CSCE 625: ARTIFICIAL INTELLIGENCE: HOMEWORK 2

- ANIKET SANJIV BONDE

1A)

Propositions:

L1W, L1Y, L1B, L2W, L2Y, L2B, L3W, L3Y, L3B,

C1W, C1Y, C1B, C2W, C2Y, C2B, C3W, C3Y, C3B,

O1W, O1Y, O2W, O2Y, O3W, O3Y

Knowledge Base:

These constraints ensure that each label (viz. W, Y, B) are only given to single distinct box:

L1W <-> ¬L2W ^ ¬L3W

L1Y <-> ¬L2Y ^ ¬L3Y

L1B <-> ¬L2B ^ ¬L3B

L2W <-> ¬L3W ^ ¬L1W

L2Y <-> ¬L3Y ^ ¬L1Y

L2B <-> ¬L3B ^ ¬L1B

L3W <-> ¬L2W ^ ¬L1W

L3Y <-> ¬L2Y ^ ¬L1Y

L3B <-> ¬L2B ^ ¬L1B

These constraints ensure that each correct label (viz. W, Y, B) are only given to single distinct box:

C1W <-> ¬C2W ^ ¬C3W

C1Y <-> ¬C2Y ^ ¬C3Y

C1B <-> ¬C2B ^ ¬C3B

C2W <-> ¬C3W ^ ¬C1W

C2Y <-> ¬C3Y ^ ¬C1Y

C2B <-> ¬C3B ^ ¬C1B

C3W <-> ¬C2W ^ ¬C1W

C3Y <-> ¬C2Y ^ ¬C1Y

C3B <-> ¬C2B ^ ¬C1B

These constraints ensure that boxes are incorrectly labelled.

L1W <-> ¬C1W

L2W <-> ¬C2W

L3W <-> ¬C3W

L1Y <-> ¬C1Y

L2Y <-> ¬C2Y

L3Y <-> ¬C3Y

L1B <-> ¬C1B

L2B <-> ¬C2B

L3B <-> ¬C3B

These constraints ensure that correct colored balls are picked from correct labelled boxes:

O1W -> C1W ∨ C1B

O2W -> C2W ∨ C2B

O3W -> C3W ∨ C3B

O1Y -> C1Y ∨ C1B

O2Y -> C2Y ∨ C2B

O3Y -> C3Y ∨ C3B

These (trivial) constraints ensure that each picked ball is of a color:

O1W <-> ¬O1Y

O2W <-> ¬O2Y

O3W <-> ¬O3Y

These constraints ensure that each label (viz. W, Y, B) must be present amongst 3 boxes:

L1W ∨ L2W ∨ L3W

L1Y ∨ L2Y ∨ L3Y

L1B ∨ L2B ∨ L3B

These constraints ensure that each label (viz. W, Y, B) must be present amongst 3 boxes:

C1W ∨ C2W ∨ C3W

C1Y ∨ C2Y ∨ C3Y

C1B ∨ C2B ∨ C3B

These constraints ensure that each box must be labelled:

C1W ∨ C1Y ∨ C1B

C2W ∨ C2Y ∨ C2B

C3W ∨ C3Y ∨ C3B

L1W ∨ L1Y ∨ L1B

L2W ∨ L2Y ∨ L2B

L3W ∨ L3Y ∨ L3B

These constraints ensure that balls picked from boxes must have at least have ‘W’ or ‘Y’ color:

O1W ∨ O2W ∨ O3W

O1Y ∨ O2Y ∨ O3Y

1B)

To prove: Knowledge Base entails C2W by Natural Deduction.

