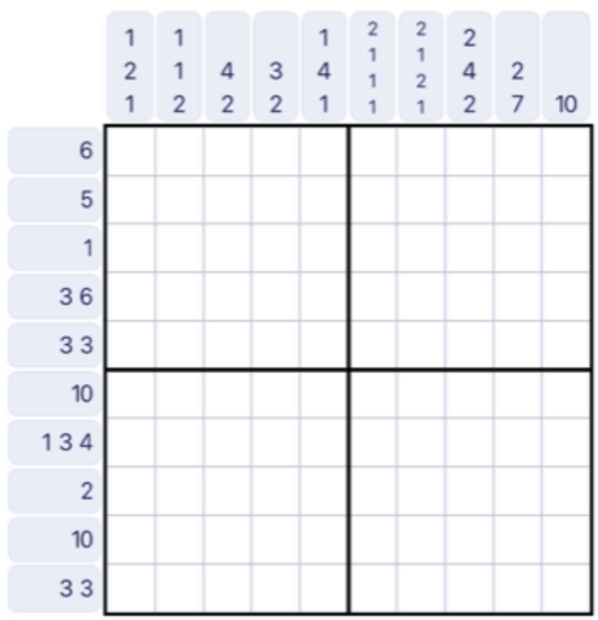
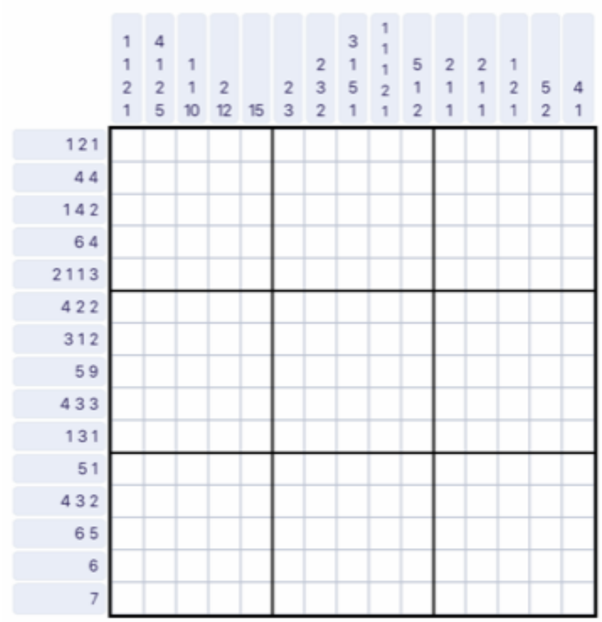
Warm-up:

Can you write these logic problems as a CSP?

What are the variables? the domains? the constraints?

What techniques could you use to solve them?





Announcements

Assignments:

- P2: Optimization
  - Due Sat 2/22, 10 pm
- HW5 out AFTER the Midterm
  - Due 2/25, 10 pm

Midterm 1 Exam

- Mon 2/17, in class
- Recitation Fri is a review session
- See Piazza post for details

AI: Representation and Problem Solving

Propositional Logic

Instructors: Pat Virtue & Stephanie Rosenthal

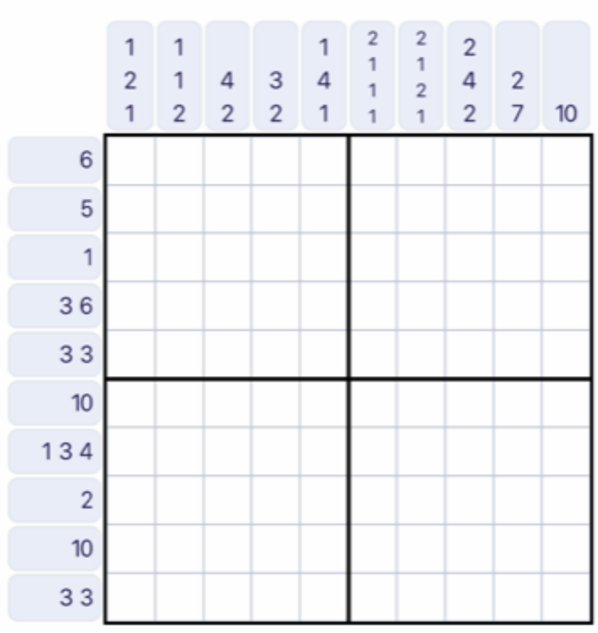
Slide credits: CMU AI, http://ai.berkeley.edu

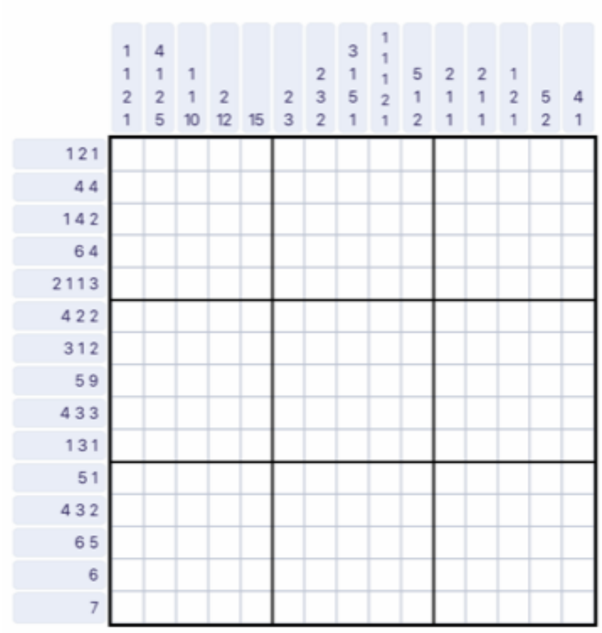
Warm-up:

Can you write these logic problems as a CSP?

What are the variables? the domains? the constraints?

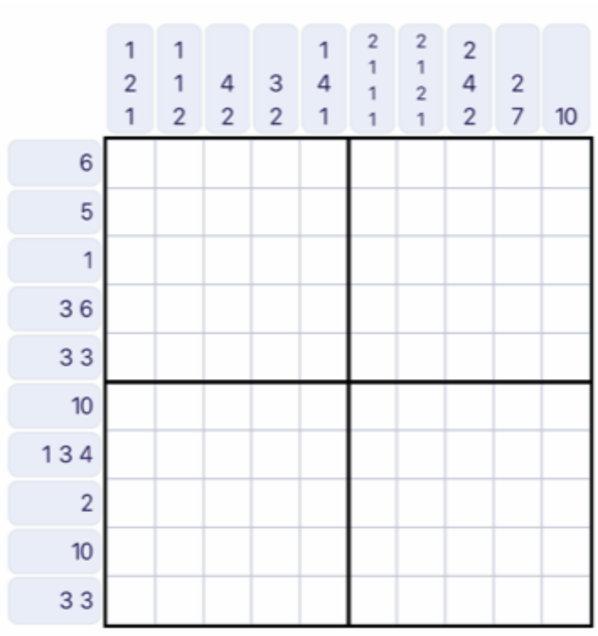
What techniques could you use to solve them?

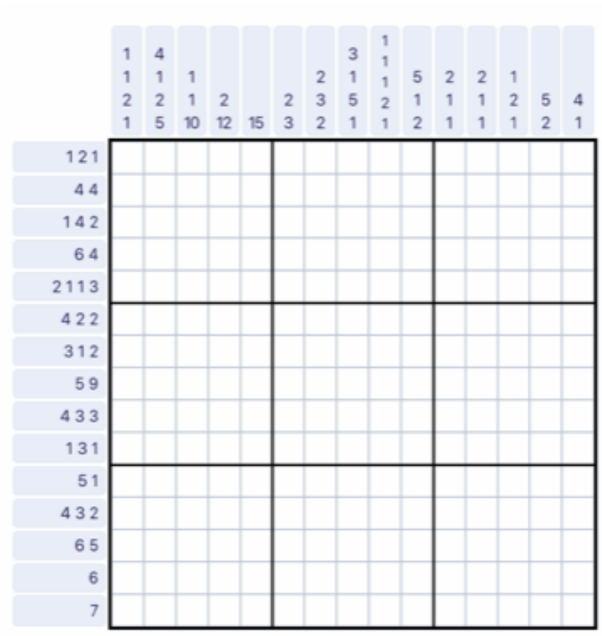




Warm-up:

Where is the knowledge held in CSPs? What is the goal of a CSP?

