

Warm-up:

What is the relationship between number of constraints and number of possible solutions?

In other words, as the number of the constraints increases, does the number of possible solutions:

- A) Increase
- B) Decrease
- C) Stay the same

Announcements

Midterm 1 Exam

- Tue 10/1, in class

Assignments:

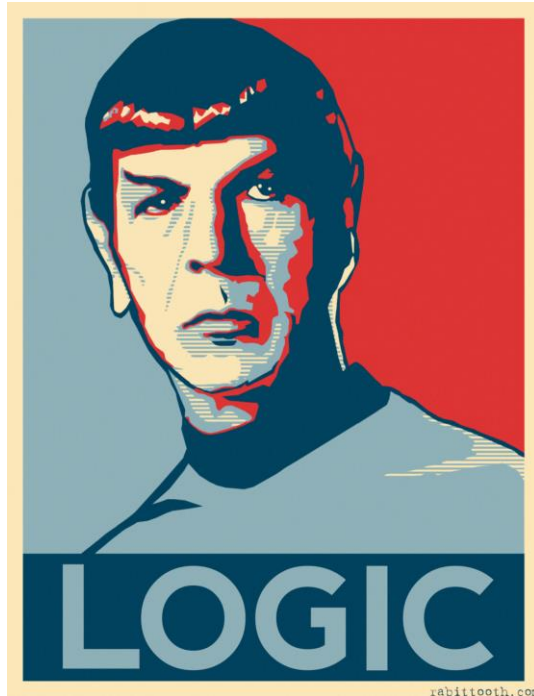
- HW3
 - Due tonight, 10 pm
- HW4
 - Due Tue 9/24, 10 pm
- P2: Logic and Planning
 - Out Thu 9/19
 - Due ~~Thu 10/3~~, 10 pm Sat 10/5, 10 pm

Recitation and feedback survey on Piazza

- Due tomorrow, 10 pm

AI: Representation and Problem Solving

Propositional Logic

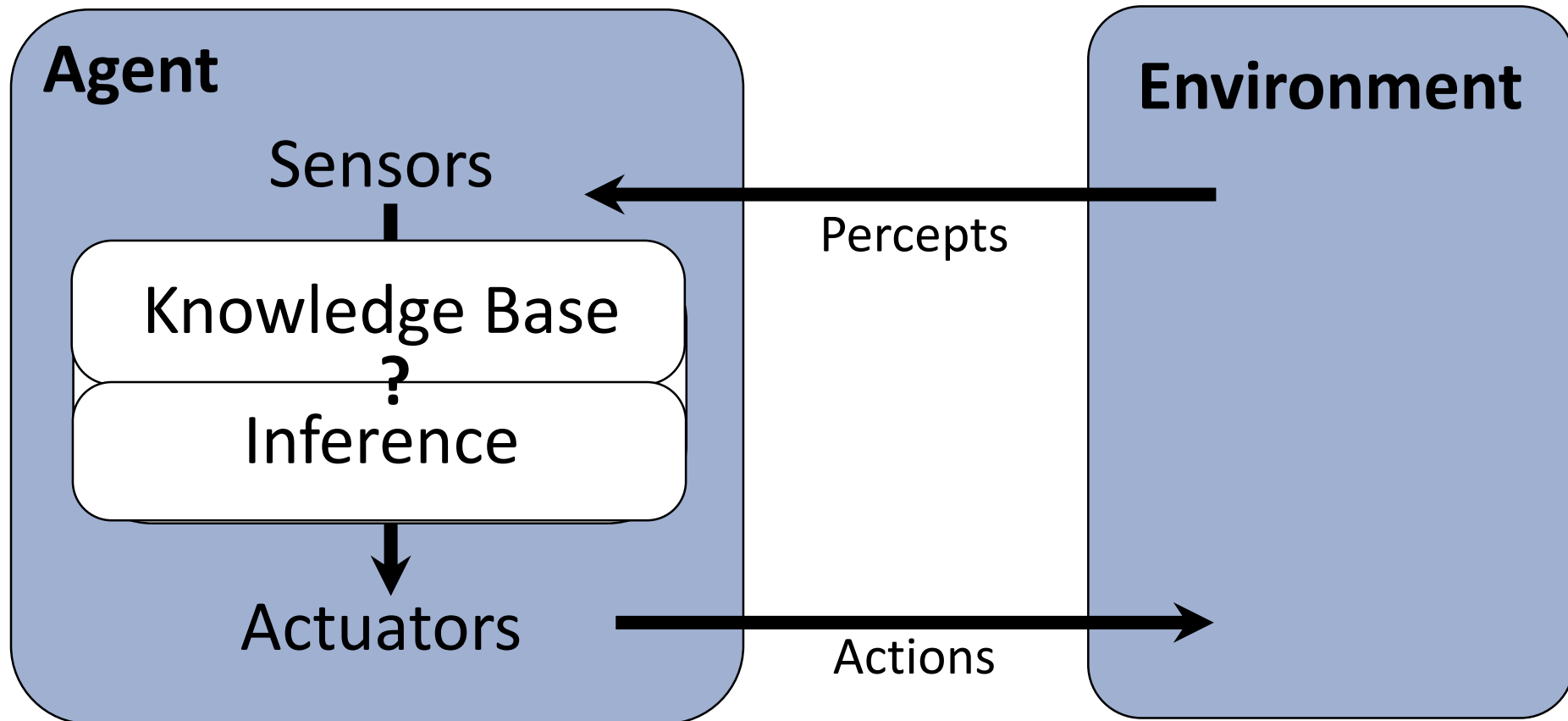


Instructors: Pat Virtue & Fei Fang

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

Logical Agents

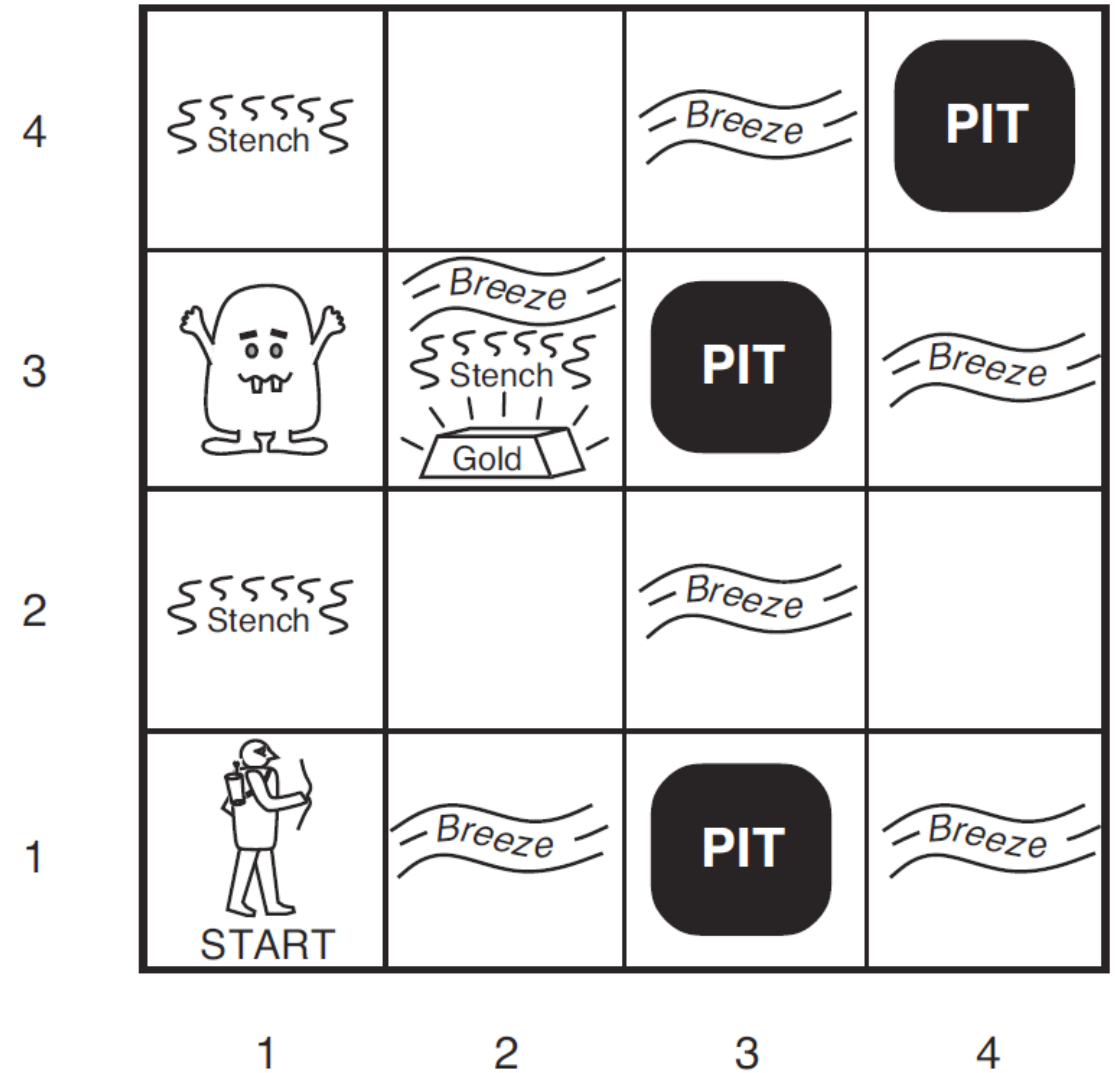
Logical agents and environments



Wumpus World

Logical Reasoning as a CSP

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



A Knowledge-based Agent

function **KB-AGENT**(percept) returns an action

persistent: **KB**, a knowledge base

t, an integer, initially 0

TELL(**KB**, **PROCESS-PERCEPT**(percept, **t**))

action $\leftarrow$ **ASK**(**KB**, **PROCESS-QUERY**(**t**))

TELL(**KB**, **PROCESS-RESULT**(action, **t**))

t $\leftarrow$ **t**+1

return action

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 ASK the agent?

