A Java Reinforcement Learning Module for the Recursive Porous Agent Simulation Toolkit

Facilitating study and experimentation with reinforcement learning in multi-agent, social science simulations

Presented by Charles Gieseler

Overview

- Agent based simulation in the Social Sciences
- The Recursive Porous Agent Toolkit (Repast)
- What is the JReLM?
- General Architecture
- Pre-implemented Structures
  - Supporting Classes
  - Graphical User Interface
- Roth-Erev Learning
- Implementation of Roth-Erev algorithms
- The Raita Economy: An Illustrative Application
- Testing and Validation
- Ongoing and future work

Agent-based Simulation in the Social Sciences*

- Social systems:
  - Patterns of the whole emerge from interaction of the many
- Agent-based Simulation
  - Computational Agents: Autonomy, self-directed action
  - Useful metaphor for studying social systems
- Emergent patterns in simulation can give insight into real world systems

* Beginner's Guide http://www.econ.iastate.edu/tesfatsi/abmread.htm

Adaptive behavior in Social Science Simulation

- Agent-based simulation a bottom-up approach
- Behavior of individuals affect macro-scale patterns
- Adaptive behavior more appropriate for metaphors of social systems
- AI, Machine Learning, Cognitive-based methods

Agent-Based Computational Economics*

- Study of economic systems using simulation models of interacting agents
- Approach gaining importance in the economics community
- North-Holland/Elsevier Handbook of Computational Economics Series: Volume 2 Agent-Based Computational Economics
  - Edited by L. Tesfatsion and K. Judd

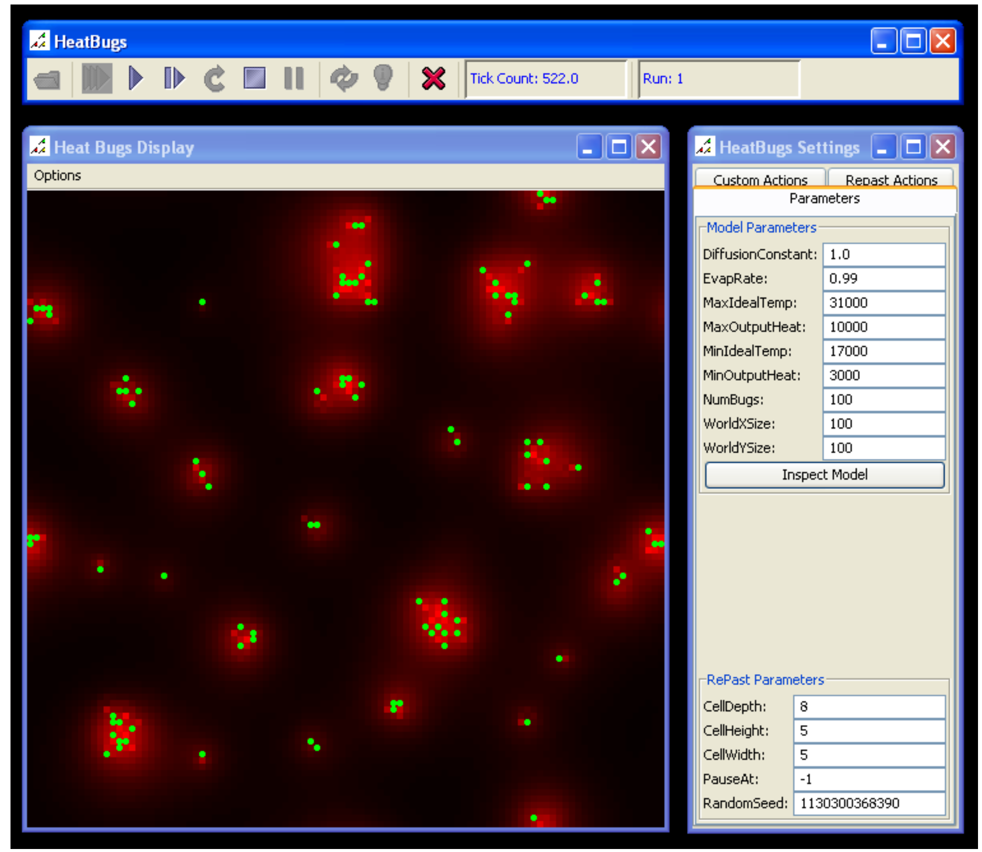
* Collection of ACE resources http://www.econ.iastate.edu/tesfatsi/ace.htm

Simulation use and programming

- Client
  - Programs in the language of the toolkit
  - Designs and implements the program components
  - Builds experimental workbench
- User (or End User)
  - Runs the simulation, usually through the Graphical User Interface (GUI)
  - Designs the experimental setup
  - Performs experimentation and analysis of results

The Recursive Porous Agent Toolkit (Repast)

- “a specification for agent-based modeling services or functions“ *
- Motivated by Swarm
- Sallach (U of Chicago), Collier, Howe, and North (Argonne National Lab)
- Repast Organization for Architecture and Development (ROAD)
- Current version 3.1
- Three flavors: RepastJ, Repast.NET, RepastPy



* Repast homepage http://repast.sourceforge.net/

Popularity of Repast

- *Evaluation of free java-libraries for social-scientific agent based simulation*, Tobias and Hoffman, 2004 *
  - Comparison of freely available agent-based simulation platforms
  - Five categories of detailed criteria, geared towards Social Science study
  - Repast the best
- Currently the primary tool used by the ACE group in the Department of Economics

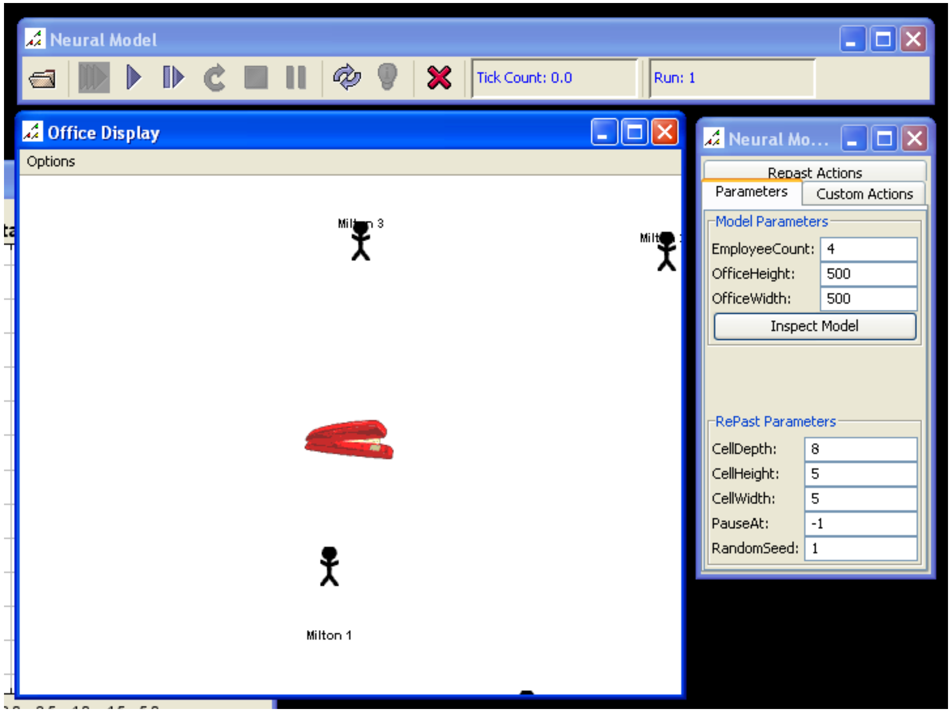
* Available at http://ideas.repec.org/a/jas/jasssj/2003-45-2.html

Structure of a Repast simulation

- Model: defines and runs the simulation
- Space: The environment or world which agents inhabit
- Dynamic GUI:
  - Displays and charts
  - Custom parameter settings
- Agents: Open, client defined
  - Allows for flexibility
  - Can be burdensome if more complex behavior is required

Adaptive behavior in Repast

- Genetic Algorithm demo model
- OpenForecast demo model
- Java Object Oriented Neural Engine (JOONE)
  - Wrapper and demo
- Must custom implement other methods
  - Hard for the novice
  - Time consuming for the expert



What is JReLM?