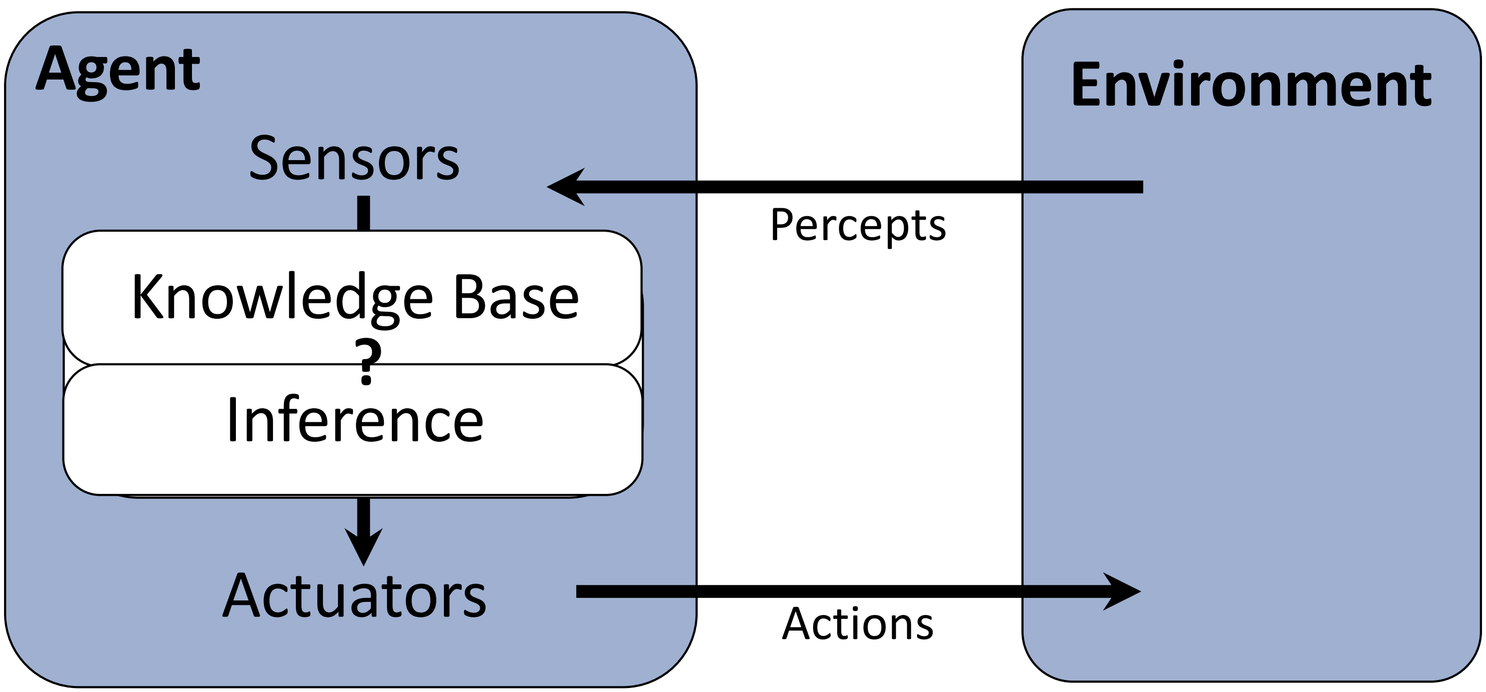




Logical Agents

What assignment of variables satisfies the constraints (knowledge base)?

What new knowledge can be inferred from the KB?



Logical Agents

So what do we tell our knowledge base (KB)?

- Facts (sentences)
    - The grass is green
    - The sky is blue
- Rules (sentences)
    - Eating too much candy makes you sick
    - When you're sick you don't go to school
- Percepts and Actions (sentences)
    - Pat ate too much candy today

What happens when we query the agent?

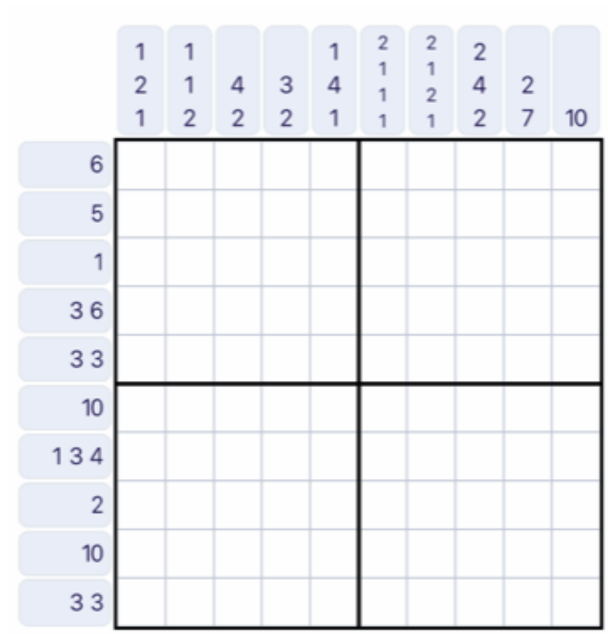
- Inference – new sentences created from old
    - Pat is not going to school today

Nonogram Puzzle

Logical Reasoning as a CSP

Binary variable for each square

Constraints:

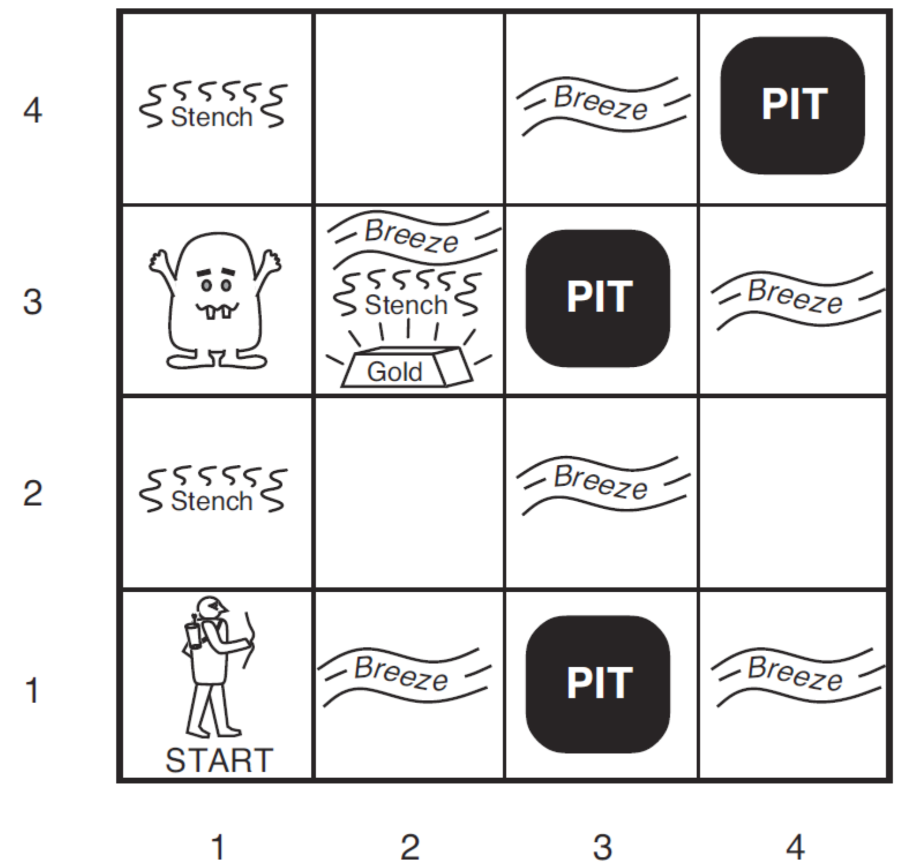


Wumpus World

Logical Reasoning as a CSP

Variables

- B_{ij} = breeze felt
- S_{ij} = stench smelt
- P_{ij} = pit here
- W_{ij} = wumpus here
- G = gold

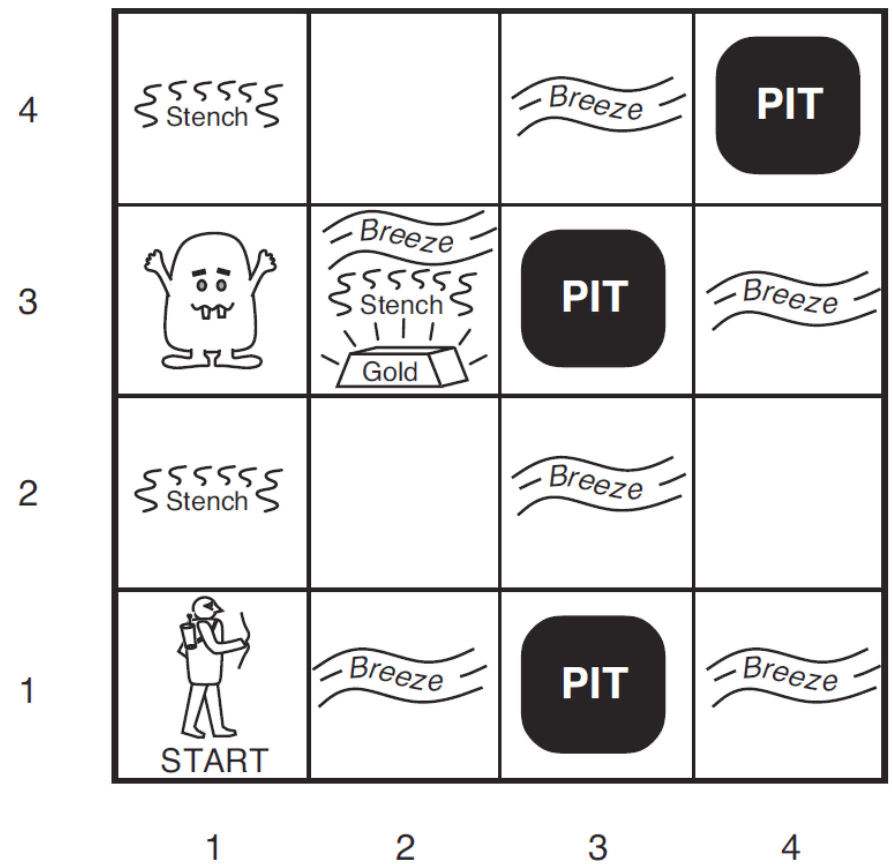


http://thiagodnf.github.io/wumpus-world-simulator/

Wumpus World

Constraints on Variables

- B_{ij} <=> >=1 neighbor is a pit
- S_{ij} <=> >=1 neighbor is wumpus
- P_{ij} <=> all NSEW neighbors B=T
- W_{ij} <=> all NSEW neighbors S=T
- G_{ij} <=> $!B_{ij}$ and $!S_{ij}$ and glitter



