Model Checking

Stephen G. Ware

CS 463G

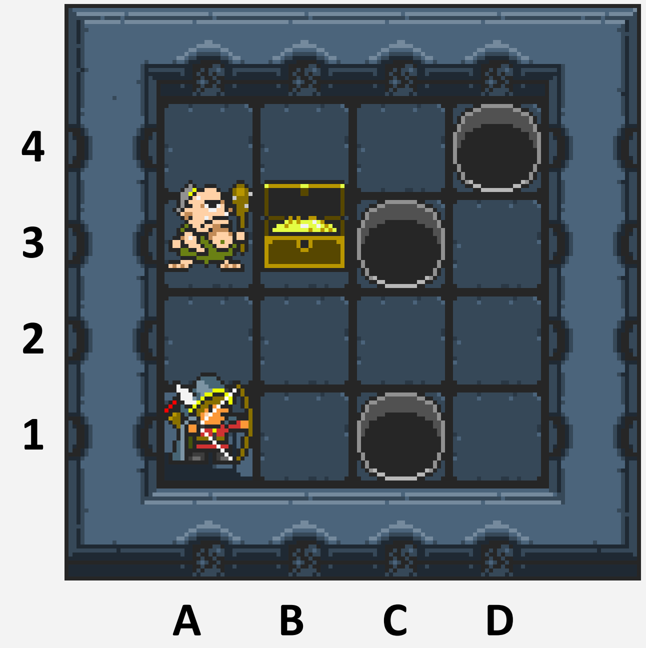
Narrative Intelligence LAB

University of Kentucky

Logical Inference

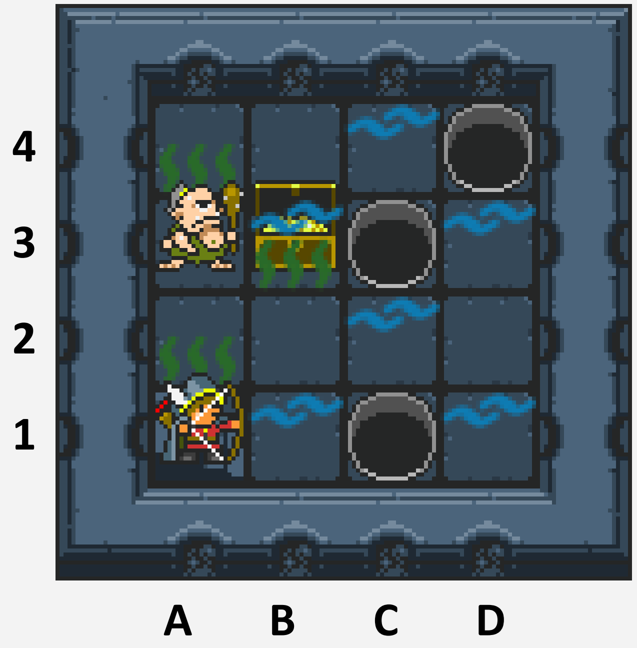
Given a **knowledge base** of known facts, can we derive new facts? Such facts are said to be **entailed** by the knowledge base.

Wumpus World



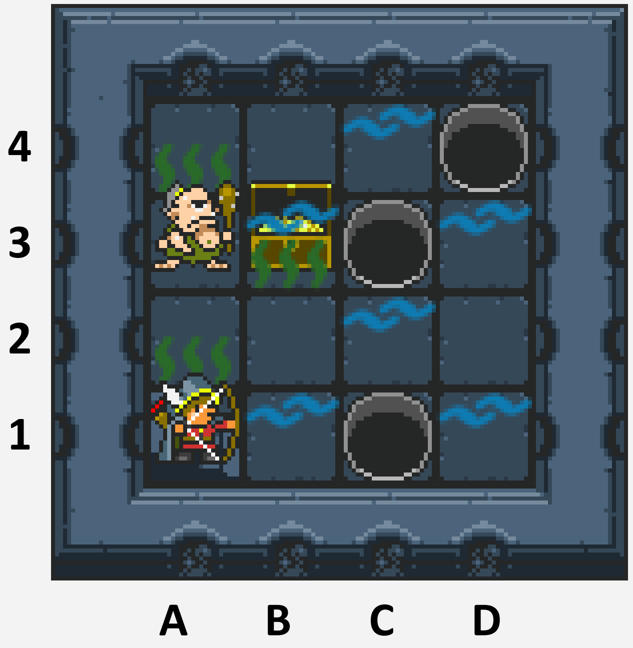
- The player, who can navigate the grid world
- The wumpus, who will eat the player if the player blunders into its square
- Bottomless pits
- A chest of gold

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:

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

… and so on

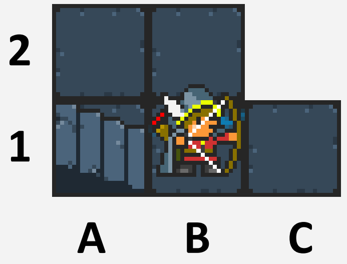
- $\neg BA1$
- $BB1$

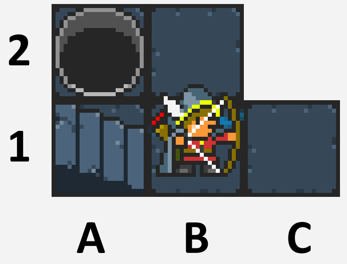
Events:

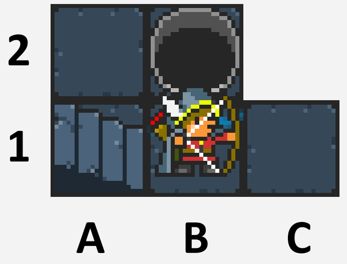
The player begins at square A1 and senses no breeze.

