

Model Checking

Stephen G. Ware

CSCI 4525 / 5525



THE UNIVERSITY *of*
NEW ORLEANS

Logical Inference

Given a **knowledge base** of known facts, can we derive new facts? Such facts are said to be **entailed** by the knowledge base.



Wumpus World



- The player, who can navigate the grid world
- The wumpus, who will eat the player if the player blunders into its square
- Bottomless pits
- A chest of gold

Wumpus World



- When adjacent to the wumpus, the player detects a stench.
- When adjacent to a pit, the player detects a breeze.
- When in the square with the gold, the player detects a glimmer.

Wumpus World



Actions:

- Move N, S, E, W
- Grab the treasure



Propositional Representation

Propositions:

- $BA1$ = “There is a breeze at A1.”
- $BB1$ = “There is a breeze at B1.”
- $PC1$ = “There is a pit at C1.”
- $PA2$ = “There is a pit at A2.”
- $PB2$ = “There is a pit at B2.”

... and so on



Knowledge Base + Sensing

Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$

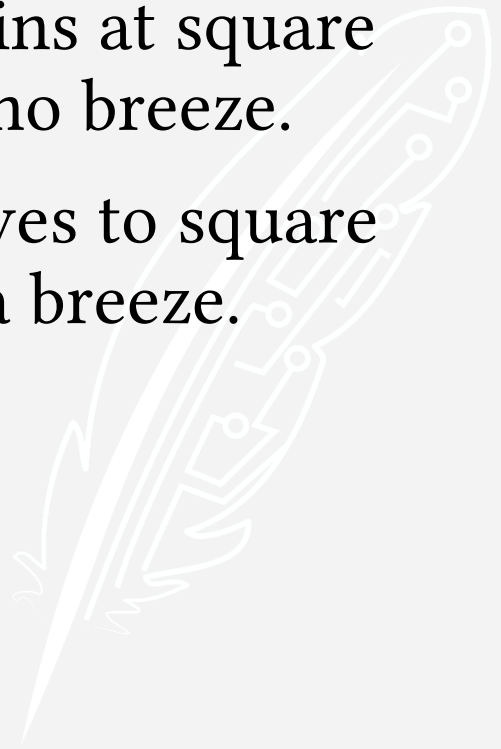
... and so on

- $\neg BA1$
- $BB1$

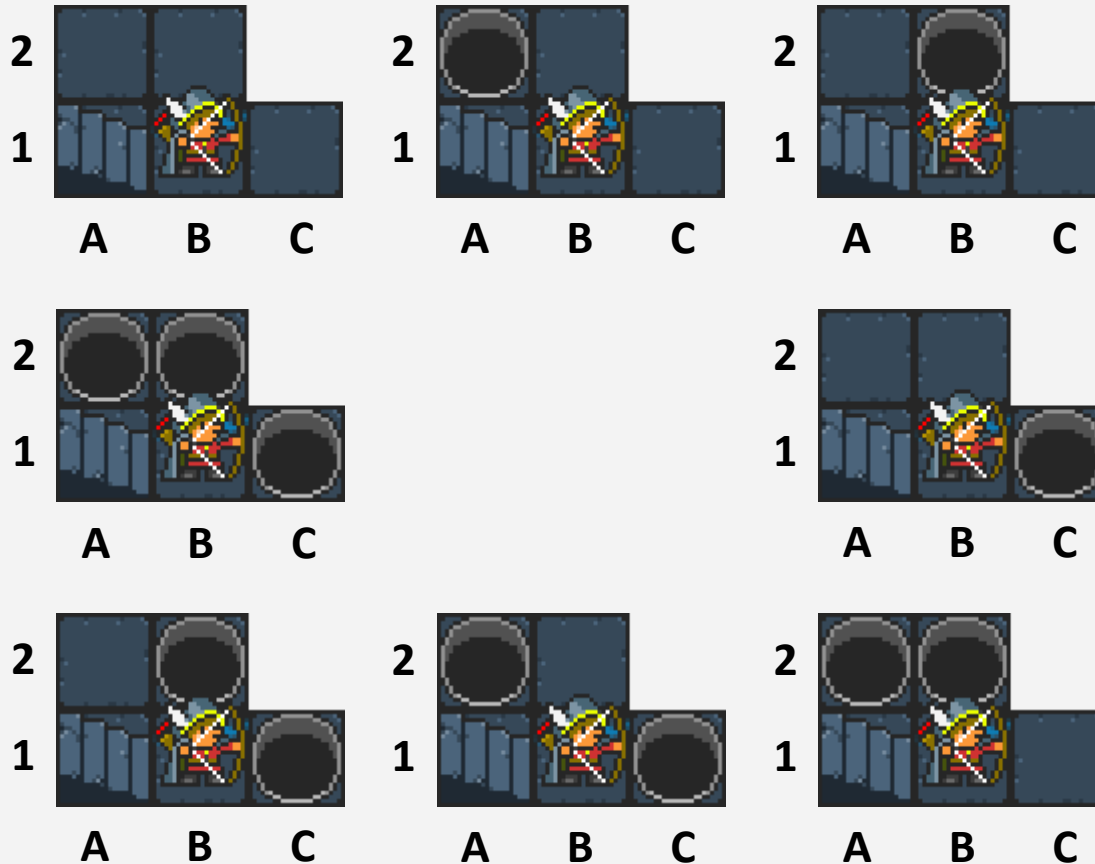
Events:

The player begins at square A1 and senses no breeze.

The player moves to square B1 and senses a breeze.