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

Logical Agents

Sherlock Agent

- Really good knowledge base
 - Evidence
 - Understanding of how the world works (physics, chemistry, sociology)
- Really good inference
 - Skills of deduction
 - “It’s elementary my dear Watson”



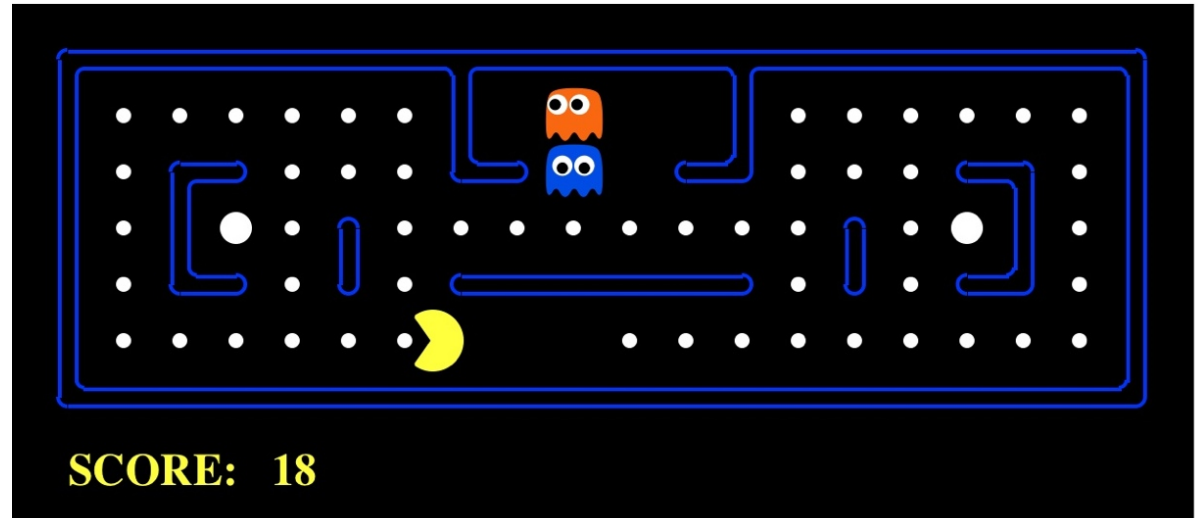
Dr. Strange?
Alan Turing?
Kahn?

Worlds

What are we trying to figure out?



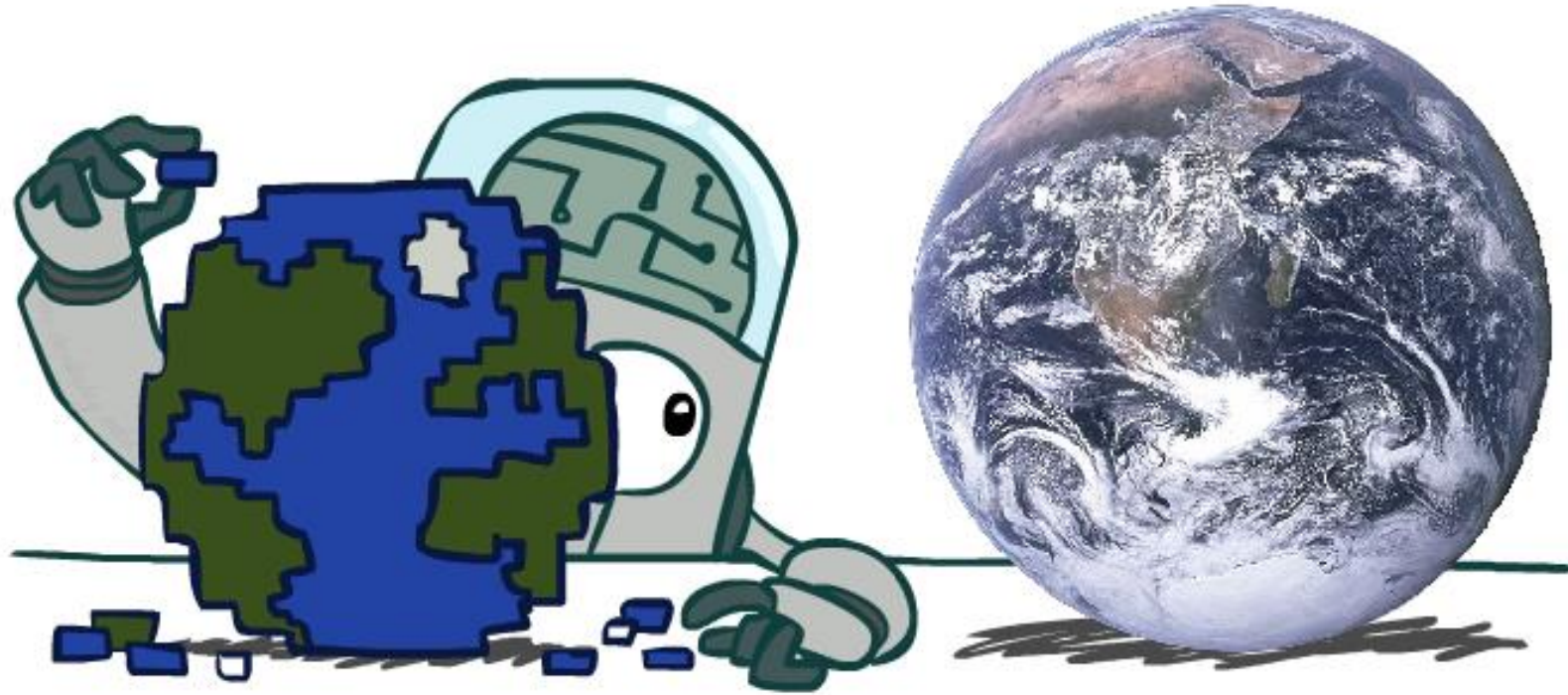
- Who, what, when, where, why
- Time: past, present, future



- Actions, strategy
- Partially observable? Ghosts, Walls

Which world are we living in?

Models

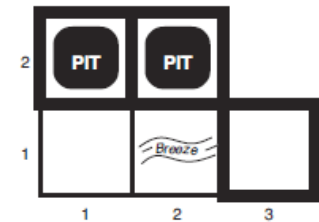
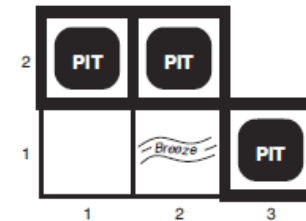
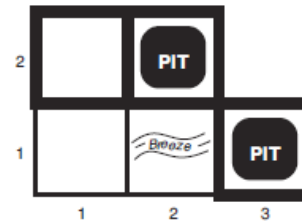
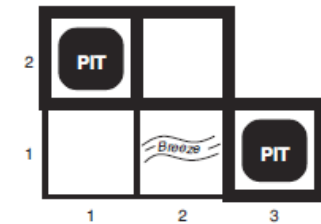
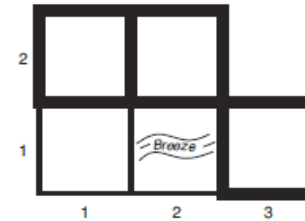
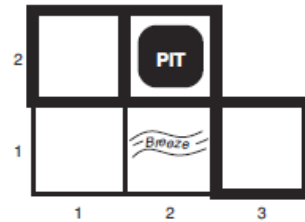
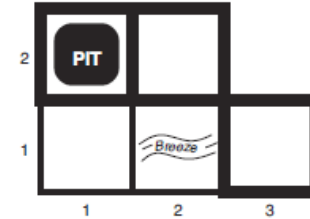
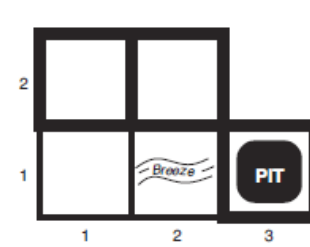


How do we represent possible worlds with models and knowledge bases?
How do we then do inference with these representations?

