Learning to Interpret Natural Language Instructions

Monica Babeş-Vroman[+] , James MacGlashan[*], Ruoyuan Gao[+] , Kevin Winner[*]
Richard Adjogah[*] , Marie desJardins[*] , Michael Littman[+] and **Smaranda Muresan**[++]

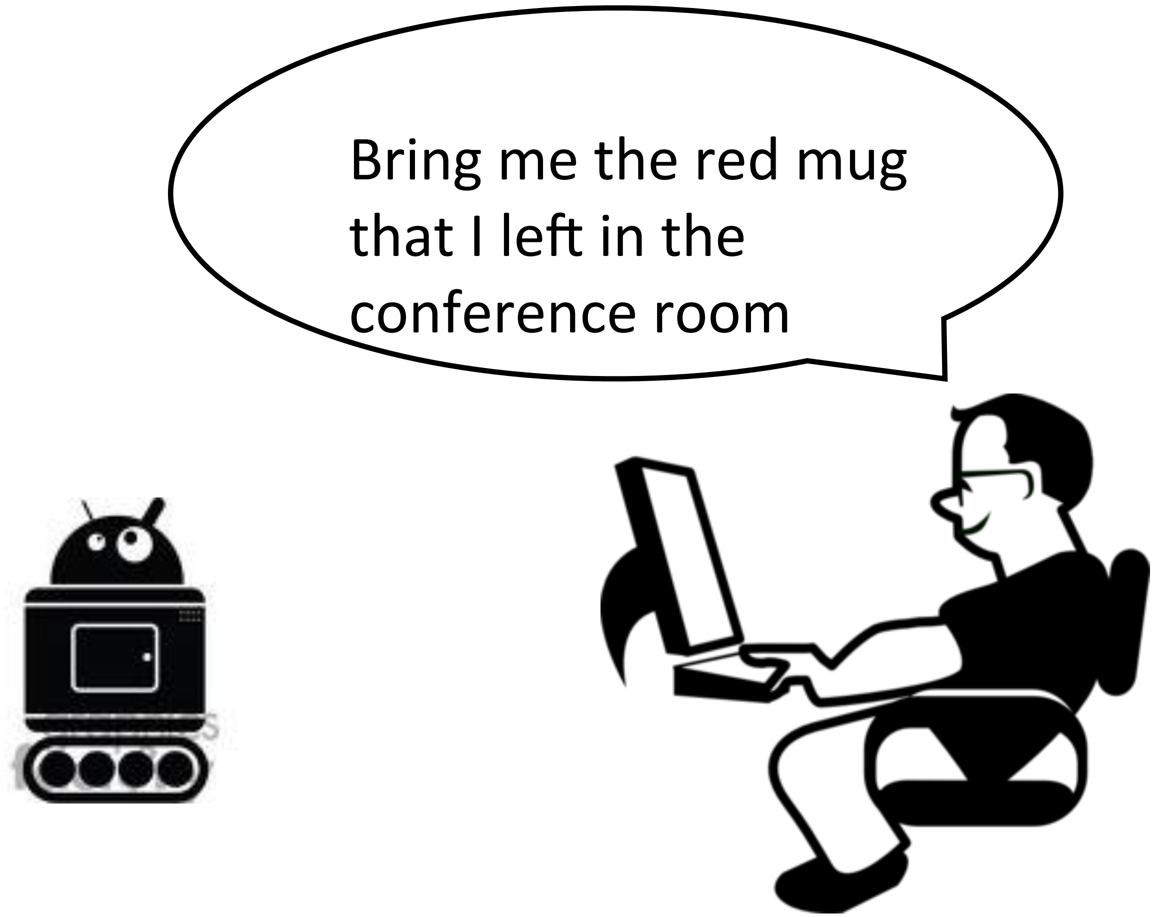
*Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
+Computer Science Department, Rutgers University
++School of Communication and Information, Rutgers University

NAACL-2012 WORKSHOP
From Words to Actions: Semantic Interpretation in an Actionable Context

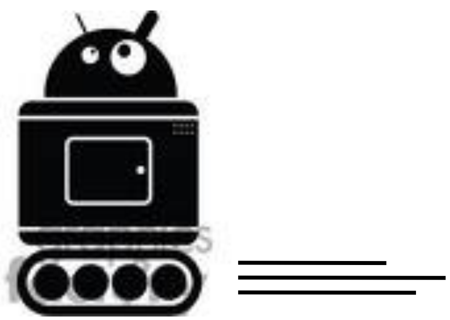
Outline

- Motivation and Problem
- Our approach
- Pilot study
- Conclusions and future work