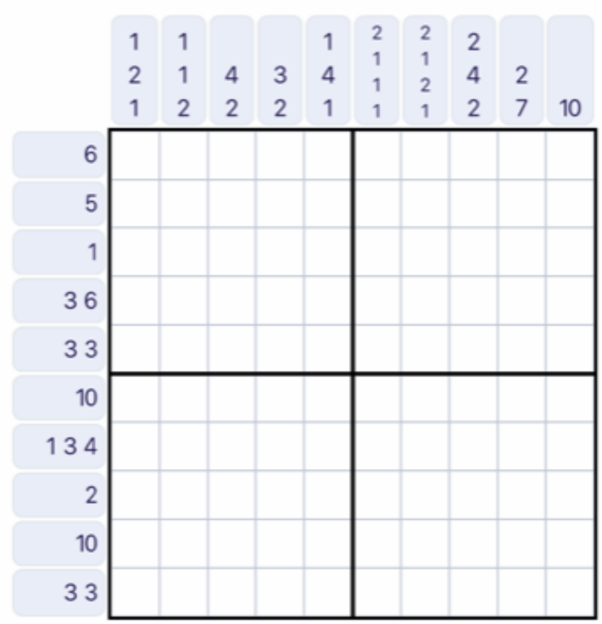
http://thiagodnf.github.io/wumpus-world-simulator/

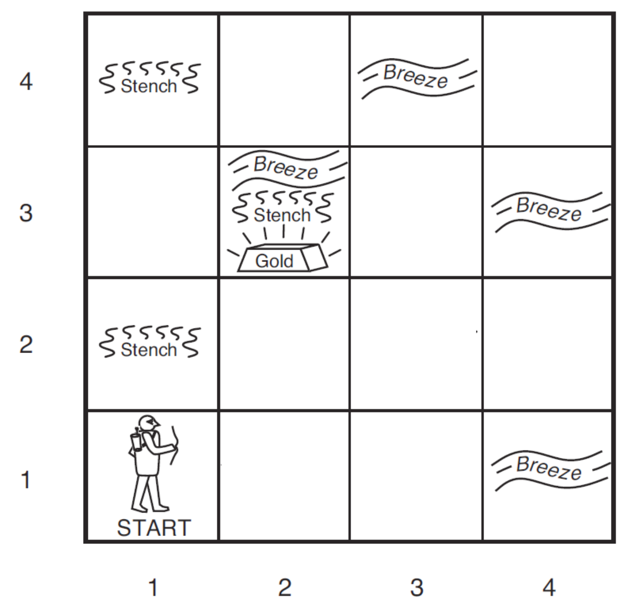
Worlds

We have a set of variables and constraints.

What are we trying to figure out?

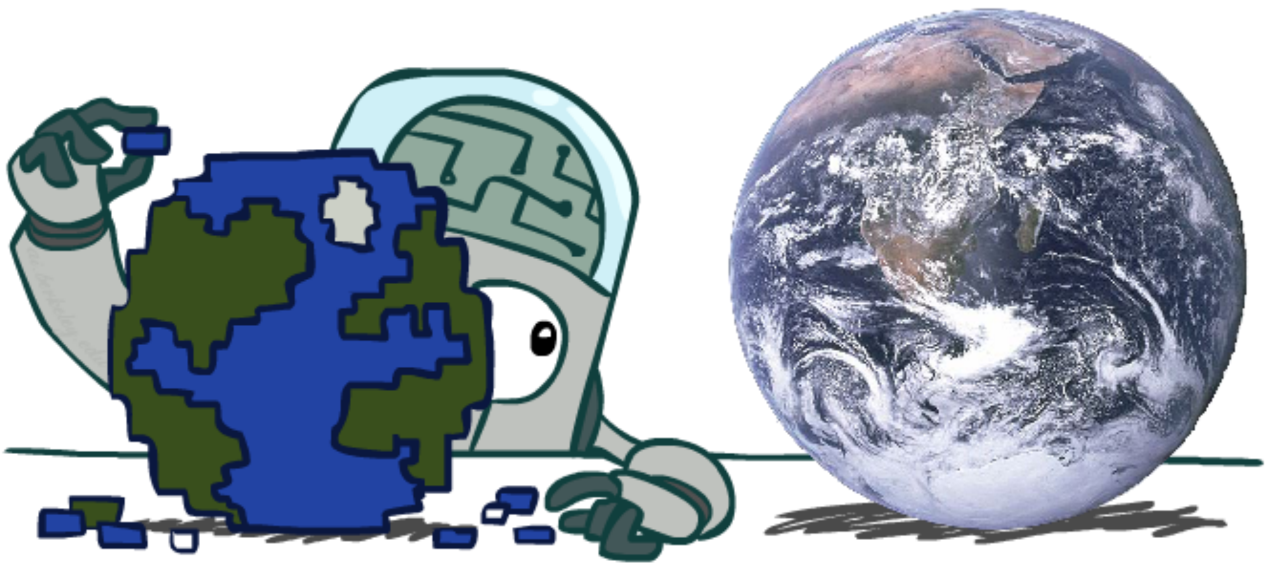
What worlds are possible given the information that we have?





Models

Assignments of values to variables



How do we represent possible worlds with models and knowledge bases?

How do we then do inference with these representations?

Wumpus World

World has 5 locations

[1,1], [2,1], [3,1], [1,2], [2,2]

Knowledge base

Nothing in [1,1]
Breeze in [2,1]

What do we know about the pit locations?

$P_{1,1} = F$
$P_{2,1} = F$
Everything else is unknown

Wumpus World

World has 5 locations

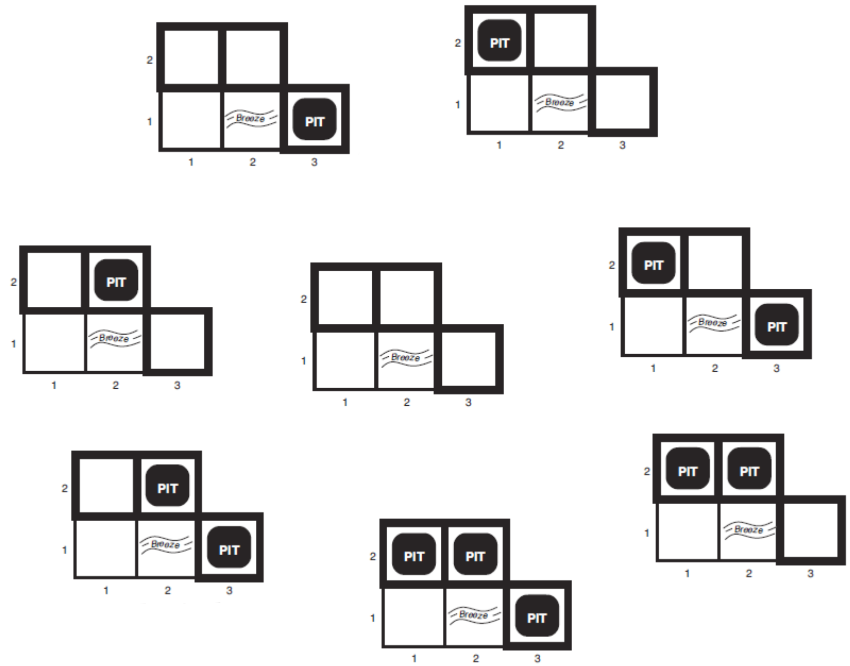
[1,1], [2,1], [3,1], [1,2], [2,2]

Knowledge base

Nothing in [1,1]
Breeze in [2,1]

Possible Models for Pits

$P_{1,1}$=F, $P_{2,1}$=F, $P_{1,2}$,$P_{2,2}$,$P_{3,1}$



Wumpus World

World has 5 locations

[1,1], [2,1], [3,1], [1,2], [2,2]

