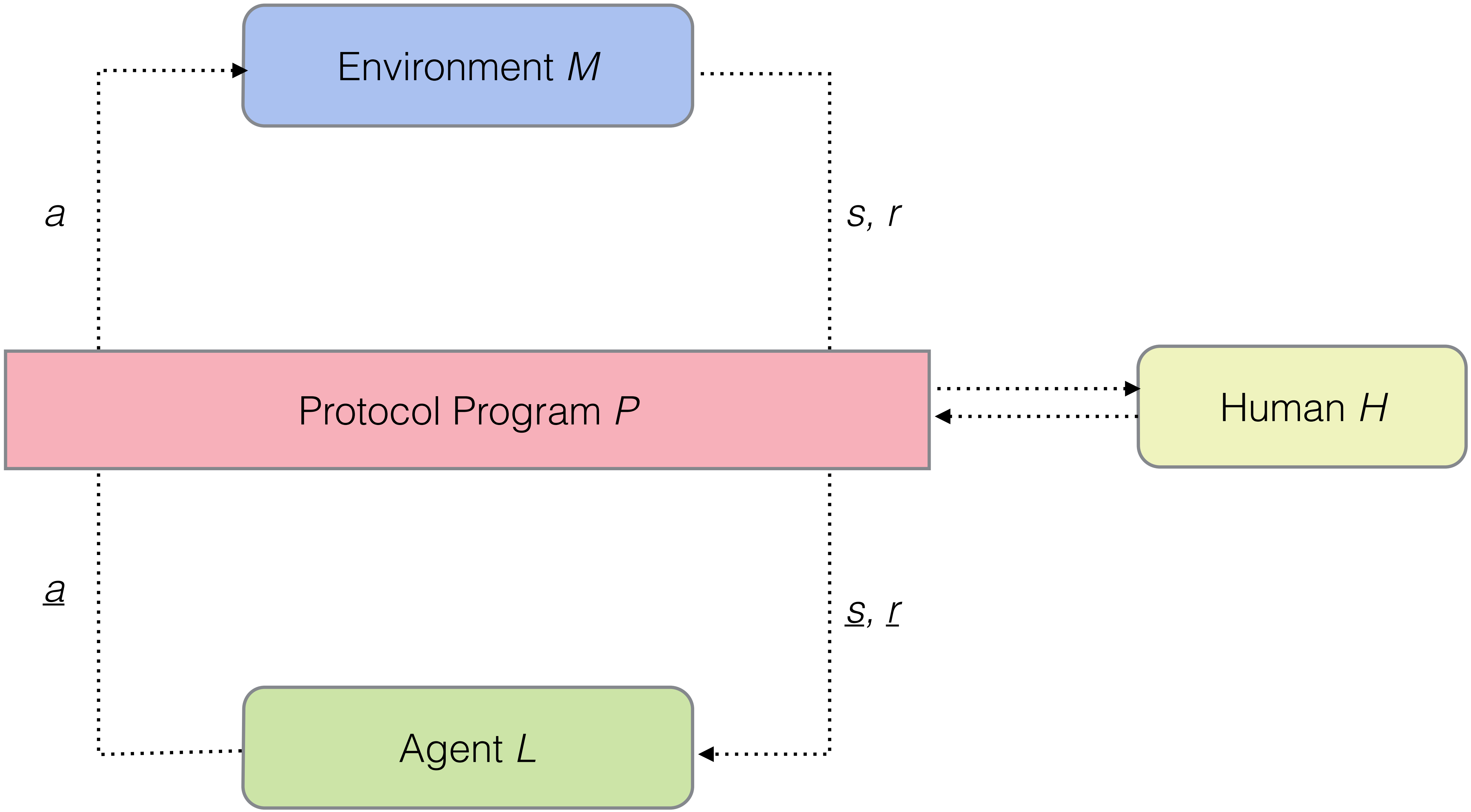
3. **Applications of our framework: reward shaping and simulations.**

