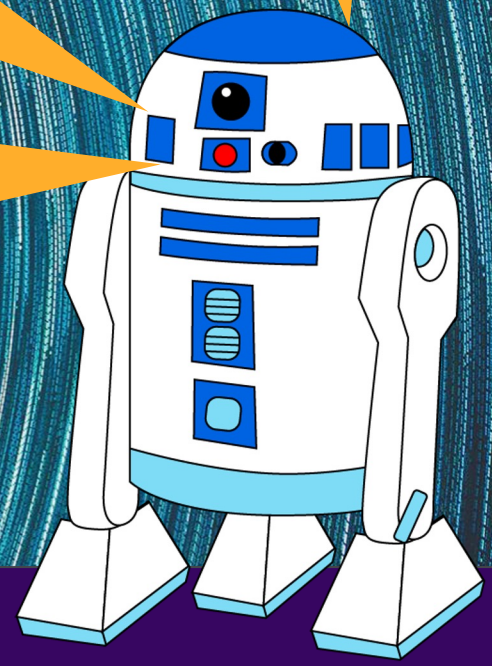
CIS 4210/5210:
ARTIFICIAL INTELLIGENCE

Logical Agents

HW5 is due on Wednesday.

Midterm 1 will be in-class on Tuesday October 10.

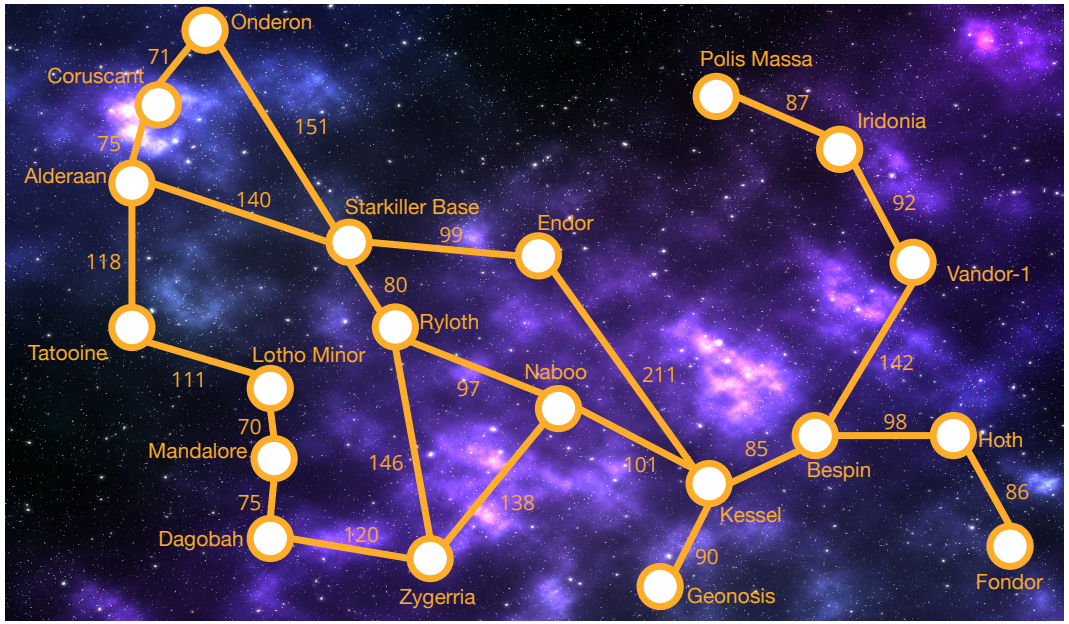
It will cover all modules so far, including this one.



Knowledge-based Agents

Knowledge-based agents use a process of **reasoning** over an internal **representation** of knowledge to decide what action to take.

So far, our **problem-solving** agents have performed a **search** over **states** to find a plan. The representation of states has been **atomic**.



Limited to commands like "Navigate to Kessel"

"Take me to the nearest habitable planet where I can store my perishable cargo"

Knowledge-based Agents

A central component of a knowledge-based agent is a **knowledge base** or KB.

A KB contains a set of **sentence** that are written in a **knowledge representation language.** The sentence contains some assertion about the world.

Natural language sentences	Knowledge representation language sentence
Hoth is a planet	`planet(hoth)`
Hoth is habitable	`habitable(hoth)`
Hoth is far from its sun	`far_from(hoth, sol)`
If a planet is far from its sun then it is cold	`planet(x) and sun(y) and far_from(x,y) →cold(x)`

Knowledge-based Agents

There are two kinds of sentences:

- **Axioms** – a sentence that is given
- **Derived sentences** – a new sentence that is derived from others sentences

The process of deriving new sentences from old sentences is called **inference**.

A KB can initially contain some **background information** about the world, and a knowledge-based agent can add to the information in the KB through its observations of the world.

In addition to asserting new knowledge into its KB, a knowledge-based agent can also query the KB and ask it to derive new knowledge to select what action it should take.

Hunt the Wampas

Wampa World

Our **knowledge-based agent**, R2D2, explores **a cave** consisting of **rooms** connected by passageways.

Lurking somewhere in the cave is the **Wampa**, a beast that eats any agent that enters its room.

Some rooms contain bottomless **pits** that trap any agent that wanders into the room.

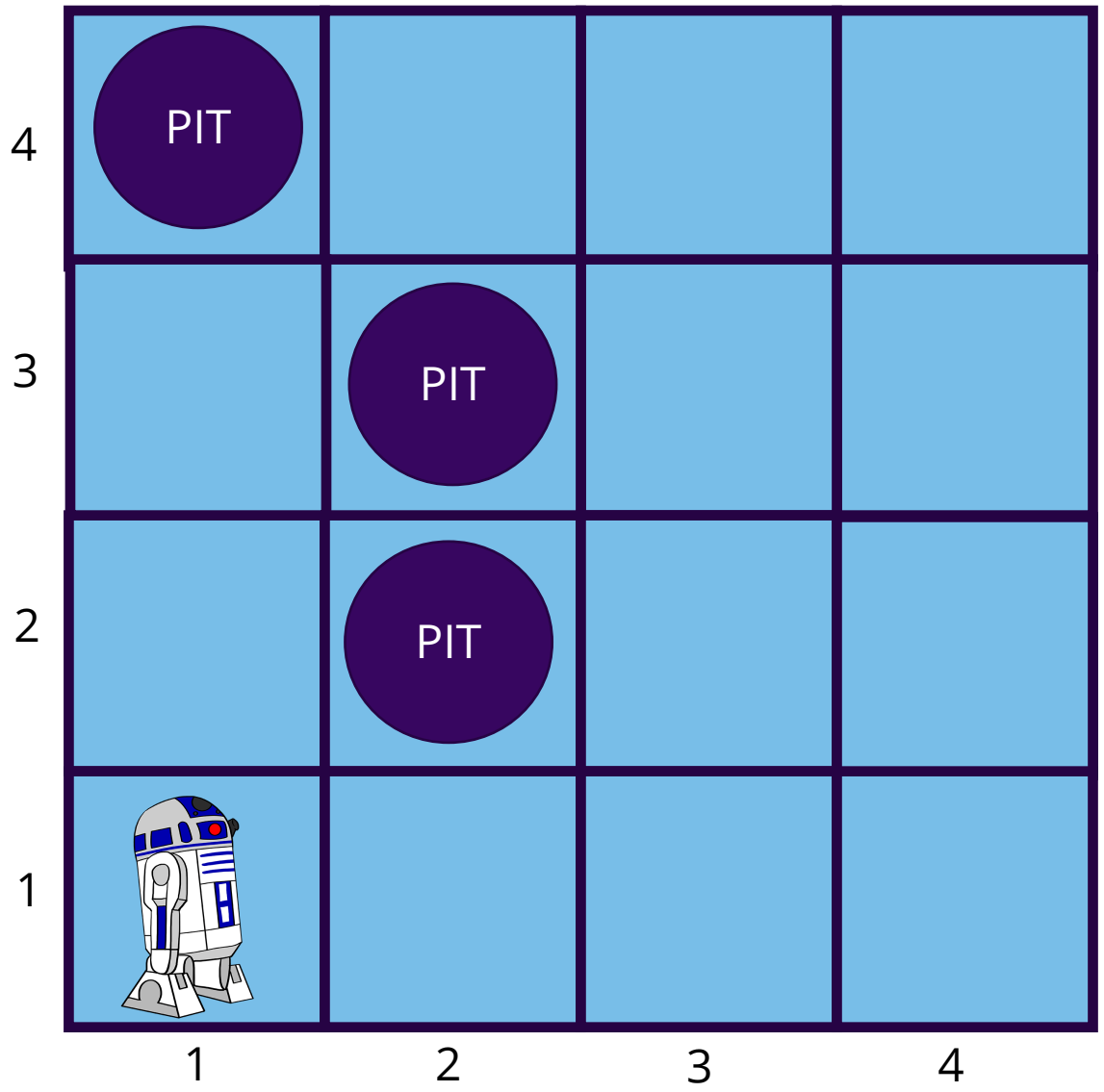
In one room is master **Luke**.

The goal is:

- collect Luke
- exit the world
- without being eaten

Wampa World

Environment: A 4x4 grid of rooms. The agent starts in the square [1,1]. Wampa and Luke are randomly placed in other squares. Each square can be pit with 20% probability.

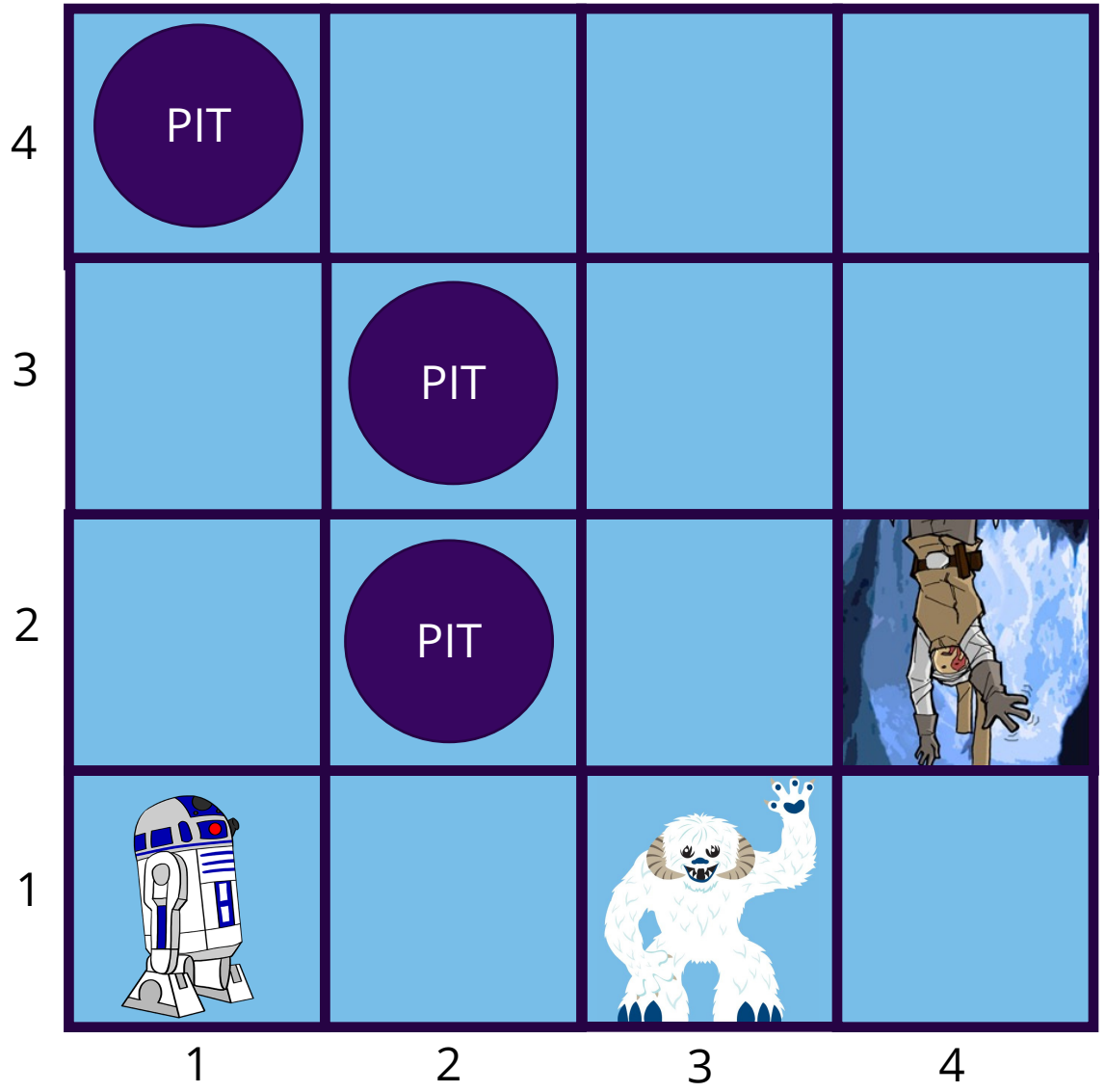


Wampa World

Performance measure:

+1000 points for rescuing Luke and leaving the cave

-1000 for falling into a pit or being eaten by the Wampa
-1 for each action taken
-10 for using up your blaster fire



Wampa World

Actuators:

R2 can move *Forward, TurnLeft, Turn right.*

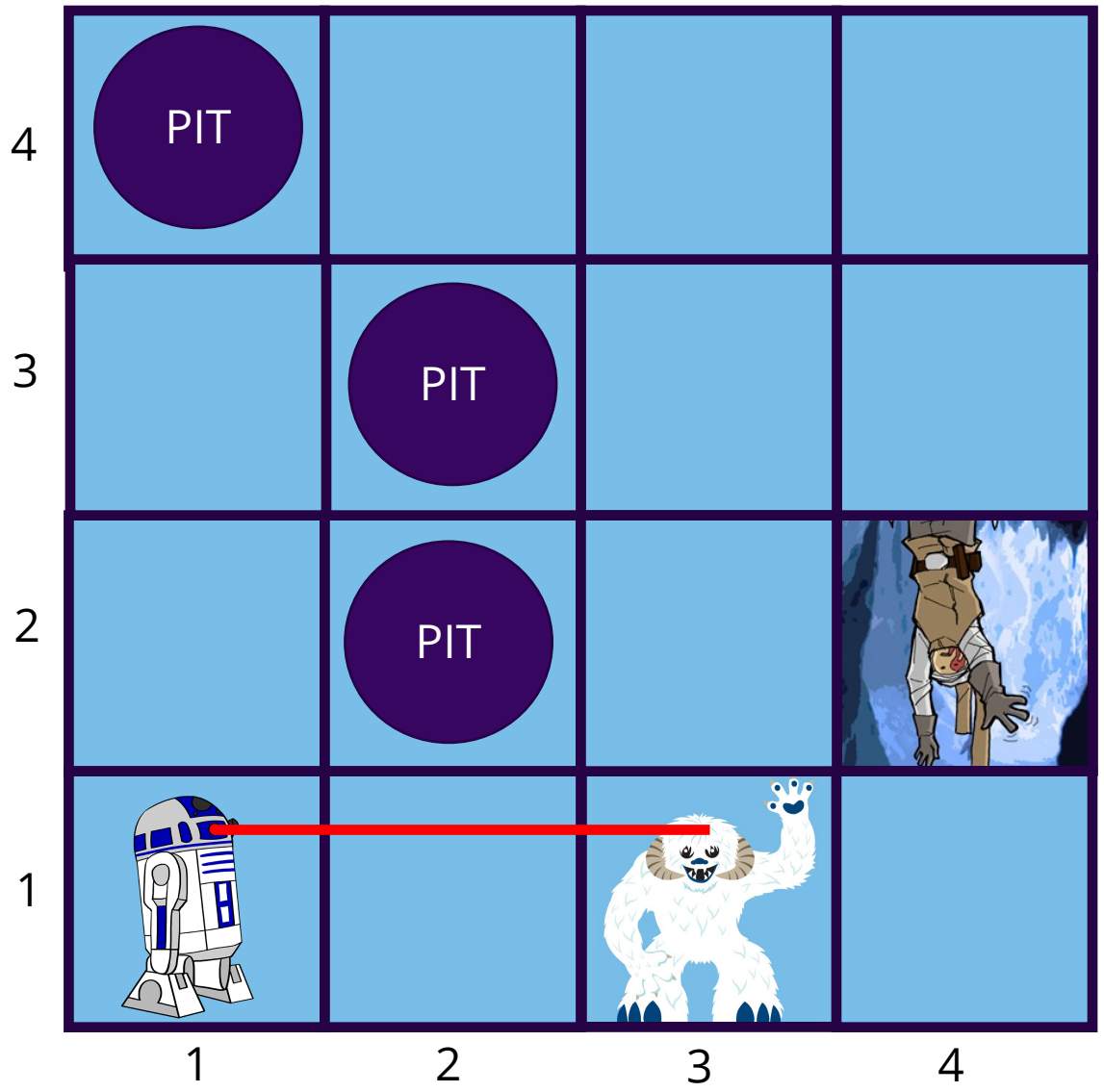
Agent dies if it moves into a pit or a Wumpus square.

Grab can pick up Luke.

Shoot fires blaster bolt in a straight line in the direction that R2D2 is facing.

If the blaster hits the Wampa, it dies. R2 only has enough power for one shot.

Climb gets R2 out of the cave but only works in [1, 1]



Wampa World

Sensors:

In each square adjacent to the Wampa, R2D2's olfactory sensor perceives a *Stench*

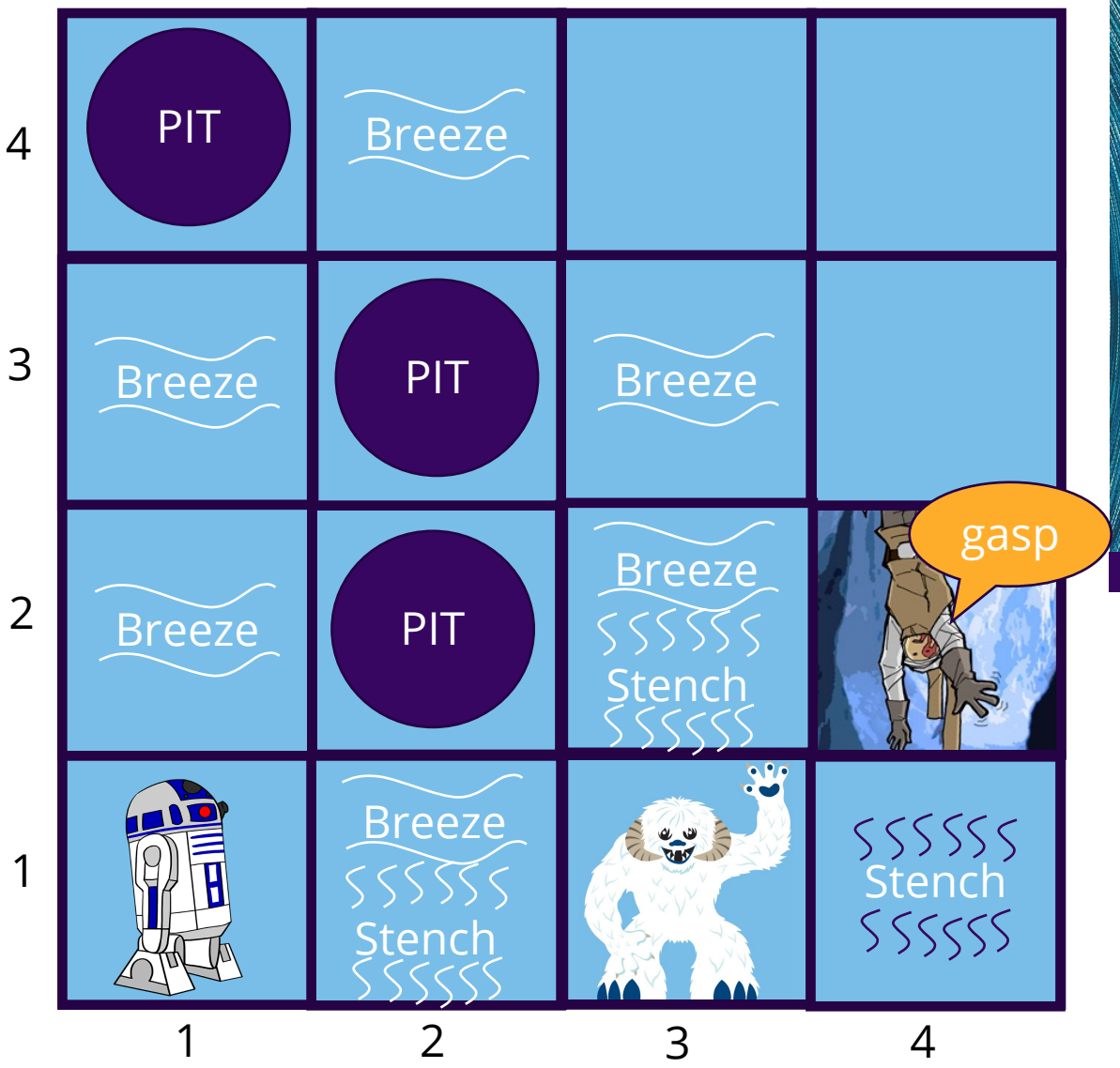
In each square adjacent to a pit, R2D2's wind sensor perceives a *Breeze*

In the square with Luke, R2D2's audio sensor perceives a *Gasp*

When R2D2 walks into a wall it perceives a *Bump*

When the Wampa is killed , R2D2's audio sensor perceives a *Scream*

Percept=[*Stench, Breeze, Gasp, None, None*]



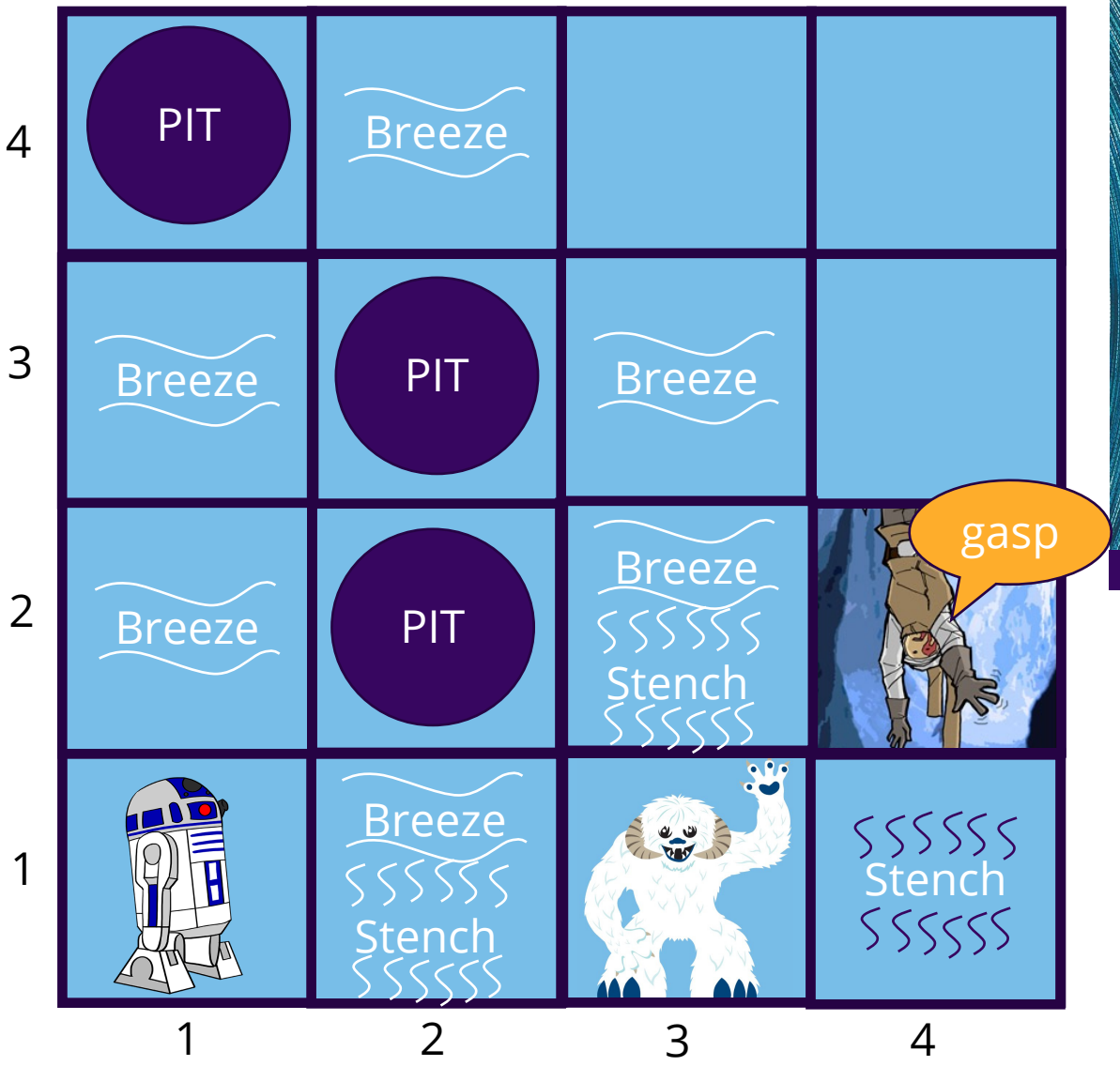
Wampa World

Deterministic, discrete, static, single-agent (Wampa doesn't move)

Sequential because reward doesn't come for many steps

Partially observable because some parts of the state are not directly perceivable:

- Location of Luke, Wampa, and pits aren't directly observable.