Proof:

Knowledge Base:

1) O1Y

2) O2W

3) O3Y

4) L1W

5) L2Y

6) L3B

Relevant constraints Knowledge Base

7) C2W ⇒ ¬C2Y ^ ¬C2B

8) C2W ⇒ ¬L2W

9) O1Y ⇒ C1Y ∨ C1B

10) O2W ⇒ C2W ∨ C2B

11) O2Y ⇒ C2Y ∨ C2B

12) C2W ∨ C2Y ∨ C2B

13) O3Y ⇒ C3Y ∨ C3B

14) L3B ⇒ ¬C3B

15) C3Y ⇒ ¬C2Y ^ ¬C1Y

16) C1B ^ C3Y ⇒ C2W

By applying Modus Ponens to rule no. 10 and 2,

17) C2W ∨ C2B

By applying Modus Ponens to rule no. 9 and 1,

18) C1Y ∨ C1B

By applying Modus Ponens to rule no. 13 and 3,

19) C3Y ∨ C3B

By applying Modus Ponens to rule no. 14 and 6,

20) ¬C3B

By applying $(\neg\alpha \vee \beta) \equiv (\alpha \Rightarrow \beta)$ to rule no. 19,

21) $\neg C3B \Rightarrow C3Y$

By applying Modus Ponens to rule no. 20 and 21,

22) C3Y

By applying Modus Ponens to rule no. 15 and 22,

23) $\neg C2Y \wedge \neg C1Y$

By rule no. 23, these rules can be inferred:

24) $\neg C2Y$
25) $\neg C1Y$

By applying $(\neg\alpha \vee \beta) \equiv (\alpha \Rightarrow \beta)$ to rule no. 18,

26) $\neg C1Y \Rightarrow C1B$

By applying Modus to rule no. 25 and 26,

27) C1B

By rules 16, 22 and 27,

28) C2W

1C)

To prove: Knowledge Base entails C2W by Resolution refutation

Proof:

Knowledge Base:

1) O1Y
2) O2W
3) O3Y
4) L1W
5) L2Y
6) L3B

Relevant constraints KB:

Converting O2W ⇒ C2W ∨ C2B to Conjunctive Normal Form, we get:

7) ¬O2W ∨ C2W ∨ C2B

Converting O1Y ⇒ C1Y ∨ C1B to Conjunctive Normal Form, we get:

8) ¬O1Y ∨ C1Y ∨ C1B

Converting O3Y ⇒ C3Y ∨ C3B to Conjunctive Normal Form, we get:

9) ¬O3Y ∨ C3Y ∨ C3B

Converting L3B ⇒ ¬C3B to Conjunctive Normal Form, we get:

10) ¬L3B ∨ ¬C3B

Converting C3Y ⇒ ¬C1Y ^ ¬C2Y to Conjunctive Normal Form, we get:

11) (¬C3Y ∨ ¬C1Y) ^ (¬C3Y ∨ ¬C2Y)

Converting C1B ^ C3Y ⇒ C2W to Conjunctive Normal Form, we get:

12) ¬C1B ∨ ¬C3Y ∨ C2W

By rule no. 11, these can be inferred:

13) ¬C3Y ∨ ¬C1Y
14) ¬C3Y ∨ ¬C2Y

Applying resolution rule to rule no. 7 and 2:

15) C2W ∨ C2B

Applying resolution rule to rule no. 1 and 8:

16) C1Y ∨ C1B

Applying resolution rule to rule no. 3 and 9:

17) C3Y ∨ C3B

Applying resolution rule to rule no. 6 and 10:

18) ¬C3B

Applying resolution rule to rule no. 17 and 18:

19) C3Y

Applying resolution rule to rule no. 13 and 19:

20) ¬C1Y

Applying resolution rule to rule no. 14 and 19:

21) ¬C2Y

Applying resolution rule to rule no. 8 and 20:

22) ¬O1Y ∨ C1B

Applying resolution rule to rule no. 1 and 22:

23) C1B

Applying resolution rule to rule no. 12 and 23:

24) ¬C3Y ∨ C2W

Applying resolution rule to rule no. 19 and 24:

25) C2W

Now, we have to prove that (KB ∧ ¬C2W) is un-satisfiable. So, by rule no. 25 and the previous fact:

26) C2W ∧ ¬C2W is **FALSE,** hence this is an empty clause.