4. Case study: prevent catastrophes without side-effects.

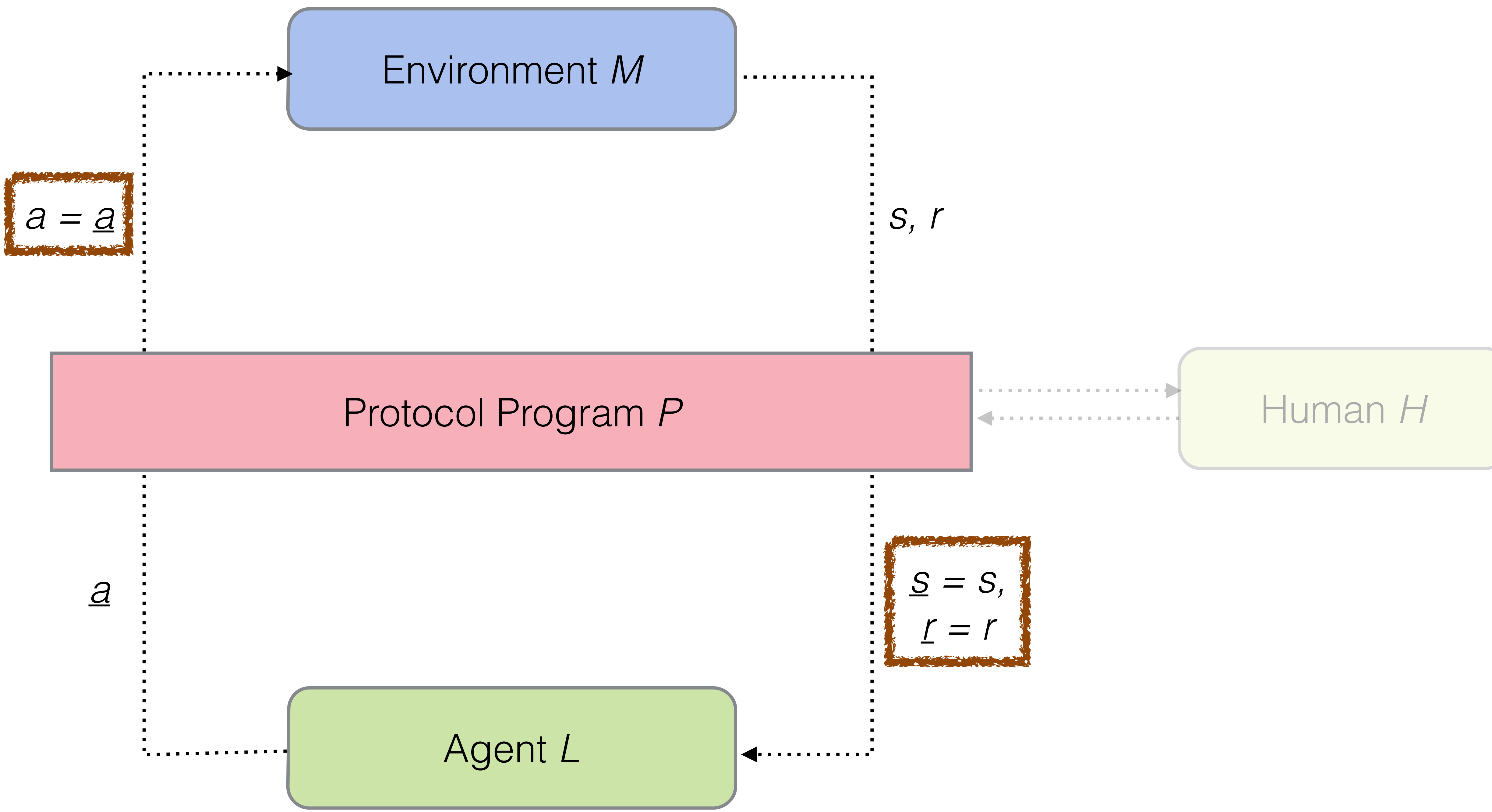
Standard RL



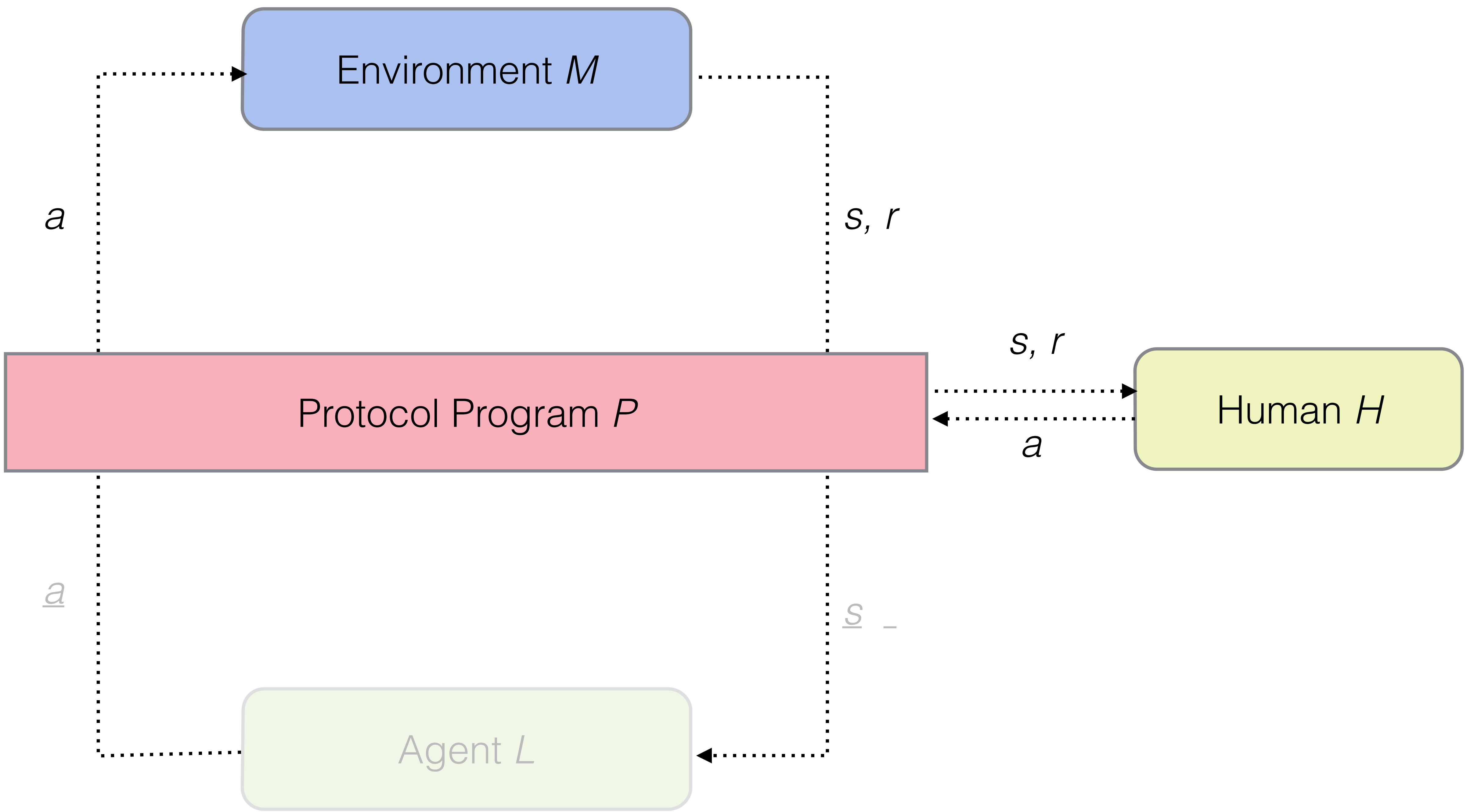
Interactive Framework



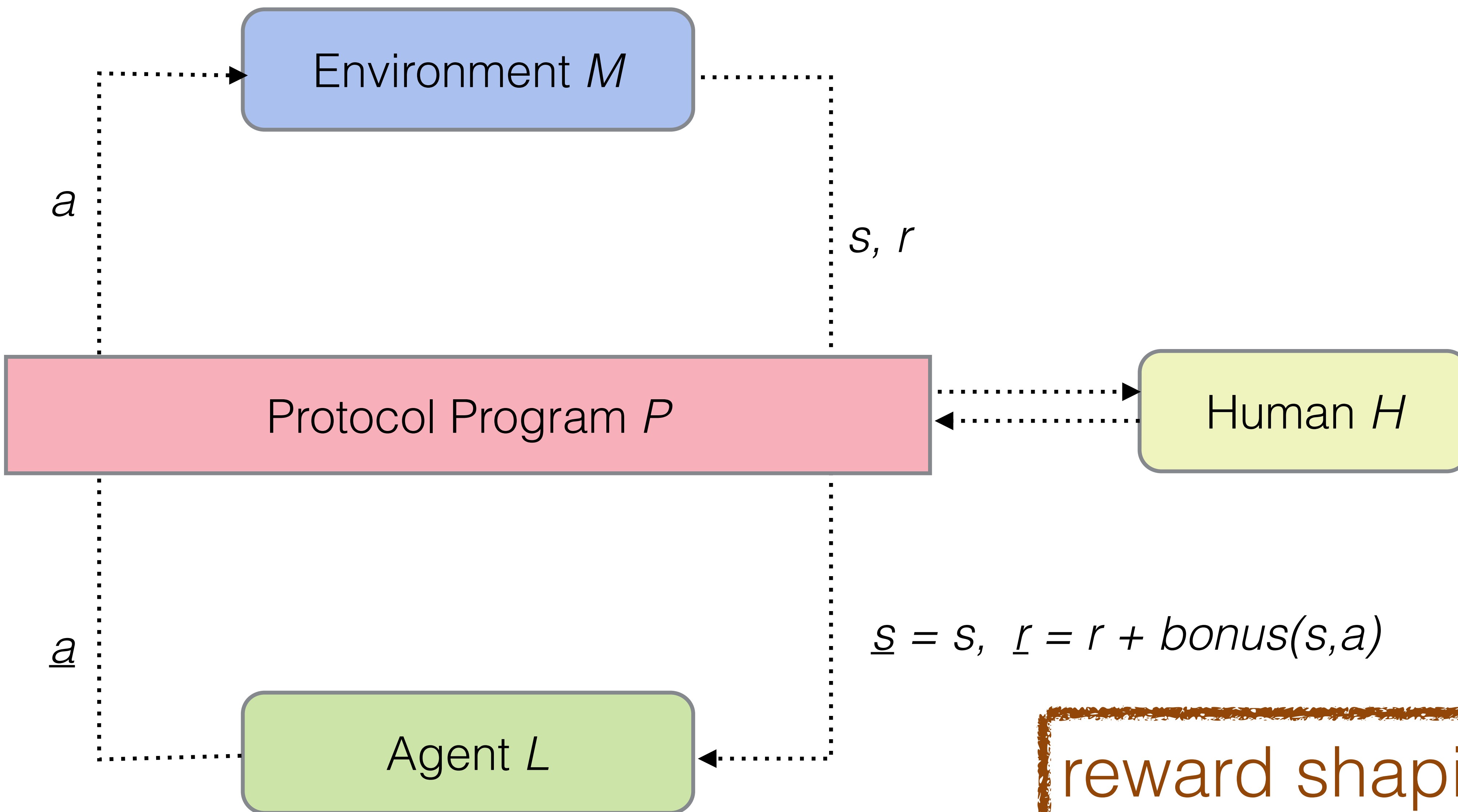
Example 1: Standard RL



Example 2: Human control

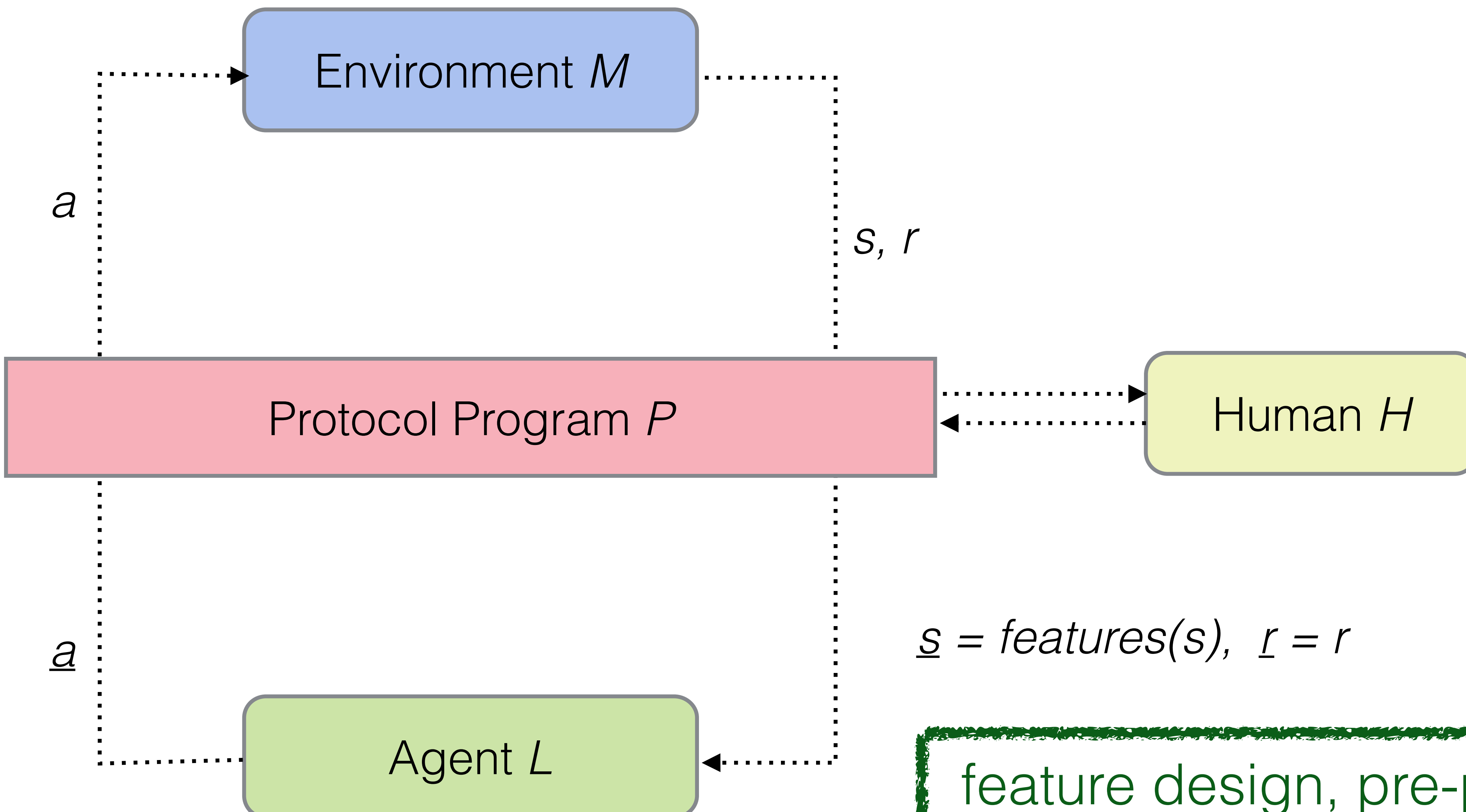


Overview of protocols



reward shaping

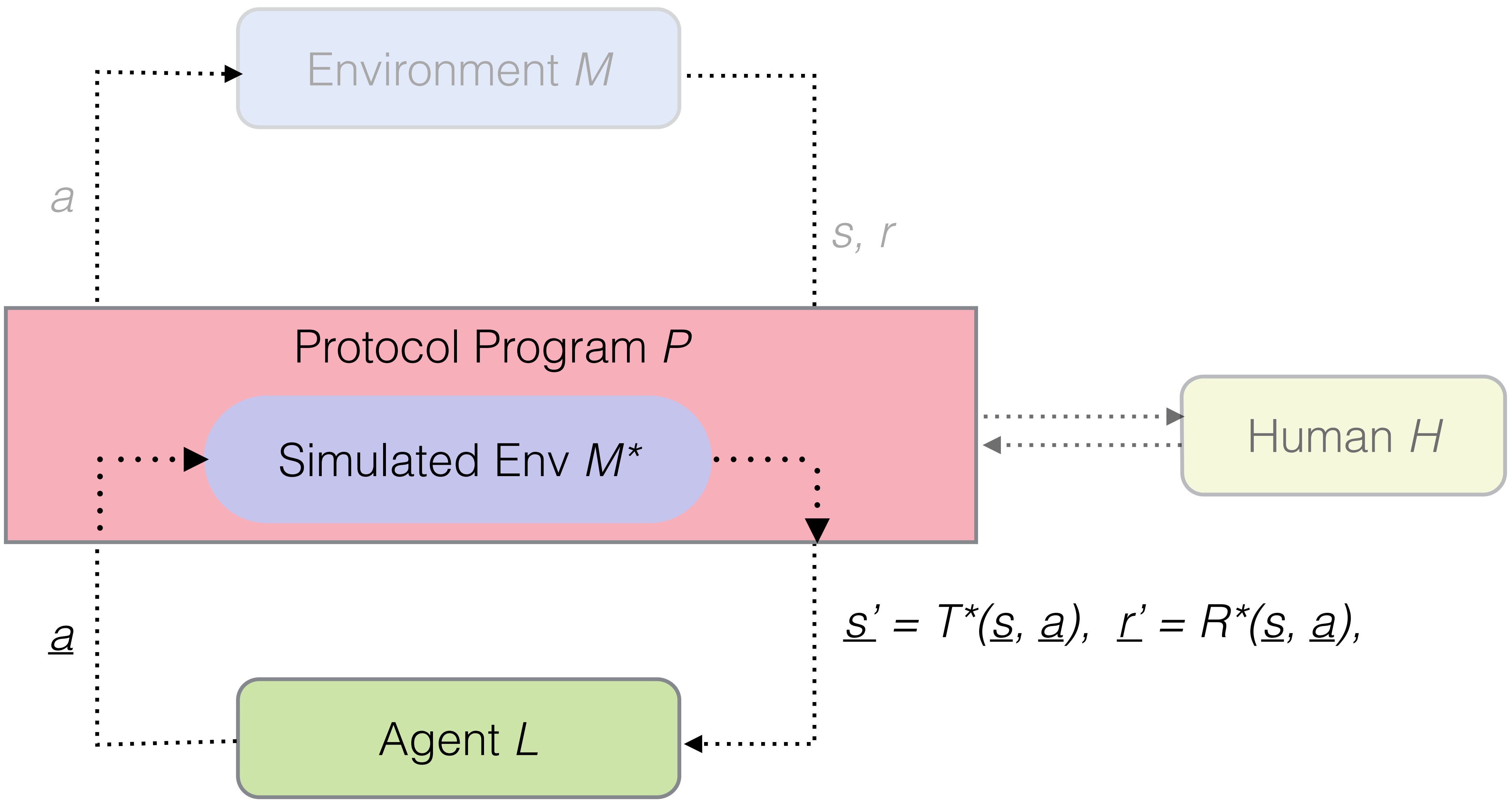
Overview of protocols



$\underline{s}$ = *features(s)*, $\underline{r}$ = *r*

feature design, pre-processing

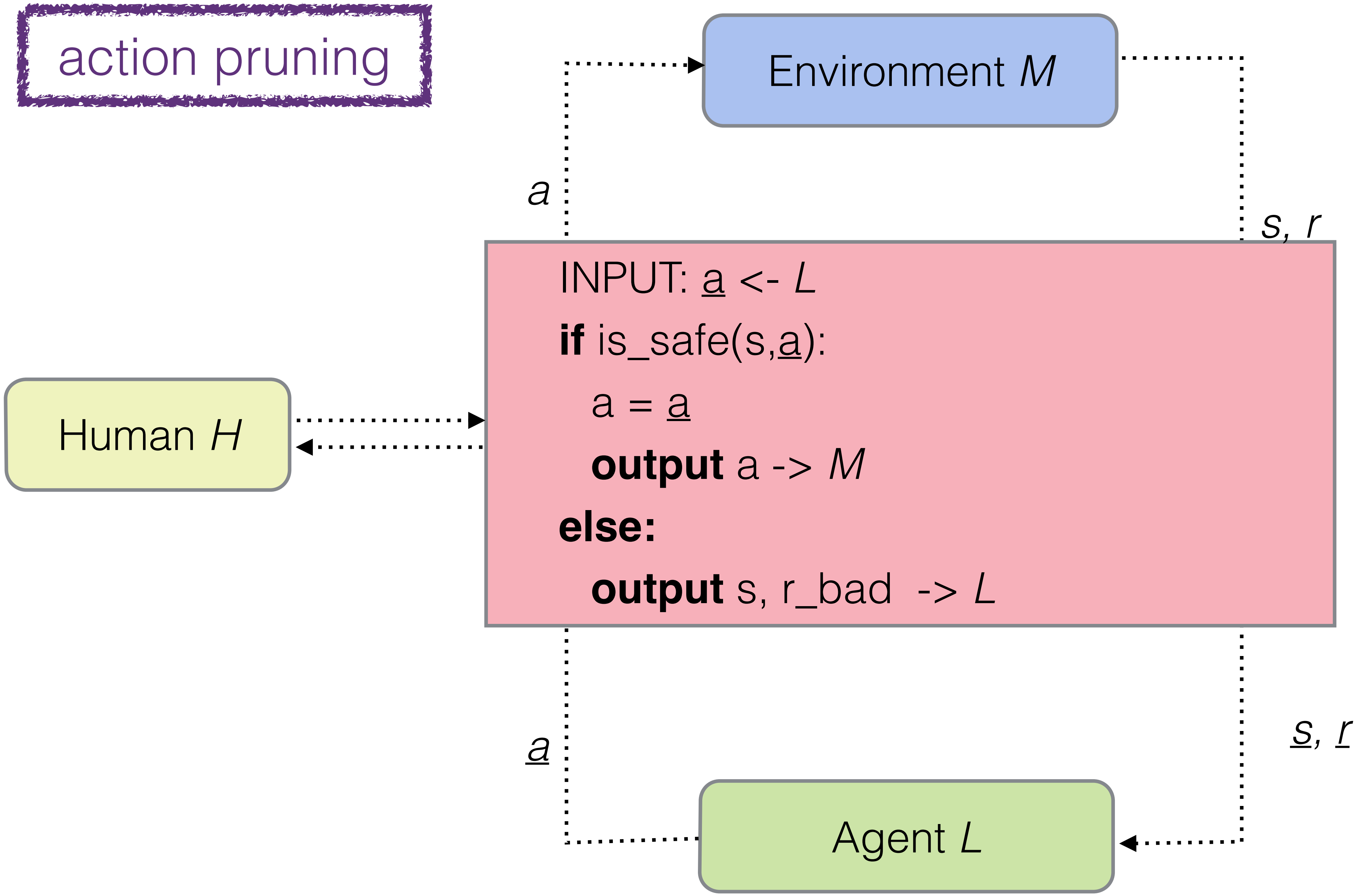
Overview of protocols



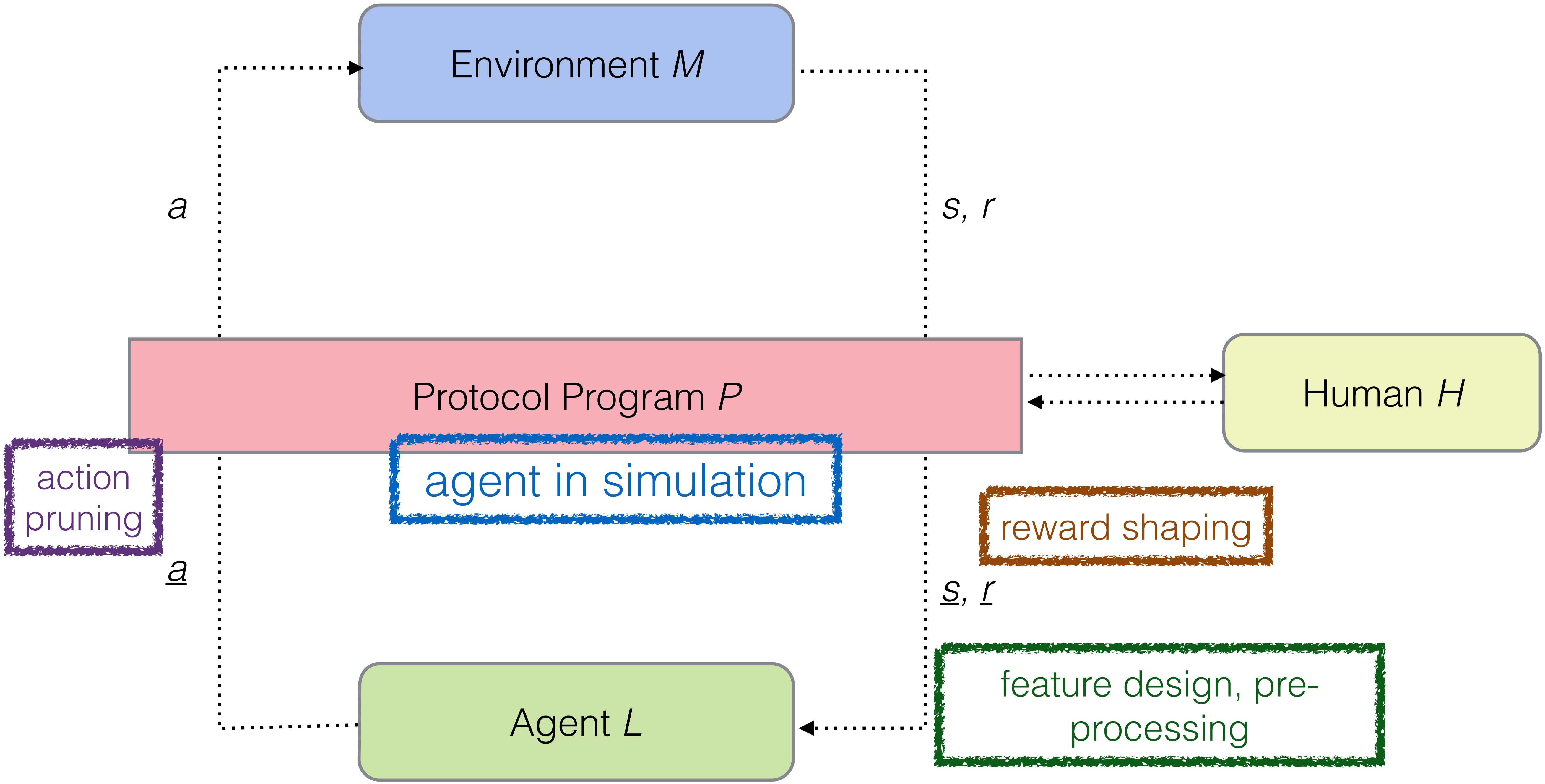
agent in simulation

Overview of protocols

action pruning



Overview of protocols



Overview

1. Autonomous vs. human-controlled/interactive RL
2. Framework for interactive RL
3. Applications of our framework: reward shaping and simulations.