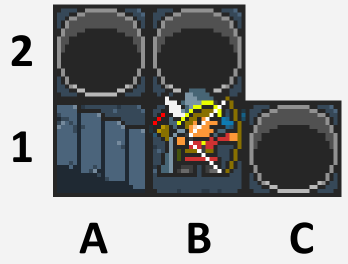
The player moves to square B1 and senses a breeze.

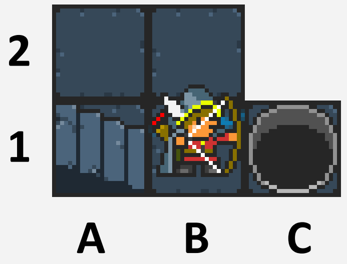
Possible Worlds

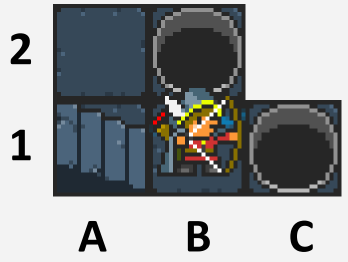


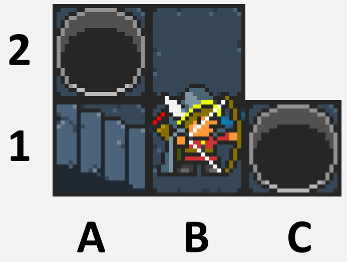


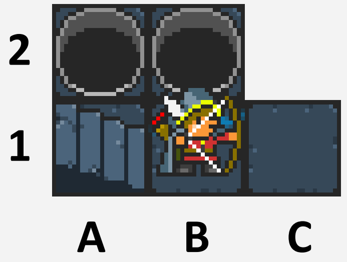












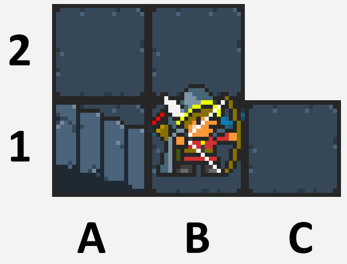
Knowledge Base:

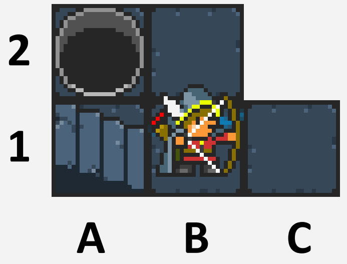
- $BA1 \leftrightarrow PA2 \lor PB1$
- $BB1 \leftrightarrow PB2 \lor PC1$

