1 CSP as IP

Alice, Bob, and Charles want to study in Gates, which has 9 floors. To maximize productivity, we need to assign each of them to a separate floor without violating the following constraints:

- Alice only has access to floors 4 through 8
- Bob only has access to floors 3 through 7
- Charles must be at least 2 floors higher than Bob
- Alice must be at least 1 floor higher than Bob
- Alice must be at least 1 floor lower than Charles

Our goal is to assign Alice, Bob, and Charles to the highest possible floors.

1. Formulate the problem as a CSP.

   Variables: X_A, X_B, X_C (where X_A is Alice's floor number, X_B is Bob's floor number, and X_C is Charles's floor number)

   Domains:

   - $D_A \in \{4, 5, 6, 7, 8\}$
   - $D_B \in \{3, 4, 5, 6, 7\}$
   - $D_C \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

   Constraints:

   - $X_C \geq 2 + X_B$ (Charles must be at least 2 floors higher than Bob)
   - $X_A \geq 1 + X_B$ (Alice must be at least 1 floor higher than Bob)
   - $X_A \leq X_C - 1$ (Alice must be at least 1 floor lower than Charles)

2. Formulate the problem as an IP problem.

   Variables: X_A, X_B, X_C (where X_A is Alice's floor number, X_B is Bob's floor number, and X_C is Charles's floor number).

   Goal: We want to find the $max_{X_A,X_B,X_C} X_A + X_B + X_C$. This is equivalent to saying we want to find $min_{X_A,X_B,X_C}(-X_A - X_B - X_C)$

   Constraints:

   - Alice only has access to floors 4 through 8

     $$X_A \geq 4, \text{ which is equivalent to } -X_A \leq -4$$
     $$X_A \leq 8$$

   - Bob only has access to floors 3 through 7

     $$X_B \geq 3, \text{ which is equivalent to } -X_B \leq -3$$
     $$X_B \leq 7$$

- Charles only has access to floors 1 through 9 (Note: this constraint was not explicitly given, but Gates has 9 floors, and we need to make sure that Charles is assigned to a floor between 1 and 9, inclusive)

$$X_C \geq 1, \text{ which is equivalent to } -X_C \leq -1$$
$$X_C \leq 9$$

- Charles must be at least 2 floors higher than Bob

$$X_C \geq 2 + X_B, \text{ which is equivalent to } X_B - X_C \leq -2$$

- Alice must be at least 1 floor higher than Bob

$$X_A \geq 1 + X_B, \text{ which is equivalent to } X_B - X_A \leq -1$$

- Alice must be at least 1 floor lower than Charles

$$X_A \leq X_C - 1, \text{ which is equivalent to } X_A - X_C \leq -1$$

Therefore, we want to find $min_{X_A,X_B,X_C}(-X_A - X_B - X_C)$ such that

$$-X_A \leq -4$$
$$X_A \leq 8$$

$$-X_B \leq -3$$
$$X_B \leq 7$$

$$-X_C \leq -1$$
$$X_C \leq 9$$

$$X_B - X_C \leq -2$$
$$X_B - X_A \leq -1$$
$$X_A - X_C \leq -1$$

2 Baymax's Factory

Baymax and the 281 TAs have opened a factory to produce special medicine and bandages. These are really difficult to produce and require the collaboration of robots and humans.

To produce an ounce of medicine, it takes 0.2 hours of human labor and 4 hours of robot labor. To produce an inch of bandage, it takes 0.5 hours of human labor and 2 hours of robot labor. An ounce of medicine sells for \$30 and an inch of bandages sells for \$30. Medicine and bandages can be sold in fractions of an ounce or inch.

We want to maximize our profit so we can buy gifts for all the students. However, the TAs are really busy so they can only devote 90 human hours. In addition, Baymax can only devote 800 robot hours because he has other obligations to tend to. How can we maximize our profit?

1. Is this a linear, mixed or integer programming problem? Formulate and solve it.

   It is a linear programming problem, as the medicine and bandages can be sold a fraction of a unit. Let x be the ounces of medicine and y be the inches of bandages produced.