Wumpus World

Possible Models

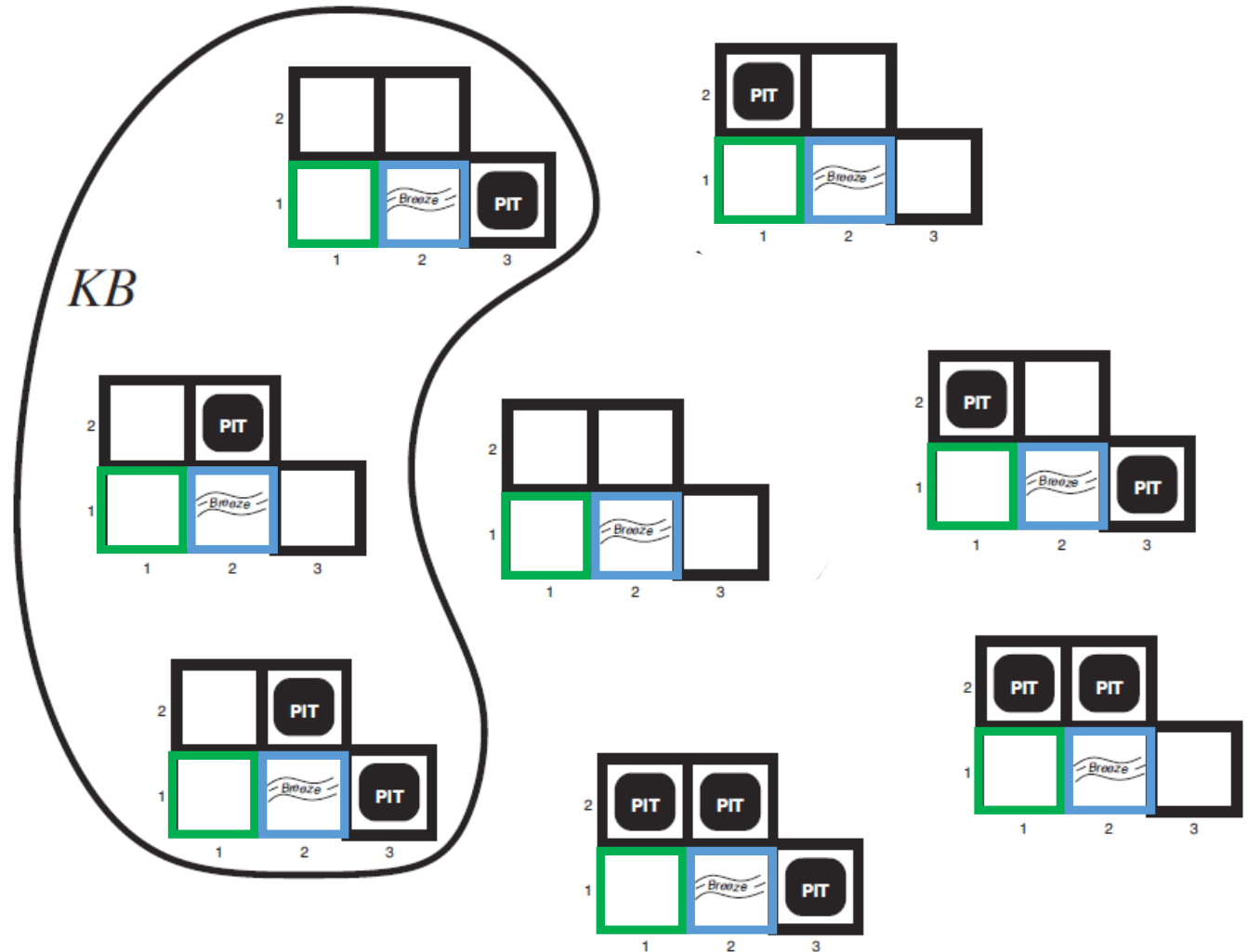
- $P_{1,2} P_{2,2} P_{3,1}$



Wumpus World

Possible Models

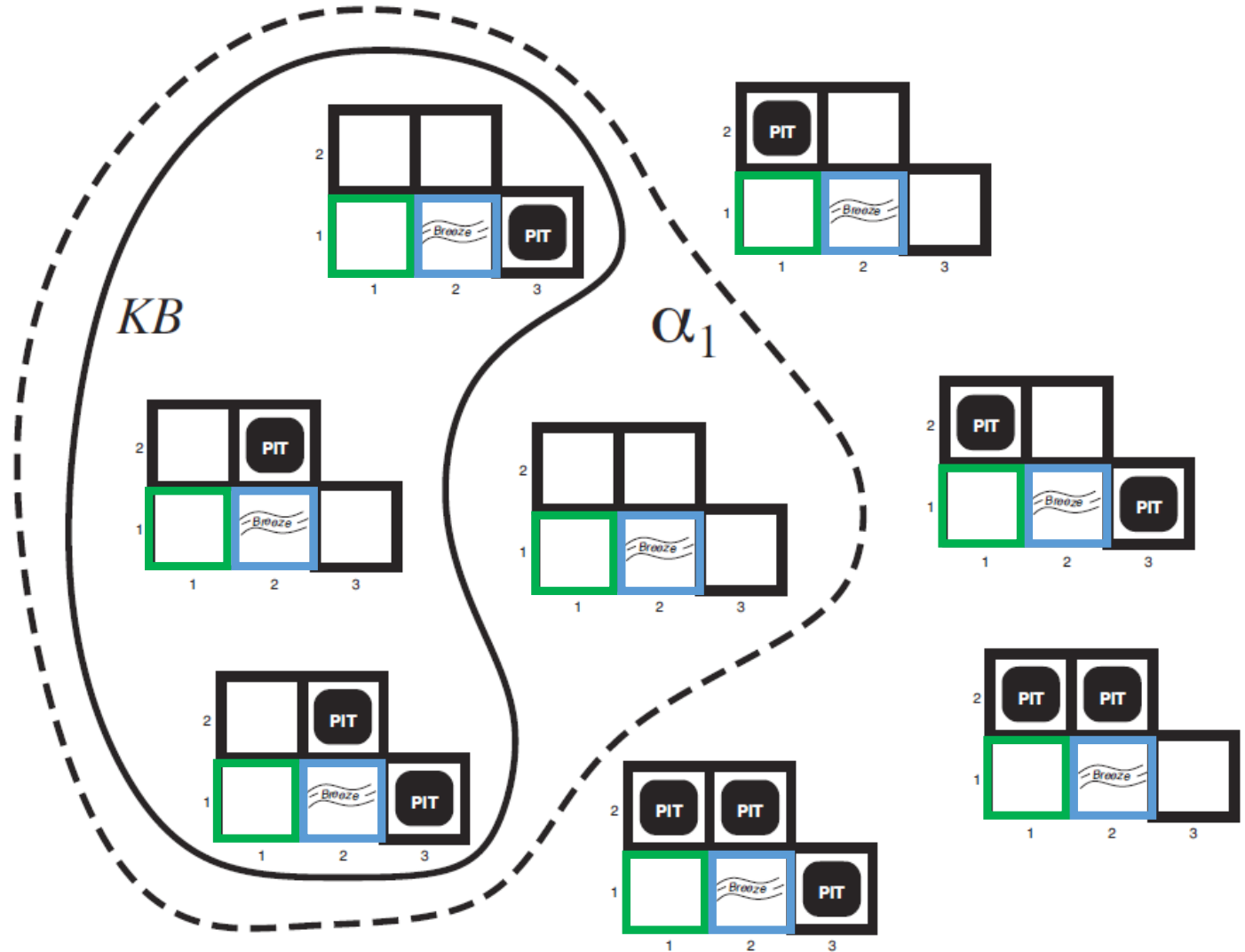
- $P_{1,2} P_{2,2} P_{3,1}$
- Knowledge base
 - Breeze $\Rightarrow$ Adjacent P
 - Nothing in $[1,1]$
 - Breeze in $[2,1]$