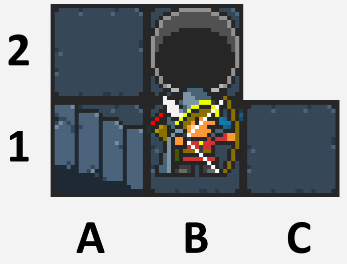
… and so on

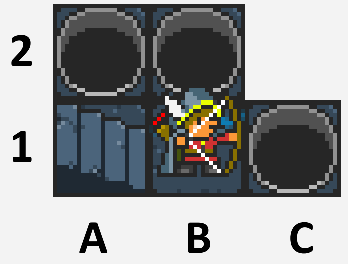
- $\neg BA1$
- $BB1$

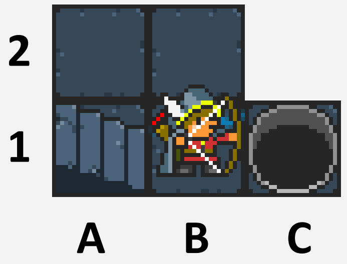
Possible Worlds

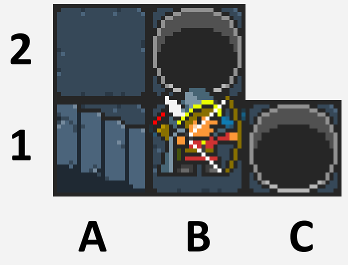


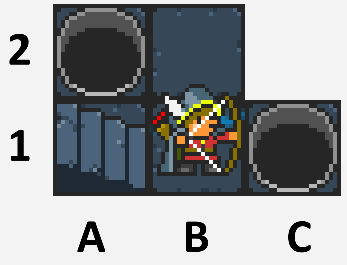


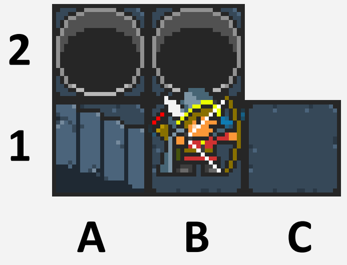












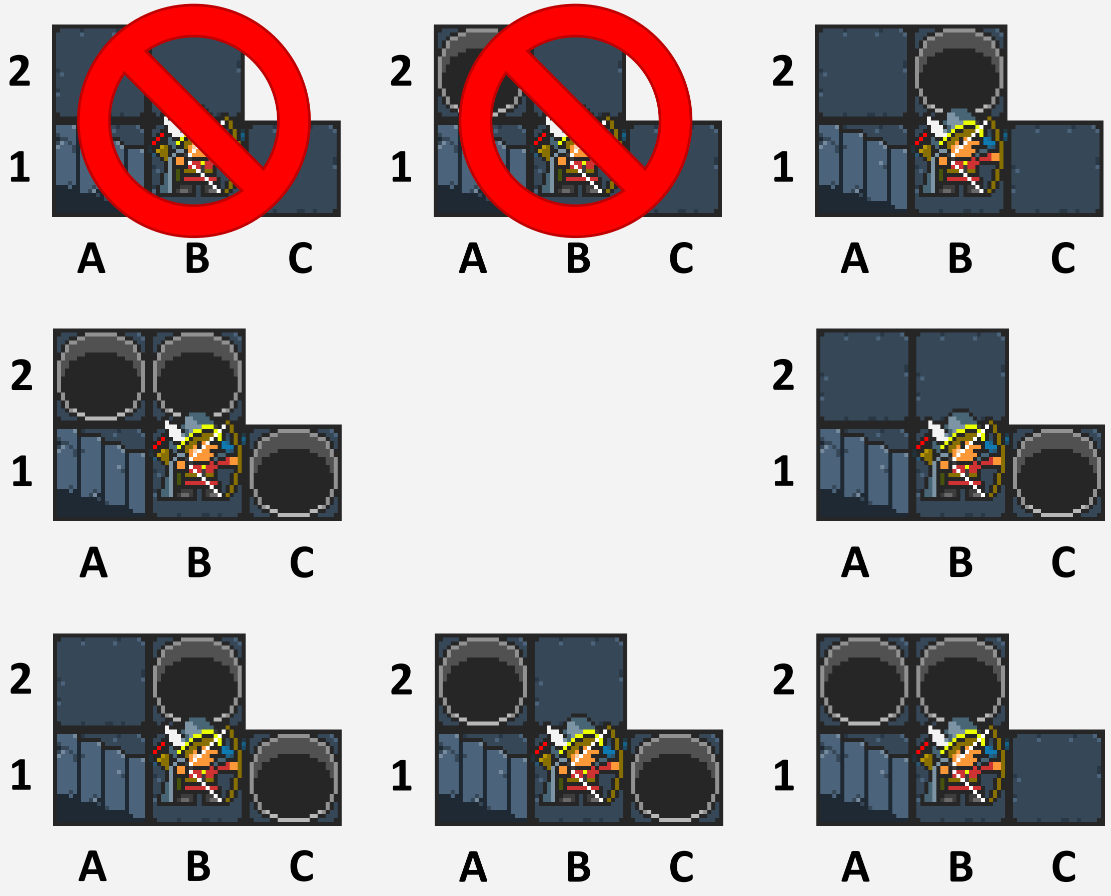
Knowledge Base:

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

… and so on

- $\neg BA1$
- $BB1$

Possible Worlds



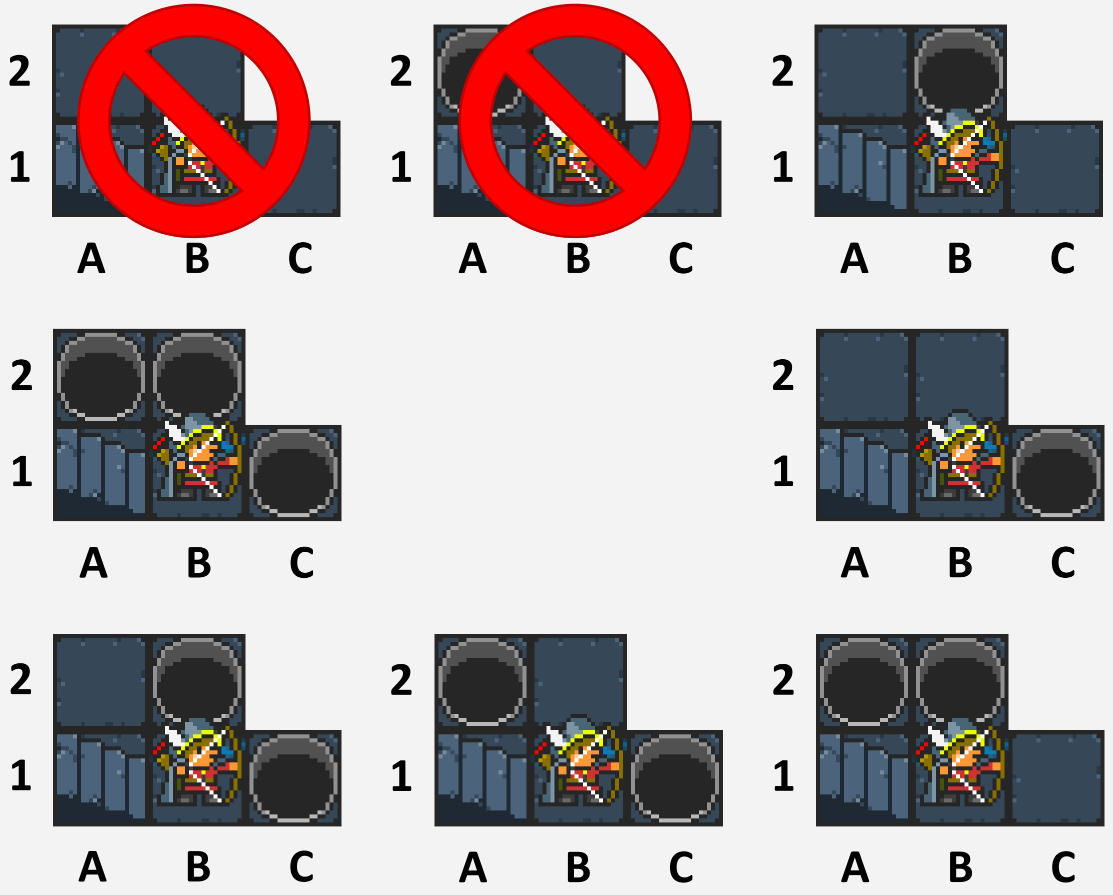
Knowledge Base:

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

… and so on

- $\neg BA1$
- $BB1$

Possible Worlds



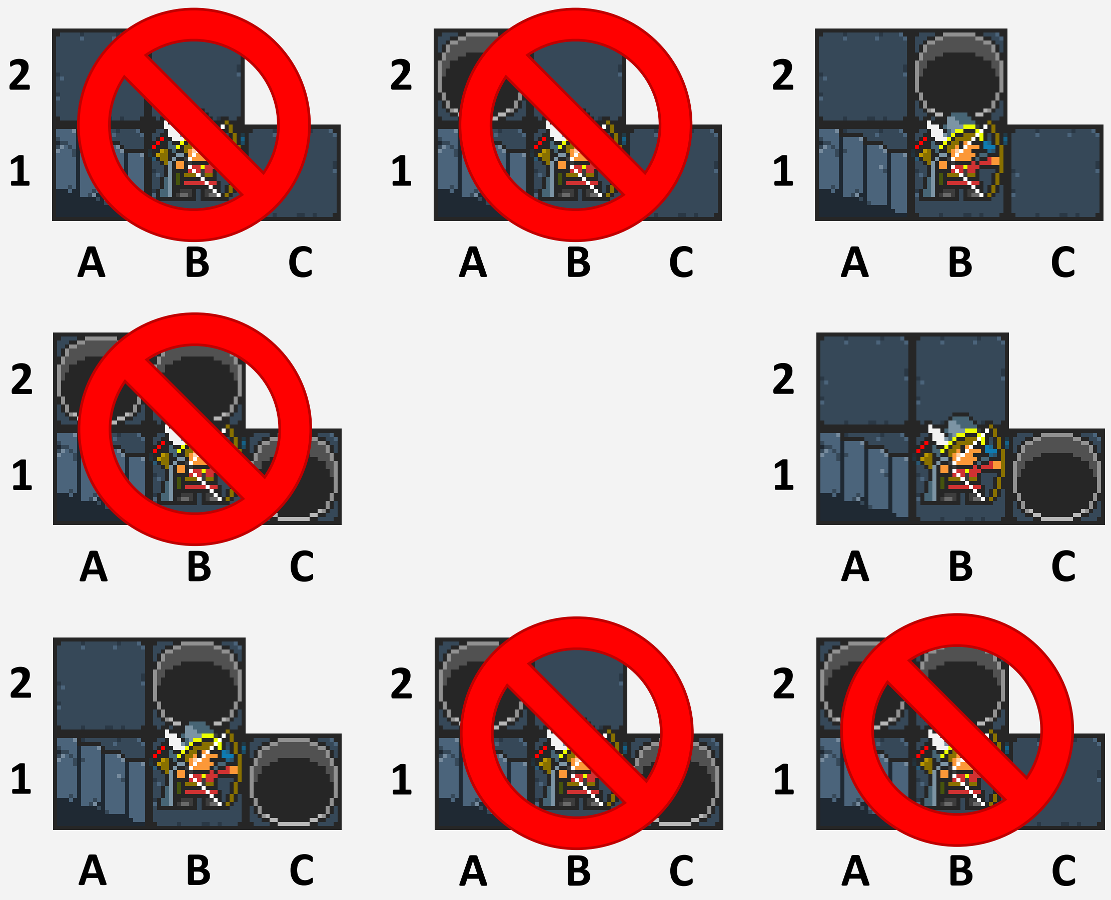
Knowledge Base:

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

… and so on

- $\neg BA1$
- $BB1$

Possible Worlds



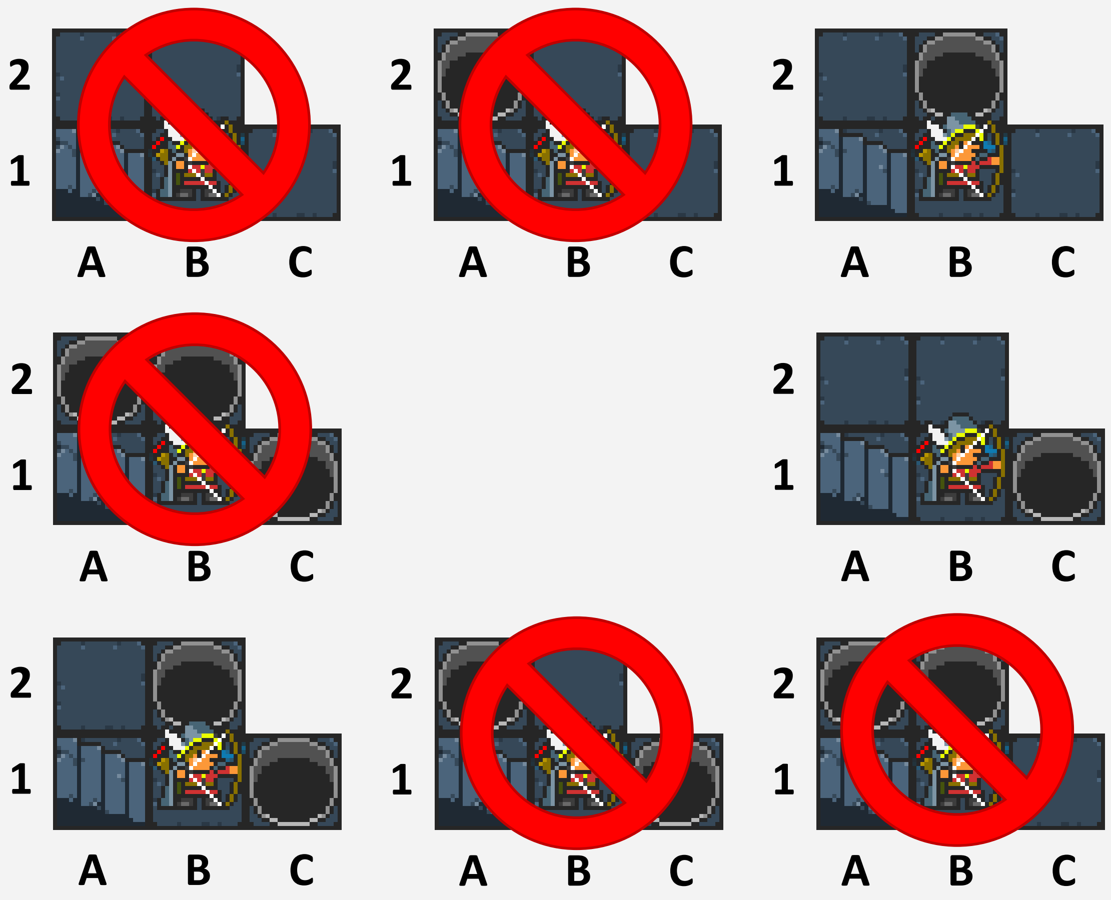
Knowledge Base:

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

… 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:

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

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

Wumpus in CNF

Knowledge Base:

$$\big((BA1 \rightarrow PA2 \vee PB1) \wedge (PA2 \vee PB1 \rightarrow BA1)\big) \wedge$$
$$\big((BB1 \rightarrow PB2 \vee PC1) \wedge (PB2 \vee PC1 \rightarrow BB1)\big)$$

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

Wumpus in CNF

Knowledge Base:

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

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

Wumpus in CNF

Knowledge Base:

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

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

Wumpus in CNF

Knowledge Base:

$(\neg BA1 \lor PA2 \lor PB1) \land$

$(\neg(PA2 \lor PB1) \lor BA1) \land$

$(\neg BB1 \lor PB2 \lor PC1) \land$

$(\neg(PB2 \lor PC1) \lor BB1)$

Remember: $\neg(x \lor y) \quad \leftrightarrow \quad (\neg x \land \neg y)$

Wumpus in CNF

Knowledge Base:

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

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

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$\big((\neg PB2 \wedge \neg PC1) \vee BB1\big)$

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

Wumpus in CNF

Knowledge Base:

$(\neg BA1 \lor PA2 \lor PB1) \land$

$\big((BA1 \lor \neg PA2) \land (BA1 \lor \neg PB1)\big) \land$

$(\neg BB1 \lor PB2 \lor PC1) \land$

$\big((BB1 \lor \neg PB2) \land (BB1 \lor \neg PC1)\big)$

Remember: $x \land (y \land z) \quad \leftrightarrow \quad x \land y \land z$

Wumpus in CNF

Knowledge Base:

$(\neg BA1 \lor PA2 \lor PB1) \land$
$(BA1 \lor \neg PA2) \land$
$(BA1 \lor \neg PB1) \land$
$(\neg BB1 \lor PB2 \lor PC1) \land$
$(BB1 \lor \neg PB2) \land$
$(BB1 \lor \neg PC1)$

Wumpus in CNF

Knowledge Base:

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

Each of these is a disjunctive clause.

Wumpus in CNF

Knowledge Base:

$$\left.\begin{array}{l} (\neg BA1 \lor PA2 \lor PB1) \land \\ (BA1 \lor \neg PA2) \land \\ (BA1 \lor \neg PB1) \land \\ (\neg BB1 \lor PB2 \lor PC1) \land \\ (BB1 \lor \neg PB2) \land \\ (BB1 \lor \neg PC1) \end{array}\right\}$$

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:

$(\neg BA1 \lor PA2 \lor PB1) \land$
$(BA1 \lor \neg PA2) \land$
$(BA1 \lor \neg PB1) \land$
$(\neg BB1 \lor PB2 \lor PC1) \land$
$(BB1 \lor \neg PB2) \land$
$(BB1 \lor \neg PC1)$

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:

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

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:

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

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:

$(\neg F \lor PA2 \lor PB1) \land$
$(F \lor \neg PA2) \land$
$(F \lor \neg PB1) \land$
$(\neg BB1 \lor PB2 \lor PC1) \land$
$(BB1 \lor \neg PB2) \land$
$(BB1 \lor \neg PC1)$

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:

$(T \vee PA2 \vee PB1) \wedge$

$(F \vee \neg PA2) \wedge$

$(F \vee \neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \wedge$

$(F \vee \neg PA2) \wedge$

$(F \vee \neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \wedge$
$(F \vee \neg PA2) \wedge$
$(F \vee \neg PB1) \wedge$
$(\neg BB1 \vee PB2 \vee PC1) \wedge$
$(BB1 \vee \neg PB2) \wedge$
$(BB1 \vee \neg PC1)$

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:

$T \wedge$

$(\neg PA2) \wedge$

$(\neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \wedge$

$(\neg PA2) \wedge$

$(\neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \wedge$

$(\neg T) \wedge$

$(\neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \land$

$F \land$

$(\neg PB1) \land$

$(\neg BB1 \lor PB2 \lor PC1) \land$

$(BB1 \lor \neg PB2) \land$

$(BB1 \lor \neg PC1)$

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:

$T \wedge$

$F \wedge$

$(\neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \land$

$(\neg PA2) \land$

$(\neg PB1) \land$

$(\neg BB1 \lor PB2 \lor PC1) \land$

$(BB1 \lor \neg PB2) \land$

$(BB1 \lor \neg PC1)$

Variables:

$BA1 = F$

$BB1 =?$

$PA2 =?$

$PB1 =?$

$PB2 =?$

$PC1 =?$

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

Wumpus in CNF

Knowledge Base:

$T \wedge$

$(\neg F) \wedge$

$(\neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \wedge$

$T \wedge$

$(\neg PB1) \wedge$

$(\neg BB1 \vee PB2 \vee PC1) \wedge$

$(BB1 \vee \neg PB2) \wedge$

$(BB1 \vee \neg PC1)$

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:

$T \wedge$

$T \wedge$

$(\neg F) \wedge$

$(\neg F \vee F \vee F) \wedge$

$(F \vee \neg F) \wedge$

$(F \vee \neg F)$

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.

Wumpus in CNF

Knowledge Base:

$T \wedge$

$T \wedge$

$(T) \wedge$

$(T \vee F \vee F) \wedge$

$(F \vee T) \wedge$

$(F \vee T)$

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.

Wumpus in CNF

Knowledge Base:

$T \wedge$
$T \wedge$
$T \wedge$
$T \wedge$
$T \wedge$
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.

Wumpus in CNF

Knowledge Base:

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

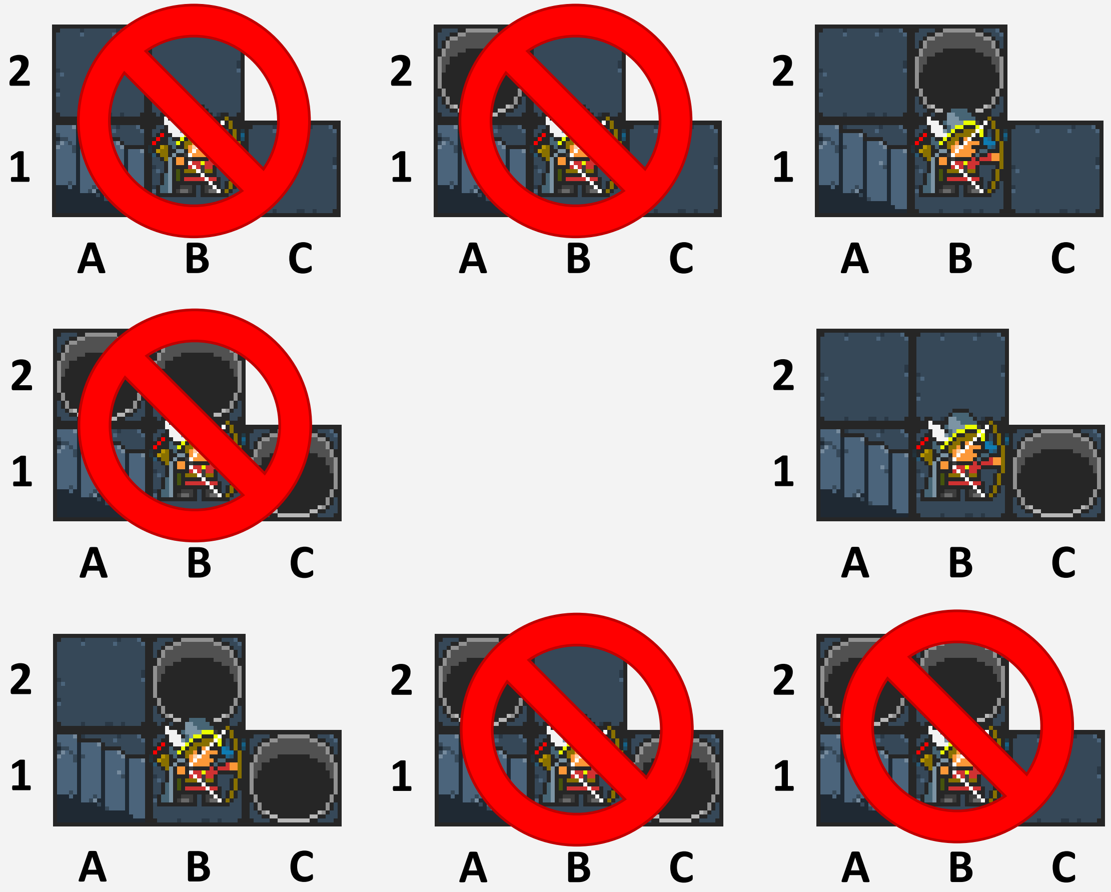
Given a knowledge base (which is known to be satisfiable) and some fact, how can we tell if that fact is entailed?

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.

