### **Propositional Knowledge Bases**

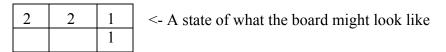
M satisfies a knowledge base means that M is a truth assignment which makes each formula in the knowledge base true.

M can be thought of as a possible world in which KB happens i.e., a model for the knowledge base.

We want to know if a given formula F follows from a knowledge base. To see if it does, we can look at each M such that M satisfies the knowledge base (M |= KB) and check does M |= F.

If answer is always yes, then in every possible world in which the KB holds, the formula f holds. We then say that  $KB \models F$  (knowledge entails/implies F)

## **Example: Minesweeper**



A knowledge board for minesweeper might use a fixed number of variables to code square (i, j), of the board.

## Example: the square(0, 1) is 2 could be represented as

 $\begin{array}{l} X_{013} = false \\ X_{012} = false \\ X_{011} = true \\ X_{010} = false \end{array}$ 

Knowledge base for minesweeper might have formulas to represent for each square if the squares value is x, how it affects the value of neighbour squares.

# Example: Given the rules of minesweeper and the board above, is the code of square (1, 0) a bomb?

This is equivalent to: does KB\_{minesweeper} and known squares as above  $|= x_{103}$  AND  $!x_{102}$  AND  $x_{101}$  AND  $!x_{100}$ 

Square 10 is a bomb

Would like algorithms to say what follows from a given set of propositional formulas.

Brute-force algorithm (model checking)

Both KB and F involve only a finite number of variables.

For each possible truth assignment, verify that it does not make KB true & F false, if verification succeeds  $KB \models F$ .

 $2^n$  truth assigns so algorithm is  $O(2^n)$  time

Another approach is to use a proof system and then develop algorithms for efficiently finding proofs in this system.

# **Example: Frege Proof System**

Proof in this system consists of sequences of formulas F1, F2, ..., Fn. Last statement is what we proved.

Fi in list must be either

- 1) A substitution instance of an axiom
- 2) a member of a knowledge base
- exists an Fk and an Fj := Fk -> Fi which appear earlier in the proof. Here Fk -> Fi is an abbreviation for ((NOT Fk) OR Fi) 3)