Self-Directed Learning of Action Models using Exploratory Planning

Dustin Dannenhauer

Matthew Molineaux

Michael W. Floyd

Noah Reifsnyder

David W. Aha

Slack channel:
#paper29-dannenhauer

ACS 2021

Outline

- Research Problem
- Lifted Linked Clause (LLC)
- Self-Directed Exploration
- Empirical Results
- Summary

Research Problem

Motivation:

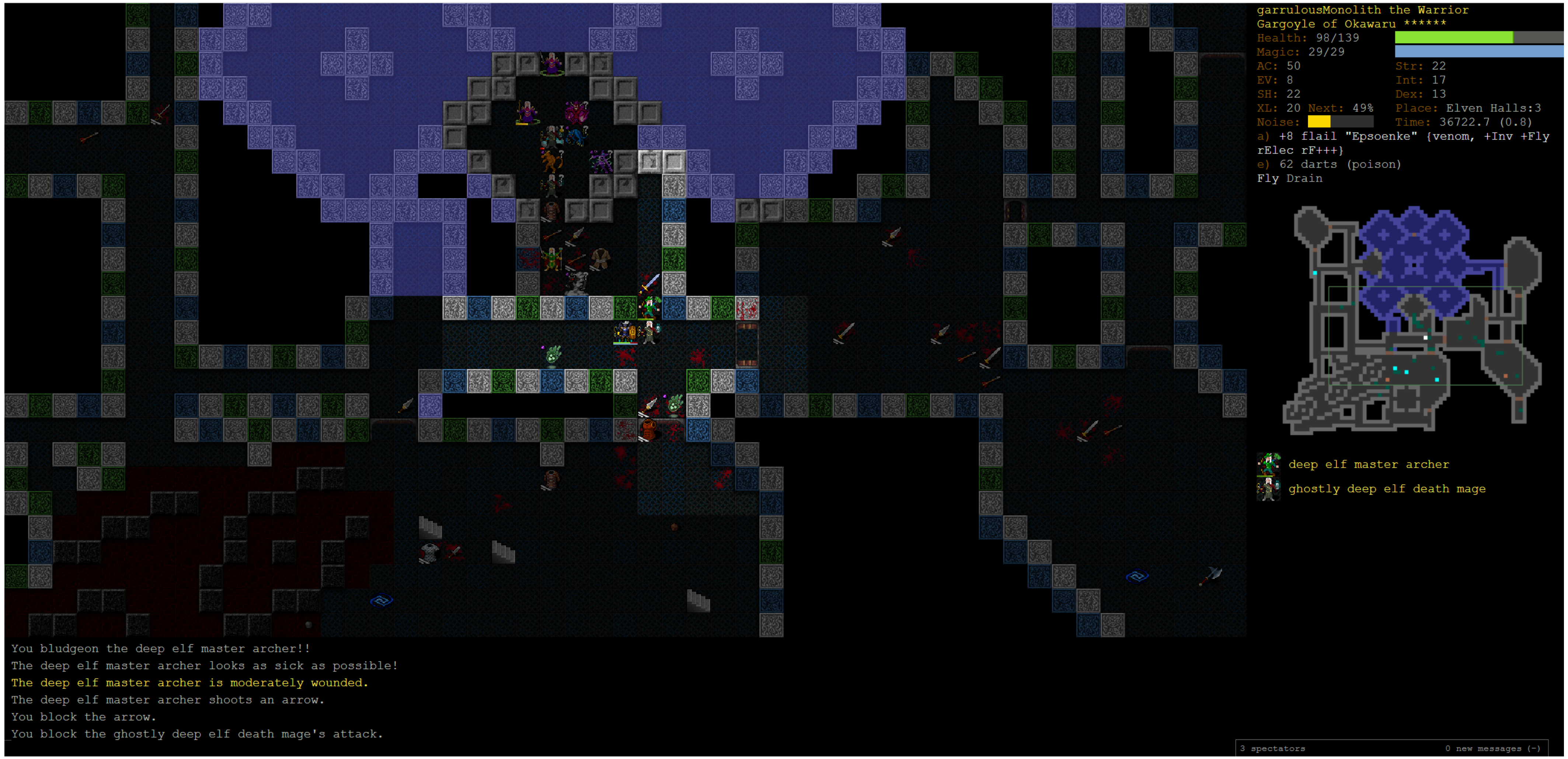
- Agents capable of learning in unknown environments
- Building an exploration-focused planning system

Assumptions:

- Agent does not require a goal, but can be tasked in a goal-directed manner
- Agent does not require a complete action model
- Agent does not have an extrinsic reward function
- Symbolic state representation

For more details on DCSS see:
https://github.com/dtdannen/dcss-ai-wrapper

An Example Domain: Dungeon Crawl Stone Soup (DCSS)



```
garrulousMonolith the Warrior
Gargoyle of Okawaru ******
Health: 98/139
Magic: 29/29
AC: 50                 Str: 22
EV: 8                  Int: 17
SH: 22                 Dex: 13
XL: 20 Next: 49%       Place: Elven Halls:3
Noise:                 Time: 36722.7 (0.8)
a) +8 flail "Epsoenke" {venom, +Inv +Fly
rElec rF+++}
e) 62 darts (poison)
Fly Drain
```

```
deep elf master archer
ghostly deep elf death mage
```

```
You bludgeon the deep elf master archer!!
The deep elf master archer looks as sick as possible!
The deep elf master archer is moderately wounded.
The deep elf master archer shoots an arrow.
You block the arrow.
You block the ghostly deep elf death mage's attack.
```

```
3 spectators                                   0 new messages (-)
```

Motivation for Real-World Domains

- Complex real-world domains may not be fully modelled
- Domains may change over time (open-world novelty)
- Maintaining an accurate action model will aid in task achievement

Related Work

<u>Intrinsically motivated reinforcement learning</u>

- Exploration-based rewards (Hester and Stone 2017)
- Intrinsic motivations based on appraisal dimensions: novelty, motivation, control, and valence (Sequeira, Melo, and Paiva 2011)

Difference from intrinsic RL – we consider a goal-directed agent

<u>Inductive Learning for constructing action models</u>

- Heuristic guided search (Hayes-roth and McDermott 1978; Vere 1980; Watanabe and Rendell 1990)
- Greedy algorithms: FOIL (Quinlan 1990)
- Learning from expert traces: OBSERVER (Wang, 1995)

Difference from prior inductive learning of action models work:

- Eliminating the need for expert traces
- Agent actively performs information gathering by finding new situations and trying actions

Outline

- Research Problem
- **Lifted Linked Clause (LLC)**
- Self-Directed Exploration
- Empirical Results
- Summary

Action Model Example

```
(:action move_or_attack_n
    :parameters (?currcell ?destcell)
    :precondition
    (and
        (northof ?currcell ?destcell)
        (not (wall ?destcell))
        (not (statue ?destcell))
        (not (lava ?destcell))
        (not (plant ?destcell))
        (not (tree ?destcell))
        (not (closeddoor ?destcell))
        (playerat ?currcell)
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)
```

Lifted Linked Clause (LLC)

- LLC definition:
  - A lifted subset of predicates of a state space AND
  - Predicates share at least one variable across their arguments

- More formally, an LLC is a first-order relational conjunct

$$c_1 \wedge c_2 \wedge \ \dots \wedge c_n$$

that refers to one or more existentially quantified variables and is satisfied by some states $S' \subset S$

- An LLC is <u>**active**</u> if it unifies with the current state

Lifted Linked Clause (LLC)

- First, by example: agent on the same tile as a shaft:

Image

Symbolic State

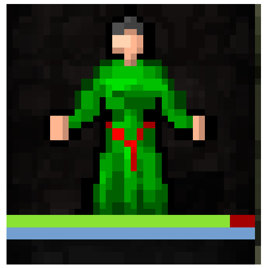
<u>LLC</u>

Tile1

agent-at(5, 7)
shaft-at(5, 7)

{agent-at(?X1, ?Y1),
shaft-at(?X1, ?Y1)}

For reference, the agent on a regular (non-shaft) tile:



Lifted Linked Clause (LLC)

- Another example: agent in front of a closed door:

	Image	Symbolic State
Tile2		closed-door (5, 3)
Tile1		agent-at(5, 2)

north(3, 2)

<u>LLC</u>

{agent-at(**?X1**, ?Y1),
closed-door(**?X1**, ?Y2),
north(?Y2, ?Y1)}

Lifted Linked Clause (LLC)

- Another example: agent east of a monster:

Image

Tile2 Tile1

Symbolic Annotation

monster-at(5, 1)	agent-at(4, 1)

west(5, 4)

<u>LLC</u>

{agent-at(?X1, **?Y1**),
monster-at(?X2, **?Y1**),
west(?X2, ?X1)}

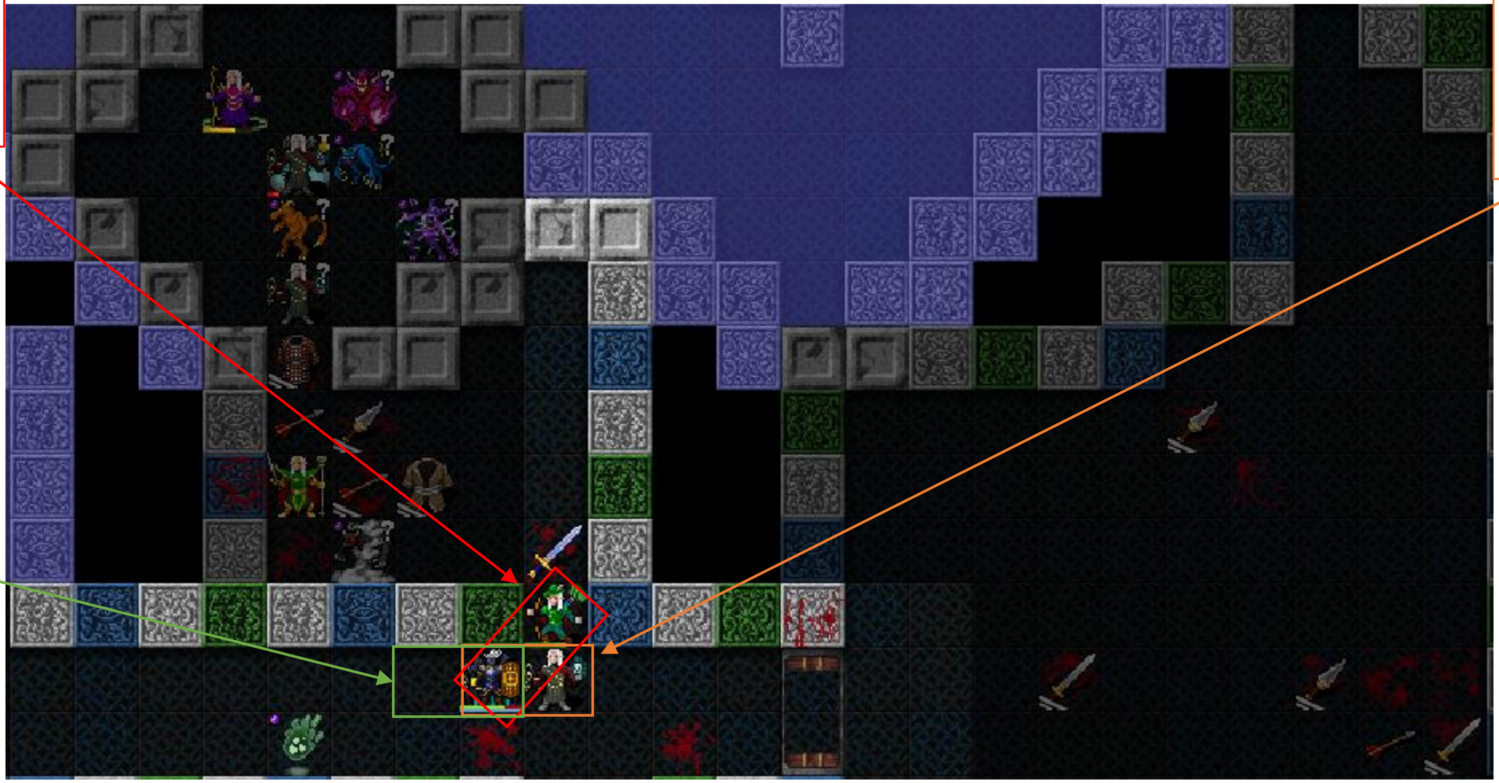
Multiple LLCs are often active in a single state

{agent-at(?X1, ?Y1),
monster-at(?X2, ?Y2),
east(?X2, ?X1)}
north(?Y2, ?Y1)}

Monster northeast of agent

{agent-at(?X1, **?Y1**),
empty(?X2, **?Y1**),
west(?X2, ?X1)}

There is an open tile west of the agent



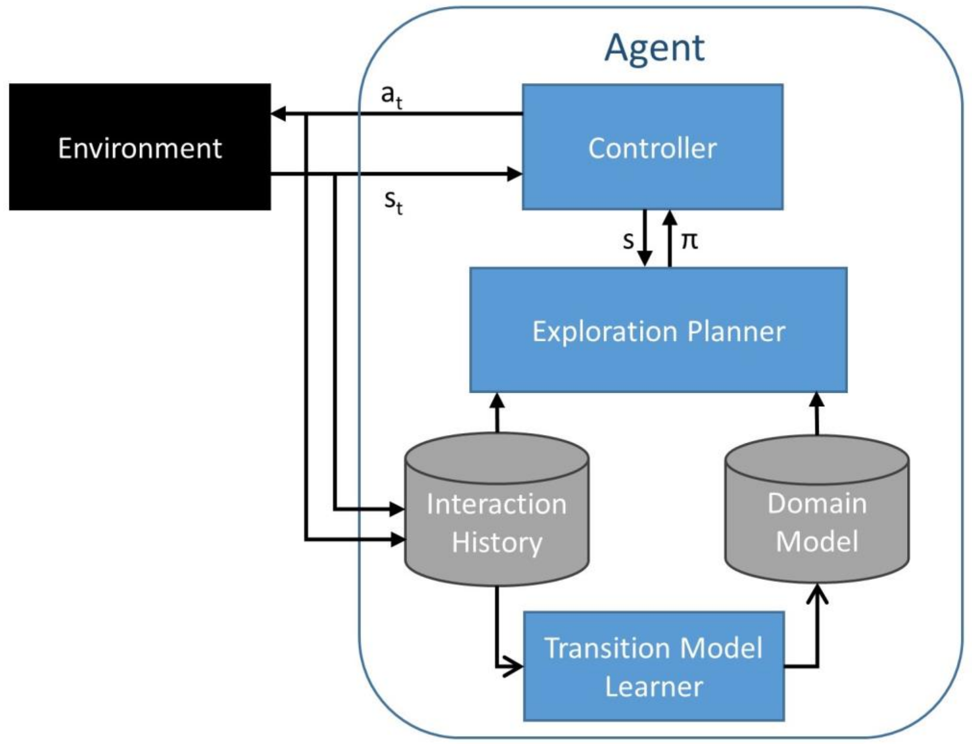
{agent-at(?X1, **?Y1**),
monster-at(?X2, **?Y1**),
east(?X2, ?X1)}