Motivation



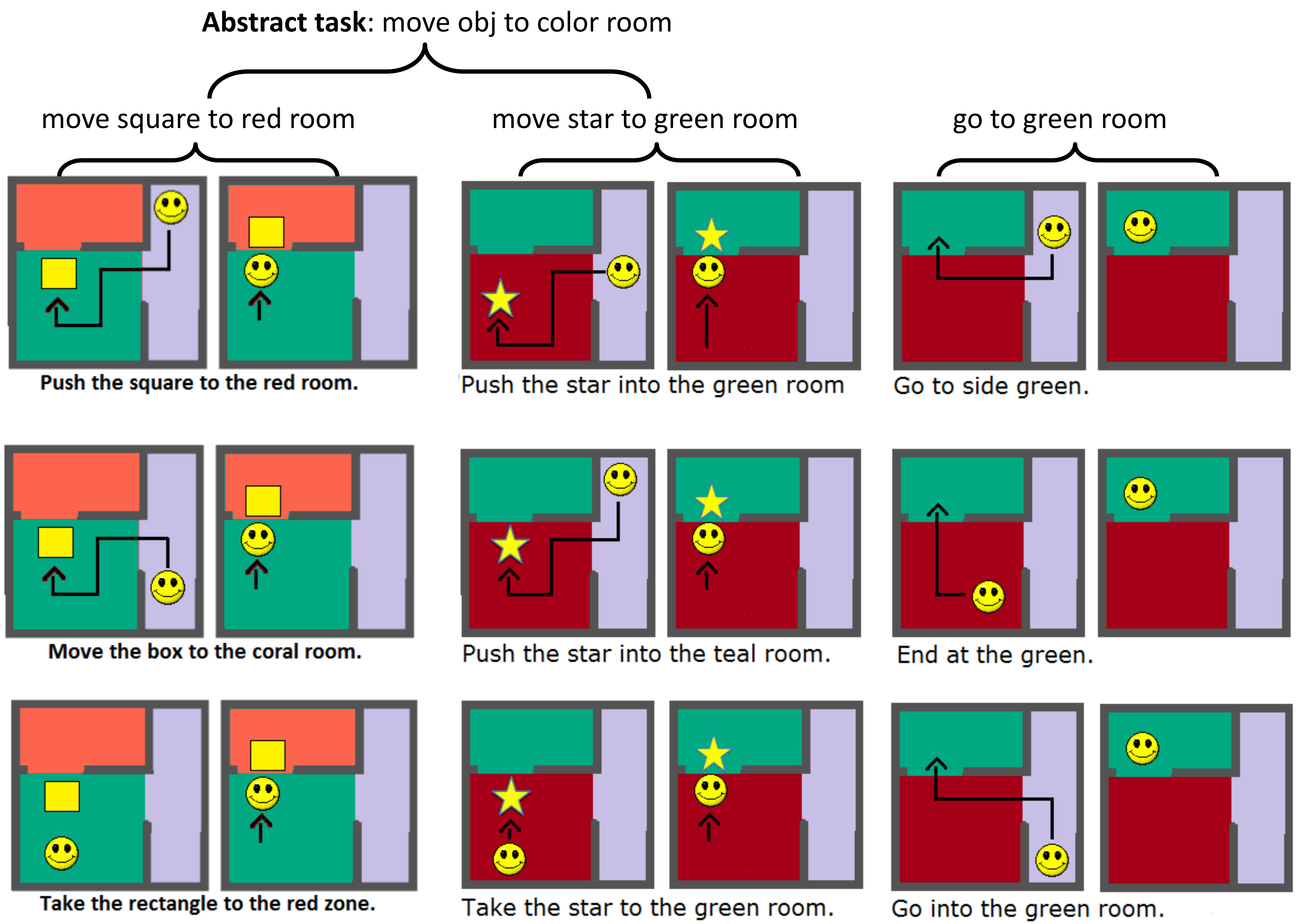
Motivation



Motivation

Problem

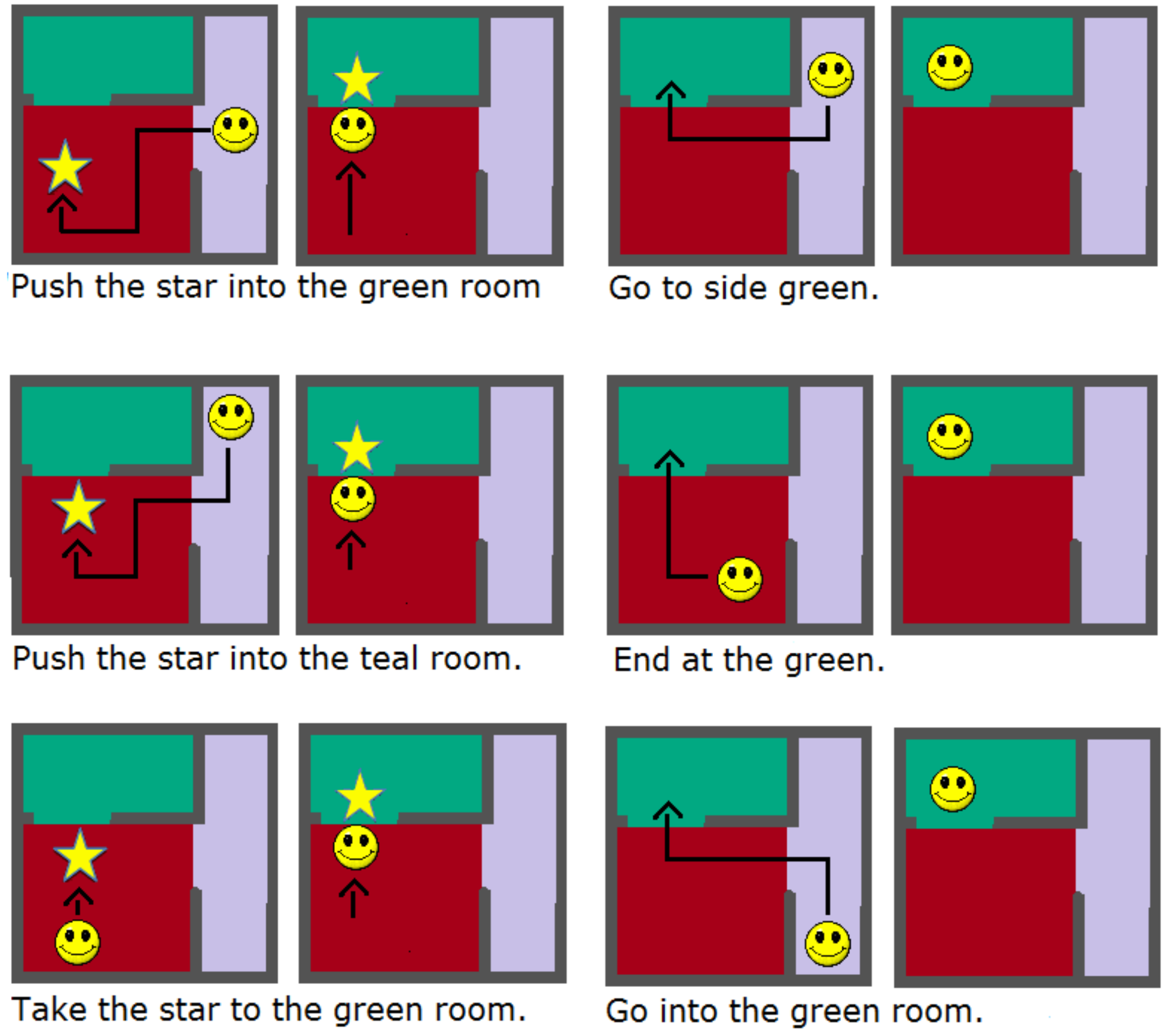
Train an artificial agent to learn to carry out complex multistep tasks specified in natural language.

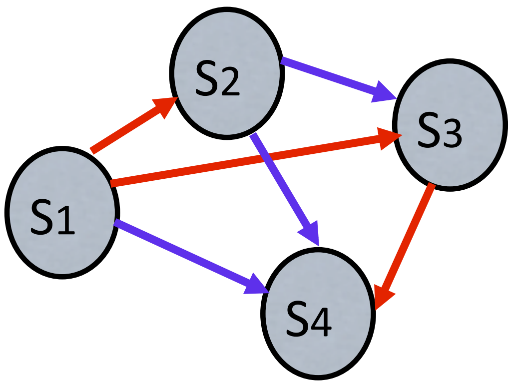

Abstract task: move obj to color room
move square to red room
move star to green room
go to green room
Push the square to the red room.
Push the star into the green room
Go to side green.
Move the box to the coral room.
Push the star into the teal room.
End at the green.
Take the rectangle to the red zone.
Take the star to the green room.
Go into the green room.

Outline

- What is the problem?
- Our approach
- Pilot study
- Conclusions and future work

Training Data:

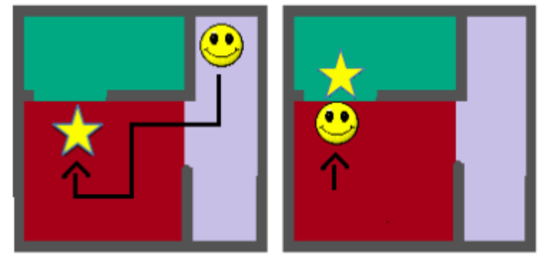




Object-oriented Markov Decision Process (OO-MDP)

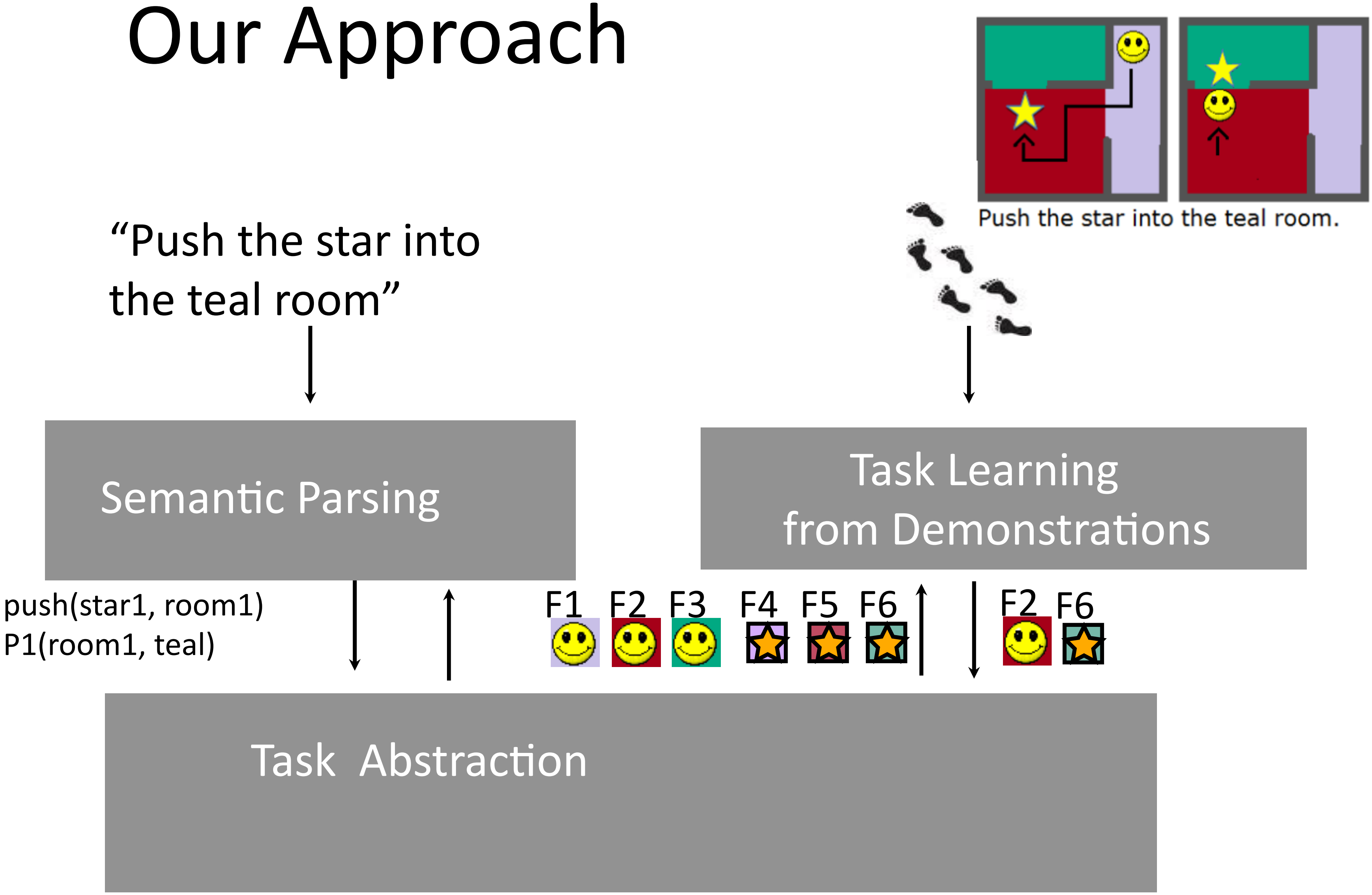
[Diuk et al., 2008]

“Push the star into the teal room”

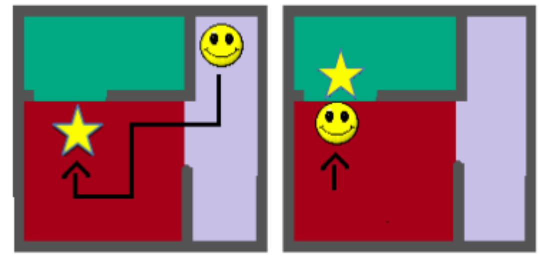


Push the star into the teal room.

Our Approach



Our Approach



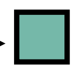
Push the star into the teal room.

