Introduction to Artificial Intelligence

English practicals 4: (Propositional) logical reasoning

Marika Ivanová

Department of Theoretical Computer Science and Mathematical Logic (KTIML)
Faculty of Mathematics and Physics

March 8th 2022

Solution to assignment # 1

Grid 2D (without diagonals)

Solution to assignment # 1

Grid 2D (without diagonals)

- Euclidean distance $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$ is admissible, but not tight enough

Solution to assignment # 1

Grid 2D (without diagonals)

- Euclidean distance $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$ is admissible, but not tight enough

Grid 3D (without diagonals)

Solution to assignment # 1

Grid 2D (without diagonals)

- Euclidean distance $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$ is admissible, but not tight enough

Grid 3D (without diagonals)

- Exactly the same holds in 3D without diagonals. Just add one dimension

Solution to assignment # 1

Grid 2D diagonal

Solution to assignment # 1

Grid 2D diagonal

- Euclidean distance from origin to (1,1) is $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2} = \sqrt{2} > 1$, thus not admissible

Solution to assignment # 1

Grid 2D diagonal

- Euclidean distance from origin to (1,1) is $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2} = \sqrt{2} > 1$, thus not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|\}$ works here

Solution to assignment # 1

Grid 2D diagonal

- Euclidean distance from origin to (1,1) is $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2} = \sqrt{2} > 1$, thus not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|\}$ works here

Grid 3D all diagonal

Solution to assignment # 1

Grid 2D diagonal

- Euclidean distance from origin to (1,1) is $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2} = \sqrt{2} > 1$, thus not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|\}$ works here

Grid 3D all diagonal

- Again, the same arguments apply for 3D

Solution to assignment # 1

Grid 3D face diagonal

Solution to assignment # 1

Grid 3D face diagonal

- Euclidean distance is not admissible

Solution to assignment # 1

Grid 3D face diagonal

- Euclidean distance is not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|\}$ is admissible, but sometimes not tight enough
- Consider the point (2,2,1), which is in distance 3 from the origin, but the maximum heuristic gives 2

Solution to assignment # 1

Grid 3D face diagonal

- Euclidean distance is not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|\}$ is admissible, but sometimes not tight enough
- Consider the point (2,2,1), which is in distance 3 from the origin, but the maximum heuristic gives 2
- Better would be $(|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|)/2 = 2.5$

Solution to assignment # 1

Grid 3D face diagonal

- Euclidean distance is not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|\}$ is admissible, but sometimes not tight enough
- Consider the point (2,2,1), which is in distance 3 from the origin, but the maximum heuristic gives 2
- Better would be $(|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|)/2 = 2.5$
- But for the point (0,0,2), we get a better estimate by the maximum heuristic

Solution to assignment # 1

Grid 3D face diagonal

- Euclidean distance is not admissible
- Maximum heuristic $\max\{|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|\}$ is admissible, but sometimes not tight enough
- Consider the point (2,2,1), which is in distance 3 from the origin, but the maximum heuristic gives 2
- Better would be $(|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|)/2 = 2.5$
- But for the point (0,0,2), we get a better estimate by the maximum heuristic
- Therefore, always use the tighter one: $\max\{(|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|)/2, |x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|\}$

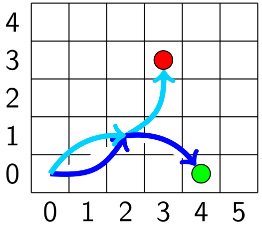
Solution to assignment # 1

Knight

Solution to assignment # 1

Knight

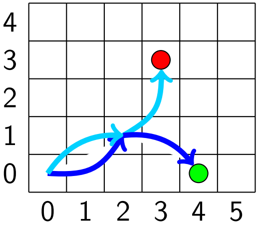
- Consider the two extremes: both positions are in distance 2



Solution to assignment # 1

Knight

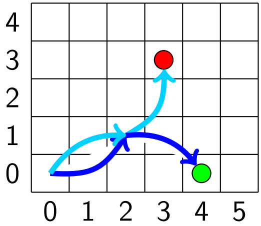
- Consider the two extremes: both positions are in distance 2
- The best heuristic for the green point is $\max\{(|x_1 - x_2|, |y_1 - y_2|)\}/2 = 2$