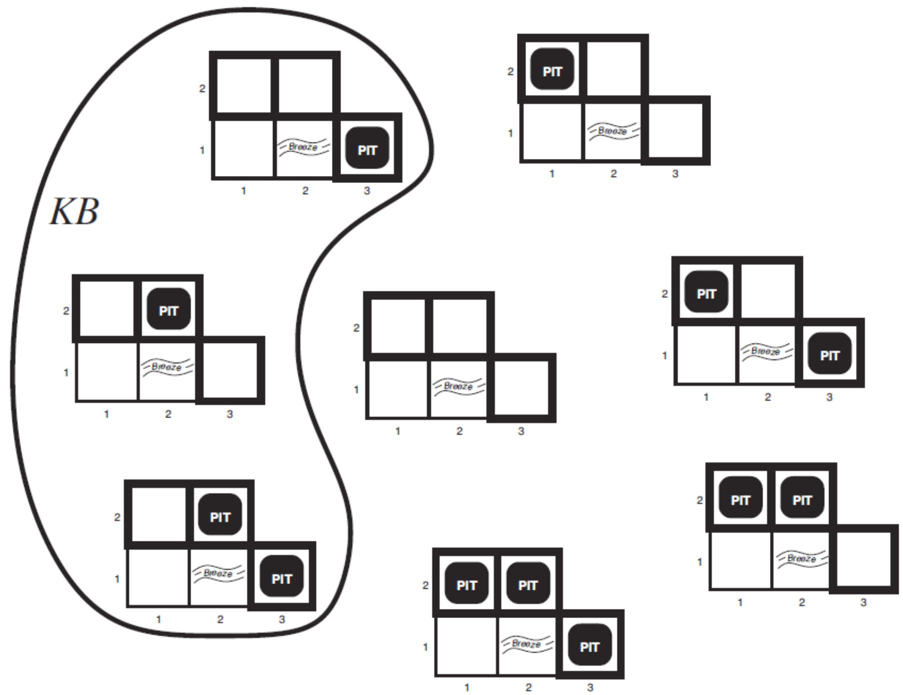
Knowledge base

Nothing in [1,1]

Breeze in [2,1]

Possible Models for Pits

$P_{1,1}$=F, $P_{2,1}$=F, $P_{1,2}$,$P_{2,2}$,$P_{3,1}$



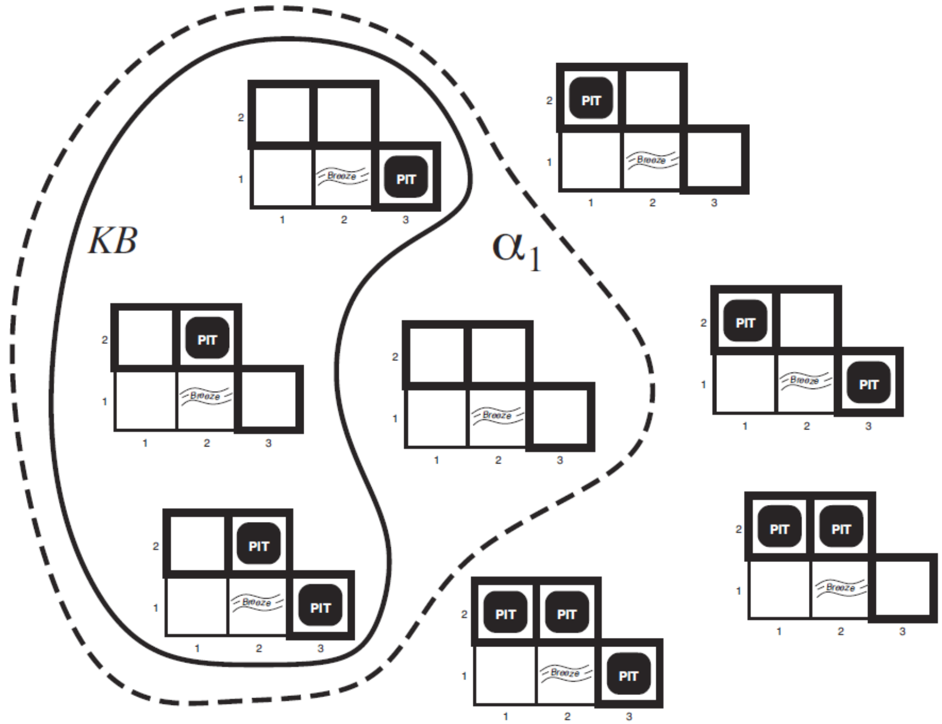
Using Knowledge base rules, infer some of these models aren't possible

possible worlds that could satisfy this KB are circled

Wumpus World

Possible Models

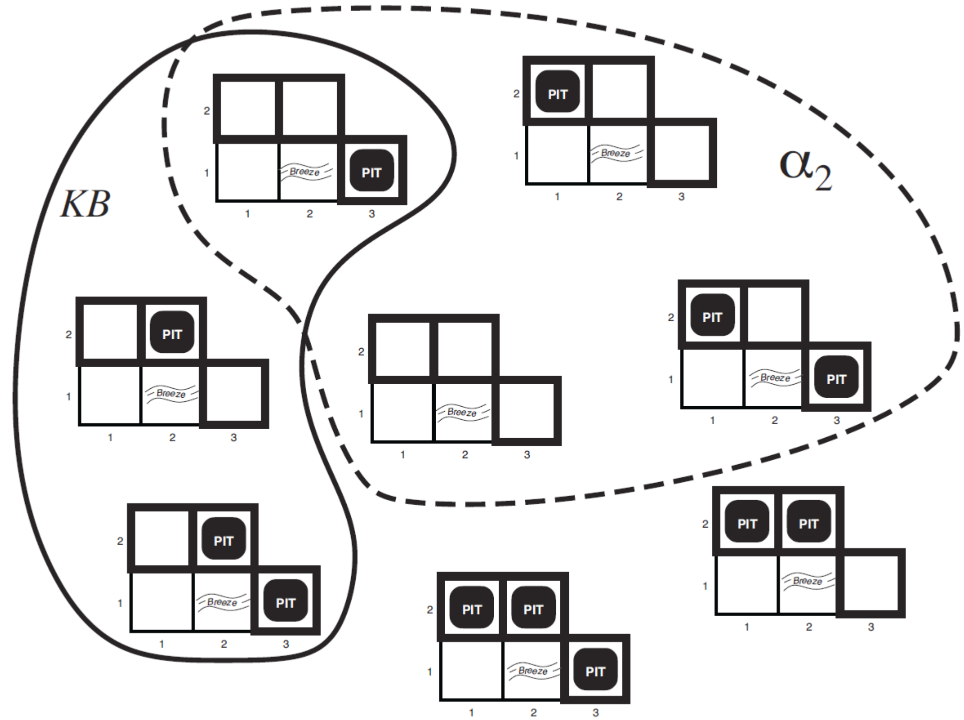
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_1:
  - No pit in [1,2]



Wumpus World

Possible Models

- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_2:
  - No pit in [2,2]



Role of Queries in Logical Agents

In both CSPs and logic, we can determine whether there is a satisfying assignment of values to variables

In CSPs, we use arc consistency and forward chaining to eliminate single elements of a domain, one at a time

In logic, we can **query** the KB to determine if every possible assignment of variables has particular properties

This allows us to “learn” or infer new information

Logic Language

Natural language?

Propositional logic

- Syntax: $P \vee (\neg Q \wedge R)$; $X_1 \Leftrightarrow (\text{Raining} \Rightarrow \text{Sunny})$
- Possible world: {P=true, Q=true, R=false, S=true} or 1101
- Semantics: $\alpha \wedge \beta$ is true in a world iff is α true and β is true (etc.)

First-order logic

- Syntax: $\forall x\ \exists y\ P(x,y) \wedge \neg Q(Joe,f(x)) \Rightarrow f(x)=f(y)$
- Possible world: Objects o_1, o_2, o_3; P holds for <o_1,o_2>; Q holds for <o_3>; $f(o_1)=o_1$; $Joe=o_3$; etc.
- Semantics: $\phi(\sigma)$ is true in a world if $\sigma=o_j$ and ϕ holds for o_j; etc.

Propositional Logic

Piazza Poll 1

If we know that $A \vee B$ and $\neg B \vee C$ are true,

what do we know about $A \vee C$?

i. $A \vee C$ is guaranteed to be true
ii. $A \vee C$ is guaranteed to be false
iii. We don't have enough information to say anything definitive about $A \vee C$

Piazza Poll 1

If we know that $A \vee B$ and $\neg B \vee C$ are true, what do we know about $A \vee C$?

A	B	C	$A \vee B$	$\neg B \vee C$	$A \vee C$
false	false	false	false	true	false
false	false	true	false	true	true
false	true	false	true	false	false
false	true	true	true	true	true
true	false	false	true	true	true
true	false	true	true	true	true
true	true	false	true	false	true
true	true	true	true	true	true

Piazza Poll 1

If we know that $A \vee B$ and $\neg B \vee C$ are true, what do we know about $A \vee C$?

A	B	C	$A \vee B$	$\neg B \vee C$	$A \vee C$
false	false	false	false	true	false
false	false	true	false	true	true
false	true	false	true	false	false
false	true	true	true	true	true
true	false	false	true	true	true
true	false	true	true	true	true
true	true	false	true	false	true
true	true	true	true	true	true

Piazza Poll 1

If we know that $A \vee B$ and $\neg B \vee C$ are true,

what do we know about $A \vee C$?

i. $A \vee C$ is guaranteed to be true

ii. $A \vee C$ is guaranteed to be false

iii. We don't have enough information to say anything definitive about $A \vee C$

Piazza Poll 2

If we know that $A \vee B$ and $\neg B \vee C$ are true,

what do we know about A?

i. A is guaranteed to be true

ii. A is guaranteed to be false

iii. We don't have enough information to say anything definitive about A

Piazza Poll 2

If we know that $A \vee B$ and $\neg B \vee C$ are true, what do we know about A?