4. **Case study: prevent catastrophes without side-effects.**


Credit Card
0001 4492 0781 7376

Prevent Catastrophes with Interactive RL

(Informally) A state-action (s,a) is catastrophic (w.r.t. $\epsilon > 0$) if:

Human requires that: *P(“*agent does *(s,a)”)* $< \epsilon$

Prevent Catastrophes with Interactive RL

(Informally) A state-action (s,a) is catastrophic (w.r.t. $\epsilon > 0$) if:

Human requires that: *P(“*agent does *(s,a)”)* $< \epsilon$

Examples:

- irreversible damage to property (robot destroys itself)
- breaking laws / moral rules
- physically harm humans
- manipulate or psychologically harm humans

Prevent Catastrophes with Interactive RL

Related work: Safe RL and avoiding SREs (Moldovan and Abeel, Frank et al., Paul et. al, Lipton et al.)

Challenge:

- Simulation often inadequate (esp. for extreme events)
- RL agents learn by **trial** and **error** (don't know R and T in advance)
- Solution: human blocks catastrophes **before** they happen

Prevent Catastrophes with Interactive RL

Assumptions:

- agent interacts with real-world environment
- agent might try catastrophic actions (e.g. previous training was insufficient)
- human recognizes catastrophic actions **before** outcome

Prevent Catastrophes with Interactive RL

Goal: Find protocol program with following properties

1. For given ϵ, *P("catastrophe")* $< \epsilon$, and *complexity*(ϵ) scales well.

2. Minimize negative side-effects on agent's learning: e.g. agent still converges to optimal policy (at same rate).

3. Requirements on human (time and knowledge) are feasible.

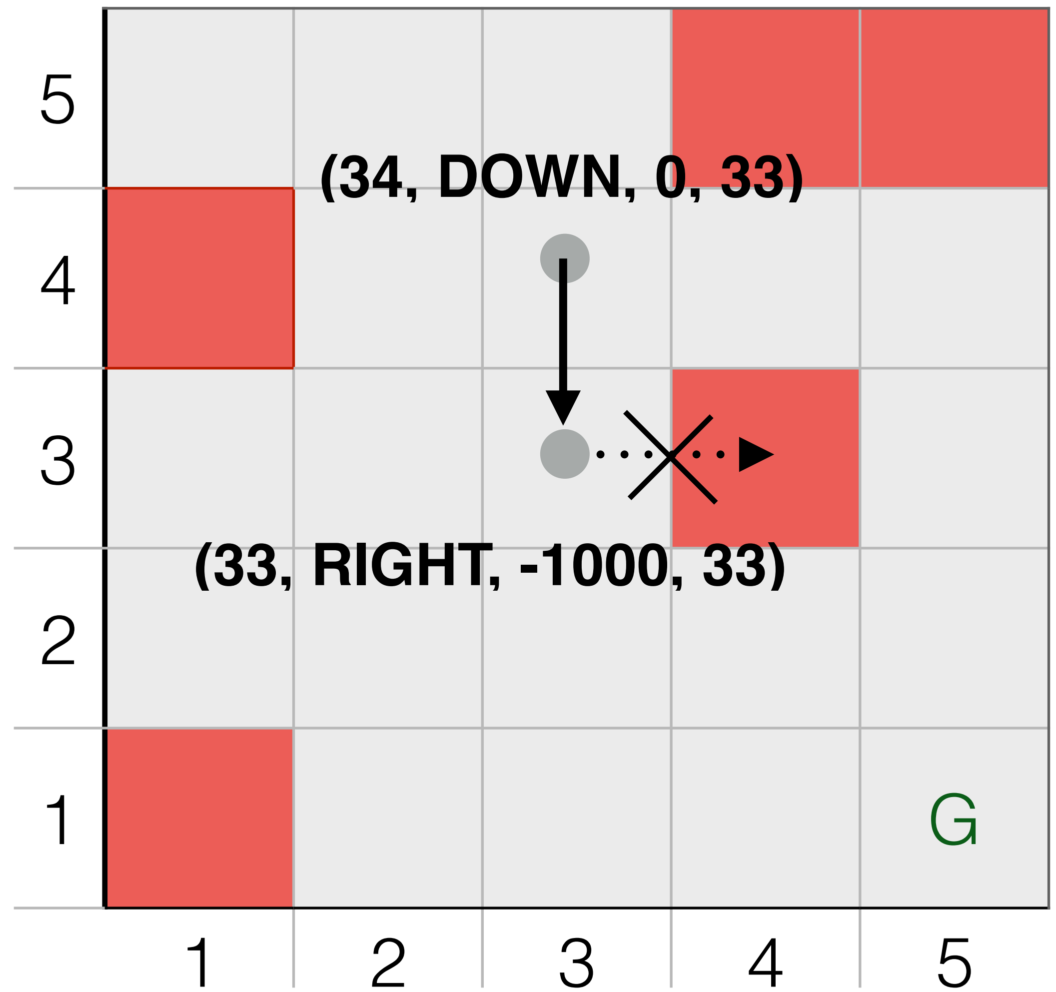
Pruning protocol program for discrete MDPs

1. Agent tries *(s,a) = (33,RIGHT)*

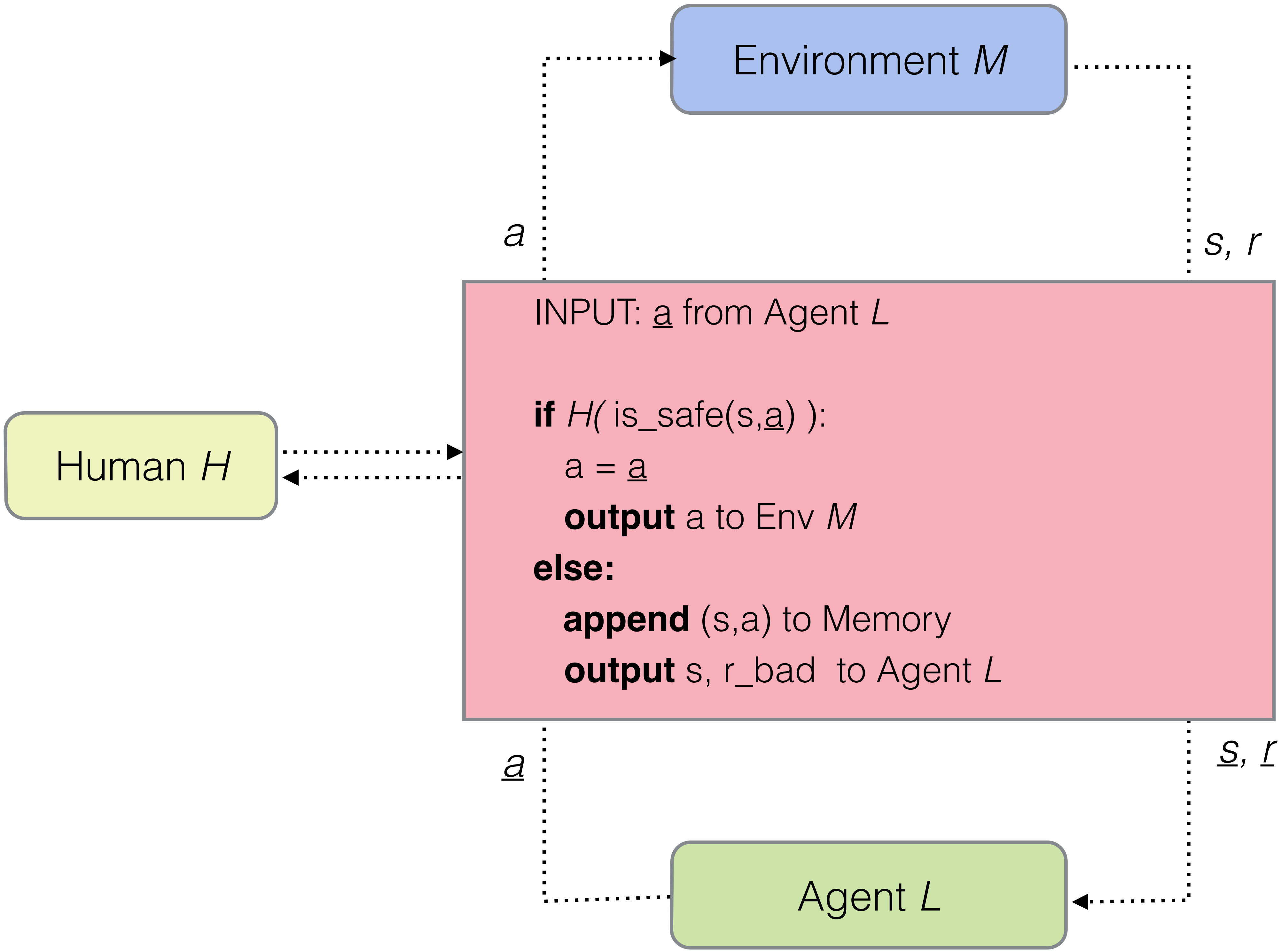
2. *H* judges *(s,a)* to be unsafe.

