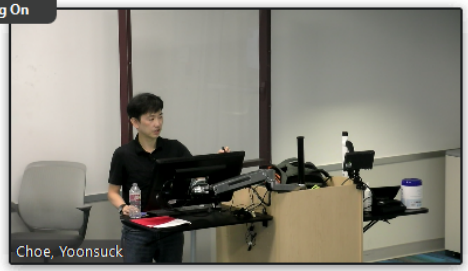
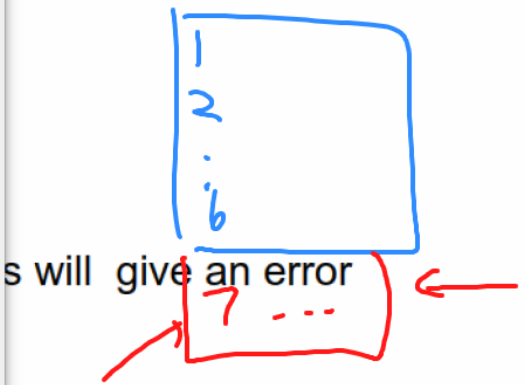
```
Run `fwupdmgr get-upgrades` for more information.

You have new mail.
Last login: Thu Sep 22 11:07:58 2022 from 128.194.120.224
yschoe@brain:~$ cd teaching/src/prover/
yschoe@brain:~/teaching/src/prover$ ls
dupe2.lsp  dupe.lsp  prover.lsp
yschoe@brain:~/teaching/src/prover$ gcl
GCL (GNU Common Lisp)  2.6.12 CLtL1    Fri Apr 22 15:51:11 UTC 2016
Source License: LGPL(gcl,gmp), GPL(unexec,bfd,xgcl)
Binary License:  GPL due to GPL'ed components: (XGCL UNEXEC)
Modifications of this banner must retain notice of a compatible license
Dedicated to the memory of W. Schelter

Use (help) to get some basic information on how to use GCL.
Temporary directory for compiler files:
/tmp/

>(load "prover.lsp")
;; Loading "prover.lsp"
;; Loading "dupe2.lsp"

"[1] T
"
"[2] NIL
"
"[3] NIL
" ;; Finished loading "dupe2.lsp"
;; Finished loading "prover.lsp"
T

>*customs*

((1 ((V X) (S X (F X))) ((E X))) (2 ((V Y) (C (F Y))) ((E Y)))
 (3 ((E (A))) NIL) (4 ((D (A))) NIL) (5 ((D Z)) ((S (A) Z)))
 (6 NIL ((D W) (V W))) (7 NIL ((D R) (C R))))

>(two-pointer *customs* 7)
```

set of suppors

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On





s will give an error

So you want to tell the approval where to look for these.

You are screen sharing Stop Share

yschoe@brain: ~/teaching/src/prover

```
>*customs*

((1 ((V X) (S X (F X))) ((E X))) (2 ((V Y) (C (F Y)))) ((E Y)))
 (3 ((E (A))) NIL) (4 ((D (A))) NIL) (5 ((D Z)) ((S (A) Z)))
 (6 NIL ((D W) (V W))) (7 NIL ((D R) (C R))))

>(two-pointer *customs* 7)
_, _: (1 ((V X) (S X (F X))) ((E X)))
_, _: (2 ((V Y) (C (F Y))) ((E Y)))
_, _: (3 ((E (A))) NIL)
_, _: (4 ((D (A))) NIL)
_, _: (5 ((D Z)) ((S (A) Z)))
_, _: (6 NIL ((D W) (V W)))
_, _: (7 NIL ((D R) (C R)))
2, 7: (8 ((V Y)) ((D (F Y))) (E Y)))
4, 7: (9 NIL ((C (A))))
5, 7: (10 NIL ((C Z)) (S (A) Z)))
3, 8: (11 ((V (A))) ((D (F (A)))))
5, 8: (12 ((V Y)) ((E Y)) (S (A) (F Y))))
6, 8: (13 NIL ((E Y) (D (F Y)) (D Y)))
1,10: (14 ((V (A))) ((C (F (A))) (E (A))))
5,11: (15 ((V (A))) ((S (A) (F (A)))))
6,11: (16 NIL ((D (F (A))) (D (A))))
1,12: (17 ((V (A))) ((E (A))))
6,12: (18 NIL ((S (A) (F Y)) (E Y) (D Y)))
4,13: (19 NIL ((D (F (A))) (E (A))))
5,13: (20 NIL ((D (F Z)) (E Z) (S (A) Z)))
3,14: (21 ((V (A))) ((C (F (A)))))
6,14: (22 NIL ((E (A)) (C (F (A))) (D (A))))
6,15: (23 NIL ((S (A) (F (A))) (D (A))))
4,16: (24 NIL ((D (F (A)))))
5,16: (25 NIL ((D (F (A))) (S (A) (A))))
3,17: (26 ((V (A))) NIL)
6,17: (27 NIL ((E (A)) (D (A))))
1,18: (28 ((V (A))) ((D (A)) (E (A))))
4,18: (29 NIL ((E (A)) (S (A) (F (A)))))
5,18: (30 NIL ((E Z) (S (A) (F Z)) (S (A) Z)))
1,20: (31 ((V (A))) ((E (F (A))) (D (F (F (A)))) (E (A))))
```

function

$V(Y) \vee C(F(Y)) \vee \neg E(Y)$

$\neg P(R) \vee C(R)$

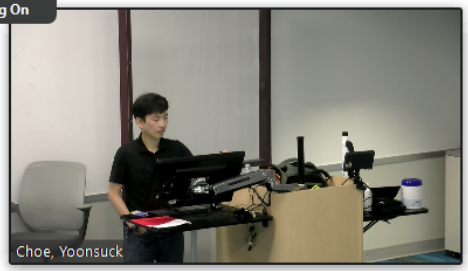
fun

$(E(A))$

i. constant.

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On



1
2
.
.
6

7 ...

s will give an error

$\neg G$

(Q) How many steps will this take?

① 5

② 10

③ >30

you have not deal with. so this is on the positive side and this is the negative side, if, when you have your why so you have that, that and that as the resulting clothes.

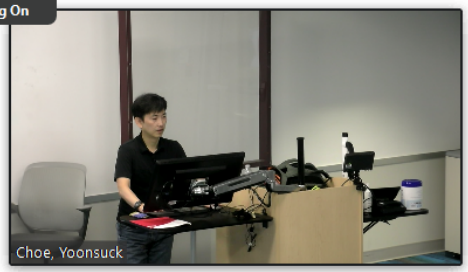
You are screen sharing Stop Share

```
 5,22: (34 NIL ((C (F (A))) (E (A)) (S (A) (A))))
 4,23: (35 NIL ((S (A) (F (A)))))
 5,23: (36 NIL ((S (A) (F (A))) (S (A) (A))))
 6,26: (37 NIL ((D (A))))
 4,27: (38 NIL ((E (A))))
 5,27: (39 NIL ((E (A)) (S (A) (A))))
 3,28: (40 ((V (A))) ((D (A))))
 5,28: (41 ((V (A))) ((E (A)) (S (A) (A))))
 1,30: (42 ((V (A))) ((S (A) (F (F (A)))) (E (F (A))) (E (A))))
 3,31: (43 ((V (A))) ((D (F (F (A)))) (E (F (A)))))
 6,31: (44 NIL ((E (A)) (D (F (F (A)))) (E (F (A))) (D (A))))
 4,32: (45 NIL ((C (F (A)))))
 5,32: (46 NIL ((C (F (A))) (S (A) (A))))
 4,37: (47 NIL NIL)
T

>(unit-preference *customs*)
 _, _: (1 ((V X) (S X (F X))) ((E X)))
 _, _: (2 ((V Y) (C (F Y))) ((E Y)))
 _, _: (3 ((E (A))) NIL)
 _, _: (4 ((D (A))) NIL)
 _, _: (5 ((D Z)) ((S (A) Z)))
 _, _: (6 NIL ((D W) (V W)))
 _, _: (7 NIL ((D R) (C R)))
 4, 6: (8 NIL ((V (A))))
 3, 1: (9 ((S (A) (F (A))) (V (A))) NIL)
 4, 7: (10 NIL ((C (A))))
 3, 2: (11 ((C (F (A))) (V (A))) NIL)
 8, 9: (12 ((S (A) (F (A)))) NIL)
12, 5: (13 ((D (F (A)))) NIL)
 8,11: (14 ((C (F (A)))) NIL)
13, 6: (15 NIL ((V (F (A)))))
 8, 1: (16 ((S (A) (F (A)))) ((E (A))))
13, 7: (17 NIL ((C (F (A)))))
14,17: (18 NIL NIL)
T

>
```

two-pointer (40)

s will give an error

unit-preference. (11)



And this took 11 steps.

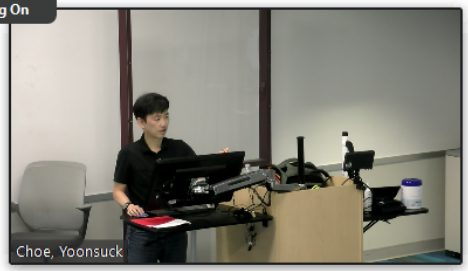
```
 1,30: (42 ((V (A))) ((S (A) (F (F (A)))) (E (F (A))) (E (A))))
 3,31: (43 ((V (A))) ((D (F (F (A)))) (E (F (A)))))
 6,31: (44 NIL ((E (A)) (D (F (F (A)))) (E (F (A))) (D (A))))
 4,32: (45 NIL ((C (F (A)))))
 5,32: (46 NIL ((C (F (A))) (S (A) (A))))
 4,37: (47 NIL NIL)
T

>(unit-preference *customs*)
 _, _: (1 ((V X) (S X (F X))) ((E X)))
 _, _: (2 ((V Y) (C (F Y))) ((E Y)))
 _, _: (3 ((E (A))) NIL)
 _, _: (4 ((D (A))) NIL)
 _, _: (5 ((D Z)) ((S (A) Z)))
 _, _: (6 NIL ((D W) (V W)))
 _, _: (7 NIL ((D R) (C R)))
 4, 6: (8 NIL ((V (A))))
 3, 1: (9 ((S (A) (F (A))) (V (A))) NIL)
 4, 7: (10 NIL ((C (A))))
 3, 2: (11 ((C (F (A))) (V (A))) NIL)
 8, 9: (12 ((S (A) (F (A)))) NIL)
12, 5: (13 ((D (F (A)))) NIL)
 8,11: (14 ((C (F (A)))) NIL)
13, 6: (15 NIL ((V (F (A)))))
 8, 1: (16 ((S (A) (F (A)))) ((E (A))))
13, 7: (17 NIL ((C (F (A)))))
14,17: (18 NIL NIL)
T

>*howling*

((1 ((HOWL Z)) ((HOUND Z)))
 (2 NIL ((HAVE X Y) (CAT Y) (HAVE X Z) (MOUSE Z)))
 (3 NIL ((LS W) (HAVE W V) (HOWL V))) (4 ((HAVE (JOHN) (A))) NIL)
 (5 ((CAT (A)) (HOUND (A))) NIL) (6 ((MOUSE (B))) NIL)
 (7 ((LS (JOHN))) NIL) (8 ((HAVE (JOHN) (B))) NIL))

>
```

6

6

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On



s will give an error

So when you run the two pointer method, you have to specify number six.