As we got an empty clause, its proved that Knowledge Base entails C2W.

Q2)

Propositions:

QA1, QA2, QA3, QA4, QB1, QB2, QB3, QB4, QC1, QC2, QC3, QC4, QD1, QD2, QD3, QD4

Knowledge Base =

These constraints ensure that only one queen is placed in a column:

1) (QA1∨ QA2∨ QA3∨ QA4)
2) (QB1∨ QB2∨ QB3∨ QB4)
3) (QC1∨ QC2∨ QC3∨ QC4)
4) (QD1∨ QD2∨ QD3∨ QD4)

These constraints rule out the corresponding affected boxes in rows and columns, when a queen is placed:

5) QA1 -> (¬QA2 ^ ¬QA3 ^ ¬QA4 ^ ¬QB1 ^ ¬QC1 ^ ¬QD1)
   ≡ (¬QA1 ∨ ¬QA2) ^ (¬QA1 ∨ ¬QA3) ^ (¬QA1 ∨ ¬QA4) ^ (¬QA1 ∨ ¬QB1) ^ (¬QA1 ∨ ¬QC1) ^ (¬QA1 ∨ ¬QD1)
6) QA2 -> (¬QA1 ^ ¬QA3 ^ ¬QA4 ^ ¬QB2 ^ ¬QC2 ^ ¬QD2)
   ≡ (¬QA2 ∨ ¬QA1) ^ (¬QA2 ∨ ¬QA3) ^ (¬QA2 ∨ ¬QA4) ^ (¬QA2 ∨ ¬QB2) ^ (¬QA2 ∨ ¬QC2) ^ (¬QA2 ∨ ¬QD2)
7) QA3 -> (¬QA1 ^ ¬QA2 ^ ¬QA4 ^ ¬QB3 ^ ¬QC3 ^ ¬QD3)
   ≡ (¬QA3 ∨ ¬QA1) ^ (¬QA3 ∨ ¬QA2) ^ (¬QA3 ∨ ¬QA4) ^ (¬QA3 ∨ ¬QB3) ^ (¬QA3 ∨ ¬QC3) ^ (¬QA3 ∨ ¬QD3)
8) QA4 -> (¬QA1 ^ ¬QA2 ^ ¬QA3 ^ ¬QB4 ^ ¬QC4 ^ ¬QD4)
   ≡ (¬QA4 ∨ ¬QA1) ^ (¬QA4 ∨ ¬QA2) ^ (¬QA4 ∨ ¬QA3) ^ (¬QA4 ∨ ¬QB4) ^ (¬QA4 ∨ ¬QC4) ^ (¬QA4 ∨ ¬QD4)
9) QB1 -> (¬QB2 ^ ¬QB3 ^ ¬QB4 ^ ¬QA1 ^ ¬QC1 ^ ¬QD1)
   ≡ (¬QB1 ∨ ¬QB2) ^ (¬QB1 ∨ ¬QB3) ^ (¬QB1 ∨ ¬QB4) ^ (¬QB1 ∨ ¬QA1) ^ (¬QB1 ∨ ¬QC1) ^ (¬QB1 ∨ ¬QD1)
10) QB2 -> (¬QB1 ^ ¬QB3 ^ ¬QB4 ^ ¬QA2 ^ ¬QC2 ^ ¬QD2)
   ≡ (¬QB2 ∨ ¬QB1) ^ (¬QB2 ∨ ¬QB3) ^ (¬QB2 ∨ ¬QB4) ^ (¬QB2 ∨ ¬QA2) ^ (¬QB2 ∨ ¬QC2) ^ (¬QB2 ∨ ¬QD2)
11) QB3 -> (¬QB1 ^ ¬QB2 ^ ¬QB4 ^ ¬QA3 ^ ¬QC3 ^ ¬QD3)
   ≡ (¬QB3 ∨ ¬QB1) ^ (¬QB3 ∨ ¬QB2) ^ (¬QB3 ∨ ¬QB4) ^ (¬QB3 ∨ ¬QA3) ^ (¬QB3 ∨ ¬QC3) ^ (¬QB3 ∨ ¬QD3)
12) QB4 -> (¬QB1 ^ ¬QB2 ^ ¬QB3 ^ ¬QA4 ^ ¬QC4 ^ ¬QD4)
   ≡ (¬QB4 ∨ ¬QB1) ^ (¬QB4 ∨ ¬QB2) ^ (¬QB4 ∨ ¬QB3) ^ (¬QB4 ∨ ¬QA4) ^ (¬QB4 ∨ ¬QC4) ^ (¬QB4 ∨ ¬QD4)
13) QC1 -> (¬QC2 ^ ¬QC3 ^ ¬QC4 ^ ¬QA1 ^ ¬QB1 ^ ¬QD1)
   ≡ (¬QC1 ∨ ¬QC2) ^ (¬QC1 ∨ ¬QC3) ^ (¬QC1 ∨ ¬QC4) ^ (¬QC1 ∨ ¬QA1) ^ (¬QC1 ∨ ¬QB1) ^ (¬QC1 ∨ ¬QD1)
14) QC2 -> (¬QC1 ^ ¬QC3 ^ ¬QC4 ^ ¬QA2 ^ ¬QB2 ^ ¬QD2)
   ≡ (¬QC2 ∨ ¬QC1) ^ (¬QC2 ∨ ¬QC3) ^ (¬QC2 ∨ ¬QC4) ^ (¬QC2 ∨ ¬QA2) ^ (¬QC2 ∨ ¬QB2) ^ (¬QC2 ∨ ¬QD2)
15) QC3 -> (¬QC1 ^ ¬QC2 ^ ¬QC4 ^ ¬QA3 ^ ¬QB3 ^ ¬QD3)

