Learning to Interpret Natural Language Instructions

Shawn Squire, Monica Babes-Vroman, Marie desJardins, Ruoyuan Gao, Michael Littman, James MacGlashan, Smaranda Muresan

UMBC, Brown University, Columbia University, Rutgers University

The Problem

1. Supply an agent with an arbitrary linguistic command
2. Agent determines a task to perform
3. Agent plans out a solution and executes task

- Planning and execution is easy
- Learning task semantics and intended task is hard

The Solution

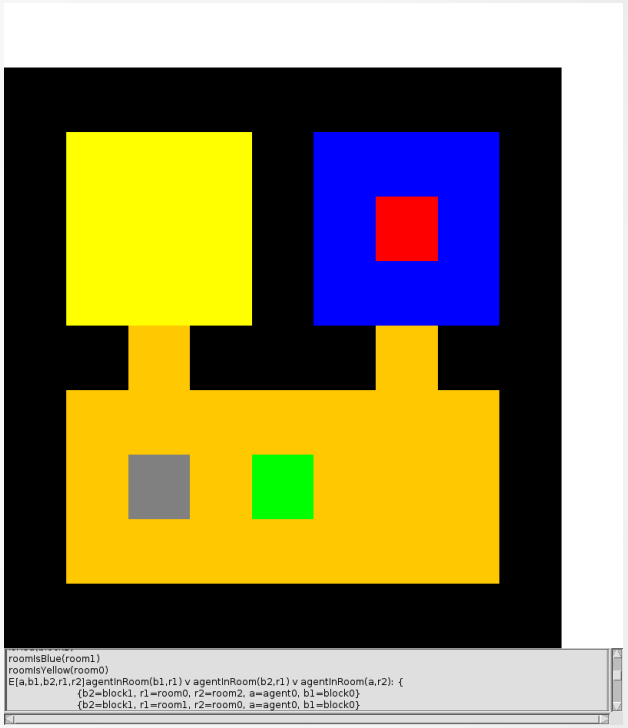
- Use expectation maximization (EM) and a generative model to learn semantics
- Pair command with demonstration of exemplar behavior
  - This is our training data
- Find highest-probability tasks and goals

Representation

- Tasks are represented using Object-Oriented Markov Decision Processes (OO-MDP)
- Defines relationship between objects
- Each state is:
  - Unordered instantiation of objects
  - Propositional functions that operate on objects
  - Goal definitions

Sample Domain (Sokoban)

- isStar(block0)
- isStar(block1)
- agentInRoom(agent0, room2)
- blockInRoom(block0, room2)
- blockInRoom(block1, room1)
- isRed(block1)
- roomIsBlue(room1)
- roomIsYellow(room0)



The System

- Three main components
  - Semantic Parsing (SP)
    - Sentences → Logical Representations
  - Inverse Reinforcement Learning (IRL)
    - Generate intent from observed behavior
  - Task Abstraction (TA)
    - Ground logic from SP into specific objects

Semantic Parsing

- Bag-of-words multinomial mixture model
  - Each propositional function has multinomial word distribution
  - Given task, a word is generated by using a word distribution from task's propositional functions
  - Don't need to learn meaning of words in every task context