   Objective: Maximize total profit:

   $$\min_{x,y} -30x - 30y$$

   Constraints:

   $$0.2x + 0.5y \leq 90$$
   $$4x + 2y \leq 800$$
   $$x \geq 0, y \geq 0$$

   Putting this problem in inequality form we have:

   $$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \; s.t. \; A\mathbf{x} \leq \mathbf{b}$$

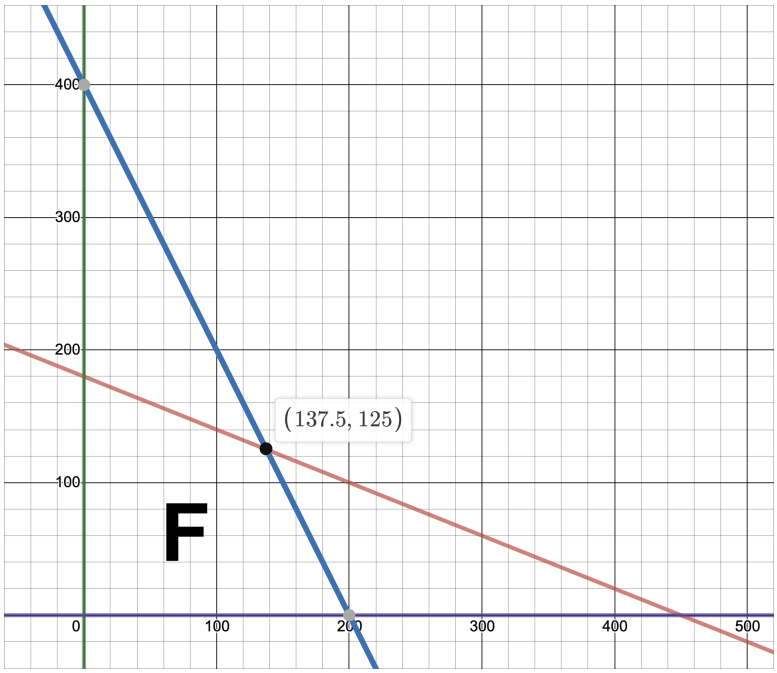
   where

   $$\mathbf{c} = [-30 \;\; -30]^\top$$
   $$\mathbf{x} = [x \; y]^\top$$
   $$\mathbf{b} = [0 \; 0 \; 90 \; 800]$$
   $$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0.2 & 0.5 \\ 4 & 2 \end{bmatrix}$$

   Given the constraints, we can solve for x and y:

   $$y = 180 - 0.4x$$
   $$y = 400 - 2x$$

   That gives us the following graph:



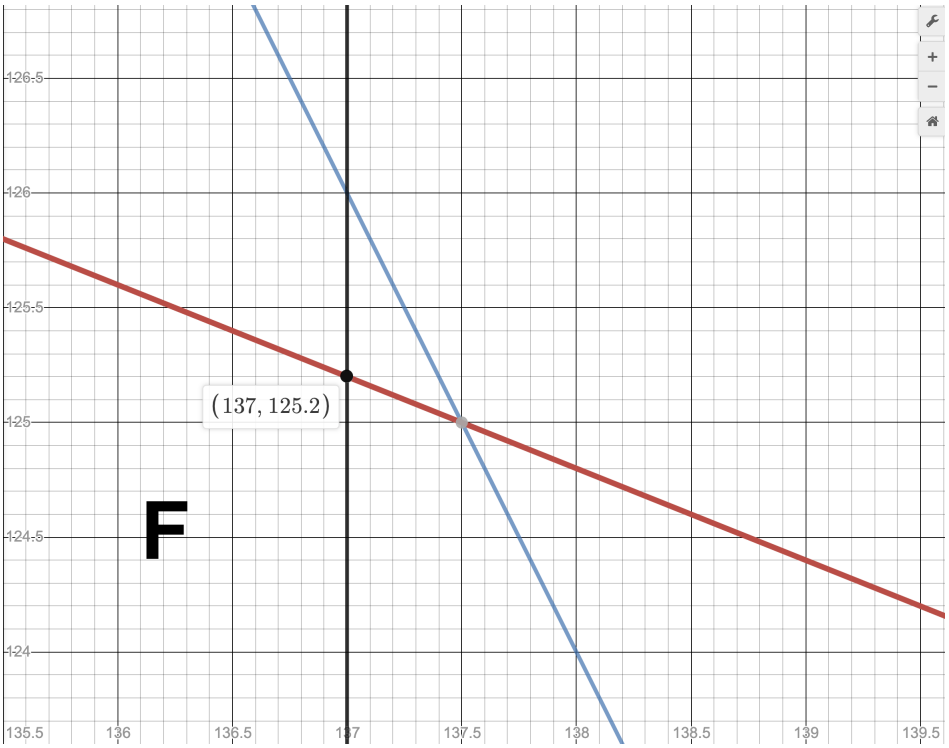
Since we want to maximize profit, we want to choose the furthest point, giving us $x = 137.5$ and $y = 125$.