You are screen sharing Stop Share

```
yschoe@brain: ~/teaching/src/prover
26,36: (79 NIL ((HAVE X (B)) (HAVE X (A))))
 8,37: (80 NIL ((HOUND (B))))
33,37: (81 NIL ((MOUSE (B)) (HAVE (JOHN) (A))))
21,39: (82 NIL ((HAVE (JOHN) (A)) (HAVE (JOHN) (B))))
34,39: (83 NIL ((HAVE (JOHN) (A))))
26,44: (84 NIL ((MOUSE (A))))
 8,45: (85 ((HOUND (A))) NIL)
12,47: (86 ((HOWL (A))) NIL)
28,47: (87 NIL ((LS W) (HAVE W (A))))
31,47: (88 NIL ((LS (JOHN))))
26,48: (89 NIL ((HAVE (JOHN) (B))))
26,49: (90 NIL ((MOUSE (B))))
47,54: (91 NIL NIL)
T

>(unit-preference *howling*)
 _, _: (1 ((HOWL Z)) ((HOUND Z)))
 _, _: (2 NIL ((HAVE X Y) (CAT Y) (HAVE X G46405) (MOUSE G46405)))
 _, _: (3 NIL ((LS W) (HAVE W V) (HOWL V)))
 _, _: (4 ((HAVE (JOHN) (A))) NIL)
 _, _: (5 ((CAT (A)) (HOUND (A))) NIL)
 _, _: (6 ((MOUSE (B))) NIL)
 _, _: (7 ((LS (JOHN))) NIL)
 _, _: (8 ((HAVE (JOHN) (B))) NIL)
 4, 3: (9 NIL ((HOWL (A)) (LS (JOHN))))
 4, 2: (10 NIL ((MOUSE G46405) (HAVE (JOHN) G46405) (CAT (A))))
 4, 2: (11 NIL ((MOUSE (A)) (CAT Y) (HAVE (JOHN) Y)))
 4,10: (12 NIL ((CAT (A)) (MOUSE (A))))
 7, 9: (13 NIL ((HOWL (A))))
13, 1: (14 NIL ((HOUND (A))))
 6,10: (15 NIL ((CAT (A)) (HAVE (JOHN) (B))))
14, 5: (16 ((CAT (A))) NIL)
 7, 3: (17 NIL ((HOWL V) (HAVE (JOHN) V)))
 8,15: (18 NIL ((CAT (A))))
16,18: (19 NIL NIL)
T

>
```

(mouse G46405)

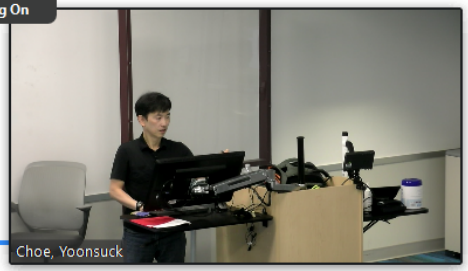
mouse (G46405)

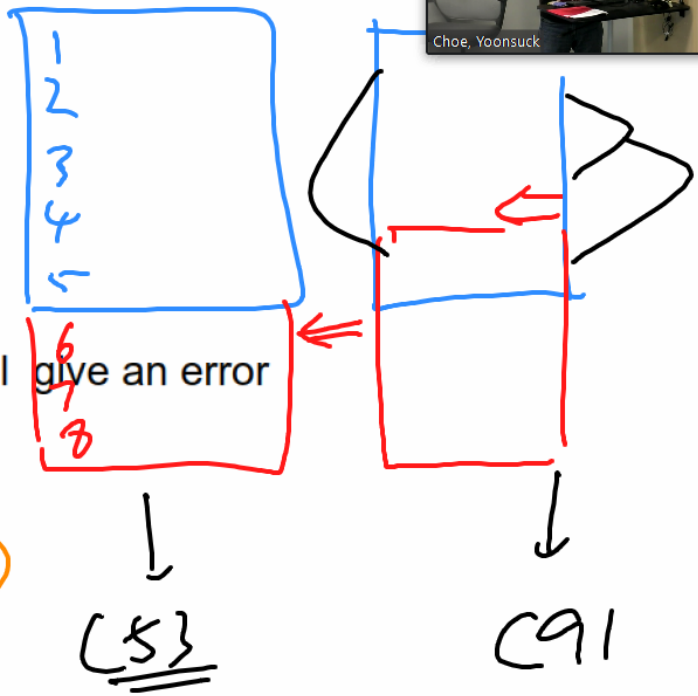
C1 : P(x)

C2 : P(x)

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

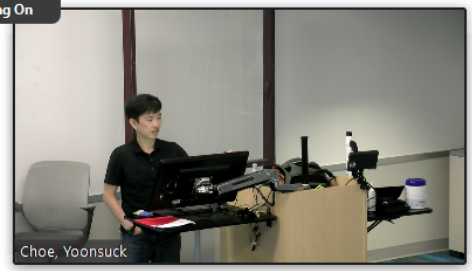
Who can see what you share here? Recording On



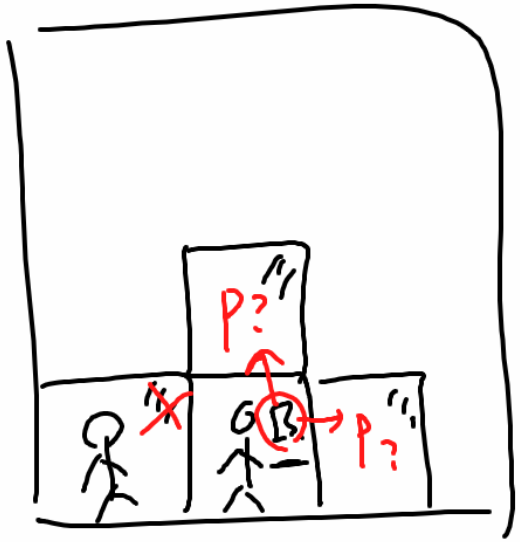


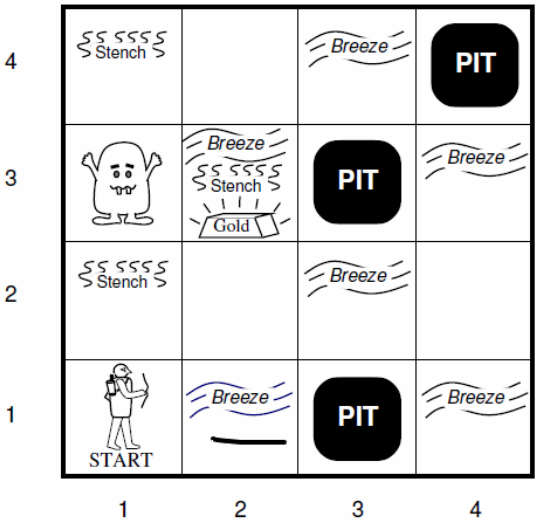
Before we continue, education. And that's where these thin

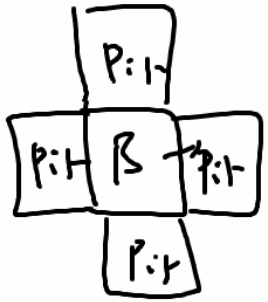
You are screen sharing Stop Share



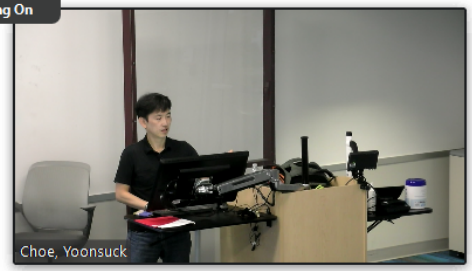
Example Domain: Wumpus World



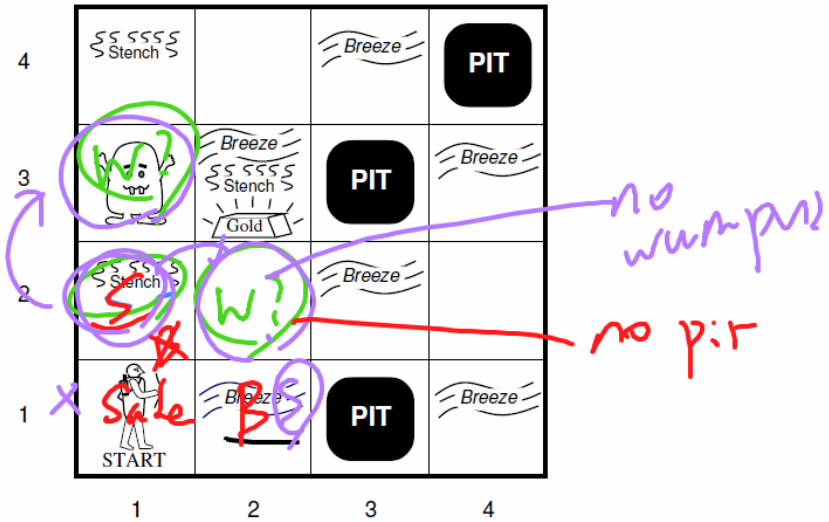


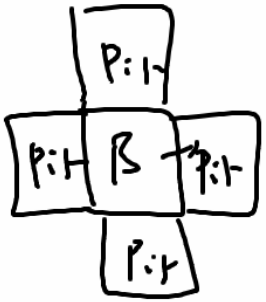


- Want to get to the gold and grab it.
- Want to avoid pits and the “wumpus”.
- Clues: breeze near pits and stench near the wumpus.
- Other sensors: wall (bump), gold (glitter), kill (scream)
- Actions: move, grab, or shoot