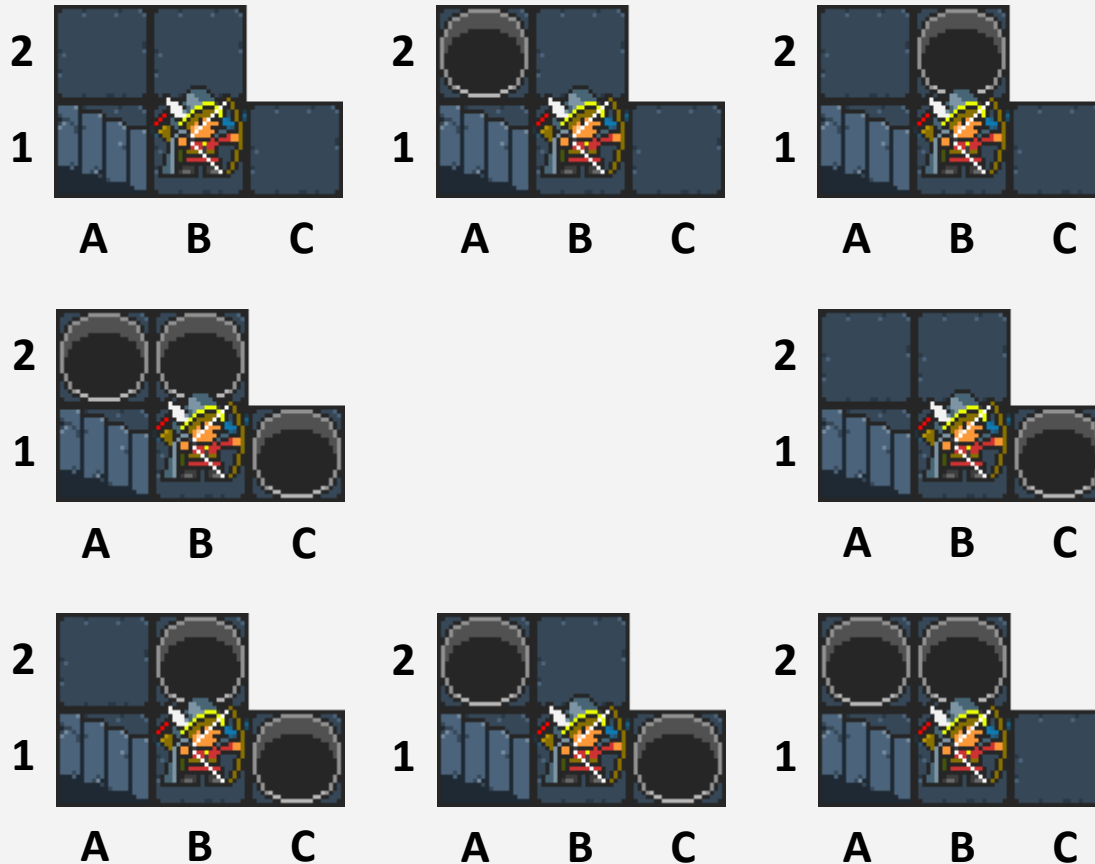
Possible Worlds



Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$
- ... and so on
- $\neg BA1$
- $BB1$