3. *H* blocks *(s,a)* and appends to memory.

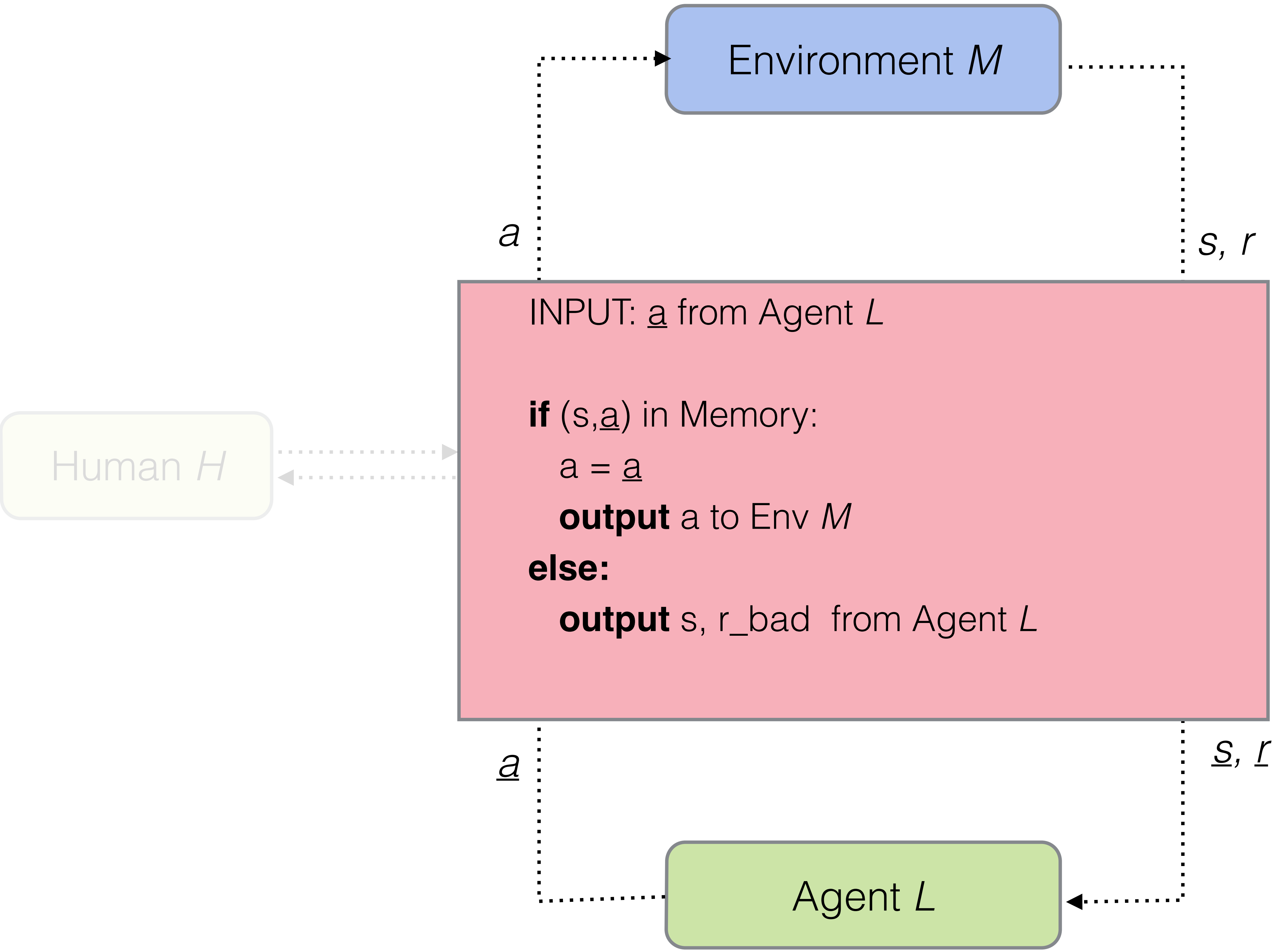
4. Agent receives *(33, RIGHT, -1000, 33)*.



Action Pruning 1: human evaluates



Action Pruning 2: imitator evaluates



Prevent Catastrophes with Human Overseer

Why do we need an imitator to block the agent?

1. Some RL algorithms would keep trying catastrophic action. For example: epsilon-greedy, RMAX.

2. Note: no guarantee that agent *learns* to avoid catastrophes.

Goal: Find protocol program with following properties

1. Given ϵ, *P("catastrophe")* $< \epsilon$, and *complexity*(ϵ) scales well. ✓

2. Minimize negative side-effects on agent's learning: agent still converges to optimal policy (at same rate). ✓
   - Transition function is modified but still learnable.

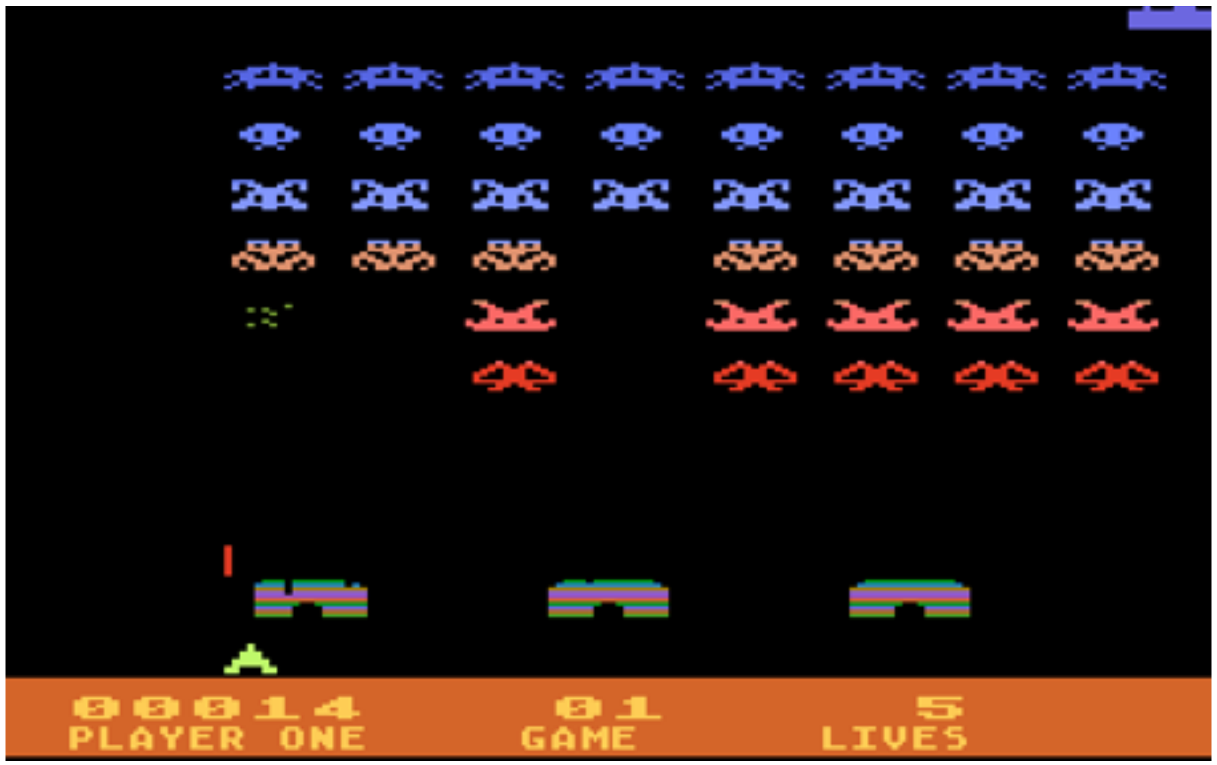
3. Requirements on human (time and knowledge) are feasible.
   - Human can retire when each (s,a) tried once. ?
   - Human can't make mistakes. ?

Result from paper

Idea: Suppose human identifies the (avoidable) catastrophic actions but otherwise can't tell which actions are better. Then all catastrophic actions are blocked and agent still learns optimal policy.

Result: Suppose that human *H* has only a ß-optimal *Q*-function Q_h. There exists a protocol program s.t. (1) agent never takes an action more than 4ß from optimal action, (2) no optimal actions are pruned.