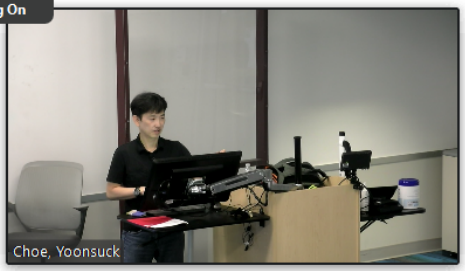
Example Domain: Wumpus World



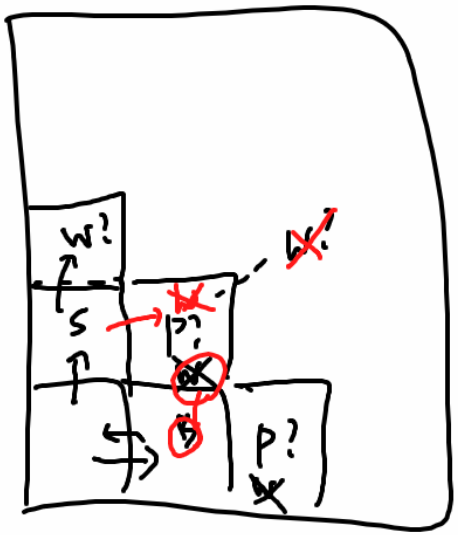


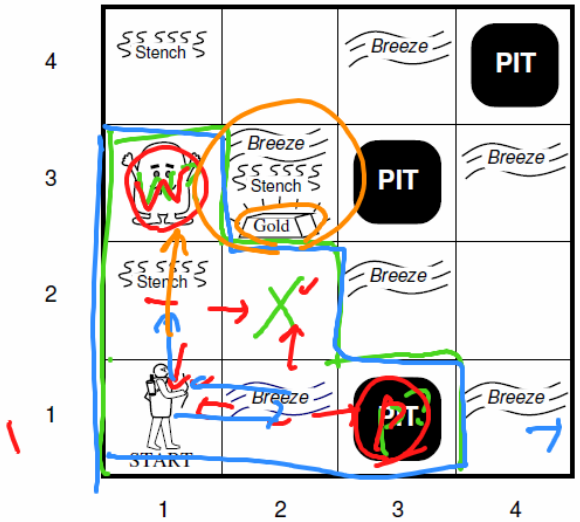
- Want to get to the gold and grab it.
- Want to avoid pits and the “wumpus”.
- Clues: breeze near pits and stench near the wumpus.
- Other sensors: wall (bump), gold (glitter), kill (scream)
- Actions: move, grab, shoot

You can use all of the information that you get so far, to eliminate all of these possibilities. Right.

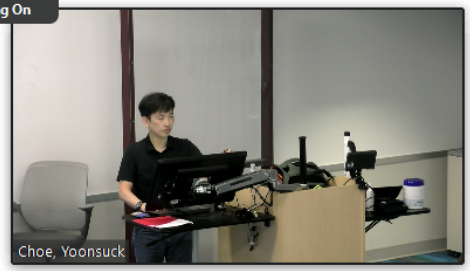


Example Domain: Wumpus World

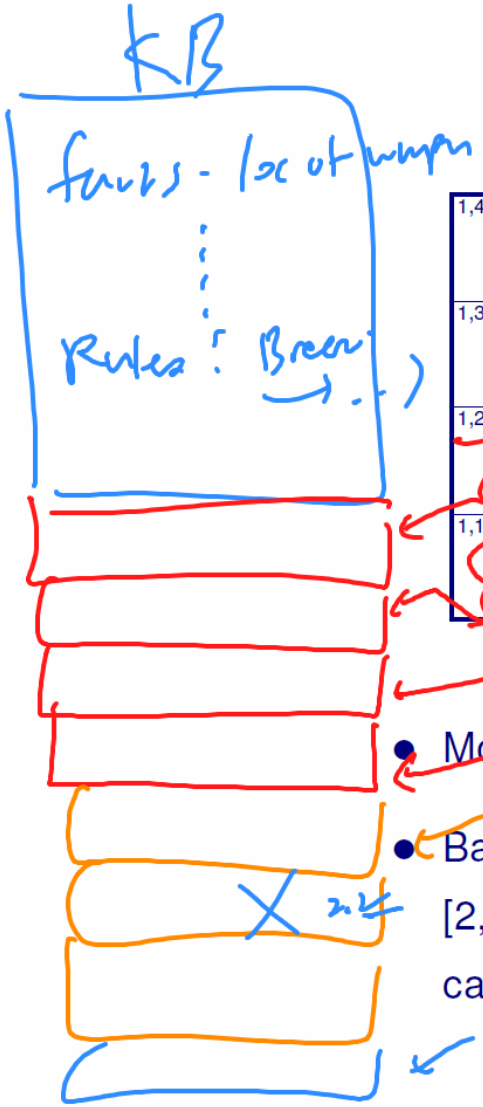




- Want to get to the gold and grab it.
- Want to avoid pits and the "wumpus".
- Clues: breeze near pits and stench near the wumpus.
- Other sensors: wall (bump), gold (glitter), kill (scream)
- Actions: move, grab, or shoot



Evolution of Knowledge in WW



1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

(a)

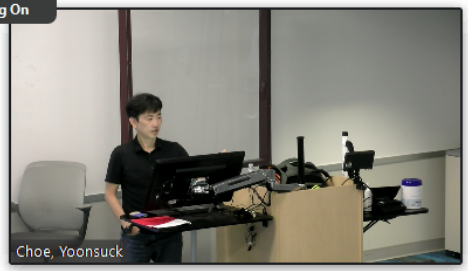
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

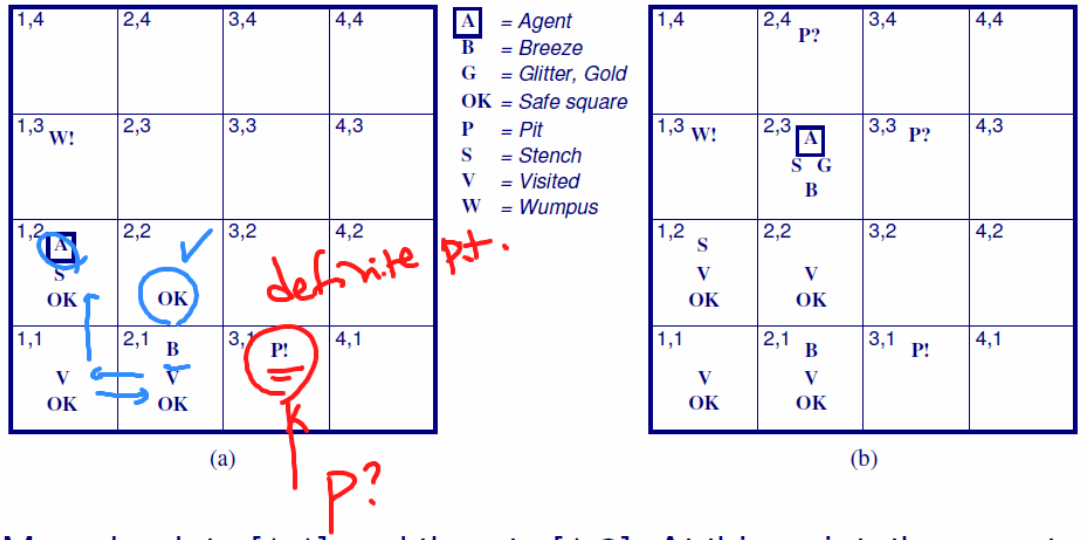
(b)

- Move from [1,1] to [2,1].
- Based on the sensory data (breeze: marked **B**), we can mark [2,2] and [3,1] as potential pits (marked **P?**), but not [1,1] since we came from there and we already know there's no pit there.

So keep updating knowledge base base on this kind of exploration.

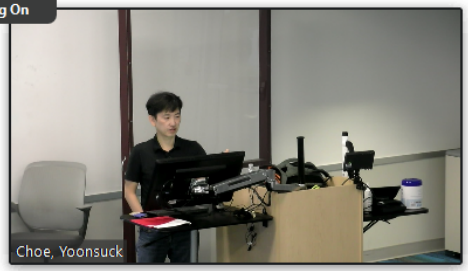


Evolution of Knowledge in WW

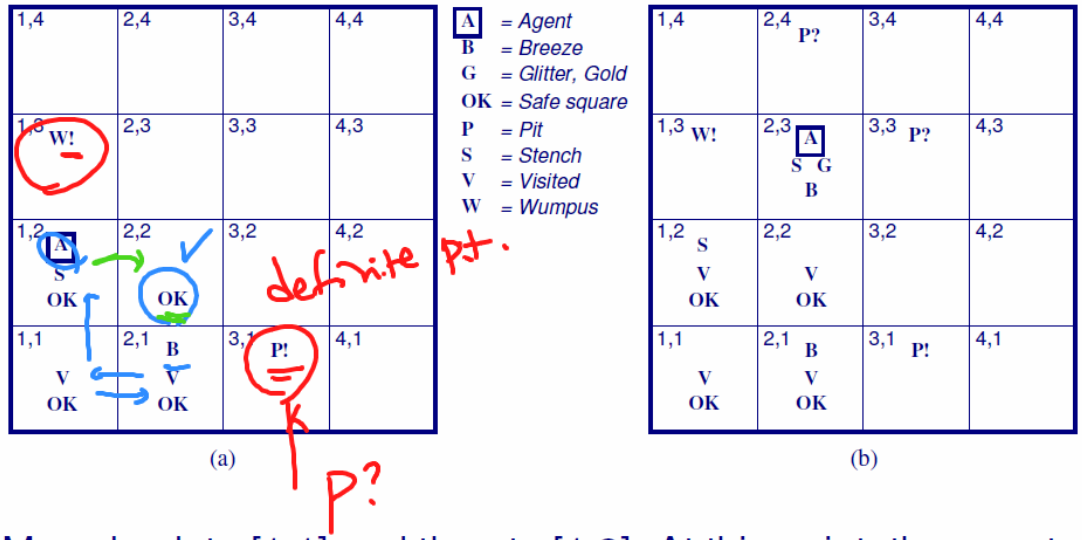


- Move back to [1,1] and then to [1,2]. At this point, the agent can infer that the wumpus is in [1,3]
  - In [1,2], stench **S** is perceived, and all neighboring grid except [1,3] are known to be **OK** (been to [1,1], and no breeze **B**, so [2,2] is **OK**).
- Then move to [2,2] and then to [2,3] where the gold can be gound (glitter).