Wumpus World

Possible Models

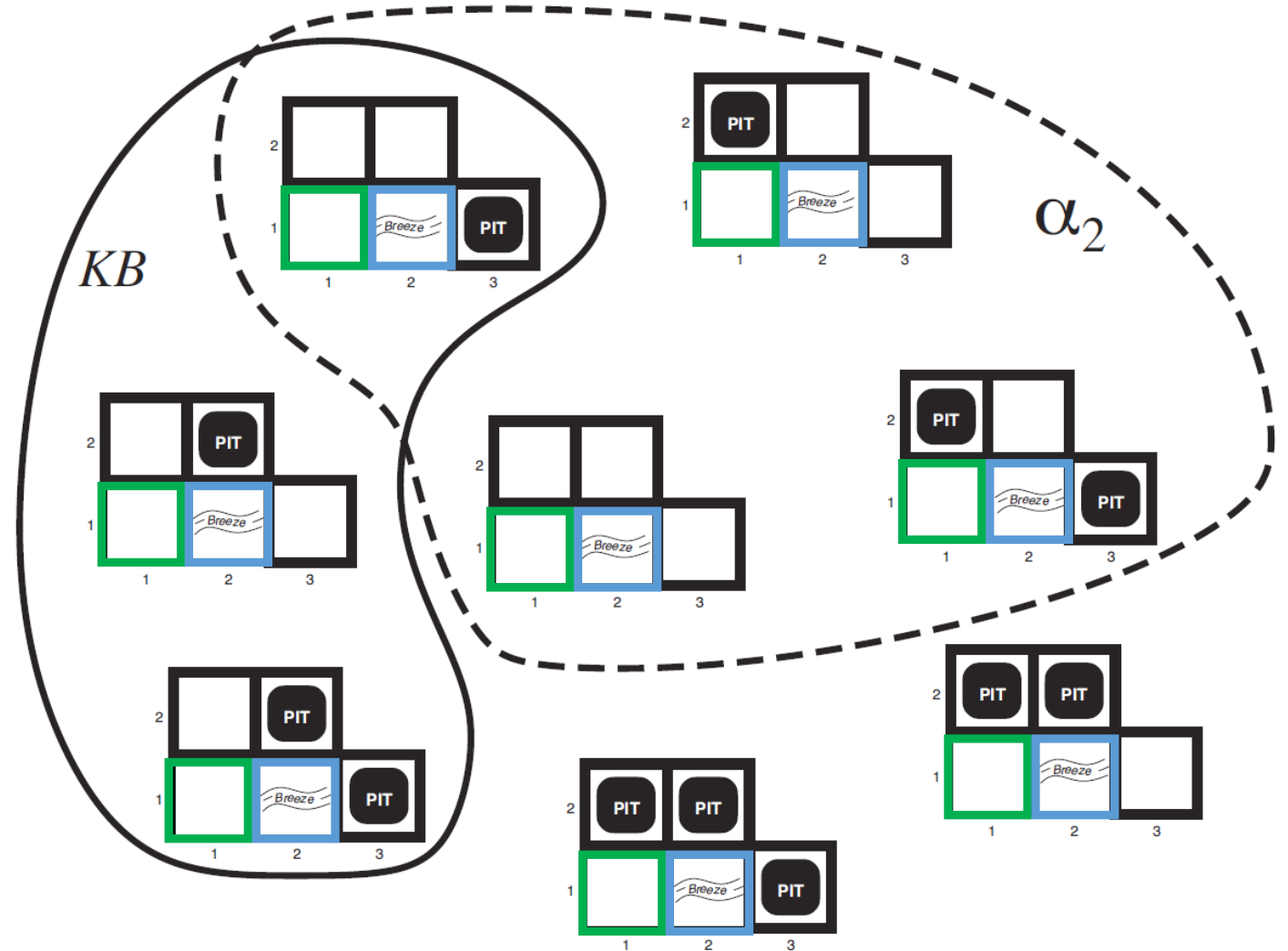
- $P_{1,2} P_{2,2} P_{3,1}$
- Knowledge base
 - Breeze $\Rightarrow$ Adjacent Pit
 - Nothing in [1,1]
 - Breeze in [2,1]
- Query α_1 :
 - No pit in [1,2]



Wumpus World

Possible Models

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



Logic Language

Natural language?

Propositional logic

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

First-order logic

- Syntax: $\forall x \exists y P(x,y) \wedge \neg Q(\text{Joe}, f(x)) \Rightarrow f(x)=f(y)$
- Possible world: Objects o_1, o_2, o_3 ; P holds for $\langle o_1, o_2 \rangle$; Q holds for $\langle o_3 \rangle$; $f(o_1)=o_1$; $\text{Joe}=o_3$; 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 $\{X_1, X_2, \dots, X_n\}$

- (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

$\alpha \vee \beta$ 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

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

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

- Says who?

Truth Tables

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

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

Notes on Operators

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

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

- Says who?

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

- Prove it!

Truth Tables

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

α	β	$\alpha \Leftrightarrow \beta$	$\alpha \Rightarrow \beta$	$\beta \Rightarrow \alpha$	$(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
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:

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

Propositional Logical Vocab

Literal

Vocab Alert!

- Atomic sentence: True , False , Symbol , $\neg \text{Symbol}$

Clause

- Disjunction of literals: $A \vee B \vee \neg C$

Definite clause

- Disjunction of literals, *exactly one* is positive
- $\neg A \vee B \vee \neg C$

Horn clause

- Disjunction of literals, *at most one* is positive
- All definite clauses are Horn clauses