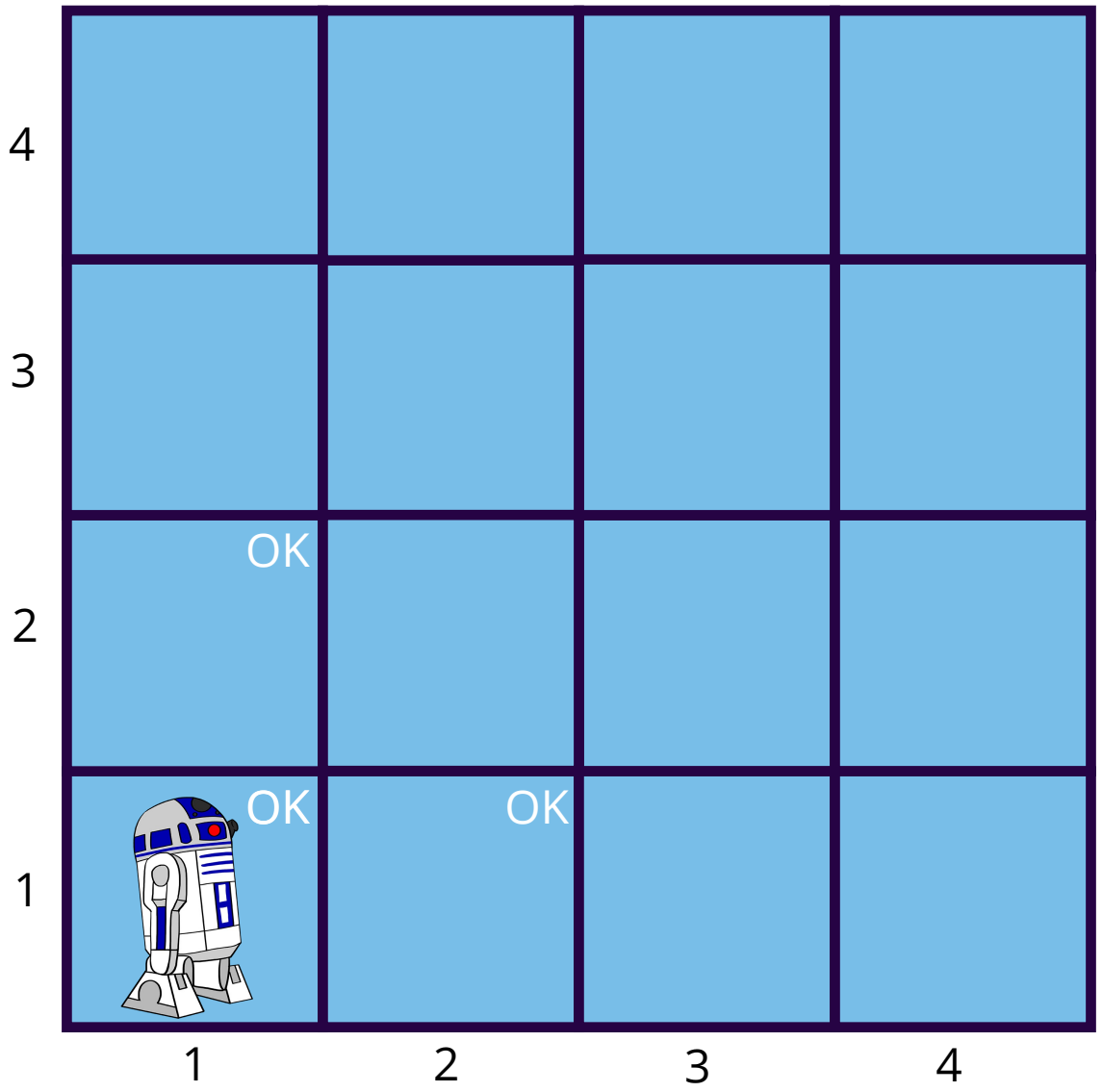
Wampa World Walkthrough

New world!

R2D2 starts in [1,1]

Percept=[*None, None, None, None, None*]

What can we conclude about [1,2] and [2,1]?

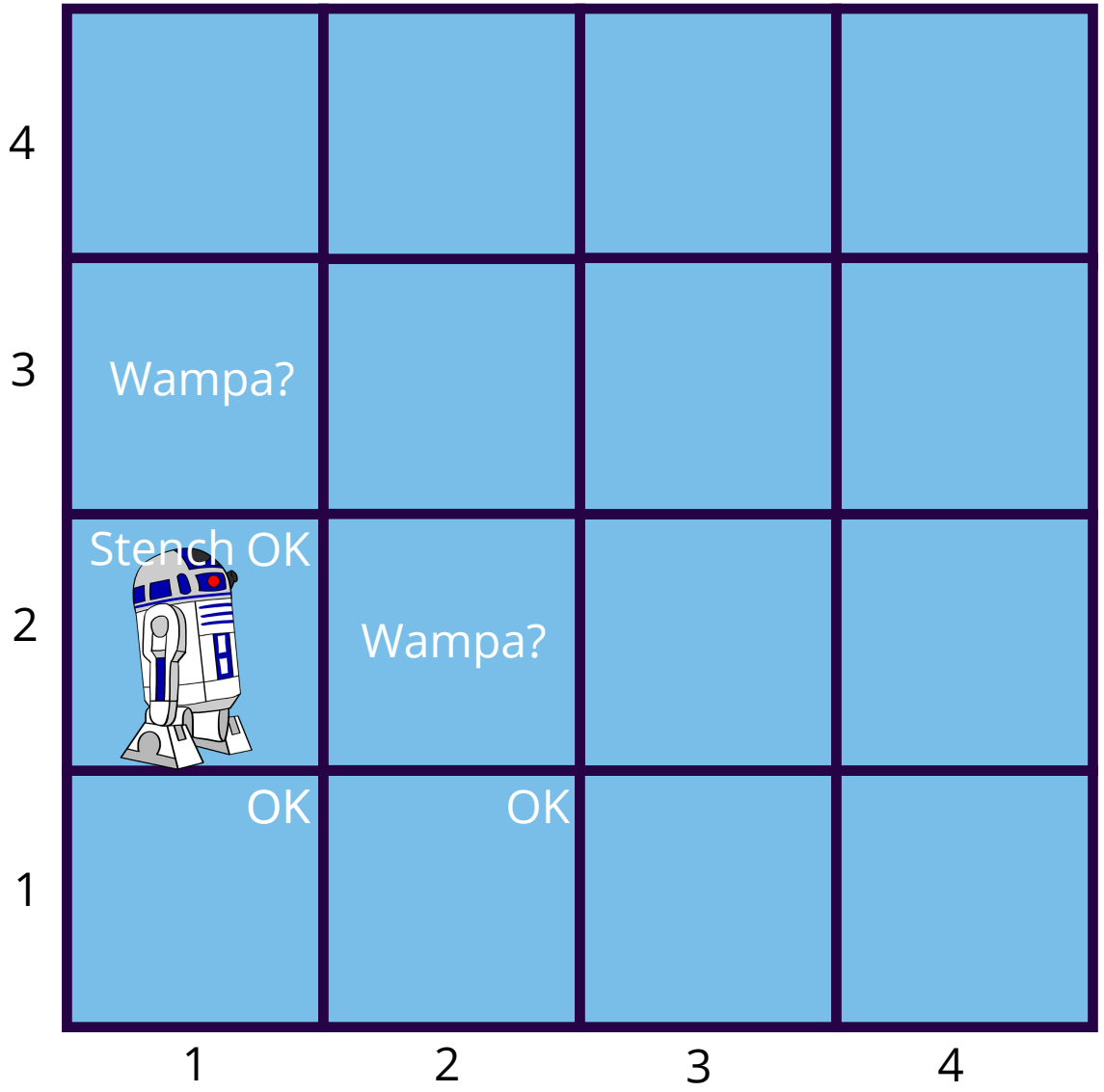


Wampa World Walkthrough

R2D2 moves to [1,2]

Percept=[*Stench, None, None, None None*]

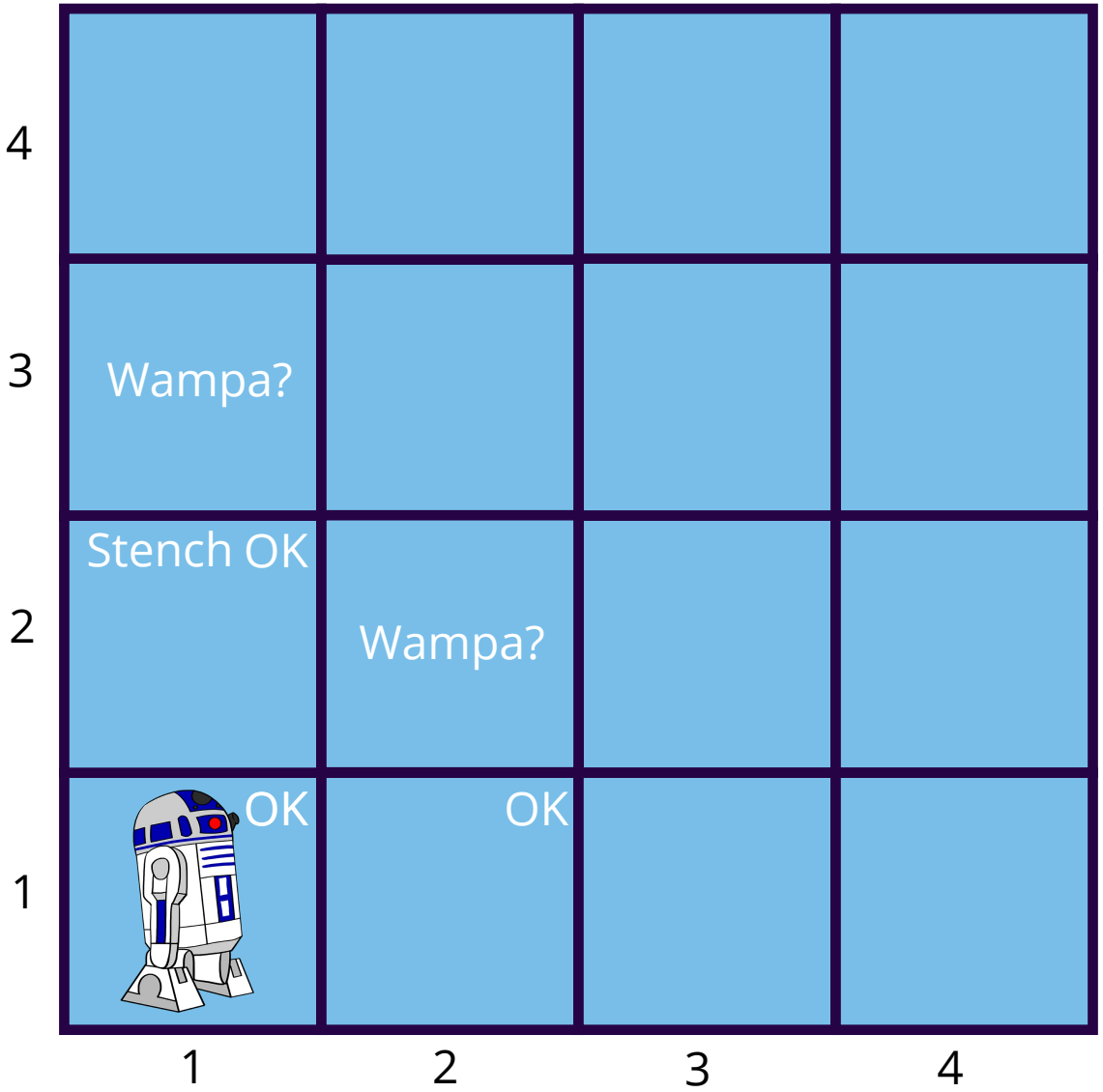
What can we conclude about [1,3] and [2,2] from the Stench?



Wampa World Walkthrough

R2D2 moves back to [1,1] and gets the same percept vector as before

Percept=[*None, None, None, None None*]

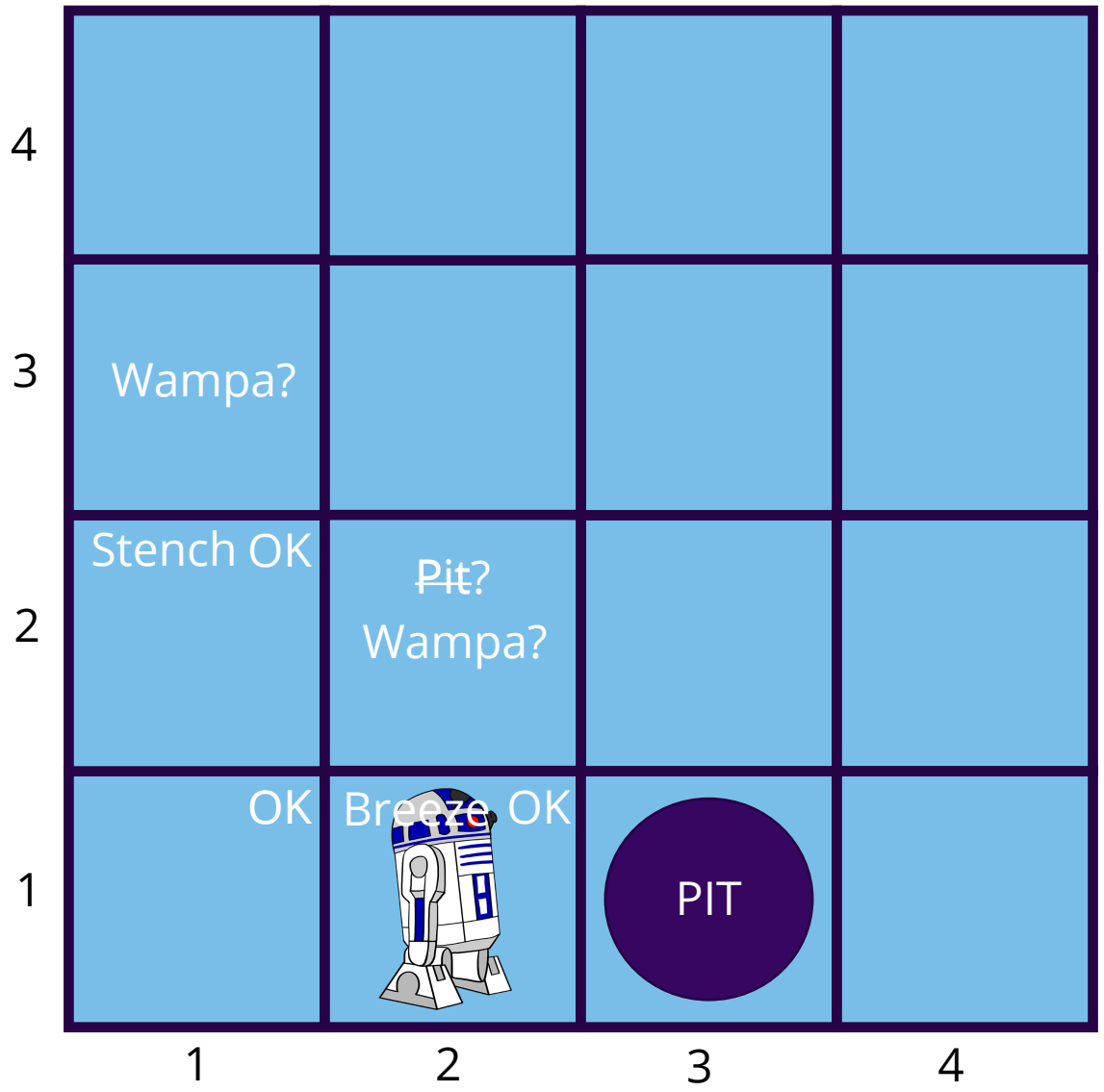


Wampa World Walkthrough

R2D2 moves to [2,1]

Percept=[*None, Breeze, None, None None*]

What can we conclude about [3,1] and [2,2] based on the Breeze?