Monster east of agent

Outline

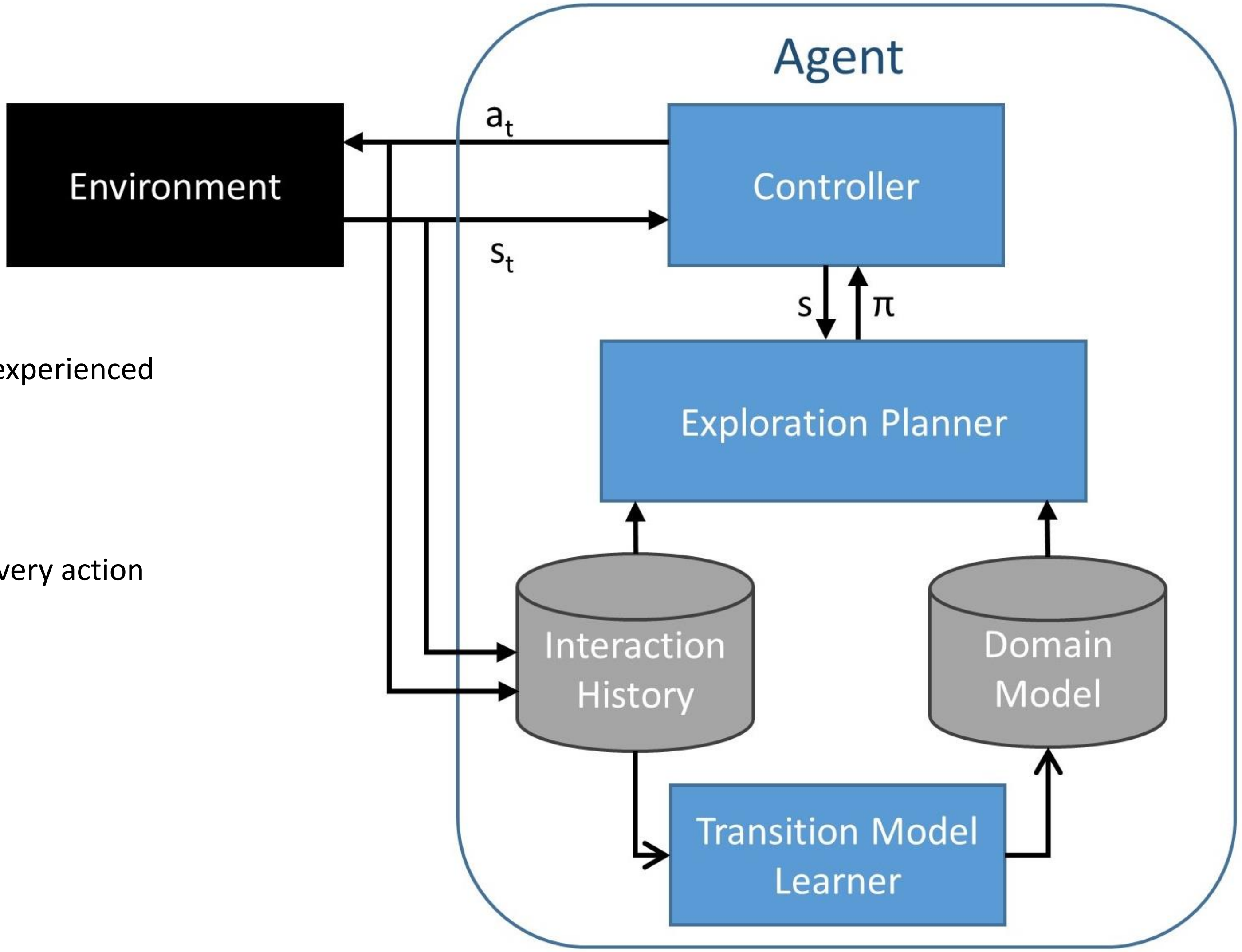
- Research Problem
- Lifted Linked Clause (LLC)
- **Self-Directed Exploration**
- Empirical Results
- Summary

Exploratory Planning Agent Architecture



Agent Architecture

- Interaction History:
  - Collection of every $< s_i, a_i, s_{i+1} >$ transition experienced by agent:
    $(< s_0, a_0, s_1 >, < s_1, a_1, s_2 >, \dots)$
- Domain Model:
  - Current preconditions and effects model for every action
- Transition Model Learner:
  - Off-the-shelf tool: ILASP
    - "Inductive Learning of Answer Set Programs"

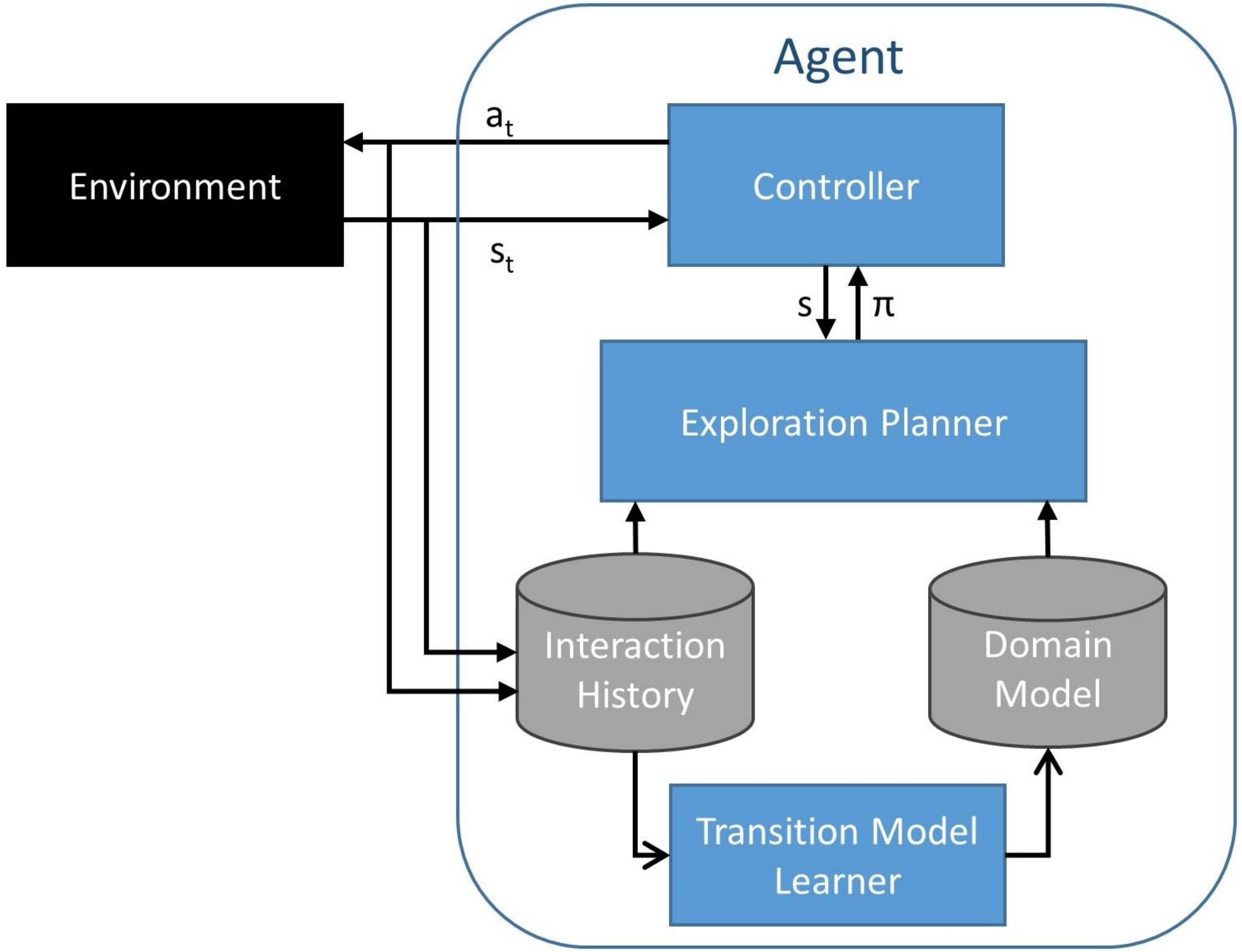


Transition Model Learner

- We used Inductive Learning of Answer Set Programs (ILASP)
  - Law, M., Russo, A., & Broda, K. (2015)
- To learn preconditions, we apply ILASP
  - Positive examples are transitions where $s_i \neq s_{i+1}$
  - Negative examples are transitions where $s_i = s_{i+1}$
- To learn effects, we take every transition $< s_i, a_i, s_{i+1} >$ for an action a_i and if $s_i \ \neq s_{i+1}$ we perform a state diff between s_i and s_{i+1}

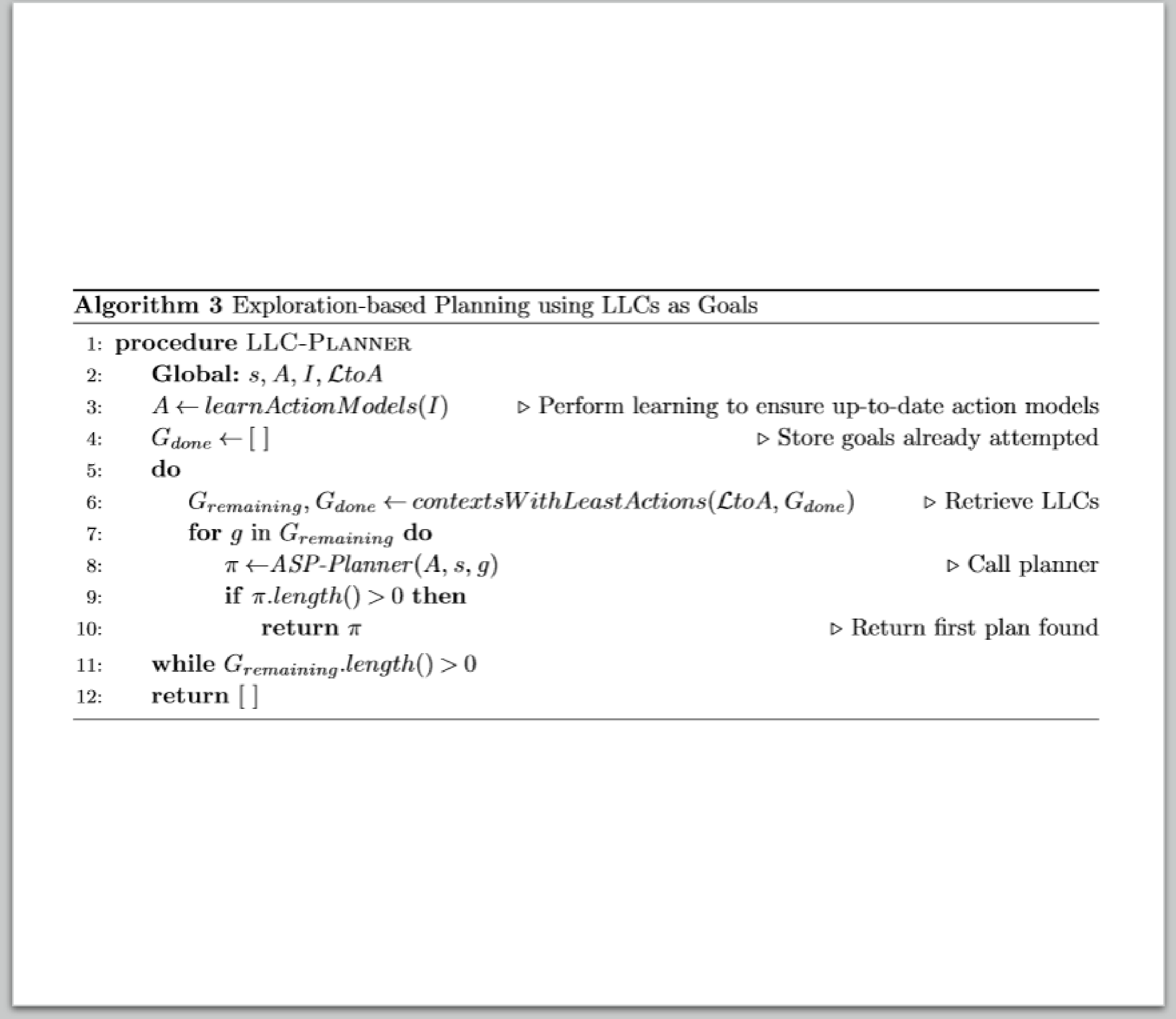
Agent Architecture

- Controller:
    - If in a new situation (new LLC is active) then try actions that have not been tried in this LLC before
    - Otherwise call exploration planner
    - If no plan, take random action
- Exploration planner:
    - Purpose: Reach new situations (where new LLCs are active)
    - When a plan is executed, either it succeeds or fails:
    - If Failure: Great, we have collected a negative transition for one of our actions, which will help update model
    - If Success: Great, we have reached a new situation and can test new actions, helping to update model



Exploration Planner

- Track which actions have been taken in which LLCs
- Exploration planner chooses an LLC to be a goal among LLCs least acted in
- Planning happens using an Answer Set Programming planner that takes the LLC directly as a goal (no grounding is necessary)