2. Now suppose the items can only be sold in whole units (by ounce/inch). Is this a linear, mixed, or integer programming problem? Perform branch and bound for one branch level. You do not have to evaluate; writing out the constraints will suffice.

   This is an integer programming problem, and the formulation is identical to part (a). However, the domains of x and y are reduced to integers. We can solve the problem with branch and bound.

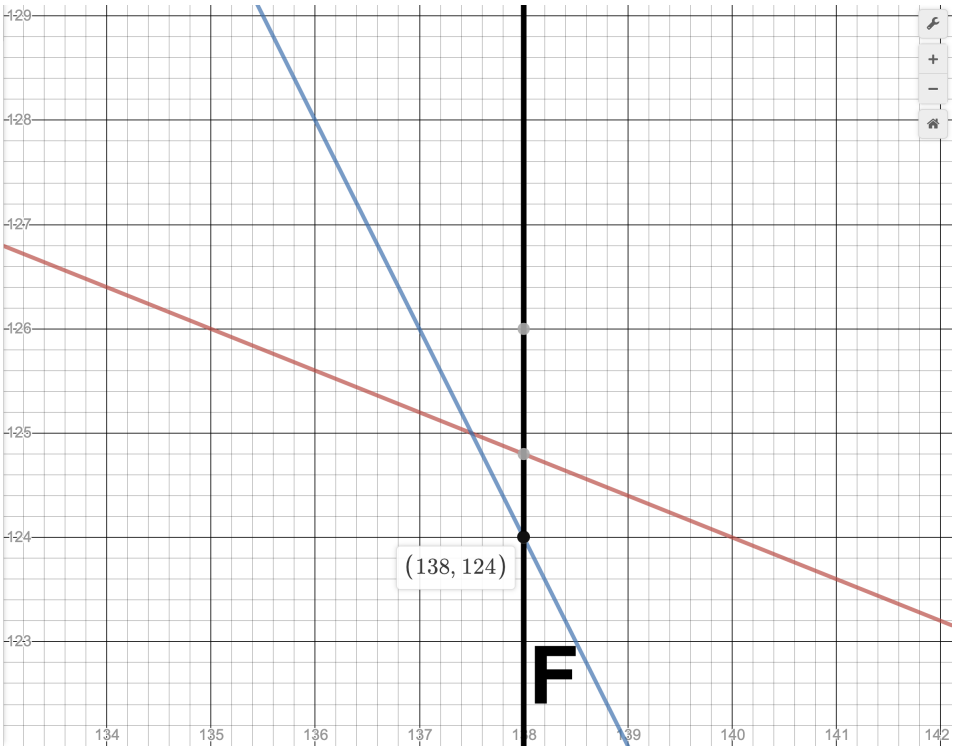
   We first use linear programming to find the optimal point of (137.5, 125), as we did in (1a). Since $x = 137.5$ is not an integer, we branch on it by adding the constraints that $x \leq 137$ or $x \geq 138$.

   Left branch:
   $x \leq 137$
   $0.2x + 0.5y \leq 90$
   $4x + 2y \leq 800$
   $x \geq 0, y \geq 0$



The linear programming solution is (137, 125.2), so we add the left branch to the pq with value -7866. Next we consider the right branch.

Right branch:
$x \geq 138$
$0.2x + 0.5y \leq 90$
$4x + 2y \leq 800$
$x \geq 0, y \geq 0$