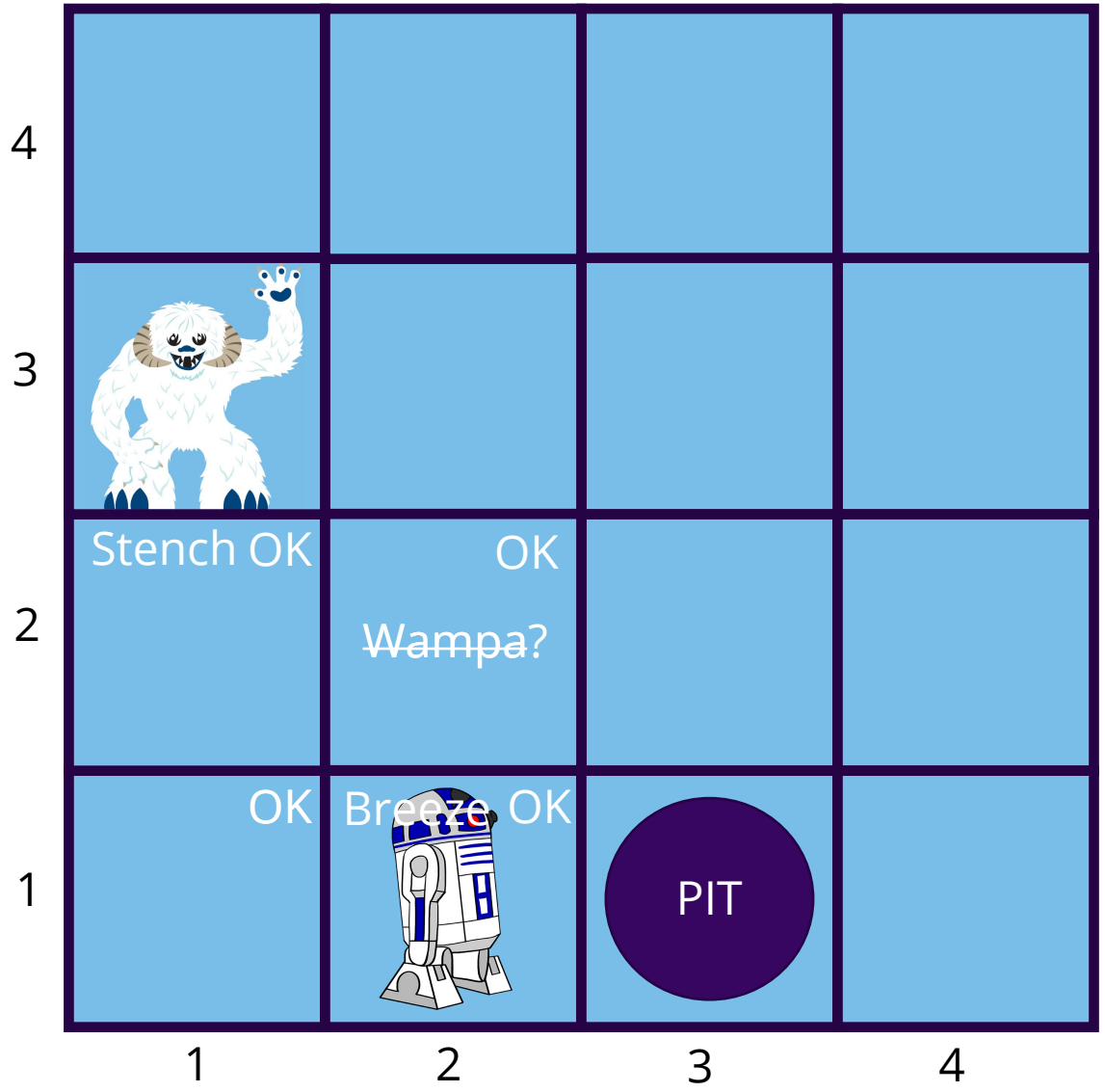


Wampa World Walkthrough

R2D2 moves to [2,1]

Percept=[*None, Breeze, None, None None*]

What can we conclude about [2,2] and [1,3] based on the lack of a Stench here?

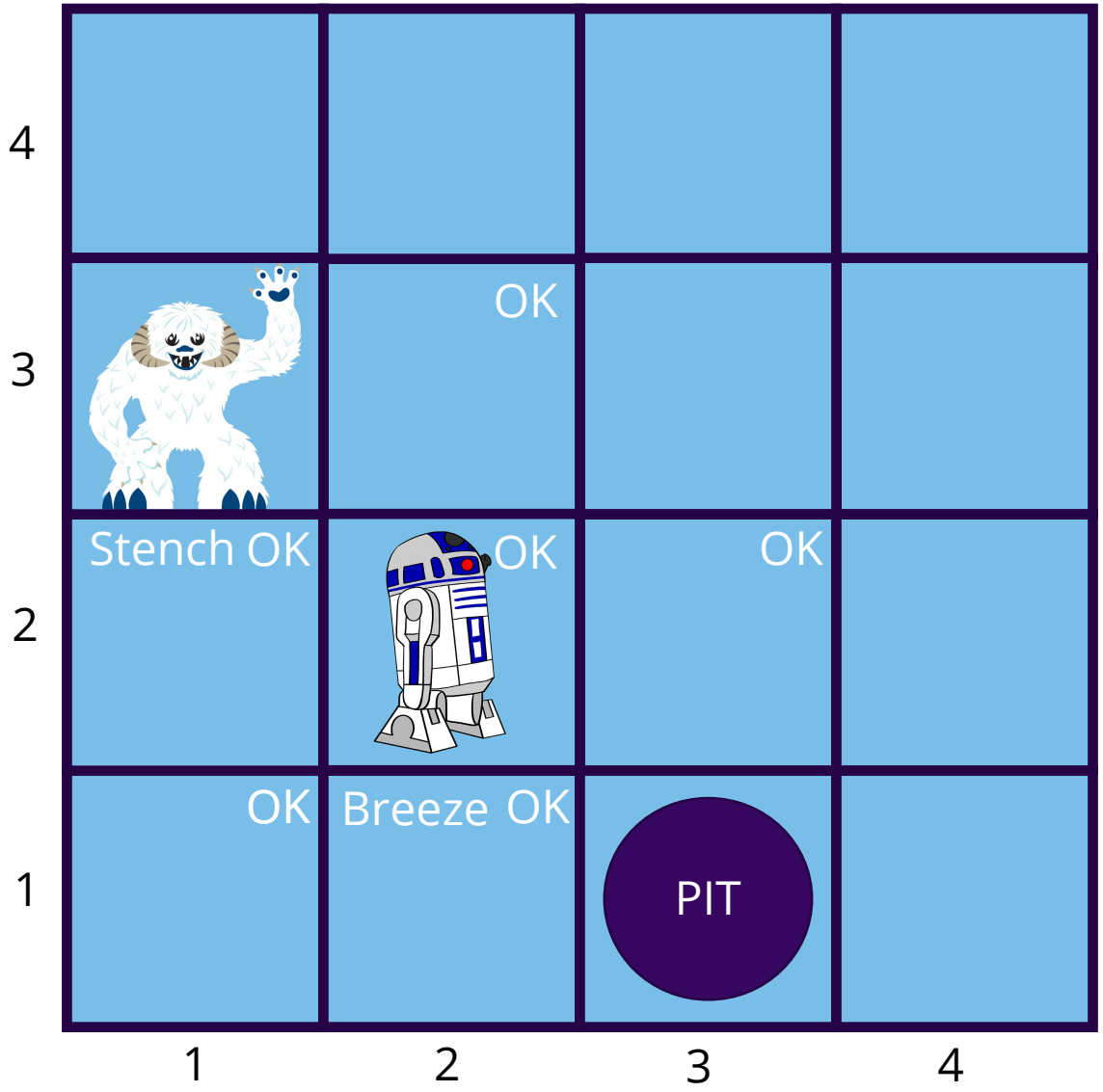


Wampa World Walkthrough

R2D2 moves to [2,2]

Percept=[*None, None, None, None, None*]

What can we conclude about [2,3] and [3,2]?

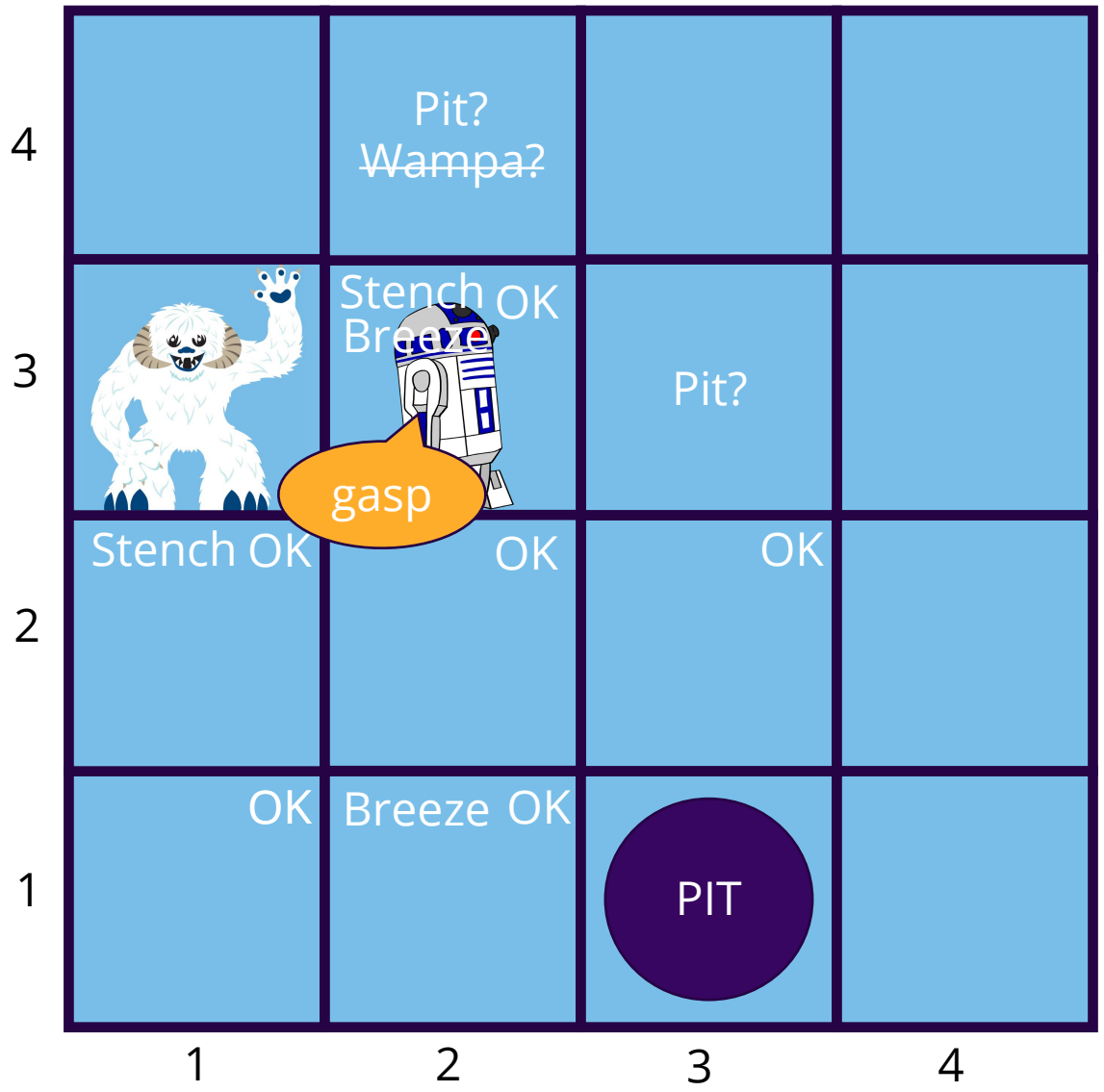


Wampa World Walkthrough

R2D2 moves to [2,3]

Percept=[*Stench, Breeze, Gasp, None, None*]

What can we conclude about [2,3]?

