AIHW3-Berat Barakat

* 1. Poker: Utility based agent, so it can compute the expected utility of all possible outcome states and choose the best one. It has a stochastic, partially observable, discrete, sequential, static(nothing changes during its action) and multi agent environment.
  2. Easy Sudoku’s possible number of states is small enough which can be represented by condition action rules, so a simple reflex agent is appropriate. Hard Sudoku’s possible number of states is huge, so the agent needs to consider what it will be like if it does a specific action, so a goal based agent is appropriate. It has a deterministic, full observable, sequential, discrete, static and single agent environment.
  3. Robot: Utility based agent, so it can consider what it will be like if it does a specific action and compute the expected utility and choose the best one. It has a stochastic, partially observable, continuous, dynamic, sequential and multi agent environment.
  4. Internet shopping: Utility based agent, so it can consider what it will be like if it does a specific action and compute the expected utility and choose the best one. Stochastic (available tickets and prices can change during after its shopping), partial observable (assuming its sensors don’t provide access to all the sellers in the internet), discrete (buys a ticket or not buy, a ticket exists or not exists, but prices are continuous), sequential (if it buys ticket to city A then it must select next ticket accordingly), dynamic (prices and routes can change when it is searching for a solution), single agent (assuming it does not communicate with other buyers) environment.
  5. Go: Utility based agent. Deterministic, full observable, sequential, discrete, static, multi agent.

S0: 012 h=0 f=0

Take s0, not goal, expand s0

S1: 113 h=3 f=3

Take s1, not goal, expand s1

S2: 002 h=2 f=2

S3: 333 h=0 f=0

Take s3, goal, return s3

S0: 012 g=0 h=0 f=0

Take s0, not goal, expand s0

S1: 113 g=1 h=3 f=4

Take s1, not goal, expand s1

S2: 002 g=2 h=2 f=4

S3: 333 g=5 h=0 f=5

Take s2, not goal, expand s2

S4: 22-6 g=3 h=1 f=4

Take s4, not goal, expand s4

S5: 00-6 g=4 h=2 f= 6

S6: 33-6 g=4 h=0 f=4

Take s6, goal, return s6

S0: 012 g=0 h=0 f=0

Take s0, not goal, expand s0

S1: 113 g=1 h=3 f=4

Take s1, not goal, expand s1

S2: 002 g=2 h=2 f=4

S3: 333 g=5 h=0 f=5

Cutoff, move to s2

S2: 002 g=2 h=2 f=4

Take s2, not goal, expand s2

S4:22-6 g=3 h=1 f=4

Take s4, not goal, expand s4

S5: 00-6 g=4 h=2 f= 6

S6: 33-6 g=4 h=0 f=4

Cutoff, move to s6

S6: 33-6 g=4 h=0 f=4

Take s6, goal, return s6

d.

h’(n) is not admissible. For example, in s1 it overestimates the real cost to goal. Also A\* does not find the optimal path because h’(n) is not admissible.

S0: 012 h=0 f=0

Take s0, not goal, expand s0

S1: 113 h=6 f=6

Take s1, not goal, expand s1

S2: 002 h=4 f=4

S3: 333 h=0 f=0

Take s3, goal, return s3

e.

S0: 012 g=0 h=0 f=0

Take s0, not goal, expand s0

S1: 113 g=1 h=6 f=7

Take s1, not goal, expand s1

S2: 002 g=2 h=4 f=6

S3: 333 g=5 h=0 f=5

Cutoff, move to s3

S3: 333 g=5 h=0 f=5

Take s3, goal, return s3

1. A B C D maximize

A1 ----- B1 ---- C1 ----- D1 --- (8 2 5 4)

C2 ----- D1 --- (0 1 2 3)

B2 ---- C1 ----- D1 --- (4 3 9 3)

C2 ----- D1 --- (4 0 10 2)

B3 ---- C1 ----- D1 --- (4 1 5 1)