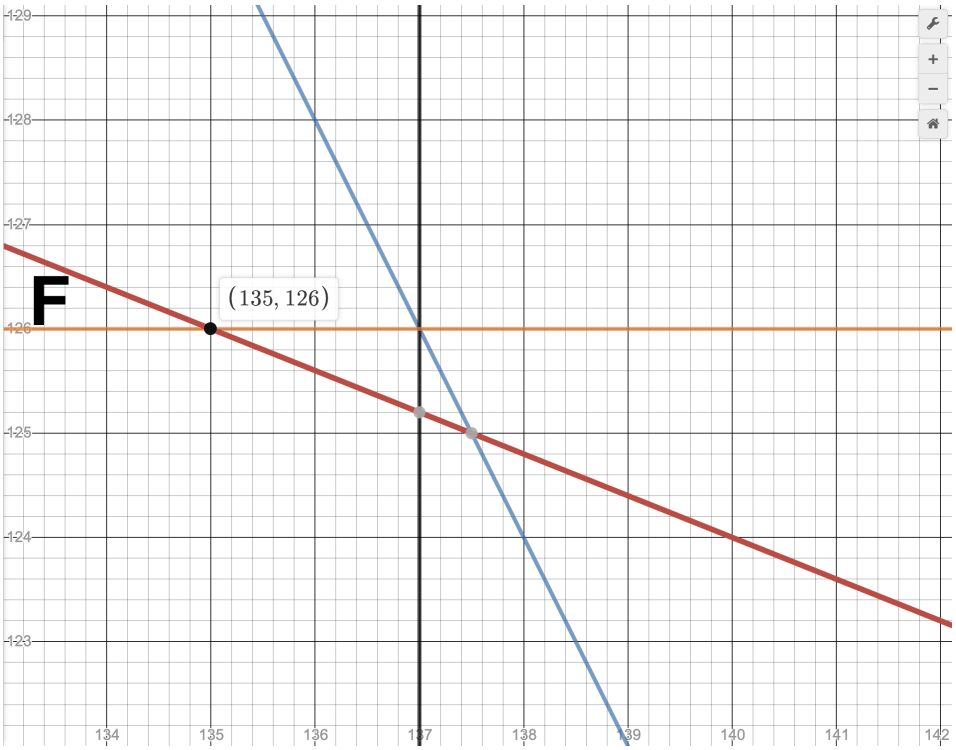
the linear programming solution (138, 124) has a value of -7860 and we similarly add it to the pq.

The left branch has a better value so it is popped of the priority queue first and then since the left branch solution is not an integer value the left-right and left-left branches are considered. Specifically we now need to branch on the y value by adding the constraints that $y \leq 125$ or $y \geq 126$.

Left-Left branch:
$x \leq 137$
$0.2x + 0.5y \leq 90$
$4x + 2y \leq 800$
$x \geq 0, y \geq 0$
$y \leq 125$

The linear programming solution is (137, 125) and it is added to the priority queue along with the value -7860.

Left-Right branch:
$x \leq 137$
$0.2x + 0.5y \leq 90$
$4x + 2y \leq 800$
$x \geq 0, y \geq 0$
$y \geq 126$



The linear programming solution is (135, 126) and it is added to the pq along with the value -7830.

Next on the pq there is a tie between right and left-left branches. Both solutions are integer so the one that is popped of the queue next will be returned as our solution with objective -7860 and point (138, 124) or (137, 125).

3. Now assume medicine can be sold in fractions but bandages can only be sold in whole units. What kind of a programming problem would this be, and how would our evaluation process differ from the problem type in part b?

   This will be a mixed integer linear programming problem. We will evaluate by only branching and bounding on the number of bandages.

4. How many optimal solutions can a LP have? How about IP?

   Both LP and IP can have an infinite number of optimal solutions. Imagine a cost vector that's perpendicular to a constraint boundary. Then, we could have that constraint boundary cross infinitely many integers/real numbers (i.e. the line $x = 0$).

3 4-Queens

Recall the 4-Queens problem. The goal is to place 4 chess queens on a 4x4 chess board such at no two queens are in the same row, column and diagonal.

Formulate the 4-Queens problem as an integer programming problem.

Let our variables be x_{ij} for $0 \leq i \leq 3$, $0 \leq j \leq 3$, representing whether there is a queen in row i, column j. We want to find $\max_x \sum_i \sum_j x_{ij}$ such that $x_{ij} \in \{0, 1\}$.

Check: only one queen in each row - fix i and iterate over each column, ensuring they sum up to ≤ 1.

$$\sum_j x_{ij} = 1 \forall i \in \{0, 3\}$$

Check: only one queen in each column: fix j and iterate over each row, ensuring they sum up to ≤ 1.

$$\sum_i x_{ij} = 1 \forall j \in \{0, 3\}$$

Check: at most one queen in positive-slope diagonals (stretching from top left to bottom right):

$$\sum_{i,j:i+j=k} x_{ij} \leq 1, \forall k \in \{0, 1, 2, ..., 6\}$$

(k=0: (0, 0) | k=1: (0,1), (1,0) | k=2: (0,2), (1,1), (2,0) | k=3 ...)