Possible Worlds



Knowledge Base:

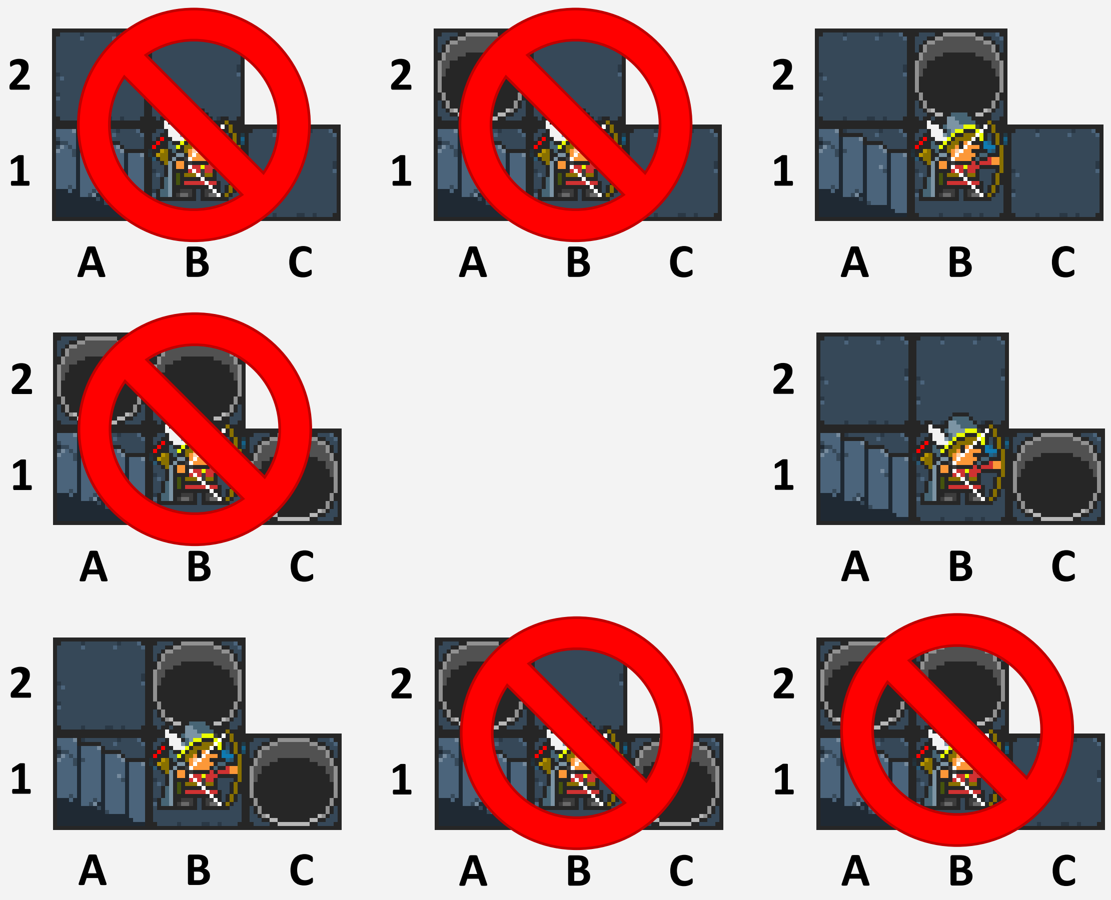
- $BA1 \leftrightarrow PA2 \lor PB1$
- $BB1 \leftrightarrow PB2 \lor PC1$

Query:

- $PB2$

It *might* be true, but also it might not be.

Possible Worlds



Knowledge Base:

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

Query:

- $\neg PA2$

It is true in all possible worlds!

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:

- There is a breeze at A1 if and only if there is a pit at A2 or a pit at B1. $(BA1 \leftrightarrow PA2 \lor PB1) \land$
- There is no breeze at A1. $\neg BA1$

Query:

There is no pit in square A2. $\neg PA2$

Wumpus SAT

Knowledge Base:

- There is a breeze at A1 if and only if there is a pit at A2 or a pit at B1. $(BA1 \leftrightarrow PA2 \lor PB1) \land$
- There is no breeze at A1. $\neg BA1$

Negated Query:

There is a pit in square A2. $PA2$

Wumpus SAT

Knowledge Base:

- There is a breeze at A1 if and only if there is a pit at A2 or a pit at B1. $(BA1 \leftrightarrow PA2 \lor PB1) \land$
- There is no breeze at A1. $\neg BA1 \land$
- There is a pit in square A2. $PA2$

We add the negated query to the knowledge base and hope that it is unsatisfiable.

Wumpus SAT

Knowledge Base:

- There is a breeze at A1 if and only if there is a pit at A2 or a pit at B1. $(BA1 \leftrightarrow PA2 \lor PB1) \land$
- There is no breeze at A1. $\neg BA1 \land$
- There is a pit in square A2. $PA2$

In other words, we can prove $\neg PA2$ by showing that it can't possibly be the case that $PA2$.

Wumpus SAT

Knowledge Base:

$(BA1 \leftrightarrow PA2 \vee PB1) \wedge$

$\neg BA1 \wedge$

$PA2$

In other words, we can prove $\neg PA2$ by showing that it can't possibly be the case that $PA2$.

Wumpus SAT

Knowledge Base:

$(BA1 \leftrightarrow PA2 \vee PB1) \wedge$

$\neg BA1 \wedge$

$PA2$

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

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

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

$\left(\neg BA1 \lor (PA2 \lor PB1)\right) \land$

$(\neg(PA2 \lor PB1) \lor BA1) \land$

$\neg BA1 \land$

$PA2$

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

$$\left(\neg BA1 \lor (PA2 \lor PB1)\right) \land$$
$$\left((\neg PA2 \land \neg PB1) \lor BA1\right) \land$$
$$\neg BA1 \land$$
$$PA2$$

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

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

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

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

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

$(\neg BA1 \lor PA2 \lor PB1) \land$

$(BA1 \lor \neg PA2) \land$

$(BA1 \lor \neg PB1) \land$

$\neg BA1 \land$

$PA2$

First, convert to conjunctive normal form.