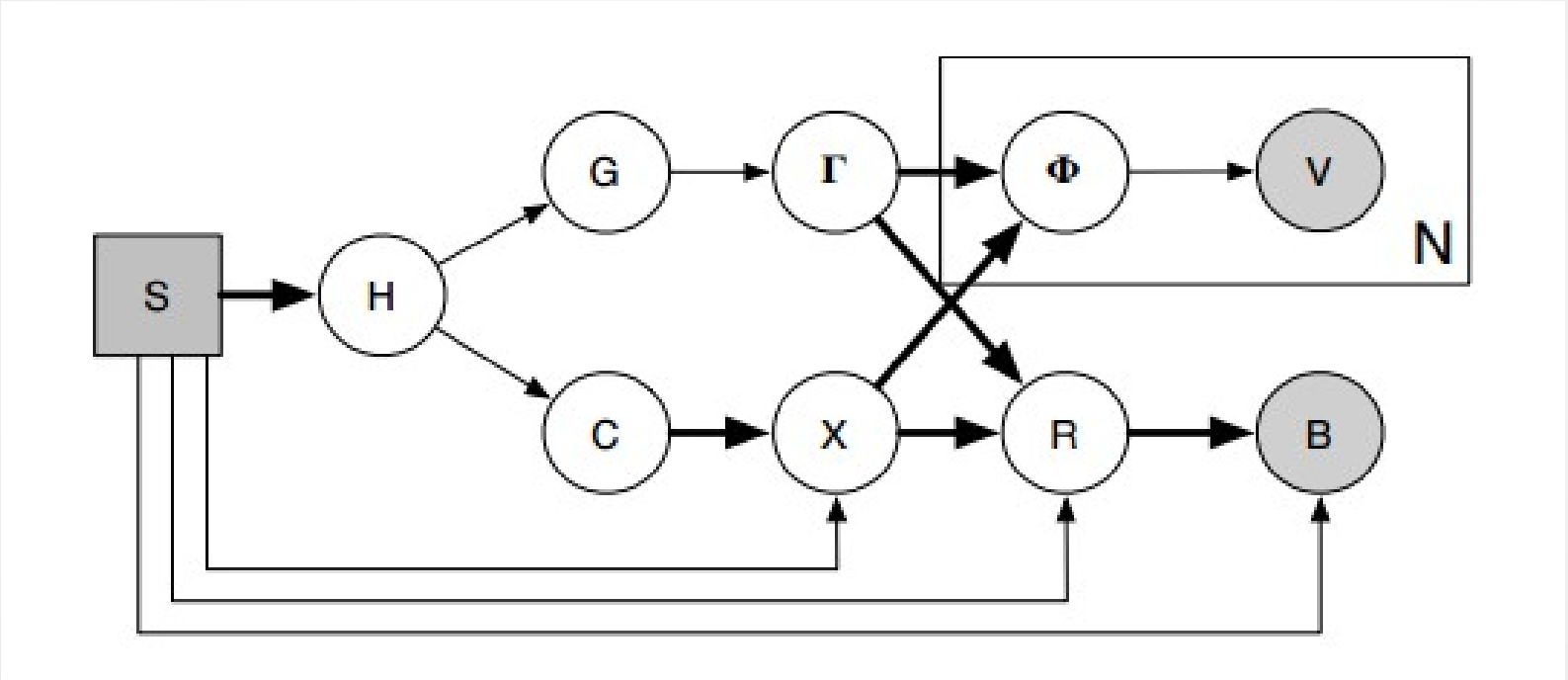
Inverse Reinforcement Learning

- Based on Maximum Likelihood Inverse Reinforcement Learning (MLIRL)
- Takes demonstration of agent behaving optimally
- Extracts a most probable reward function

Task Abstraction

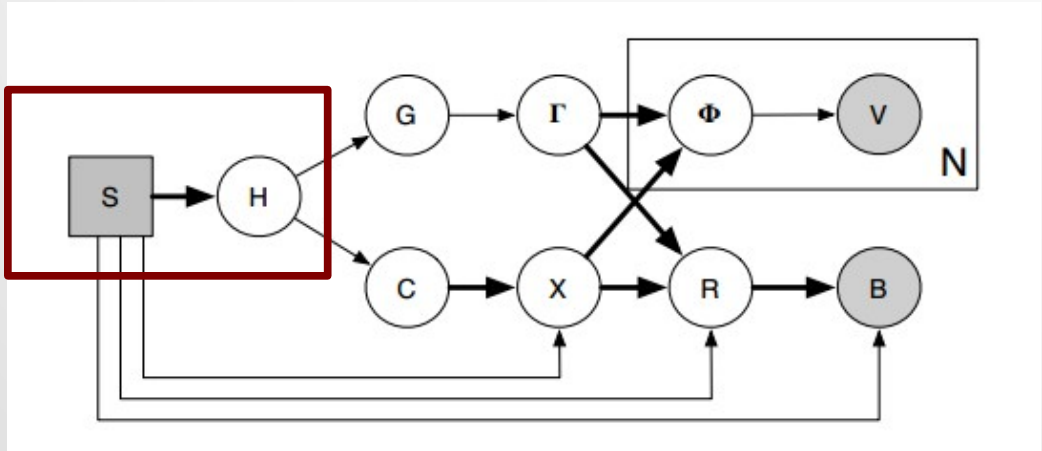
- Handles abstraction of domain into first-order logic
- Grounds generated first-order logic to domain
- Performs expectation maximization between SP and IRL