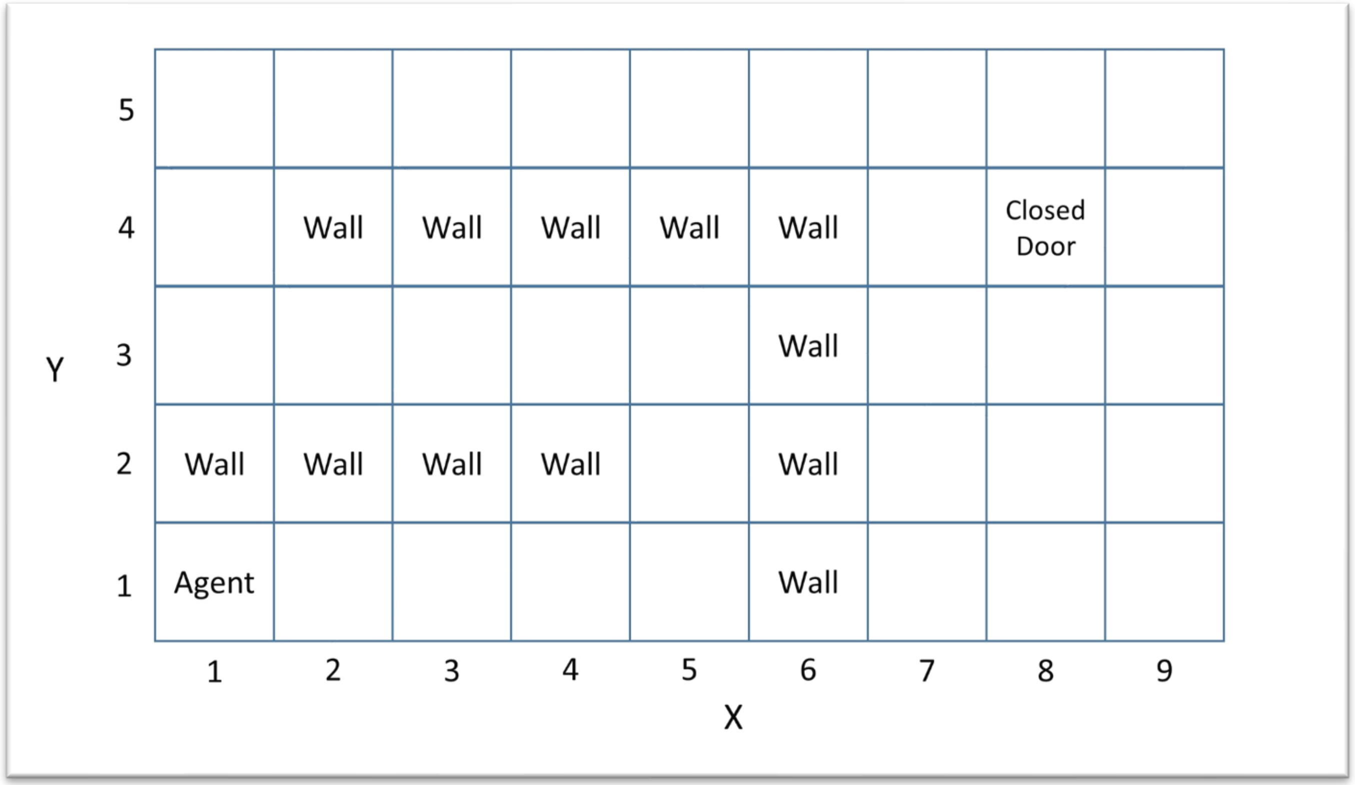


Algorithm 3 Exploration-based Planning using LLCs as Goals

```
1:  procedure LLC-PLANNER
2:      Global: $s, A, I, \mathcal{L}toA$
3:      $A \leftarrow learnActionModels(I)$        ▷ Perform learning to ensure up-to-date action models
4:      $G_{done} \leftarrow [\,]$                                          ▷ Store goals already attempted
5:      do
6:          $G_{remaining}, G_{done} \leftarrow contextsWithLeastActions(\mathcal{L}toA, G_{done})$    ▷ Retrieve LLCs
7:          for $g$ in $G_{remaining}$ do
8:              $\pi \leftarrow ASP\text{-}Planner(A, s, g)$                    ▷ Call planner
9:              if $\pi.length() > 0$ then
10:                 return $\pi$                                    ▷ Return first plan found
11:     while $G_{remaining}.length() > 0$
12:     return [ ]
```

Walkthrough Example

State 0
Domain model: Empty
Agent Location: 1,1



Y									
5									
4		Wall	Wall	Wall	Wall	Wall		Closed Door	
3						Wall			
2	Wall	Wall	Wall	Wall		Wall			
1	Agent					Wall			
	1	2	3	4	5	6	7	8	9
					X				

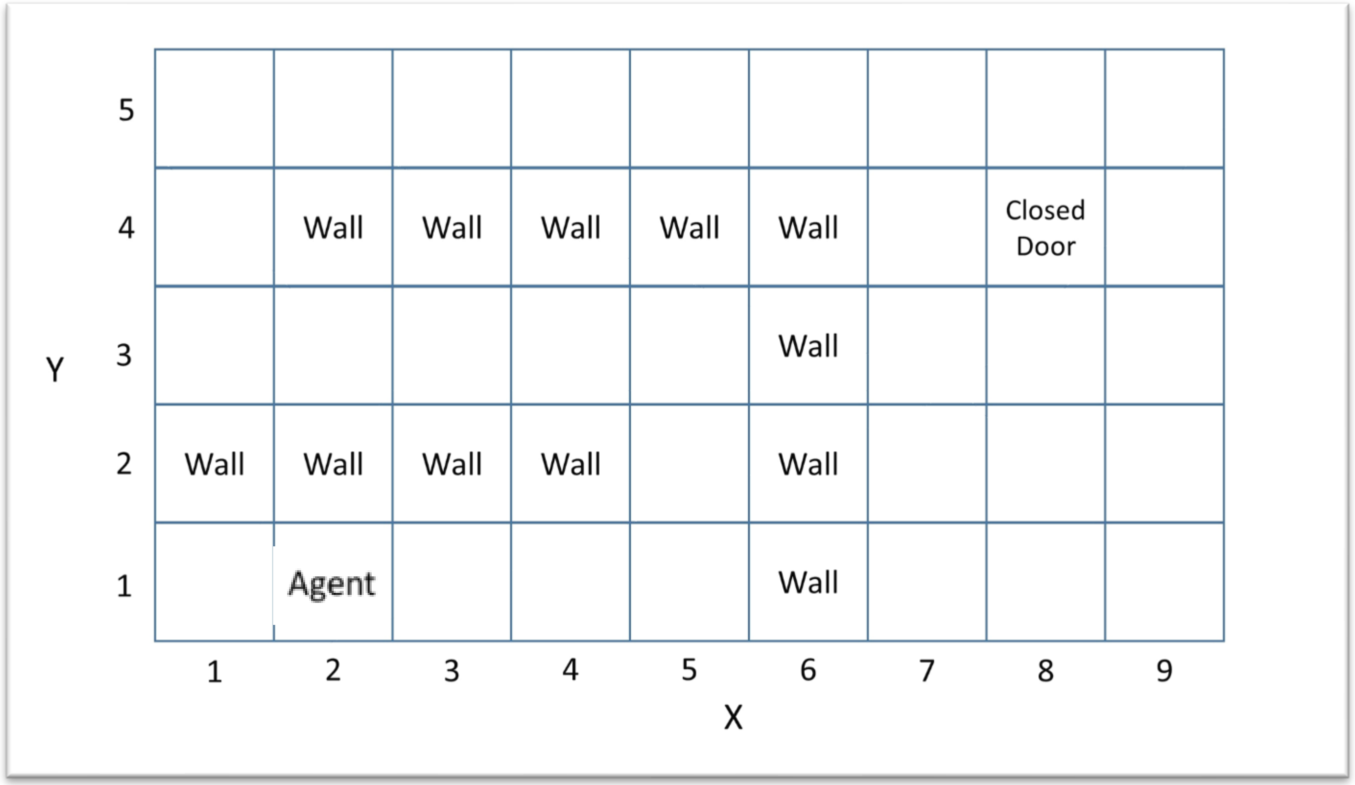
Walkthrough Example

State 15: Agent has successfully moved east after trying a number of failed random actions

Domain model:

```
  move-east(?X1, ?Y1):
      pre: (and
                (agent-at ?X1, ?Y1)
                (east ?X2 ?X1))
```

Agent Location: 2,1



Walkthrough Example

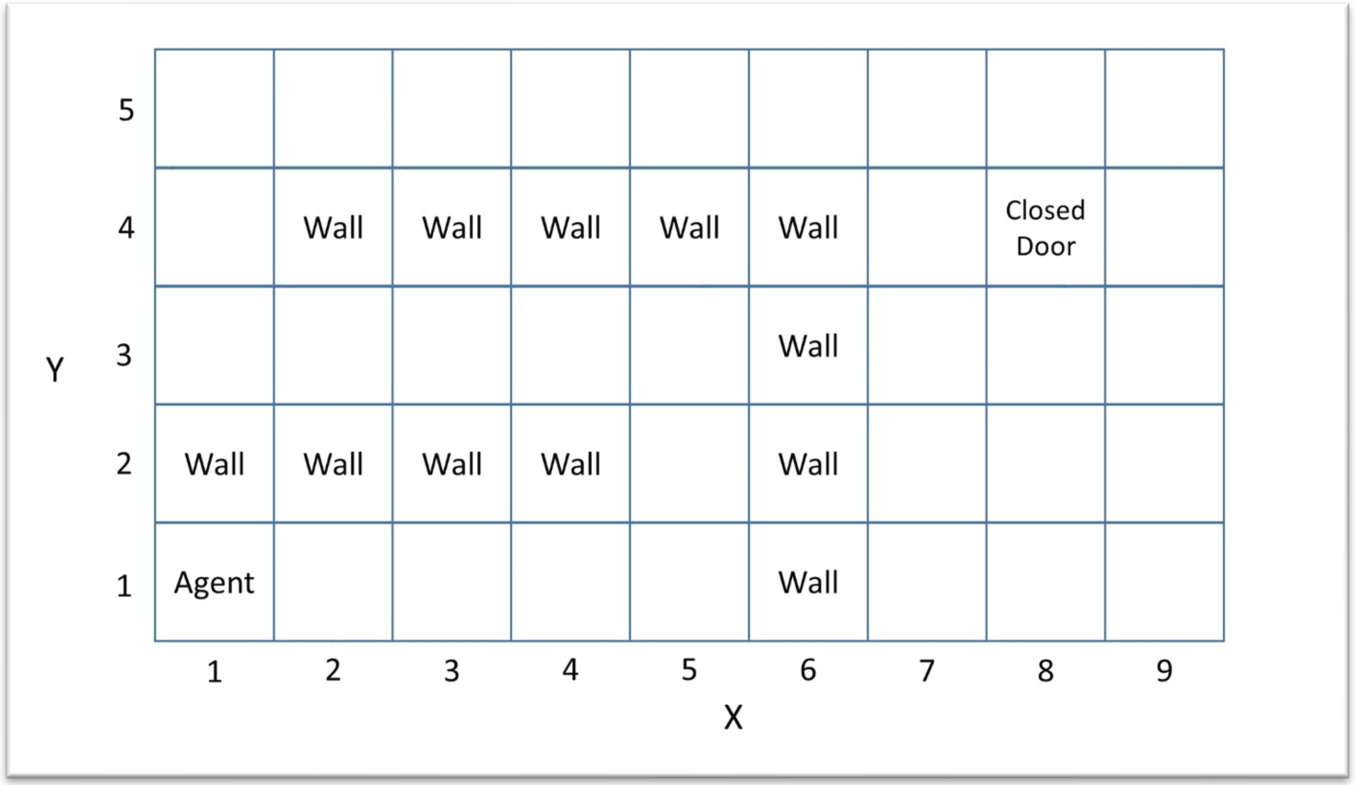
State 41: Agent has moved west

Domain model:

```
  move-east(?X1, ?Y1):
      pre: (and
              (agent-at ?X1, ?Y1)
              (east ?X2 ?X1))

move-west(?X1, ?Y1):
      pre: (and
              (agent-at ?X1, ?Y1)
              (west ?X2 ?X1))
```

Agent Location: 1,1



Walkthrough Example

At this point the agent has taken all actions in the current state (all active LLCs) – this triggers exploratory planning

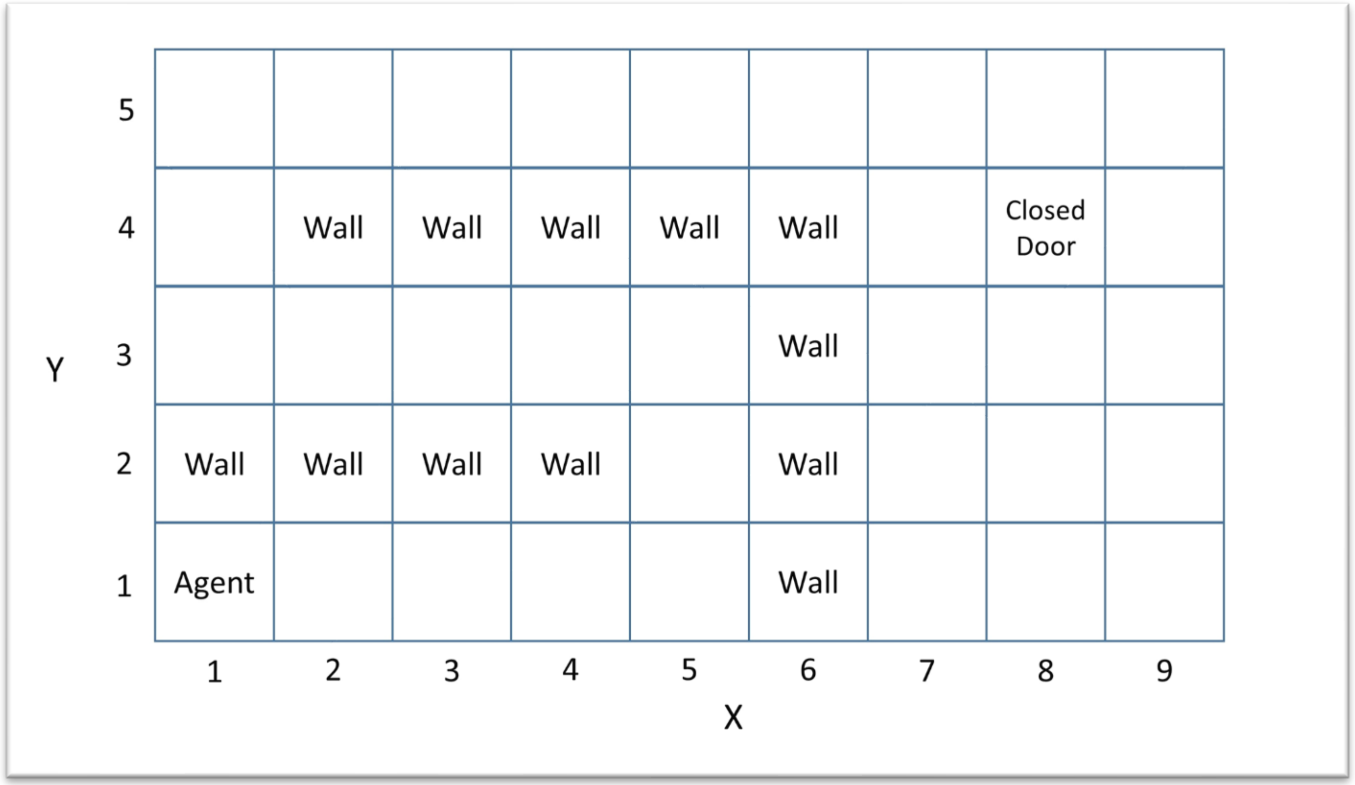
State 41: Agent has moved west

Domain model:

```
  move-east(?X1, ?Y1):
      pre: (and
               (agent-at ?X1, ?Y1)
               (east ?X2 ?X1))

move-west(?X1, ?Y1):
      pre: (and
               (agent-at ?X1, ?Y1)
               (west ?X2 ?X1))
```