Solution to assignment # 1

Knight

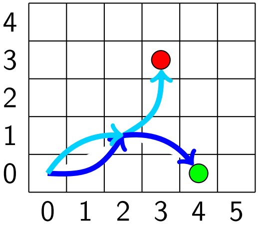
- Consider the two extremes: both positions are in distance 2
- The best heuristic for the green point is $\max\{(|x_1 - x_2|, |y_1 - y_2|)\}/2 = 2$
- For the red one we have $(|x_1 - x_2| + |y_1 - y_2|)/3 = 2$



Knight

- Consider the two extremes: both positions are in distance 2
- The best heuristic for the green point is $\max\{(|x_1 - x_2|, |y_1 - y_2|)\}/2 = 2$
- For the red one we have $(|x_1 - x_2| + |y_1 - y_2|)/3 = 2$
- Again, let's pick the tighter one:

$$\max\{(|x_1 - x_2| + |y_1 - y_2|)/3, |x_1 - x_2|/2, |y_1 - y_2|/2\}$$



A small reminder

Modelling a problem as a boolean formula and finding a satisfying evaluation of variables is another general way of solving combinatorial problems

- Boolean variables attain values 0 or 1
- A formula φ is satisfiable iff there exists a value assignment for each variable so that φ becomes true
- Literal is a single variable or its negation (x, $\neg y$)
- Clause is a disjunction of literals ($x \vee y \vee \neg z$)
- Typically we aim for a CNF formula (conjunction of clauses)
  $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge (\neg z)$
- A DNF formula is a disjunction of conjunctions
  $(x \wedge y \wedge \neg z) \vee (\neg x \wedge \neg y) \vee (\neg z \wedge x)$

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation $(x \wedge \neg x)$

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation $(x \wedge \neg x)$
  Then why don't we model problems as DNF?

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation ($x \wedge \neg x$)
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation $(x \wedge \neg x)$
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)
- If we simplify a formula after removing a pure symbol (appears only as a positive or only as a negative literal), can a new pure symbol appear? And what a new unit clause?

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation $(x \wedge \neg x)$
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)
- If we simplify a formula after removing a pure symbol (appears only as a positive or only as a negative literal), can a new pure symbol appear? And what a new unit clause?
  yes, no

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation ($x \wedge \neg x$)
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)
- If we simplify a formula after removing a pure symbol (appears only as a positive or only as a negative literal), can a new pure symbol appear? And what a new unit clause?
  yes, no
- If we simplify a formula after satisfying a unit clause (consist of only one literal), can a new pure symbol appear? And what a new unit clause?

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation ($x \wedge \neg x$)
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)
- If we simplify a formula after removing a pure symbol (appears only as a positive or only as a negative literal), can a new pure symbol appear? And what a new unit clause?
  yes, no
- If we simplify a formula after satisfying a unit clause (consist of only one literal), can a new pure symbol appear? And what a new unit clause?
  yes, yes

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation $(x \wedge \neg x)$
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)
- If we simplify a formula after removing a pure symbol (appears only as a positive or only as a negative literal), can a new pure symbol appear? And what a new unit clause?
  yes, no
- If we simplify a formula after satisfying a unit clause (consist of only one literal), can a new pure symbol appear? And what a new unit clause?
  yes, yes
  $(x \vee y \vee \neg z) \wedge (\neg y) \wedge (\neg y \vee z) \wedge (y \vee q)$

Selected quiz questions

- Suggest an algorithm to verify if a formula in DNF is satisfiable
  Check if any of the DNF clauses contains both a literal and its negation $(x \wedge \neg x)$
  Then why don't we model problems as DNF?
  No polynomial algorithm that transforms a formula into DNF is known (and does not exist unless P=NP)
- If we simplify a formula after removing a pure symbol (appears only as a positive or only as a negative literal), can a new pure symbol appear? And what a new unit clause?
  yes, no
- If we simplify a formula after satisfying a unit clause (consist of only one literal), can a new pure symbol appear? And what a new unit clause?
  yes, yes
  $(x \vee y \vee \neg z) \wedge (\neg y) \wedge (\neg y \vee z) \wedge (y \vee q)$
  $(x \vee \neg z) \wedge (q)$

Conversion to CNF

Convert the following formula into CNF

$$p \Leftrightarrow (q \wedge r)$$

Conversion to CNF

Convert the following formula into CNF

$$p \Leftrightarrow (q \wedge r)$$

- $(\neg p \vee (q \wedge r)) \wedge (\neg(q \wedge r) \vee p)$

Conversion to CNF

Convert the following formula into CNF

$$p \Leftrightarrow (q \land r)$$

- $(\neg p \lor (q \land r)) \land (\neg(q \land r) \lor p)$
- $(\neg p \lor q) \land (\neg p \lor r) \land (\neg q \lor \neg r \lor p)$