A2 ----- B1 ---- C1 ----- D1 --- (3 2 12 1)

C2 ----- D1 --- (0 0 2 1)

B2 ---- C1 ----- D1 --- (4 3 9 0)

C2 ----- D1 --- (8 0 10 0)

A3 ----- B1 ---- C1 ----- D1 --- (7 1 0 2)

B2 ---- C1 ----- D1 --- (4 0 0 2)

A4 ----- B1 ---- C1 ----- D1 --- (10 5 0 1)

B2 ---- C1 ----- D1 --- (0 10 0 1)

A5 ----- B1 ---- C1 ----- D1 --- (6 0 0 1)

B and C against A paranoid

A1 ----- B1 ---- C1 ----- D1 --- (8 2 5 4)

C2 ----- D1 --- (0 1 2 3)

B2 ---- C1 ----- D1 --- (4 3 9 3)

C2 ----- D1 --- (4 0 10 2)

B3 ---- C1 ----- D1 --- (4 1 5 1)

A2 ----- B1 ---- C1 ----- D1 --- (3 2 12 1)

C2 ----- D1 --- (0 0 2 1)

B2 ---- C1 ----- D1 --- (4 3 9 0)

C2 ----- D1 --- (8 0 10 0)

A3 ----- B1 ---- C1 ----- D1 --- (7 1 0 2)

B2 ---- C1 ----- D1 --- (4 0 0 2)

A4 ----- B1 ---- C1 ----- D1 --- (10 5 0 1)

B2 ---- C1 ----- D1 --- (0 10 0 1)

A5 ----- B1 ---- C1 ----- D1 --- (6 0 0 1)

B and C agree

A1 ----- B1 ---- C1 ----- D1 --- (8 2 5 4)

C2 ----- D1 --- (0 1 2 3)

B2 ---- C1 ----- D1 --- (4 3 9 3)

C2 ----- D1 --- (4 0 10 2)

B3 ---- C1 ----- D1 --- (4 1 5 1)

A2 ----- B1 ---- C1 ----- D1 --- (3 2 12 1)

C2 ----- D1 --- (0 0 2 1)

B2 ---- C1 ----- D1 --- (4 3 9 0)

C2 ----- D1 --- (8 0 10 0)

A3 ----- B1 ---- C1 ----- D1 --- (7 1 0 2)

B2 ---- C1 ----- D1 --- (4 0 0 2)

A4 ----- B1 ---- C1 ----- D1 --- (10 5 0 1)

B2 ---- C1 ----- D1 --- (0 10 0 1)

A5 ----- B1 ---- C1 ----- D1 --- (6 0 0 1)

A and C maximize their summation

A1 ----- B1 ---- C1 ----- D1 --- (8 2 5 4)

C2 ----- D1 --- (0 1 2 3)

B2 ---- C1 ----- D1 --- (4 3 9 3)

C2 ----- D1 --- (4 0 10 2)

B3 ---- C1 ----- D1 --- (4 1 5 1)

A2 ----- B1 ---- C1 ----- D1 --- (3 2 12 1)

C2 ----- D1 --- (0 0 2 1)

B2 ---- C1 ----- D1 --- (4 3 9 0)

C2 ----- D1 --- (8 0 10 0)

A3 ----- B1 ---- C1 ----- D1 --- (7 1 0 2)

B2 ---- C1 ----- D1 --- (4 0 0 2)

A4 ----- B1 ---- C1 ----- D1 --- (10 5 0 1)

B2 ---- C1 ----- D1 --- (0 10 0 1)

A5 ----- B1 ---- C1 ----- D1 --- (6 0 0 1)

A and B agree

A1 ----- B1 ---- C1 ----- D1 --- (8 2 5 4)

C2 ----- D1 --- (0 1 2 3)

B2 ---- C1 ----- D1 --- (4 3 9 3)