Check: at most one queen in negative-slope diagonals (stretching from bottom left to top right):

$$\sum_{i,j:i-j=k} x_{ij} \leq 1, \forall k \in \{-3, -2, -1, ..., 3\}$$

Note that the equalities should all be represented as inequalities ≤ 1 and the negation of it ≤ -1. Putting this problem in standard form we have:

$$\min_{\mathbf{x}} 0 \ s.t. \ A\mathbf{x} = \mathbf{b}$$

where

$$\mathbf{x} = [x_{00} \ x_{01} \ x_{02} \ x_{03} \ x_{10} \ x_{11} \ x_{12} \ x_{13} \ x_{20} \ x_{21} \ x_{22} \ x_{23} \ x_{30} \ x_{31} \ x_{32} \ x_{33}]^\top$$

$$\mathbf{b} = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the first 4 rows of A are row restrictions, the next 4 are column restrictions, and the last 7 are diagonal restrictions.

4 Conceptual Review

1. Vocabulary check: Are you familiar with the following terms?

   - Symbols:

     Variables that can be T/F (capital letter)

   - Operators:

     And ($\wedge$), Or ($\vee$), Implies ($\Rightarrow$), Equivalent ($\Longleftrightarrow$)

   - Sentences:

     Symbols connected with operators, can be T/F

   - Equivalence:

     True in all models that A and B imply each other (A equivalent to B)

   - Literals:

     Atomic Sentence

   - Knowledge Base:

     Sentences known to be true

   - Entailment:

     A entails B iff for every model that satisfies A, B is also true

   - Clauses (Definite vs. Horn Clauses):

     Clause: A conjunction of literals
     Definite Clause: Clause with exactly one positive literal
     Horn Clause: Clause with at most one positive literal

   - Model Checking:

     Check if sentences are true in given model/check entailment

   - Theorem Proving:

     Search for sequence of proof steps (e.g. Forward Chaining)

   - Modus Ponens:

     From P and $(P \Rightarrow Q)$, infer Q

2. Recall the definitions of satisfiability and entailment.

   - **Satisfiability**:

     A sentence is *satisfied by* some model (an assignment of values to variables) m if m makes the sentence true.
     A sentence is satisfiable if there exists a model that satisfies it.

   - **Entailment**:

Entailment: $a \models b$ ("a entails b" or "b follows from a") iff every model that satisfies a also satisfies b. In other words, the a-worlds (worlds where a is true) are a subset of the b-worlds [$models(a) \subseteq models(b)$].

3. What is the difference between satisfiability and entailment?

   Satisfiability holds if there exists a single model that in which the sentence is true. This is a property of a sentence (a sentence is satisfiable or it is not).

   Entailment holds if all models which satisfy one sentence (query) also satisfy another sentence (knowledge base). It relates two sentences (sentence a entails b, or a does not entail b), and requires checking all models that satisfy a.

4. Suppose $A \models B$. Consider all models assigning values to variables in sentences A and B. Which of the following sentences must be true in all possible models (even if either or both A/B are false)?

   (a) $A \wedge B$
   (b) $A \Rightarrow B$
   (c) $B \Rightarrow A$
   (d) $A \vee B$
   (e) B

   (b) $A \Rightarrow B$
   By definition of implication, in all models (i.e., truth assignments) where A is true, B is also true. Thus in all models, $A \Rightarrow B$ is satisfied.
   (Thinking of it conversely, there would never be a model where B is false and A is true. By definition of implication, this means no model the rule $A \Rightarrow B$.)

5. Determine which of the following are correct, and explain your reasoning.

   - $(A \vee B) \models (A \Rightarrow B)$

     False (when A is true and B is False, $A \vee B$ is true but $A \Rightarrow B$ is false)

   - $A \iff B \models A \vee \neg B$

     True (the RHS is $B \Rightarrow A$, which is one of the conjuncts in the definition of $A \iff B$)

   - $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable

     True (the model has A and $\neg B$)

6. How would we formulate the SAT problem as a CSP? What are the variables? Domains? Constraints?

   SAT can be modeled as a CSP in which the variables are literals with domain $\{\top, \bot\}$, and the constraints are the clauses themselves.