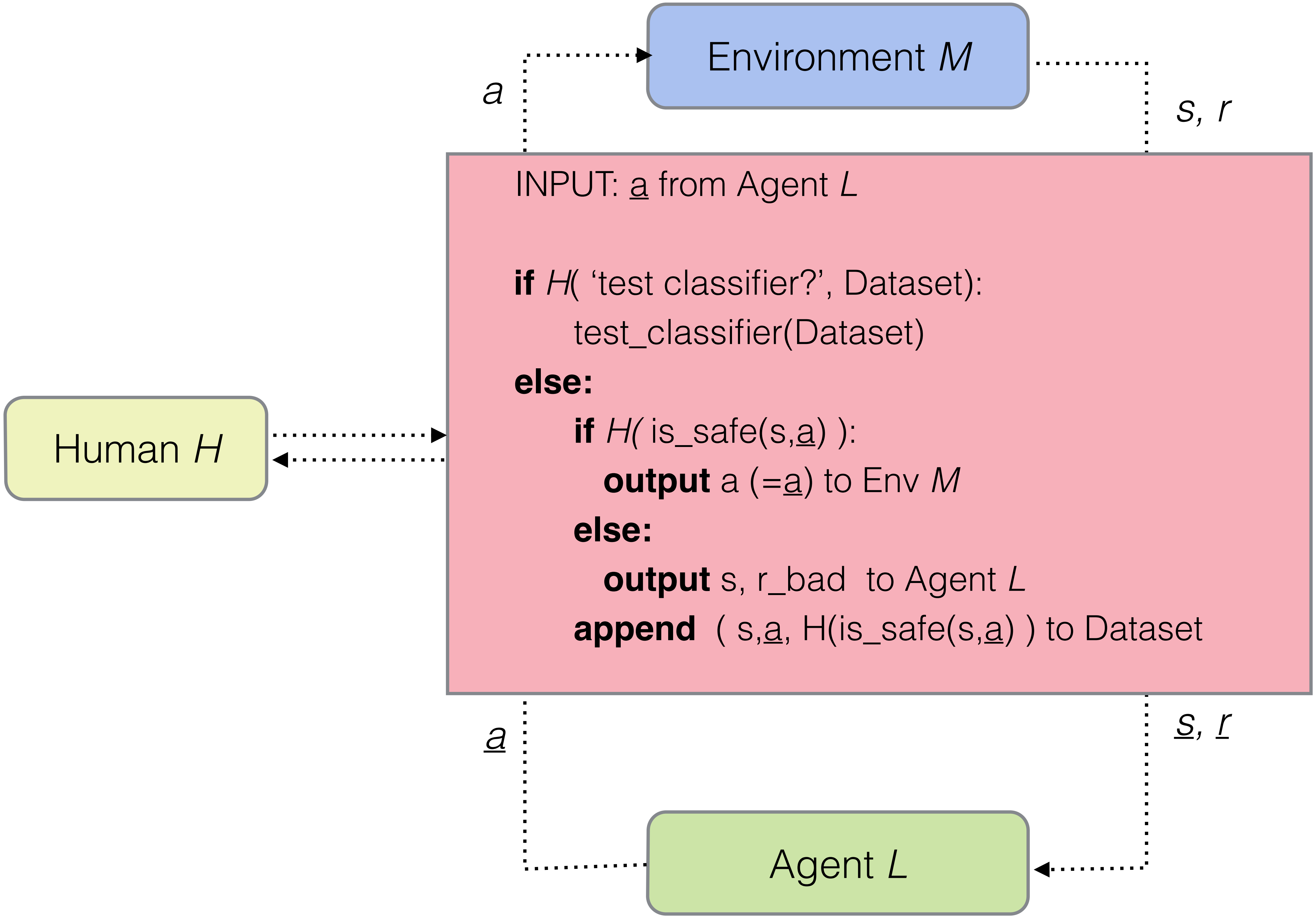
Prevent Catastrophes in Continuous MDPs



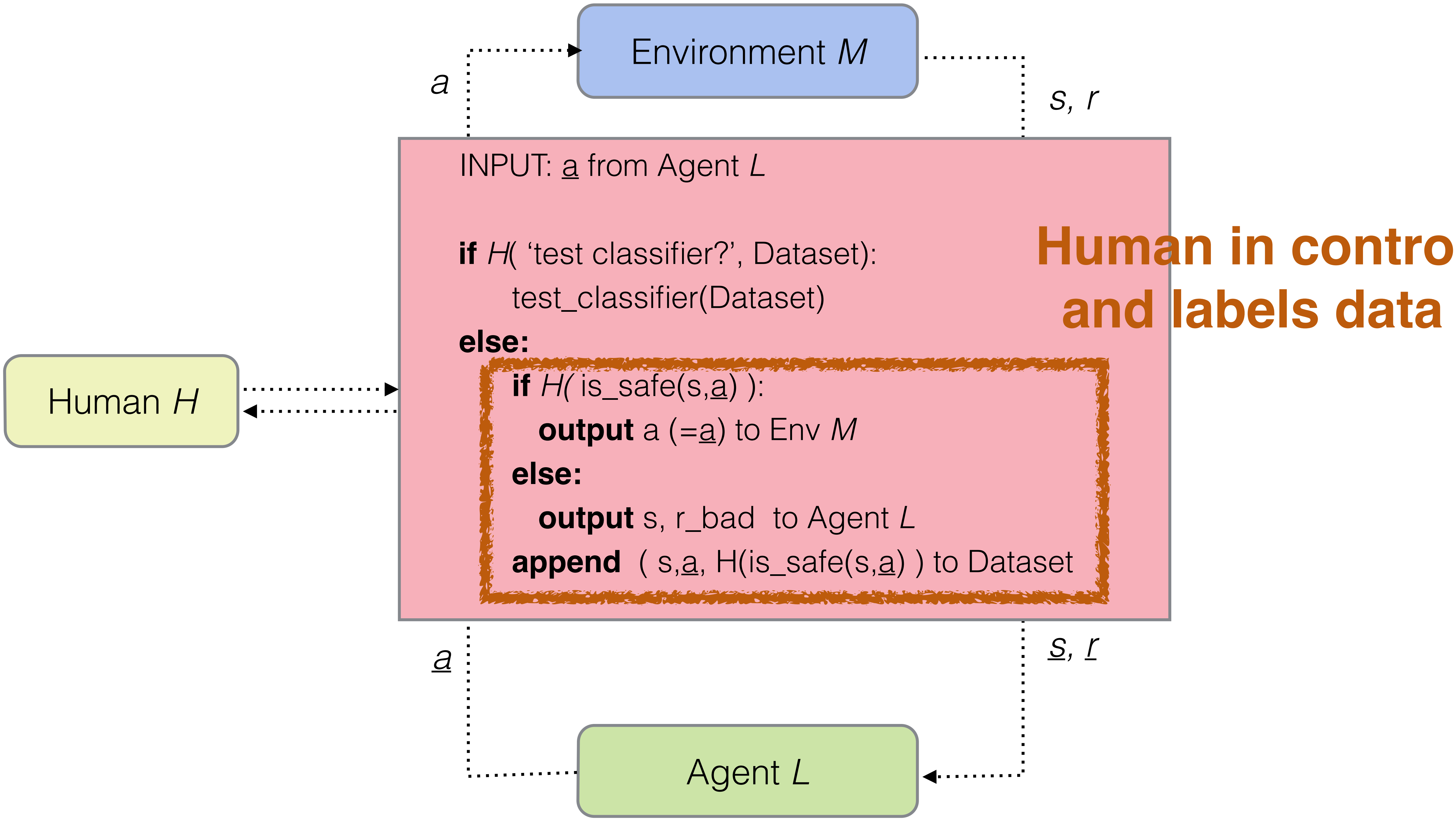
Prevent Catastrophes in Continuous MDPs

- With large/infinite state space, human cannot label all catastrophic actions.
- Instead humans trains a classifier on data labeled either “catastrophic” or “not”.
- If classifier succeeds on held-out test set, human retires.