Agent Location: 1,1



Walkthrough Example

State 41: Agent has moved west

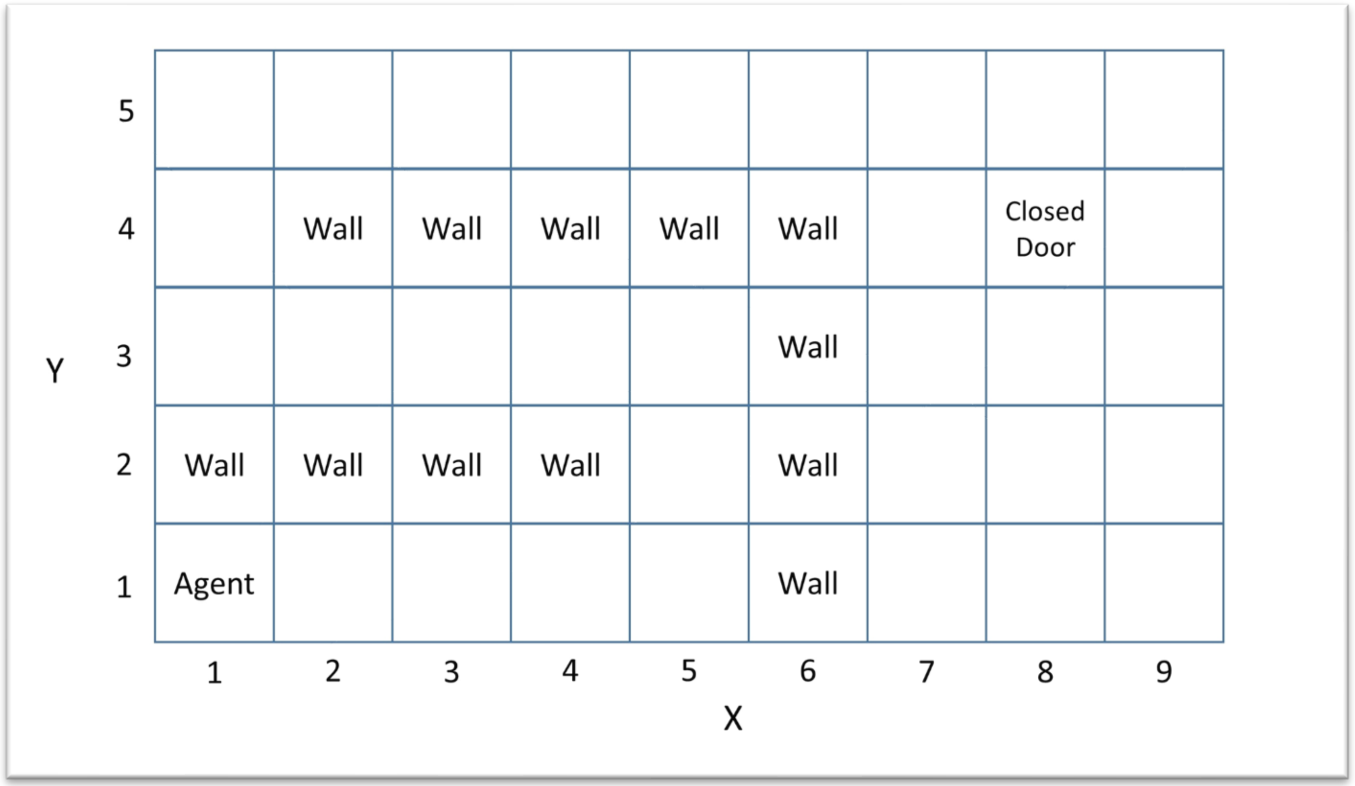
Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

Domain model:

```
 move-east(?X1, ?Y1):
    pre: (and
              (agent-at ?X1, ?Y1)
              (east ?X2 ?X1))

move-west(?X1, ?Y1):
    pre: (and
              (agent-at ?X1, ?Y1)
              (west ?X2 ?X1))
```

Agent Location: 1,1



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

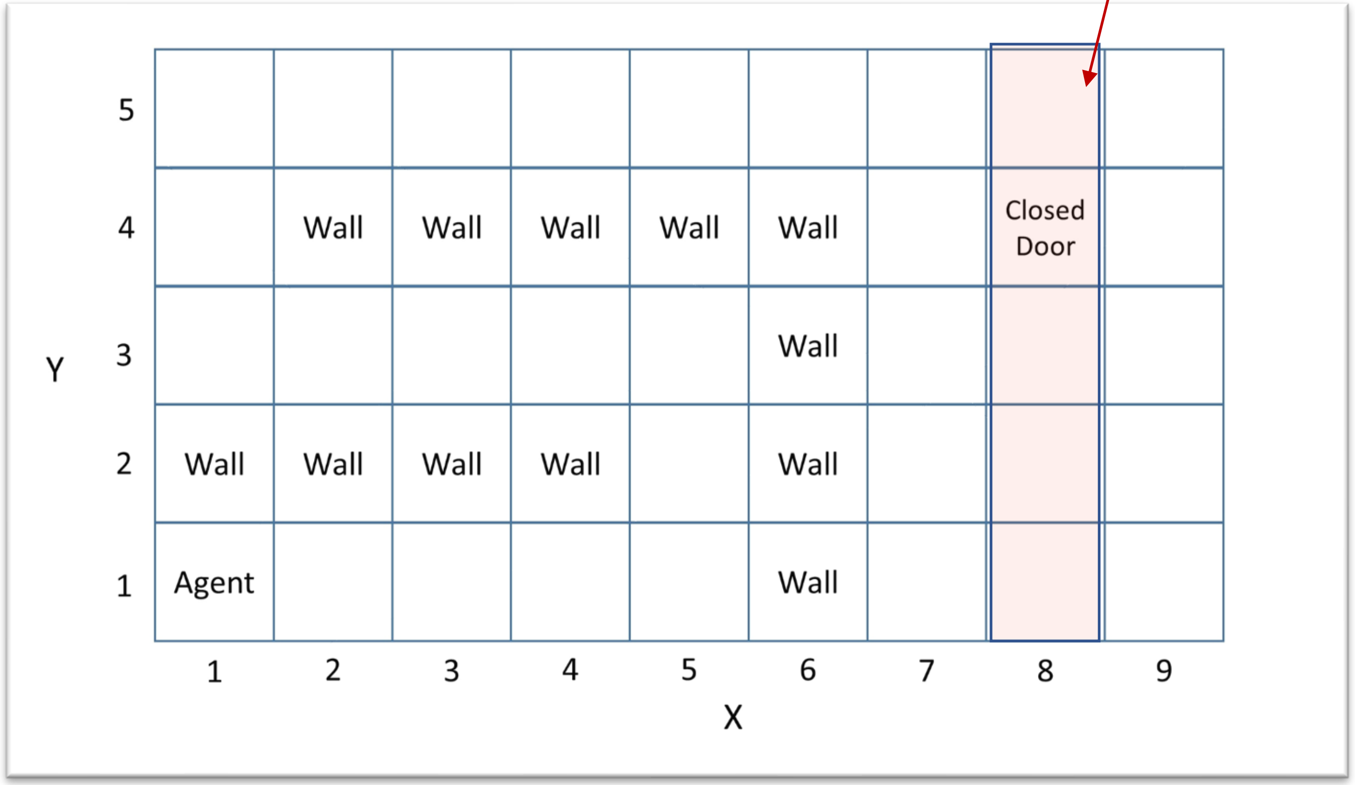
State 41: Agent has moved west

Domain model:

```
 move-east(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (east ?X2 ?X1))

move-west(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (west ?X2 ?X1))
```

Agent Location: 1,1



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

State 41: Agent has moved west

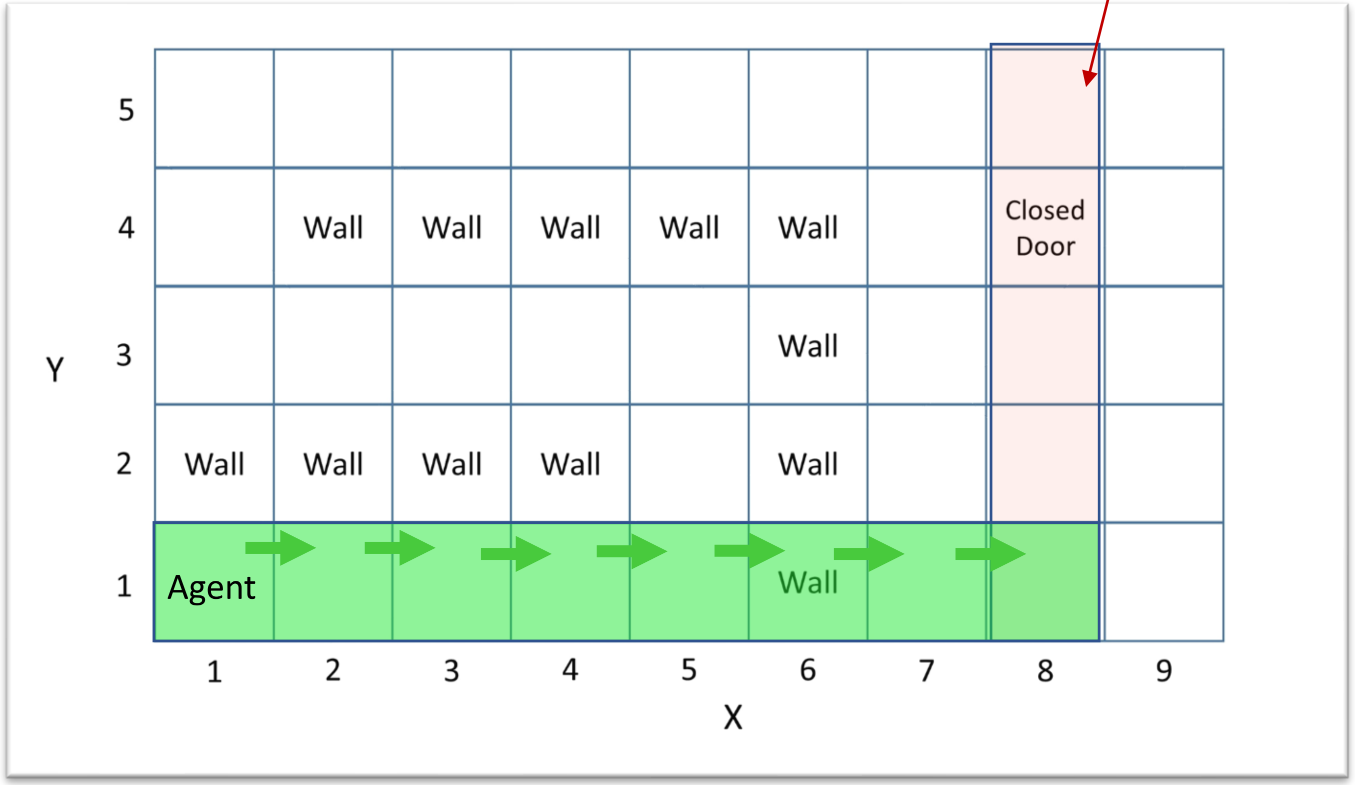
Domain model:

```
move-east(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (east ?X2 ?X1))

move-west(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (west ?X2 ?X1))
```

Agent Location: 1,1

Exploratory Plan is:
1. move-east
2. move-east
3. move-east
4. move-east
5. move-east
6. move-east
7. move-east



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

State 42: Agent has moved east

Domain model:

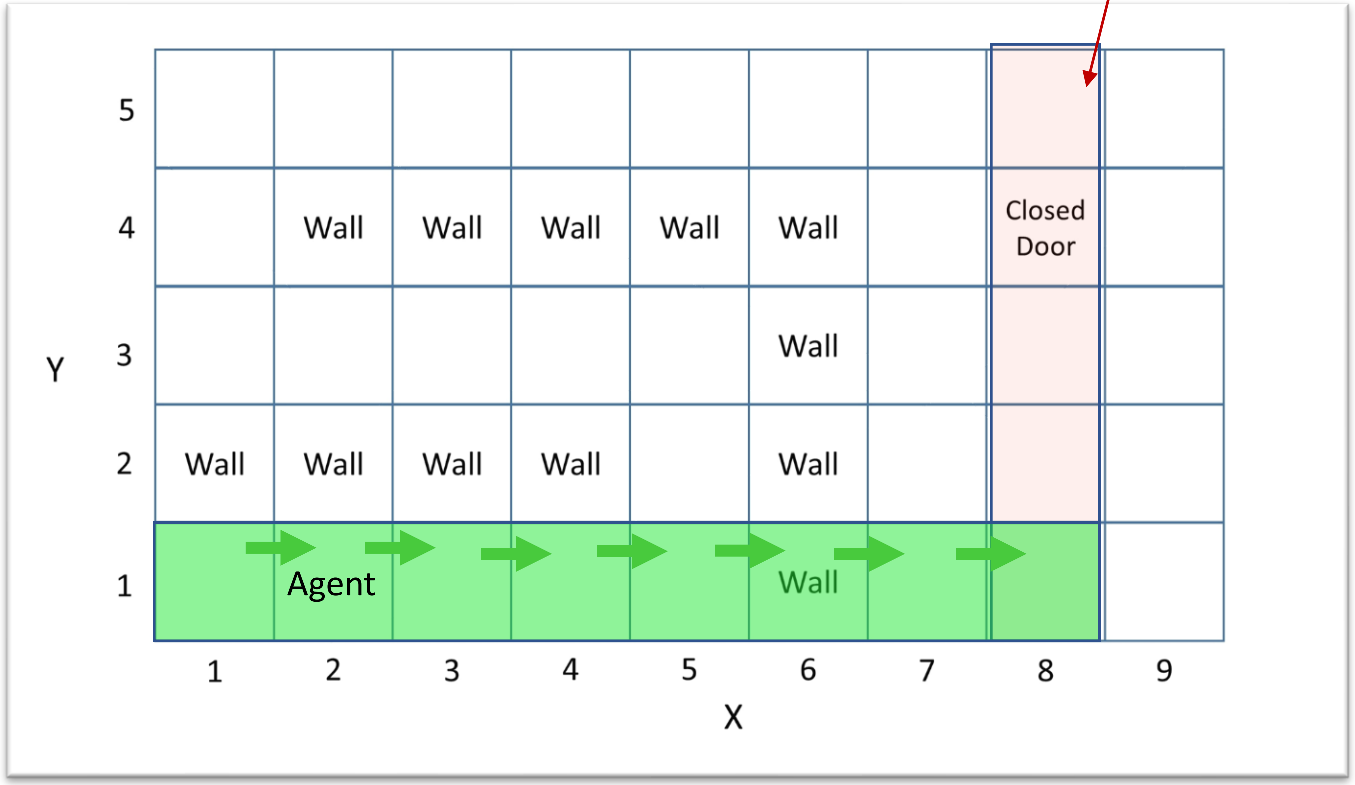
move-east(?X1, ?Y1):
    pre: (and
        (agent-at ?X1, ?Y1)
        (east ?X2 ?X1))

move-west(?X1, ?Y1):
    pre: (and
        (agent-at ?X1, ?Y1)
        (west ?X2 ?X1))

Agent Location: 2,1

Exploratory Plan is:

1. ~~move-east~~
2. move-east
3. move-east
4. move-east
5. move-east
6. move-east
7. move-east



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

State 43: Agent has moved east

Domain model:

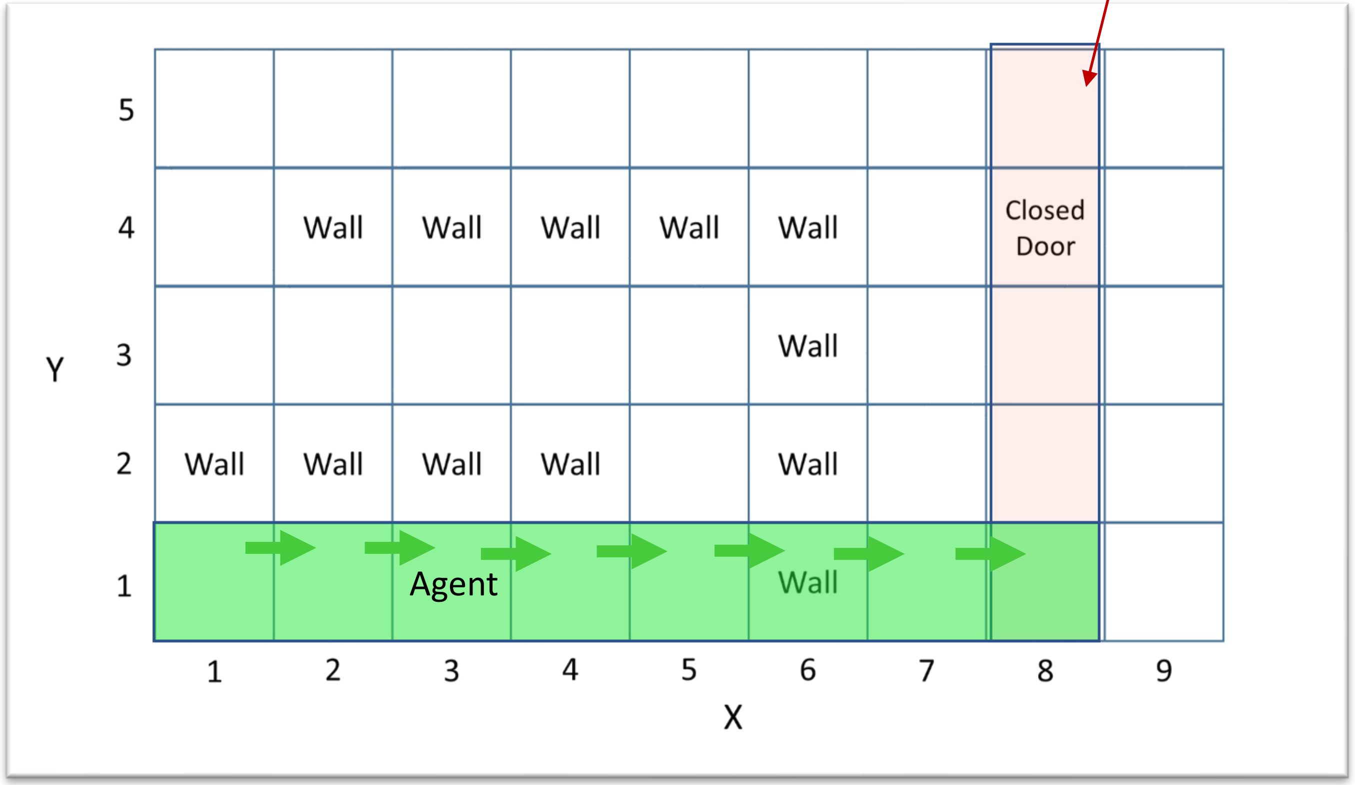
move-east(?X1, ?Y1):
pre: (and
(agent-at ?X1, ?Y1)
(east ?X2 ?X1))

move-west(?X1, ?Y1):
pre: (and
(agent-at ?X1, ?Y1)
(west ?X2 ?X1))

Agent Location: 3,1

Exploratory Plan is:

1. ~~move-east~~
2. ~~move-east~~
3. move-east
4. move-east
5. move-east
6. move-east
7. move-east



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

State 44: Agent has moved east

Domain model:

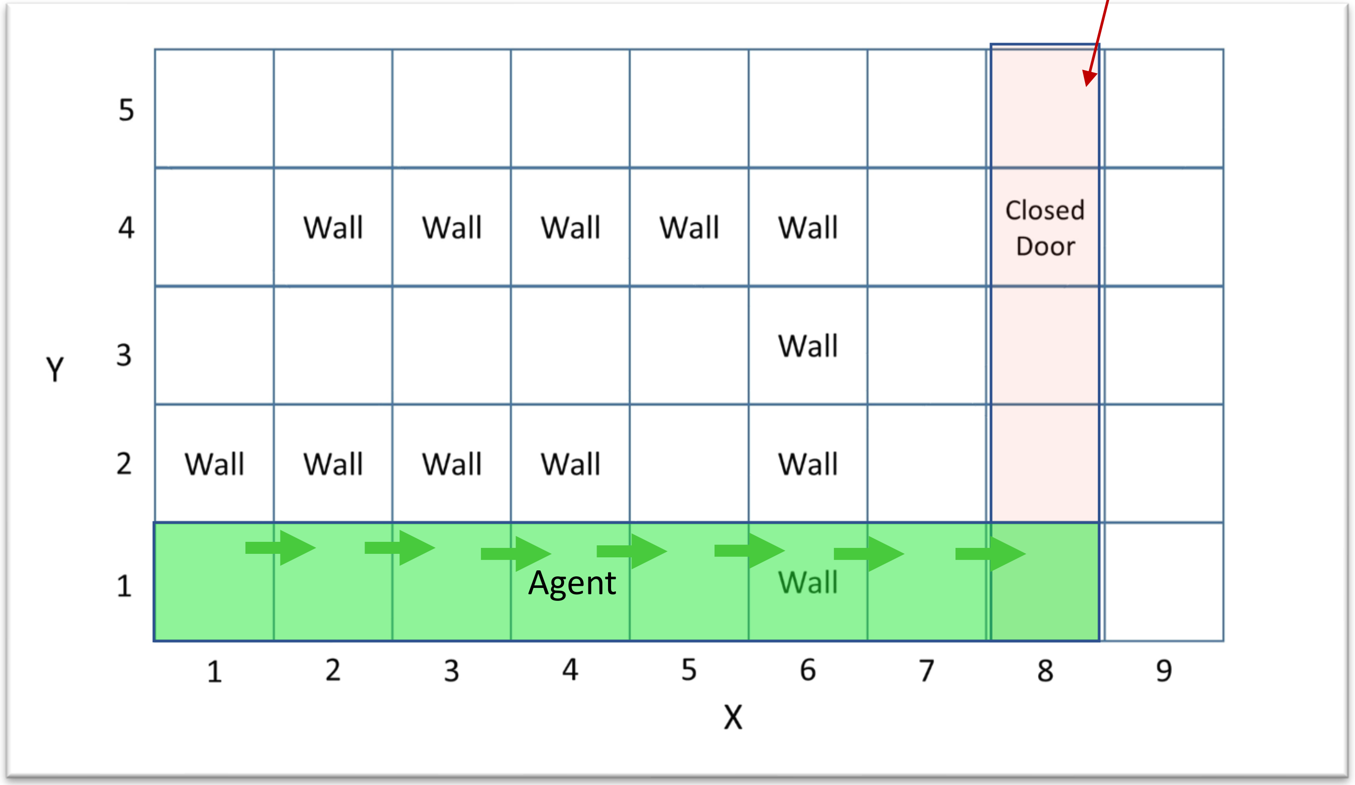
```
move-east(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (east ?X2 ?X1))

move-west(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (west ?X2 ?X1))
```

Agent Location: 4,1

Exploratory Plan is:

~~1. move-east~~
~~2. move-east~~
~~3. move-east~~
4. move-east
5. move-east
6. move-east
7. move-east



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

State 45: Agent has moved east

Domain model:

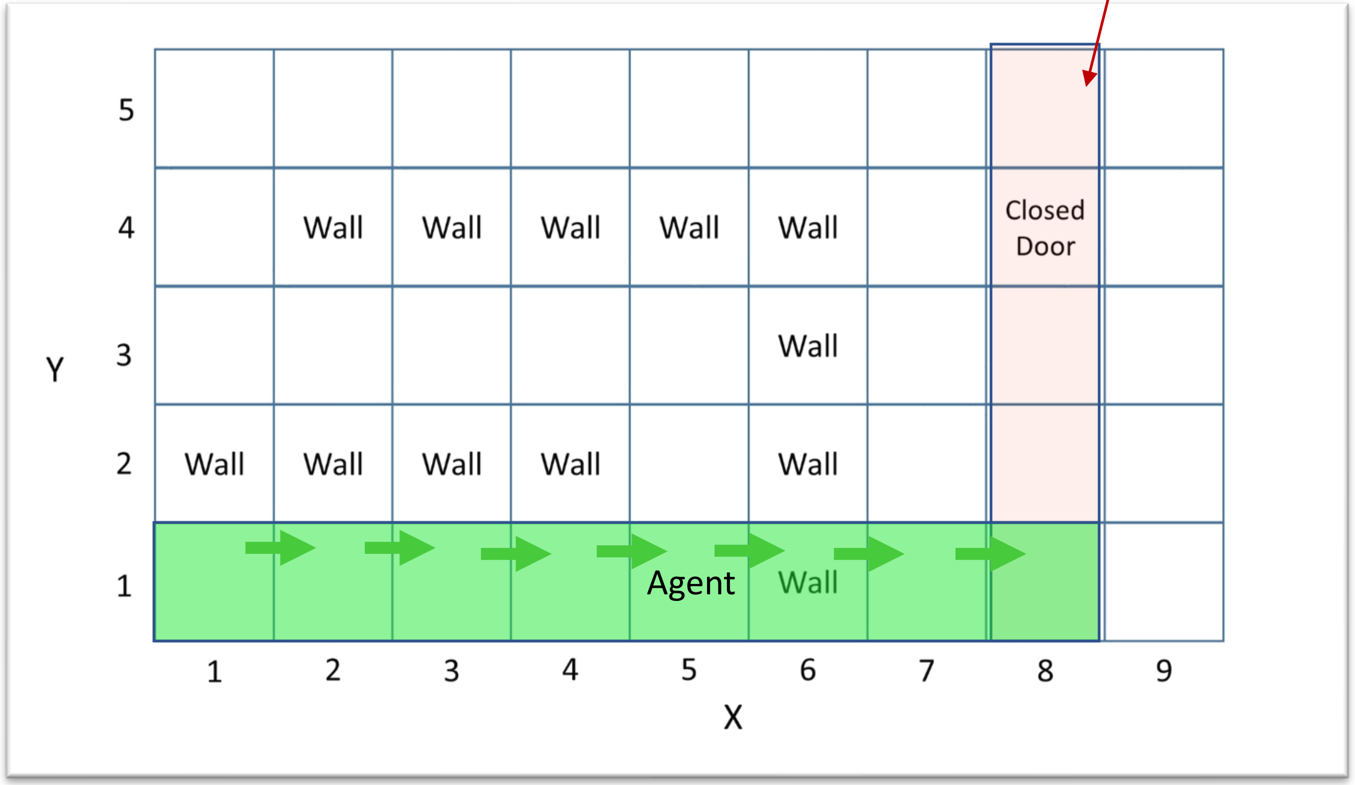
```
move-east(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (east ?X2 ?X1))

move-west(?X1, ?Y1):
    pre: (and
            (agent-at ?X1, ?Y1)
            (west ?X2 ?X1))
```

Agent Location: 5,1

Exploratory Plan is:

1. ~~move-east~~
2. ~~move-east~~
3. ~~move-east~~
4. ~~move-east~~
5. move-east
6. move-east
7. move-east



Walkthrough Example

This LLC represents the context where an agent is in a tile that shares a X-value with a closed-door cell.