Generative Model



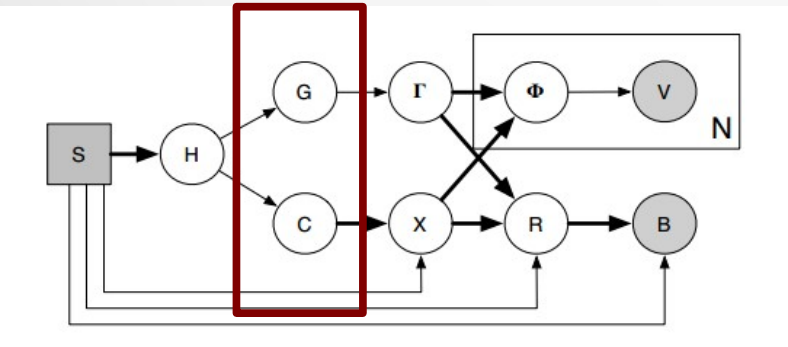
Generative Model

- **S**: initial state (input)
  - Allows determining relevant first-order logic
- **H**: hollow task
  - FOL variables and OO-MDP object classes
  - ∃b,r BLOCK(b)^ROOM(r)



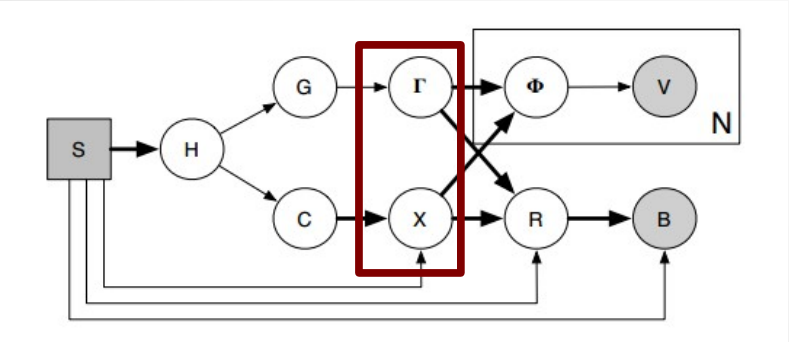
Generative Model

- **G**: abstract goal conditions
  - FOL variables and propositional function classes
  - blockPosition(b,r)
- **C**: abstract object binding (constraint)
  - FOL vars and prop. functions only guaranteed in initial state
  - roomColor(r) ∧ blockShape(b)