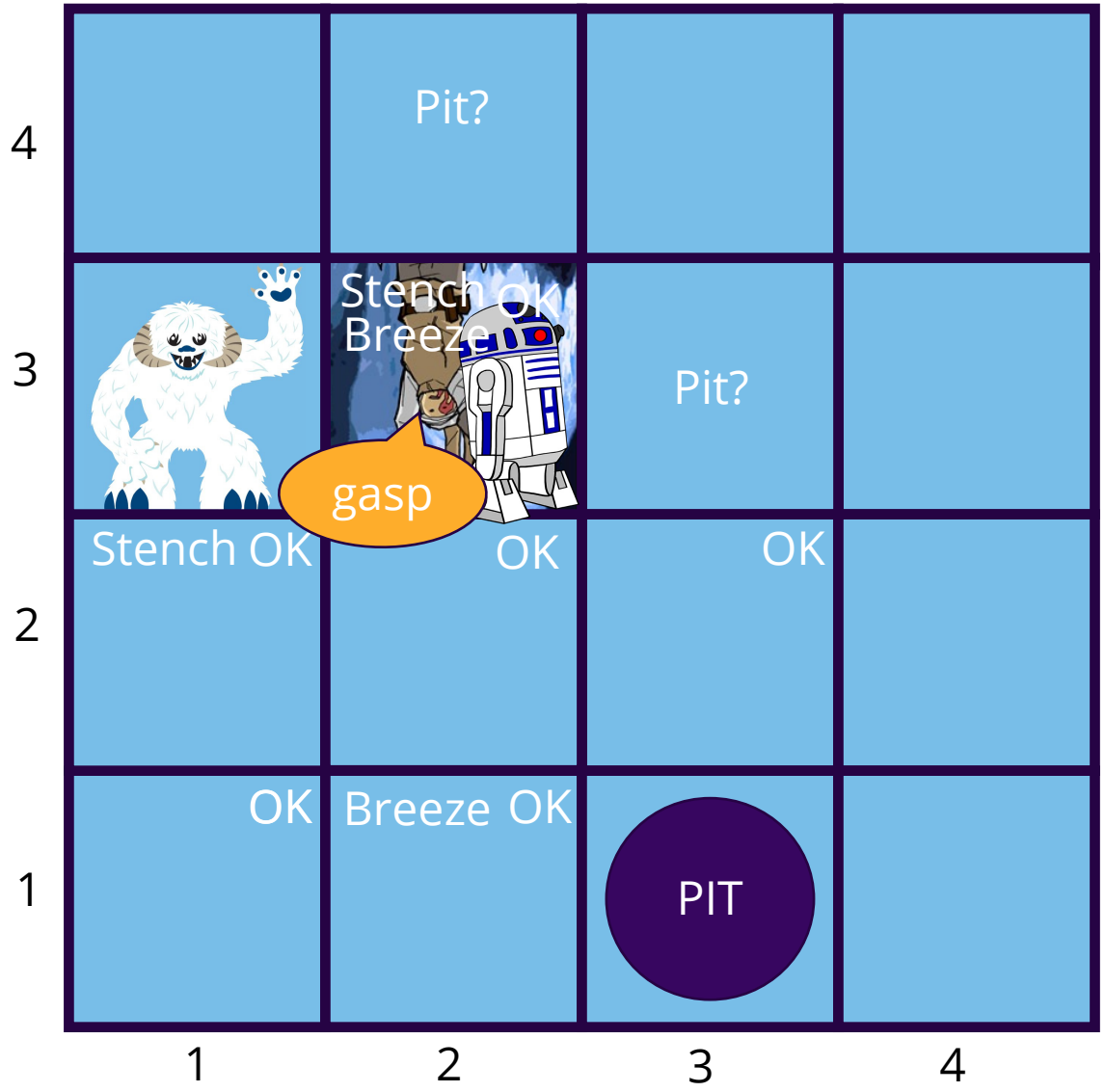


Wampa World Walkthrough

R2D2 moves to [2,3]

Percept=[*Stench, Breeze, Gasp, None, None*]

We heard a Gasp, so Luke is here!



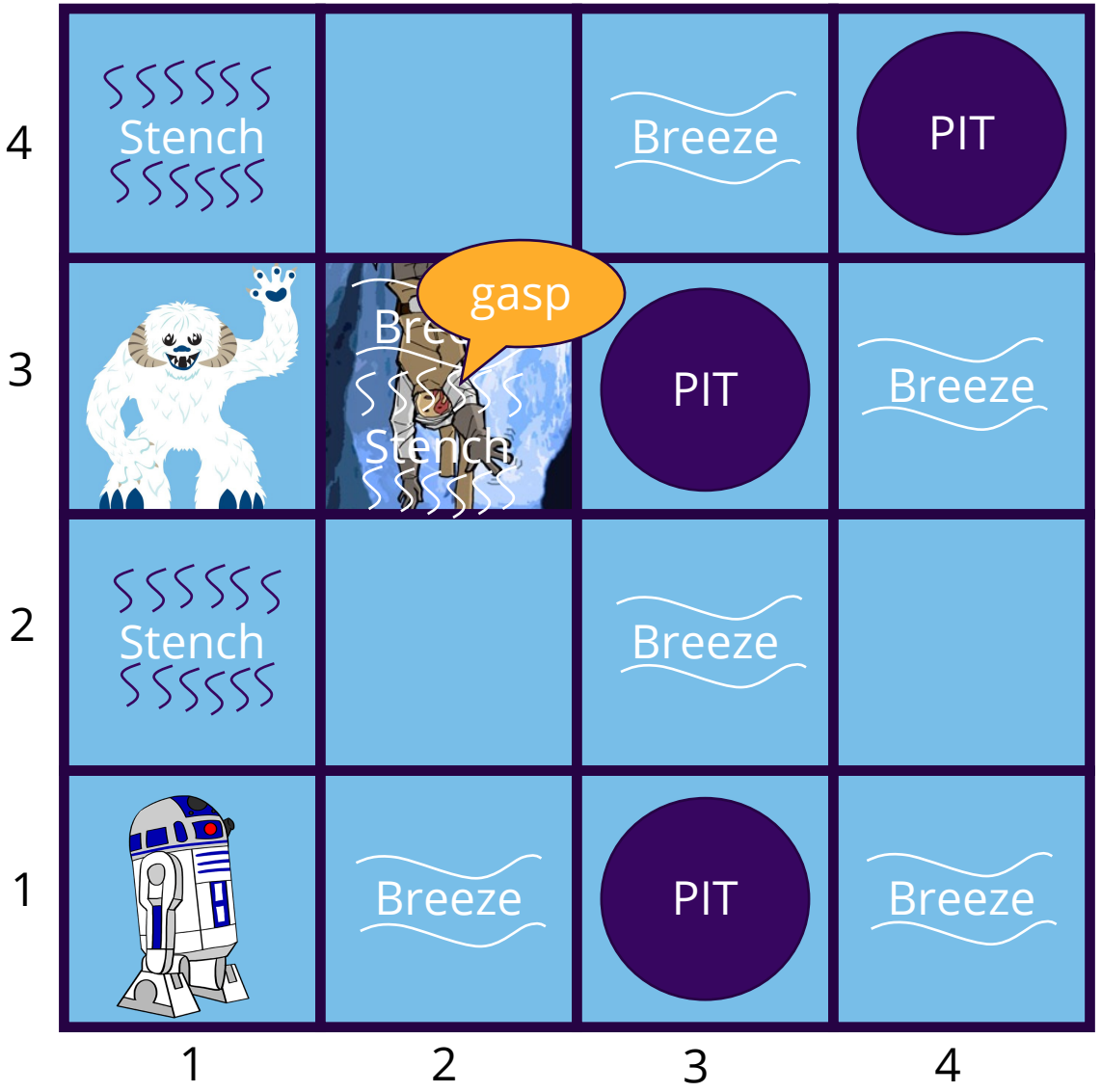
Wampa World Revealed

Deterministic, discrete, static, single-agent (Wampa doesn't move)

Sequential because reward doesn't come for many steps

Partially observable because some parts of the state are not directly perceivable:

- Location of Luke, Wampa, and pits aren't directly observable.



Logic

Logic can serve as a general class of **representations** for knowledge-based agents. Here we are going to examine **Propositional Logic.**

- KB consists of **sentences** in the **representation language**
- Representation language has a **syntax** that specifies sentences that are well-formed
- A logic defines the **semantics** of the sentences, which is their **meaning**
- The semantics defines the **truth** of each sentence with respect to a **possible world,** which we will often call a **model**.

Possible worlds and models

- Models are mathematical abstractions that have a fixed set of **truth values** which are **{true, false}** for each sentence.
- If sentence **α** is true in model **m** then we say
  - **m** satisfies **α**, or
  - **m** is a model of **α**
- We use the notation **M(α)** to mean the **set of all models** of **α**.

For instance, **α** could be a sentence that means “there is no pit in [2,2]”. In that case, **M(α)** would be all instances of Wampa World where [2,2] doesn’t have a pit.

Logical Entailment

Once we have a notion of truth, we can start to define **logical reasoning**. Logical reasoning involves the **entailment** relation between sentence.

In plain English, entailment is the idea that a sentence **follows logically** from another sentence.

To write sentence **α** entails sentence **β** in mathematical notation we use the **⊨ symbol:**

$$\alpha \models \beta$$

The definition is

$\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$

This means that **α** is more specific, or stronger than, **β.** For instances, **β** could mean that “The agent is a robot” and **α** could mean “The agent is an astromech”.

Knowledge Base

The KB can be thought of as a set of sentences.

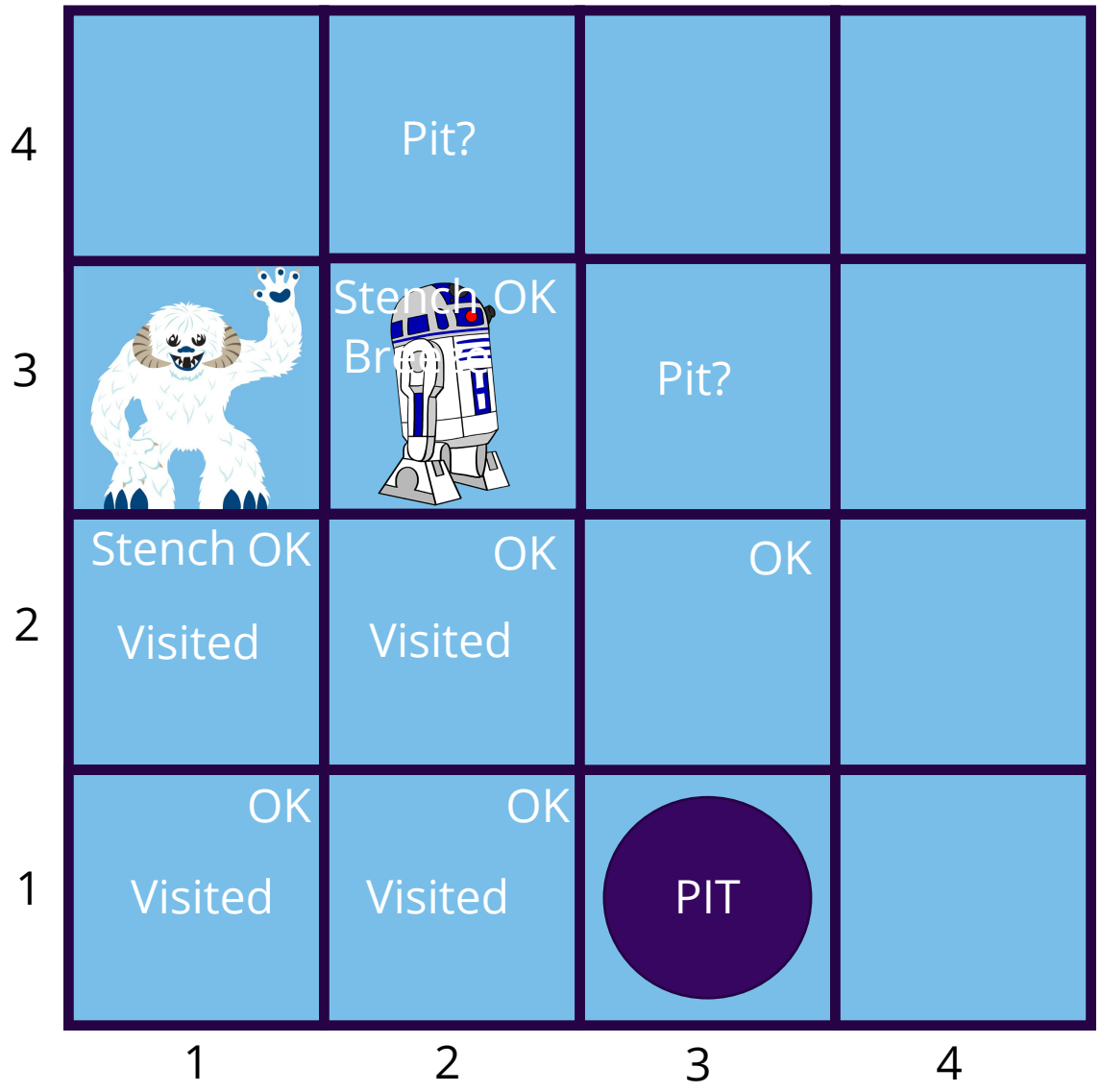
α_1 = "There is no pit in [1,2]"

α_2 = "There is a pit in [3,1]"

α_3 = "There is a wampa in [1,3]"

The KB is false in models that contradict what the agent knows. For example, the KB is false in any model **m** where [1,2] contains a pit.

Possible Worlds is the process of enumerating all Possible Worlds that are compatible with the KB. **M(**KB) ⊆ **M(**α_1 **)**



Logical inference

Entailment can be applied to derive conclusions, which is the process of **logical inference.**

We can think about the consequences of a KB as a large set of additional sentences that are entailed given the sentences that have been added to the KB.

We would like to design **inference algorithms** to enumerate these sentences.

When an inference algorithm **i** allows us to conclude that **α** is true, then we write

$$\mathbf{KB} \vdash_{\mathbf{i}} \boldsymbol{\alpha}$$

"**α** is derived from **KB** by **i**"

Propositional Logic

Propositional Logic

Atomic sentences are represented with a single **propositional symbols**.

Propositional symbols **stand for a statement** that can be true or false.

For example, $\mathbf{W_{1,3}}$ is a propositional symbol that we choose to stand for

"There is a Wampa at location [1,3]"

That statement can be true or false.

The symbol **FacingEast** could stand for "The agent is currently facing East".