Conversion to CNF

Convert the following formula into CNF

$$p \Leftrightarrow (q \wedge r)$$

- $(\neg p \vee (q \wedge r)) \wedge (\neg(q \wedge r) \vee p)$
- $(\neg p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee \neg r \vee p)$

Convert negation of this CNF formula to CNF

$$(p \vee \neg q) \wedge (\neg r \vee s)$$

Conversion to CNF

Convert the following formula into CNF

$$p \Leftrightarrow (q \wedge r)$$

- $(\neg p \vee (q \wedge r)) \wedge (\neg(q \wedge r) \vee p)$
- $(\neg p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee \neg r \vee p)$

Convert negation of this CNF formula to CNF

$$(p \vee \neg q) \wedge (\neg r \vee s)$$

1. Negate
2. Apply DeMorgan rules 2 times
3. Apply distribution rules 2 times

Conversion to CNF

Convert the following formula into CNF

$$p \Leftrightarrow (q \wedge r)$$

- $(\neg p \vee (q \wedge r)) \wedge (\neg(q \wedge r) \vee p)$
- $(\neg p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee \neg r \vee p)$

Convert negation of this CNF formula to CNF

$$(p \vee \neg q) \wedge (\neg r \vee s)$$

1. Negate
2. Apply DeMorgan rules 2 times
3. Apply distribution rules 2 times
4. $$(\neg p \vee r) \wedge (q \vee r) \wedge (\neg p \vee \neg s) \wedge (q \vee \neg s)$$

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \lor q), (\neg q \lor r)$

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$
- we get 4 clauses

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$
- we get 4 clauses
  1. $(\neg p \vee q)$ (premise)
  2. $(\neg q \vee r)$ (premise)

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$
- we get 4 clauses
  1. $(\neg p \vee q)$ (premise)
  2. $(\neg q \vee r)$ (premise)
  3. p (from negated conclusion)
  4. $\neg r$ (from negated conclusion)

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$
- we get 4 clauses
  1. $(\neg p \vee q)$ (premise)
  2. $(\neg q \vee r)$ (premise)
  3. p (from negated conclusion)
  4. $\neg r$ (from negated conclusion)
  5. q (1,3, variable p)

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$
- we get 4 clauses
  1. $(\neg p \vee q)$ (premise)
  2. $(\neg q \vee r)$ (premise)
  3. p (from negated conclusion)
  4. $\neg r$ (from negated conclusion)
  5. q (1,3, variable p)
  6. r (2,5, variable q)

Resolution

Prove using resolution that

$$\{(p \Rightarrow q), (q \Rightarrow r)\} \models (p \Rightarrow r)$$

- Convert each premise to CNF: $(\neg p \vee q), (\neg q \vee r)$
- Convert the negation of the conclusion to CNF

  $\neg(\neg p \vee r) \leftrightarrow p \wedge \neg r$
- we get 4 clauses
  1. $(\neg p \vee q)$ (premise)
  2. $(\neg q \vee r)$ (premise)
  3. p (from negated conclusion)
  4. $\neg r$ (from negated conclusion)
  5. q (1,3, variable p)
  6. r (2,5, variable q)
  7. $\perp$ (4,6, variable r)

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

Variables:

$x_i = \mathit{true} \Leftrightarrow$ node x is assigned color i

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

Variables:

$x_i = \mathit{true} \Leftrightarrow$ node x is assigned color i

Constraints:

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

Variables:

$x_i = \mathit{true} \Leftrightarrow$ node x is assigned color i

Constraints:

1. For each edge (x, y) and each color $i \in \{1, \ldots, k\} : \neg x_i \vee \neg y_i$

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

Variables:

$x_i = \mathit{true} \Leftrightarrow$ node x is assigned color i

Constraints:

1. For each edge (x, y) and each color $i \in \{1, \ldots, k\}$: $\neg x_i \vee \neg y_i$
2. For each vertex x: $\bigvee_{i \in \{1,\ldots,k\}} x_i$

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

Variables:

$x_i = true \Leftrightarrow$ node x is assigned color i

Constraints:

1. For each edge (x, y) and each color $i \in \{1, \ldots, k\} : \neg x_i \vee \neg y_i$
2. For each vertex x: $\bigvee_{i \in \{1,\ldots,k\}} x_i$
3. For each vertex x and for every 2 colors $i, j \in \{1, \ldots, k\}, i \neq j : \neg x_i \vee \neg x_j$

SAT modeling

Assume a graph-coloring problem. How do you encode it as a satisfiability problem?