≡ (¬QC3 ᵛ ¬QC1) ^ (¬QC3 ᵛ ¬QC2) ^ (¬QC3 ᵛ ¬QC4) ^ (¬QC3 ᵛ ¬QA3) ^ (¬QC3 ᵛ ¬QB3) ^ (¬QC3 ᵛ ¬QD3)

16) QC4 -> (¬QC1 ^ ¬QC2 ^ ¬QC3 ^ ¬QA4 ^ ¬QB4 ^ ¬QD4)

≡ (¬QC4 ᵛ ¬QC1) ^ (¬QC4 ᵛ ¬QC2) ^ (¬QC4 ᵛ ¬QC3) ^ (¬QC4 ᵛ ¬QA4) ^ (¬QC4 ᵛ ¬QB4) ^ (¬QC4 ᵛ ¬QD4)

17) QD1 -> (¬QD2 ^ ¬QD3 ^ ¬QD4 ^ ¬QA1 ^ ¬QB1 ^ ¬QC1)

≡ (¬QD1 ᵛ¬QD2) ^ (¬QD1 ᵛ ¬QD3) ^ (¬QD1 ᵛ ¬QD4) ^ (¬QD1 ᵛ ¬QA1) ^ (¬QD1 ᵛ ¬QB1) ^ (¬QD1 ᵛ ¬QC1)

18) QD2 -> (¬QD1 ^ ¬QD3 ^ ¬QD4 ^ ¬QA2 ^ ¬QB2 ^ ¬QC2)

≡ (¬QD2 ᵛ ¬QD1) ^ (¬QD2 ᵛ ¬QD3) ^ (¬QD2 ᵛ ¬QD4) ^ (¬QD2 ᵛ ¬QA2) ^ (¬QD2 ᵛ ¬QB2) ^ (¬QD2 ᵛ ¬QC2)

19) QD3 -> (¬QD1 ^ ¬QD2 ^ ¬QD4 ^ ¬QA3 ^ ¬QB3 ^ ¬QC3)