Propositional Logic

Complex sentences are constructed from simpler ones using parentheses and **logical connectives**.

Logical Connective	Meaning
¬ (not)	$\neg \mathbf{W_{1,3}}$ is the negation of $\mathbf{W_{1,3}}$
∧ (and)	$\mathbf{W_{1,3}} \wedge \mathbf{P_{3,1}}$ is called a conjunction
∨ (or)	$\mathbf{W_{1,3}} \vee \mathbf{P_{3,1}}$ is called a disjunction
⟹ (implies)	$\mathbf{W_{1,3}} \Longrightarrow \mathbf{S_{1,2}}$ is called an implication. $\mathbf{W_{1,3}}$ is its **premise or antecede** and $\mathbf{S_{1,2}}$ is its **conclusion or consequence**
⇔ (if and only if)	$\mathbf{W_{1,3}} \Leftrightarrow \neg \mathbf{W_{3,4}}$ is called an biconditional

Truth Tables

Negation

P	¬P
True	False
False	True

"**It is not the case that** the Death Star is a moon" is **true** because "the Death Star is a moon" is **false**.

"**It is not the case that Wampas** smell bad" is **false** because "Wampas smell bad" is **true**.

Truth Tables

Conjunction

P	Q	P ∧ Q
True	True	True
True	False	False
False	True	False
False	False	False

P and Q are both true:
"Wampas smell bad **and** Tauntauns smell bad." (This sentence is true)

P is true and Q is false:
"Wampas smell bad **and** Tauntauns are robots." (This sentence is false)

P is false and Q is false:
"Wampas smell good **and** Tauntauns are robots." (This sentence is false)

Truth Tables

Disjunction

P	Q	P V Q
True	True	True
True	False	True
False	True	True
False	False	False

P and Q are true:
“Wampas smell bad **or** Tauntauns smell bad.” (This sentence is true)

P is true and Q is false:
“Wampas smell bad **or** Tauntauns are robots.” (This sentence is true)

P is false and Q is false:
“Wampas smell good **or** Tauntauns are robots.” (This sentence is false)

Truth Tables

Conditional

P	Q	P ⟹ Q
True	True	True
True	False	False
False	True	True
False	False	True

Which of these scenarios did break my promise to you?

To understand why the conditional is defined this way assume that I tell y… …s P ⟹ Q:
If you join the dark side then we will rule the galaxy together.

In which of these four scenario did I tell a lie?

1. **You join the dark side,** and **we rule the galaxy together.** (Both P and Q a… …rue)
2. You join the dark side, **but** we **don't** rule the galaxy together. (P is True, Q is False)
3. You **don't** join the dark side, **but** we **still** rule the galaxy together. (P is False, Q is True)
4. You **don't** join the dark side, **and** we **don't** rule the galaxy together. (P is False, Q is True)

This one.

Truth Tables

Shorthand for
$P \Longrightarrow Q \wedge Q \Longrightarrow P$

Biconditional

P	Q	P ⇔ Q
True	True	True
True	False	False
False	True	False
False	False	True

I tell you: The Death Star can be destroyed **if and only if** your missile hits its vulnerable spot**.**

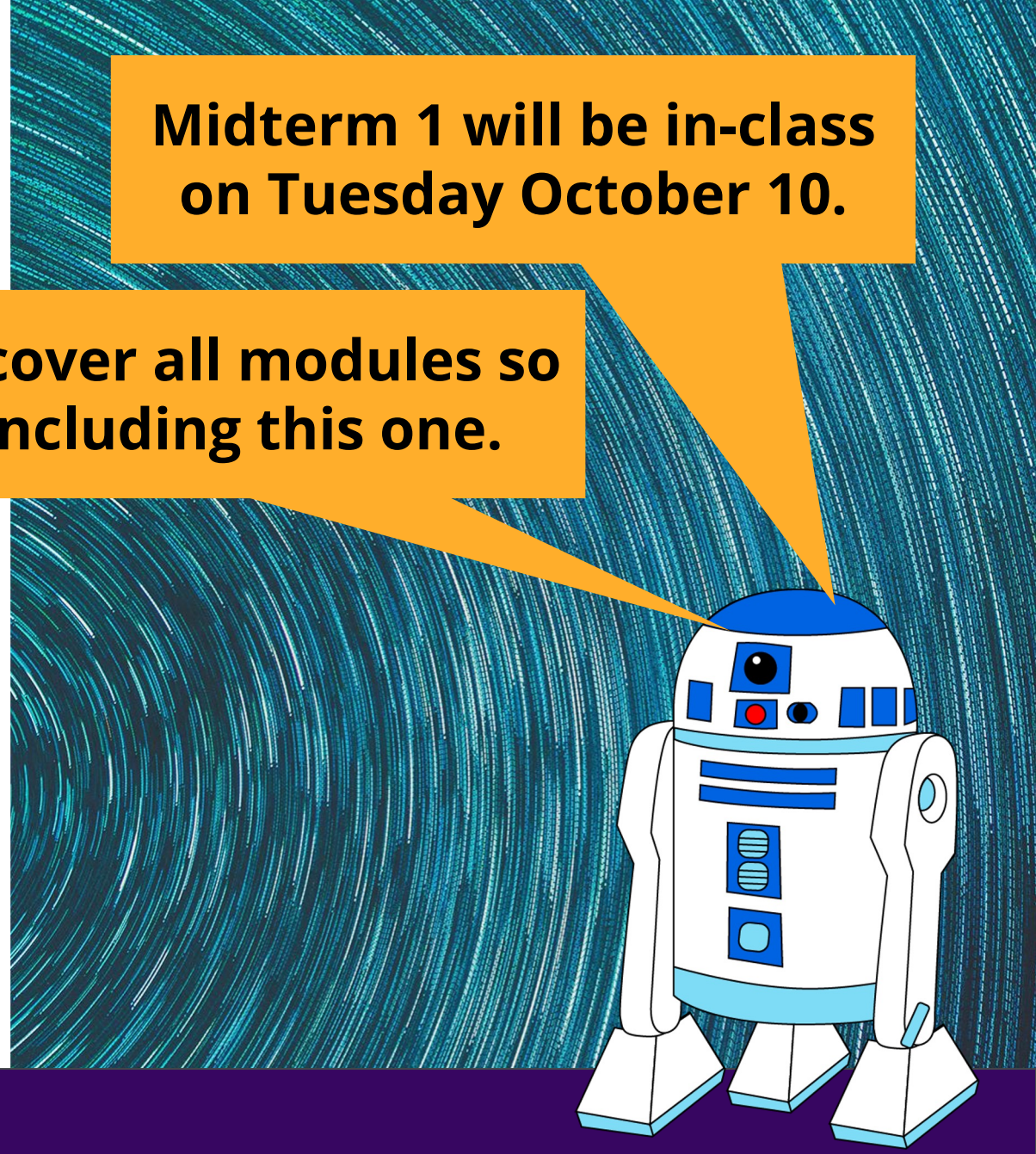
1. **The Death Star is destroyed,** and **you hit the vulnerable spot.** (Both P and Q are True)
2. The Death Star **is** destroyed, **but** you **didn't** hit its vulnerable spot. (P is True, Q is False)
3. The Death Star **isn't** destroyed, **but** you **did** hit its vulnerable spot. (P is False, Q is True)
4. The Death Star **isn't** destroyed, **and** you **also didn't** hit its vulnerable spot. (P is False, Q is False)

Propositional Logic

- Atomic sentences and logical connectives define the **syntax** of propositional logic and specifies which sentences are well-formed
- Truth tables define the **semantics** of the sentences, which is their **meaning**
- The semantics defines the **truth** of each sentence with respect to a **possible world,** which we will often call a **model**.
- Our agent can maintain a **knowledge base** which contains a set sentences whose truth value is known.

CIS 4210/5210:
ARTIFICIAL INTELLIGENCE

Logical Agents part 2

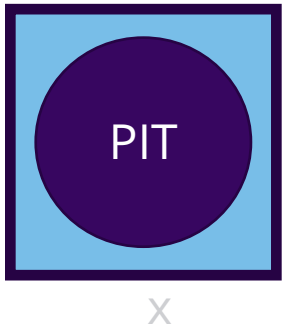


Knowledge Bases

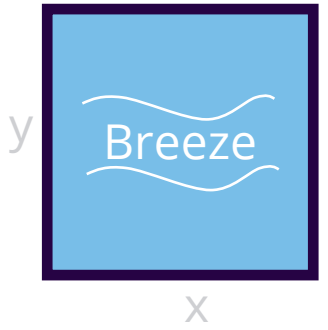
A Simple Knowledge Base

We can construct a knowledge base for Wampa World.
Let's start with a set of symbols for each location [x,y]:

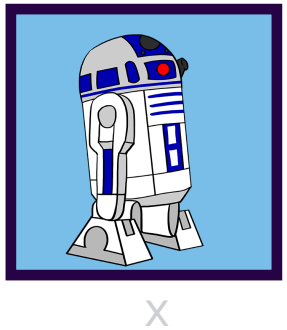
$P_{x,y}$ is true if there is a pit in [x,y]



$B_{x,y}$ is true if there is a breeze in [x,y]

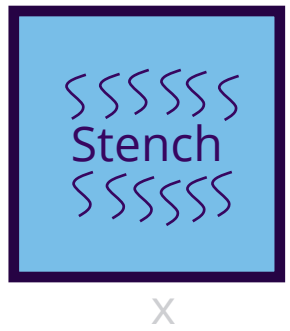


$L_{x,y}$ is true if the agent is in [x,y]



$W_{x,y}$ is true if there is a Wampa in [x,y]

$S_{x,y}$ is true if there is a stench in [x,y]



A Simple Knowledge Base

We can construct sentences out of these using our logical connectors. We'll label each sentence.

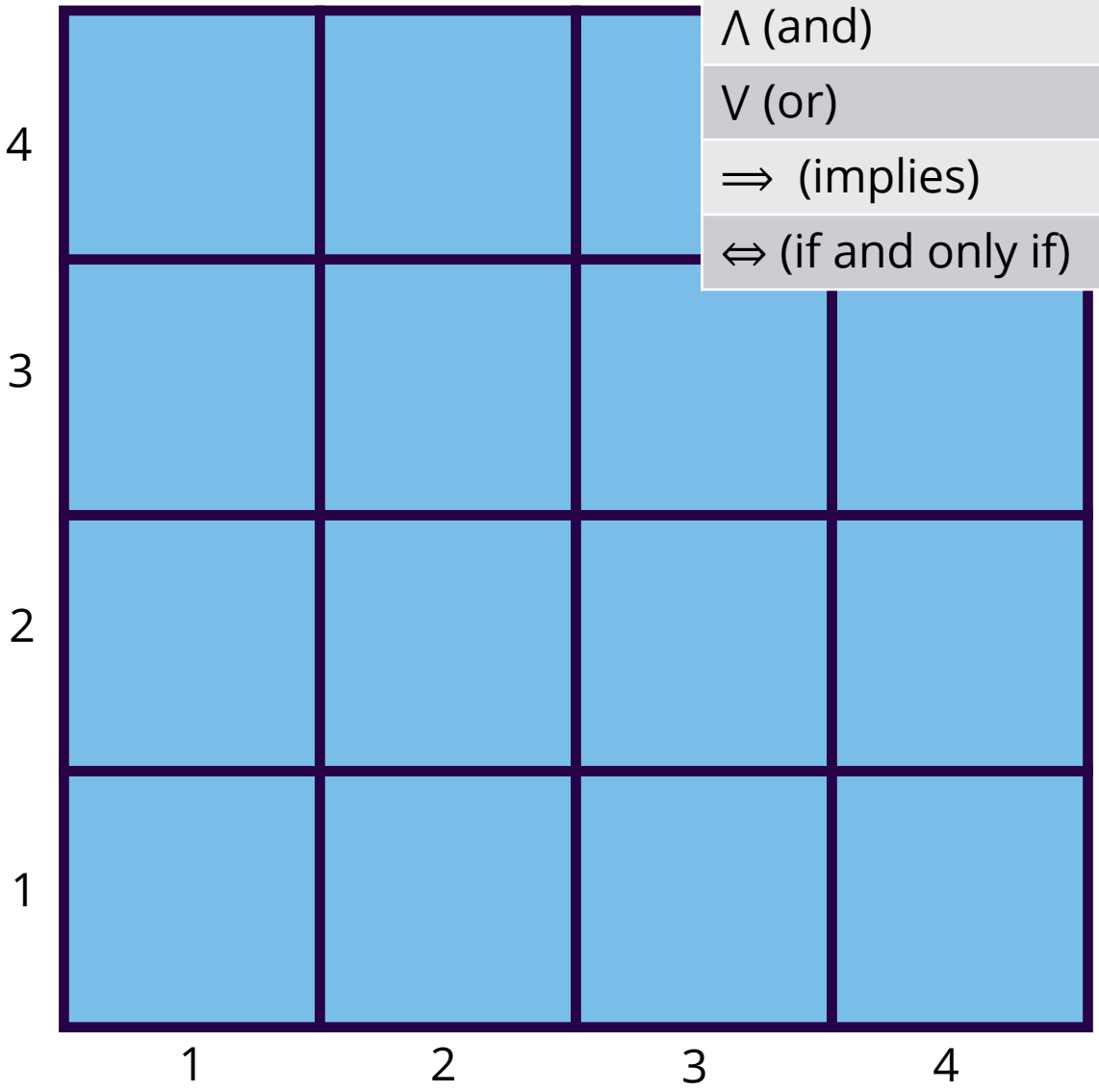
R1: $\neg P_{1,1}$

R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

These are true of all Wampa Worlds.