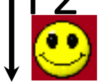
"Push the star into the teal room"

Semantic Parsing

Task Learning from Demonstrations

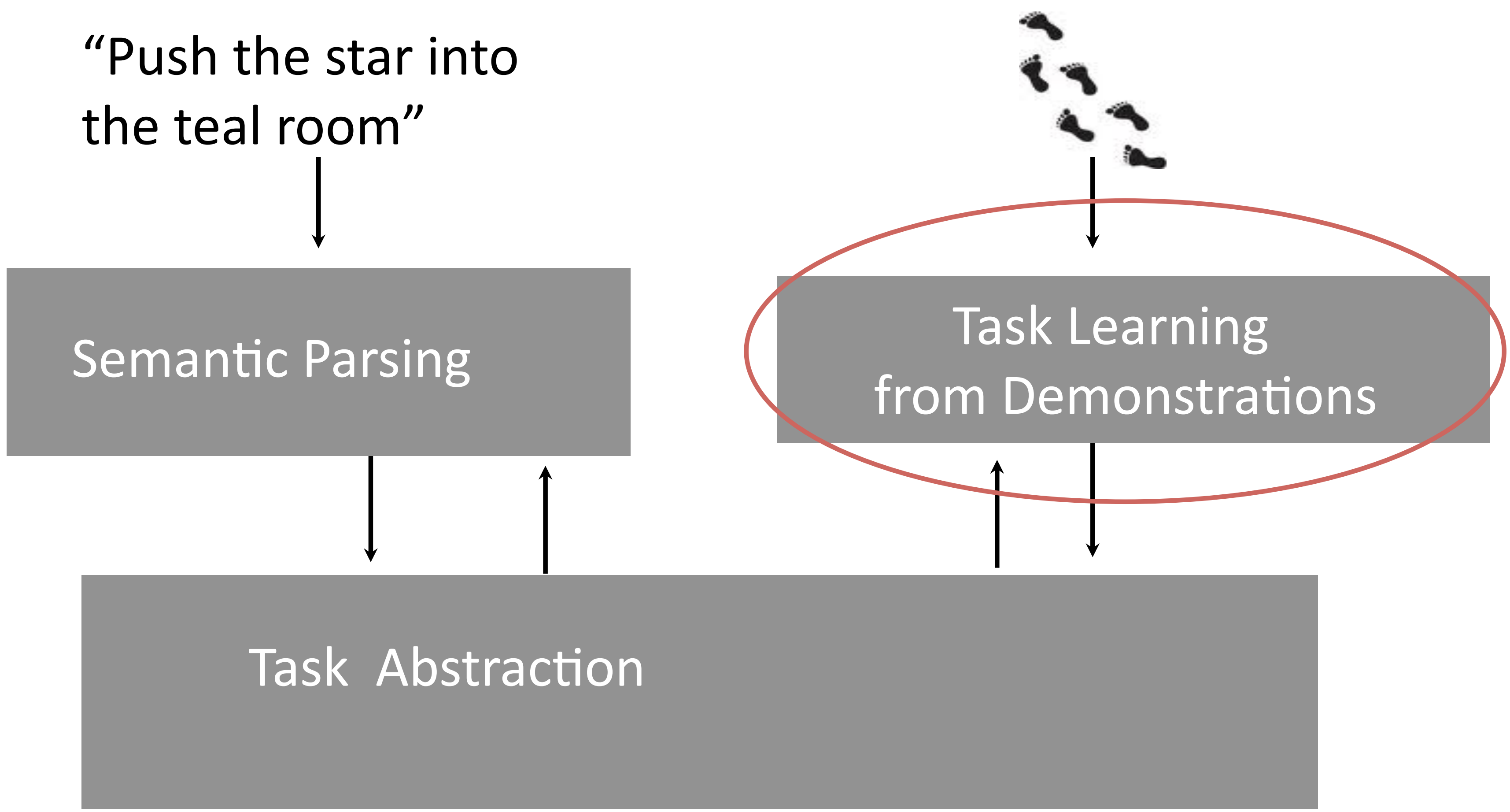
push(star1, room1)
P1(room1, teal)

"teal" → 
P1 → color
"push" → objToRoom

F2  F6 

Task Abstraction

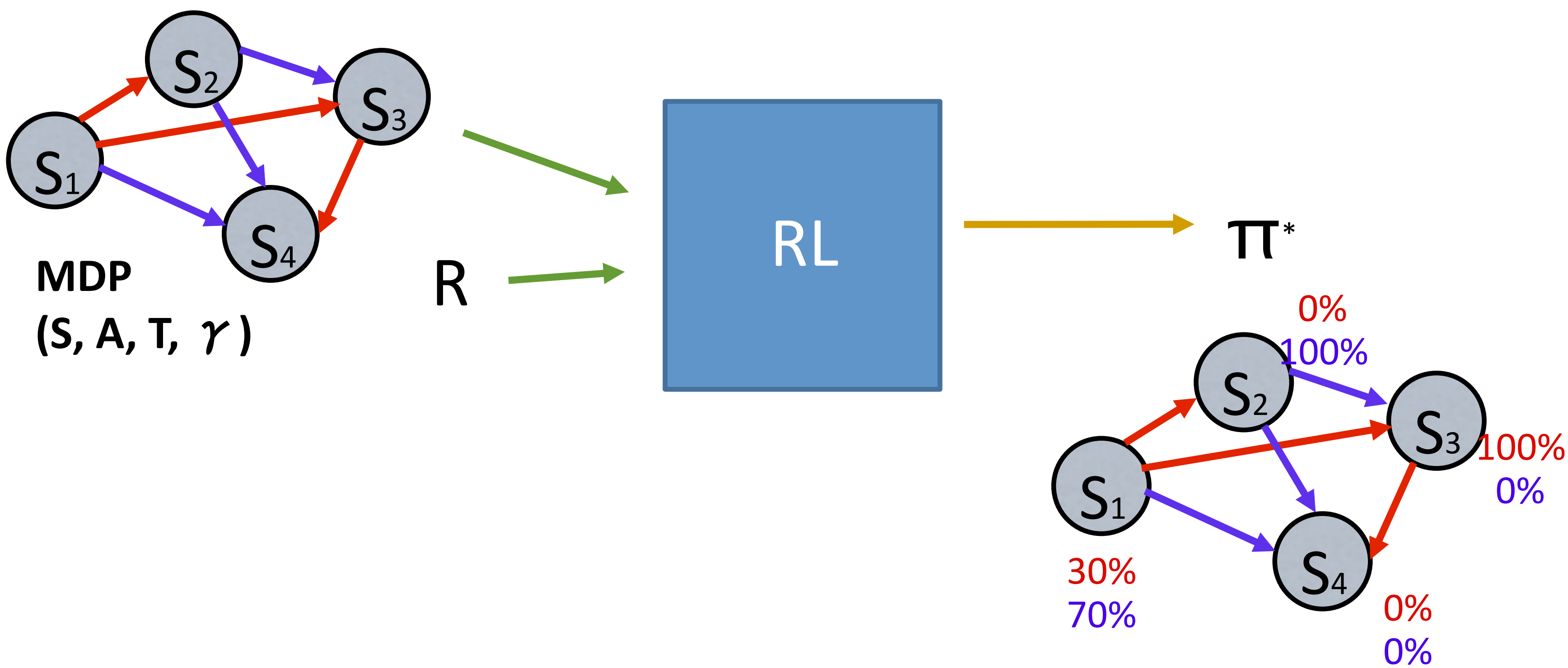
Our Approach



Task Learning from Demonstration

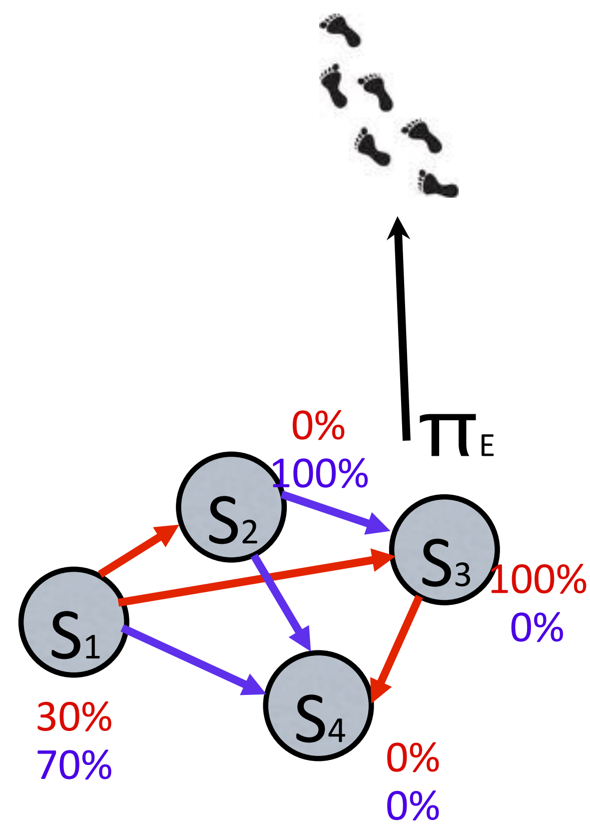
= Inverse Reinforcement Learning (IRL)