Java Reinforcement Learning Module

- Platform for implementing and using reinforcement learning in Repast
- Ease the burden of design and implementation for the client
- Allow the user to manage learning settings through the Repast GUI
- Designed specifically for use in RepastJ
- Open Source, Release with Repast

What JReLM offers

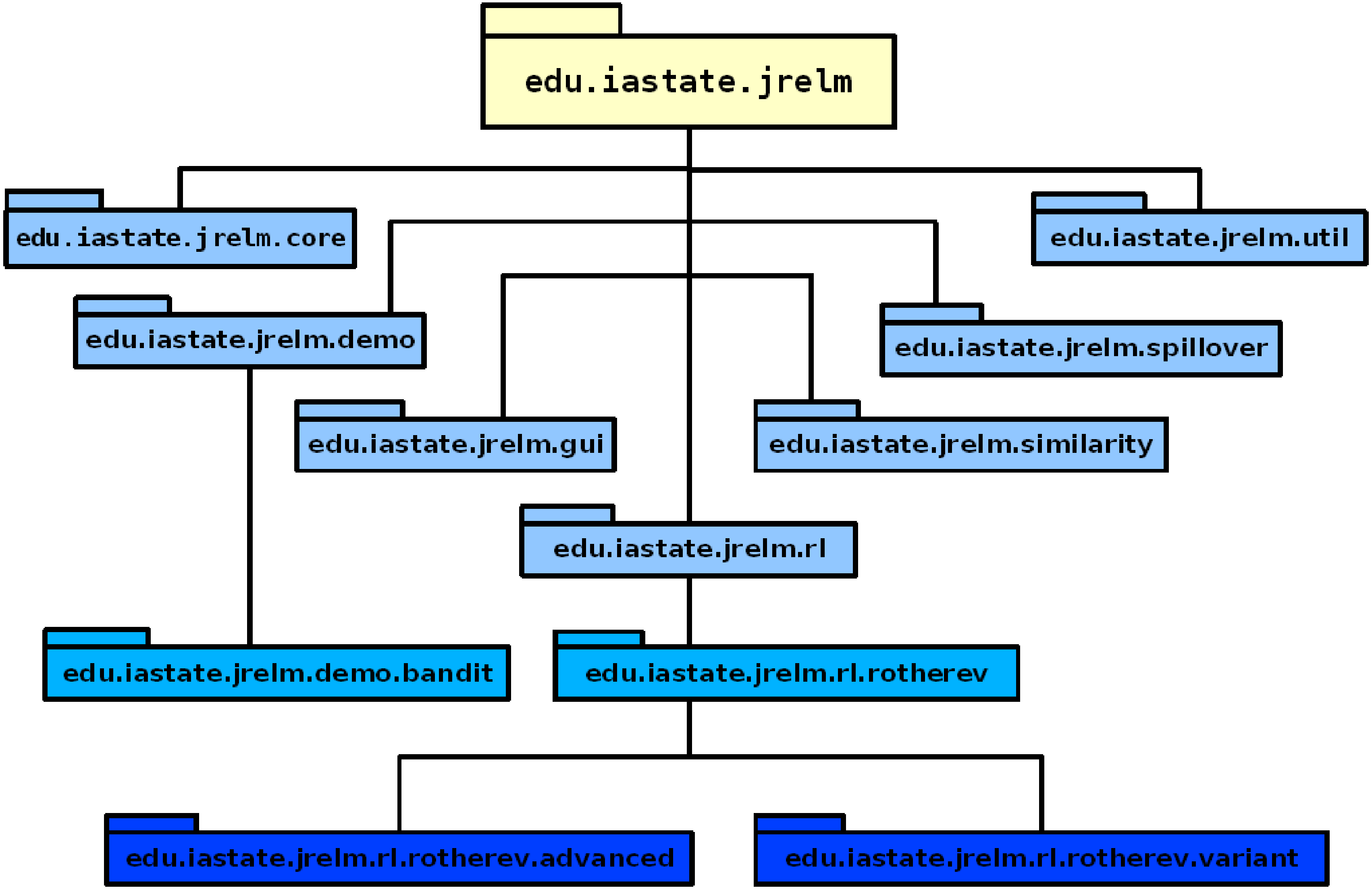
- Platform for algorithm implementation
  - Framework of structures common to many types of reinforcement learning
- Includes algorithms currently in use in social science applications
- Graphical User Interface
- Integrated into Repast

General Architecture

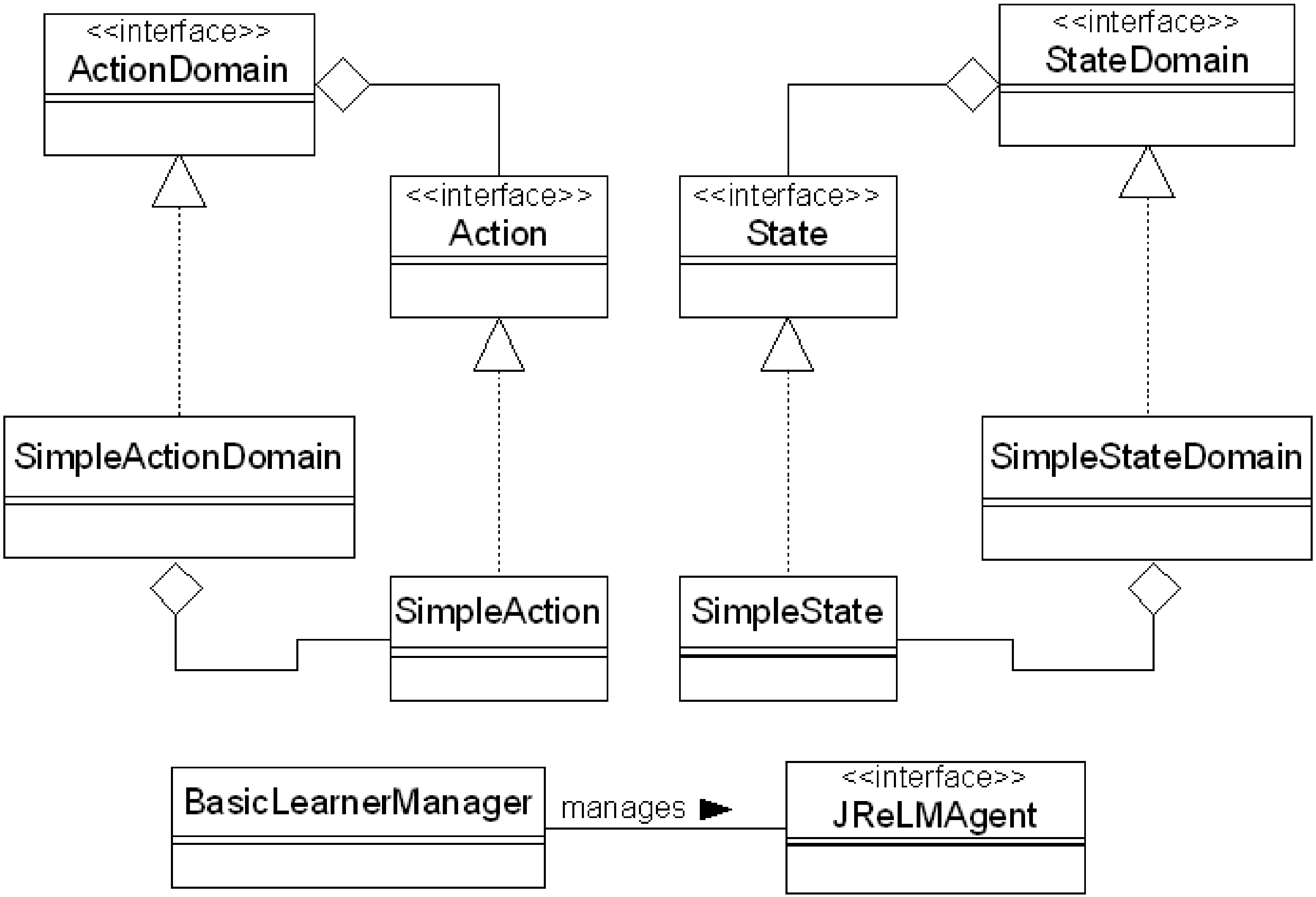
- Motivated by Sutton's and Barto's description of Reinforcement Learning*
- Component-based
  - Plug into client defined agents
- Flexible
  - Arbitrary simulation contexts
  - Arbitrary agents
- Extensible
  - Object-oriented design
  - Documentation (Javadocs)