≡ (¬QD3 ᵛ¬QD1) ^ (¬QD3 ᵛ ¬QD2) ^ (¬QD3 ᵛ ¬QD4) ^ (¬QD3 ᵛ ¬QA3) ^ (¬QD3 ᵛ ¬QB3) ^ (¬QD3 ᵛ ¬QC3)

20) QD4 -> (¬QD1 ^ ¬QD2 ^ ¬QD3 ^ ¬QA4 ^ ¬QB4 ^ ¬QC4)

≡ (¬QD4 ᵛ ¬QD1) ^ (¬QD4 ᵛ ¬QD2) ^ (¬QD4 ᵛ ¬QD3) ^ (¬QD4 ᵛ ¬QA4) ^ (¬QD4 ᵛ ¬QB4) ^ (¬QD4 ᵛ ¬QC4)

These constraints rules out the corresponding affected boxes in diagonal, when a queen is placed:

21) QA1 -> (¬QB2 ^ ¬QC3 ^ ¬QD4)

≡ (¬QA1 ᵛ ¬QB2) ^ (¬QA1 ᵛ ¬QC3) ^ (¬QA1 ᵛ ¬QD4)

22) QA2 -> (¬QB1 ^ ¬QB3 ^ ¬QC4)

≡ (¬QA2 ᵛ ¬QB1) ^ (¬QA2 ᵛ ¬QB3) ^ (¬QA2 ᵛ ¬QC4)

23) QA3 -> (¬QB2 ^ ¬QB4 ^ ¬QC1)

≡ (¬QA3 ᵛ ¬QB2) ^ (¬QA3 ᵛ ¬QB4) ^ (¬QA3 ᵛ ¬QC1)

24) QA4 -> (¬QB3 ^ ¬QC2 ^ ¬QD1)

≡ (¬QA4 ᵛ ¬QB3) ^ (¬QA4 ᵛ ¬QC2) ^ (¬QA4 ᵛ ¬QD1)

25) QB1 -> (¬QA2 ^ ¬QC2 ^ ¬QD3)

≡ (¬QB1 ᵛ ¬QA2) ^ (¬QB1 ᵛ ¬QC2) ^ (¬QB1 ᵛ ¬QD3)

26) QB2 -> (¬QA1 ^ ¬QA3 ^ ¬QC1 ^ ¬QC3 ^ ¬QD4)

≡ (¬QB2 ᵛ ¬QA1) ^ (¬QB2 ᵛ ¬QA3) ^ (¬QB2 ᵛ ¬QC1) ^ (¬QB2 ᵛ ¬QC3) ^ (¬QB2 ᵛ ¬QD4)

27) QB3 -> (¬QA2 ^ ¬QA4 ^ ¬QC2 ^ ¬QC4 ^ ¬QD1)

≡ (¬QB3 ᵛ ¬QA2) ^ (¬QB3 ᵛ ¬QA4) ^ (¬QB3 ᵛ ¬QC2) ^ (¬QB3 ᵛ ¬QC4) ^ (¬QB3 ᵛ ¬QD1)

28) QB4 -> (¬QA3 ^ ¬QC3 ^ ¬QD2)

≡ (¬QB4 ᵛ ¬QA3) ^ (¬QB4 ᵛ ¬QC3) ^ (¬QB4 ᵛ ¬QD2)

29) QC1 -> (¬QA3 ^ ¬QB2 ^ ¬QD2)

≡ (¬QC1 ᵛ ¬QA3) ^ (¬QC1 ᵛ ¬QB2) ^ (¬QC1 ᵛ ¬QD2)

30) QC2 -> (¬QA4 ^ ¬QB1 ^ ¬QB3 ^ ¬QD1 ^ ¬QD3)

≡ (¬QC2 ˅ ¬QA4) ^ (¬QC2 ˅ ¬QB1) ^ (¬QC2 ˅ ¬QB3) ^ (¬QC2 ˅ ¬QD1) ^ (¬QC2 ˅ ¬QD3)