Exploratory planning chooses the following goal:
{(agent-at **?X1**, ?Y1), (closed-door **?X1**, ?Y2)}

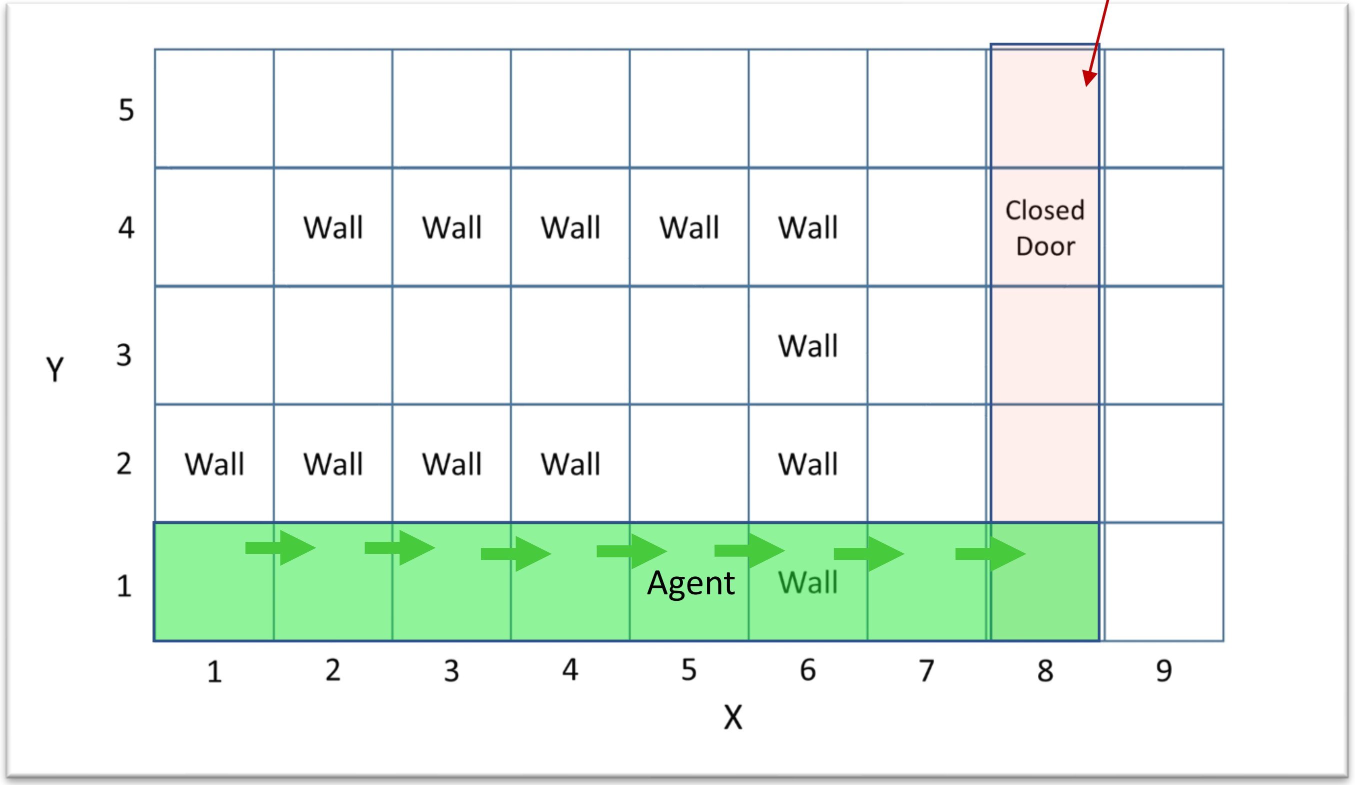
State 46: Agent failed to move east

Domain model:
move-east(?X1, ?Y1):
pre: (and
(agent-at ?X1, ?Y1)
(east ?X2 ?X1)
(not (wall ?X2 ?Y1)))

move-west(?X1, ?Y1):
pre: (and
(agent-at ?X1, ?Y1)
(west ?X2 ?X1))

Agent Location: 5,1

Exploratory Plan is:
1. ~~move-east~~
2. ~~move-east~~
3. ~~move-east~~
4. ~~move-east~~
5. **move-east (FAIL)**
6. move-east
7. move-east



Outline

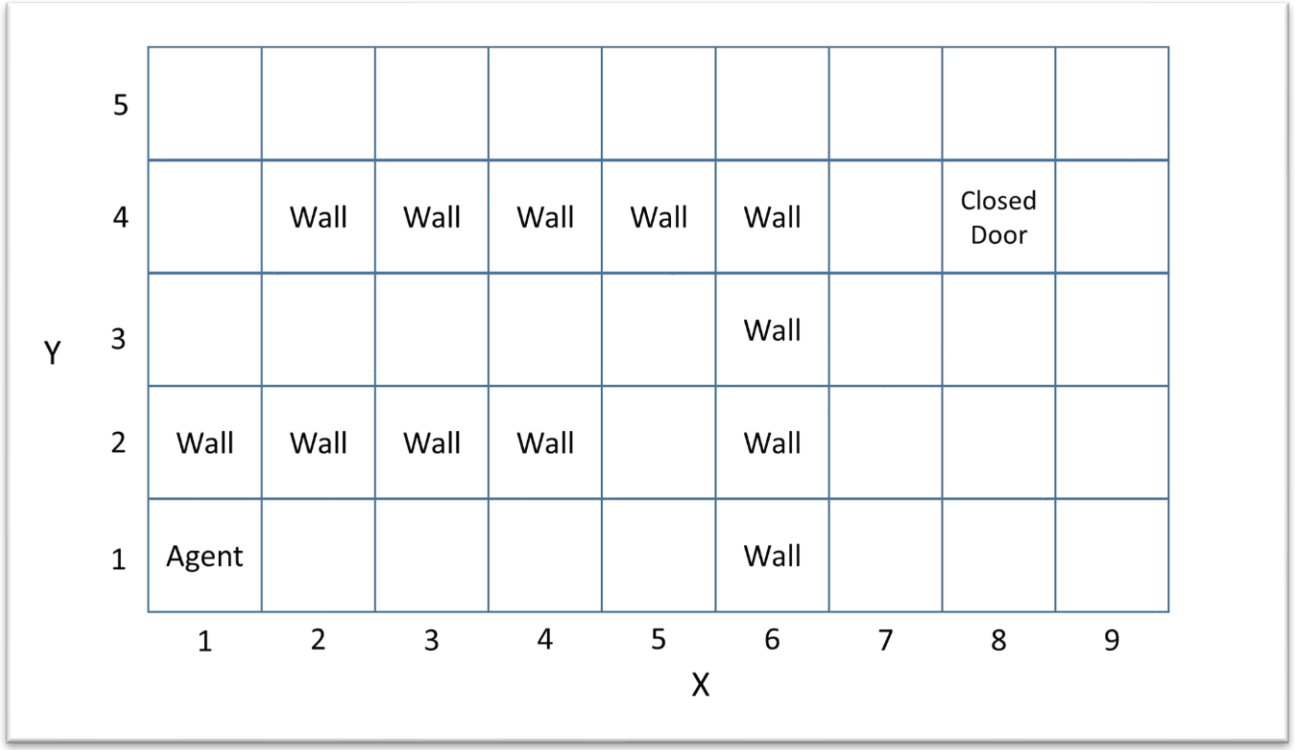
- Research Problem
- Lifted Linked Clause (LLC)
- Self-Directed Exploration
- **Empirical Results**
- Summary

Experimental Setup

- Domain model with no preconditions and effects for actions
  - Predicates and action signatures are given
  - 24 actions:
    - 8 cardinal directions
    - 8 actions to open door in each cardinal direction
    - 8 actions to close door in each cardinal direction
- Static and fully observable scenarios
- Pre-processing step: generate all LLCs up to a certain size (n=2) for each agent
- Three agents:
  - Random
  - Explore Local: Choose action taken the least among all currently active LLCs
    - No planning, only choosing next action
  - Planning:
    - Same as explore local, except when current state is fully explored -> attempt planning to reach state with an active LLC for which no actions have been tried

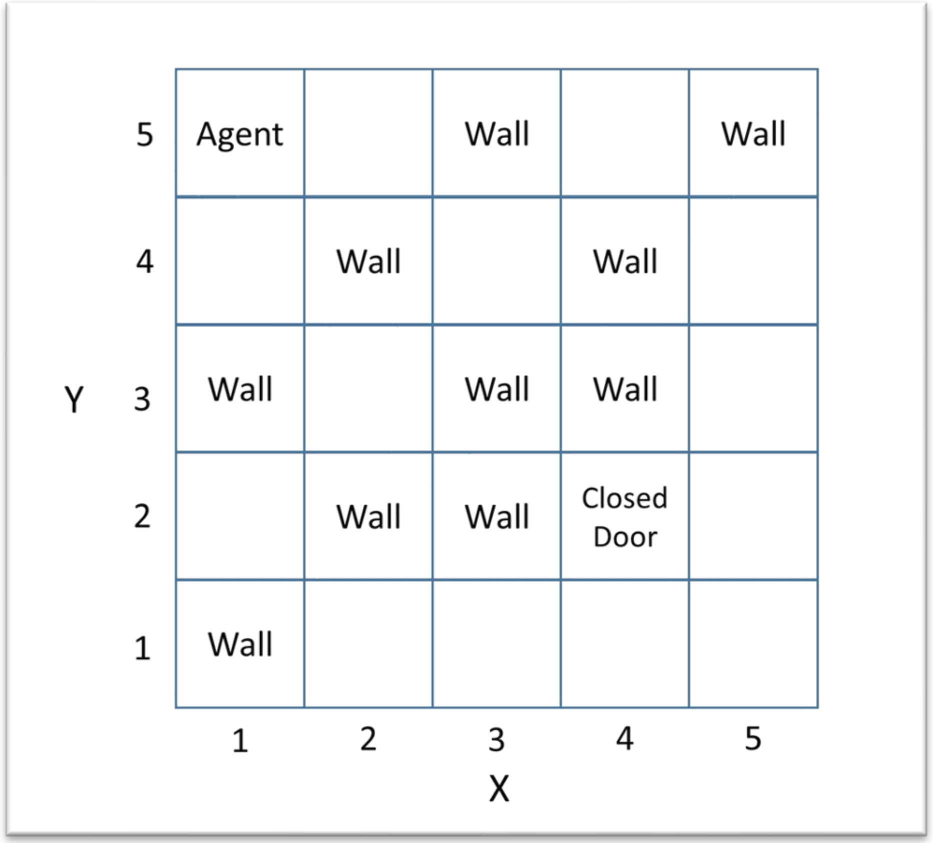
Experimental Setup

- Two scenarios *inspired* by DCSS



Y									
5									
4		Wall	Wall	Wall	Wall	Wall		Closed Door	
3						Wall			
2	Wall	Wall	Wall	Wall		Wall			
1	Agent					Wall			
X	1	2	3	4	5	6	7	8	9

Scenario 1



Y					
5	Agent		Wall		Wall
4		Wall		Wall	
3	Wall		Wall	Wall	
2		Wall	Wall	Closed Door	
1	Wall				
X	1	2	3	4	5

Scenario 2

Hypotheses

[H1] All agents will be able to learn some action preconditions from collected interactions.

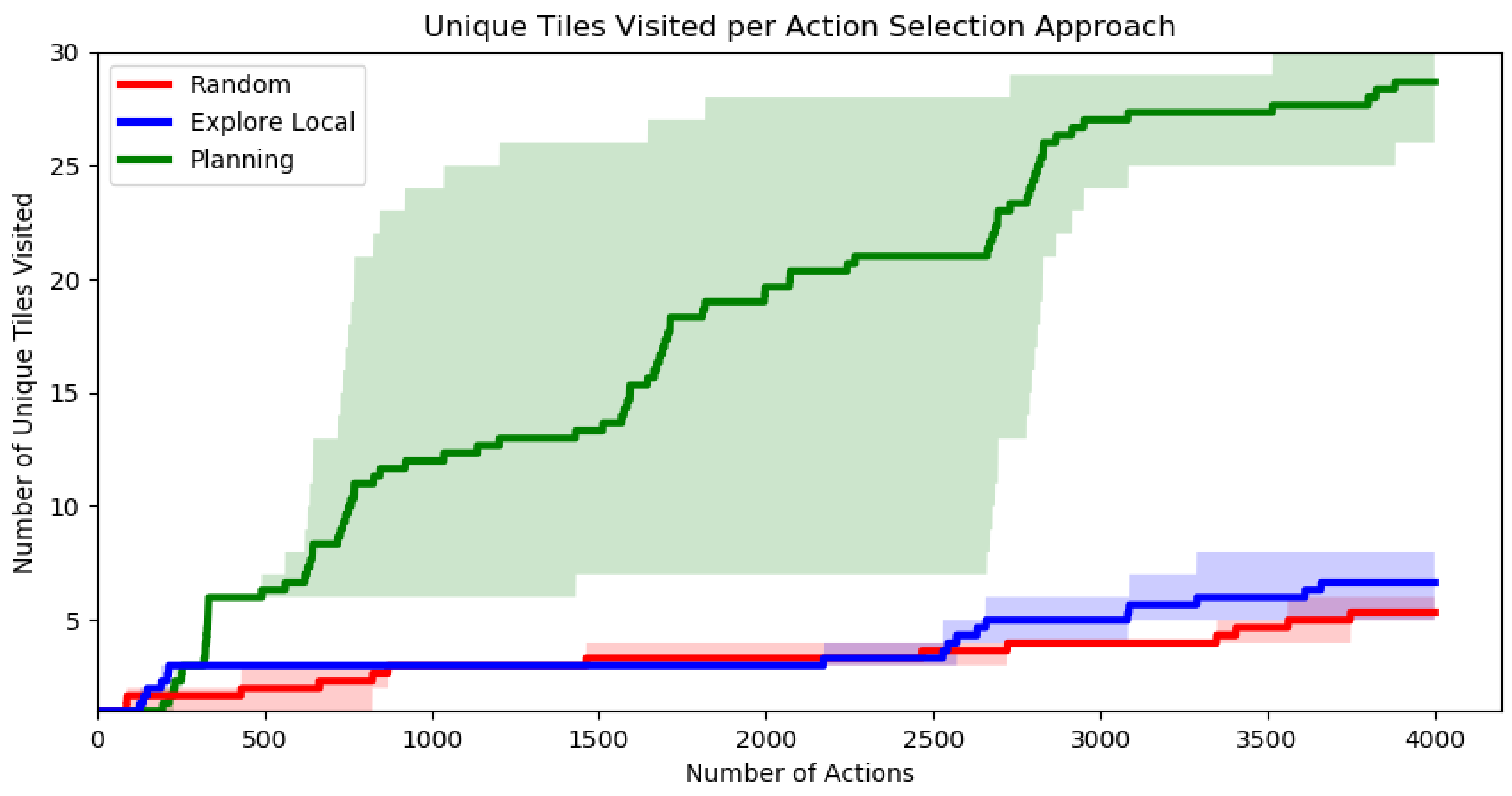
[H2] The Explore Local Agent will learn better preconditions for more actions than a baseline agent taking only random actions.