A	B	C	$A \vee B$	$\neg B \vee C$	$A \vee C$
false	false	false	false	true	false
false	false	true	false	true	true
false	true	false	true	false	false
false	true	true	true	true	true
true	false	false	true	true	true
true	false	true	true	true	true
true	true	false	true	false	true
true	true	true	true	true	true

Piazza Poll 2

If we know that $A \vee B$ and $\neg B \vee C$ are true, what do we know about A?

i. A is guaranteed to be true
ii. A is guaranteed to be false
iii. We don't have enough information to say anything definitive about A

Propositional Logic

Symbol:

- Variable that can be true or false
- We'll try to use capital letters, e.g. A, B, $P_{1,2}$
- Often include True and False

Operators:

- $\neg$ A: not A
- A $\wedge$ B: A and B (conjunction)
- A $\vee$ B: A or B (disjunction) Note: this is not an "exclusive or"
- A $\Rightarrow$ B: A implies B (implication). If A then B
- A $\Leftrightarrow$ B: A if and only if B (biconditional)

Sentences

Propositional Logic Syntax

Given: a set of proposition symbols $\{X_1, X_2, \ldots, X_n\}$

- (we often add True and False for convenience)

X_i is a sentence

If α is a sentence then $\neg\alpha$ is a sentence

If α and β are sentences then $\alpha \wedge \beta$ is a sentence

If α and β are sentences then $\alpha \vee \beta$ is a sentence

If α and β are sentences then $\alpha \Rightarrow \beta$ is a sentence

If α and β are sentences then $\alpha \Leftrightarrow \beta$ is a sentence

And p.s. there are no other sentences!

Notes on Operators

$\boldsymbol{\alpha} \lor \boldsymbol{\beta}$ is inclusive or, not exclusive

Truth Tables

$\alpha \vee \beta$ is inclusive or, not exclusive

α	β	$\alpha \wedge \beta$
F	F	F
F	T	F
T	F	F
T	T	T

α	β	$\alpha \vee \beta$
F	F	F
F	T	T
T	F	T
T	T	T

Notes on Operators

$\alpha \lor \beta$ is inclusive or, not exclusive

$\alpha \Rightarrow \beta$ is equivalent to $\neg\alpha \lor \beta$

- Says who?

Truth Tables

$\boldsymbol{\alpha} \Rightarrow \boldsymbol{\beta}$ is equivalent to $\neg\boldsymbol{\alpha} \vee \boldsymbol{\beta}$

$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\alpha} \Rightarrow \boldsymbol{\beta}$	$\neg\boldsymbol{\alpha}$	$\neg\boldsymbol{\alpha} \vee \boldsymbol{\beta}$
F	F	T	T	T
F	T	T	T	T
T	F	F	F	F
T	T	T	F	T

Notes on Operators

$\boldsymbol{\alpha} \vee \boldsymbol{\beta}$ is inclusive or, not exclusive

$\boldsymbol{\alpha} \Rightarrow \boldsymbol{\beta}$ is equivalent to $\neg\boldsymbol{\alpha} \vee \boldsymbol{\beta}$

- Says who?

$\boldsymbol{\alpha} \Leftrightarrow \boldsymbol{\beta}$ is equivalent to $(\boldsymbol{\alpha} \Rightarrow \boldsymbol{\beta}) \wedge (\boldsymbol{\beta} \Rightarrow \boldsymbol{\alpha})$

- Prove it!

Truth Tables

$\alpha \Leftrightarrow \beta$ is equivalent to $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$

α	β	$\alpha \Leftrightarrow \beta$	$\alpha \Rightarrow \beta$	$\beta \Rightarrow \alpha$	$(\alpha\Rightarrow\beta) \land (\beta\Rightarrow\alpha)$
F	F	T	T	T	T
F	T	F	T	F	F
T	F	F	F	T	F
T	T	T	T	T	T

Equivalence: it's true in all models. Expressed as a logical sentence:

$$(\alpha \Leftrightarrow \beta) \Leftrightarrow [(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)]$$

Literals

A *literal* is an atomic sentence:

- True
- False
- Symbol
- $\neg$ Symbol

Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

KB: $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

KB: $[(P \land \neg Q) \lor (Q \land \neg P)] \Rightarrow R$

KB: R, $[(P \land \neg Q) \lor (Q \land \neg P)] \Rightarrow R$

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Entailment

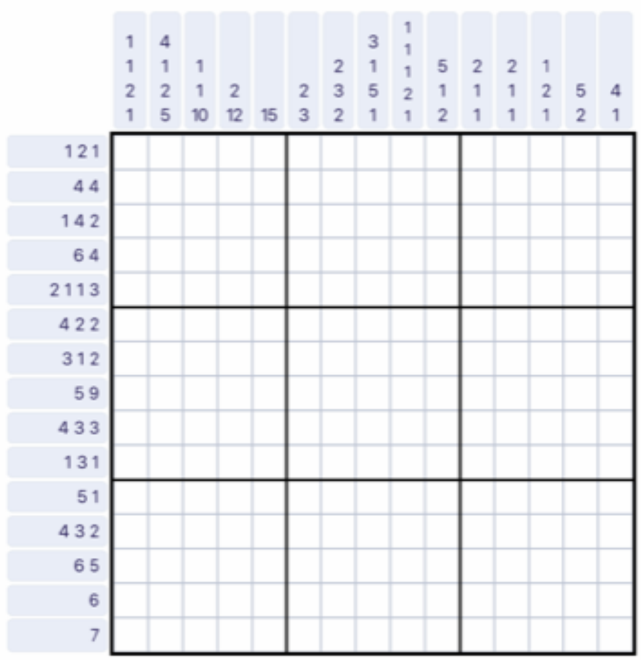
Entailment: $\alpha \models \beta$ ("α entails β" or "β follows from α") iff in every world where α is true, β is also true

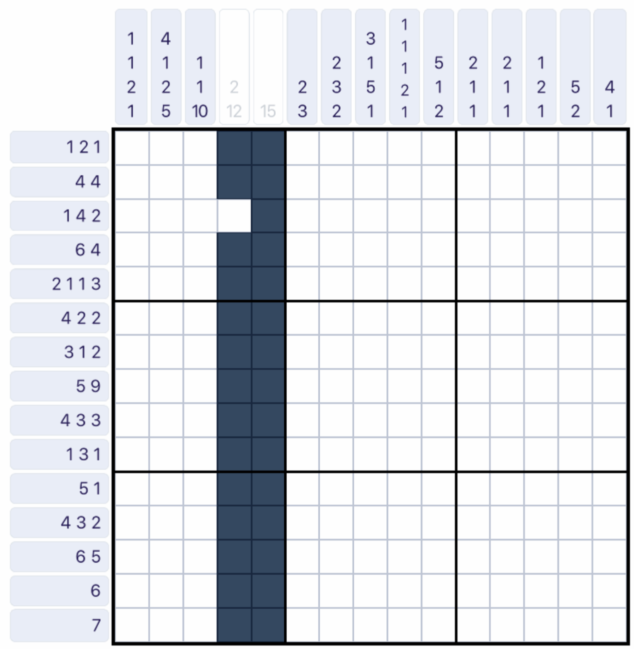
- I.e., the α-worlds are a subset of the β-worlds [*models*(α) $\subseteq$ *models*(β)]

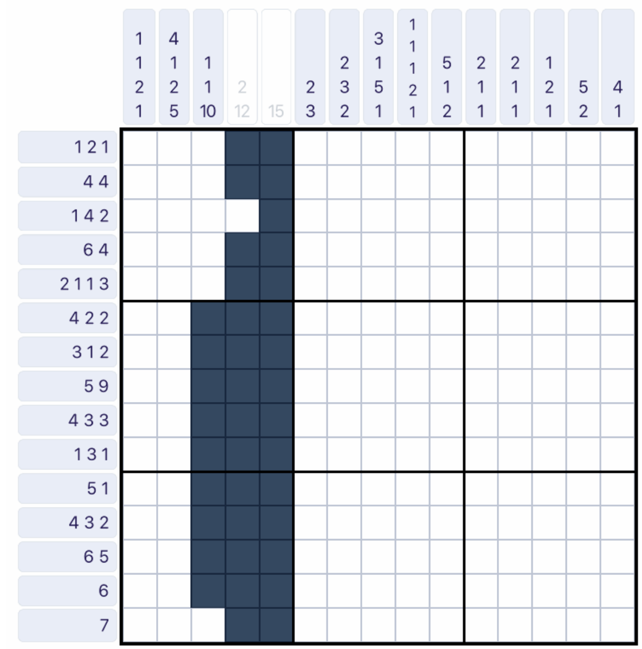
Usually we want to know if *KB* $\models$ *query*

- *models*(*KB*) $\subseteq$ *models*(*query*)
- In other words
  - *KB* removes all impossible models (any model where *KB* is false)
  - If β is true in all of these remaining models, we conclude that β must be true

Nonogram Example







Given the KB of constraints, we can query particular squares to determine if they are true or false in all models, or if they are unknown.