* Reinforcement Learning: An Introduction
http://www.cs.ualberta.ca/~sutton/book/the-book.html

Package Hierarchy



edu.iastate.jrelm.core



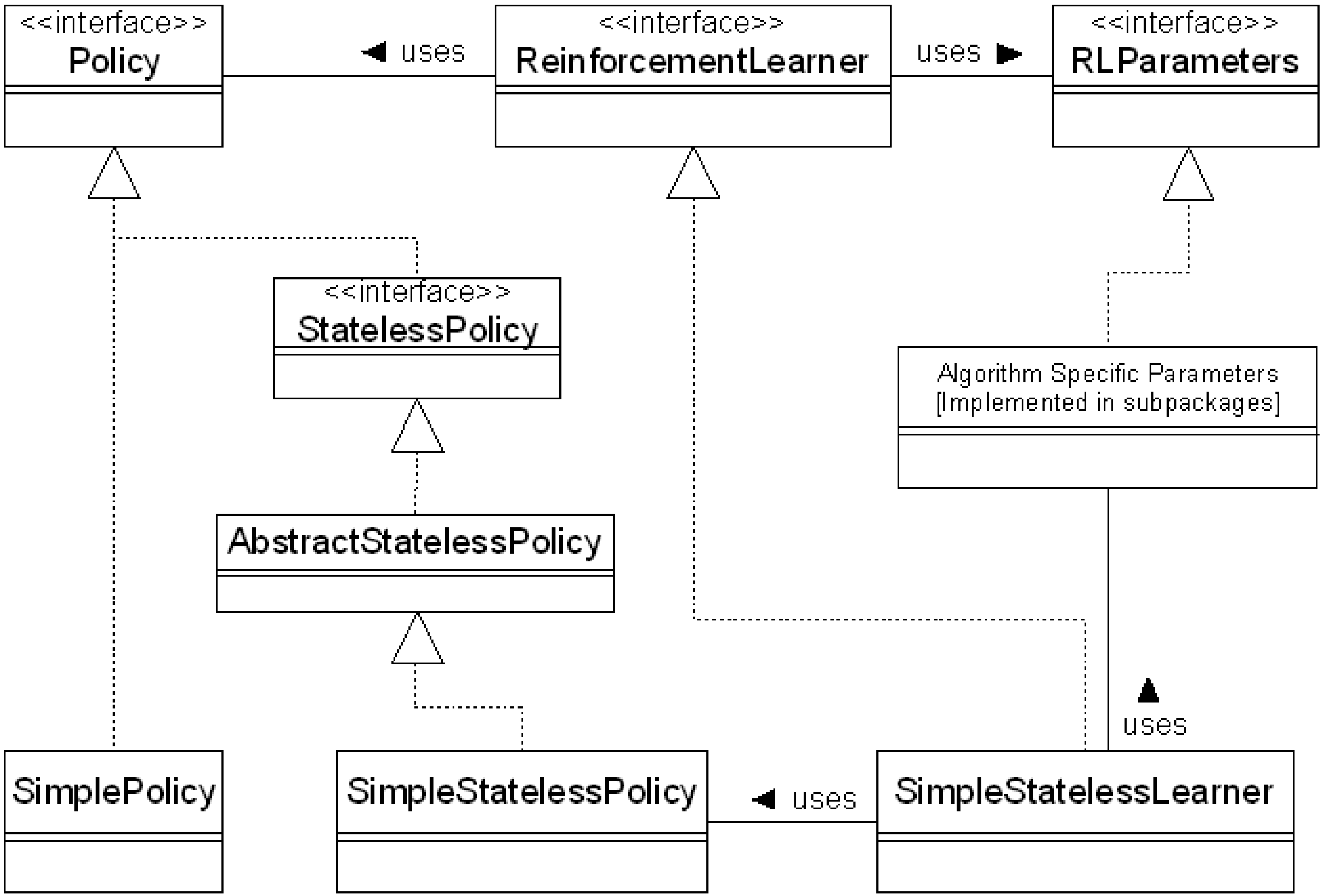
Action, ActionDomain, State, StateDomain

- Interfaces, not classes
- Important! Separates representation from implementation
- Flexibility: Applied to wide variety of simulation contexts
- Limitations:
  - Burden of domain implementation on the client. Hard to avoid without over-customization.
  - Discrete, finite domains only

SimpleAction, SimpleState, SimpleActionDomain, SimpleStateDomain

- Basic implementation of the domain interfaces
- Wrappers around other objects, Collections
- Bridge between existing action choice/world states and JReLM
- Help ease burden in simpler simulations