7. Suppose we have an algorithm which determines whether a sentence is satisfiable or not. Given two sentences A and B, how could we determine whether $A \models B$?

   If $A \models B$, then $A \wedge \neg B$ should be unsatisfiable (this is proof via reductio ad absurdum - reduction to an absurd thing).

5 SATurdays are for everyone

1. Determine whether the sentences below are satisfiable or unsatisfiable (using any method you like).

   (a) $(\neg(Y \vee \neg Y) \vee X) \wedge (X \vee (Z \iff \neg Z))$

   Satisfiable

   Logical reduction:

$(\neg(Y \vee \neg Y) \vee X) \wedge (X \vee (Z \iff \neg Z))$	*original sentence*
$((\neg Y \wedge Y) \vee X) \wedge (X \vee (Z \iff \neg Z))$	*De Morgan's Law*
$(\bot \vee X) \wedge (X \vee (Z \iff \neg Z))$	$(\neg Y \wedge Y)$ *reduces to* $\bot$
$X \wedge (X \vee (Z \iff \neg Z))$	$(\bot \vee X)$ *reduces to* X
$X \wedge (X \vee ((Z \Rightarrow \neg Z) \wedge (\neg Z \Rightarrow Z)))$	*Biconditional Elimination*
$X \wedge (X \vee ((\neg Z \vee \neg Z) \wedge (Z \vee Z)))$	$(\neg Z \vee \neg Z)$ *reduces to* $\neg Z$, $(Z \vee Z)$ *reduces to* Z
$X \wedge (X \vee (\neg Z \wedge Z))$	$(\neg Z \wedge Z)$ *reduces to* $\bot$
$X \wedge (X \vee \bot)$	$(X \vee \bot)$ *reduces to* X
$X \wedge X$	$(X \wedge X)$ *reduces to* X
X	

   Assign $X = \top$. Then, we can choose any arbitrary assignment for Y and Z, and the sentence will be satisfied.

   (b) $\neg(X \vee \neg(X \wedge (Z \vee \top))) \Longrightarrow \neg(Y \wedge (\neg Y \vee (\top \Longrightarrow \bot)))$

   Satisfiable

   Logical reduction:

$\neg(X \vee \neg(X \wedge (Z \vee \top))) \Longrightarrow \neg(Y \wedge (\neg Y \vee (\top \Longrightarrow \bot)))$	*original sentence*
$(X \vee \neg(X \wedge (Z \vee \top))) \vee \neg(Y \wedge (\neg Y \vee (\top \Longrightarrow \bot)))$	*Implication Elimination*
$(X \vee \neg(X \wedge (Z \vee \top))) \vee \neg(Y \wedge (\neg Y \vee (\bot \vee \bot)))$	*Implication Elimination*
$(X \vee \neg(X \wedge (Z \vee \top))) \vee \neg(Y \wedge (\neg Y \vee \bot))$	$(\bot \vee \bot)$ *reduces to* $\bot$
$(X \vee \neg(X \wedge (Z \vee \top))) \vee \neg(Y \wedge \neg Y)$	$(\neg Y \vee \bot)$ *reduces to* $\neg\, Y$
$(X \vee \neg(X \wedge (Z \vee \top))) \vee \neg(\bot)$	$(Y \wedge \neg Y)$ *reduces to* $\bot$
$(X \vee \neg(X \wedge (Z \vee \top))) \vee \top$	$\neg(\bot)$ *reduces to* $\top$
$(X \vee \neg(X \wedge \top)) \vee \top$	$Z \vee \top$ *reduces to* $\top$
$(X \vee \neg X) \vee \top$	$X \wedge \top$ *reduces to* X
$\top \vee \top$	$X \vee \neg X$ *reduces to* $\top$
$\top$	$\top \vee \top$ *reduces to* $\top$

   We can choose any arbitrary assignment for X, Y and Z and the sentence will be satisfied.