Logical Connectives
¬ (not)
Λ (and)
V (or)
⟹ (implies)
⇔ (if and only if)



A Simple Knowledge Base

We can construct sentences out of these using our logical connectors. We'll label each sentence.

R1: $\neg P_{1,1}$

R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

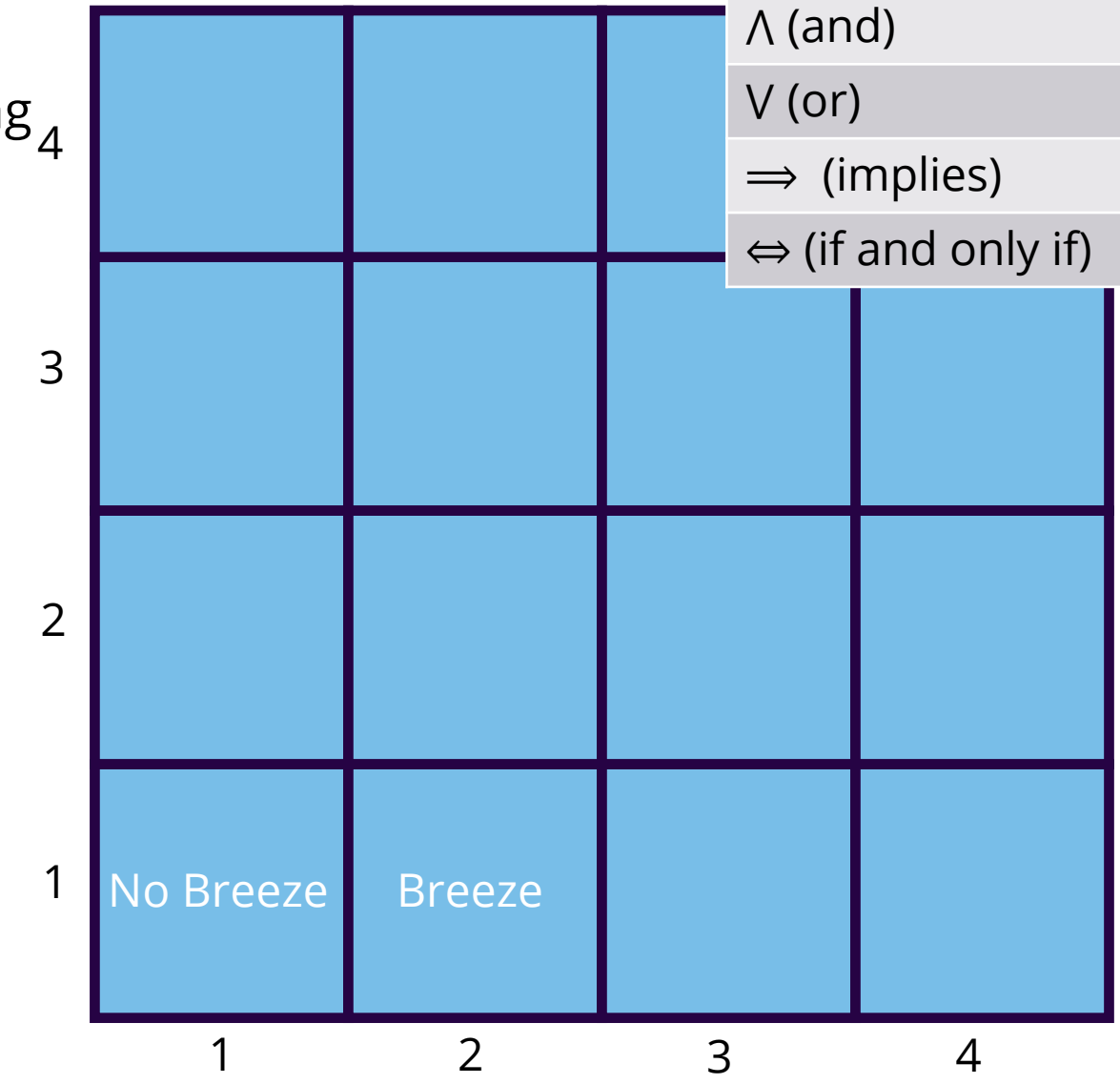
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

What if we perceive the presence or absence of breeze in [1,1], [2,1]?

R4: $\neg B_{1,1}$

R5: $B_{2,1}$

Logical Connectives
¬ (not)
Λ (and)
V (or)
⟹ (implies)
⇔ (if and only if)



A Simple Knowledge Base

We can construct sentences out of these using our logical connectors. We'll label each sentence.

R1: $\neg P_{1,1}$

R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

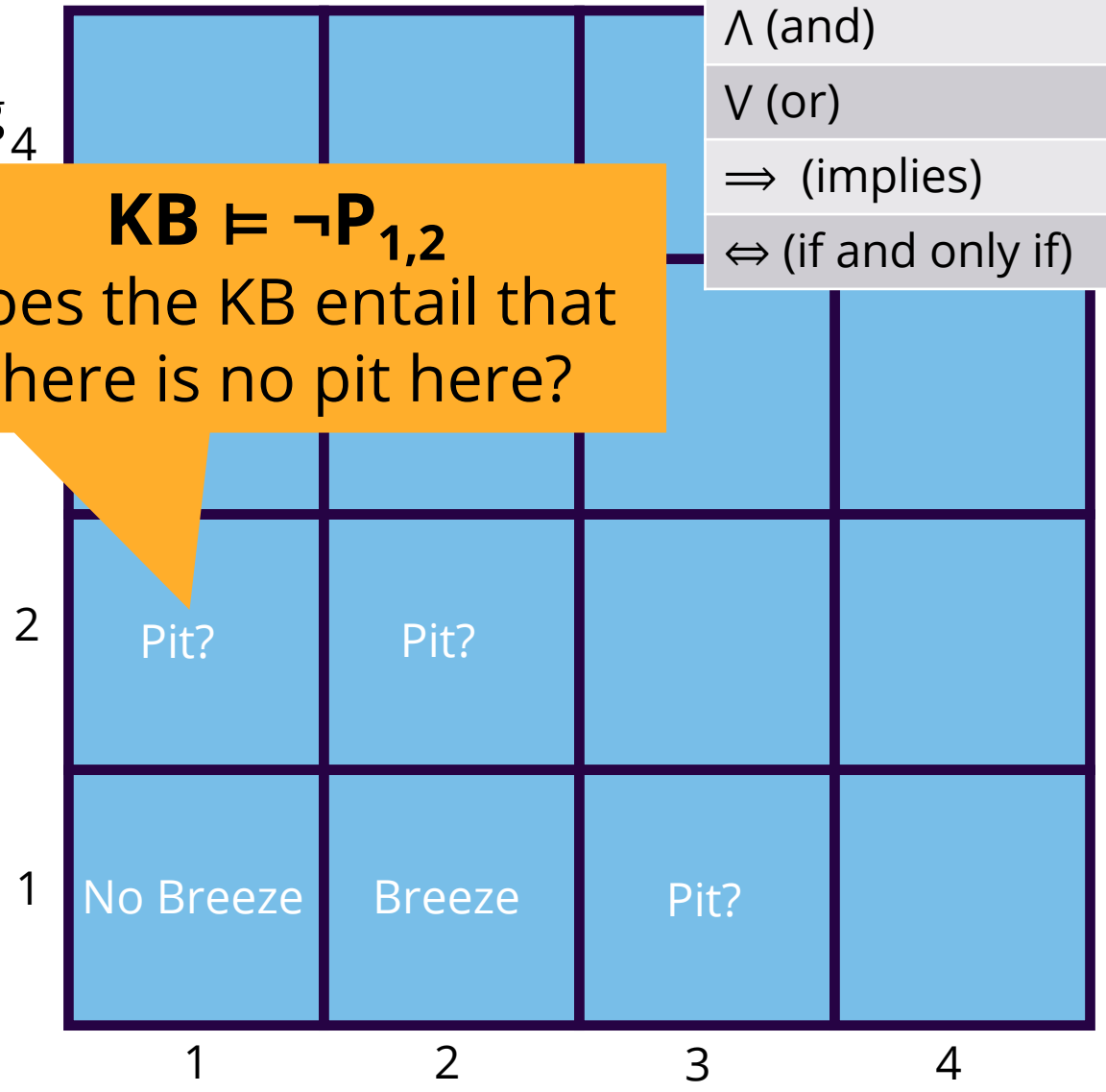
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R4: $\neg B_{1,1}$

R5: $B_{2,1}$

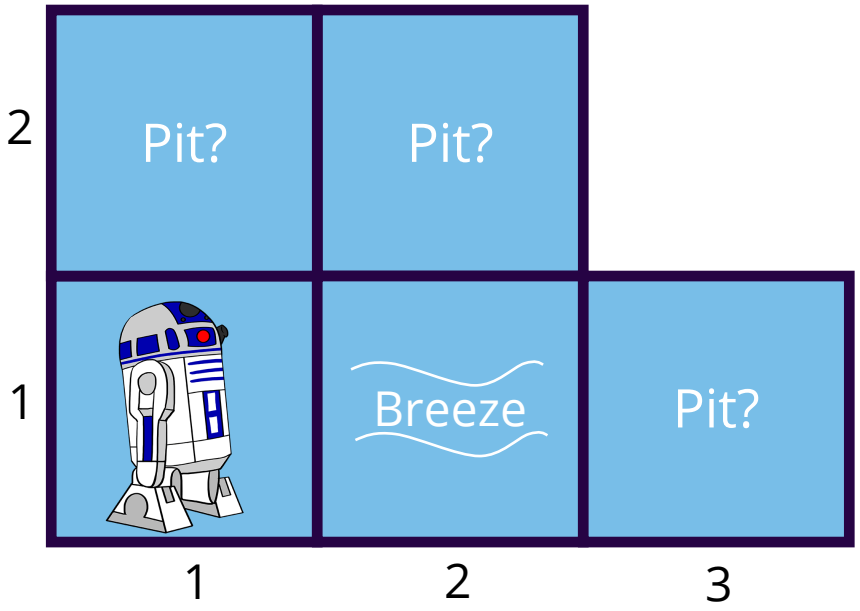
Can mechanically combine the sentences in our KB to prove that a pit exists at (or is absent from) any location?

Logical Connectives
¬ (not)
Λ (and)
V (or)
⟹ (implies)
⇔ (if and only if)



$KB \models \neg P_{1,2}$
Does the KB entail that there is no pit here?

Possible Worlds



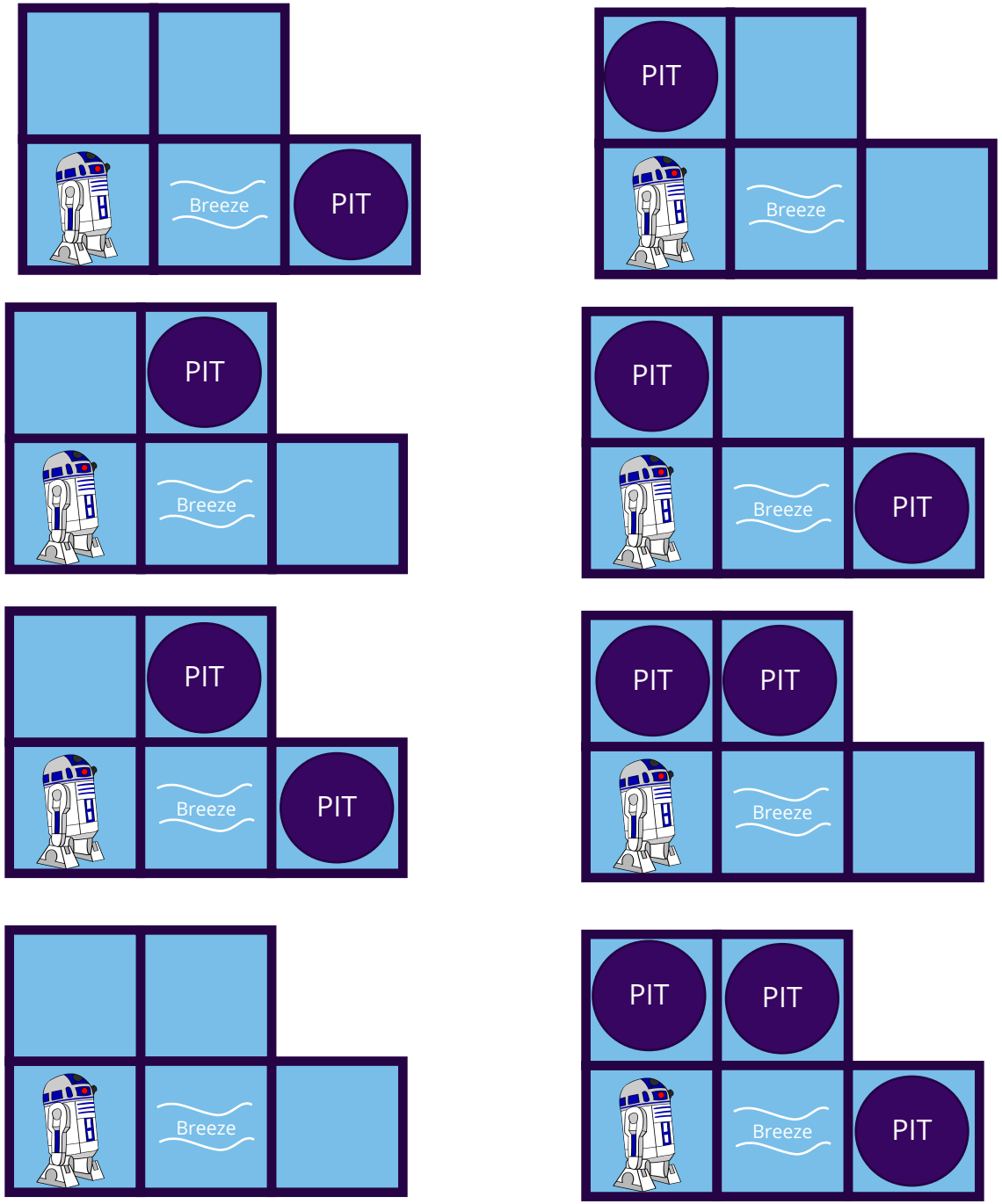
Models are mathematical abstractions that have a fixed set of **truth values** which are **{true, false}** for each sentence.