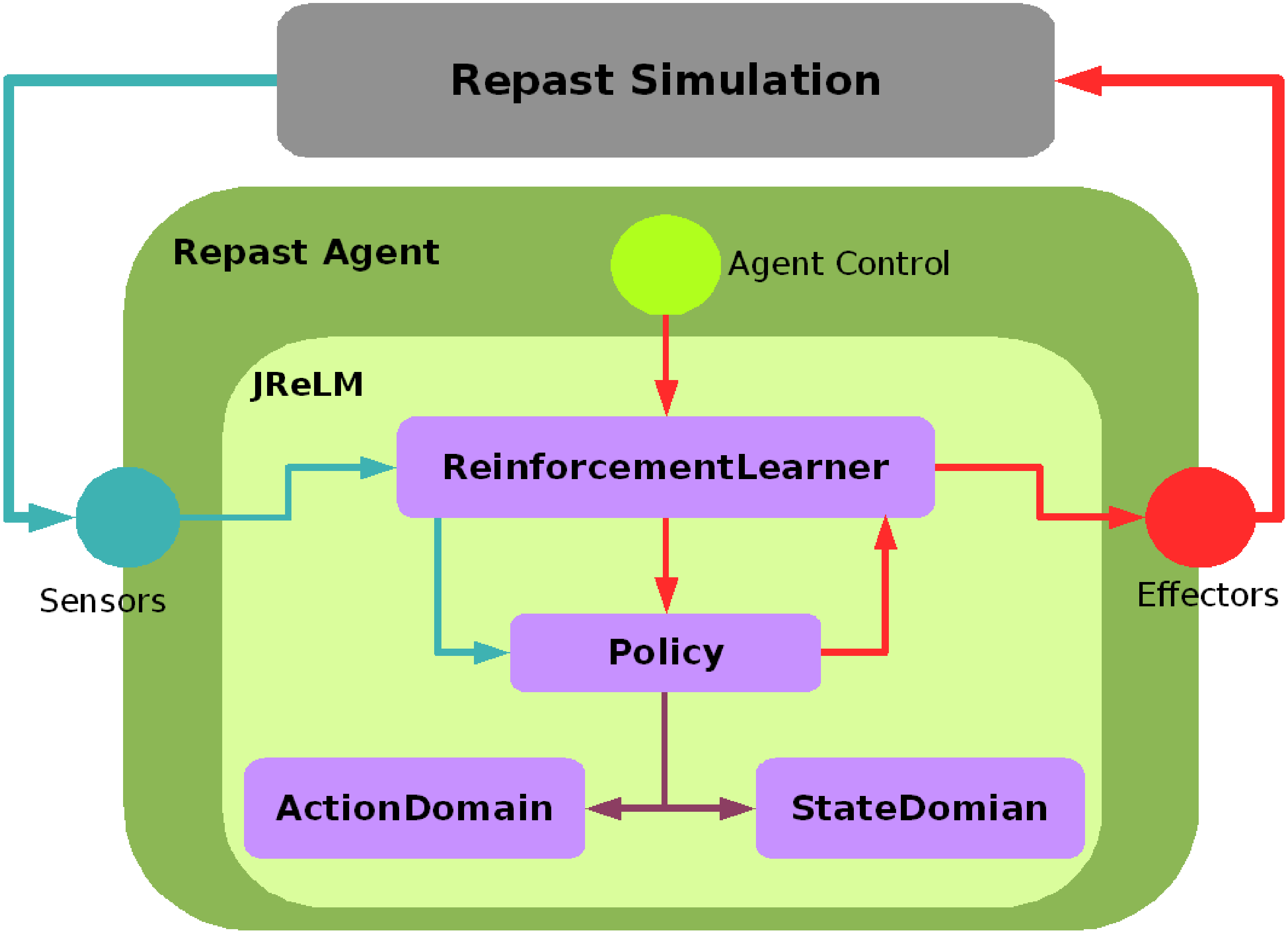
edu.iastate.jrelm.rl



Base learning components

- ReinforcementLearner
  - Interface for all RL implementations (learners)
  - Work with an ActionDomain, a Policy, and a sometimes a StateDomain
- Policy
  - Mapping from State-Action pairs to probability values
    - Distributions for action choice likelihood
  - Generate new action choices
  - Compatible with any ActionDomain and StateDomain
- StatelessPolicy
- RLParameters
  - Encapsulate parameters for an algorithm
  - Used in building custom GUI

Interaction of JReLM components

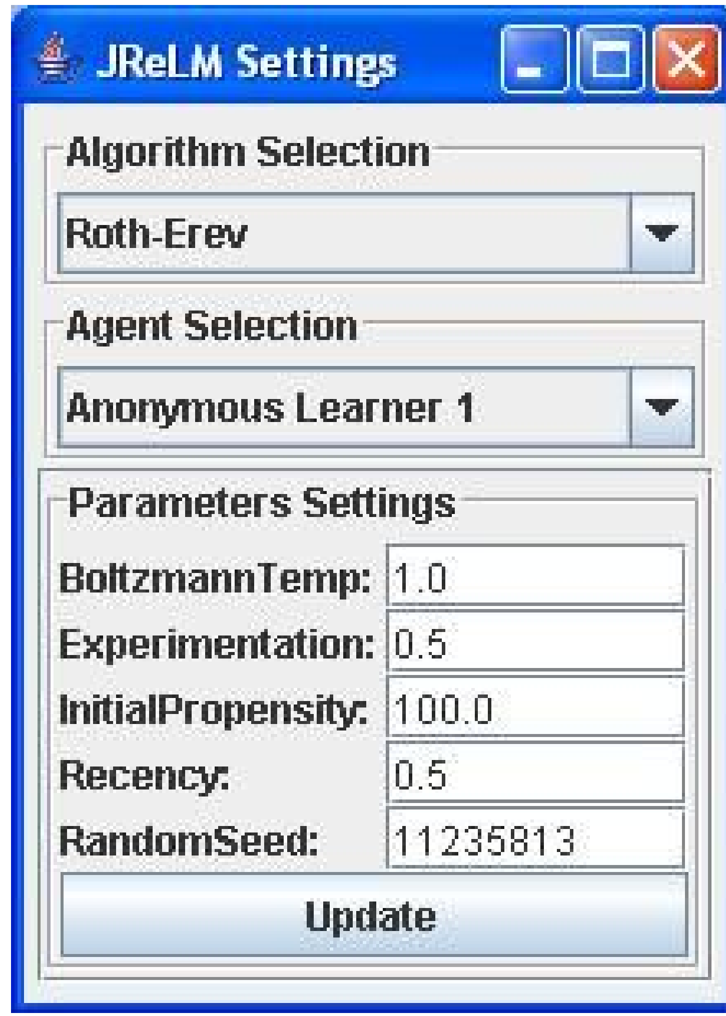


Simple*

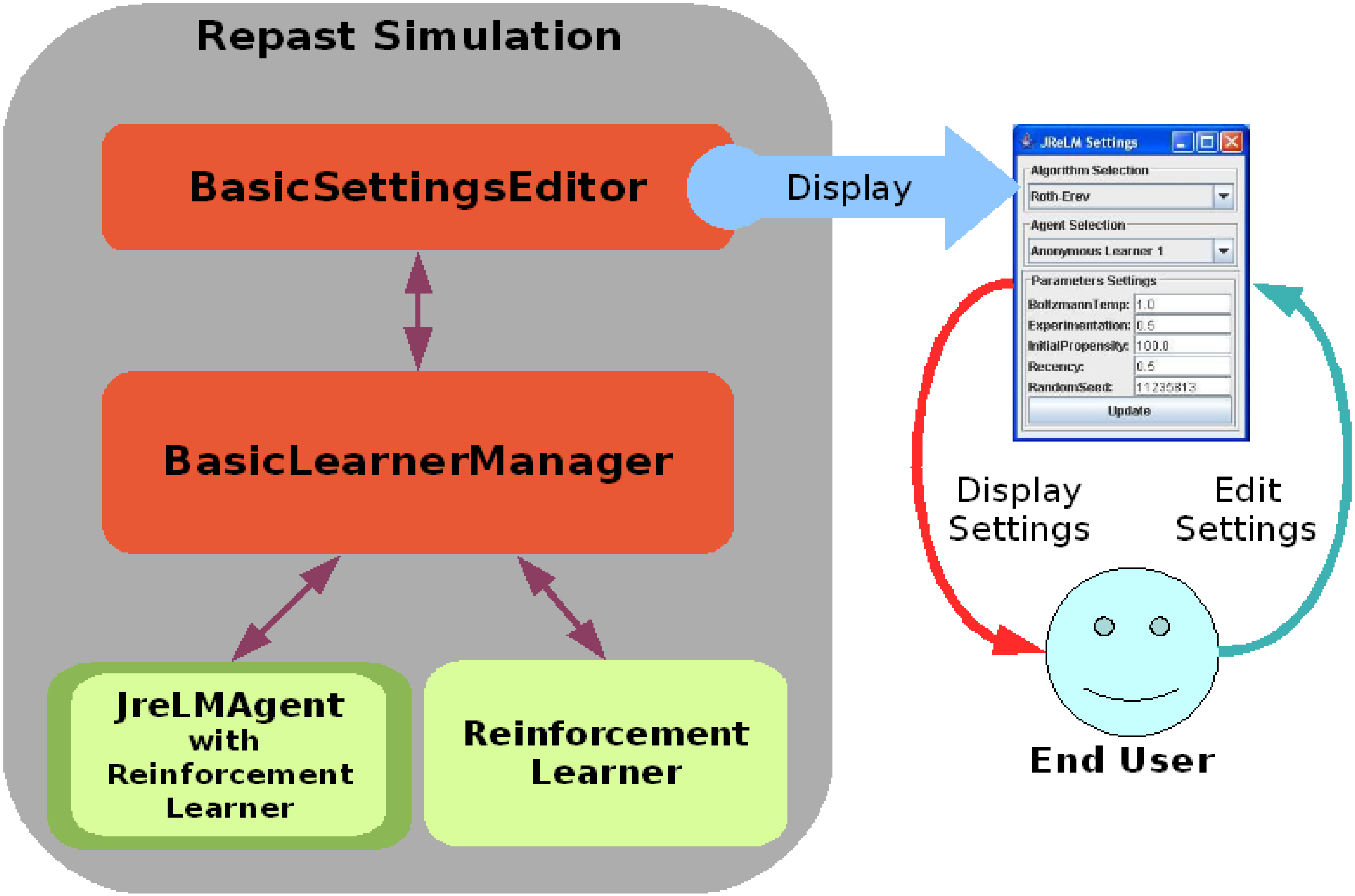
- SimplePolicy and SimpleStatelessPolicy
  - Basic implementations of Policy interfaces
- SimpleLearner and SimpleStatelessLearner
  - Tie together all pre-implemented learners
  - Algorithm to use determined by the type of RLParameters given
  - May be given Collections as domains
  - Simplified use, but limited