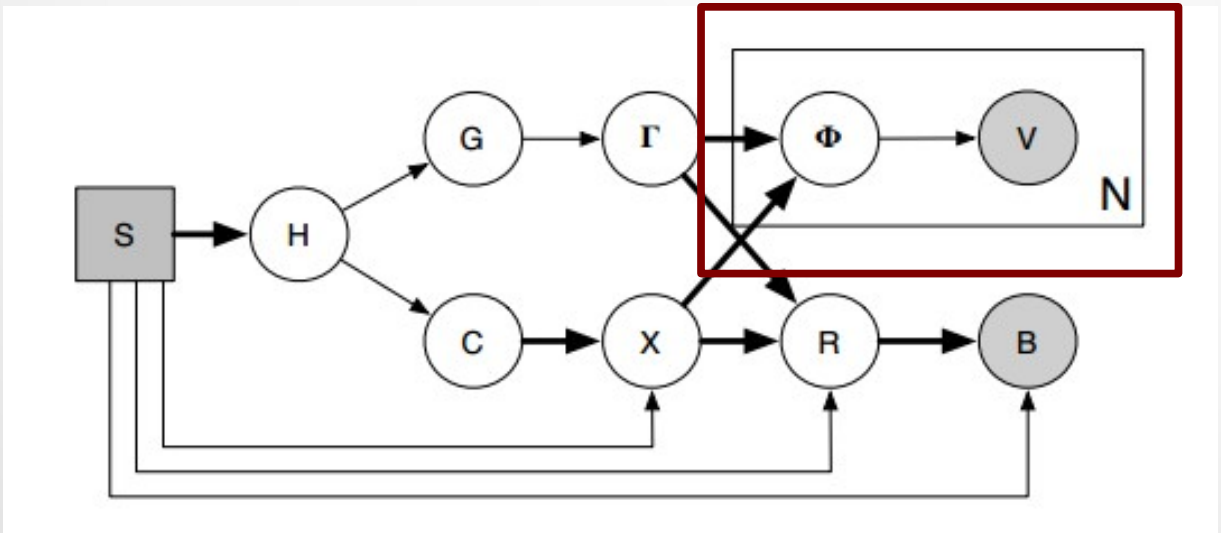
Generative Model

- **Γ**: object binding for *G*
  - Function instances of prop. function classes
  - blockInRoom(b, r)
- **X**: object binding for *C*
  - Function instances of prop. function classes
  - isGreen(r) ∧ isStar(b)



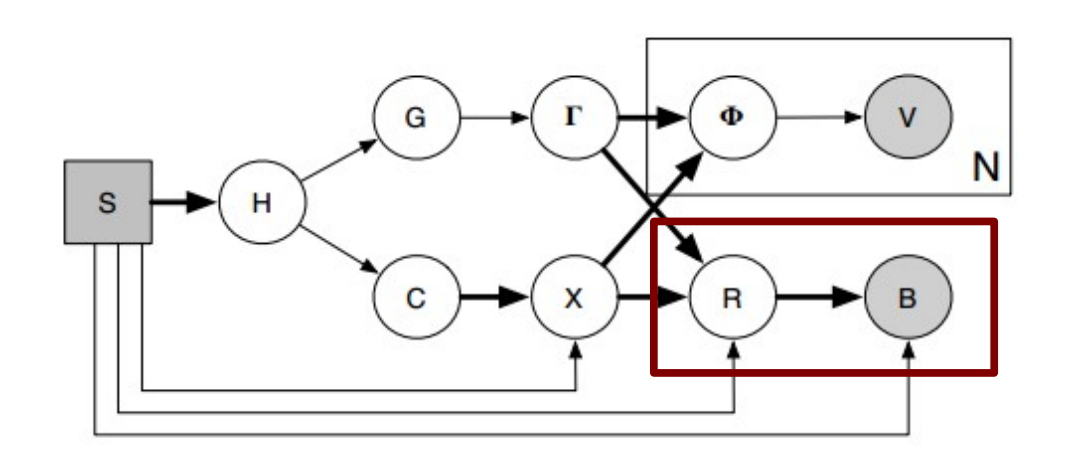
Generative Model

- **Φ**: randomly selected propositional function from **Γ** or **X**
  - blockInRoom, isGreen, or isStar
- **V**: a word from vocabulary
- **N**: number of words from *V* in a given command