Graph coloring

1. Two adjacent nodes cannot have the same color
2. Each node is assigned at least one of the available colors
3. (Each node is assigned at most one of the colors)

Variables:

$x_i = \mathit{true} \Leftrightarrow$ node x is assigned color i

Constraints:

1. For each edge (x, y) and each color $i \in \{1, \dots, k\} : \neg x_i \vee \neg y_i$
2. For each vertex x: $\bigvee_{i \in \{1,\dots,k\}} x_i$
3. For each vertex x and for every 2 colors $i, j \in \{1, \dots, k\}, i \neq j : \neg x_i \vee \neg x_j$

How to find a chromatic number of a graph?

SAT modeling

Formulate sudoku as SAT

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = \textit{true} \Leftrightarrow$ cell at position i, j is assigned value k

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = true \Leftrightarrow$ cell at position i, j is assigned value k

Constraints:

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = true \Leftrightarrow$ cell at position i, j is assigned value k

Constraints:

1. There is at least one number in each entry
2. Each number appears at most once in each column
3. Each number appears at most once in each row
4. Each number appears at most once in each block

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = \mathit{true} \Leftrightarrow$ cell at position i, j is assigned value k

Constraints:

1. There is at least one number in each entry
   $\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigvee_{k=1}^{9} x_{ijk}$
2. Each number appears at most once in each column
3. Each number appears at most once in each row
4. Each number appears at most once in each block

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = true \Leftrightarrow$ cell at position i, j is assigned value k

Constraints:

1. There is at least one number in each entry
   $\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigvee_{k=1}^{9} x_{ijk}$
2. Each number appears at most once in each column
   $\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{i=1}^{8} \bigwedge_{i'=i+1}^{9} (\neg x_{ijk} \vee \neg x_{i'jk})$
3. Each number appears at most once in each row
4. Each number appears at most once in each block

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = \mathit{true} \Leftrightarrow$ cell at position i, j is assigned value k

Constraints:

1. There is at least one number in each entry
   $\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigvee_{k=1}^{9} x_{ijk}$
2. Each number appears at most once in each column
   $\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{i=1}^{8} \bigwedge_{i'=i+1}^{9} (\neg x_{ijk} \vee \neg x_{i'jk})$
3. Each number appears at most once in each row
   $\bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{j=1}^{8} \bigwedge_{j'=j+1}^{9} (\neg x_{ijk} \vee \neg x_{ij'k})$
4. Each number appears at most once in each block

SAT modeling

Formulate sudoku as SAT

Variables:

$x_{ijk} = true \Leftrightarrow$ cell at position i, j is assigned value k

Constraints:

1. There is at least one number in each entry
   $\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigvee_{k=1}^{9} x_{ijk}$
2. Each number appears at most once in each column
   $\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{i=1}^{8} \bigwedge_{i'=i+1}^{9} (\neg x_{ijk} \vee \neg x_{i'jk})$
3. Each number appears at most once in each row
   $\bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{j=1}^{8} \bigwedge_{j'=j+1}^{9} (\neg x_{ijk} \vee \neg x_{ij'k})$
4. Each number appears at most once in each block
   $\bigwedge_{k=1}^{9} \bigwedge_{i'=0}^{2} \bigwedge_{j'=0}^{2} \bigwedge_{i=1}^{3} \bigwedge_{j=1}^{3} \bigwedge_{j''=j+1}^{3}$
   $(\neg x_{(3i'+i)(3j'+j)k} \vee \neg x_{(3i'+i)(3j'+j'')k})$
   $\bigwedge_{k=1}^{9} \bigwedge_{i'=0}^{2} \bigwedge_{j'=0}^{2} \bigwedge_{i=1}^{3} \bigwedge_{j=1}^{3} \bigwedge_{i''=x+1}^{3} \bigwedge_{j''=y+1}^{3}$
   $(\neg x_{(3i'+i)(3j'+j)k} \vee \neg x_{(3i'+i'')(3j'+j'')k})$