Graphical User Interface

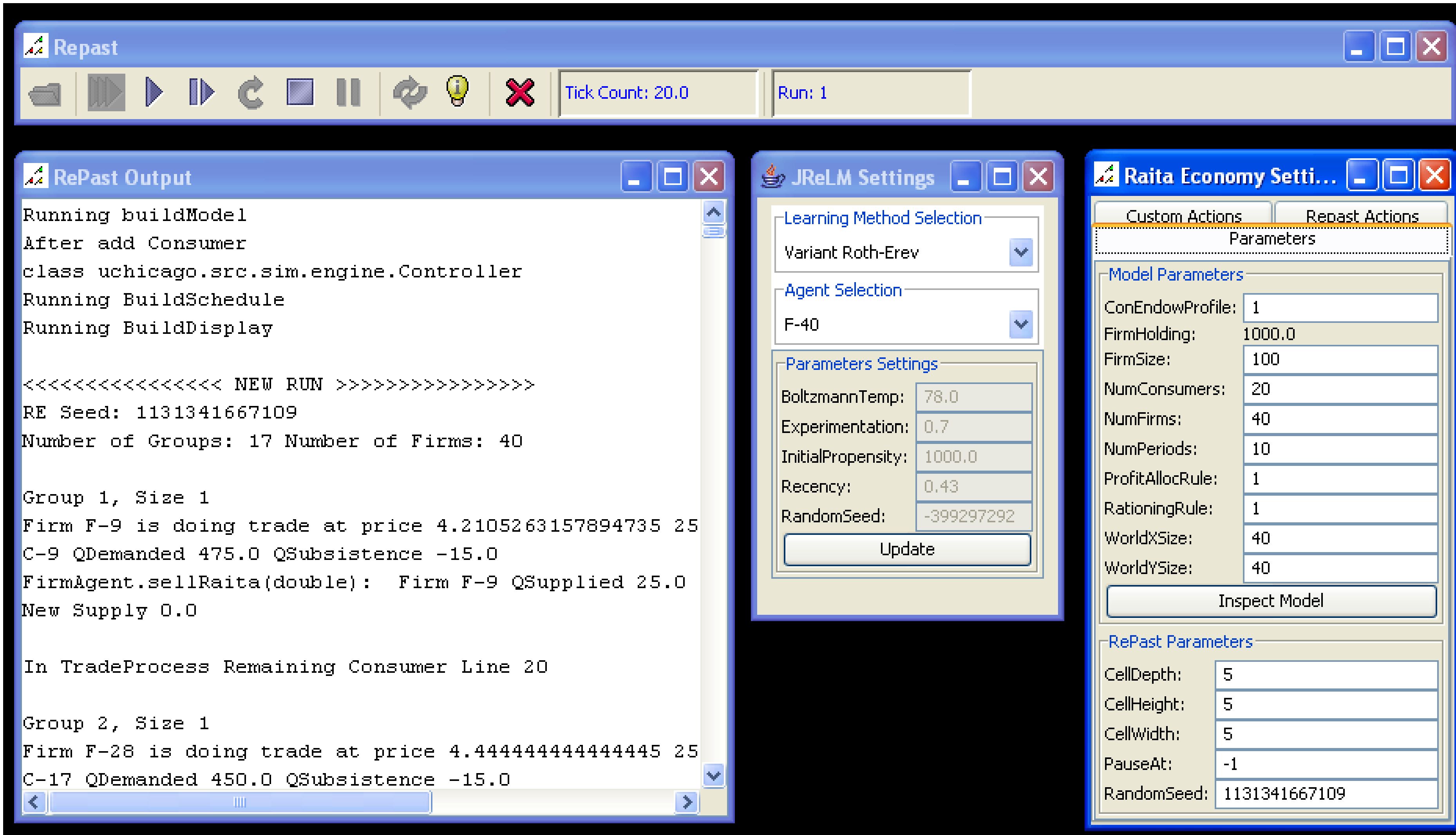
- Goals:
  - Allow the user to modify learning settings without programming
  - Track and manage all learning methods used in a model
- Challenge:
  - How to do this in arbitrary agents and models?



Graphical User Interface cont.



Graphical User Interface cont.