C2 ----- D1 --- (4 0 10 2)

B3 ---- C1 ----- D1 --- (4 1 5 1)

A2 ----- B1 ---- C1 ----- D1 --- (3 2 12 1)

C2 ----- D1 --- (0 0 2 1)

B2 ---- C1 ----- D1 --- (4 3 9 0)

C2 ----- D1 --- (8 0 10 0)

A3 ----- B1 ---- C1 ----- D1 --- (7 1 0 2)

B2 ---- C1 ----- D1 --- (4 0 0 2)

A4 ----- B1 ---- C1 ----- D1 --- (10 5 0 1)

B2 ---- C1 ----- D1 --- (0 10 0 1)

A5 ----- B1 ---- C1 ----- D1 --- (6 0 0 1)

* 1. Assuming scores ϵ {1, 2, 3, …16}. (9-) means 9 or less. (9+) means 9 or more

Max min max min

11A----11B----11A-------11B---11

B---12

A-------9-B---9

B---10pruned

B---15+A-------15B---15

B---16

A--------- B---13pruned

B---14pruned

A----- 3- B------3A-------3-B---3

B---4pruned

A-------1-B---1

B---2pruned

B--------A-------- B--7pruned

B---8pruned

A---------B---5pruned

B---6pruned

Max min max min

-1A------ -1B------ -1A----- -1B--- -1

B---pruned

A------ -1B--- -1

B---pruned

B-------- A------ B---pruned

B---pruned

A------- B---pruned

B---pruned

A------- -1B------ -1A------- -1B--- -1

B--- pruned

A------- -1B--- -1

B---pruned

B--------A-------- B---pruned

B---pruned

A---------B---pruned

B---pruned

Or another scenario

Max min max min

1A------ 1B------ 1A----- 1B--- 1

B--- 1

A------ B--- pruned

B---pruned

B------- 1A------ 1B---1

B---1

A------- B---pruned

B---pruned

A------- B------ A------- B---pruned

B--- pruned

A------- B--- pruned

B---pruned

B--------A-------- B---pruned

B---pruned

A---------B---pruned

B---pruned

5.

a) (not A and not B and not C and not D and not E) or (A and B and C and D and E)

32 models

All T and all F, 2 models are satisfiable

30 models are unsatisfiable

b) (A or B or C or D or E) and (not A or not B or not C or not D or not E)

32 models

All T and all F, 2 models are unsatisfiable

30 models are satisfiable

c) (A or B and (D or not A) and (E or A) -> (B or C and (not D or E)) and (A and not A)

32 models

(A and not A) is unsatisfiable and so the whole sentence is unsatisfiable

d) (A and (A -> B) and (B -> C)) -> not C

8 models

All T, 1 model is unsatisfiable

7 models are satisfiable

e) not ((A and (A -> B) and (B -> C)) -> C)

8 models

8 models are unsatisfiable

f) (A -> not A) and (not A -> A)

2 models

2 models are unsatisfiable

g) (A and (A -> B) and (B -> C)) -> C

8 models

Tautology

6.

A) Effect and frame axioms for traverse(e) action

As we don't do noop, its axioms are not shown

Since there is only one type of key and lock "fifi", it is taken as a constant K

for all s,e,u,v

Loc(u,s) and Edge(e,u,v) and not Lock(v,K,s)=>Loc(v,Result(traverse(e),s))

for all s,e,u,v

Loc(u,s) and Edge(e,u,v) and Lock(v,K,s) and Carrying(K,s)=>Loc(v,Result(traverse(e),s))

for all v,s,e

Loc(v,s) and Carrying (K,s) and Lock(v,K,s)=>not Lock(v,K,Result(traverse(e),s)

for all v,s,e

Loc(v,s) and Carrying (K,s) and Lock(v,K,s)=>not Carrying(v,K,Result(traverse(e),s)

for all v,s,e

Loc(v,s) and Key(v,K,s) => Carrying(K,Result(traverse(e),s)

for all v,s,e

Carrying (K,s)=>Carrying (K,Result(traverse(e),s)

for all v,s,e

not Carrying (K,s)=> not Carrying(K,Result(traverse(e),s)

for all v,s,e

Key(v,K,s)=>Key(v,K,Result(traverse(e),s))

for all v,s,e

Lock(v,K,s)=>Lock(v,K,Result(traverse(e),s))

for all v,s,e

not Lock(v,K,s)=>not Lock(v,K,Result(traverse(e),s))

