Fan Ding
ding0322@umn.edu

Definition from lecture : $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$

1. a. T

when $\alpha \models r$, $M(\alpha) \subseteq M(r)$

That means, for all models that satisfy α will statisfy r

Easy to see, for all models that satisfy $\alpha \wedge \beta$ will statisfy α

Thus, $M(\alpha \wedge \beta) \subseteq M(\alpha) \subseteq M(r)$

$M(\alpha \wedge \beta) \subseteq M(r)$

$\alpha \wedge \beta \models r$

when $\beta \models r$, $M(\beta) \subseteq M(r)$

That means, for all models satisfy β will statisfy r

Easy to see, for all models satisfy $\alpha \wedge \beta$ will statisfy β

Thus, $M(\alpha \wedge \beta) \subseteq M(\beta) \subseteq M(r)$

$M(\alpha \wedge \beta) \subseteq M(r)$

$\alpha \wedge \beta \models r$

when $\alpha \models r$ and $\beta \models r$, apply either one case above will do.

b. F

Counter example : $\alpha = \{A\}$

$\beta = \{A, B\}$

$r = \{A, \neg B\}$

$\beta \vee r = \{A\}$

Explaination : $\alpha \models \beta \vee r$ is obvious, because they are the same

$\alpha \not\models \beta$, because $\exists$ modle has $\neg B$ satisfy α, but not β

$\alpha \not\models r$, because $\exists$ modle has B satisfy α, but not r.

C. T By definition $M(\alpha) \subseteq M(\beta \wedge \gamma)$

For all models satisfy α will satisfy $\beta \wedge \gamma$

To statisfy $\beta \wedge \gamma$, models must statisfy β and γ at the same time

Thus, For all models satisfy α will satisfy β

$$M(\alpha) \subseteq M(\beta)$$

$$\alpha \models \beta$$

the same For all mode satisfy α will satisfy γ

$$M(\alpha) \subseteq M(\gamma)$$

$$\alpha \models \gamma$$

2. See additional Submisson with code & summary

3. def equal (x, y) : $(x < y+1) \wedge (y < x+1)$

part a. def Odd (x) : $\exists a$ equal $(a+a+1, x)$

part b. def prime (x) : $\forall a, b \left((1<a) \wedge (1<b)\right) \Longrightarrow \neg$ equal $(a \times b, x)$

part c. Math conjecture : $\forall x, \exists a, b, c \left(\text{Odd}(x) \wedge (1+1+1+1+1+1+1) < x\right) \Longrightarrow$
equal $(x, (a+b+c)) \wedge$ Odd $(a) \wedge$ Odd $(b) \wedge$ Odd (c)
$\wedge$ prime $(a) \wedge$ prime $(b) \wedge$ prime (c)

4.

part a

α: for all x, there exist y s.t x is greater equal to y

β: there exist y, s.t for all x, x is greater equal to y

part b

α is true. In our domain, no matter what x is, as long as we pick $y=0$. $x \geq y$

β is true. Such y will be 0.

part c

Both.

part d

By definition $a \models b$ iff $\mathcal{M}(a) \subseteq \mathcal{M}(b)$

In our case. $\mathcal{M}(\alpha)$ is actually all possible worlds

Since α is true all the time. And also

$\mathcal{M}(\beta)$ is all possible worlds. too. β is true all the time

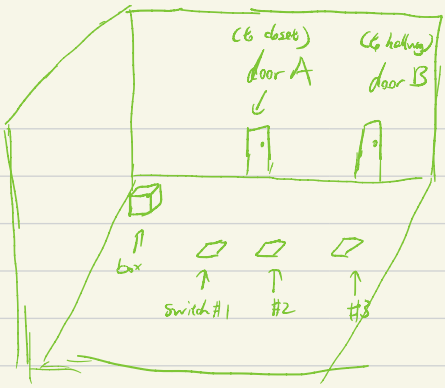
Thus $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$ $\quad \alpha \models \beta$

$\mathcal{M}(\beta) \subseteq \mathcal{M}(\alpha)$ $\quad \beta \models \alpha$

5. Objects: box #1, box #2

room,

door A, door B



part a. $In(box\#1, room) \wedge Room(room)$
$\wedge Block(box\#1) \quad \wedge Block(box\#2)$
$\wedge Door(doorA) \quad \wedge Door(doorB)$

part b. $open(door\ B)$

part c. $getExtraBox(a, b, r)$

precondition: $Block(a) \wedge Block(b) \wedge Room(r) \wedge a \neq b \wedge In(a, r) \wedge \neg In(b, r)$

Effect: $In(b, r)$

$openDoor(a, b, r, d)$

precondition: $Block(a) \wedge Block(b) \wedge Room(r) \wedge Door(d) \wedge a \neq b \wedge \neg open(d) \wedge In(a, r) \wedge In(b, r)$

Effect: $open(d)$

6.a. Uninformed Search & Informed Search

Ways to represent the problem: A state is a 2D-array with all visited position value -1. For next step, the knight usually has 8 options. which means our tree has factor 8. Keep expand the tree until find a way visited 64 squares without visiting the same space twice

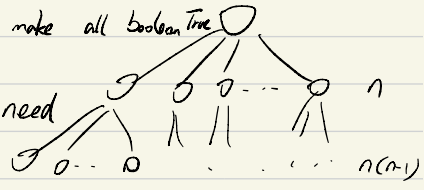
Big O for Uninformed Search & Informed Search are

$O(8^{64})$: the worst case

However, Informed Search is better, because we can use heristic function to let knight goes to "no man's land" first. Go to place with least "-1", or "visited spot" surround. In this case, we explore our tree with "direction" While Uninformed Search just trying exploring all tree without "direction".

b. Constraint Satisfaction & Uninformed Search

ways to represent: Assume will n words, which means n "slots" to contain words. Each slot may have different length. Now We use a n sized array of boolean to represent each slot is perfect fit by a world Success when find a way make all boolean True



If we use Uninformed Search, we will need to try all the possible combination for words which is $O(n!)$

If we use Constraint Satisfaction. We could fill the slots with most connections with other slots. In this case, we reduce our search tree a lot. Assume we can elimate $(n-3)$ cases at each level. That leaves us only 3 branches need to search, which makes $O(3n)$. However, the worst case will still be $O(n!)$. which is we can't eliminate any branches.

6 c Planning & Uninformed Search

Representation: Use 2-D array to present every location in Room, give value -1 if it's blocked, give value 0 if it's not visited. 1 if visited

Planning: Assume we have actions. Like turn 90°, turn 180° turn 270°, move forward, ... N possible actions.
And we average need a moves to visit all locations
$O(N^a)$

Uninformed Search: Do a simple BFS. Assume we have N positions in total. Robot need visit every position $\underbrace{2}_{\text{average}}$ times
$O(2N)$

Uninformed is Better