- But first let's see Reinforcement Learning Problem



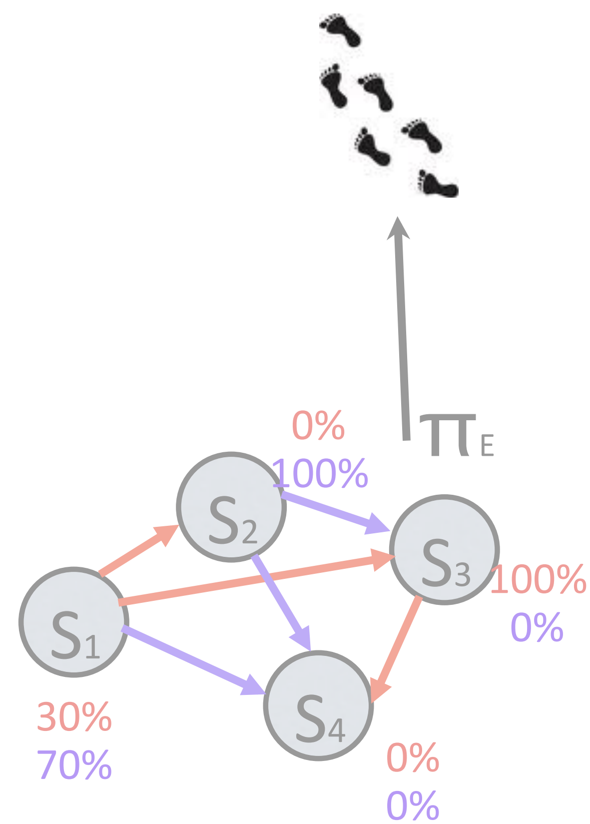
Task Learning From Demonstrations

- Inverse Reinforcement Learning



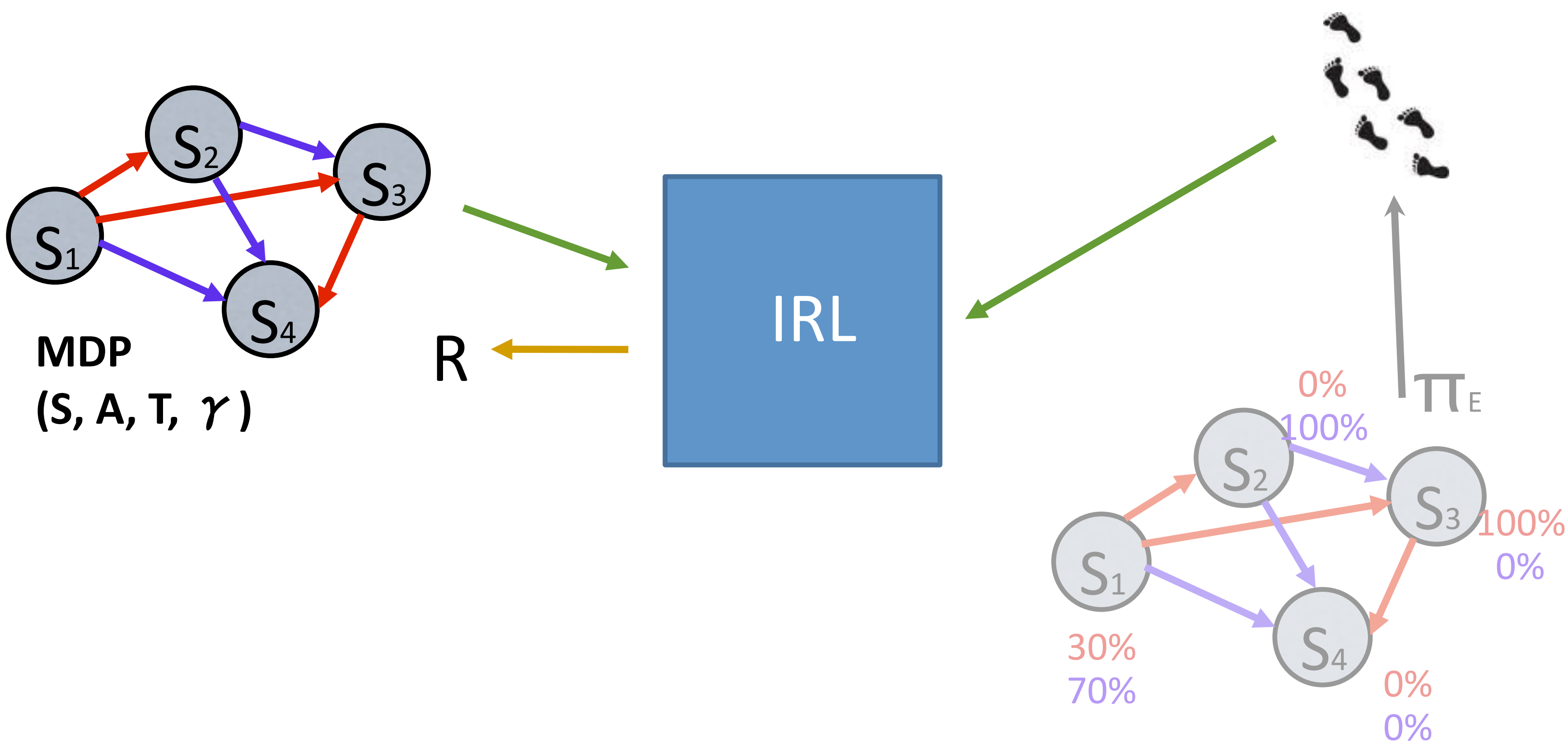
Task Learning From Demonstrations

- Inverse Reinforcement Learning



Task Learning From Demonstrations

- Inverse Reinforcement Learning



MLIRL

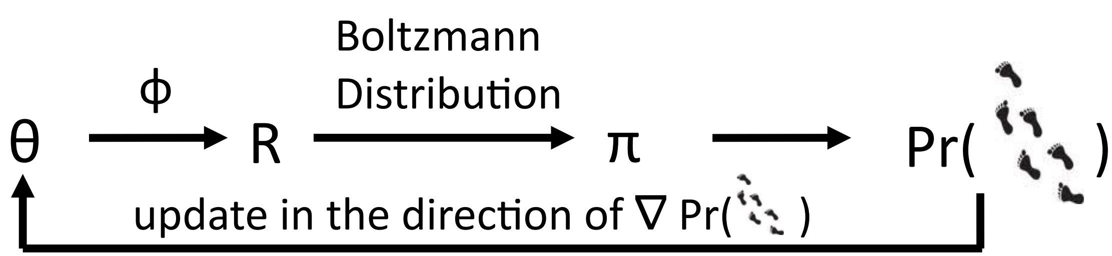
- Maximum Likelihood IRL

[M. Babeş, V. Marivate, K. Subramanian and M. Littman 2011]

Assumption: Reward function is a linear combination of a known set of features. (e.g., )

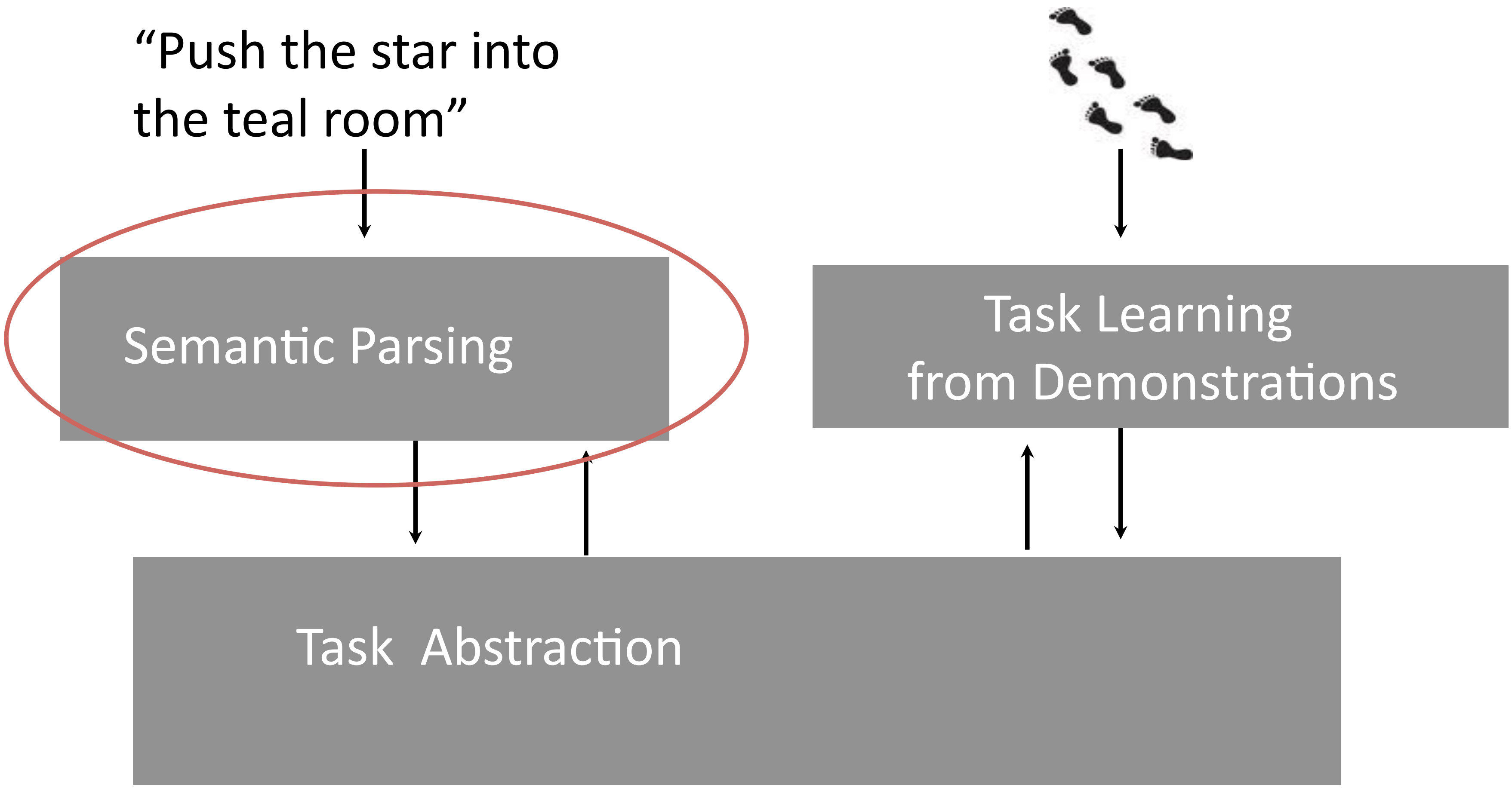
$$R(s,a) = \sum_i \vartheta_i \varphi_i(s,a) = \theta \times \phi(s,a)$$

Goal: Find θ to maximize the probability of the observed trajectories.