B)Prove

a)CNF:

Remove quantifiers:

Loc(u,s) and Edge(e,u,v) and not Lock(v,K,s)=>Loc(v,Result(traverse(e),s))

Loc(u,s) and Edge(e,u,v) and Lock(v,K,s) and Carrying(K,s)=>Loc(v,Result(traverse(e),s))

Loc(v,s) and Carrying (K,s) and Lock(v,K,s)=>not Lock(v,K,Result(traverse(e),s)

Loc(v,s) and Carrying (K,s) and Lock(v,k,s)=>not Carrying(v,K,Result(traverse(e),s)

Loc(v,s) and Key(v,K,s) => Carrying(K,Result(traverse(e),s)

Carrying (K,s)=>Carrying (K,Result(traverse(e),s)

not Carrying (K,s)=> not Carrying(K,Result(traverse(e),s)

Key(v,K,s)=>Key(v,K,Result(traverse(e),s))

Lock(v,K,s)=>Lock(v,K,Result(traverse(e),s))

not Lock(v,K,s)=>not Lock(v,K,Result(traverse(e),s))

Remove implications and negations:

1. not Loc(u,s) or not Edge(e,u,v) or Lock(v,K,s) or Loc(v,Result(traverse(e),s))

2. not Loc(u,s) or not Edge(e,u,v) or not Lock(v,K,s) or not Carrying(K,s) or Loc(v,Result(traverse(e),s))

3. not Loc(v,s) or not Carrying (K,s) or not Lock(v,K,s) or not Lock(v,K,Result(traverse(e),s)

4. not Loc(v,s) or not Carrying (K,s) or not Lock(v,K,s) or not Carrying(v,K,Result(traverse(e),s)

5. not Loc(v,s) or not Key(v,K,s) or Carrying(K,Result(traverse(e),s)

6. not Carrying (K,s) or Carrying (K,Result(traverse(e),s)

7. Carrying (K,s) or not Carrying(K,Result(traverse(e),s)

8. not Key(v,K,s) or Key(v,K,Result(traverse(e),s))

9. not Lock(v,K,s) or Lock(v,K,Result(traverse(e),s))

10.Lock(v,K,s) or not Lock(v,K,Result(traverse(e),s))

Facts: All edges should be represented symmetrically however for simplicity we show only the needed ones

11. Edge(E1, V0, V1)

12. Edge(E1, V1, V0)

13. Edge(E2, V0, V2)

14. Edge(E3, V2, G)

15. Goal(G)

S0 fluents: Since there is only one type of key and lock "fifi", it is taken as a constant K.

The frame facts which will not be used are not shown for simplicity.

16. Loc(V0, S0)

17. Lock(V2, K, S0)

18. not Lock(V0,K,S0)

19. not Lock (V1,K,S0)

20. not Lock (G,K,S0)

21. Key(V1, K, S0)

b) Theorem:

exists v [Goal(v) and Loc(v,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1), S0)))))]

Negation:

not exist v [Goal(v) and Loc(v,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1), S0)))))]

0. not Goal(v) or not Loc(v,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1), S0)))))

c) Resolution:

1-16 {u|V0,s|S0}

1. not Loc(u,s) or not Edge(e,u,v) or Lock(v,K,s) or Loc(v,Result(traverse(e),s))