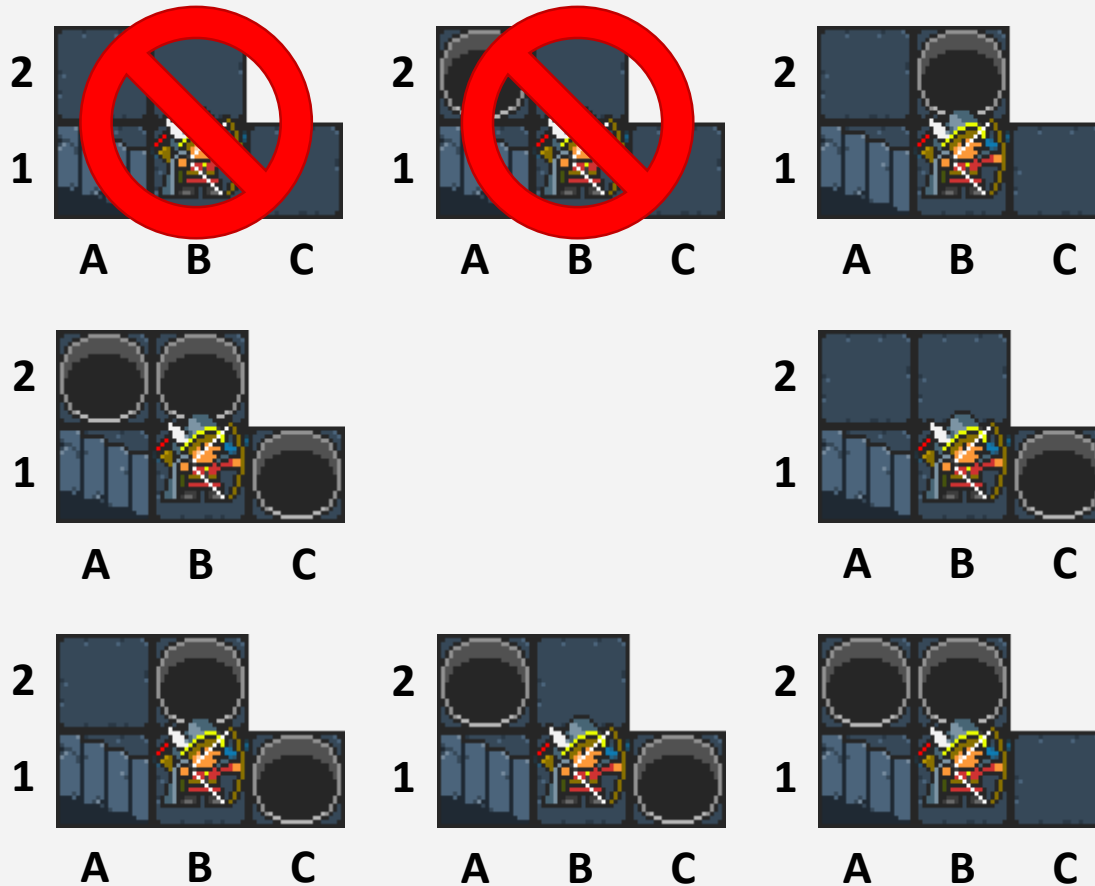
Possible Worlds



Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$
- ... and so on
- $\neg BA1$
- $BB1$

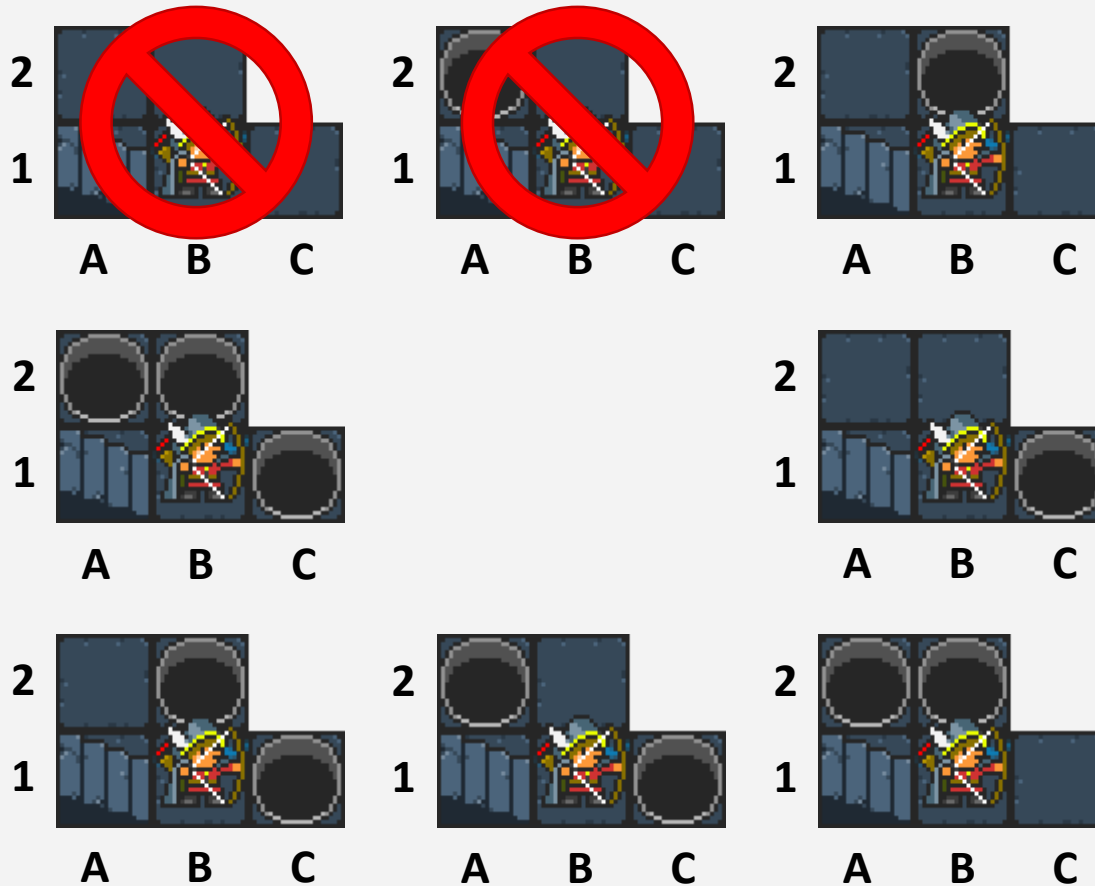
Possible Worlds



Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$
- ... and so on
- $\neg BA1$
- $BB1$

Possible Worlds



Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$
- ... and so on
- $\neg BA1$
- $BB1$

Possible Worlds



Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$
- ... and so on
- $\neg BA1$
- $BB1$

Possible Worlds



Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$
- ... and so on
- $\neg BA1$
- $BB1$
- $\neg PA2$
- $PB2 \vee PC1$

Logical Entailment

A fact is logically **entailed** by a knowledge base if it is true in every possible world.



Computational Deduction

How can we replicate this logical inference procedure on a computer using a simple knowledge representation and a reusable algorithm?

Begin by putting everything into a common format, such as conjunctive normal form (CNF).

Then, explore assignments of *true* or *false* to variables until we find one that works.

Model Checking

A **model** is an assignment (or partial assignment) of truth values to all the propositional variables for a problem.

The goal of model checking is to find a model which **satisfies** all the known facts in the world.

Model checking is very similar to constraint satisfaction problem solving.

Propositional Satisfiability

Given a logical expression in CNF and a model, we say that the model satisfies the expression if the expression can be reduced to *true*.

Interesting historical fact: Propositional SAT was the first problem proven to be NP-complete and is the basis on which all NP-completeness proofs rest.