			theorem proving				11:59pm		
6	9/27/2022	9/29/2022	First-order logic and theorem proving	Ch 8, 9	Ch 8, 9			slide03-logic	
7	10/4/2022	10/6/2022	First-order logic and theorem proving; Prolog ; Planning;	Ch 8, 9	Ch 8, 9			slide03-logic, slide04-planning	
8	**no class**	10/13/2022 **Midterm Exam**	**Midterm Exam (Thu, in class)**	Ch 10	Ch 11	hw3 announced	~~hw2 due: 10/11 Tuesday 11:59pm~~; hw2 due postponed to Sunday 10/16 11:59pm		10/10: midsemester grades due
9	10/18/2022	10/20/2022	Uncertainty and probabilistic reasoning	Ch 13, 14	Ch 12, 13				
10	10/25/2022	10/27/2022	Uncertainty and probabilistic reasoning	Ch 13, 14	Ch 12, 13				
11	11/1/2022	11/3/2022	Machine learning intro	Ch 18.1-18.2	Ch 19.1-19.2	hw4 announced	hw3 due		
12	11/8/2022	11/10/2022	Machine learning : decision tree	Ch 18.3	Ch 19.3				
13	11/15/2022	11/17/2022	Machine learning:	Ch 18.7	Ch 21.1				

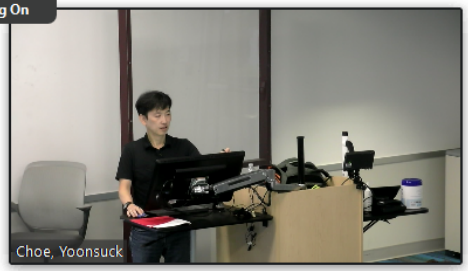


Evolution of Knowledge in WW

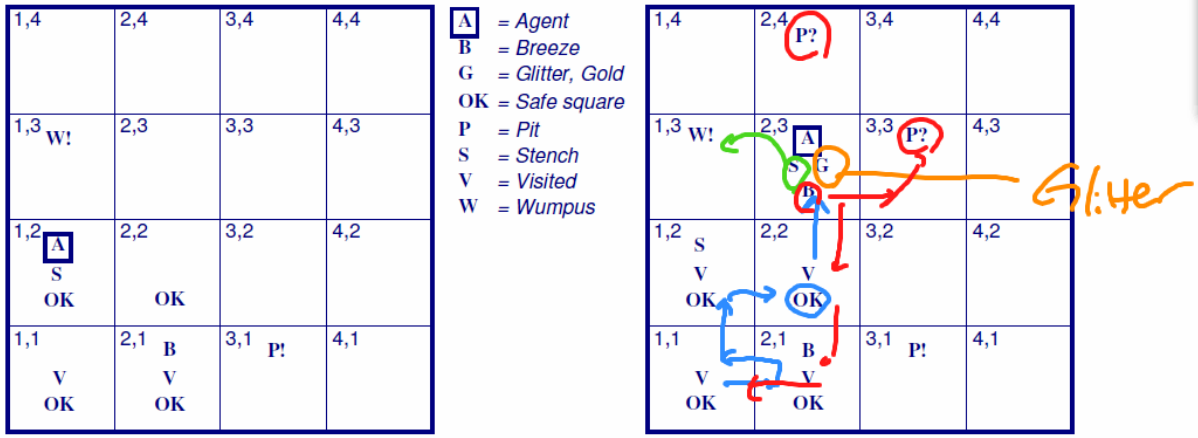


- Move back to [1,1] and then to [1,2]. At this point, the agent can infer that the wumpus is in [1,3]
  - In [1,2], stench **S** is perceived, and all neighboring grid except [1,3] are known to be **OK** (been to [1,1], and no breeze **B**, so [2,2] is **OK**).
- Then move to [2,2] and then to [2,3] where the gold can be found (glitter).

So that's the left.



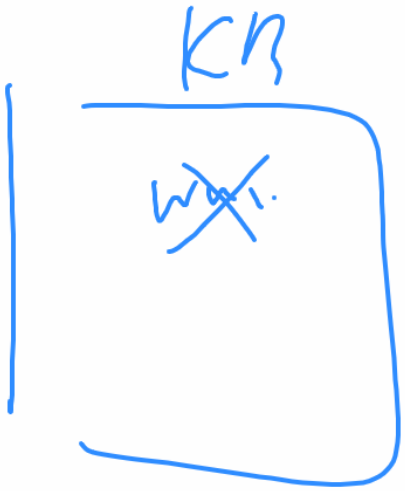
Evolution of Knowledge in WW

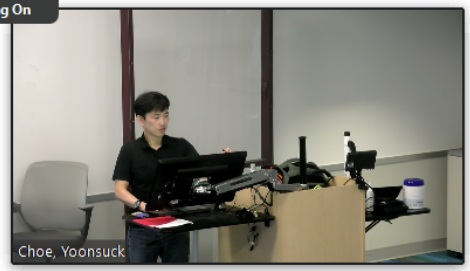


(a) (b)

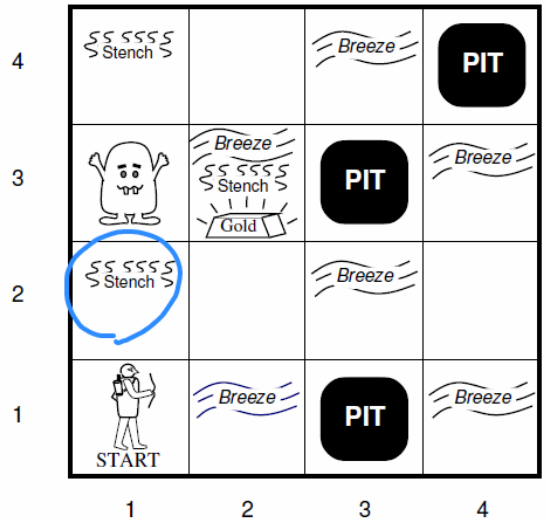
- Move back to [1,1] and then to [1,2]. At this point, the agent can infer that the wumpus is in [1,3]
  - In [1,2], stench **S** is perceived, and all neighboring grid except [1,3] are known to be **OK** (been to [1,1], and no breeze **B**, so [2,2] is **OK**).
- Then move to [2,2] and then to [2,3] where the gold can be gound (glitter).

And then you can grab the gold, and you'd be done was some of these tests might actually asked you to bring it back to the initial position.





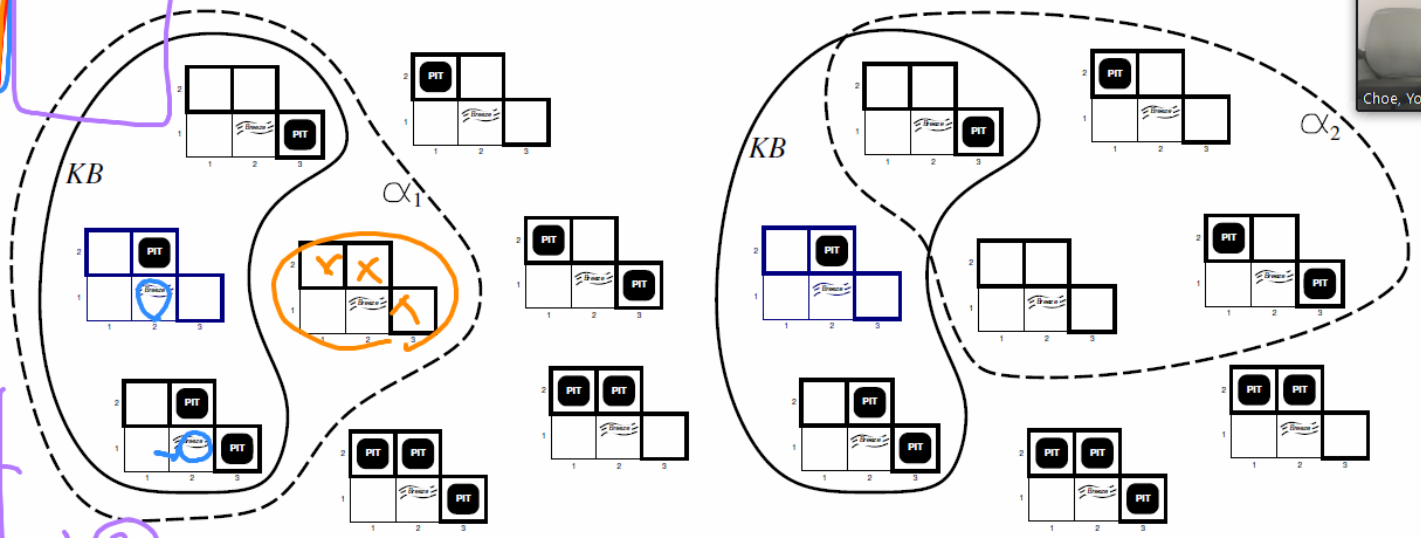
Inference in Wumpus World



- Knowledge Base: basic rules of the Wumpus World.
- Additional knowledge is added to the KB: facts you gather as you explore ([x,y] has stench, breeze, etc.)
- We can ask if a certain statement is a logical consequence of the KB: "There is a pit in [1,2]"

So, one to this this is the location.