   (c) $(((\top \iff \neg(X \vee \neg X)) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Longrightarrow X))$

   Unsatisfiable

Logical reduction:

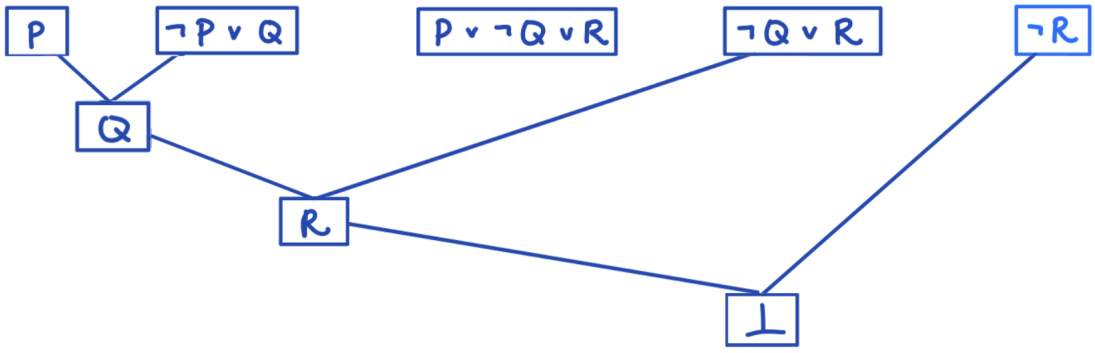
$(((\top \iff \neg(X \vee \neg X)) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	*original sentence*
$(((\top \iff \neg(\top)) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	$(X \vee \neg X)$ *reduces to* $\top$
$(((\top \iff \bot) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	$\neg(\top)$ *reduces to* $\bot$
$(((\top \Rightarrow \bot) \wedge (\bot \Rightarrow \top)) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	*Biconditional Elimination*
$(((\bot \vee \bot) \wedge (\top \vee \top)) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	*Implication Elimination*
$((\bot \wedge \top) \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	$(\bot \vee \bot)$ *reduces to* $\bot$, $(\top \vee \top)$ *reduces to* $\top$
$(\bot \vee Z) \vee Z) \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	$(\bot \wedge \top)$ *reduces to* $\bot$
$Z \wedge \neg(Z \wedge ((Z \wedge \neg Z) \Rightarrow X))$	$((\bot \vee Z) \vee Z)$ *reduces to* Z
$Z \wedge \neg(Z \wedge (\bot \Rightarrow X))$	$(Z \wedge \neg Z)$ *reduces to* $\bot$
$Z \wedge \neg(Z \wedge (\top \vee X))$	*Implication Elimination*
$Z \wedge \neg(Z \wedge \top)$	$(\top \vee X)$ *reduces to* $\top$
$Z \wedge \neg Z$	$(Z \wedge \top)$ *reduces to* Z
$\bot$	$Z \wedge \neg Z$ *reduces to* $\bot$

This sentence is logically equivalent to $\bot$, and therefore cannot be satisfied.

2. Given the following knowledge base (propositions we know to be true):

   - P
   - $\neg P \vee Q$
   - $P \vee \neg Q \vee R$
   - $\neg Q \vee R$

   Prove R to be true using resolution.



6 Wandering in Wumpus World

We bring together what we have learned in lecture as well as the ideas of search so far in order to construct wumpus world agents that use propositional logic. The first step is to enable the agent to deduce, to the extent possible, the state of the world given its percept history. This requires writing down a complete logical model of the effects of actions. We also show how the agent can keep track of the world efficiently without going into the percept history for each inference. Finally, we show how the agent can use logical inference to construct plans that are guaranteed to achieve its goals.

Try it out: http://thiagodnf.github.io/wumpus-world-simulator/ Note that there are some slight differences between this online version and the version we describe below.

Throughout this question, we will present several screenshots from the Wumpus World simulator linked previously. Note that the location of the explorer can be ignored. We just tried to place him somewhere off screen!

Recall that an agent in the Wumpus World has access to the following percepts:

- In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a Stench.
- In the squares directly adjacent to a pit, the agent will perceive a Breeze.
- In the square where the gold is, the agent will perceive a Glitter.
- When an agent walks into a wall, it will perceive a Bump.
- When the wumpus is killed, it emits a woeful Scream that can be perceived anywhere in the cave.