Wumpus in CNF

Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $BB1 \leftrightarrow PB2 \vee PC1$

Remember: $(x \leftrightarrow y) \leftrightarrow ((x \rightarrow y) \wedge (y \rightarrow x))$



Wumpus in CNF

Knowledge Base:

- $BA1 \rightarrow PA2 \vee PB1$
- $PA2 \vee PB1 \rightarrow BA1$
- $BB1 \rightarrow PB2 \vee PC1$
- $PB2 \vee PC1 \rightarrow BB1$

Remember: $(x \rightarrow y) \leftrightarrow (\neg x \vee y)$



Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee (PA2 \vee PB1)$
- $\neg(PA2 \vee PB1) \vee BA1$
- $\neg BB1 \vee (PB2 \vee PC1)$
- $\neg(PB2 \vee PC1) \vee BB1$

Remember: $(x \rightarrow y) \leftrightarrow (\neg x \vee y)$



Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $\neg(PA2 \vee PB1) \vee BA1$
- $\neg BB1 \vee PB2 \vee PC1$
- $\neg(PB2 \vee PC1) \vee BB1$

Remember: $(x \rightarrow y) \leftrightarrow (\neg x \vee y)$



Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $\neg(PA2 \vee PB1) \vee BA1$
- $\neg BB1 \vee PB2 \vee PC1$
- $\neg(PB2 \vee PC1) \vee BB1$

Remember: $\neg(x \vee y) \leftrightarrow (\neg x \wedge \neg y)$



Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $(\neg PA2 \wedge \neg PB1) \vee BA1$
- $\neg BB1 \vee PB2 \vee PC1$
- $(\neg PB2 \wedge \neg PC1) \vee BB1$

Remember: $\neg(x \vee y) \leftrightarrow (\neg x \wedge \neg y)$



Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $(\neg PA2 \wedge \neg PB1) \vee BA1$
- $\neg BB1 \vee PB2 \vee PC1$
- $(\neg PB2 \wedge \neg PC1) \vee BB1$

Remember: $(x \wedge y) \vee z \leftrightarrow (z \vee x) \wedge (z \vee y)$

Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $(BA1 \vee \neg PA2) \wedge (BA1 \vee \neg PB1)$
- $\neg BB1 \vee PB2 \vee PC1$
- $(BB1 \vee \neg PB2) \wedge (BB1 \vee \neg PC1)$

Remember: $(x \wedge y) \vee z \iff (z \vee x) \wedge (z \vee y)$

Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Each of these is a disjunctive clause.

Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Together, they form a conjunction of disjunctive clauses... a CNF expression!

Wumpus in CNF

Equivalent SAT file for Project 2:

```
(and (or (not BA1) PA2 PB1)
      (or BA1 (not PA2))
      (or BA1 (not PB1))
      (or (not BB1) PB2 PC1)
      (or BB1 (not PB2))
      (or BB1 (not PC1)))
```

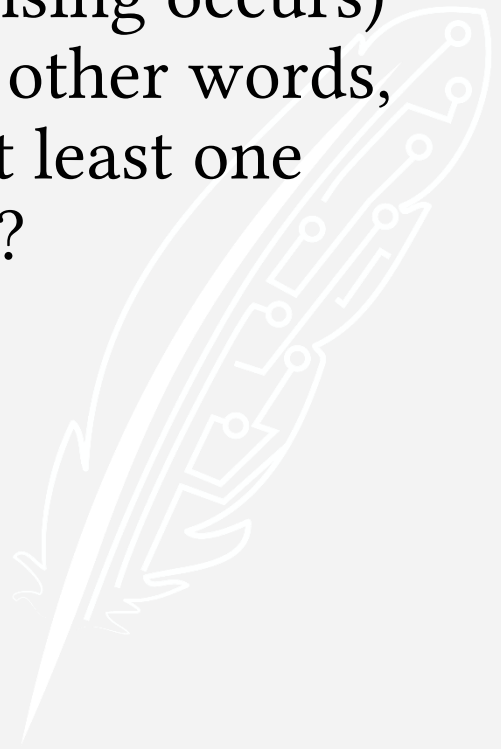


Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Is this initial set of facts (before any sensing occurs) satisfiable? In other words, does it allow at least one possible world?



Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = ?$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

Try to find an assignment of T/F to the variables which reduces all clauses to T.

Wumpus in CNF

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = ?$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

Each clause is a disjunction. Only one variable in it needs to be true. As soon as one variable in the clause is T, the whole clause is T.

Wumpus in CNF

Knowledge Base:

- $\neg F \vee PA2 \vee PB1$
- $F \vee \neg PA2$
- $F \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

Each clause is a disjunction. Only one variable in it needs to be true. As soon as one variable in the clause is T, the whole clause is T.

Wumpus in CNF

Knowledge Base:

- $T \vee PA2 \vee PB1$
- $F \vee \neg PA2$
- $F \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

Each clause is a disjunction. Only one variable in it needs to be true. As soon as one variable in the clause is T, the whole clause is T.

Wumpus in CNF

Knowledge Base:

- T
- $F \vee \neg PA2$
- $F \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

Each clause is a disjunction. Only one variable in it needs to be true. As soon as one variable in the clause is T, the whole clause is T.

Wumpus in CNF

Knowledge Base:

- T
- $F \vee \neg PA2$
- $F \vee \neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

When we know a literal is F, we can remove it from the clause, because it cannot help to make it T.

Wumpus in CNF

Knowledge Base:

- T
- $\neg PA2$
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

When we know a literal is F, we can remove it from the clause, because it cannot help to make it T.

Wumpus in CNF

Knowledge Base:

- T
- $\neg PA2$
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = ?$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

If ever a clause becomes F, we know the model cannot satisfy the expression.