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( $\alpha$ ) =  $\neg$  then return not(PL-TRUE?(Arg1( $\alpha$ ),model))
```

[illegible]

etc.

(Sometimes called “recursion over syntax”)

Warm-up:

What is the relationship between number of constraints and number of possible solutions?

In other words, as the number of the constraints increases, does the number of possible solutions:

- A) Increase
- B) Decrease
- C) Stay the same

Where is the knowledge in our CSPs?

Piazza Poll 3

What is the relationship between the size of the knowledge base and number of satisfiable models?

In other words, as the number of the knowledge base rules increases, does the number of satisfiable models:

- A) Increase
- B) Decrease
- C) Stay the same

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

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

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

Sherlock Entailment

“When you have eliminated the impossible, whatever remains, however improbable, must be the truth” – *Sherlock Holmes via Sir Arthur Conan Doyle*

(Not quite)

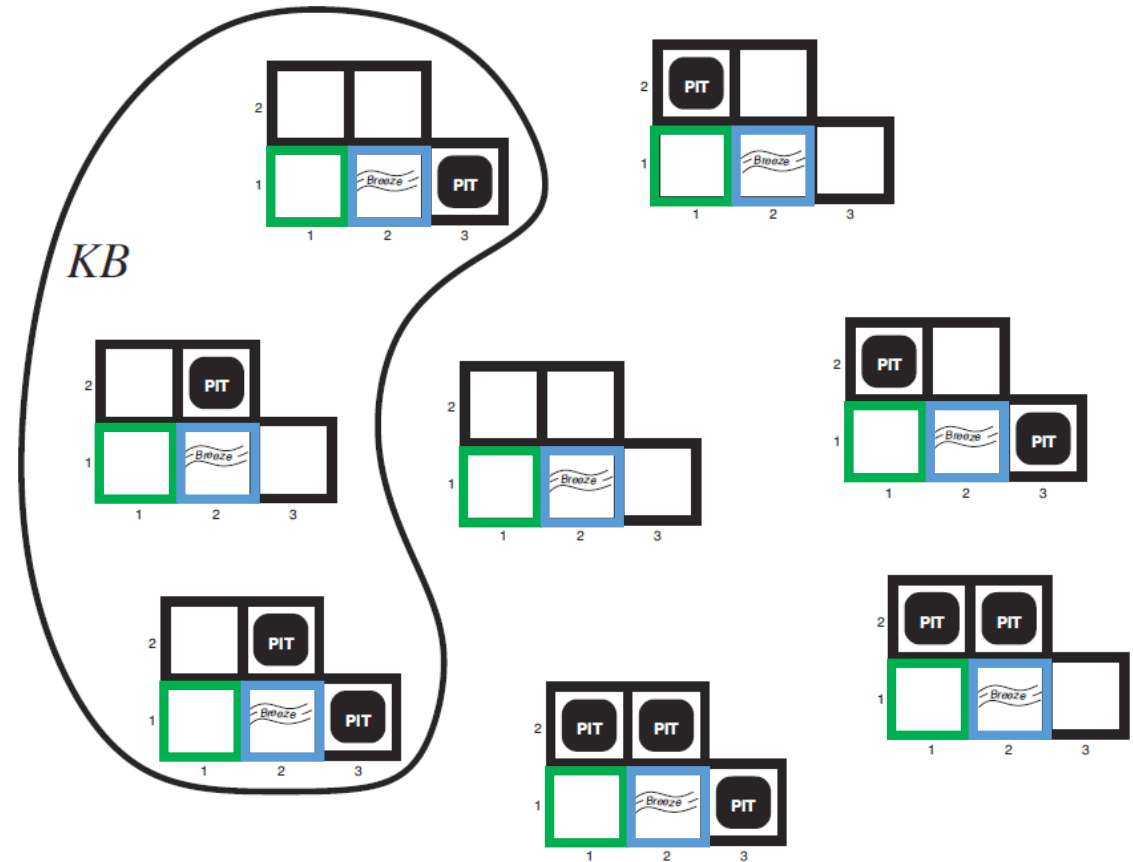
- Knowledge base and inference allow us to remove impossible models, helping us to see what is true in all of the remaining models



Wumpus World

Possible Models

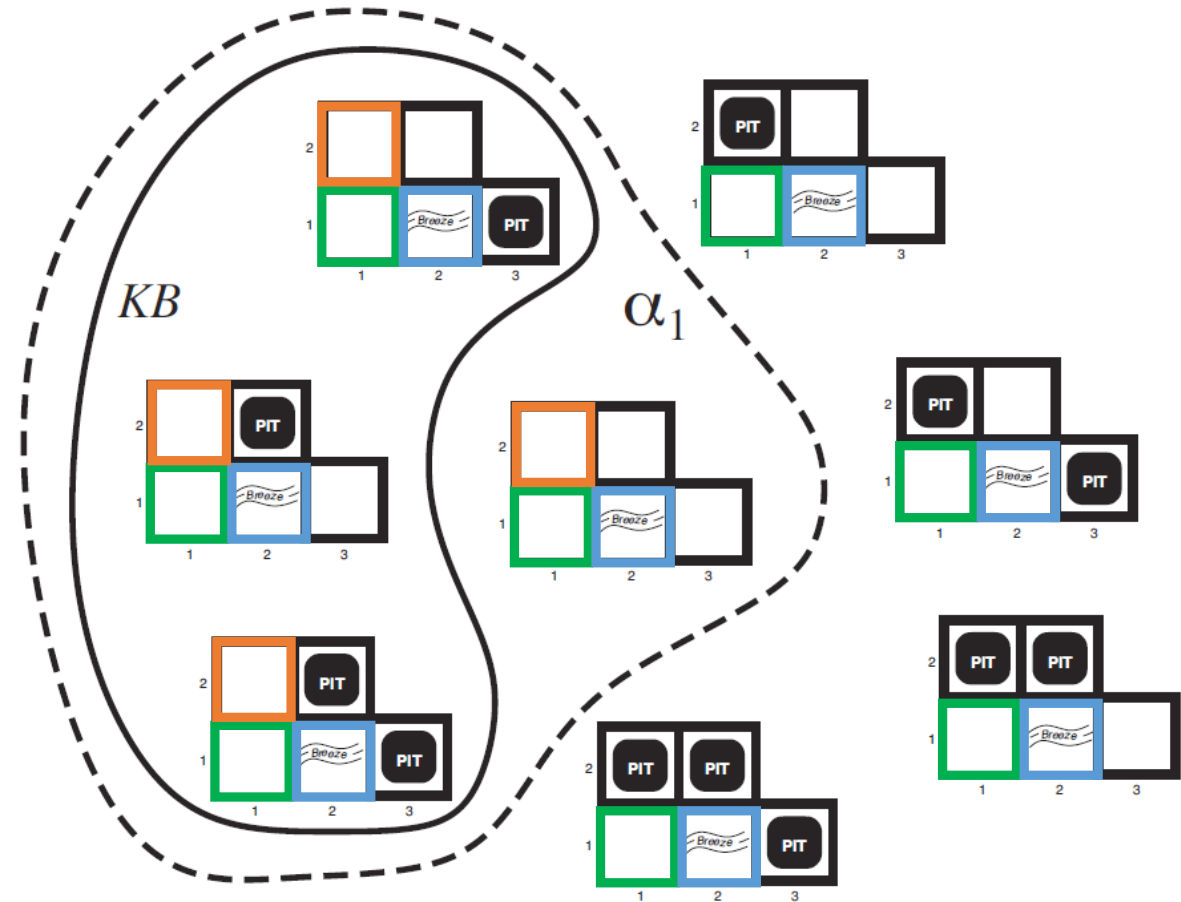
- $P_{1,2} P_{2,2} P_{3,1}$
- Knowledge base
 - Breeze $\Rightarrow$ Adjacent Pit
 - Nothing in $[1,1]$
 - Breeze in $[2,1]$



Wumpus World

Possible Models

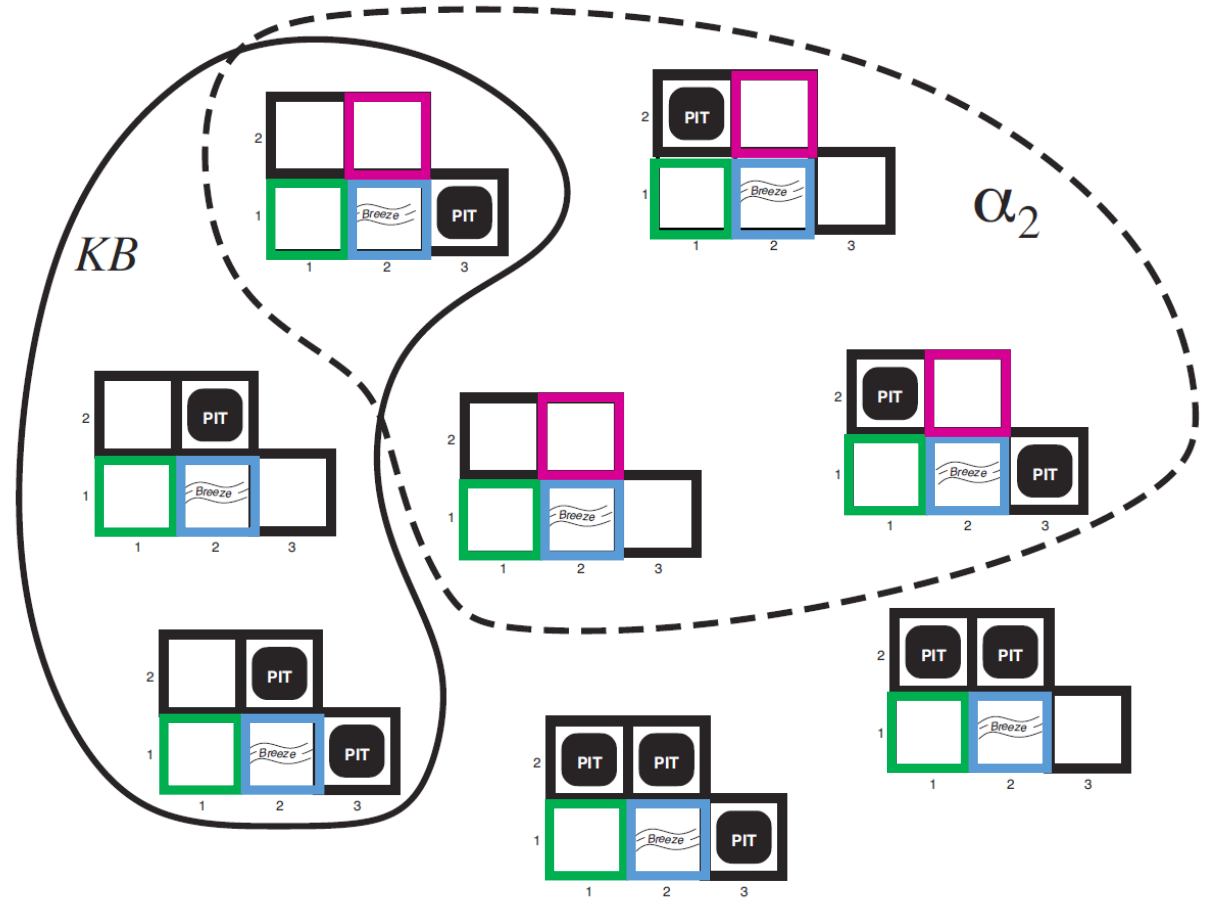
- $P_{1,2} P_{2,2} P_{3,1}$
- Knowledge base
 - Breeze $\Rightarrow$ Adjacent Pit
 - Nothing in [1,1]
 - Breeze in [2,1]
- Query α_1 :
 - No pit in [1,2]



Wumpus World

Possible Models

- $P_{1,2} P_{2,2} P_{3,1}$
- Knowledge base
 - Breeze $\Rightarrow$ Adjacent Pit
 - Nothing in $[1,1]$