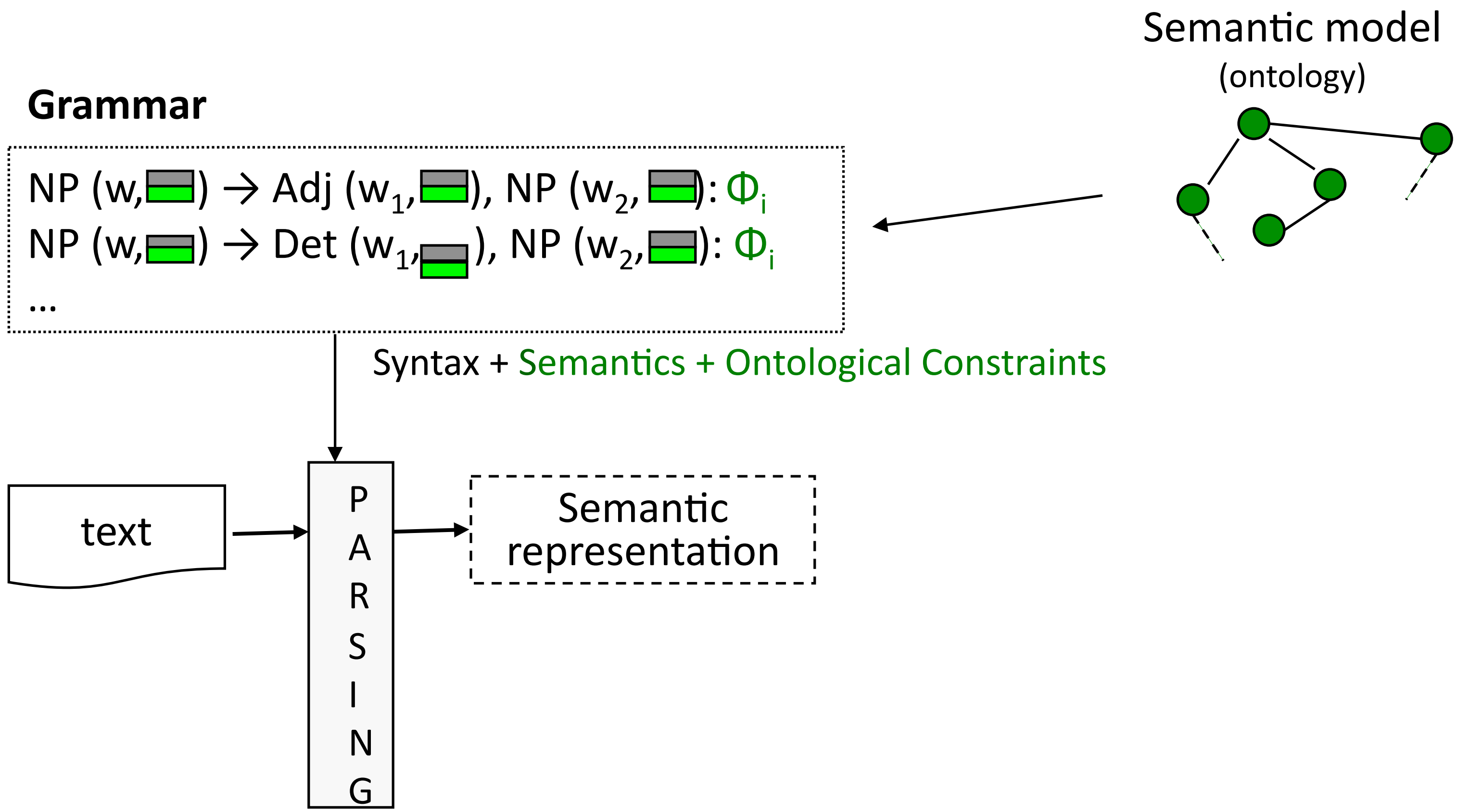
(s,a): (state, action) pair
R: reward function
ϕ: feature vector
π: policy
Pr( ): probability of demonstrations

Our Approach



Semantic Parsing

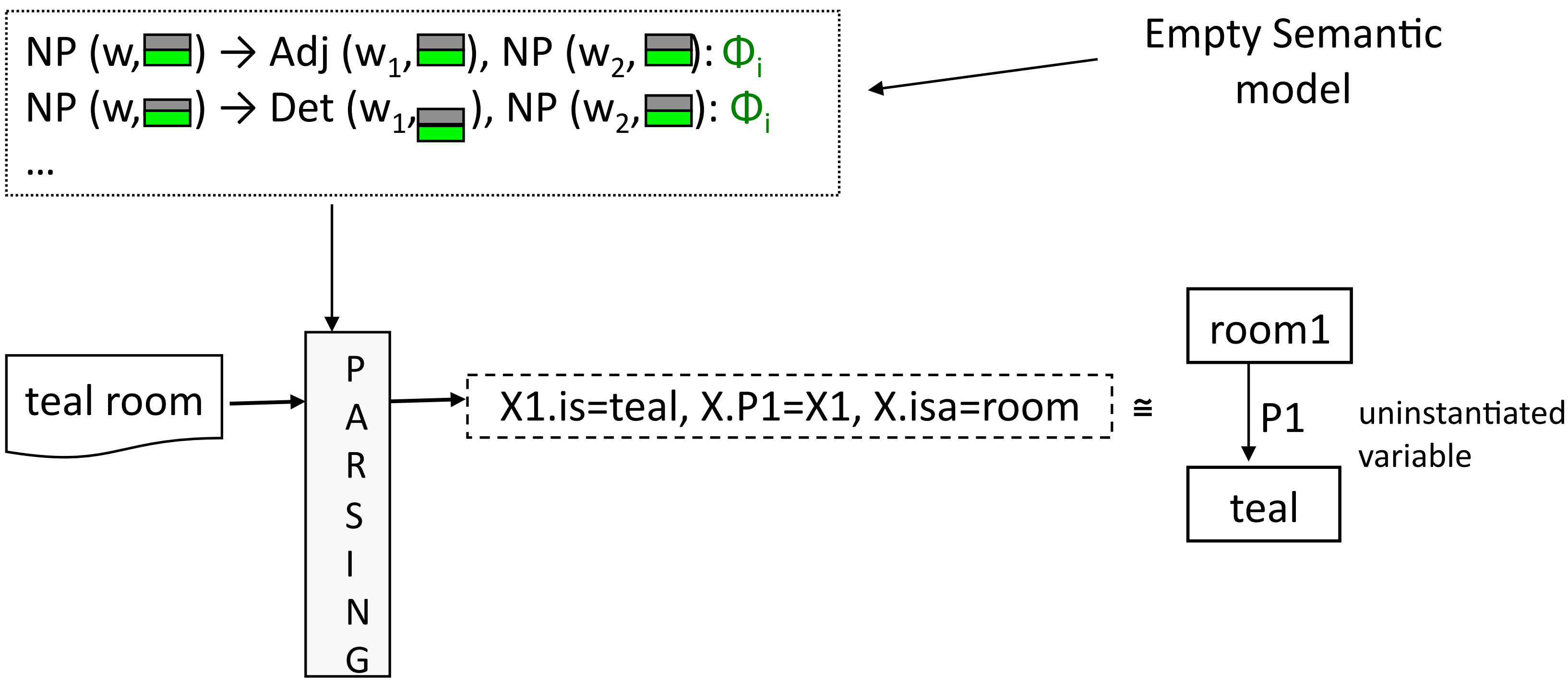
- Lexicalized Well-Founded Grammars (LWFG)
  [Muresan, 2006; 2008; 2012]