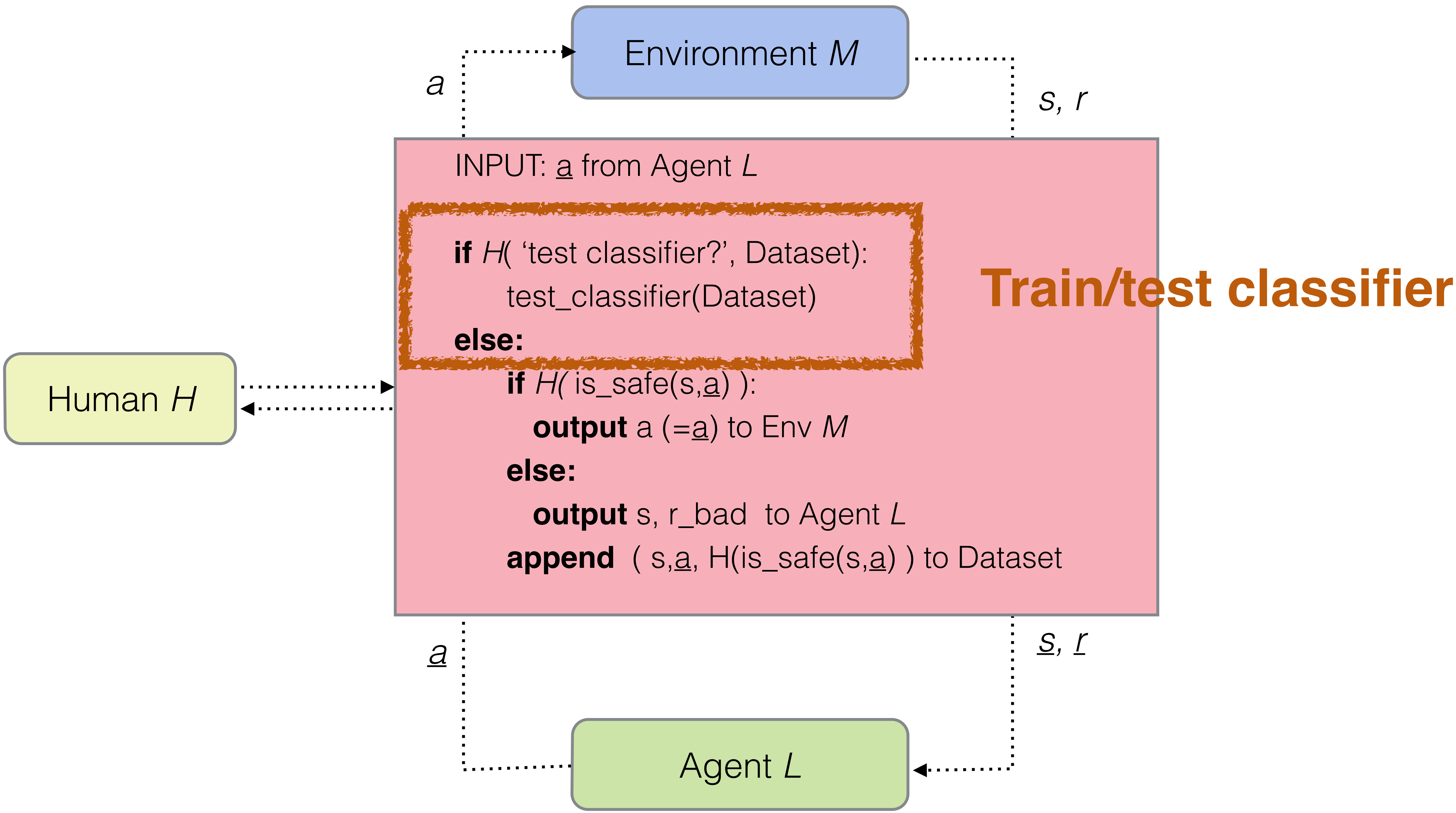
Continuous action pruning: overview



Continuous action pruning: overview



Continuous action pruning: overview



Generalizing about catastrophes

- Need classifier that mislabels a catastrophe with prob $< \epsilon$.
- Dataset must contain all classes of catastrophe.
- Standard problem with RL:
  samples (s, a, *H*(is_safe(s,a)) are not iid.

Generalizing about catastrophes

- Need classifier that mislabels a catastrophe with prob $< \epsilon$ (as in PAC or SLT guarantees).

- Standard problem with RL:
samples (s, a, *H*(is_safe(s,a)) are not iid.

- **Robust generalization**: avoid overconfident errors out of sample.

Generalizing about catastrophes

- **Robust generalization**: avoid overconfident errors out of sample.
- Special case: no false negatives but false positives acceptable, can ask human when uncertain (asynchronously).
- Human can block broader, easier-to-learn class of actions.

Generalizing about catastrophes

Problem 2: Requires agent to visit each kind of catastrophic state (SLOW).

After catastrophic (s,a) has been observed. Human adds noise to it (without changing label).

A generative model could also help produce data for the classifier.

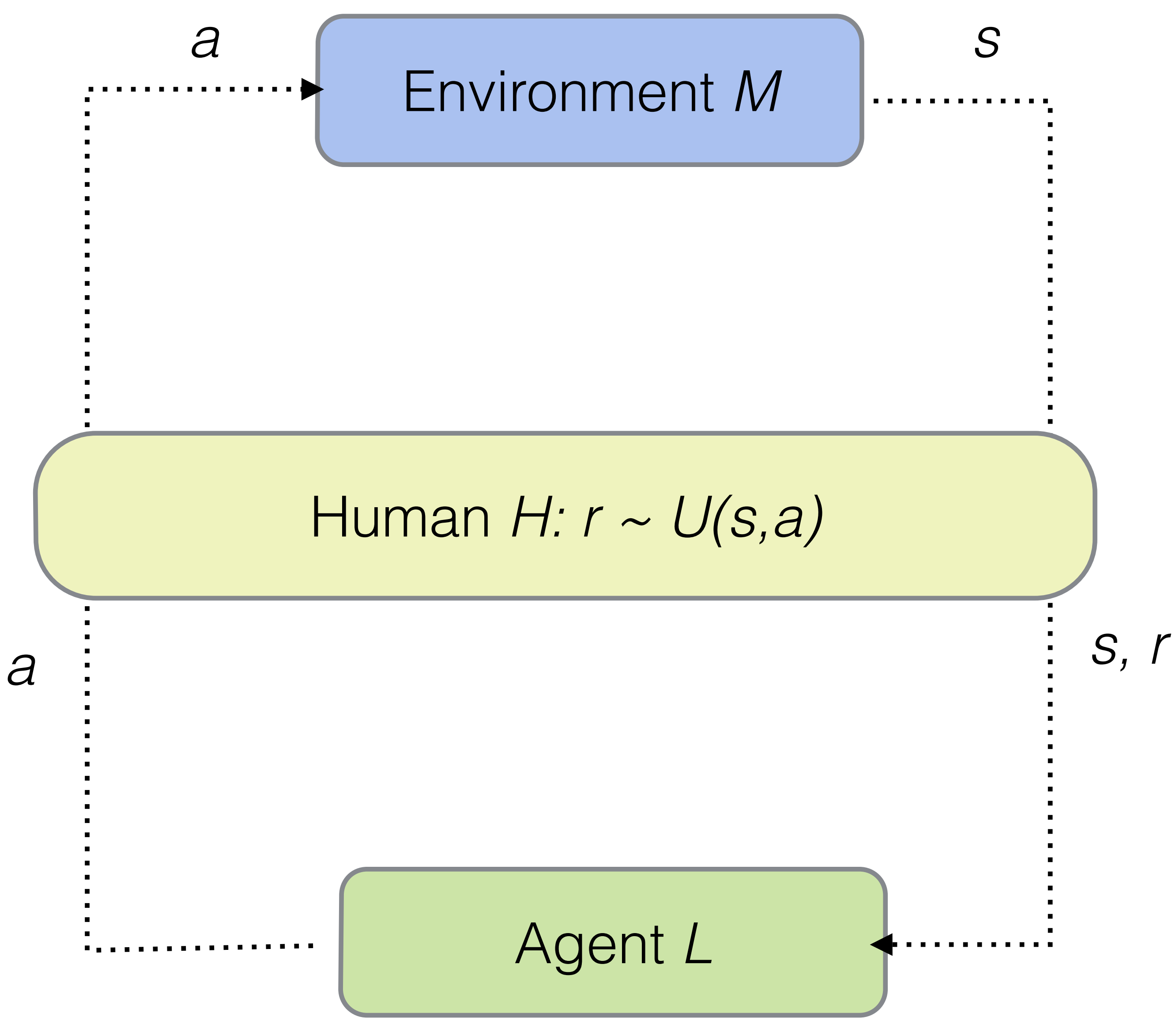
Thanks!

Prevent Catastrophes in Continuous MDPs

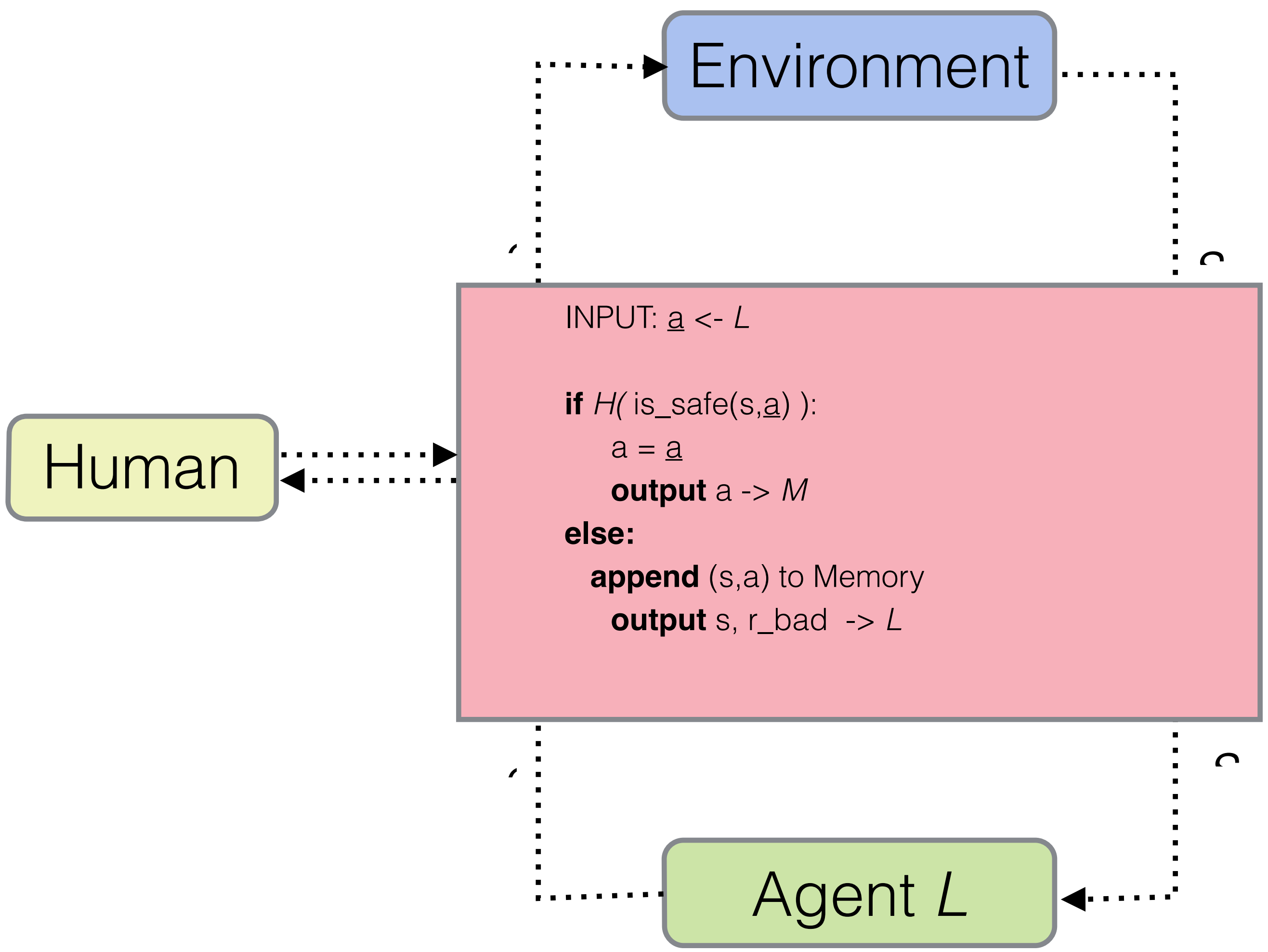
Problem 3: An advanced agent would realize the human is involved