[H3] The Explore Local Agent will explore more of its environment than the random baseline agent.

[H4] The Planning Agent will learn better preconditions for more actions than both Explore Local Agent and the random baseline agent.

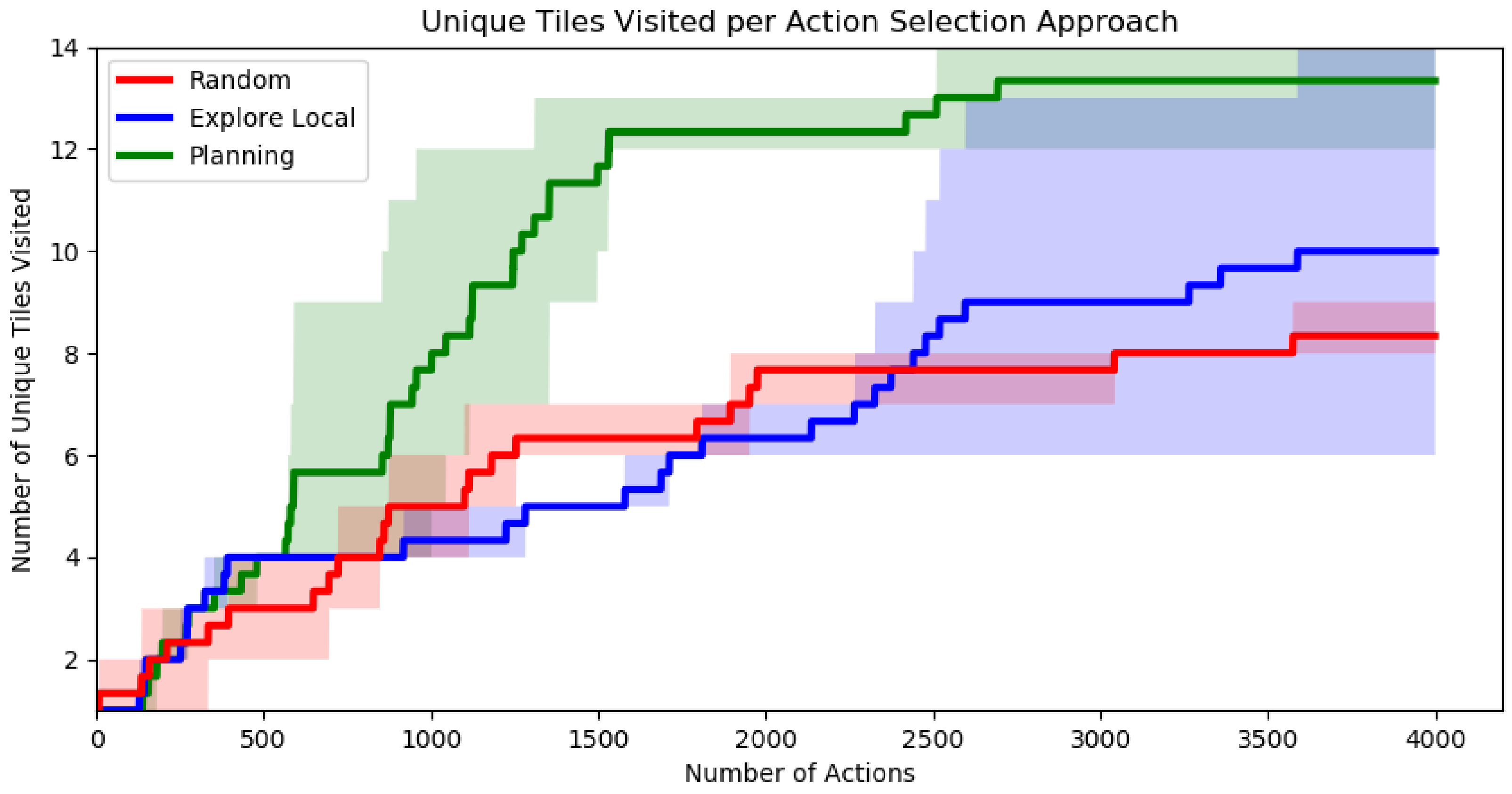
[H5] The Planning Agent will explore more of its environment than both the Explore Local Agent and the random baseline agent.

Results: Exploration of Scenario 1



Average performance over three trials

Results: Exploration of Scenario 2



Average performance over three trials

F1 Scores per Each Action Model

- Special evaluation required
  - Cannot just compare action models learned vs. static hand-coded "correct" models – what if agent learns a valid action model that's different?
  - Example:

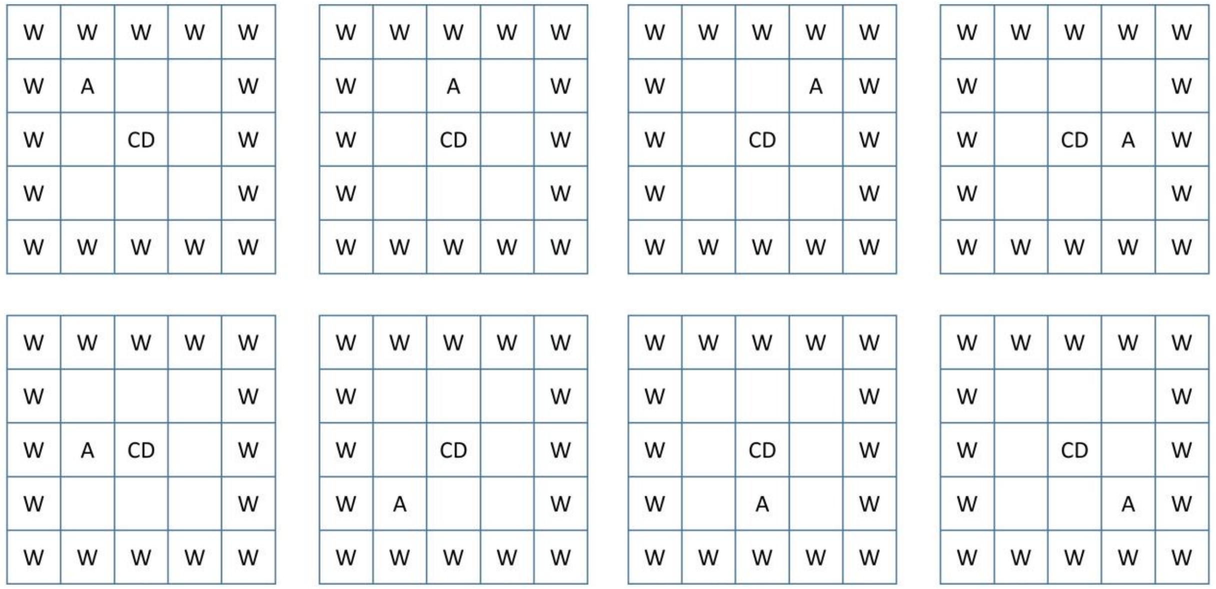
```
move-east(?X1, ?X2):
    pre: (and
            (agent-at ?X1, ?Y1)
            (east ?X2, ?X1)
            (not (wall ?X2, ?Y1)))
```

=

```
move-east(?X1, ?X2):
    pre: (and
            (agent-at ?X1, ?Y1)
            (west ?X1, ?X2)
            (not (wall ?X2, ?Y1)))
```

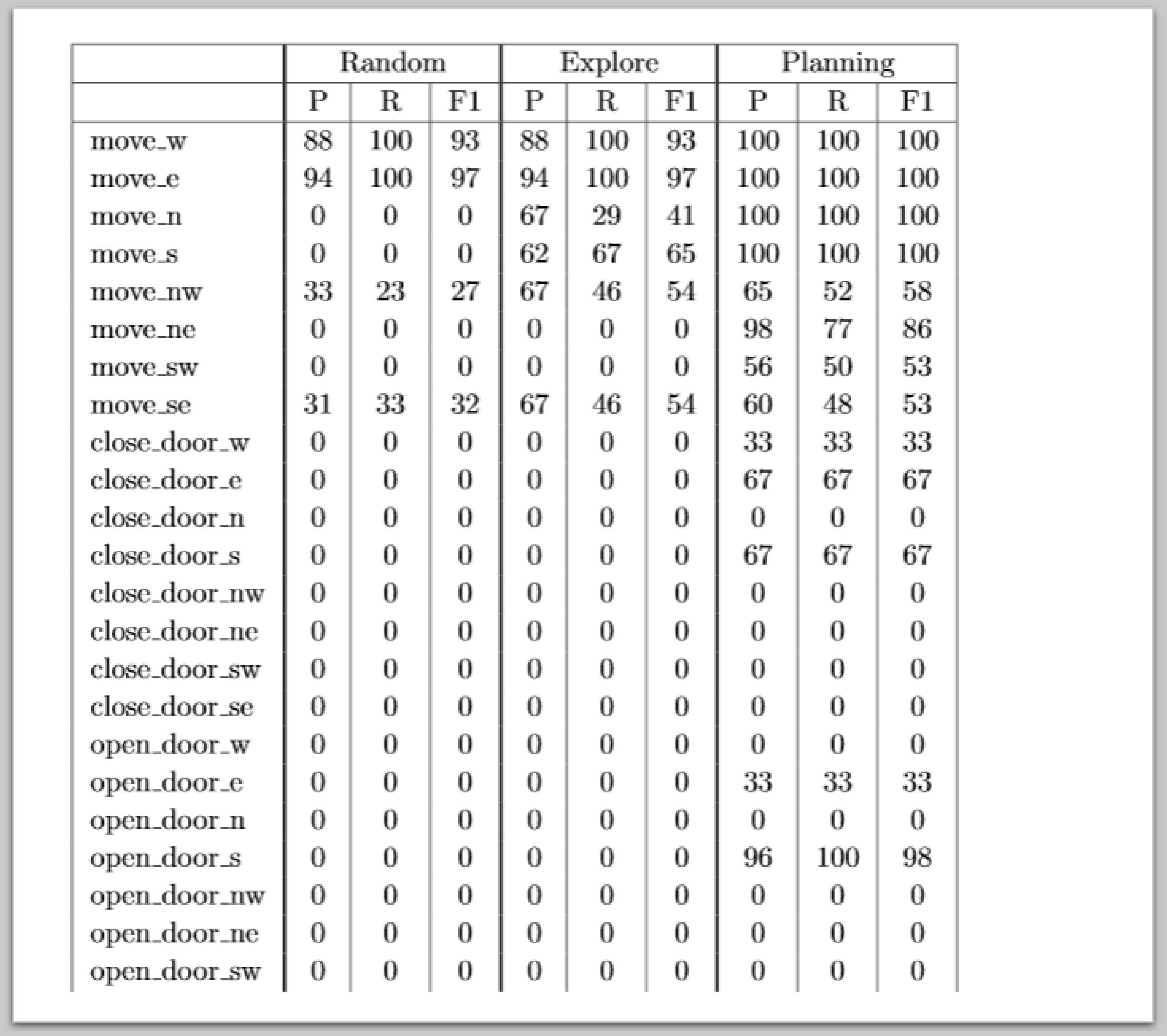
Careful Evaluation of F1 Scores

- Need to create test scenarios, compare resulting state from learned action against ground-truth action



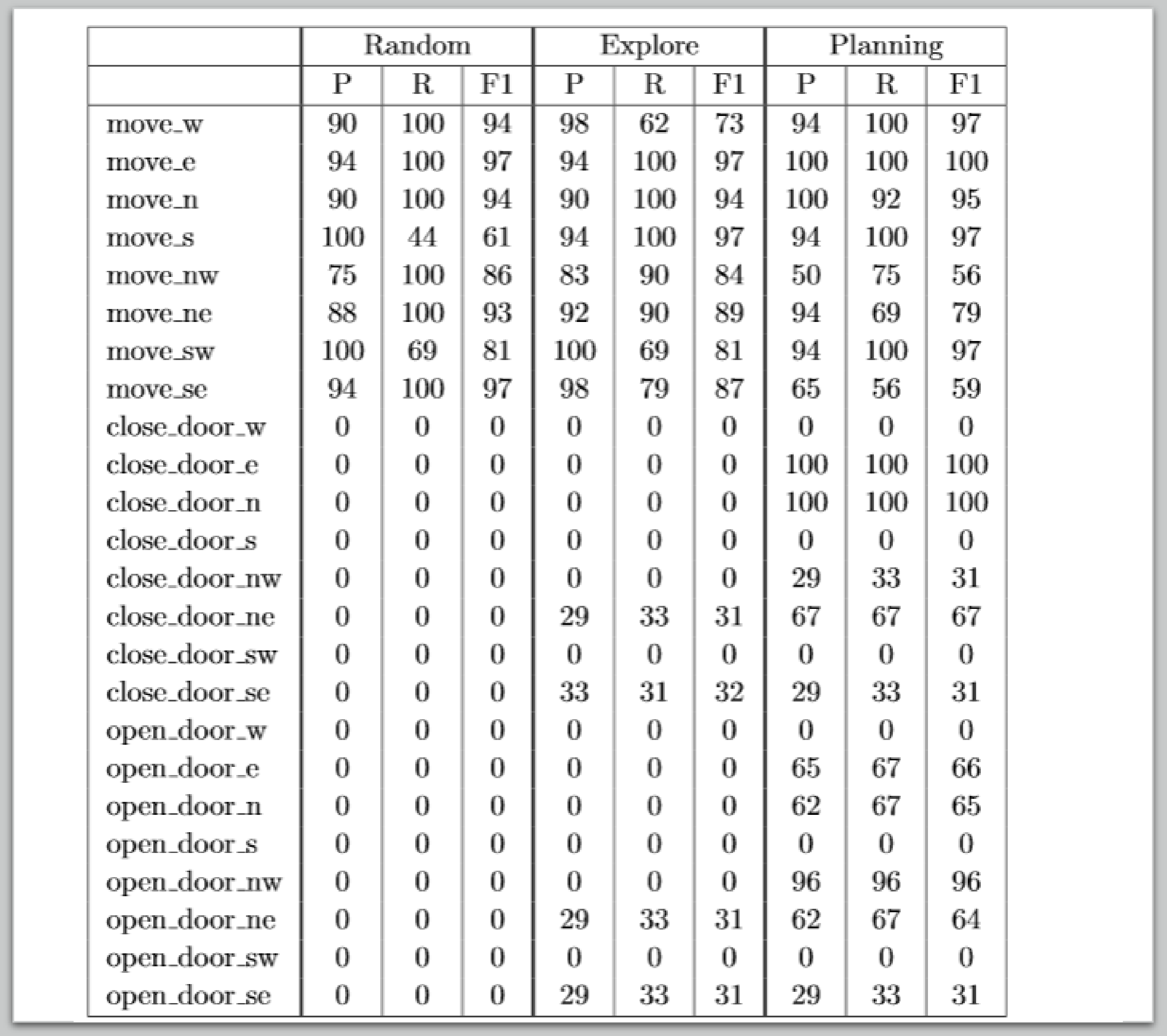
16 scenarios used to test all actions (8 open-door test scenarios not shown)