Semantic Parsing

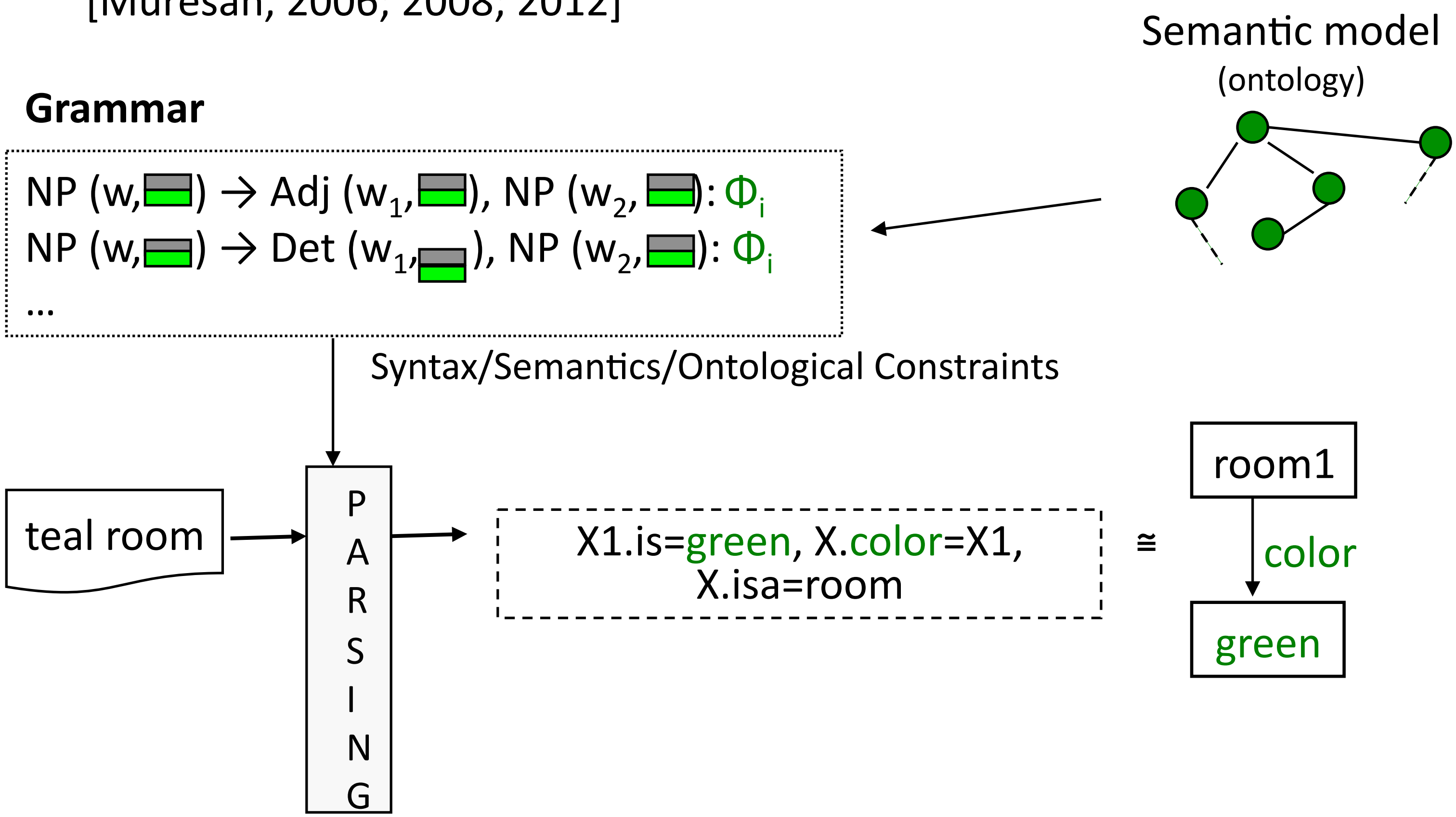
- Lexicalized Well-Founded Grammars (LWFG)
  [Muresan, 2006; 2008; 2012]

Grammar



Semantic Parsing

- Lexicalized Well-Founded Grammars (LWFG)
  [Muresan, 2006; 2008; 2012]

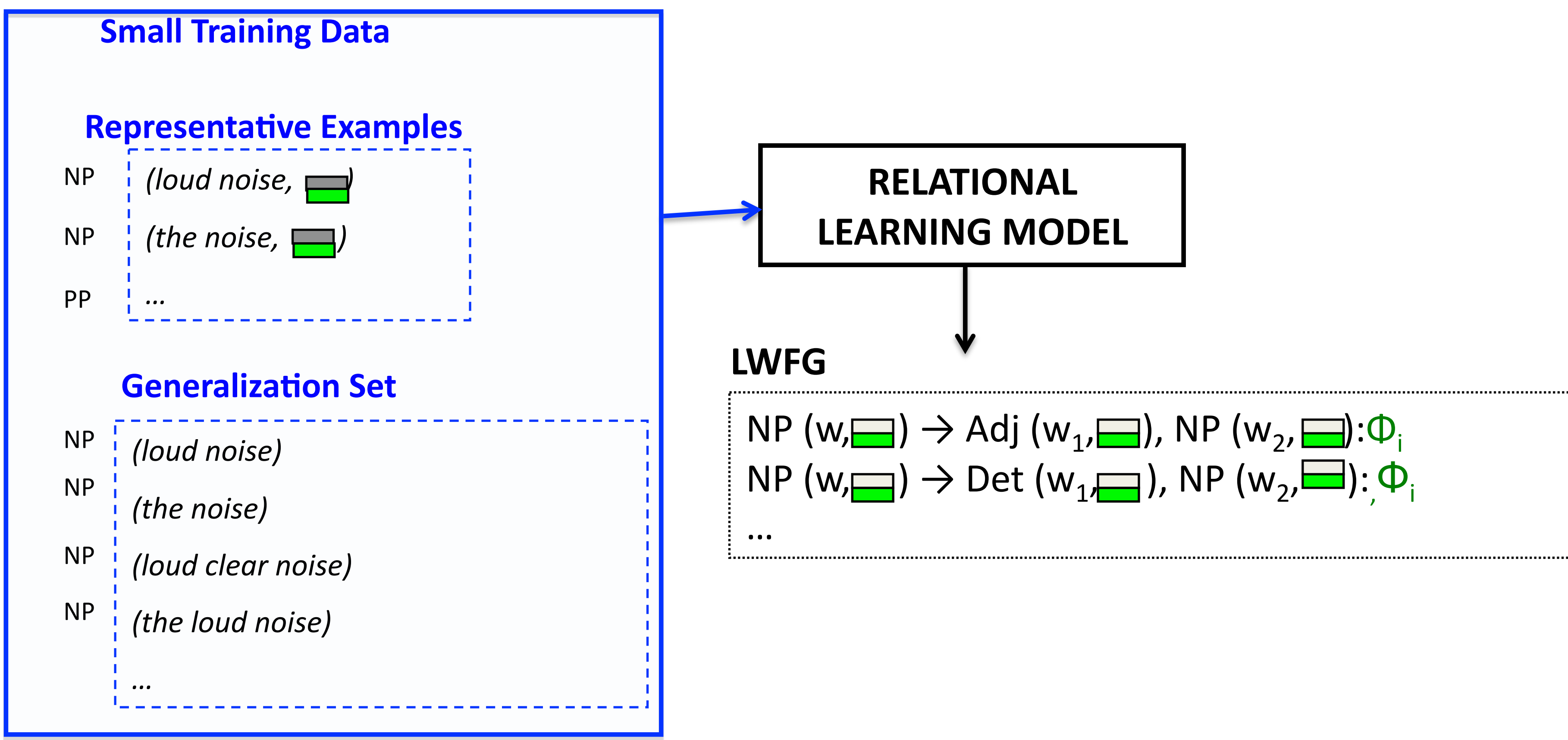


Learning LWFG

- [Muresan and Rambow 2007; Muresan, 2010; 2011]

  Currently learning is done offline

  No assumption of existing semantic model



Using LWFG for Semantic Parsing

- Obtain representation with same underlying meaning despite different syntactic forms