16. Loc(V0, S0)

22. not Edge(e,V0,v) or Lock(v,K,S0) or Loc(v,Result(traverse(e),S0))

22-11 {e|E1,v|V1}

22. not Edge(e,V0,v) or Lock(v,K,S0) or Loc(v,Result(traverse(e),S0))

11. Edge(E1, V0, V1)

23. Lock(V1,K,S0) or Loc(V1,Result(traverse(E1),S0))

23-19

23. Lock(V1,K,S0) or Loc(V1,Result(traverse(E1),S0))

19. not Lock (V1,K,S0)

\*24. Loc(V1,Result(traverse(E1),S0))

24-5 {v|V1, s|Result(traverse(E1),S0)}

24. Loc(V1,Result(traverse(E1),S0))

5. not Loc(v,s) or not Key(v,K,s) or Carrying(K,Result(traverse(e),s)

\*25. not Key(V1,K,Result(traverse(E1),S0)) or Carrying(K,Result(traverse(e),Result(traverse(E1),S0))

8-21 {v|V1,s|S0|

8. not Key(v,K,s) or Key(v,K,Result(traverse(e),s))

21. Key(V1, K, S0)

26. Key(V1,K,Result(traverse(e),S0))

25-26 {e|E1}

25. not Key(V1,K,Result(traverse(E1),S0)) or Carrying(K,Result(traverse(e),Result(traverse(E1),S0))

26. Key(V1,K,Result(traverse(e),S0))

\*27. Carrying(K,Result(traverse(E1),Result(traverse(E1),S0))

24-1 {u|V1, s|Result(traverse(E1),S0))

24. Loc(V1,Result(traverse(E1),S0))

1. not Loc(u,s) or not Edge(e,u,v) or Lock(v,K,s) or Loc(v,Result(traverse(e),s))

28. not Edge(e,V1,v) or Lock(v,K,Result(traverse(E1),S0)) or Loc(v,Result(traverse(e),Result(traverse(E1),S0)))

28-12 {e|E1,v|V0}

12. Edge(E1, V1, V0)

28. not Edge(e,V1,v) or Lock(v,K,Result(traverse(E1),S0)) or Loc(v,Result(traverse(e),Result(traverse(E1),S0)))

\*29. Lock(V0,K,Result(traverse(E1),S0)) or Loc(V0,Result(traverse(E1),Result(traverse(E1),S0)))

18-10 {v|V0, s|S0)

10.Lock(v,K,s) or not Lock(v,K,Result(traverse(e),s))

18. not Lock(V0,K,S0)

30. not Lock(V0,K,Result(traverse(e),S0))

29-30 {e|E1}

29. Lock(V0,K,Result(traverse(E1),S0)) or Loc(V0,Result(traverse(E1),Result(traverse(E1),S0)))

30. not Lock(V0,K,Result(traverse(e),S0))

31. Loc(V0,Result(traverse(E1),Result(traverse(E1),S0)))

31-2 {u|V0, s|Result(traverse(E1),Result(traverse(E1),S0))}

2. not Loc(u,s) or not Edge(e,u,v) or not Lock(v,K,s) or not Carrying(K,s) or Loc(v,Result(traverse(e),s))

31. Loc(V0,Result(traverse(E1),Result(traverse(E1),S0)))

32. not Edge(e,V0,v) or not Lock(v,K,Result(traverse(E1),Result(traverse(E1),S0)))

or not Carrying(K,Result(traverse(E1),Result(traverse(E1),S0)))

or Loc(v,Result(traverse(e),Result(traverse(E1),Result(traverse(E1),S0))))

32-13{e|E2, v|V2}

13. Edge(E2, V0, V2)

32. not Edge(e,V0,v) or not Lock(v,K,Result(traverse(E1),Result(traverse(E1),S0)))

or not Carrying(K,Result(traverse(E1),Result(traverse(E1),S0)))

or Loc(v,Result(traverse(e),Result(traverse(E1),Result(traverse(E1),S0))))