Wumpus World

World has 5 locations

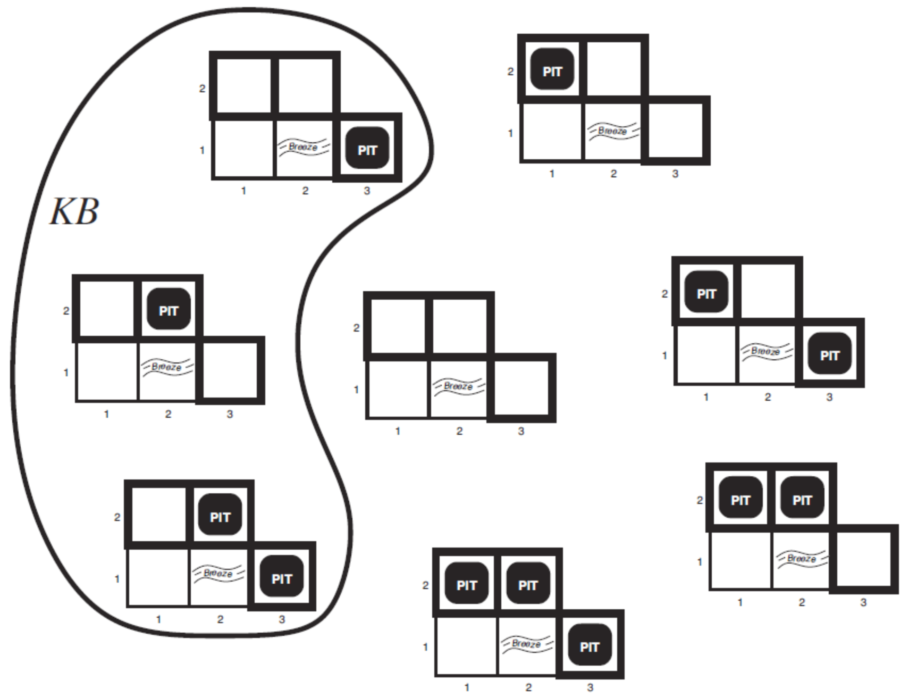
[1,1], [2,1], [3,1], [1,2], [2,2]

Knowledge base

Nothing in [1,1]
Breeze in [2,1]

Possible Models for Pits

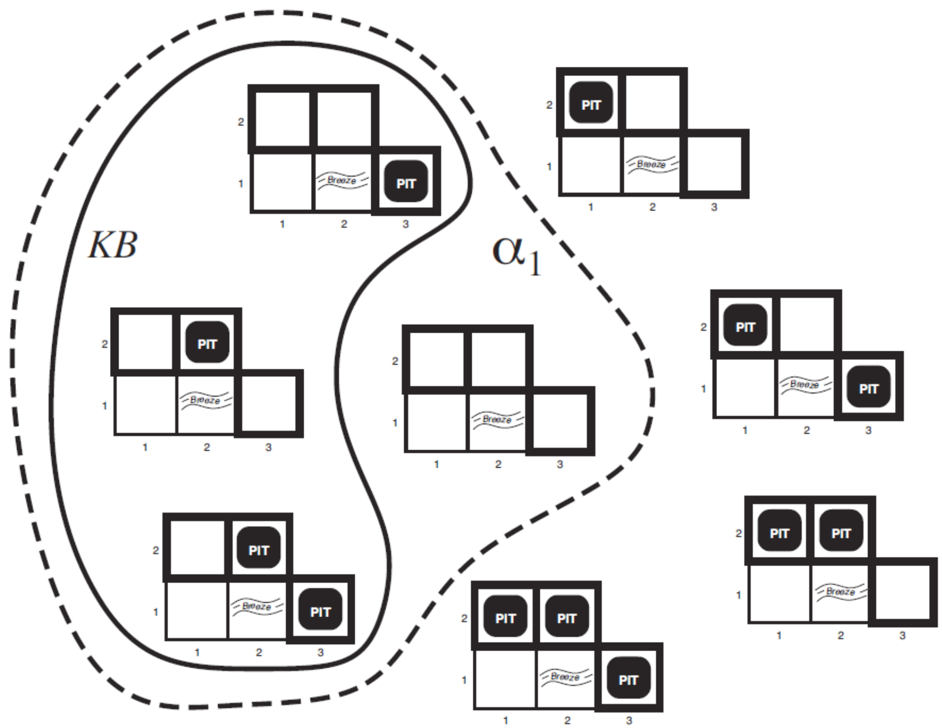
$P_{1,1}$=F, $P_{2,1}$=F, $P_{1,2}$,$P_{2,2}$,$P_{3,1}$



Wumpus World

Possible Models

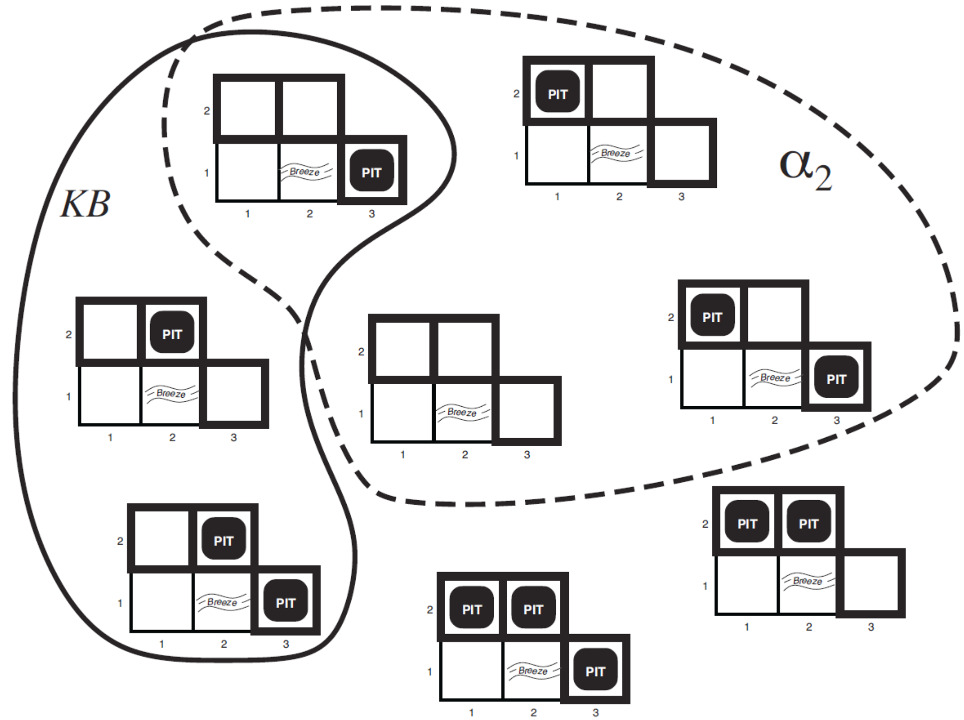
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- KB entails α_1?
  - Yes! No pit in [1,2]
  - We can add this fact to our KB



Wumpus World

Possible Models

- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- KB entails α_2?
  - No! We don't know whether there is a pit in [2,2]



Entailment

Entailment: $\alpha \models \beta$ ("α entails β" or "β follows from α") iff in every world where α is true, β is also true

- I.e., the α-worlds are a subset of the β-worlds [*models*(α) $\subseteq$ *models*(β)]

Usually we want to know if *KB* |= *query*

- *models*(*KB*) $\subseteq$ *models*(*query*)
- In other words
  - *KB* removes all impossible models (any model where *KB* is false)
  - If β is true in all of these remaining models, we conclude that β must be true

Entailment and implication are very much related

- However, entailment relates two sentences, while an implication is itself a sentence (usually derived via inference to show entailment)

Propositional Logic Models

All Possible Models

Model Symbols

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1

Piazza Poll 3

Does the KB entail query C?

Entailment: $\alpha \models \beta$

"α entails β" iff in every world where α is true, β is also true

All Possible Models

Model Symbols	A	0	0	0	0	1	1	1	1
	B	0	0	1	1	0	0	1	1
	C	0	1	0	1	0	1	0	1
Knowledge Base	A	0	0	0	0	1	1	1	1
	$B \Rightarrow C$	1	1	0	1	1	1	0	1
	$A \Rightarrow B \vee C$	1	1	1	1	0	1	1	1
Query	C	0	1	0	1	0	1	0	1

Entailment

How do we implement a logical agent that proves entailment?