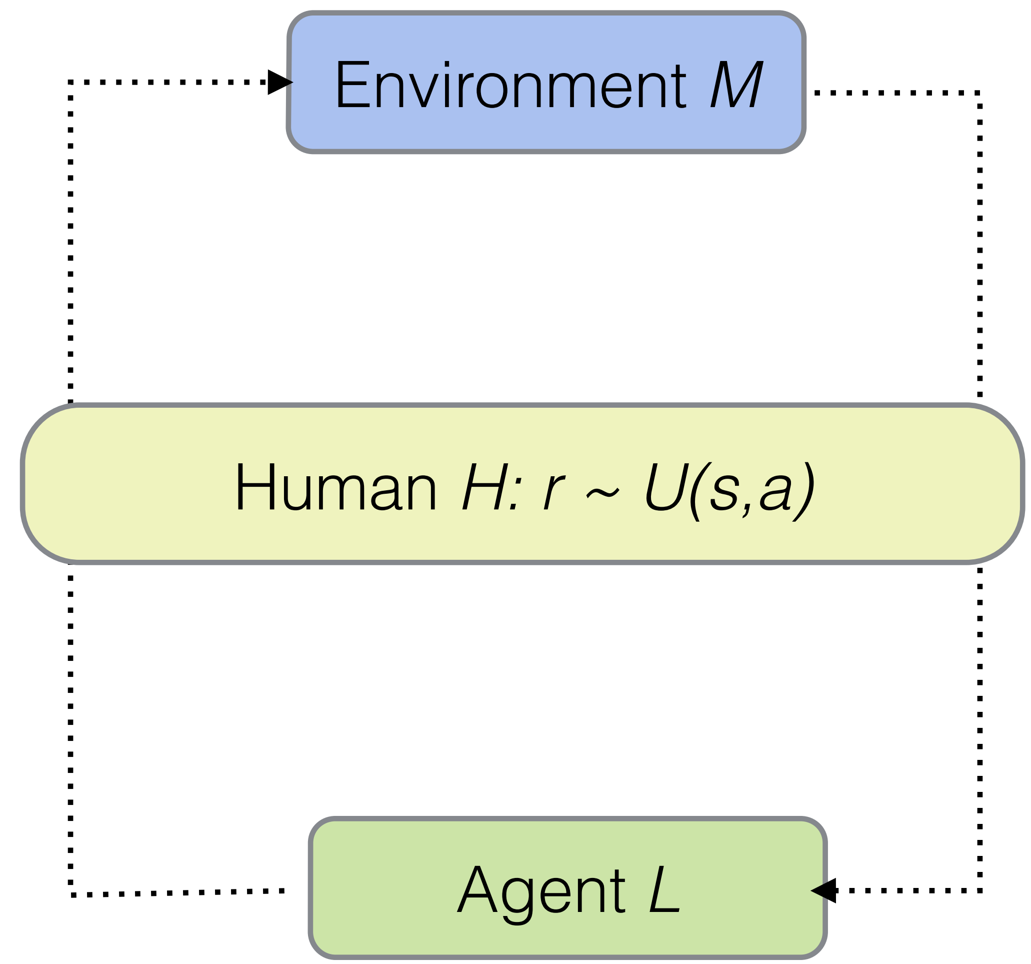
- Current RL agents will not model the human's role.
- Advanced agents might know when the human "retires".
- However: for any agent, imitator blocks catastrophic actions. (No reason to behave differently once human retires.)
- Adversarial attacks: Agent generates dataset for classifier. Response: (a) if too slow to learn, human will shut down, (b) human can run dumber agents to explore space and compare distribution.
- [NB: protocol does involve lots of interaction with agent]

Relation to Christiano's Bootstrapping: Online rewards (TAMER, ARL)



Christiano Bootstrapping vs. Catastrophe Prevention





Relation to Christiano's Bootstrapping

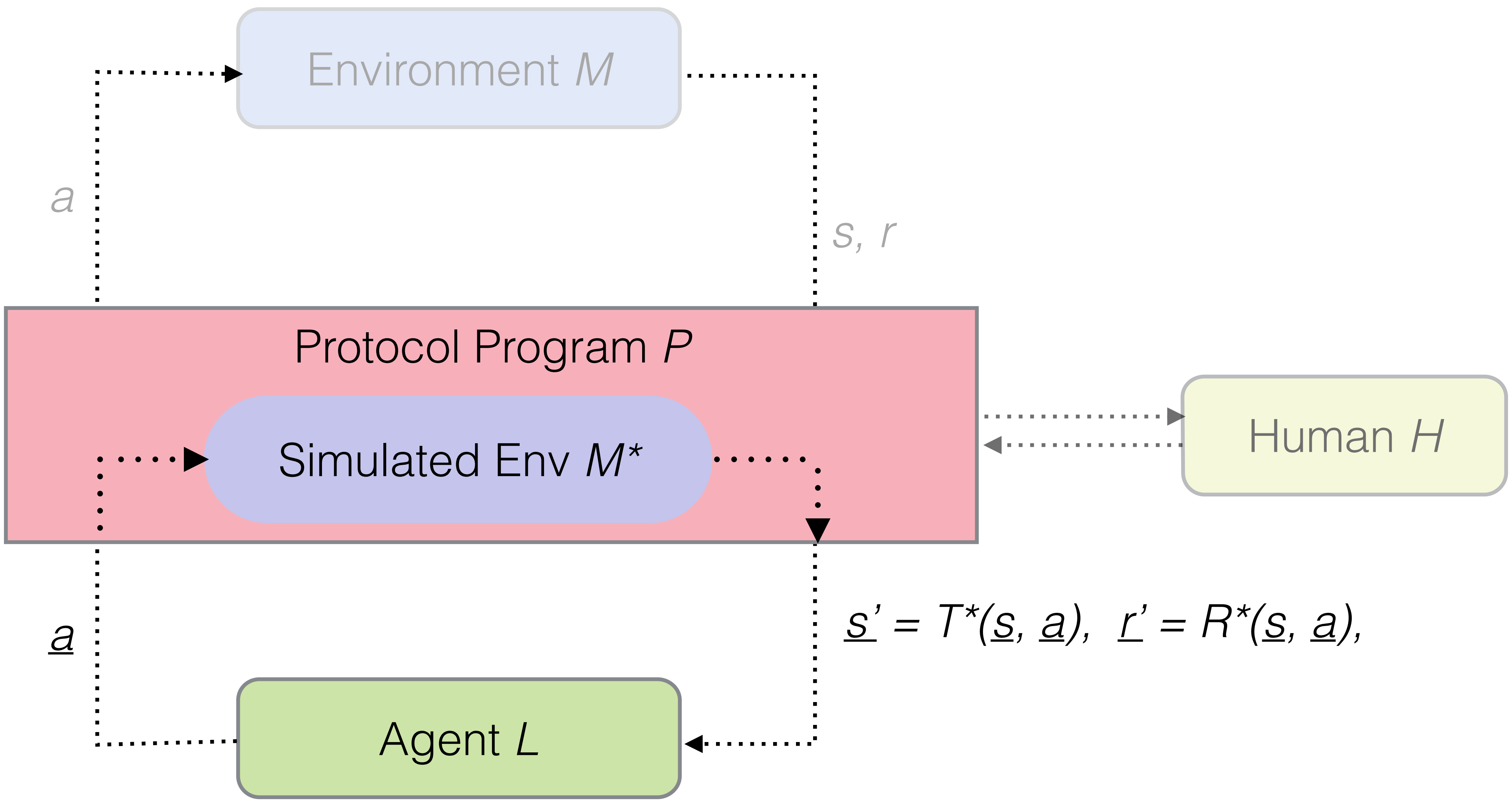
Christiano: Human gives reward for every (tiny) action. Infeasible.

- One approach: agent uses semi-supervised or active learning
- Related approach: use simpler, more trusted learner to provide reward signals (call human when uncertain).

Our protocol: Every action must be checked for catastrophes. Infeasible.

- Choose learner for task of classifying catastrophic actions (should be trusted, robustly generalize, transparent, etc.)

Training in Simulation



Training in Simulation

Alternative approach to training against catastrophes is using simulation (providing lots of dangerous states — maybe generated by other agents). Not easy to work (without deluding agent):

- Simulation not good enough (esp. on weird events)

- Catastrophes too rare for agent to learn them well from natural distribution.

- One option: break black box (Osborne and Whiteson, Precup)

- Another option: bite bullet and train agent on skewed distribution (yields a timid paranoiac)

- Train a classifier on dataset skewed to catastrophes (generated by agent that seeks out catastrophes in the simulation). Use this classifier to block catastrophes in the real world (using our other protocol).