Results: F1 Score of Each Action – Scenario 1



	Random			Explore			Planning		
	P	R	F1	P	R	F1	P	R	F1
move_w	88	100	93	88	100	93	100	100	100
move_e	94	100	97	94	100	97	100	100	100
move_n	0	0	0	67	29	41	100	100	100
move_s	0	0	0	62	67	65	100	100	100
move_nw	33	23	27	67	46	54	65	52	58
move_ne	0	0	0	0	0	0	98	77	86
move_sw	0	0	0	0	0	0	56	50	53
move_se	31	33	32	67	46	54	60	48	53
close_door_w	0	0	0	0	0	0	33	33	33
close_door_e	0	0	0	0	0	0	67	67	67
close_door_n	0	0	0	0	0	0	0	0	0
close_door_s	0	0	0	0	0	0	67	67	67
close_door_nw	0	0	0	0	0	0	0	0	0
close_door_ne	0	0	0	0	0	0	0	0	0
close_door_sw	0	0	0	0	0	0	0	0	0
close_door_se	0	0	0	0	0	0	0	0	0
open_door_w	0	0	0	0	0	0	0	0	0
open_door_e	0	0	0	0	0	0	33	33	33
open_door_n	0	0	0	0	0	0	0	0	0
open_door_s	0	0	0	0	0	0	96	100	98
open_door_nw	0	0	0	0	0	0	0	0	0
open_door_ne	0	0	0	0	0	0	0	0	0
open_door_sw	0	0	0	0	0	0	0	0	0

Results: F1 Score of Each Action – Scenario 2



	Random			Explore			Planning		
	P	R	F1	P	R	F1	P	R	F1
move_w	90	100	94	98	62	73	94	100	97
move_e	94	100	97	94	100	97	100	100	100
move_n	90	100	94	90	100	94	100	92	95
move_s	100	44	61	94	100	97	94	100	97
move_nw	75	100	86	83	90	84	50	75	56
move_ne	88	100	93	92	90	89	94	69	79
move_sw	100	69	81	100	69	81	94	100	97
move_se	94	100	97	98	79	87	65	56	59
close_door_w	0	0	0	0	0	0	0	0	0
close_door_e	0	0	0	0	0	0	100	100	100
close_door_n	0	0	0	0	0	0	100	100	100
close_door_s	0	0	0	0	0	0	0	0	0
close_door_nw	0	0	0	0	0	0	29	33	31
close_door_ne	0	0	0	29	33	31	67	67	67
close_door_sw	0	0	0	0	0	0	0	0	0
close_door_se	0	0	0	33	31	32	29	33	31
open_door_w	0	0	0	0	0	0	0	0	0
open_door_e	0	0	0	0	0	0	65	67	66
open_door_n	0	0	0	0	0	0	62	67	65
open_door_s	0	0	0	0	0	0	0	0	0
open_door_nw	0	0	0	0	0	0	96	96	96
open_door_ne	0	0	0	29	33	31	62	67	64
open_door_sw	0	0	0	0	0	0	0	0	0
open_door_se	0	0	0	29	33	31	29	33	31

Outline

- Research Problem
- Lifted Linked Clause (LLC)
- Self-Directed Exploration
- Empirical Results
- **Summary**

Summary

- Strong evidence for hypotheses H1 and H5:
  - [H1] All agents will be able to learn some action preconditions from collected interactions.
  - [H5] The Planning Agent will explore more of its environment than both Explore Local Agent and the random baseline agent.
- Support for hypotheses H2, H3, H4, however not strictly true:
  - [H2] The Explore Local Agent will learn better preconditions for more actions than a baseline agent taking only random actions.
    - Scenario 2 – random learned a better model for ‘move_se’ than explore_local
  - [H3] The Explore Local Agent will explore more of its environment than the random baseline agent.
    - Only after enough actions taken
  - [H4] The Planning Agent will learn better preconditions for more actions than both the Explore Local Agent and the random baseline agent.
    - True for most actions

Summary (2)

- Solution to guiding planning and acting with incomplete action model
- Lifted Linked Clause (LLC) is a potential approach to representing contexts to guide exploration
  - Potential scaling issues, small LLCs are still useful
- Exploratory planning increases accuracy of learning action model over time vs. baselines
- Many avenues for future work:
  - Learning goals vs. achievement goals
  - Dynamic environments
  - Partially observable environments
  - Integrate into a larger cognitive architecture

Additional Material

References

- Hester, T., & Stone, P. (2017). Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, *247*, 170-186.
- Hayes-Roth, F. and McDermott, J. (1978). An interference matching technique for inducing abstractions. Communications of the ACM, 21(5):401-411.
- Law, M., Russo, A., & Broda, K. (2015). The ILASP system for learning answer set
- programs. https://www.doc.ic.ac.uk/~ml1909/ILASP.
- Sequeira, P., Melo, F. S., & Paiva, A. (2011, October). Emotion-based intrinsic motivation for reinforcement learning agents. In *International Conference on Affective Computing and Intelligent Interaction* (pp. 326-336). Springer, Berlin, Heidelberg.
- Vere, S. A. (1980). Multilevel counterfactuals for generalizations of relational concepts and productions. *Artificial intelligence*, 14(2):139-164.
- Watanabe, L. and Rendell, L. A. (1990). Effective generalization of relational descriptions. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 875-881.
- Wang, X. (1995). Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the 12th International Conference on Machine Learning*, pages 549-557.

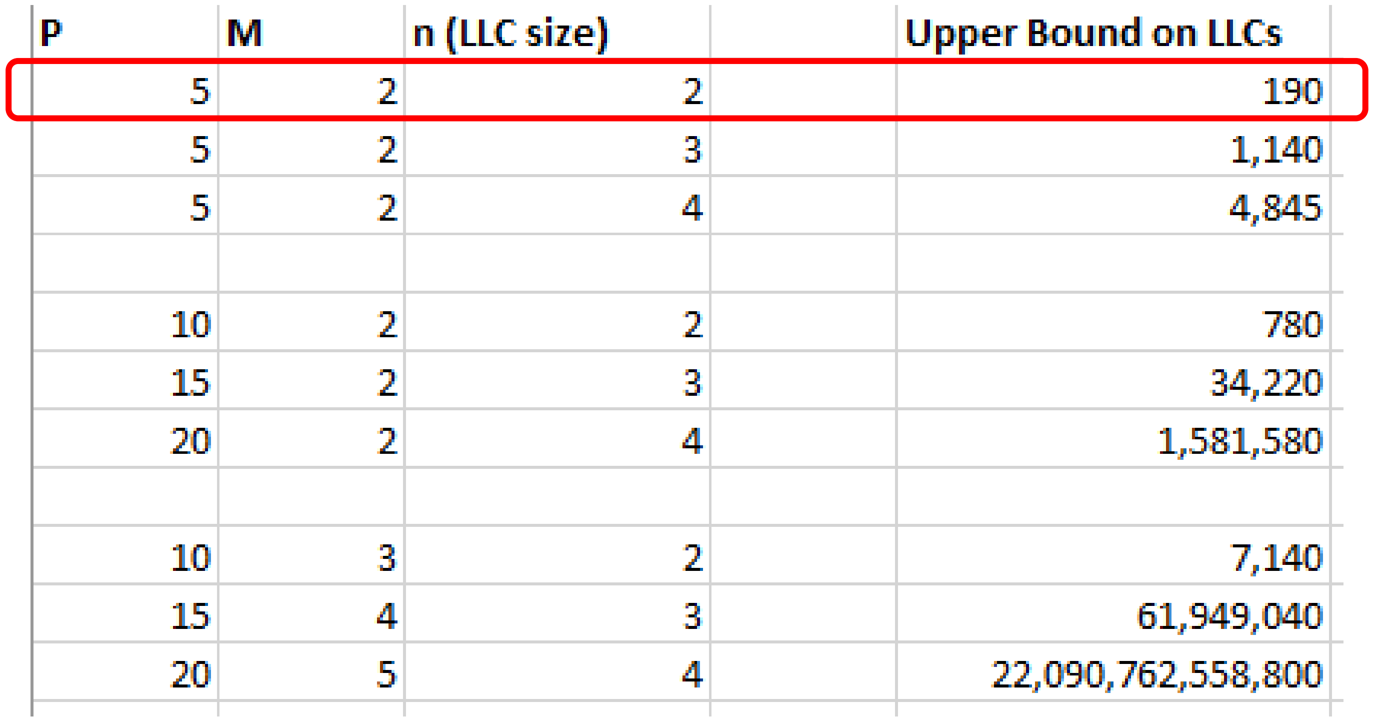
Lifted Linked Clause (LLC)

We say an LLC having n terms is of size n.

An upper bound C on the number of possible LLCs for a given size n is:

$$C = \binom{M! \times 2 \times |P|}{n}$$

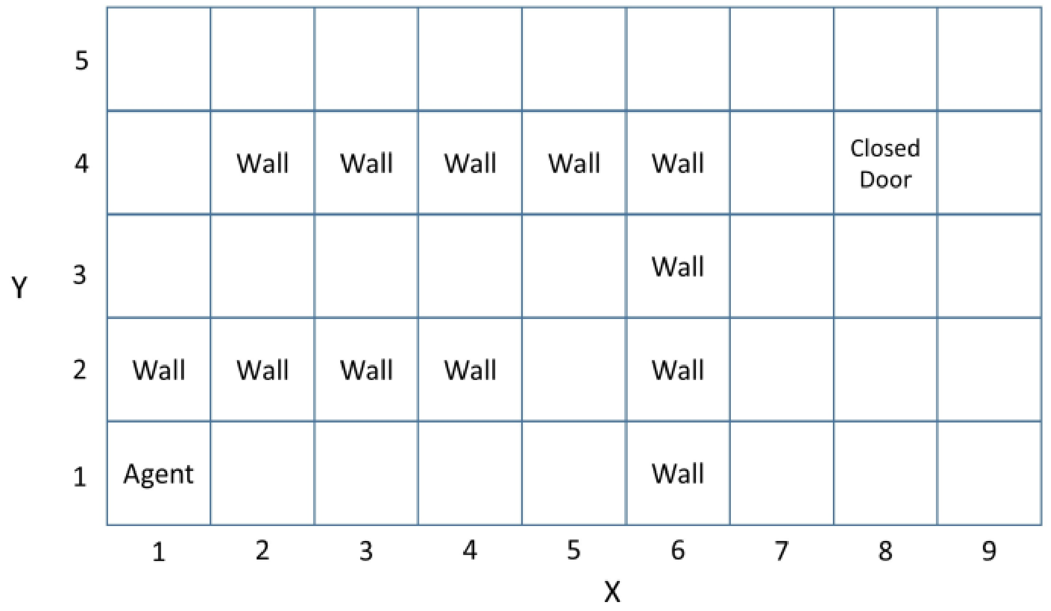
Given P predicates in a domain, and M maximum number of arguments for a single predicate $p \in P$



P	M	n (LLC size)		Upper Bound on LLCs
5	2	2		190
5	2	3		1,140
5	2	4		4,845
10	2	2		780
15	2	3		34,220
20	2	4		1,581,580
10	3	2		7,140
15	4	3		61,949,040
20	5	4		22,090,762,558,800

Learning Task	Normal Rules	Choice Rules	Constraints	Classical Negation	Brave	Cautious	Weak Constraints	Context	Algorithm for optimal solutions	Noise
Brave Induction [Sakama, Inoue 2009], XHAIL [Ray 2009], ASPAL [Corapi et al 2011], RASPAL [Athakravi et al 2013], ILED [Katzouris 2015], Inspire [Schüller 2016]	✔	✔	✖	✔	✔	✖	✖	✖	✔	✔
Cautious Induction [Sakama, Inoue 2009]	✔	✔	✖	✔	✖	✔	✖	✖	✖	✖
Induction of Stable Models [Otero 2001]	✔	✖	✖	✖	✔	✖	✖	✖	✖	✖
Induction from Answer Sets [Sakama 2005]	✔	✖	✔	✔	✔	✔	✖	✖	✖	✖
LAS [Law et al 2014]	✔	✔	✔	✖	✔	✔	✖	✖	✔	✖
LOAS [Law et al 2015]	✔	✔	✔	✖	✔	✔	✔	✖	✔	✖
Context-dependent LOAS [Law et al 2016]	✔	✔	✔	✖	✔	✔	✔	✔	✔	✖
Learning from Noisy Answer Sets [Law et al 2018]	✔	✔	✔	✖	✔	✔	✔	✔	✔	✔

PDDL for Scenario 1



(a) Graphic Representation

```
(define (problem dcss)
  (:domain dcss)
  (:objects
    x1 x2 x3 x4 x5 x6 x7 x8 x9 - xcoord
    y1 y2 y3 y4 y5 - ycoord)
  (:init
    (agentat x1 y1) (wall x1 y2)
    (north y2 y1)   (wall x2 y2)
    (north y3 y2)   (wall x3 y2)
    (north y4 y3)   (wall x4 y2)
    (north y5 y4)   (wall x2 y4)
    (west x2 x1)    (wall x3 y4)
    (west x3 x2)    (wall x4 y4)
    (west x4 x3)    (wall x5 y4)
    (west x5 x4)    (wall x6 y1)
    (west x6 x5)    (wall x6 y2)
    (west x7 x6)    (wall x6 y3)
    (west x8 x7)    (wall x6 y4)
    (west x9 x8)    (cdoor x8 y4)))
```

(b) PDDL Representation

Agents Using LLCs

1. Generate all LLCs up to a certain size (n=2)
2. Determine all LLCs active in the current state
3. Track which actions are executed per LLC (if active)
4. Use step 3 to inform action selection and goal selection
    1. If in a state with a new active LLC, try out new actions
    2. If fully state is fully explored w.r.t. actions, choose LLC that has never been visited as new goal