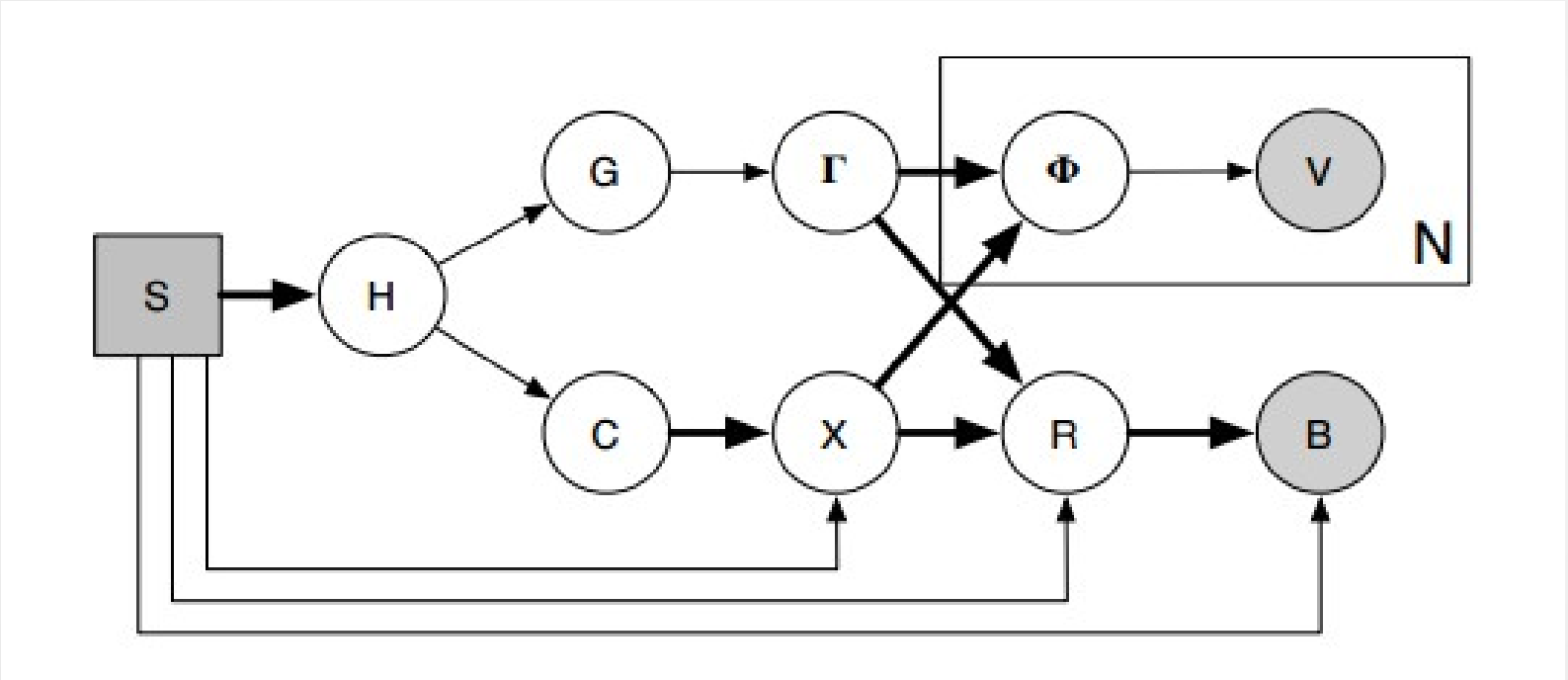


Generative Model

- **R**: reward function dictating behavior
  - Goal condition specified in **Γ** bound to objects in **X**
  - blockInRoom(block0, room2)
- **B**: behavioral trajectory
  - Starts in *S* and derived by *R*



Generative Model



Expectation Maximization

- Iterative method for maximum likelihood
- Uses observable variables
  - Initial state, behavior, and linguistic command
- Find distribution of latent variables
  - $Pr(g \mid h)$, $Pr(c \mid h)$, $Pr(y \mid g)$, and $Pr(v \mid \varphi)$
- Additive smoothing seems to have a positive effect

Training / Testing

- Gathered data from Mechanical Turk
  - 40 commands for each of 6 example tasks
- Sokoban test domain with leave-one-out cross validation
- Accuracy is computed as comparison of most likely reward function versus actual reward function
- 70%+ accuracy with 5 EM iterations

Current / Future Tasks

- Fully implement the system to build abstract tasks from language information and feature relevance
- Continue testing with Mechanical Turk data
- Implement richer language models
- Incorporate options to represent subgoals

Summary

- Learning tasks from verbal commands
- Uses generative model and expectation maximization
- Train using command and behavior
- Commands should generate correct task goal and behavior