Wumpus in CNF

Knowledge Base:

- T
- $\neg T$
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = T$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

If ever a clause becomes F, we know the model cannot satisfy the expression.

Wumpus in CNF

Knowledge Base:

- T
- F
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = T$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

If ever a clause becomes F, we know the model cannot satisfy the expression.

Wumpus in CNF

Knowledge Base:

- T
- F
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = T$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

We need to backtrack and try a different value for PA2.

Wumpus in CNF

Knowledge Base:

- T
- $\neg F$
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = F$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

We need to backtrack and try a different value for PA2.

Wumpus in CNF

Knowledge Base:

- T
- T
- $\neg PB1$
- $\neg BB1 \vee PB2 \vee PC1$
- $BB1 \vee \neg PB2$
- $BB1 \vee \neg PC1$

Variables:

$BA1 = F$

$BB1 = ?$

$PA2 = F$

$PB1 = ?$

$PB2 = ?$

$PC1 = ?$

We need to backtrack and try a different value for PA2.

Wumpus in CNF

Knowledge Base:

- T
- T
- T
- T
- T
- T

Variables:

$$BA1 = F$$

$$BB1 = F$$

$$PA2 = F$$

$$PB1 = F$$

$$PB2 = F$$

$$PC1 = F$$

This model satisfies the expression. The Wumpus World rules allow at least one possible world.

Naïve SAT Solver

Begin with every variable's value unassigned.

To find a model which satisfies a CNF expression:

 If every clause is true, return true.

 If any clause is empty, return false.

 Choose an unassigned variable V .

 Set $V=T$. Try to find a model that satisfies.

 Set $V=F$. Try to find a model that satisfies.

 Return false.

Notice the similarity to the CSP solver!

Using SAT for Inference

Given a knowledge base (which is known to be satisfiable) and some fact, how can we tell if that fact is entailed?

One approach: Add the fact to the knowledge base and check if it is still satisfiable.

Will this work?

No! This will only tell us if there exists a possible world in which that fact is true, not if it is actually true.

Proof By Contradiction

Given some proposition P that you want to prove true, assume $\neg P$ and show that this leads to an absurd conclusion.

In other words, prove P by showing that $\neg P$ is impossible.

In other words, add $\neg P$ to the knowledge base and show that it has become unsatisfiable.

Wumpus SAT

Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $\neg BA1$

Query: $PA2$



Wumpus SAT

Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $\neg BA1$

Query: $PA2$

Negate the query and add it to the knowledge base.

Wumpus SAT

Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $\neg BA1$
- $\neg PA2$

Negate the query and add it to the knowledge base.

Wumpus SAT

Knowledge Base:

- $BA1 \leftrightarrow PA2 \vee PB1$
- $\neg BA1$
- $\neg PA2$

Convert the knowledge base to CNF.



Wumpus SAT

Knowledge Base:

- $BA1 \rightarrow PA2 \vee PB1$
- $PA2 \wedge PB1 \rightarrow BA1$
- $\neg BA1$
- $\neg PA2$

Convert the knowledge base to CNF.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $\neg(PA2 \vee PB1) \vee BA1$
- $\neg BA1$
- $\neg PA2$

Convert the knowledge base to CNF.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $(\neg PA2 \wedge \neg PB1) \vee BA1$
- $\neg BA1$
- $\neg PA2$

Convert the knowledge base to CNF.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $(BA1 \vee \neg PA2) \wedge (BA1 \vee \neg PB1)$
- $\neg BA1$
- $\neg PA2$

Convert the knowledge base to CNF.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BA1$
- $\neg PA2$

Convert the knowledge base to CNF.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BA1$
- $\neg PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

Begin with all variables unassigned.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BA1$
- $\neg PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

All clauses T? No. Some clause F? No.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BA1$
- $\neg PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

Choose some unassigned variable, such as $BA1$.

Wumpus SAT

Knowledge Base:

- $\neg T \vee PA2 \vee PB1$
- $T \vee \neg PA2$
- $T \vee \neg PB1$
- $\neg T$
- $\neg PA2$

Set $BA1 = T$.

Variables:

$BA1 = T$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- $F \vee PA2 \vee PB1$
- $T \vee \neg PA2$
- $T \vee \neg PB1$
- F
- $\neg PA2$

Set $BA1 = T$.

Variables:

$BA1 = T$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- $PA2 \vee PB1$
- T
- T
- F
- $\neg PA2$

Set $BA1 = T$.

Variables:

$BA1 = T$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- $PA2 \vee PB1$
- T
- T
- F
- $\neg PA2$

Variables:

$$BA1 = T$$

$$PA2 = ?$$

$$PB1 = ?$$

All clauses T? No. Some clause F? Yes.



Wumpus SAT

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BA1$
- $\neg PA2$

Backtrack.

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- $\neg F \vee PA2 \vee PB1$
- $F \vee \neg PA2$
- $F \vee \neg PB1$
- $\neg F$
- $\neg PA2$

Set $BA1 = F$.

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- $T \vee PA2 \vee PB1$
- $F \vee \neg PA2$
- $F \vee \neg PB1$
- T
- $\neg PA2$

Set $BA1 = F$.

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- $\neg PA2$
- $\neg PB1$
- T
- $\neg PA2$

Set $BA1 = F$.

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- $\neg PA2$
- $\neg PB1$
- T
- $\neg PA2$

Variables:

$$BA1 = F$$

$$PA2 = ?$$

$$PB1 = ?$$

All clauses T? No. Some clause F? No.