- Logic language
  - Propositional logic
  - First order logic

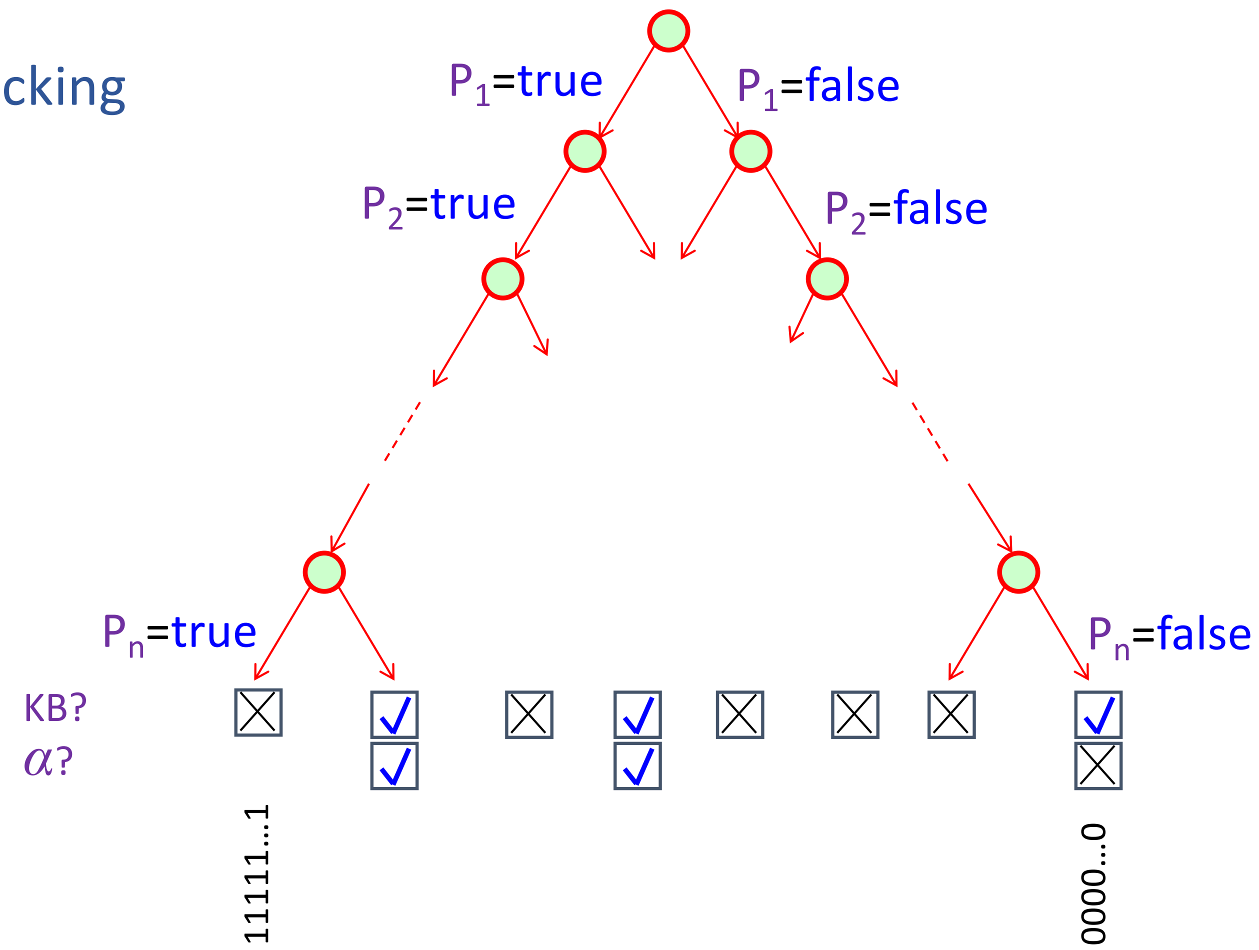
- Inference algorithms
  - Theorem proving
  - Model checking

Simple Model Checking

Same recursion as backtracking

$O(2^n)$ time, linear space

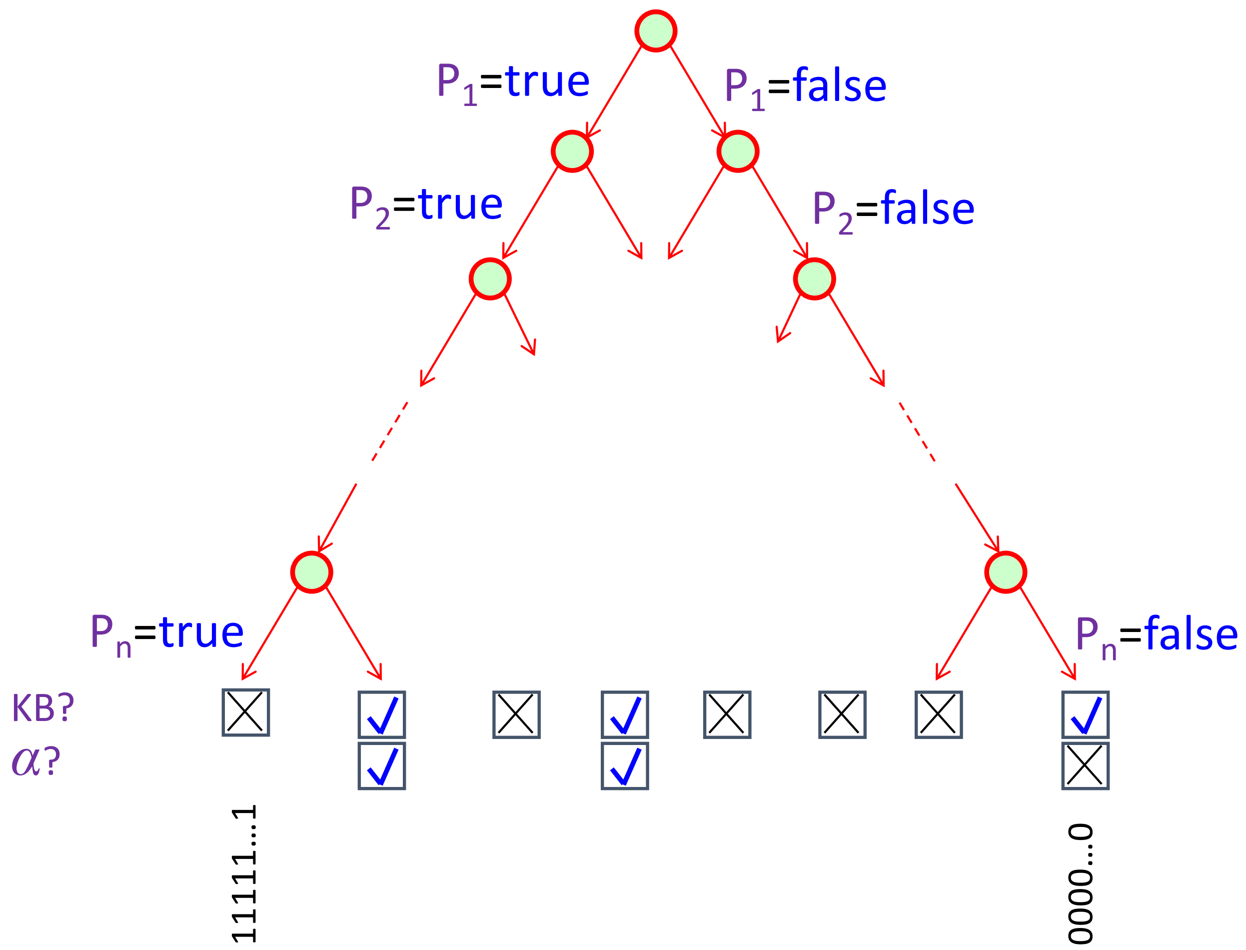
We can do much better!



Piazza Poll 4

Which would you choose?

- DFS
- BFS



Simple Model Checking

```
function TT-ENTAILS?(KB, α) returns true or false
   return TT-CHECK-ALL(KB, α, symbols(KB) U symbols(α), {})

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if empty?(symbols) then
         if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
         else return true
   else
         P ← first(symbols)
         rest ← rest(symbols)
         return  and (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
                      TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

Propositional Logic

Check if sentence is true in given model

In other words, does the model *satisfy* the sentence?