Inference in Wumpus World



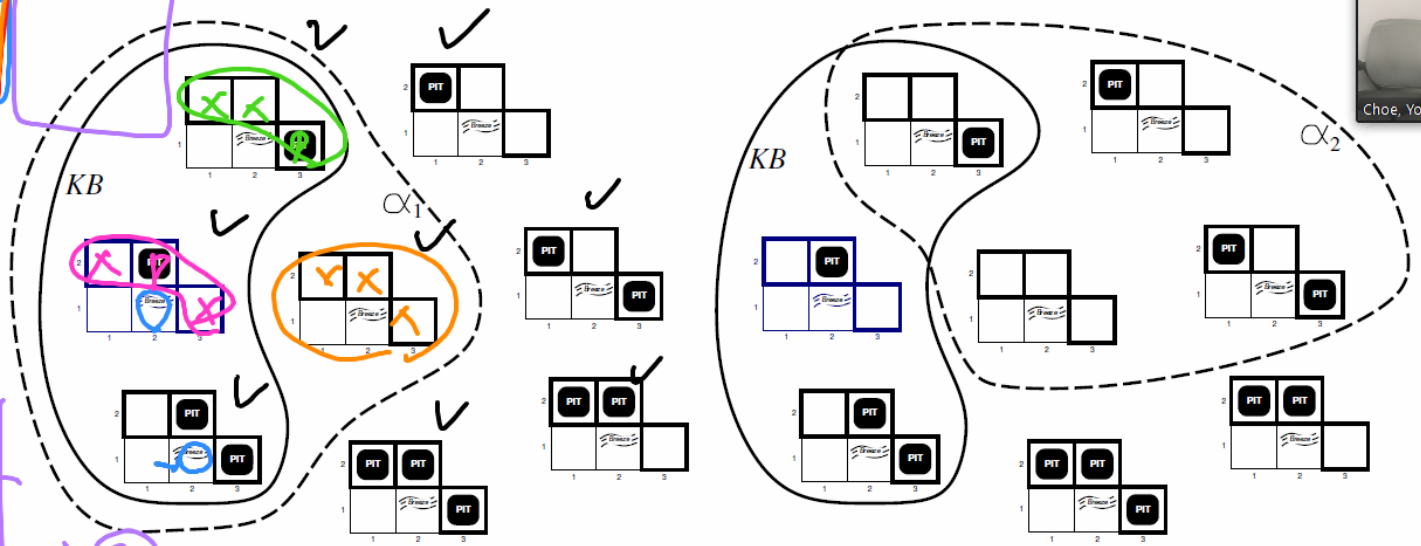
KB: basic rules, plus [1,1] and [2,1] explored

Pits in three unvisited grid [1,2], [2,2], [3,1]: $2^3 = 8$ possible cases

- α_1 = "There is no pit in [1,2]"
- α_2 = "There is no pit in [2,2]"
- Only α_1 follows from the KB (KB is the subset of α_1), but not α_2.

And that's exactly what we have here so this would be this case know pit no kids no pets.

Inference in Wumpus World



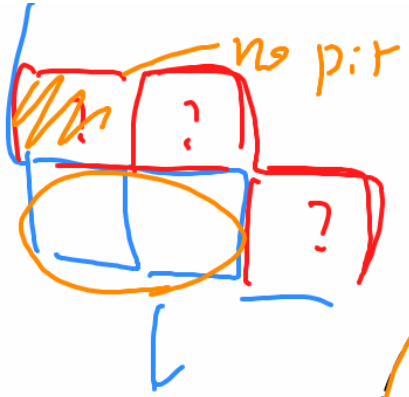
KB: basic rules, plus [1,1] and [2,1] explored

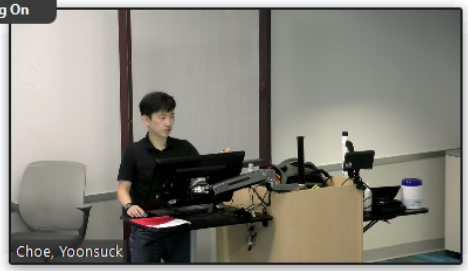
Pits in three unvisited grid [1,2], [2,2], [3,1]: $2^3 = 8$ possible cases

- α_1 = "There is no pit in [1,2]"
- α_2 = "There is no pit in [2,2]"
- Only α_1 follows from the KB (KB is the subset of α_1), but not α_2.

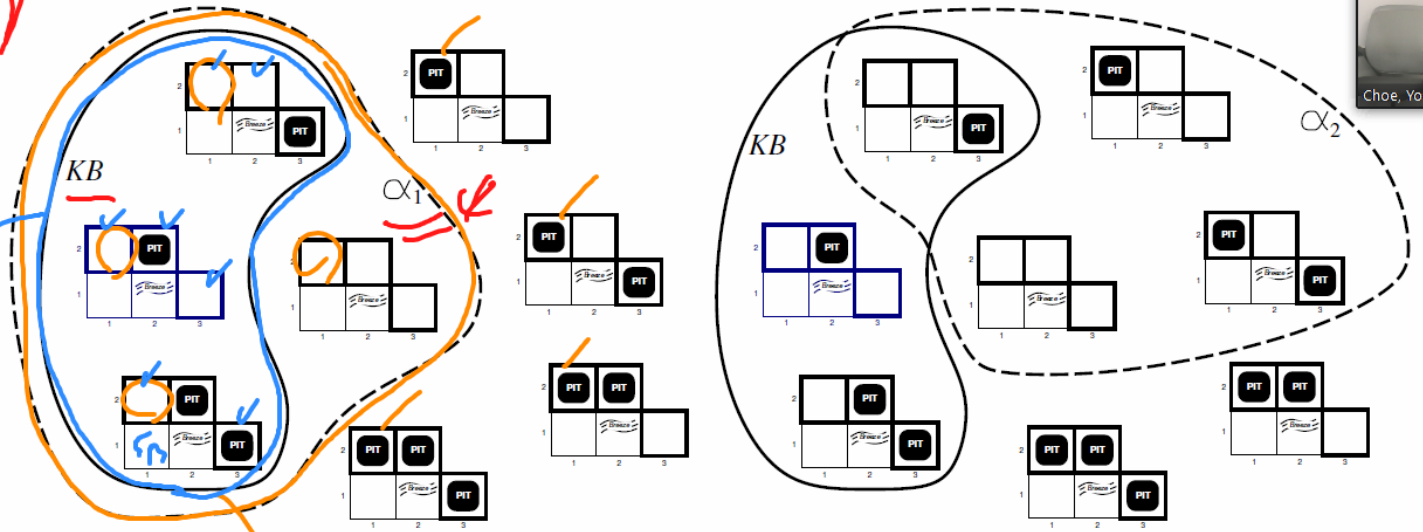
So, what do you what what what should it.

You are screen sharing Stop Share





Inference in Wumpus World



KB: basic rules, plus [1,1] and [2,1] explored

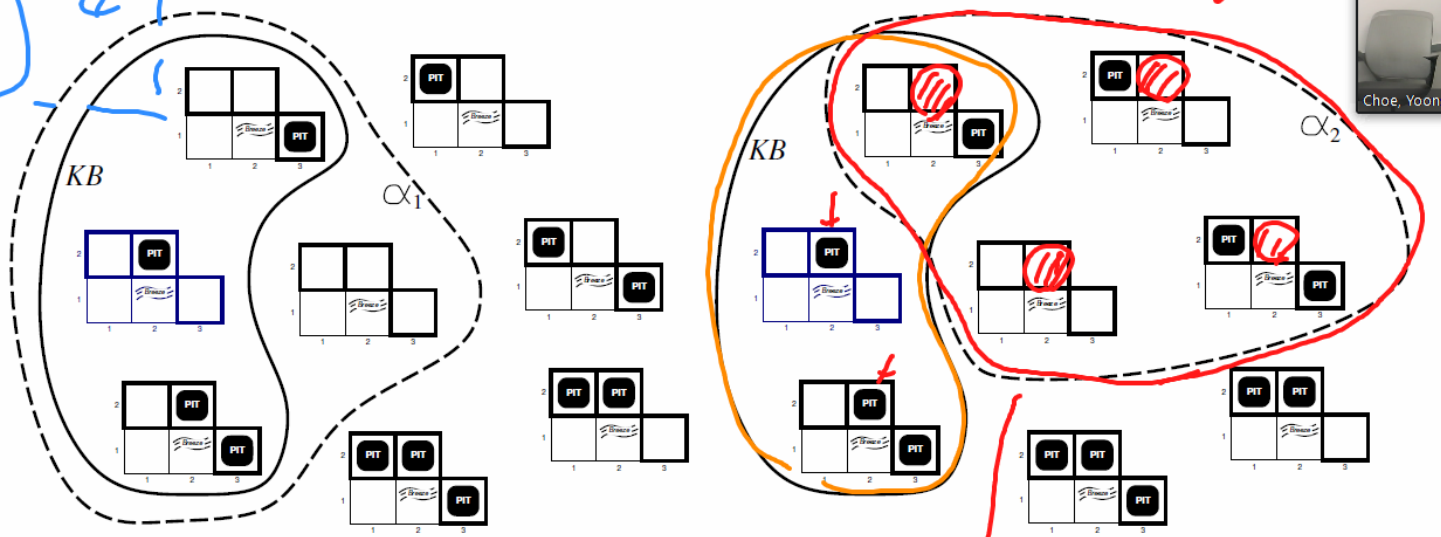
Pits in three unvisited grid [1,2], [2,2], [3,1]: $2^3 = 8$ possible cases

- α_1 = "There is no pit in [1,2]"
- α_2 = "There is no pit in [2,2]"
- Only α_1 follows from the KB (KB is the subset of α_1), but not α_2.

no pit in [1,2] is true

So now using the same.