 - Breeze in $[2,1]$
- Query α_2 :
 - No 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 [$models(\alpha) \subseteq models(\beta)$]

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

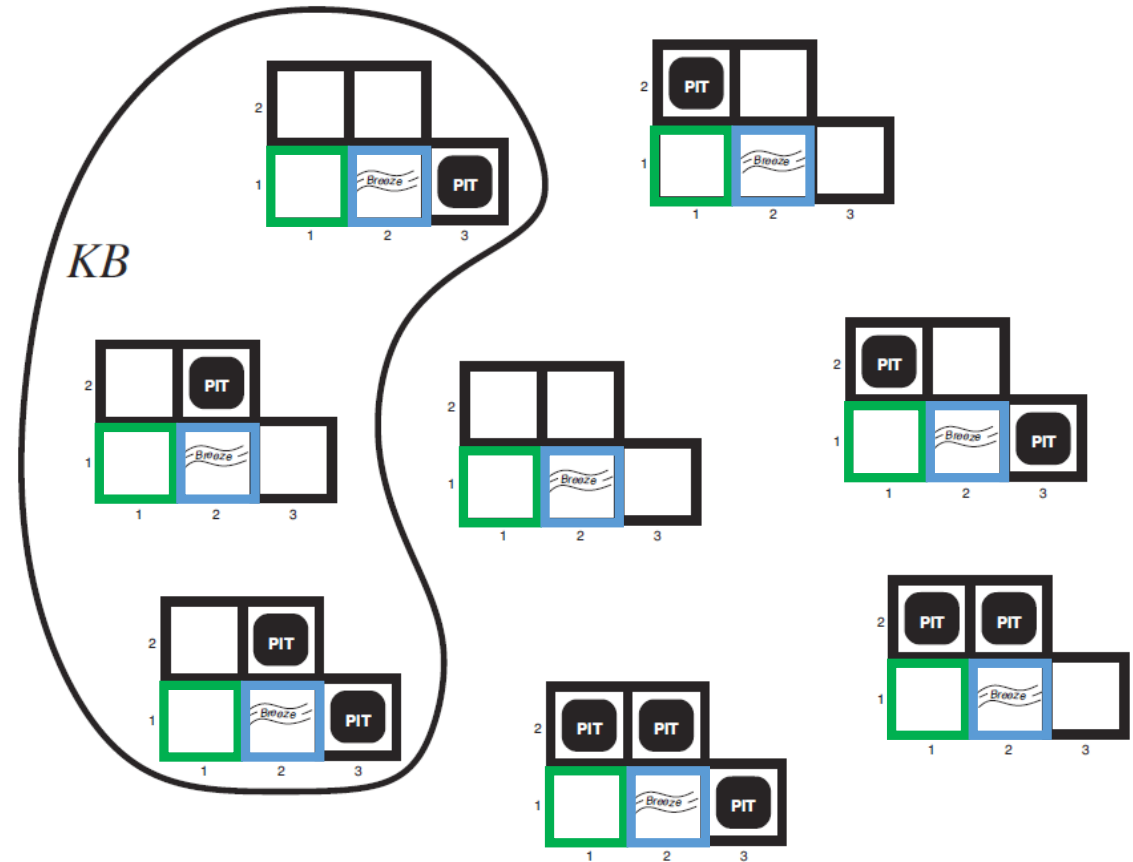
Entailment and implication are very much related

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

Wumpus World

Possible Models

- $P_{1,2} P_{2,2} P_{3,1}$
- Knowledge base
 - Breeze $\Rightarrow$ Adjacent Pit
 - Nothing in [1,1]
 - Breeze in [2,1]



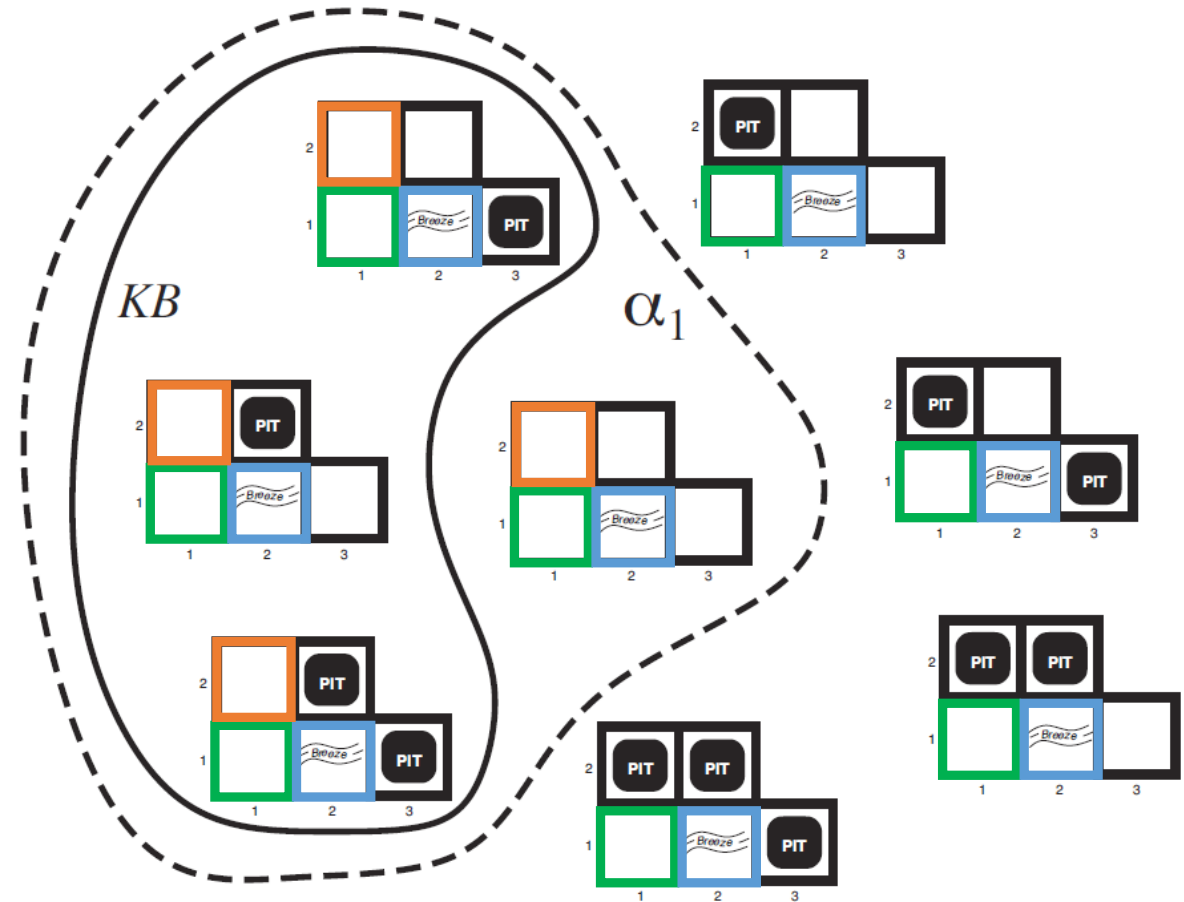
Entailment: $KB \models \alpha$

“KB entails α ” iff in every world where KB is true, α is also true

Wumpus World

Possible Models

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



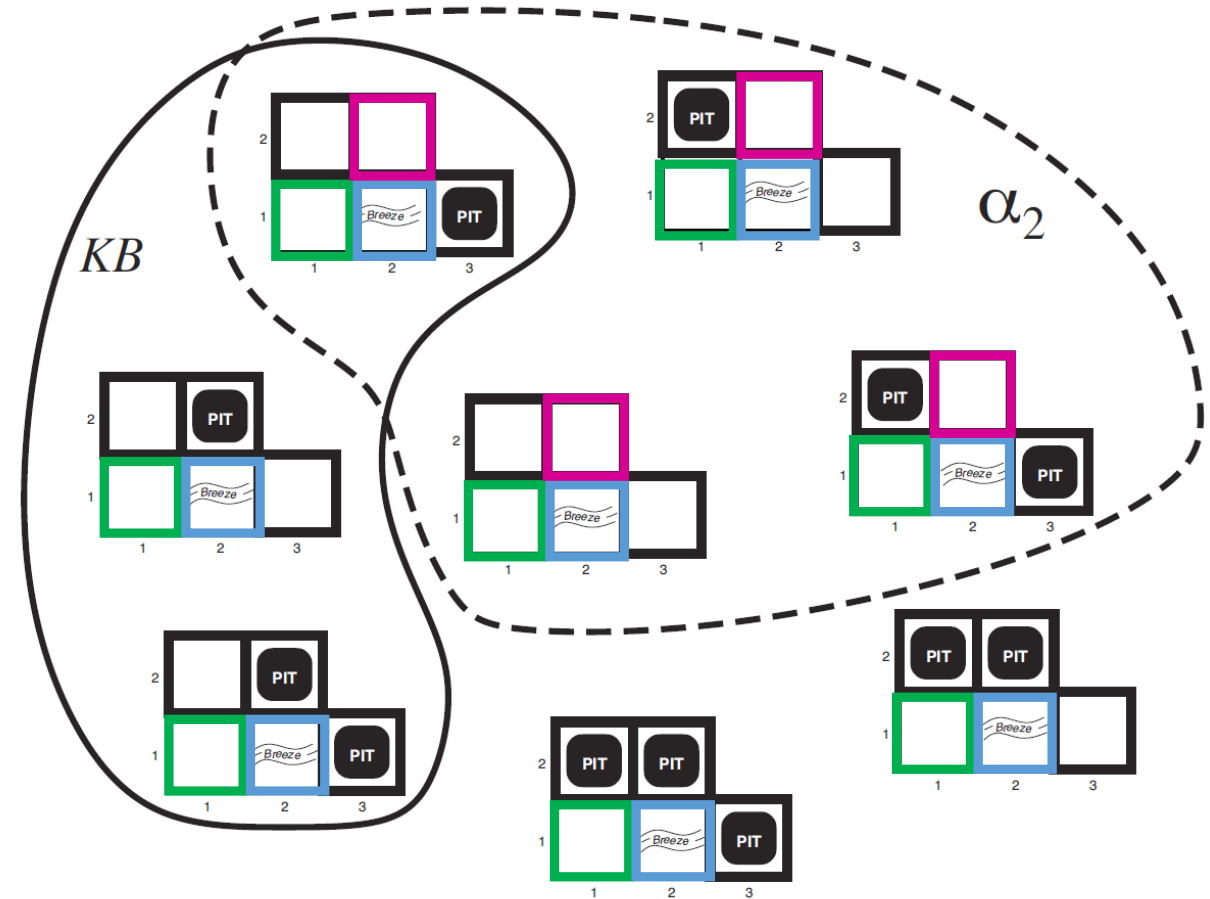
Entailment: $KB \models \alpha$

“KB entails α ” iff in every world where KB is true, α is also true

Wumpus World

Possible Models

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



Entailment: $KB \models \alpha$

“KB entails α ” iff in every world where KB is true, α is also true

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 4

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

Piazza Poll 4

Does the KB entail query C?

Yes!

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
	KB	0	0	0	0	0	1	0	1
Query	C	0	1	0	1	0	1	0	1

Entailment

How do we implement a logical agent that proves entailment?

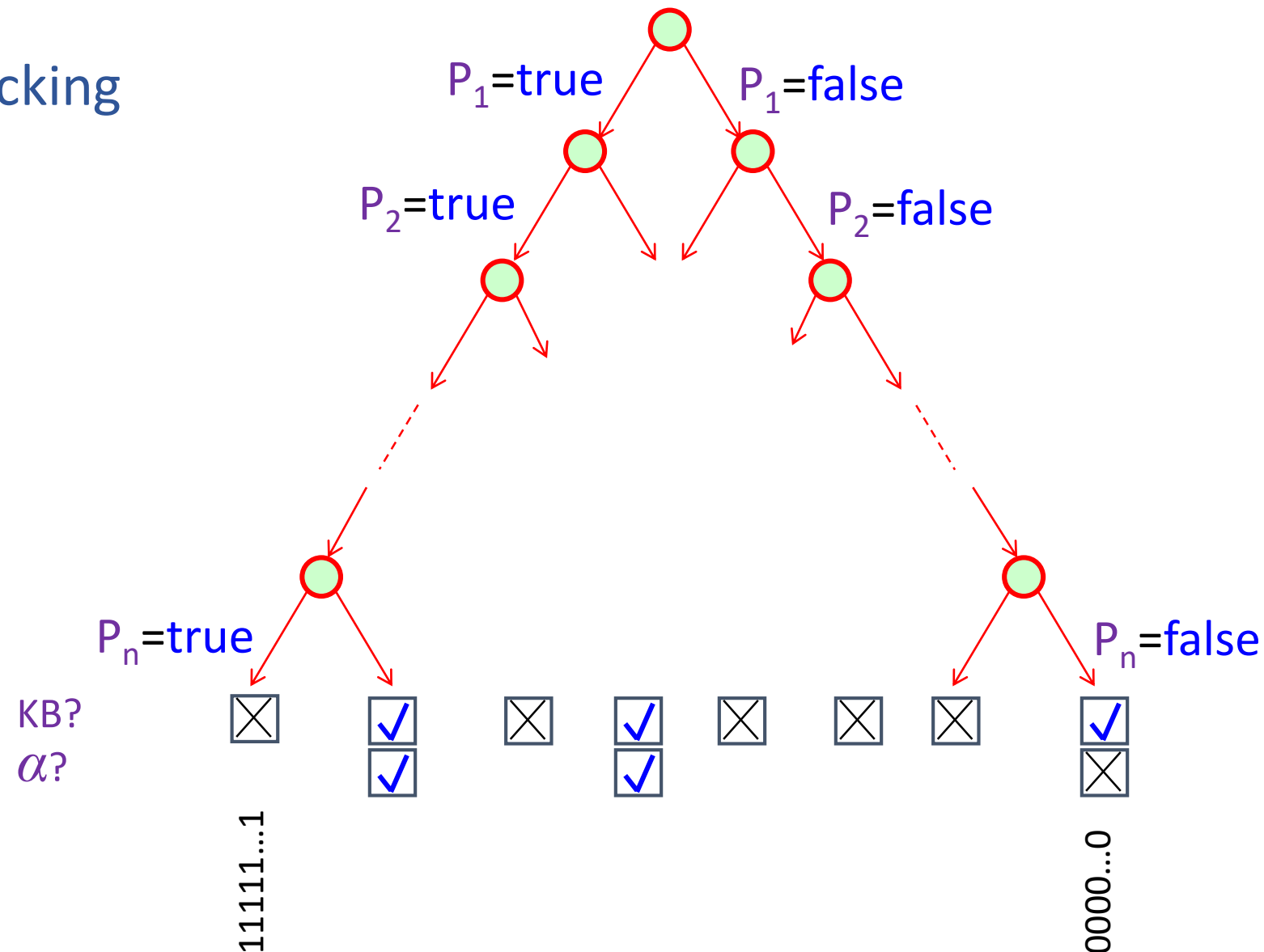
- Logic language
 - Propositional logic
 - First order logic
- Knowledge Base
 - Add known logical rules and facts
- Inference algorithms
 - Theorem proving
 - Model checking

Simple Model Checking

function **TT-ENTAILS?**(KB, α) returns true or false

Simple Model Checking, contd.

Same recursion as backtracking



Simple Model Checking

function **TT-ENTAILS?**(KB, α) returns true or false