Wumpus SAT

Knowledge Base:

- T
- $\neg PA2$
- $\neg PB1$
- T
- $\neg PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$

Choose some unassigned variable, such as PA2.

Wumpus SAT

Knowledge Base:

- T
- $\neg T$
- $\neg PB1$
- T
- $\neg T$

Set $PA2 = T$.

Variables:

$BA1 = F$

$PA2 = T$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- F
- $\neg PB1$
- T
- F

Set $PA2 = T$.

Variables:

$BA1 = F$

$PA2 = T$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- $\neg PA2$
- $\neg PB1$
- T
- $\neg PA2$

Backtrack.

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- $\neg F$
- $\neg PB1$
- T
- $\neg F$

Set $PA2 = F$.

Variables:

$BA1 = F$

$PA2 = F$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- T
- $\neg PB1$
- T
- T

Set $PA2 = F$.

Variables:

$BA1 = F$

$PA2 = F$

$PB1 = ?$



Wumpus SAT

Knowledge Base:

- T
- T
- $\neg PB1$
- T
- T

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = ?$$

Choose some unassigned variable, such as PB1.

Wumpus SAT

Knowledge Base:

- T
- T
- $\neg T$
- T
- T

Set $PB1 = T$.

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = T$$



Wumpus SAT

Knowledge Base:

- T
- T
- F
- T
- T

Set $PB1 = T$.

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = T$$



Wumpus SAT

Knowledge Base:

- T
- T
- $\neg PB1$
- T
- T

Backtrack.

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = ?$$



Wumpus SAT

Knowledge Base:

- T
- T
- $\neg F$
- T
- T

Set $PB1 = F$.

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = F$$



Wumpus SAT

Knowledge Base:

- T
- T
- T
- T
- T

Set $PB1 = F$.

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = F$$



Wumpus SAT

Knowledge Base:

- T
- T
- T
- T
- T

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = F$$

All clauses T! The expression is satisfiable.

Wumpus SAT

Knowledge Base:

- T
- T
- T
- T
- T

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = F$$

There exists some possible world in which $\neg PA2$.

Wumpus SAT

Knowledge Base:

- T
- T
- T
- T
- T

Variables:

$$BA1 = F$$

$$PA2 = F$$

$$PB1 = F$$

We fail to conclude $PA2$.



Using SAT for Inference

Query: $PA1$

$KB + \neg PA1$ is satisfiable. We cannot conclude $PA1$.

Query: $\neg PA1$

$KB + PA1$ is not satisfiable. We can conclude $\neg PA1$.

$\neg PA1$ is entailed by the knowledge base.

Improving the SAT Solver

Begin with every variable's value unassigned.

To find a model which satisfies a CNF expression:

 If every clause is true, return true.

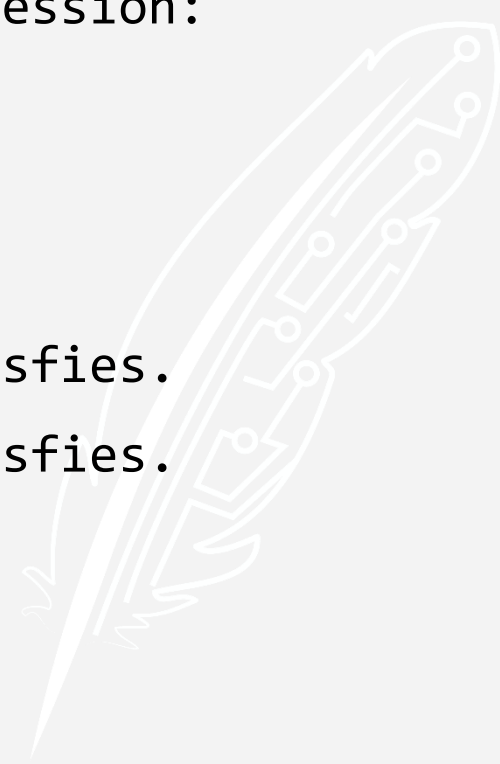
 If any clause is empty, return false.

 Choose an unassigned variable V .

 Set $V=T$. Try to find a model that satisfies.

 Set $V=F$. Try to find a model that satisfies.

 Return false.



Improving the SAT Solver

Begin with every variable's value unassigned.

To find a model which satisfies a CNF expression:

If every clause is true, return true.

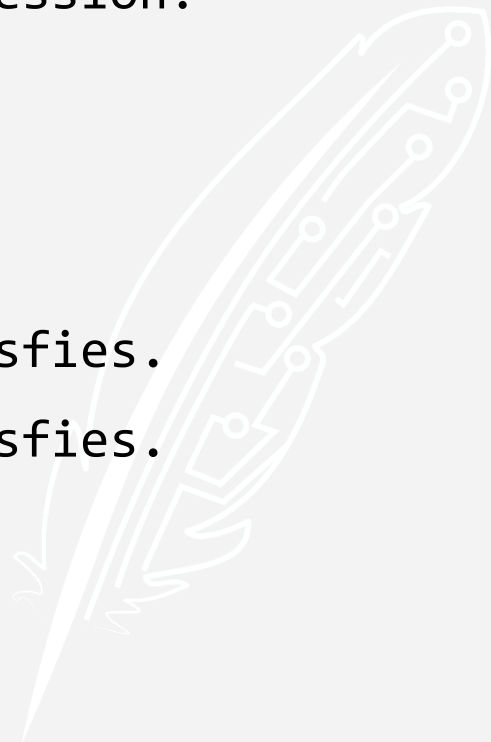
If any clause is empty, return false.

Choose an unassigned variable V .