33. not Lock(V2,K,Result(traverse(E1),Result(traverse(E1),S0)))

or not Carrying(K,Result(traverse(E1),Result(traverse(E1),S0)))

or Loc(V2,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

33-27

27. Carrying(K,Result(traverse(E1),Result(traverse(E1),S0))

33. not Lock(V2,K,Result(traverse(E1),Result(traverse(E1),S0)))

or not Carrying(K,Result(traverse(E1),Result(traverse(E1),S0)))

or Loc(V2,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

\*34. not Lock(V2,K,Result(traverse(E1),Result(traverse(E1),S0)))

or Loc(V2,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

17-9 {v|V2,s|S0}

17. Lock(V2, K, S0)

9. not Lock(v,K,s) or Lock(v,K,Result(traverse(e),s))

35. Lock(V2,K,Result(traverse(e),S0))

35-9 {v|V2, s|Result(traverse(e),S0)}

9. not Lock(v,K,s) or Lock(v,K,Result(traverse(e),s))

35. Lock(V2,K,Result(traverse(e),S0))

36. Lock(v,K,Result(traverse(e),Result(traverse(e),S0)))

34-36 {v|V2, e|E1}

34. not Lock(V2,K,Result(traverse(E1),Result(traverse(E1),S0)))

or Loc(V2,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

36. Lock(v,K,Result(traverse(e),Result(traverse(e),S0)))

37. Loc(V2,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

37-1 {u|V2,s|Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))}

37. Loc(V2,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

1. not Loc(u,s) or not Edge(e,u,v) or Lock(v,K,s) or Loc(v,Result(traverse(e),s))

38. not Edge(e,V2,v) or Lock(v,K,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

or Loc(v,Result(traverse(e),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

38-14 {e|E3, v|G}

38. not Edge(e,V2,v) or Lock(v,K,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

or Loc(v,Result(traverse(e),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

14. Edge(E3, V2, G)

\*39. Lock(G,K,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

or Loc(G,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

10-20 {v|G, s|S0}

10.Lock(v,K,s) or not Lock(v,K,Result(traverse(e1),s))

20. not Lock (G,K,S0)

40. not Lock(G,K,Result(traverse(e1),S0))

40-10 {v|G, s|Result(traverse(e1),S0)}

40. not Lock(G,K,Result(traverse(e1),S0))

10.Lock(v,K,s) or not Lock(v,K,Result(traverse(e2),s))

41. not Lock(G,K,Result(traverse(e2),Result(traverse(e1),S0)))

41-10 {v|G, s|Result(traverse(e2),Result(traverse(e1),S0))}

41. not Lock(G,K,Result(traverse(e2),Result(traverse(e1),S0)))

10.Lock(v,K,s) or not Lock(v,K,Result(traverse(e3),s))

42. not Lock(v,K,Result(traverse(e3),Result(traverse(e2),Result(traverse(e1),S0))))

42-39 {v|G,e3|E2,e2|E1,e1|E1}

\*39. Lock(G,K,Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0))))

or Loc(G,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

42. not Lock(v,K,Result(traverse(e3),Result(traverse(e2),Result(traverse(e1),S0))))

\*43. Loc(G,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

0-15 {v|G}

0. not Goal(v) or not Loc(v,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1), S0)))))

15. Goal(G)

44. not Loc(G,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1), S0)))))

43-44

\*43. Loc(G,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

44. not Loc(G,Result(traverse(E3),Result(traverse(E2),Result(traverse(E1),Result(traverse(E1),S0)))))

45. Contradiction

C)Without frame axioms we won't be able to show that for example the not locked nodes are still

not locked after the first and the further actions. As we used 40,41 and 42 to show that the goal node is still not locked after the third traverse(e) action.

D)Since all the used axioms are in horn form, yes we can prove using only forward chaining.