Inference in Wumpus World



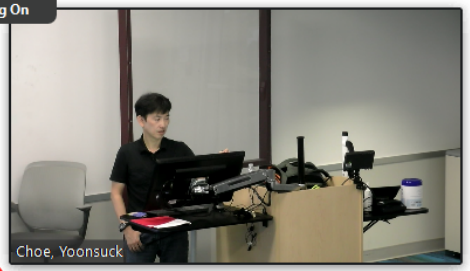
KB: basic rules, plus [1,1] and [2,1] explored

Pits in three unvisited grid [1,2], [2,2], [3,1]: $2^3 = 8$ possible cases

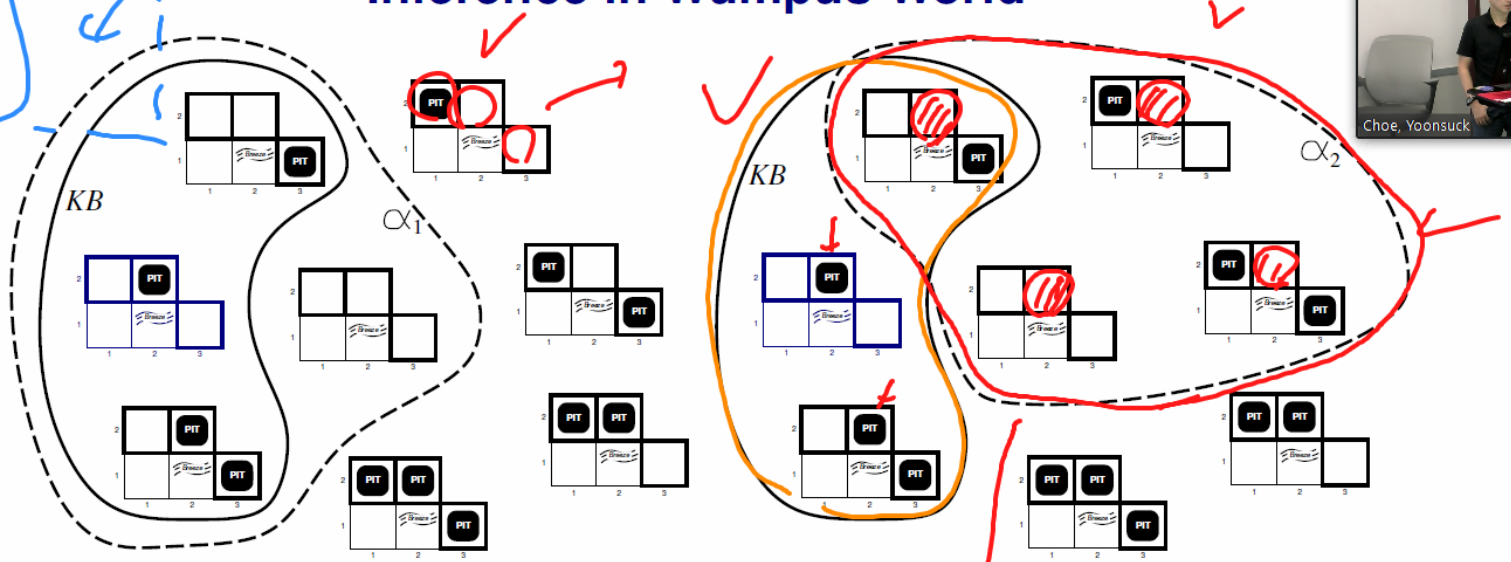
- α_1 = "There is no pit in [1,2]"
- α_2 = "There is no pit in [2,2]"
- Only α_1 follows from the KB (KB is the subset of α_1), but not α_2.

→ false. or α_2 is not a logical consequence of the KB.

of the knowledge base.



Inference in Wumpus World



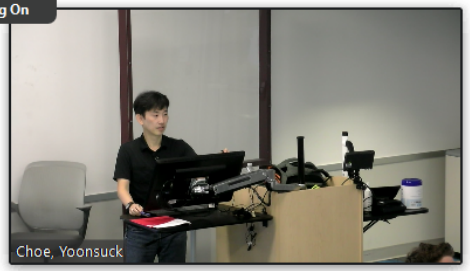
KB: basic rules, plus [1,1] and [2,1] explored

Pits in three unvisited grid [1,2], [2,2], [3,1]: $2^3 = 8$ possible cases

- α_1 = "There is no pit in [1,2]"
- α_2 = "There is no pit in [2,2]"
- Only α_1 follows from the KB (KB is the subset of α_1), but not α_2.

False. or α_2 is not a logical consequence of the KB.

And then just based on your query, get the subset, and then check if the subset includes the knowledge.

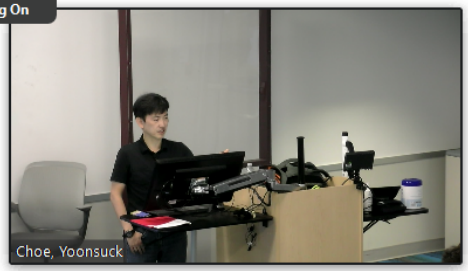


Propositonal-logic-based Agent

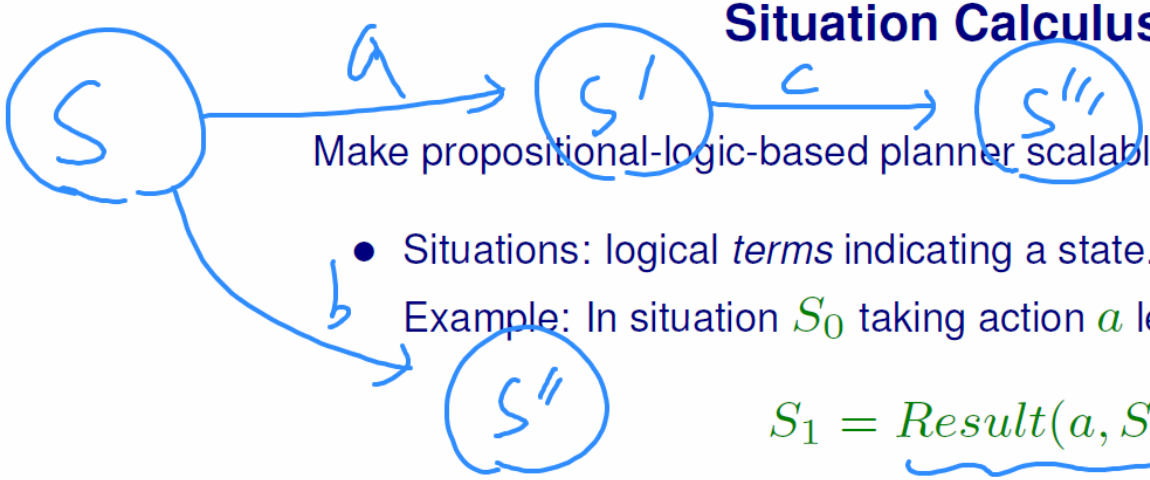
- Query KB: Is there a Wumpus in [x,y]? Is there a pit in [x,y]?
- Add knowledge to KB (perceptual input): Breeze felt in [x,y], Stench detected in [x,y], etc.
- Decide which action to take (move where, etc.): Move to [x,y], grab gold, etc.

Note: here, there's only one goal, to grab the gold. Can we specify an arbitrary goal and derive a plan?

Problem: Propositions need to be explicit about location, e.g.,
$Breeze_{x,y}, Stench_{x,y}, \neg Wumpus_{x,y}$.



Situation Calculus



Make propositional-logic-based planner scalable.

- Situations: logical *terms* indicating a state.
  Example: In situation S_0 taking action a leads to situation S_1:

$$S_1 = Result(a, S_0).$$

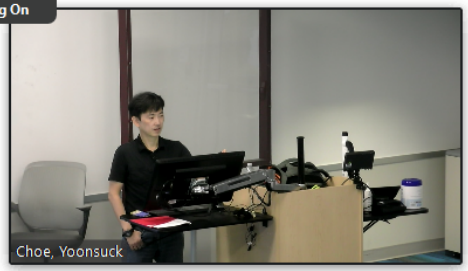
- Fluents: *funcitons* and *predicates* that vary from one situation to the next.
  Example: $\neg Holding(Gold_1, S_0), Age(Wumpus)$

Other stuff: Atemporal/eternal predicates $Gold(Gold_1)$, empty actions $Result([], s) = s$, sequence of actions $Result([a|seq], s) = Result(seq, Result(a, s))$.

So you can represent that as some.

Prolog 3/5

- you can insert
  - write('string')
  - write(X)
  - etc. anywhere in the rule to printout a message or the content of a variable.
  - It can be chained:
    - write('string: '), write(X), write('string2: '), write(Y)
- enter [user]. to begin manual entry:
  - loves(john,jane).
  - loves(jane,john).
  - couple(X,Y) :- loves(X,Y) , loves(Y,X).
  - CTRL-D to end



Situation Calculus

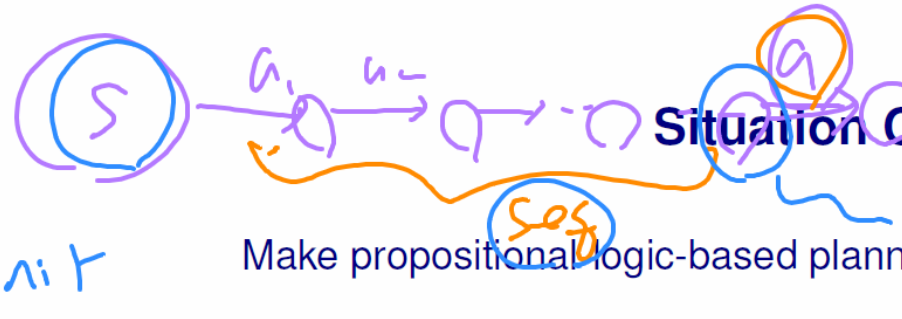
Make propositional-logic-based planner scalable.

- Situations: logical *terms* indicating a state.
  Example: In situation S_0 taking action a leads to situation S_1:

$$S_1 = Result(a, S_0).$$

- Fluents: *funcitons* and *predicates* that vary from one situation to the next.
  Example: $\neg Holding(Gold_1, S_0), Age(Wumpus)$

Other stuff: Atemporal/eternal predicates $Gold(Gold_1)$, empty actions $Result([], s) = s$, sequence of actions $Result([a|seq], s) = Result(seq, Result(a, s))$.



Situation Calculus

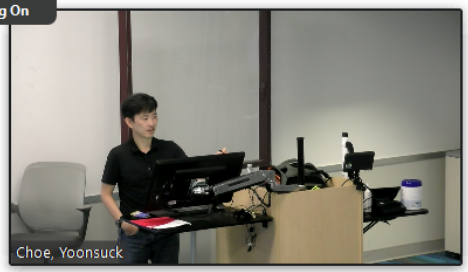
Make propositional logic-based planner scalable.

- Situations: logical *terms* indicating a state.
  Example: In situation S_0 taking action a leads to situation S_1:

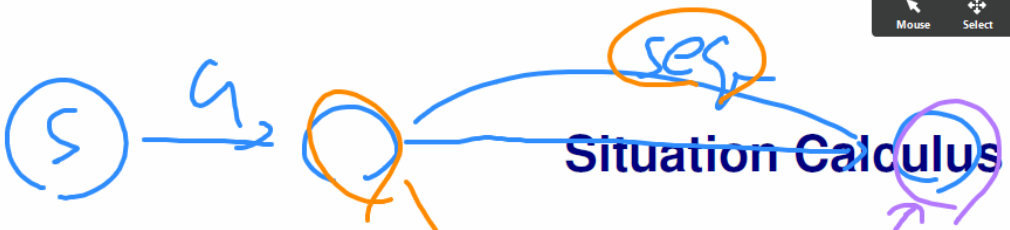
$$S_1 = Result(a, S_0).$$

- Fluents: *funcitons* and *predicates* that vary from one situation to the next.
  Example: $\neg Holding(Gold_1, S_0), Age(Wumpus)$

Other stuff: Atemporal/eternal predicates $Gold(Gold_1)$, empty actions $Result([], s) = s$, sequence of actions
$Result([a|seq], s) = Result(seq, Result(a, s))$.