Set $V=T$. Try to find a model that satisfies.

Set $V=F$. Try to find a model that satisfies.

Return false.



Which variable to choose?

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg \mathbf{BA1}$
- $\neg PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

Should we try assigning $BA1 = T$?



Which variable to choose?

Knowledge Base:

- $\neg BA1 \vee PA2 \vee PB1$
- $BA1 \vee \neg PA2$
- $BA1 \vee \neg PB1$
- $\neg BA1$
- $\neg PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

BA1 is called a “unit clause” and can only be F.

Which variable to choose?

Knowledge Base:

- $X \vee Y \vee Z$
- $\neg Y \vee Z$

Variables:

$X = ?$

$Y = ?$

$Z = ?$

Should we try assigning $Z = F$?



Which variable to choose?

Knowledge Base:

- $X \vee Y \vee Z$
- $\neg Y \vee Z$

Variables:

$X = ?$

$Y = ?$

$Z = ?$

Z is called a “pure symbol” because it always appears with the same valence. In other words, we only ever see Z , and never see $\neg Z$.

There is no reason to bother trying $Z = F$.

DPLL Algorithm

Begin with every variable's value unassigned.

To find a model which satisfies a CNF expression:

Simplify the model using unit propagation.

Simplify the model using pure symbols.

If every clause is true, return true.

If any clause is empty, return false.

Choose an unassigned variable V .

Set $V=T$. Try to find a model that satisfies.

Set $V=F$. Try to find a model that satisfies.

Return false.

Component Analysis

Knowledge Base:

- $A \vee B \vee C$
- $\neg A \vee B$
- $X \vee Y \vee Z$
- $\neg Y$

Variables:

$A = ?$

$B = ?$

$C = ?$

$X = ?$

$Y = ?$

$Z = ?$

Components can be solved separately, in parallel.

Borrowing from CSPs

- Variable and value ordering
- Intelligent backtracking and backjumping
- Local search



Review: Local Search

Traditional Search:

- Start with an empty solution.
- At each step, add one thing to the solution.
- Return solution on success; backtrack on fail.

Local Search:

- Start with a random (probably bad) solution.
- At each step, make one change.
- Run until solution found or out of time.

Local Search SAT Solver

Randomly assign T or F to every variable.

Until you run out of time:

 If every clause is true, return solution.

 Choose a variable V .

 Flip the variable ($T \rightarrow F$ or $F \rightarrow T$).



Local Search SAT Solver

Randomly assign T or F to every variable.

Until you run out of time:

If every clause is true, return solution.

Choose a variable V .

Flip the variable ($T \rightarrow F$ or $F \rightarrow T$).



Review: Local Search

Local Search:

- Start with a random (probably bad) solution.
- At each step, make one change.
 - Usually make a change that improves the solution.
 - Sometimes make a random change.
 - Restart if stuck in a local maxima.
- Run until solution found or out of time.

Review: Local Search

Local Search:

- Start with a random (probably bad) solution.
- At each step, make one change.
 - Usually make a change that **improves the solution**.
 - Sometimes make a random change.
 - Restart if stuck in a local maxima.
- Run until solution found or out of time.

Solution Quality (Utility)

How do we measure which solutions to a SAT problem are better than others?

If a problem has n clauses, then a solution has 0 unsatisfied clauses and n satisfied clauses.

The number of satisfied clauses is a good measure of how good a solution is. In other words, solutions where more clauses are satisfied are better than ones where fewer clauses are satisfied.

GSAT Algorithm

Let n be the "noise parameter," where $0 < n < 1$.

Randomly assign T or F to every variable.

Until you run out of time:

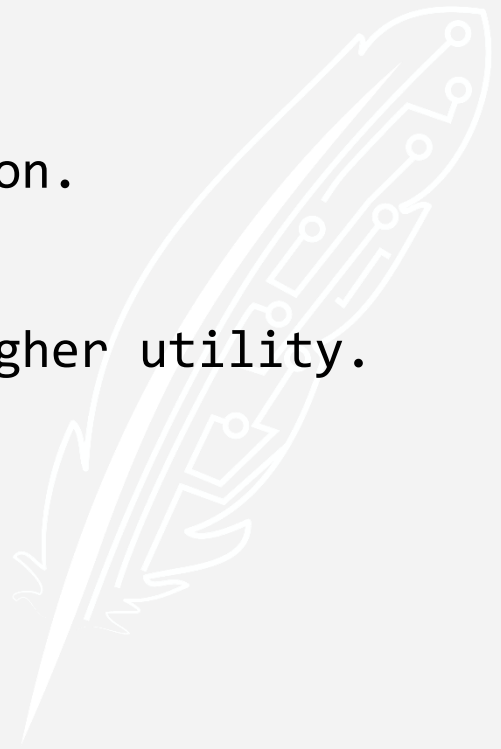
- If every clause is true, return solution.

- With probability $p < n$:

 - Flip the variable that leads to higher utility.

- Else:

 - Flip a random variable.



GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=? B=?`

Expression: `(and (or ? ?) (or (not ?) ?))`

Start with a random assignment of values.

GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=T B=F`

Expression: `(and (or T F) (or (not T) F))`


Start with a random assignment of values.

GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=T B=F`

Expression: `(and (or T F) (or (not T) F))`



Is this a solution? satisfied not satisfied

No, so we need to choose a variable to flip.

GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=T B=F`

Expression: `(and (or T F) (or (not T) F))`

Assume the noise parameter is $n = 0.25$.

25% of the time, we choose a variable at random.