Wumpus SAT

Knowledge Base:

$(\neg BA1 \lor PA2 \lor PB1) \land$

$(BA1 \lor \neg PA2) \land$

$(BA1 \lor \neg PB1) \land$

$\neg BA1 \land$

$PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$(\neg T \vee PA2 \vee PB1) \wedge$
$(T \vee \neg PA2) \wedge$
$(T \vee \neg PB1) \wedge$
$\neg T \wedge$
$PA2$

Variables:

$BA1 = T$
$PA2 = ?$
$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$(F \lor PA2 \lor PB1) \land$

$T \land$

$T \land$

$F \land$

$PA2$

Variables:

$BA1 = T$

$PA2 = ?$

$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

F

Variables:

$BA1 = T$

$PA2 = ?$

$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$(\neg BA1 \lor PA2 \lor PB1) \land$
$(BA1 \lor \neg PA2) \land$
$(BA1 \lor \neg PB1) \land$
$\neg BA1 \land$
$PA2$

Variables:

$BA1 = ?$
$PA2 = ?$
$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$(\neg F \lor PA2 \lor PB1) \land$

$(F \lor \neg PA2) \land$

$(F \lor \neg PB1) \land$

$\neg F \land$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$(T \lor PA2 \lor PB1) \land$

$(F \lor \neg PA2) \land$

$(F \lor \neg PB1) \land$

$T \land$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$T \wedge$

$\neg PA2 \wedge$

$\neg PB1 \wedge$

$T \wedge$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = ?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:	**Variables:**
$\neg PA2 \land$	$BA1 = F$
$\neg PB1 \land$	$PA2 =?$
$PA2$	$PB1 =?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \wedge$

$\neg T \wedge$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = T$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \wedge$

$F \wedge$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = T$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

F

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = T$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \land$
$\neg PB1 \land$
$PA2$

Variables:

$BA1 = F$
$PA2 =?$
$PB1 =?$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \land$

$\neg F \land$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \wedge$

$T \wedge$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \wedge$

$PA2$

Variables:

$BA1 = F$

$PA2 = ?$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg T \wedge$

T

Variables:

$BA1 = F$

$PA2 = T$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$F \land$

T

Variables:

$BA1 = F$

$PA2 = T$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

F

Variables:

$BA1 = F$

$PA2 = T$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg F \wedge$

F

Variables:

$BA1 = F$

$PA2 = F$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$T \land$

F

Variables:

$BA1 = F$

$PA2 = F$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

F

Variables:

$BA1 = F$

$PA2 = F$

$PB1 = F$

Now, check if this expression can be satisfied.
Remember: we hope that it cannot be!

Wumpus SAT

Knowledge Base:

$\neg PA2 \land$

$PA2$

Variables:

$BA1 = F$

$PA2 =?$

$PB1 = F$

There is no way to satisfy this expression!
We have proved there is no pit in A2.

Wumpus SAT

Knowledge Base:

$\neg PA2 \wedge$
$PA2$

Variables:

$BA1 = F$
$PA2 = ?$
$PB1 = F$

There is no way to satisfy this expression!
We have proved $\neg PA2$.

Wumpus SAT

Knowledge Base:

$\neg PA2 \wedge$

$PA2$

Variables:

$BA1 = F$

$PA2 =?$

$PB1 = F$

There is no way to satisfy this expression!

We have proved $\neg PA2$ because $PA2$ is impossible.

Using SAT for Inference

Query: $PA2$

KB $\wedge \neg PA2$ is satisfiable, so we cannot conclude $PA2$.

Query: $\neg PA2$

KB $\wedge$ $PA2$ is not satisfiable, so we can conclude $\neg PA2$.

$\neg PA2$ 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:

$(\neg BA1 \lor PA2 \lor PB1) \land$

$(BA1 \lor \neg PA2) \land$

$(BA1 \lor \neg PB1) \land$

$\neg BA1 \land$

$PA2$

Variables:

$BA1 = ?$

$PA2 = ?$

$PB1 = ?$

Should we try assigning $BA1 = T$?

Which variable to choose?

Knowledge Base:

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

Variables:

$BA1 = ?$
$PA2 = ?$
$PB1 = ?$

This is called a "unit clause" and $BA1$ can only be F.

Which variable to choose?

Knowledge Base:

$(X \lor Y \lor Z) \land$

$(\neg Y \lor Z)$

Variables:

$X = ?$

$Y = ?$

$Z = ?$

Is there any reason to try assigning $Z = F$?

Which variable to choose?

Knowledge Base:

$(X \lor Y \lor Z) \land$
$(\neg Y \lor Z)$

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 we 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 by assigning unit clauses.
    Simplify the model by assigning 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.
```

Which variable to choose?

Knowledge Base:

$(A \lor B \lor C) \land$

$(\neg A \lor B) \land$

$(X \lor Y \lor Z) \land$

$\neg Y$

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 → F or F → 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 → F or F → 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 a random variable.
    Else:
        Flip the variable that leads to higher utility.
```

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: $\texttt{(and}\ \underbrace{\texttt{(or T F)}}_{\text{satisfied}}\ \underbrace{\texttt{(or (not T) F)}}_{\text{not satisfied}}\texttt{)}$

Is this a solution?

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: $\texttt{(and }\underbrace{\texttt{(or F F)}}\ \underbrace{\texttt{(or (not F) F)}}\texttt{)}$

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))`

$$\texttt{(and } \underbrace{\texttt{(or F T)}}_{\text{satisfied}} \; \underbrace{\texttt{(or (not F) T)}}_{\text{satisfied}}\texttt{)}$$

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 a random variable in c.
    Else:
        Flip the variable in c that leads to higher utility.
```

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.