```
function PL-TRUE?(α,model) returns true or false
   if α is a symbol then return Lookup(α, model)
   if Op(α) = ¬ then return not(PL-TRUE?(Arg1(α),model))
   if Op(α) = ∧ then return  and(PL-TRUE?(Arg1(α),model),
                                 PL-TRUE?(Arg2(α),model))
   etc.
```

(Sometimes called “recursion over syntax”)

Inference: Proofs

A proof is a *demonstration* of entailment between α and β

Method 1: *model-checking*

- For every possible world, if α is true make sure that is β true too
- OK for propositional logic (finitely many worlds); not easy for first-order logic

Method 2: *theorem-proving*

- Search for a sequence of proof steps (applications of *inference rules*) leading from α to β
- E.g., from $P \wedge (P \Rightarrow Q)$, infer Q by *Modus Ponens*

Properties

- *Sound* algorithm: everything it claims to prove is in fact entailed
- *Complete* algorithm: every sentence that is entailed can be proved

Simple Theorem Proving: Forward Chaining

Forward chaining applies Modus Ponens to generate new facts:

- Given $X_1 \wedge X_2 \wedge \ldots X_n \Rightarrow Y$ and X_1, X_2, …, X_n
- Infer Y

Forward chaining keeps applying this rule, adding new facts, until nothing more can be added

Requires KB to contain only *definite clauses*:

- (Conjunction of symbols) $\Rightarrow$ symbol; or
- A single symbol (note that X is equivalent to $True \Rightarrow X$)

Forward Chaining Algorithm

function PL-FC-ENTAILS?(KB, q) returns true or false

count ← a table, where count[c] is the number of symbols in c's premise

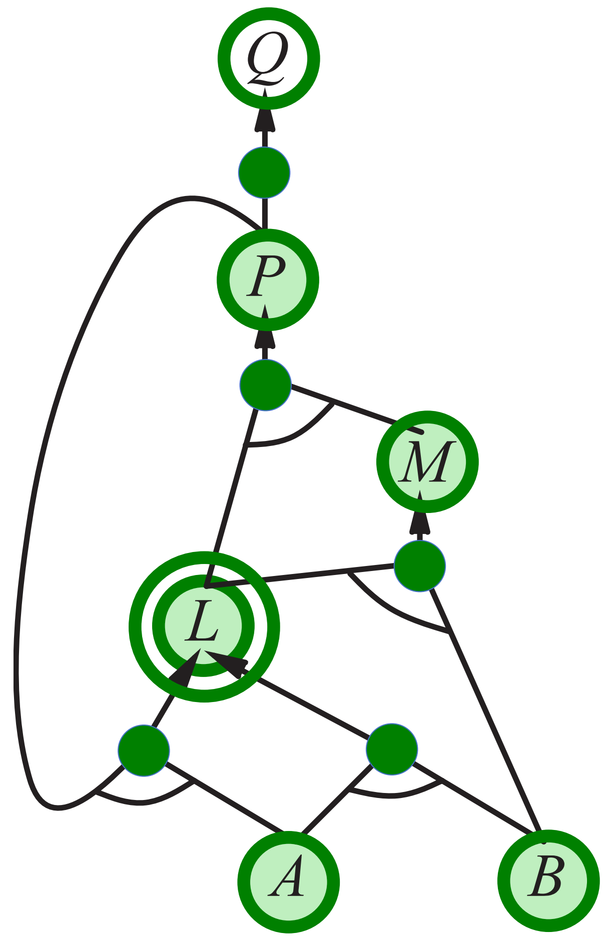
inferred ← a table, where inferred[s] is initially false for all s

agenda ← a queue of symbols, initially symbols known to be true in KB

CLAUSES	*COUNT*	*INFERRED*	*AGENDA*
$P \Rightarrow Q$	1	A false	
$L \wedge M \Rightarrow P$	2	B false	
$B \wedge L \Rightarrow M$	2	L false	
$A \wedge P \Rightarrow L$	2	M false	
$A \wedge B \Rightarrow L$	2	P false	
A	0	Q false	
B	0		

Forward Chaining Example: Proving Q

CLAUSES	COUNT	INFERRED
$P \Rightarrow Q$	~~1~~ 0	A ~~false~~ true
$L \wedge M \Rightarrow P$	~~2~~ ~~1~~ 0	B ~~false~~ true
$B \wedge L \Rightarrow M$	~~2~~ ~~1~~ 0	L ~~false~~ true
$A \wedge P \Rightarrow L$	~~2~~ ~~1~~ 0	M ~~false~~ true
$A \wedge B \Rightarrow L$	~~2~~ ~~1~~ 0	P ~~false~~ true
A	0	Q ~~false~~ true
B	0	



AGENDA

~~A~~ ~~B~~ ~~L~~ ~~M~~ ~~P~~ ~~L~~ ~~Q~~

Forward Chaining Algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
   count ← a table, where count[c] is the number of symbols in c's premise
   inferred ← a table, where inferred[s] is initially false for all s
   agenda ← a queue of symbols, initially symbols known to be true in KB
   while agenda is not empty do
         p ← Pop(agenda)
         if p = q then return true
         if inferred[p] = false then
               inferred[p]←true
               for each clause c in KB where p is in c.premise do
                     decrement count[c]
                     if count[c] = 0 then add c.conclusion to agenda
   return false
```

Properties of forward chaining

Theorem: FC is sound and complete for definite-clause KBs

Soundness: follows from soundness of Modus Ponens (easy to check)

Completeness proof:

1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final *inferred* table as a model ***m***, assigning true/false to symbols
3. Every clause in the original KB is true in ***m***

   Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in ***m***
   Then $a_1 \wedge \ldots \wedge a_k$ is true in ***m*** and b is false in ***m***
   Therefore the algorithm has not reached a fixed point!

4. Hence ***m*** is a model of KB
5. If KB |= q, q is true in every model of KB, including ***m***

A	false xxxx true
B	false xxxx true
L	false xxxx true
M	false xxxx true
P	false xxxx true
Q	false xxxx true

Inference Rules

Notation Alert!

Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

Unit Resolution

$$\frac{a \vee b, \quad \neg b \vee c}{a \vee c}$$

General Resolution

$$\frac{a_1 \vee \cdots \vee a_m \vee b, \quad \neg b \vee c_1 \vee \cdots \vee c_n}{a_1 \vee \cdots \vee a_m \vee c_1 \vee \cdots \vee c_n}$$

Resolution

Algorithm Overview

function PL-RESOLUTION?(KB, α) returns true or false

We want to prove that KB entails α

In other words, we want to prove that we cannot satisfy (KB and **not** α)

1. Start with a set of CNF clauses, including the KB as well as $\neg\alpha$
2. Keep resolving pairs of clauses until
   A. You resolve the empty clause

      Contradiction found!

      KB $\land \neg\alpha$ cannot be satisfied

      Return true, KB entails α

   B. No new clauses added

      Return false, KB does not entail α

Resolution

General Resolution

$$\frac{a_1 \vee \cdots \vee a_m \vee b, \quad \neg b \vee c_1 \vee \cdots \vee c_n}{a_1 \vee \cdots \vee a_m \vee c_1 \vee \cdots \vee c_n}$$

Example trying to prove $\neg P_{1,2}$

Knowledge Base

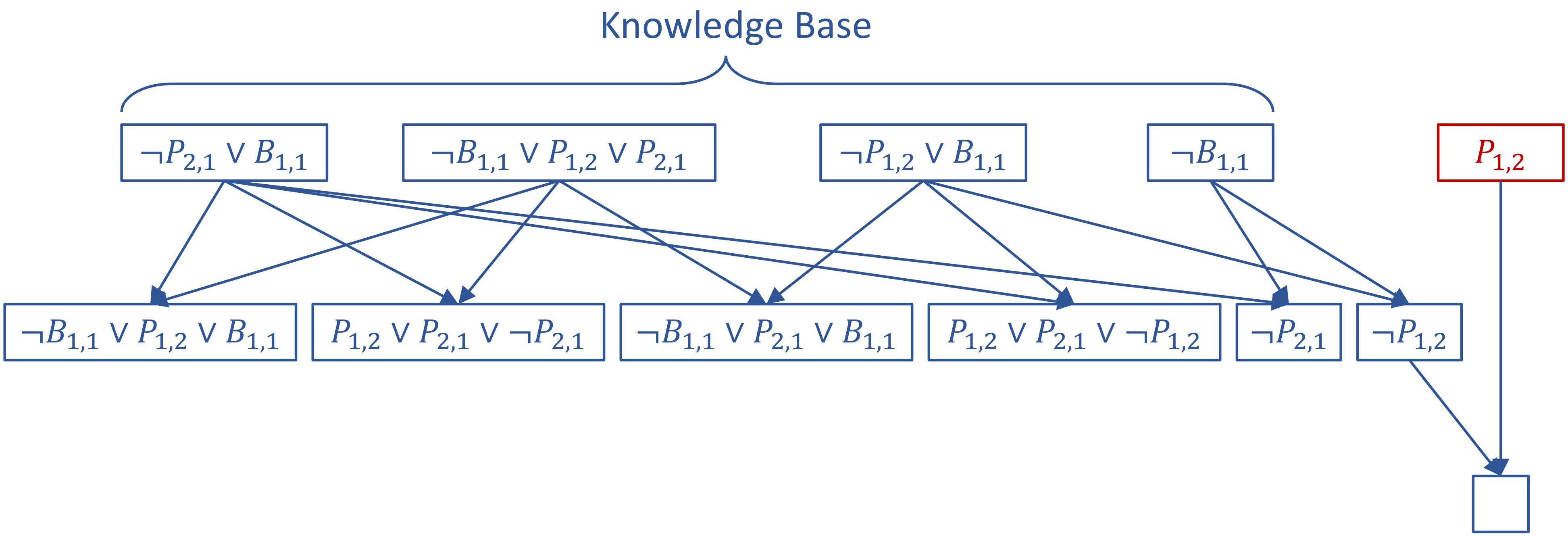
$\neg P_{2,1} \vee B_{1,1}$	$\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$	$\neg P_{1,2} \vee B_{1,1}$	$\neg B_{1,1}$	$\neg\neg P_{1,2}$

Resolution

Example trying to prove $\neg P_{1,2}$

General Resolution

$$\frac{a_1 \vee \cdots \vee a_m \vee b, \quad \neg b \vee c_1 \vee \cdots \vee c_n}{a_1 \vee \cdots \vee a_m \vee c_1 \vee \cdots \vee c_n}$$



Resolution

```
function PL-RESOLUTION?(KB, α) returns true or false
  clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
  new ← { }
  loop do
    for each pair of clauses Ci, Cj in clauses do
      resolvents ← PL-RESOLVE(Ci, Cj)
      if resolvents contains the empty clause then
        return true
      new ← new ∪ resolvants
    if new ⊆ clauses then
      return false
    clauses ← clauses ∪ new
```

Properties

Forward Chaining is:

- Sound and complete for definite-clause KBs
- Complexity: linear time

Resolution is:

- Sound and complete for any PL KBs!
- Complexity: exponential time ☹