We generate a random number $p = 0.13$ and $p < n$, so we pick a variable at random, and we get **A**.

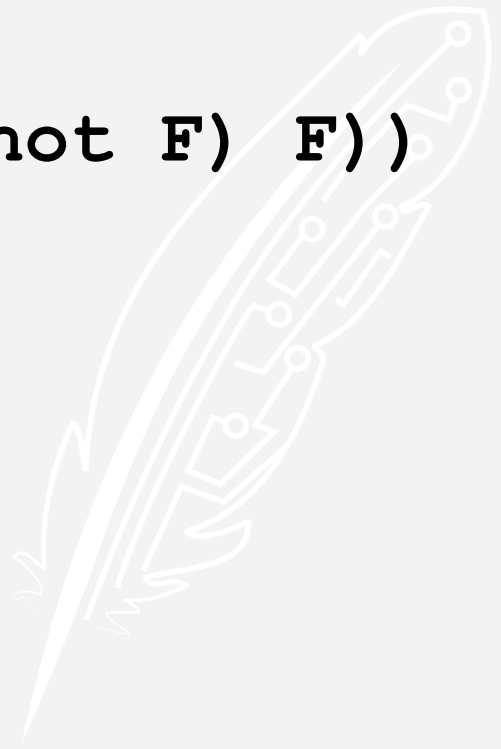
GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=F B=F`

Expression: `(and (or F F) (or (not F) F))`

Flip **A** from **T** to **F**.




GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=F B=F`

Expression: `(and (or F F) (or (not F) F))`



Is this a solution? not satisfied satisfied

No, so we need to choose a variable to flip.

GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=F B=F`

Expression: `(and (or F F) (or (not F) F))`

We generate a random number $p = 0.82$ and $p > n$, so we will flip the variable that results in the most clauses being satisfied.

GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=F B=F`

Expression: `(and (or F F) (or (not F) F))`

If we flip **A** from **F** to **T**, how many will be satisfied? 1

If we flip **B** from **F** to **T**, how many will be satisfied? 2

So **B** is the better choice.

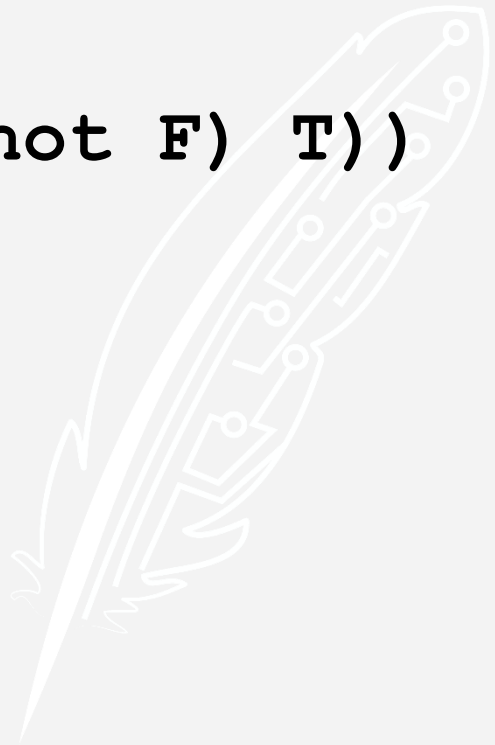
GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=F B=T`

Expression: `(and (or F T) (or (not F) T))`

Flip **B** from **F** to **T**.




GSAT Example

Problem: `(and (or A B) (or (not A) B))`

Solution: `A=F B=T`

Expression: `(and (or F T) (or (not F) T))`



Is this a solution? satisfied satisfied

Yes!

Improving GSAT

Two features of GSAT we would like to improve:

- Consider flipping every variable at every step.
- Random moves are a little too random.



WalkSAT Algorithm

Let n be the "noise parameter," where $0 < n < 1$.

Randomly assign T or F to every variable.

Until you run out of time:

- If every clause is true, return solution.

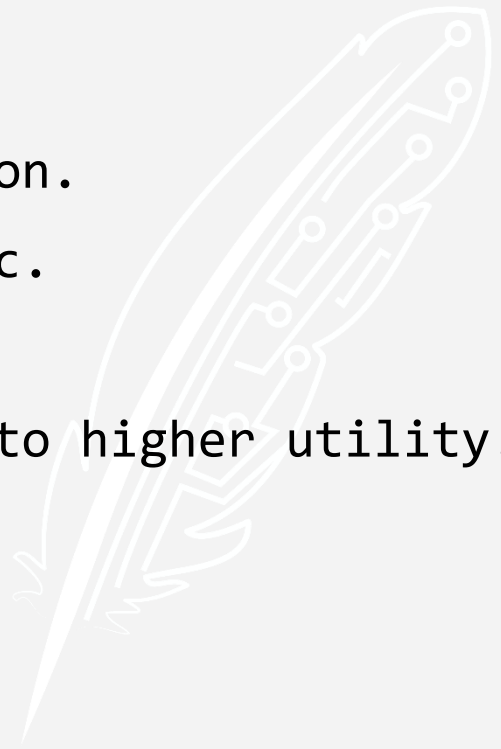
- Randomly choose an unsatisfied clause c .

- With probability $p < n$:

 - Flip the variable in c that leads to higher utility.

- Else:

 - Flip a random variable in c .



GSAT vs. WalkSAT

- Algorithms are very similar. The only difference is how they choose which variable to flip.
- GSAT considers every variable every time.
- WalkSAT first chooses an unsatisfied clause, then chooses a variable in that clause.
- WalkSAT has a chance of improving the solution even when making a random move.