 return **TT-CHECK-ALL**(KB, α , symbols(KB) $\cup$ 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 $\leftarrow$ first(symbols)

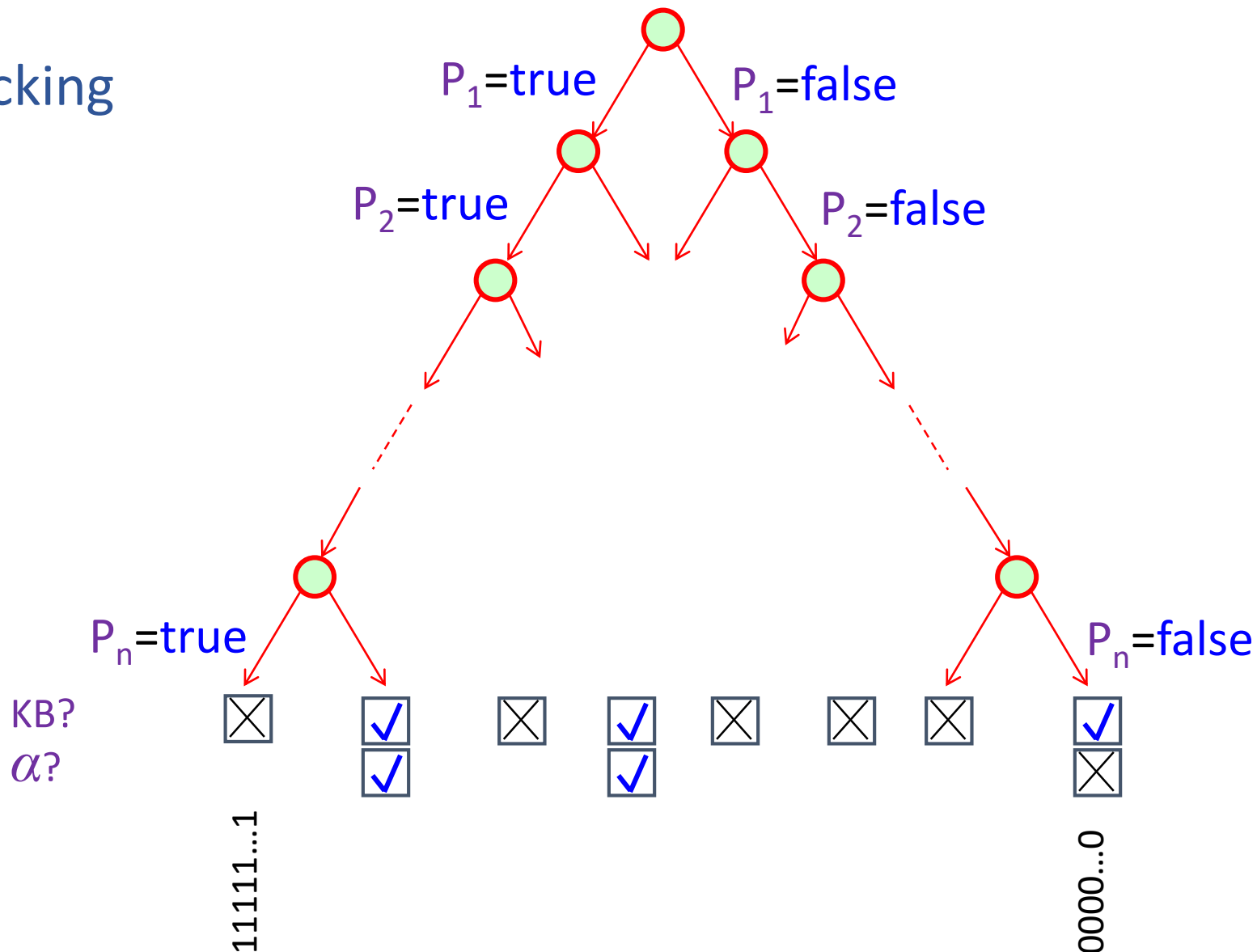
 rest $\leftarrow$ rest(symbols)

 return **and** (**TT-CHECK-ALL**(KB, α , rest, model $\cup$ {P = true})

TT-CHECK-ALL(KB, α , rest, model $\cup$ {P = false }))

Simple Model Checking, contd.

Same recursion as backtracking
 $O(2^n)$ time, linear space
Can we do better?



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 β is 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 $X_1 \wedge X_2 \wedge \dots \wedge X_n \Rightarrow Y$ and $X_1, X_2, \dots, X_n$
- 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 $\text{True} \Rightarrow X$)

Forward Chaining Algorithm

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

CLAUSES

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward Chaining Algorithm

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

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

inferred $\leftarrow$ a table, where inferred[s] is initially false for all s

agenda $\leftarrow$ a queue of symbols, initially symbols known to be true in KB

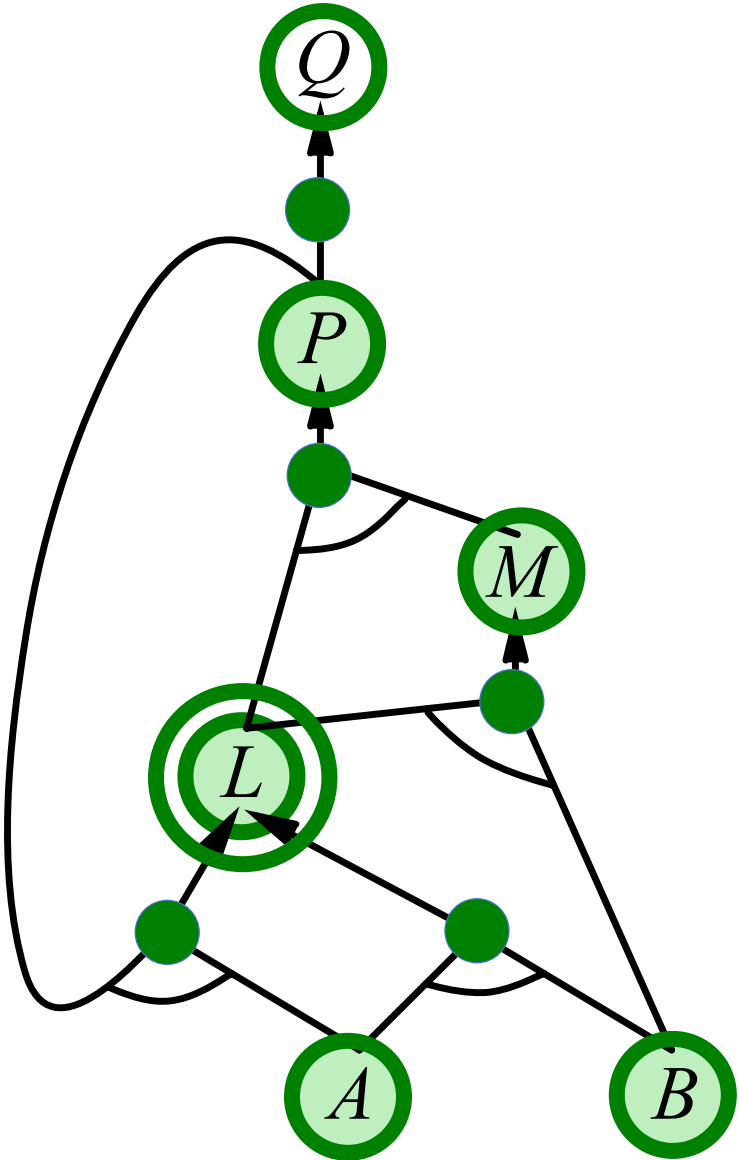
<i>CLAUSES</i>	<i>COUNT</i>	<i>INFERRED</i>	<i>AGENDA</i>
$P \Rightarrow Q$	1	A false	
$L \wedge M \Rightarrow P$	2	B false	
$B \wedge L \Rightarrow M$	2	L false	
$A \wedge P \Rightarrow L$	2	M false	
$A \wedge B \Rightarrow L$	2	P false	
A	0	Q false	
B	0		

Forward Chaining Example: Proving Q

<i>CLAUSES</i>	<i>COUNT</i>	<i>INFERRED</i>
$P \Rightarrow Q$	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~~ ~~M~~ ~~L~~ ~~P~~ ~~Q~~



Forward Chaining Algorithm

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

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

inferred $\leftarrow$ a table, where inferred[s] is initially false for all s