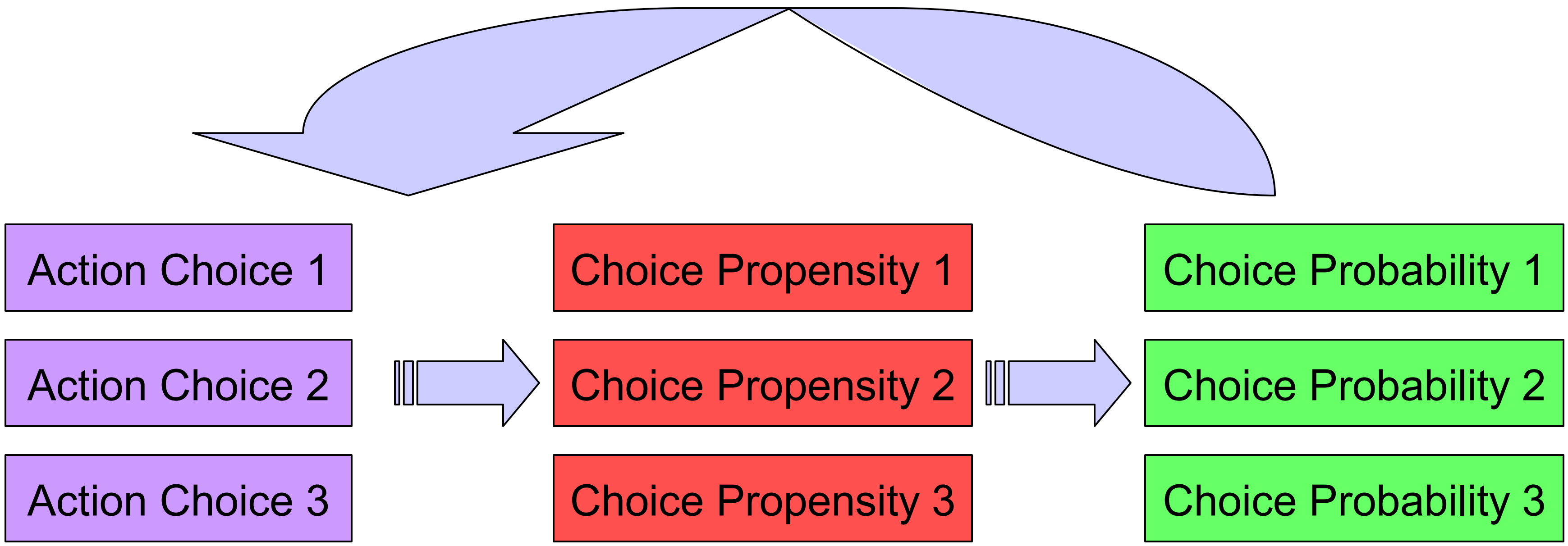
Pre-Implemented Reinforcement Learning Algorithms

- Assist novice programmer
- Convenience for experienced programmer
- Compatible with any action and state space that can be represented by an ActionDomain or StateDomain class

Roth-Erev Reinforcement Learning

- Originally developed by Alvin E. Roth and Ido Erev
  - Attempt to model how humans play in repeated games against multiple strategic players
- Later modified by Nicolaisen, Petrov and Tesfatsion
  - Problem encountered with zero-valued rewards

Roth-Erev Algorithm Structure



- Maintains action choice propensities which are translated into action choice probabilities

Algorithm Outline

1. Initialize action propensities to an initial propensity value. Initialize the action choice probabilities to a uniform distribution.
2. Generate choice probabilities for all actions using current propensities.
3. Choose an action according to the current choice probability distribution.
4. Update propensities for all actions using the reward for the last chosen action.
5. Repeat from step 2.

The update and experience functions

Parameters

- q_0 Initial Propensity
- ϵ Experimentation
- ϕ Recency:

Variables

- j Current action choice
- q_j Propensity for j
- k Last action chosen
- r_k Reward for k
- t Current timestep
- N Number of actions

$$q_j(t+1) = [1-\phi]q_j(t) + E_j(\epsilon, j, k, t)$$

$$E_j(\epsilon, j, k, t) = \begin{cases} r_k(t)[1-\epsilon] & \text{if } j = k \\ r_k(t)\frac{\epsilon}{N-1} & \text{if } j \neq k \end{cases}$$

Probability function

- Proportional distribution

$$p_j(t) = \frac{q_i(t)}{\sum_{m=1}^{n} q_m(t)}$$

Variation of Roth-Erev

- Nicolaisen, Petrov and Tesfatsion* modified the experience function in response to a problem with learning in the face of zero-value rewards.

$$E_j(\epsilon, k, t) = \begin{cases} r_k(t)[1-\epsilon] & \text{if } j = k \\ q_j(t)\frac{\epsilon}{N-1} & \text{if } j \neq k \end{cases}$$

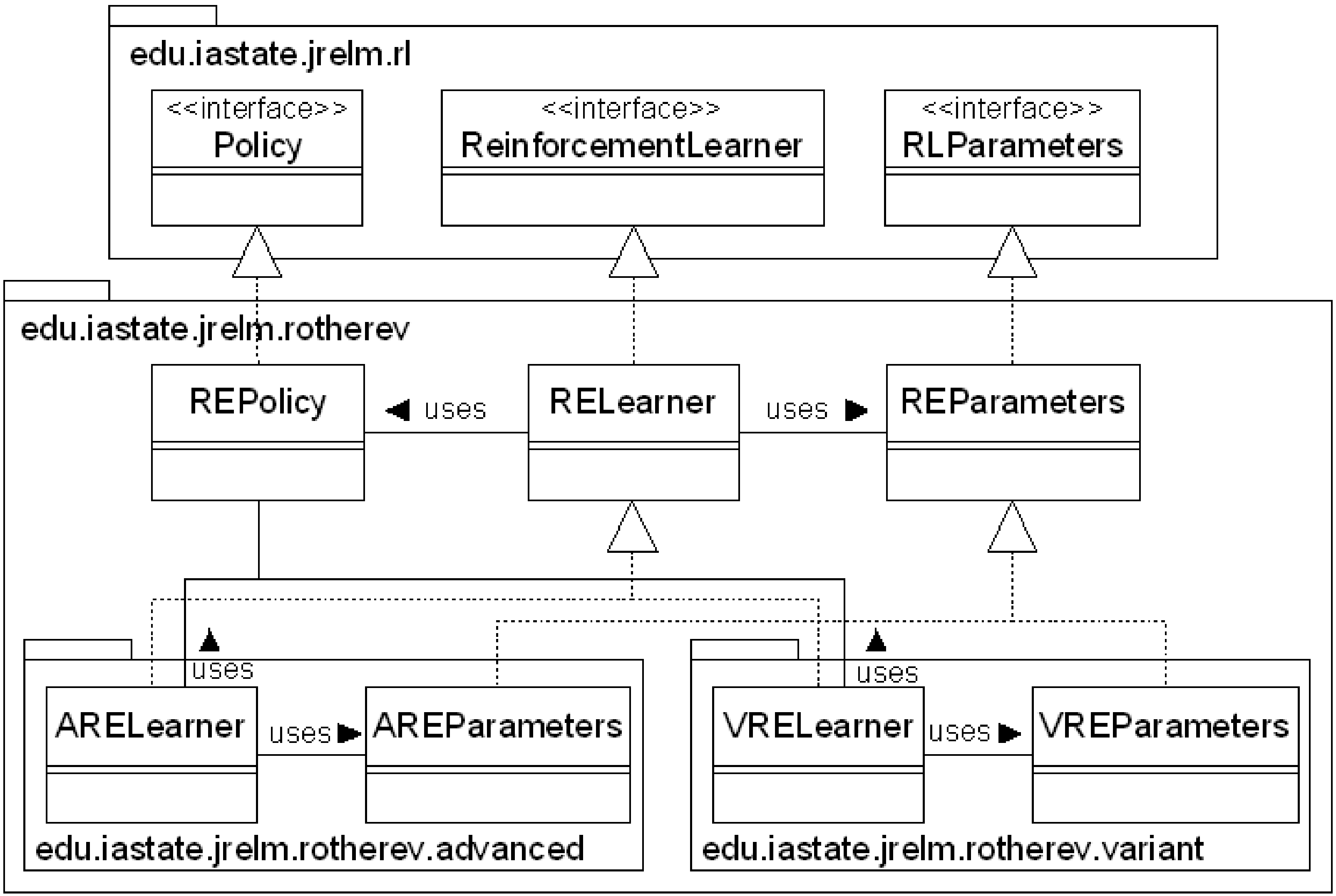
* Nicolaisen, J., Petrov, V., and Tesfatsion, L. *Market Power and Efficiency in a Computational Electricity Market with Discriminatory Double-auction Pricing*. IEEE Transactions on Evolutionary Computing 5, 5 (October 2001), 504–523.