  "go to the green room" vs *"go to the room that is green"*

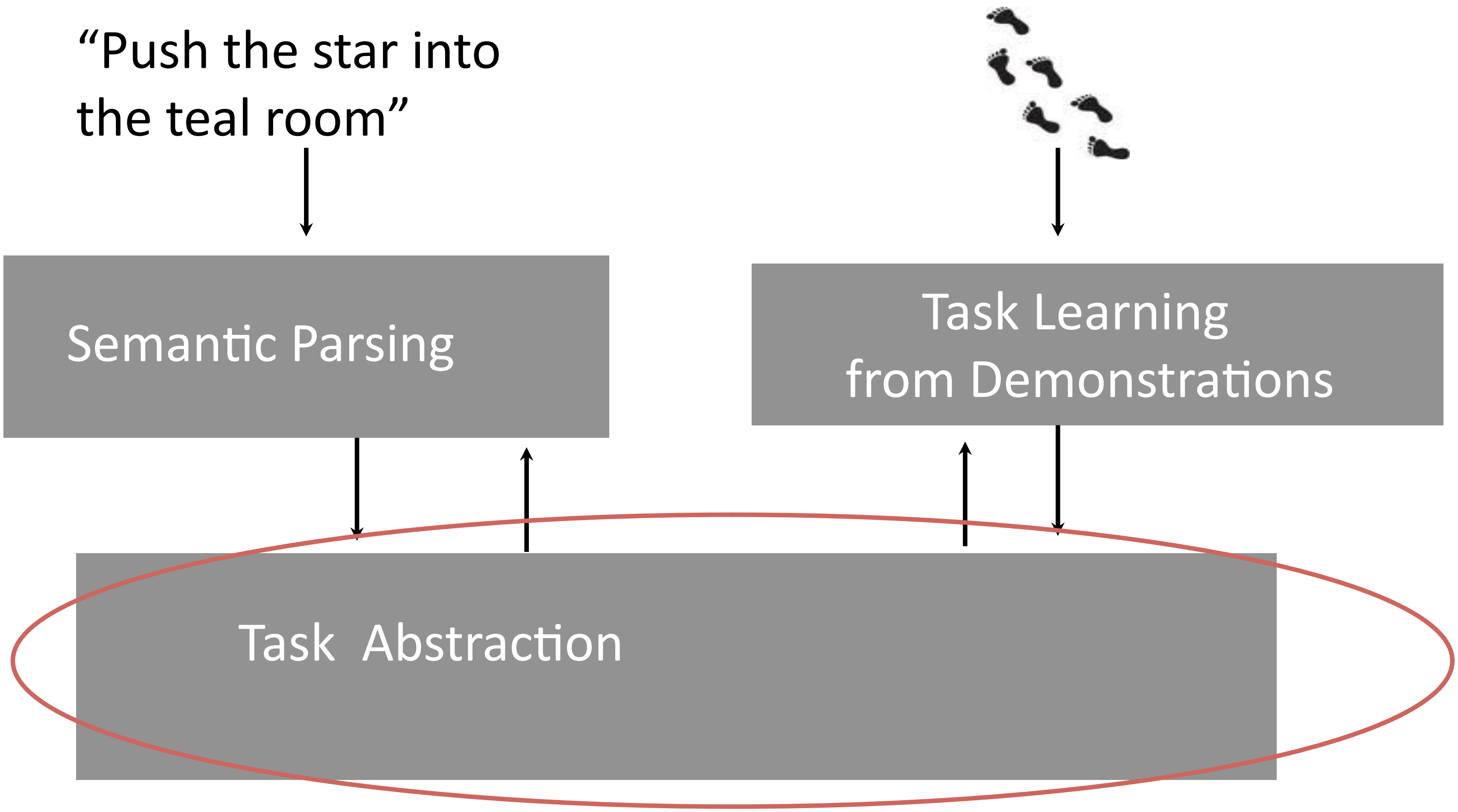
  – Dynamic Semantic Model

  Can start with an empty semantic model

  Augmented based on feedback from the TA component

  push →blockToRoom, teal → green

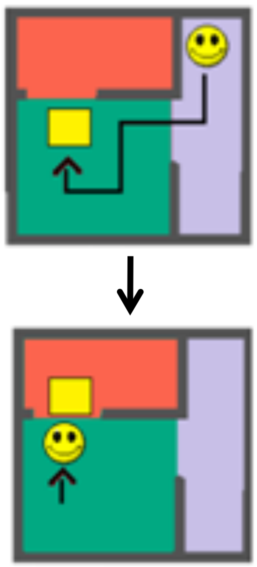
Our Approach



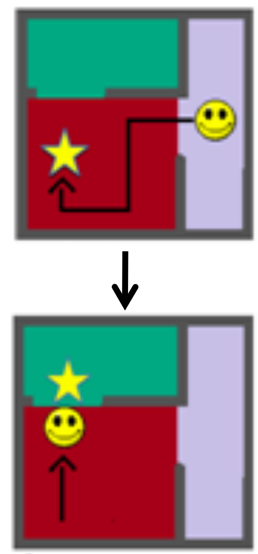
Task Abstraction

- Creates an abstract task to represent a set of similar *grounded* tasks
  - Specifies goal conditions and reward function in terms of OO-MDP propositional functions
- Combines information from SP and IRL and provides feedback to both
  - Provides SP domain specific semantic information
  - Provides IRL relevant features

Similar Grounded Tasks



Goal Condition
objectIn(o2, l1)
Other Terminal Facts
isSquare(o2)
isYellow(o2)
isRed(l1)
isGreen(l2)
agentIn(o46, l2)
…



Goal Condition
objectIn(o15, l2)
Other Terminal Facts
isStar(o15)
isYellow(o15)
isGreen(l2)
isRed(l3)
objectIn(o31, l3)
…

Some grounded tasks are similar in logical goal conditions, but different in object references and facts about those referenced objects

Abstract Task Definition

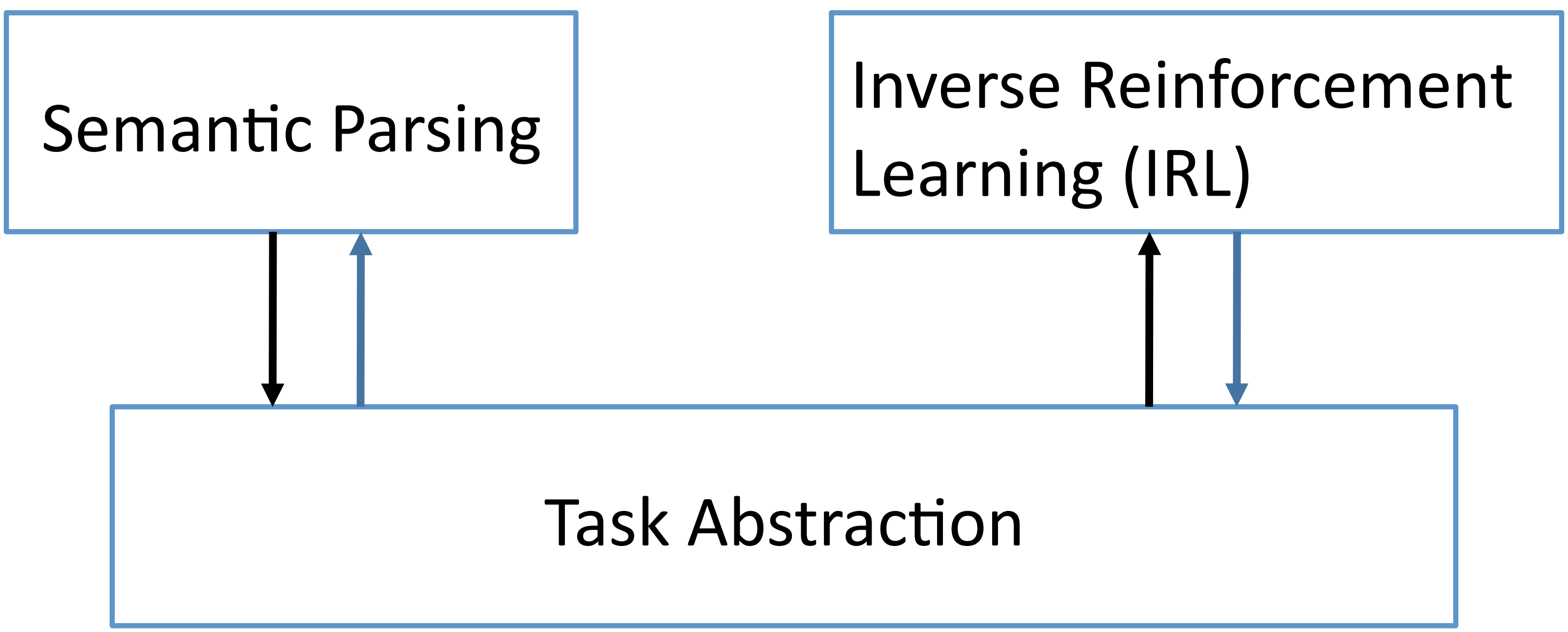
- Define task conditions as a partial first order logic expression

$$\exists_{o \in Object,\, l \in Location}\ \mathrm{ObjectIn}(o,l)$$

- The expression takes propositional function parameters to complete it

$$\exists_{o \in Object,\, l \in Location}\ \mathrm{ObjectIn}(o,l) \wedge$$
$$\mathrm{isYellow}(o) \wedge \mathrm{isStar}(o) \wedge \mathrm{isGreen}(l)$$

System Structure



Pilot Study

Unigram Language Model

Data: (S_i, T_i).

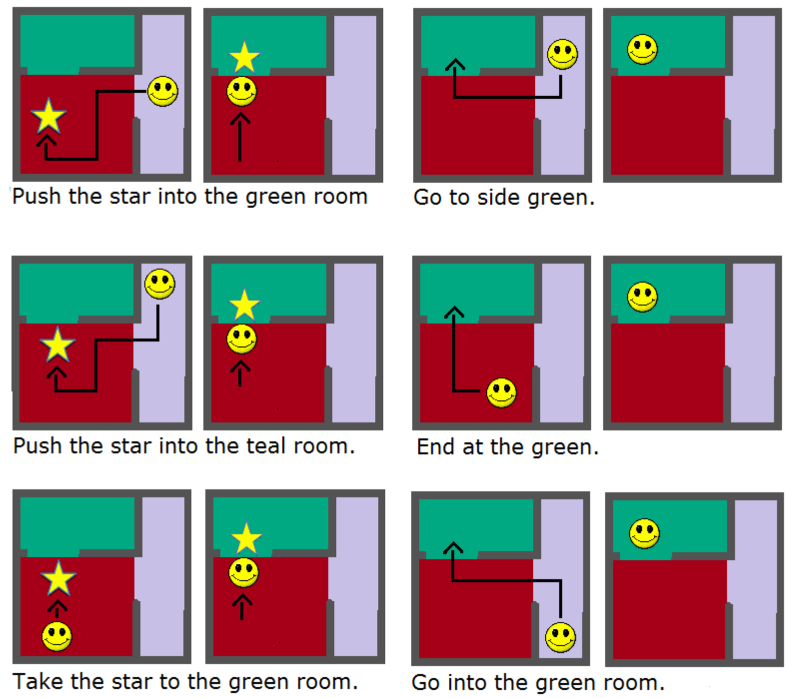
Hidden Data: z_{ji} : $\Pr(R_j \mid (S_i, T_i))$.