agenda $\leftarrow$ a queue of symbols, initially symbols known to be true in KB

while agenda is not empty do

 p $\leftarrow$ Pop(agenda)

 if p = q then return true

 if inferred[p] = false then

 inferred[p] $\leftarrow$ 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 $a_1 \wedge \dots \wedge a_k \Rightarrow b$ is false in *m*

Then $a_1 \wedge \dots \wedge a_k$ 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 $KB \models q$, *q* is true in every model of KB, including *m*

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

Inference Rules

Modus Ponens

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

Notation Alert!

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 $\wedge \neg \alpha$ cannot be satisfied

Return true, KB entails α

B. No new clauses added

Return false, KB does not entail α

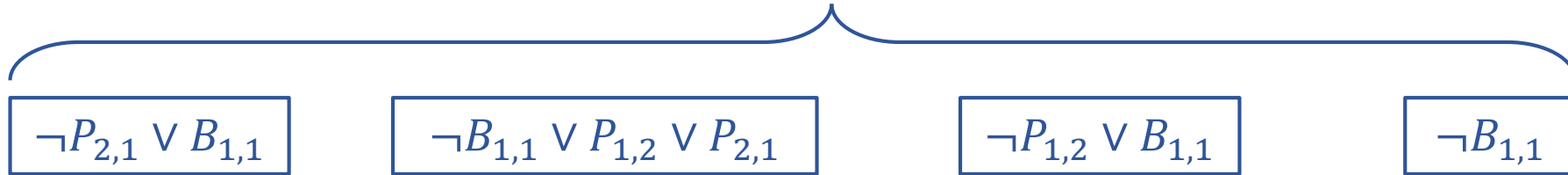
Resolution

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

General Resolution

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

Knowledge Base



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

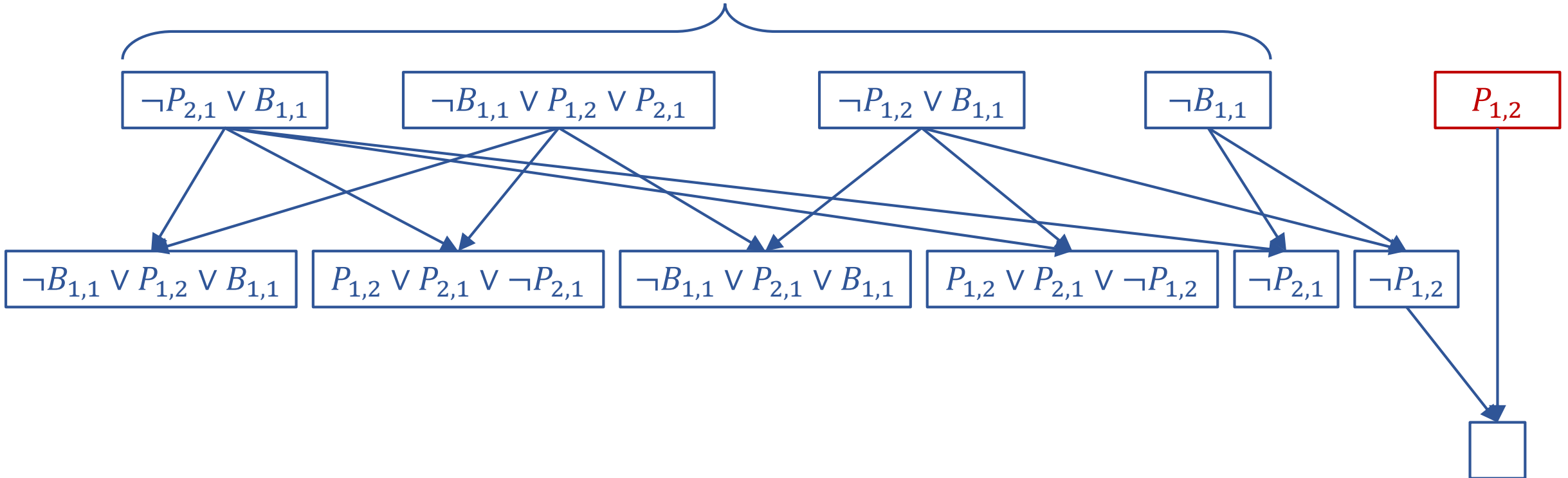
Resolution

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

General Resolution

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

Knowledge Base



Resolution

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

clauses $\leftarrow$ the set of clauses in the CNF representation of $\text{KB} \wedge \neg\alpha$

new $\leftarrow \{ \}$

loop do

for each pair of clauses C_i, C_j in clauses do

resolvents $\leftarrow$ PL-RESOLVE(C_i, C_j)

if resolvents contains the empty clause then

return true

new $\leftarrow$ new $\cup$ resolvents

if new $\subseteq$ clauses then

return false

clauses $\leftarrow$ clauses $\cup$ 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 😞