Gibbs-Boltzmann distribution

- Handle negative propensities
- T temperature parameter
  - Static, no temperature schedule

$$p_j(t) = \frac{e^{q_j(t)/T}}{\sum_{i=1}^{n} e^{q_i(t)/T}}$$

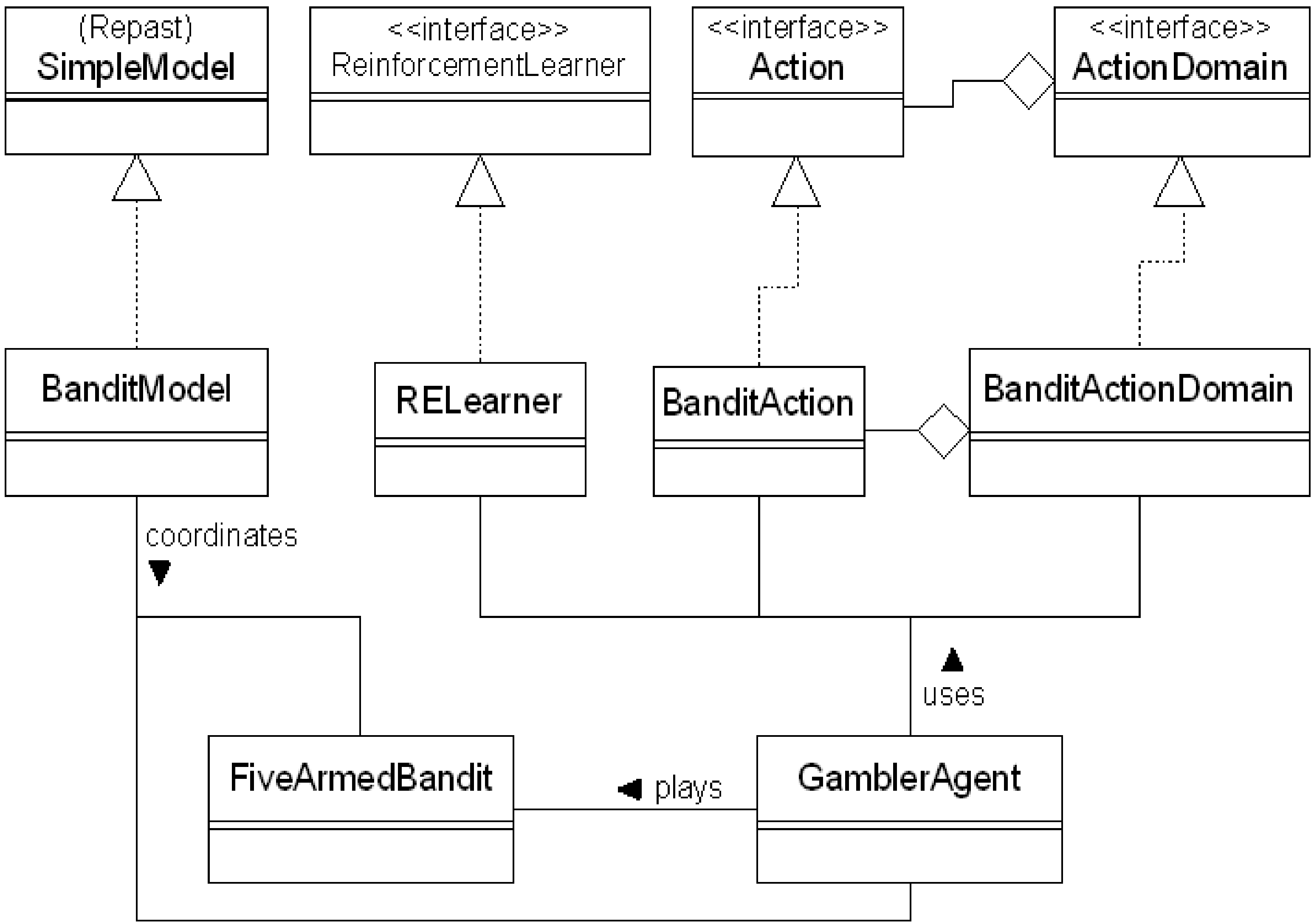
edu.iastate.jrelm.rotherev



Implementation of the Roth-Erev family

- RELearner (Roth-Erev Learner)
  - Base implementation of the original algorithm
  - REParameters
  - REPolicy
- VRELearner (Variant Roth-Erev Learner)
- ARE (Advanced Roth-Erev Learner)
  - Core structure with advanced, customizable features

edu.iastate.jrelm.demo.bandit



The Raita Economy: An illustrative application

- Repast simulation developed by Somani and Tesfatsion
- Examine market concentration in relation to market power in a dynamic, single product (raita) economy

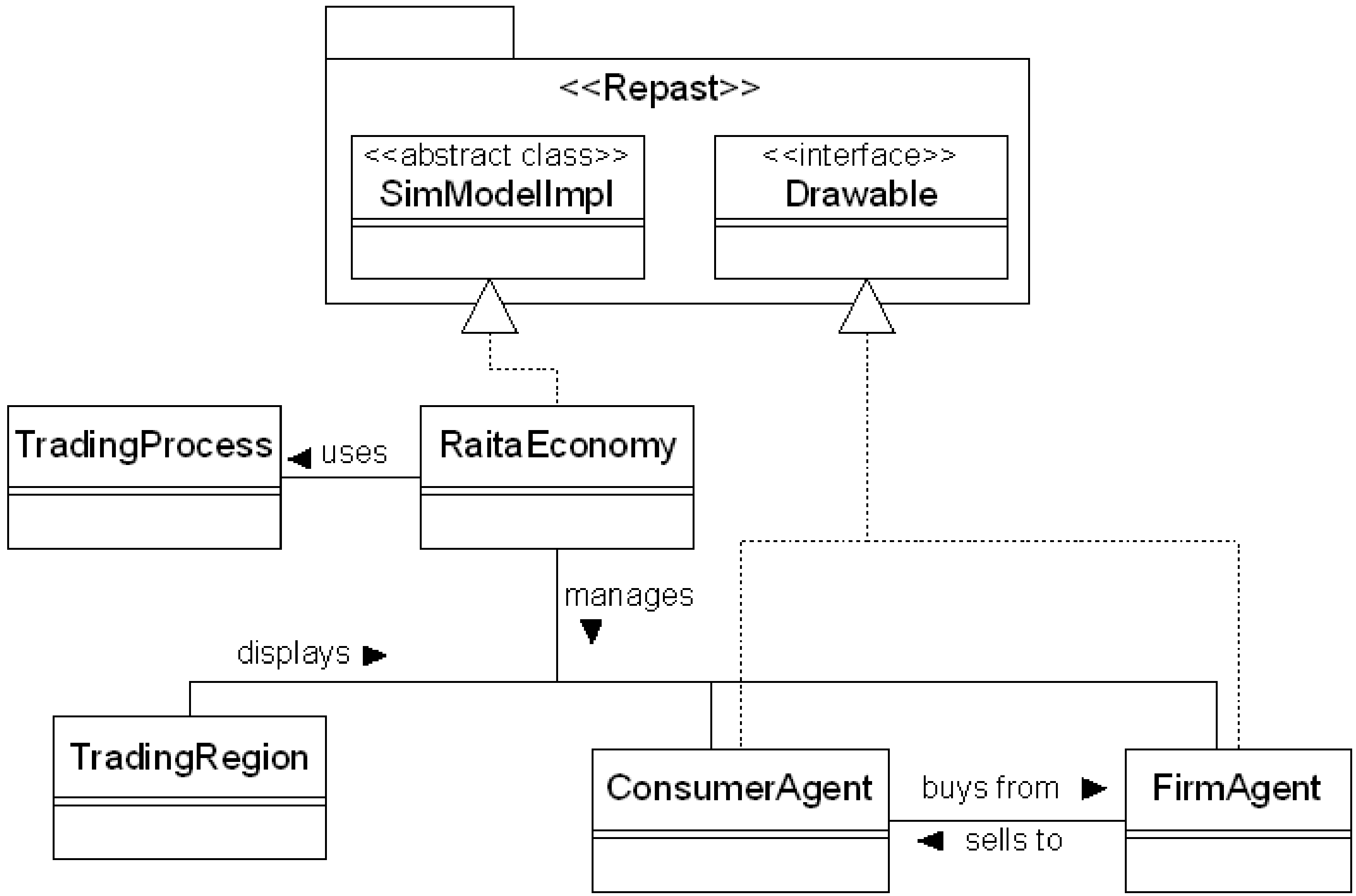
Market Concentration and Market Power

- *Market concentration*: the degree to which the majority of market activity is performed by a minority of the participants.
- *Market power:* the degree to which a participant may profitably influence prices away from competitive levels.