Parameters: x_{kj} : $\Pr(w_k \mid R_j)$.

Algorithm 3 EM Model 0

Input: Demonstrations $\{(S_1, T_1), ..., (S_N, T_N)\}$, number of reward functions J, size of vocabulary K.
Initialize: $x_{11}, \ldots, x_{JK}$, randomly.
repeat
 E Step: Compute
 $z_{ji} = \frac{\Pr(R_j)}{\Pr(S_i, T_i)} \Pr(T_i | R_j) \prod_{w_k \in S_i} x_{kj}$.

 M step: Compute
 $x_{kj} = \frac{1}{X} \frac{\Sigma_{w_k \in S_i} \Pr(R_j | S_i) + \epsilon}{\Sigma_i N(S_i) z_{ji} + \epsilon}$.
until target number of iterations completed.



Pilot Study

Unigram Language Model

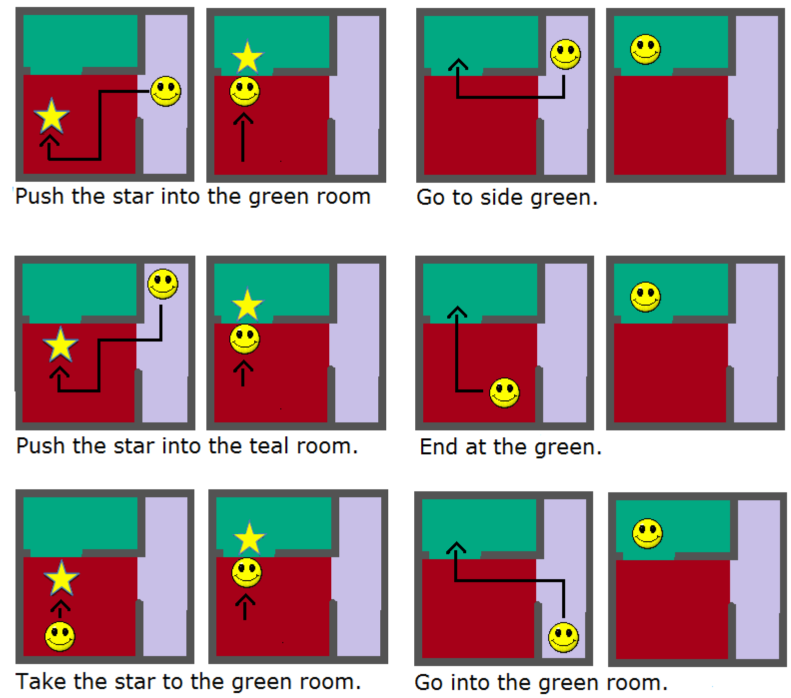
Data: (S_i, T_i).

Hidden Data: z_{ji} : $\Pr(R_j \mid (S_i, T_i))$.

Parameters: x_{kj} : $\Pr(w_k \mid R_j)$.

Algorithm 3 EM Model 0

Input: Demonstrations $\{(S_1, T_1), ..., (S_N, T_N)\}$, number of reward functions J, size of vocabulary K.
Initialize: $x_{11}, \ldots, x_{JK}$, randomly.
repeat
  E Step: Compute
  $z_{ji} = \frac{\Pr(R_j)}{\Pr(S_i, T_i)} \Pr(T_i | R_j) \prod_{w_k \in S_i} x_{kj}$.
  M step: Compute
  $x_{kj} = \frac{1}{X} \frac{\Sigma_{w_k \in S_i} \Pr(R_j | S_i) + \epsilon}{\Sigma_i N(S_i) z_{ji} + \epsilon}$.
until target number of iterations completed.



Pr("push"|R1) … Pr("room"|R1)

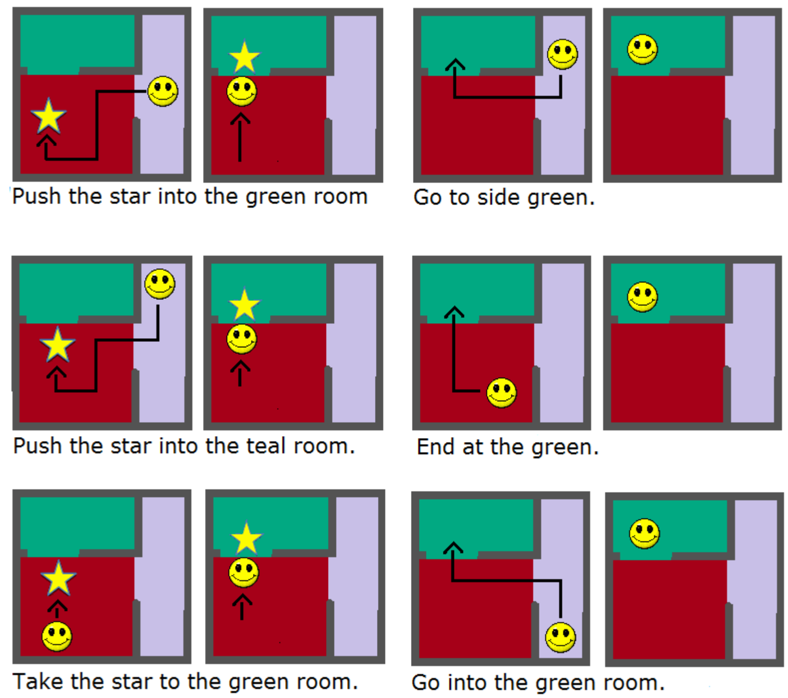
Pr("push"|R2) … Pr("room"|R2)

Pilot Study

New Instruction, SN : “Go to the green room.”

$$\Pr(S_N \mid R_1) = \prod_{w_k \in S_N} \Pr(w_k \mid R_1) = 8.6 \times 10^{-7},$$

$$\Pr(S_N \mid R_2) = \prod_{w_k \in S_N} \Pr(w_k \mid R_2) = 4.1 \times 10^{-4}$$



Pr(“push”|R1) ··· Pr(“room”|R1)

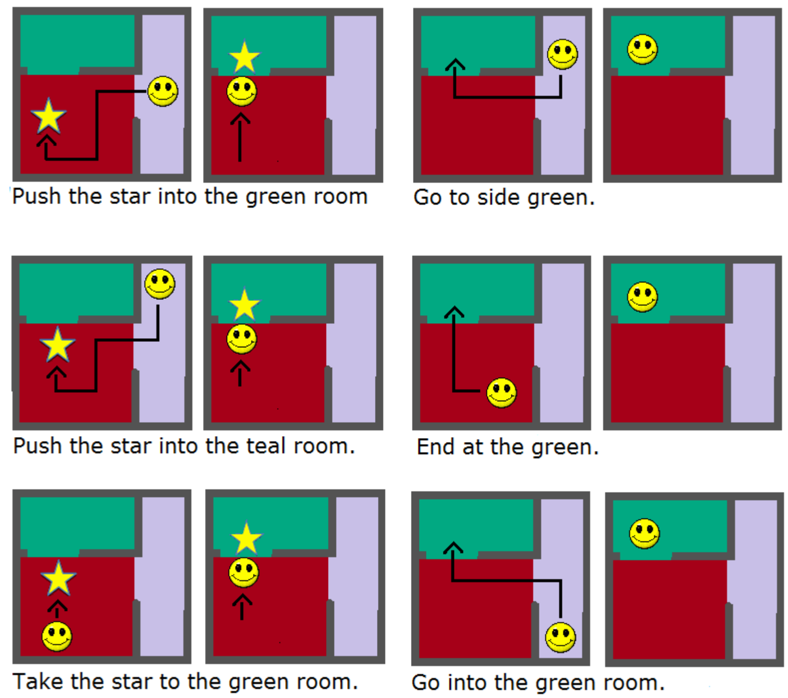
Pr(“push”|R2) ··· Pr(“room”|R2)

Pilot Study

New Instruction, S'N : "Go with the star to green."

$$\Pr(S'_N \mid R_1) = \prod_{w_k \in S'_N} \Pr(w_k \mid R_1) = 8.3 \times 10^{-7},$$

$$\Pr(S'_N \mid R_2) = \prod_{w_k \in S'_N} \Pr(w_k \mid R_2) = 2.1 \times 10^{-5}$$



Pr("push"|R1) ··· Pr("room"|R1)

Pr("push"|R2) ··· Pr("room"|R2)

Outline

- What is the problem?
- Our approach
- Pilot study
- Conclusions and future work

Conclusions

- Proposed new approach for training an artificial agent to learn to carry out complex multistep tasks specified in natural language, from pairs of instructions and demonstrations
- Showed pilot study with a simplified model

Current/Future Work

- SP, TA and IRL integration
- High-level multistep tasks
- Probabilistic LWFGs
  - Probabilistic domain (semantic) model
  - Learning/Parsing algorithms with both hard and soft constraints
- TA – add support for hierarchical task definitions that can handle temporal conditions
- IRL – find a partitioning of the execution trace where each partition has it own reward function

Thank you!