Right. So actually what is the result of as, so actually.



Situation Calculus

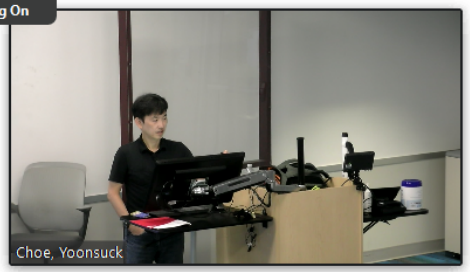
Make propositional-logic-based planner scalable.

- Situations: logical *terms* indicating a state.
  Example: In situation S_0 taking action a leads to situation S_1:

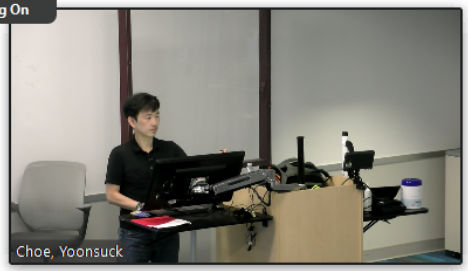
$$S_1 = Result(a, S_0).$$

- Fluents: *funcitons* and *predicates* that vary from one situation to the next.
  Example: $\neg Holding(Gold_1, S_0), Age(Wumpus)$

Other stuff: Atemporal/eternal predicates $Gold(Gold_1)$, empty actions $Result([], s) = s$, sequence of actions
$Result([a|seq], s) = Result(seq, Result(a, s))$.



And then followed by this sequence. And finally, you get the resulting situation. As a result.



Describing Actions in Situation Calculus

Two axioms:

- Possibility axiom: when it is possible to execute an action

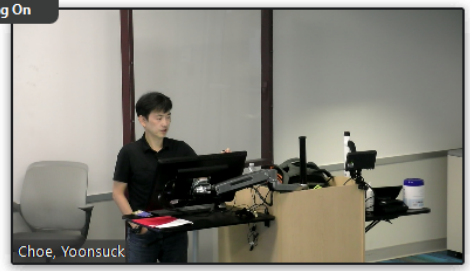
$$\text{Preconditions} \rightarrow Poss(a, s)$$

- Effect axiom: What happens when a possible action is taken

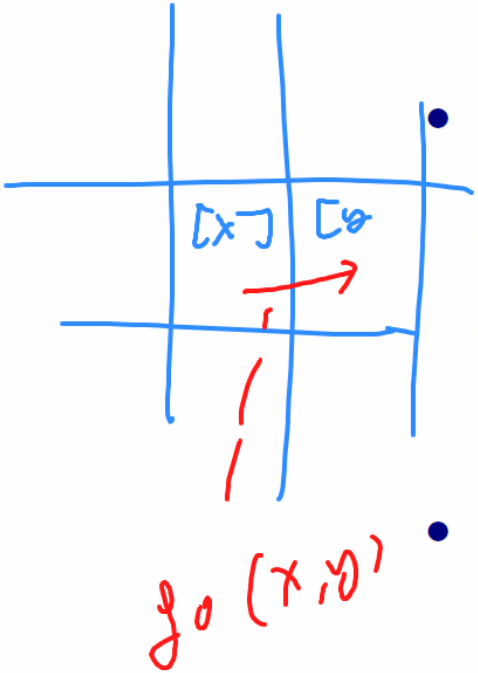
$$Poss(a, s) \rightarrow \text{Changes that result}$$

Hold

Then the fluent for who will will change.



Wumpus World: Axioms



- Possibility axioms: Move, grab, release

$$At(Agent, x, s) \wedge Adjacent((x, y), (x, y)) \rightarrow Poss(Go(x, y), s)$$

$$Gold(g) \wedge At(Agent, x, s) \wedge At(g, x, s) \rightarrow Poss(Grab(g), s)$$

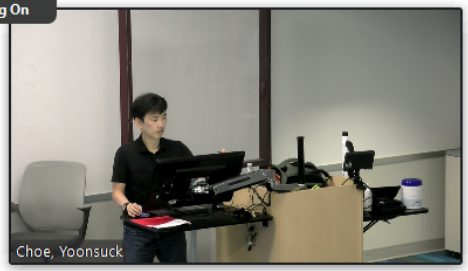
$$Holding(g, s) \rightarrow Poss(Release(g), s)$$

- Effect axioms: Move, Grab, Release

$$Poss(Go(x, y), s) \rightarrow At(Agent, y, Result(Go(x, y), s))$$

$$Poss(Grab(g), s) \rightarrow Holding(g, Result(Grab(g), s))$$

$$Poss(Release(g), s) \rightarrow \neg Holding(g, Result(Release(g), s))$$



Wumpus World: Axioms

- Possibility axioms: Move, grab, release

$$At(Agent, x, s) \wedge Adjacent(x, y) \rightarrow Poss(Go(x, y), s)$$

$$Gold(g) \wedge At(Agent, x, s) \wedge At(g, x, s) \rightarrow Poss(Grab(g), s)$$

$$Holding(g, s) \rightarrow Poss(Release(g), s)$$

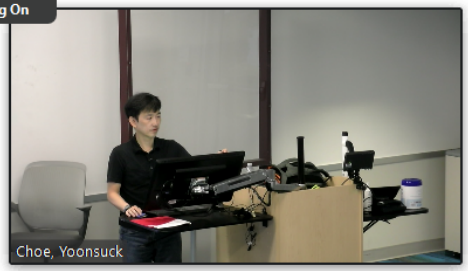
- Effect axioms: Move, Grab, Release

$$Poss(Go(x, y), s) \rightarrow At(Agent, y, Result(Go(x, y), s))$$

$$Poss(Grab(g), s) \rightarrow Holding(g, Result(Grab(g), s))$$

$$Poss(Release(g), s) \rightarrow \neg Holding(g, Result(Release(g), s))$$

In the precondition in the post condition. Right.



Wumpus World: Axioms

- Possibility axioms: Move, grab, release

$$At(Agent, x, s) \wedge Adjacent(x, y) \rightarrow Poss(Go(x, y), s)$$

$$Gold(g) \wedge At(Agent, x, s) \wedge At(g, x, s) \rightarrow Poss(Grab(g), s)$$

$$Holding(g, s) \rightarrow Poss(Release(g), s)$$

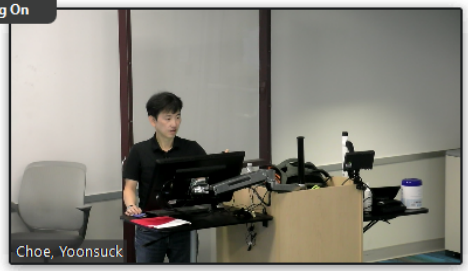
- Effect axioms: Move, Grab, Release

$$Poss(Go(x, y), s) \rightarrow At(Agent, y, Result(Go(x, y), s))$$

$$Poss(Grab(g), s) \rightarrow Holding(g, Result(Grab(g), s))$$

$$Poss(Release(g), s) \rightarrow \neg Holding(g, Result(Release(g), s))$$

What are the results, resulting changes dependent on the action.



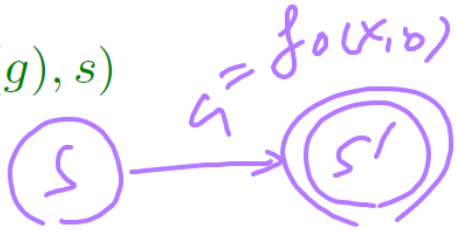
Wumpus World: Axioms

- Possibility axioms: Move, grab, release

$$At(Agent, x, s) \wedge Adjacent(x, y) \rightarrow Poss(Go(x, y(, s)$$

$$Gold(g) \wedge At(Agent, x, s) \wedge At(g, x, s) \rightarrow Poss(Grab(g), s)$$

$$Holding(g, s) \rightarrow Poss(Release(g), s)$$



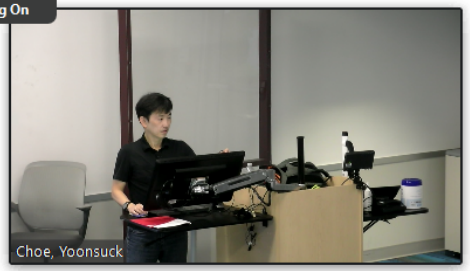
- Effect axioms: Move, Grab, Release

$$Poss(Go(x, y), s) \rightarrow At(Agent, y, Result(Go(x, y), s))$$

$$Poss(Grab(g), s) \rightarrow Holding(g, Result(Grab(g), s))$$

$$Poss(Release(g), s) \rightarrow \neg Holding(g, Result(Release(g), s))$$

this so that makes sense.



Wumpus World: Axioms

- Possibility axioms: Move, grab, release

$$At(Agent, x, s) \wedge Adjacent(x, y) \rightarrow Poss(Go(x, y(, s)$$

$$Gold(g) \wedge At(Agent, x, s) \wedge At(g, x, s) \rightarrow Poss(Grab(g), s)$$

$$Holding(g, s) \rightarrow Poss(Release(g), s)$$

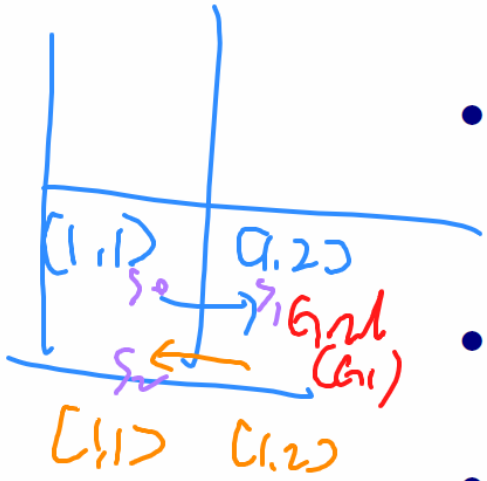
- Effect axioms: Move, Grab, Release

$$Poss(Go(x, y), s) \rightarrow At(Agent, y, Result(Go(x, y), s))$$

$$Poss(Grab(g), s) \rightarrow Holding(g, Result(Grab(g), s))$$

$$Poss(Release(g), s) \rightarrow \neg Holding(g, Result(Release(g), s))$$

And we're precondition of releasing it is your way of holding on to it. So unless you're holding it. You can release it.



Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

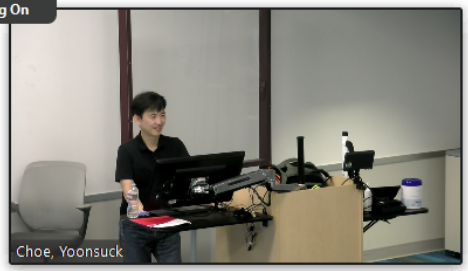
This says moving does not affect the gold when it is not held.

Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

This cannot be deduced properly.

Prolog 3/5

- you can insert
  - write('string')
  - write(X)
  - etc. anywhere in the rule to printout a message or the content of a variable.
  - It can be chained:
    - write('string: '), write(X), write('string2: '), write(Y)
- enter [user]. to begin manual entry:
  - loves(john,jane).
  - loves(jane,john).
  - couple(X,Y) :- loves(X,Y) , loves(Y,X).
  - CTRL-D to end



Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0)$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o, x, s) \wedge (o \neq Agent) \wedge \neg Holding(o, s) \rightarrow At(o, x, Result(Go(y, z), s)).$$

This says moving does not affect the gold when it is not held.

Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

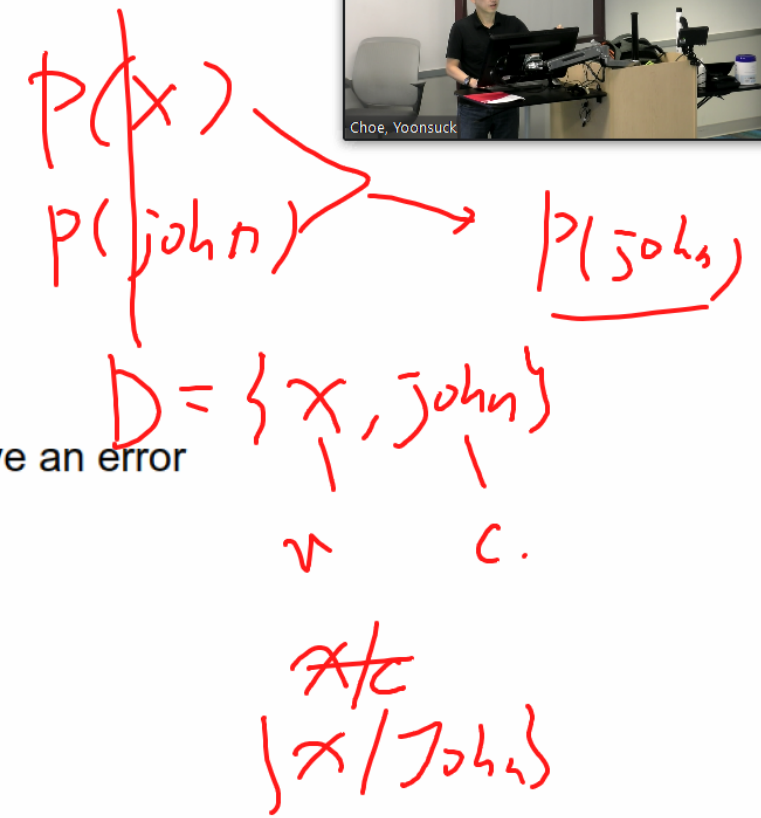
Prolog 3/5

- you can insert
  - write('string')
  - write(X)
  - etc. anywhere in the rule to printout a message or the content of a variable.
  - It can be chained:
    - write('string: '), write(X), write('string2: '), write(Y)
- enter [user]. to begin manual entry:
  - loves(john,jane).
  - loves(jane,john).
  - couple(X,Y) :- loves(X,Y) , loves(Y,X).
  - CTRL-D to end

(1) (2) Facts (3) Rules (4) building the KB.

Prolog 5/5

- You can also test unification
  - At the "?-" prompt
    - p(X) = p(john).
    - p(X,Y) = p(g(Z),Z).
    - p(X) = p(f(X)). <-- this will give an error
    - etc.

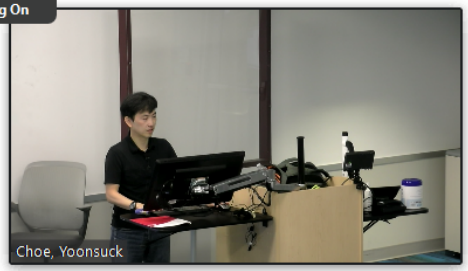


linux2.cse.tamu.edu - PuTTY

```
Action (; for next solution, a for all solutions, RET to stop) ? p
Action (; for next solution, a for all solutions, RET to stop) ?
Action (; for next solution, a for all solutions, RET to stop) ? ;

no
| ?- couple(jane,john).

true ? ;

no
| ?- couple(john,jill).

no
| ?- couple(jane,X).

X = john ? ;

no
| ?- p(X) = p(john).

X = john

yes
| ?-
```

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On



p(x)
p(john)
D = {x, john}

s will give an error

- etc.

You are screen sharing Stop Share

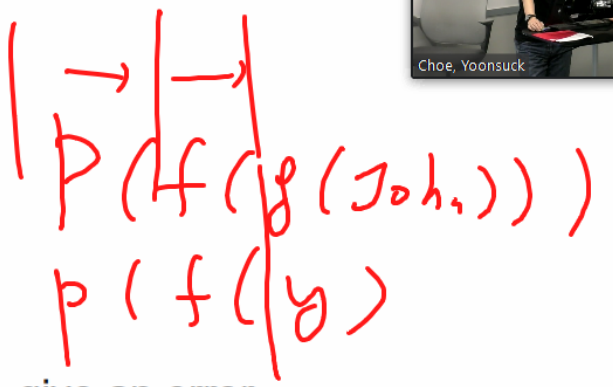
Dan can get mor Mute Stop Video Security Participants 6 Chat 1 Polls New Share Pause Share Annotate Remote Control Apps More

linux2.cse.tamu.edu - PuTTY

```
Action (; for next solution, a for all solutions, RET to stop) ? p
Action (; for next solution, a for all solutions, RET to stop) ?
Action (; for next solution, a for all solutions, RET to stop) ? ;

no
| ?- couple(jane,john).

true ? ;

no
| ?- couple(john,jill).

no
| ?- couple(jane,X).

X = john ? ;

no
| ?- p(X) = p(john).

X = john

yes
| ?- p(f(g(john))) = p(f(y)).
```

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On



s will give an error

- etc.

You are screen sharing Stop Share

Okay, but here yc of john. So you w

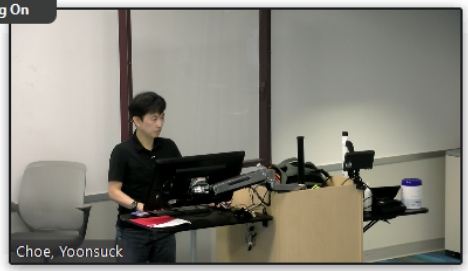
Mute Stop Video Security Participants 6 Chat 1 Polls New Share Pause Share Annotate Remote Control Apps More

linux2.cse.tamu.edu - PuTTY

```
no
| ?- couple(john,jill).

no
| ?- couple(jane,X).

X = john ? ;

no
| ?- p(X) = p(john).

X = john

yes
| ?- p(f(g(john))) = p(f(y)).

no
| ?- p(f(g(john))) = p(f(Y)).

Y = g(john)

yes
| ?-
```

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On



$|\vec{p}|\vec{p}(f(g(John)))$

$p(f(y))$

s will give an error

- etc.

$\theta = \{ y, g(John) \}$

$-- y / g(John)$

You are screen sharing Stop Share

Okay. Mute Stop Video Security Participants 6 Chat 1 Polls New Share Pause Share Annotate Remote Control Apps More

yschoe@brain: ~/teaching/src/prover

```
Run `fwupdmgr get-upgrades` for more information.

You have new mail.
Last login: Thu Sep 22 11:07:58 2022 from 128.194.120.224
yschoe@brain:~$ cd teaching/src/prover/
yschoe@brain:~/teaching/src/prover$ ls
dupe2.lsp  dupe.lsp  prover.lsp
yschoe@brain:~/teaching/src/prover$ gcl
GCL (GNU Common Lisp)  2.6.12 CLtL1    Fri Apr 22 15:51:11 UTC 2016
Source License: LGPL(gcl,gmp), GPL(unexec,bfd,xgcl)
Binary License:  GPL due to GPL'ed components: (XGCL UNEXEC)
Modifications of this banner must retain notice of a compatible license
Dedicated to the memory of W. Schelter

Use (help) to get some basic information on how to use GCL.
Temporary directory for compiler files:
/tmp/

>(load "prover.lsp")
;; Loading "prover.lsp"
;; Loading "dupe2.lsp"

"[1] T
"
"[2] NIL
"
"[3] NIL
" ;; Finished loading "dupe2.lsp"
;; Finished loading "prover.lsp"
T

>*customs*

((1 ((V X) (S X (F X))) ((E X))) (2 ((V Y) (C (F Y))) ((E Y)))
 (3 ((E (A))) NIL) (4 ((D (A))) NIL) (5 ((D Z)) ((S (A) Z)))
 (6 NIL ((D W) (V W))) (7 NIL ((D R) (C R))))

>
```

C#1

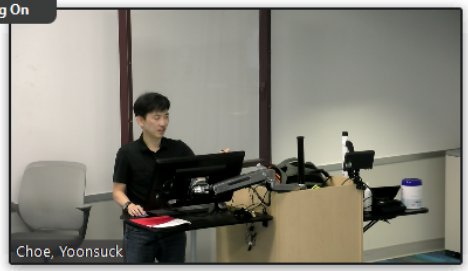
y(x)

S(x, F(x))

¬E(x)

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On



(# (pos) (neg))

s will give an error

Okay, so you don't need to know any of this, I'm just giving you some orientation on how to understand what's.

You are screen sharing Stop Share