31) QC3 -> (¬QA1 ^ ¬QB2 ^ ¬QB4 ^ ¬QD2 ^ ¬QD4)

≡ (¬QC3 ˅ ¬QA1) ^ (¬QC3 ˅ ¬QB2) ^ (¬QC3 ˅ ¬QB4) ^ (¬QC3 ˅ ¬QD2) ^ (¬QC3 ˅ ¬QD4)

32) QC4 -> (¬QA2 ^ ¬QB3 ^ ¬QD3)

≡ (¬QC4 ˅ ¬QA2) ^ (¬QC4 ˅ ¬QB3) ^ (¬QC4 ˅ ¬QD3)

33) QD1 -> (¬QA4 ^ ¬QB3 ^ ¬QC2)

≡ (¬QD1 ˅ ¬QA4) ^ (¬QD1 ˅ ¬QB3) ^ (¬QD1 ˅ ¬QC2)

34) QD2 -> (¬QB4 ^ ¬QC1 ^ ¬QC3)

≡ (¬QD2 ˅ ¬QB4) ^ (¬QD2 ˅ ¬QC1) ^ (¬QD2 ˅ ¬QC3)

35) QD3 -> (¬QB1 ^ ¬QC2 ^ ¬QC4)

≡ (¬QD3 ˅ ¬QB1) ^ (¬QD3 ˅ ¬QC2) ^ (¬QD3 ˅ ¬QC4)

36) QD4 -> (¬QA1 ^ ¬QB2 ^ ¬QC3)

≡ (¬QD4 ˅ ¬QA1) ^ (¬QD4 ˅ ¬QB2) ^ (¬QD4 ˅ ¬QC3)

2A)

Using no Heuristic with DPLL:

Consider each iteration to be a backtrack in the given table:

Propositions	**Initial**	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
QA1	?	T (assume)	T	T	F (assume next value)
QA2	?	F by R5	F by R5	F by R5	T (assume)
QA3	?	F by R5	F by R5	F by R5	F by R6
QA4	?	F by R5	F by R5	F by R5	F by R6
QB1	?	F by R5	F by R5	F by R5	F by R22
QB2	?	F by R21	F by R21	F by R21	F by R6
QB3	?	T (assume)	F (assume next value)	F	F by R22
QB4	?	F by R11	T by R2	T by R2	T by R3
QC1	?	F by R5	F by R5	F by R5	T (assume)
QC2	?	F by R27	T (assume)	F (assume next value)	F by R13
QC3	?	F by R11	F by R7 & R28	F by R28	F by R13
QC4	?	BT to QB3 (R27 & R3)	F by R7 & R28	BT to QA1 (R12 & R3)	F by R13
QD1	?	?	F by R5	?	F by R13
QD2	?	?	F by R14	?	F by R28
QD3	?	?	F by R30	?	T (assume)
QD4	?	?	BT to QC2 (R12 & R4)	?	F by R12

Solution:

Thus, we get the final solution as: ('T' indicates queen is present)

F	F	T	F
T	F	F	F
F	F	F	T
F	T	F	F

2B)

Using Pure Symbol and Unit Clause with DPLL:

Propositions	Initial	Iteration 1	Iteration 1	Iteration 2	Iteration 3	Iteration 3
QA1	?	T (assume)	T	T	F (assume next value)	F
QA2	?	F by R5 (unit clause)	F by R5	F by R5	?	T (assume)
QA3	?	F by R5 (unit clause)	F by R5	F by R5	?	F by R6 (unit clause)
QA4	?	F by R5 (unit clause)	F by R5	F by R5	?	F by R6 (unit clause)
QB1	?	F by R5 (unit clause)	F by R5	F by R5	?	F by R22 (unit clause)
QB2	?	F by R21 (unit clause)	F by R21	F by R21	?	F by R6 (unit clause)
QB3	?		T (assume)	F (assume next value)	?	F by R22 (unit clause)
QB4	?		F by R11 (unit clause)	T by R2 (unit clause)	?	T by R2 (unit clause)
QC1	?	F by R5 (unit clause)	F by R5	F by R5	?	T by R3 (unit clause)
QC2	?		F by R27 (unit clause)	T by R3 (unit clause)	?	F by R6 (unit clause)
QC3	?	F by R21 (unit clause)	F by R21	F by R21	?	F by R28 (unit clause)
QC4	?		BT to QB3	F by R12 (unit clause)	?	F by R22 (unit clause)
QD1	?	F by R5 (unit clause)	F by R5	F by R5	?	F by R13 (unit clause)
QD2	?			F by R28 (unit clause)	?	F by R6 (unit clause)
QD3	?		F by R11 (unit clause)	BT to DA1	?	T by R4 (unit clause)
QD4	?	F by R21 (unit clause)	F by R21	F by R21	?	F by R12 (unit clause)

Tracing steps at each iteration:

Iteration 1:

Assumption 1: QA1 = True

No pure symbol.

By using unit clause heuristics, we assign False to QA2, QA3, QA4, QB1, QB2, QC1, QC3, QD1 and QD4.

Assumption 2: QB3 = True

No pure symbol.

By using unit clause heuristics, we assign False to QB4, QC2, QD3

Back track at QC4

Iteration 2:

Assumption 2: QB3 = False

No pure symbol.

By using unit clause heuristics, we assign True to QB4

No pure symbol.

By using unit clause heuristics, we assign False to QC4, QD2

No pure symbol.

By using unit clause heuristics, we assign True to QC2

Solution:

Thus, we get the final solution as: ('T' indicates queen is present)

F	F	T	F
T	F	F	F
F	F	F	T
F	T	F	F

Q3)

Propositions:

X11, X12, X13, X21, X22, X23, X31, X32, X33,

O11, O12, O13, O21, O22, O23, O31, O32, O33,

?11, ?12, ?13, ?21, ?22, ?23, ?31, ?32, ?33,

MoveX11, MoveX12, MoveX13, MoveX21, MoveX22, MoveX23, MoveX31, MoveX32, MoveX33,

CanWinX11, CanWinX12, CanWinX13, CanWinX21, CanWinX22, CanWinX23, CanWinX31, CanWinX32, CanWinX33,

CanWinO11, CanWinO12, CanWinO13, CanWinO21, CanWinO22, CanWinO23, CanWinO31, CanWinO32, CanWinO33,

ForcedMoveX11, ForcedMoveX12, ForcedMoveX13, ForcedMoveX21, ForcedMoveX22, ForcedMoveX23, ForcedMoveX31, ForcedMoveX32, ForcedMoveX33,

Knowledge Base =

This constraint ensures that at a move must be played by ‘X’ when it’s his chance:

MoveX11 ∨ MoveX12 ∨ MoveX13 ∨ MoveX21 ∨ MoveX22 ∨ MoveX23 ∨ MoveX31 ∨ MoveX32 ∨ MoveX33,

These constraints identify the CanWin situations to take actions upon later:

X11 ^ X22 ^ ?33 -> CanWinX33,   O11 ^ O22 ^ ?33 -> CanWinO33,

X11 ^ X33 ^ ?22 -> CanWinX22,   O11 ^ O33 ^ ?22 -> CanWinO22,

X22 ^ X33 ^ ?11 -> CanWinX11,   O22 ^ O33 ^ ?11 -> CanWinO11,

X13 ^ X22 ^ ?31 -> CanWinX31,   O13 ^ O22 ^ ?31 -> CanWinO31,

X13 ^ X31 ^ ?22 -> CanWinX22,   O13 ^ O31 ^ ?22 -> CanWinO22,

X22 ^ X31 ^ ?13 -> CanWinX13,   O22 ^ O31 ^ ?13 -> CanWinO13,

X11 ^ X21 ^ ?31 -> CanWinX31,   O11 ^ O21 ^ ?31 -> CanWinO31,

X11 ^ X31 ^ ?21 -> CanWinX21,   O11 ^ O31 ^ ?21 -> CanWinO21,

X21 ^ X31 ^ ?11 -> CanWinX11,   O21 ^ O31 ^ ?11 -> CanWinO11,

X12 ^ X22 ^ ?32 -> CanWinX32,   O12 ^ O22 ^ ?32 -> CanWinO32, X12 ^ X32 ^ ?22 -> CanWinX22, O12 ^ O32 ^ ?22 -> CanWinO22,

X22 ^ X32 ^ ?12 -> CanWinX12, O22 ^ O32 ^ ?12 -> CanWinO12,

X13 ^ X23 ^ ?33 -> CanWinX33, O13 ^ O23 ^ ?33 -> CanWinO33,

X13 ^ X33 ^ ?23 -> CanWinX23, O13 ^ O33 ^ ?23 -> CanWinO23,

X23 ^ X33 ^ ?13 -> CanWinX13, O23 ^ O33 ^ ?13 -> CanWinO13,

X11 ^ X12 ^ ?13 -> CanWinX13, O11 ^ O12 ^ ?13 -> CanWinO13,

X11 ^ X13 ^ ?12 -> CanWinX12, O11 ^ O13 ^ ?12 -> CanWinO12,

X12 ^ X13 ^ ?11 -> CanWinX11, O12 ^ O13 ^ ?11 -> CanWinO11,

X21 ^ X22 ^ ?13 -> CanWinX23, O21 ^ O22 ^ ?13 -> CanWinO23,

X21 ^ X23 ^ ?12 -> CanWinX22, O21 ^ O23 ^ ?12 -> CanWinO22,

X22 ^ X23 ^ ?11 -> CanWinX21, O22 ^ O23 ^ ?11 -> CanWinO21,

X31 ^ X32 ^ ?13 -> CanWinX33, O31 ^ O32 ^ ?13 -> CanWinO33,

X31 ^ X33 ^ ?12 -> CanWinX32, O31 ^ O33 ^ ?12 -> CanWinO32,

X32 ^ X33 ^ ?11 -> CanWinX31, O32 ^ O33 ^ ?11 -> CanWinO31,

These constraints ensure that when there is a CanWin of ‘O’, ‘X’ should place it at that place forcefully:

CanWinO11 -> ForcedMoveX11,

CanWinO12 -> ForcedMoveX12,

CanWinO13 -> ForcedMoveX13,

CanWinO21 -> ForcedMoveX21,

CanWinO22 -> ForcedMoveX22,

CanWinO23 -> ForcedMoveX23,

CanWinO31 -> ForcedMoveX31,

CanWinO32 -> ForcedMoveX32,

CanWinO33 -> ForcedMoveX33,

These constraints ensure that when there is a situation of winning, 'X' plays that move to win:

CanWinX11 -> MoveX11,

CanWinX12 -> MoveX12,

CanWinX13 -> MoveX13,

CanWinX21 -> MoveX21,

CanWinX22 -> MoveX22,

CanWinX23 -> MoveX23,

CanWinX31 -> MoveX31,

CanWinX32 -> MoveX32,

CanWinX33 -> MoveX33,

These constraints ensure that CanWin is given more priority than ForcedMove:

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX11 -> MoveX11,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX12 -> MoveX12,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX13 -> MoveX13,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX21 -> MoveX21,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX22 -> MoveX22,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX23 -> MoveX23,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX31 -> MoveX31,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX32 -> MoveX32,

¬(CanWinX11 ˅ CanWinX12 ˅ CanWinX13 ˅ CanWinX21 ˅ CanWinX22 ˅ CanWinX23 ˅ CanWinX31 ˅ CanWinX32 ˅ CanWinX33) ^ ForcedMoveX33 -> MoveX33