Market Concentration and Market Power cont.

- Measures of market concentration are often used as indicators of market power.
- This model examines how well three common measures predict the rise of market power in a simple dynamic production economy

RaitaEconomy class structure



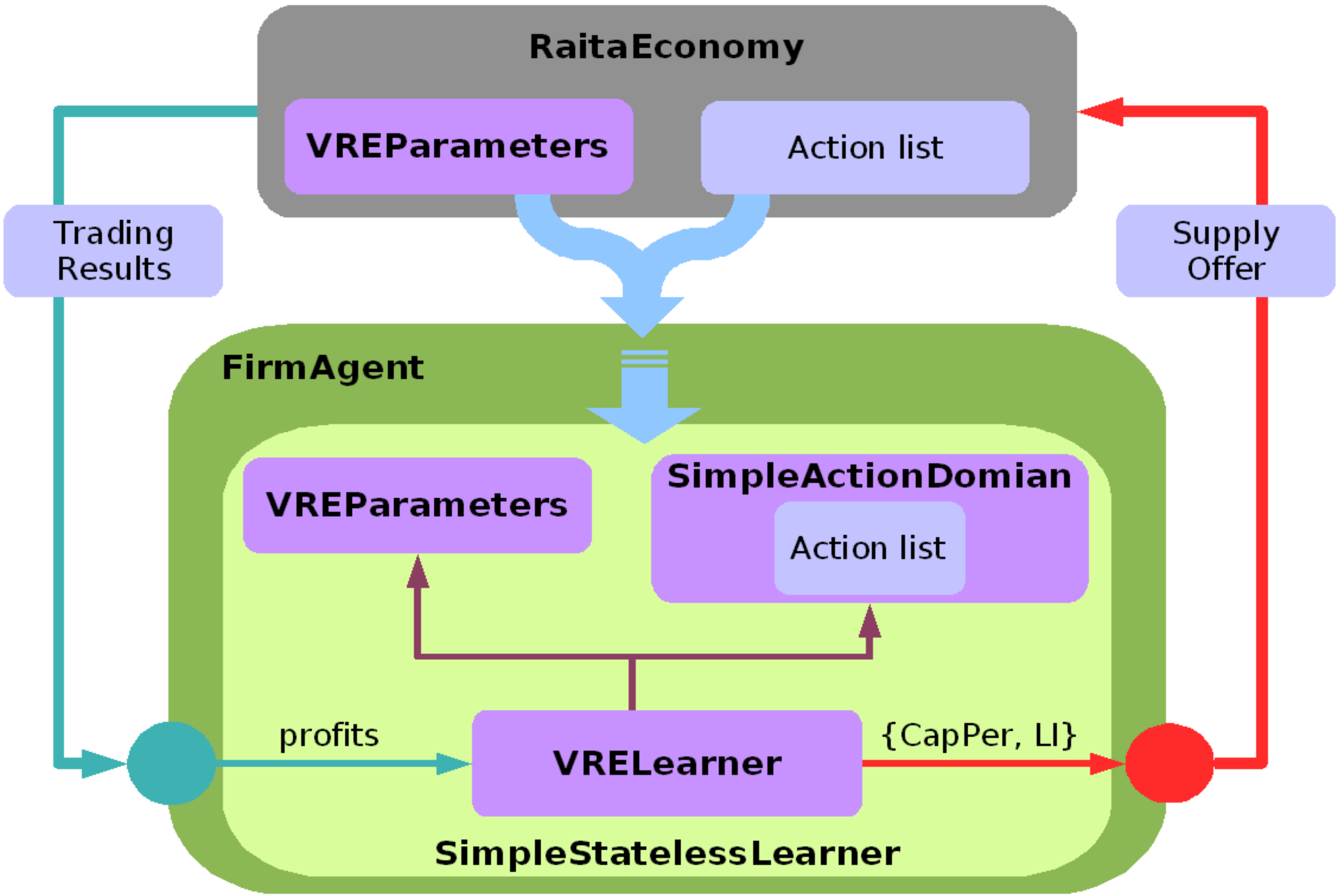
ConsumerAgent

- Simple, reactive agent
- Gains utility by consuming raita
- Seeks raita at the lowest available price
- Dies if subsistence needs are not met

FirmAgent

- Strategic, learning agent
- Gain profit by producing and selling raita
- May adjust production and price level every trading period
  - supply offers: Production quantity and unit price
- Can also invest profits in expanding production capacity
- Exits market if it goes bankrupt

JReLM in the RaitaEconomy



JReLM in the Raita Economy cont.

- Raita Economy still under construction
- First research model to use JReLM
- Balance of complexity
  - Market content more complex than the, multi-agent
  - Still simpler than other context (e.g. The AMES project)
- Valuable experience
  - What needs arise in an actual research context
  - How usable is JReLM?

Testing and Validation

- Unit testing of JReLM using JUnit*
  - Suite of tests have been built along the way
  - Still expanding
- Validation of Roth-Erev family
  - Are they behaving as expected?
  - Bandit Demo: Simple, single agent context
  - Raita Economy: More complex, multi-agent context

* JUnit is a Java unit testing package available at *http://www.junit.org/index.htm*

Ongoing and future work

- Expansion of RL methods library
- Investigation into additional methods
  - Appropriate for multi-agent contexts
  - Appropriate for Social Simulation
- Improvement of the GUI
  - Integration into the Repast control panel
  - Improve management of groups of agents

Agent-Based Modeling of Electricity Systems (AMES)

- Federal Energy Regulatory Commission Wholesale Power Market Platform
- AMES: Repast model designed to test the economic reliability of the WPMP
- JReLM: adaptive pricing and quantity offers for generators
- Bulk Energy Transportation Networks*
  - NSF funded
  - Study integrated energy networks

** Lead by primary investigator J. McCalley and co-PIs S. Ryan, S. Sapp, and L. Tesfatsion*

The NISAC Agent-Based Laboratory for Economics

- National Infrastructure and Analysis Center (NISAC)
  - Joint effort between Los Alamos and Sandia National Labs
  - Funded by the Department of Homeland Security
  - Examine critical national infrastructure
- N-ABLE (at Sandia):
  - Agent-based simulation modeling platform
- JReLM architecture and interaction, starting point for expanded adaptive behavior in N-ABLE

JReLM Distribution with Repast

- Repast an Open Source project
- Discussion with Repast developers
  - Inclusion of JReLm into the RepastJ package
- Requires a demonstration
- Goal
  - complete testing and validation of JReLM in time for Repast's next release

Questions