1. Consider the following Wumpus World state:

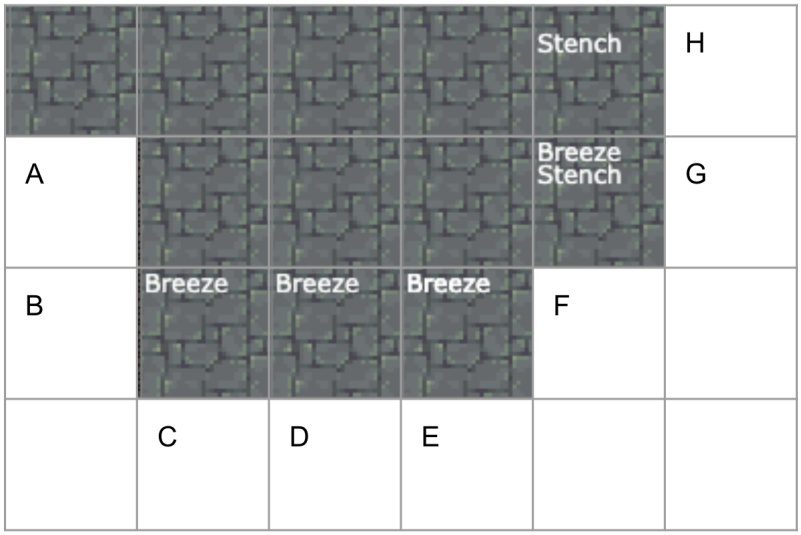
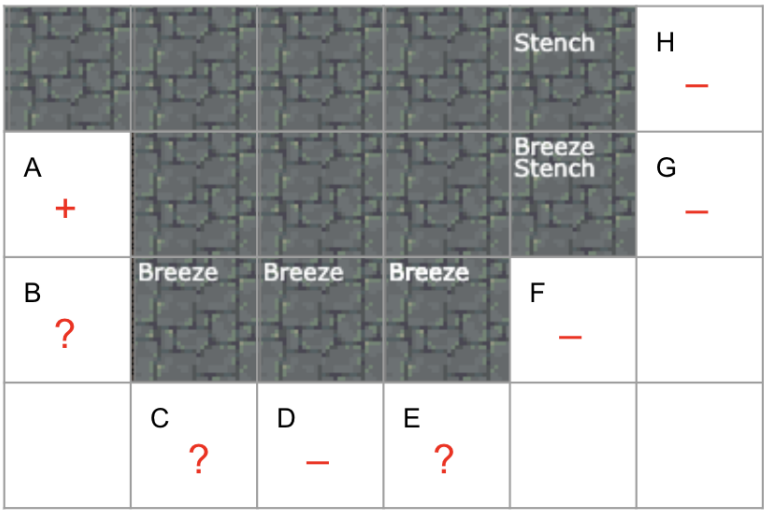


Figure 1: Safe, not safe, or unsure?

For the squares A-H, mark each of them with '+' if the square is definitely safe, '–' if it is definitely not safe, and '?' otherwise.



(a) Safe, since there is no adjacent breeze or stench.

(b) Unsure, since there may be a pit.

(c) Unsure, since there may be a pit.

(d) Not safe - it's the only square adjacent to the breeze, so it must be a pit.

(e) Unsure, since there may be a pit.

(f) Unsafe, there are two squares adjacent to the breeze/stench square, so both must be hazards.

(g) Unsafe - same reasoning as F.

(h) Unsafe. Since there is only one square adjacent to the stench, this square is a wumpus.

2. Consider the PL symbols $S_A, S_B \ldots S_H$ that represent whether or not each square A-H is safe or not safe. We also have symbols W_i and P_i for whether square i has a wumpus or a pit, respectively.

   Let our Knowledge Base (KB) be a propositional logic sentence in CNF which contains all of the logic for the rules of Wumpus World as well as the percepts for the squares already visited (i.e., info about stenches, breezes, or not in the 12 dark squares in the figure). Assume the KB also has logic related to the safety of a square, i.e.,$\neg W_i \wedge \neg P_i \iff S_i$.

   You are given access to a black box SAT solver function `PL_SATISFIES(pl_sentence}` which returns a PL model satisfying the given sentence *if* it is satisfiable and `None` otherwise.

   Write pseudocode using the SAT solver and given propositional logic symbols to determine is square i is:

   (a) Definitely safe

   ```
   if PL_SATISFIES(KB and not S_i) == None:
       return SAFE
   ```

   (b) Definitely not safe

   ```
   if PL_SATISFIES(KB and S_i) == None:
       return SAFE
   ```

(c) Unsure

```
if PL_SATISFIES(KB and S_i)  != None
&& PL_SATISFIES(KB and !S_i) != None:
    return UNSURE
```