If sentence **α** is true in model **m** then we say

m satisfies **α**, or

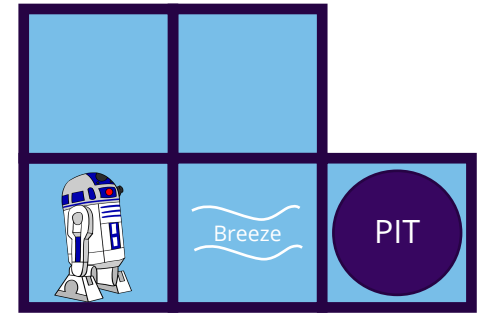
m is a model of **α**

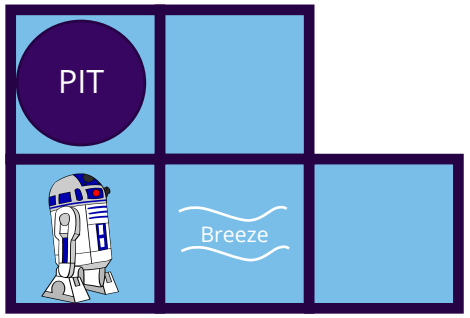
We use the notation **M(α)** to mean the **set of all models** of **α**.

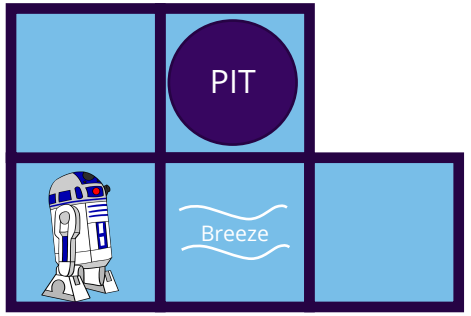


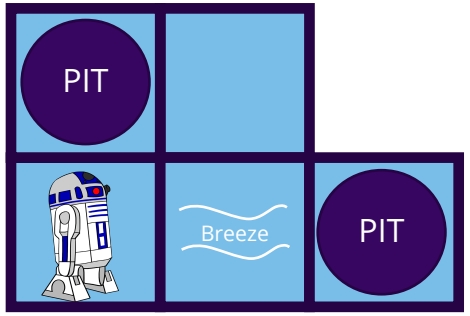
Possible Worlds

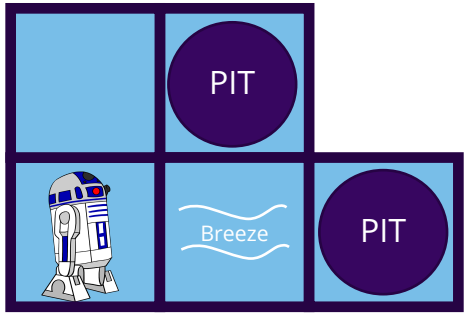
α_1 = “There is no pit in [1,2]”

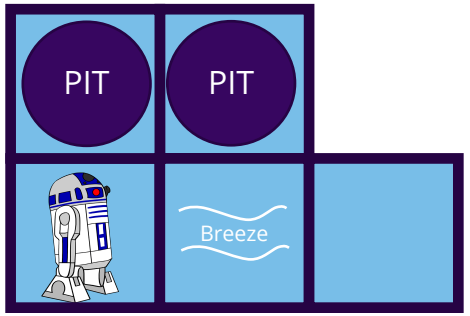


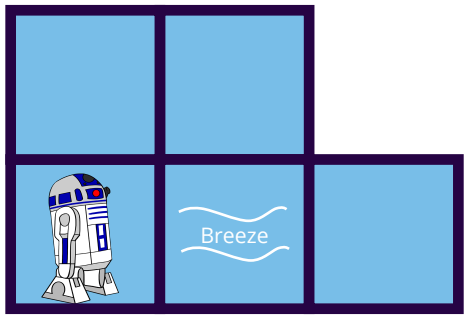


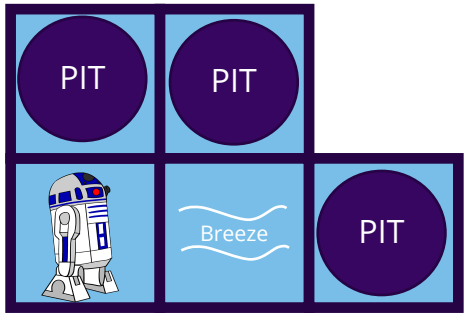






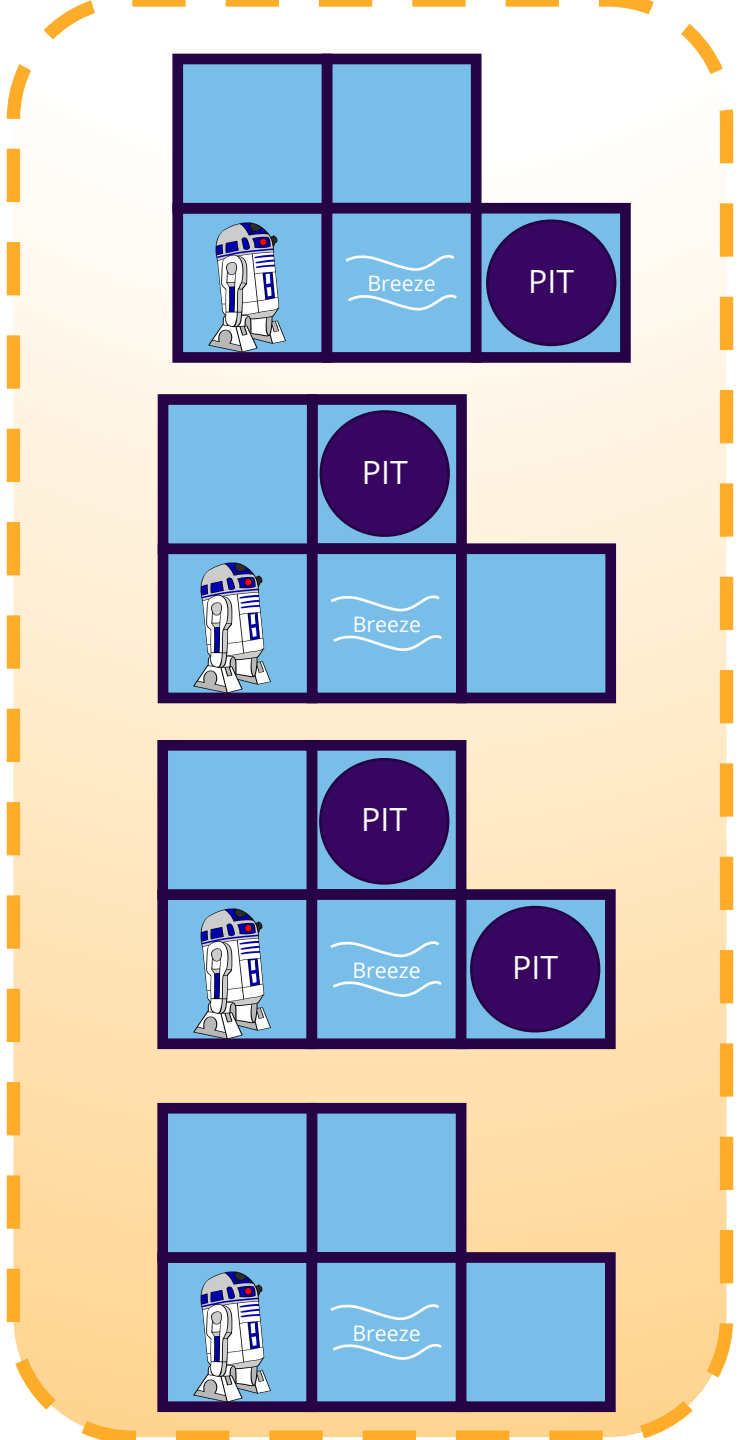


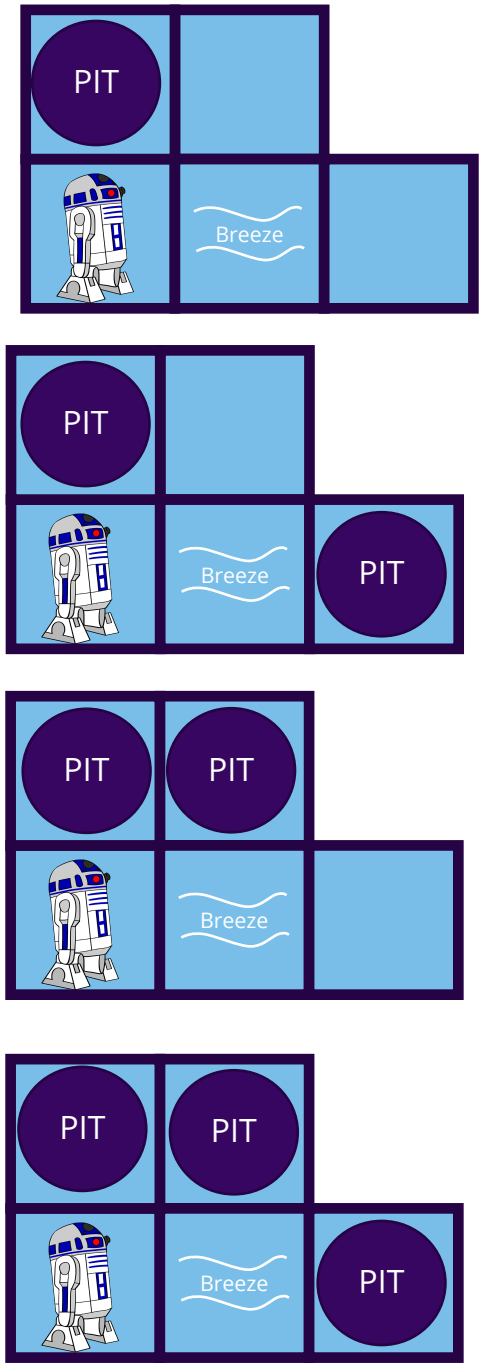




Possible Worlds

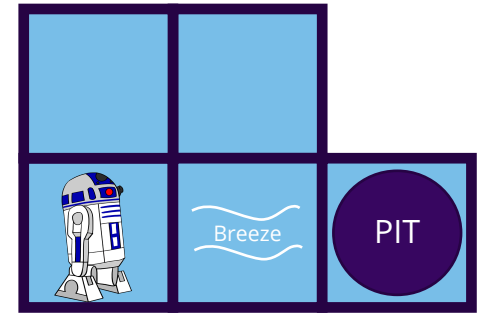
α_1 = “There is no pit in [1,2]”

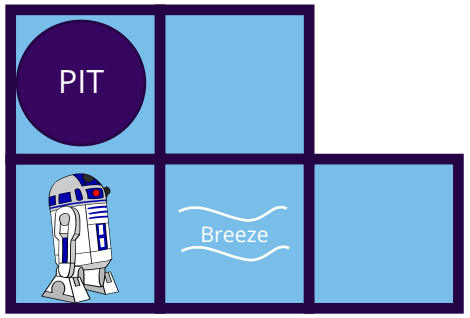


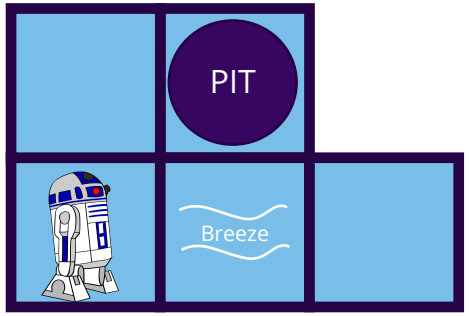


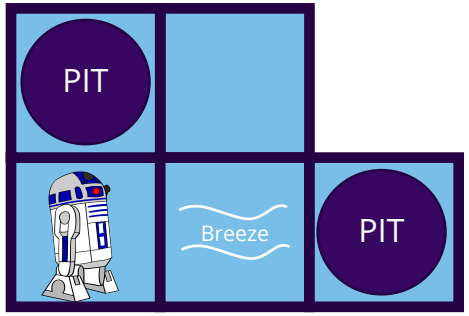
Possible Worlds

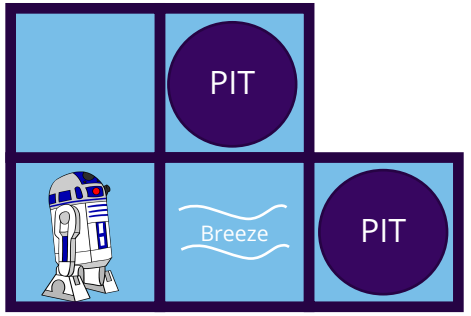
α_2 = “There is no pit in [2,2]”

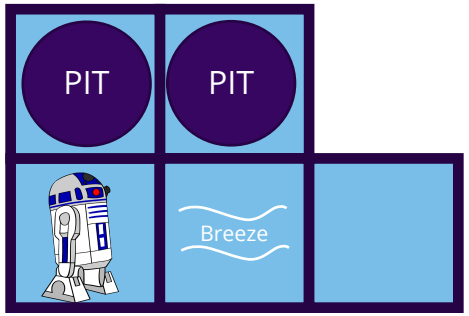


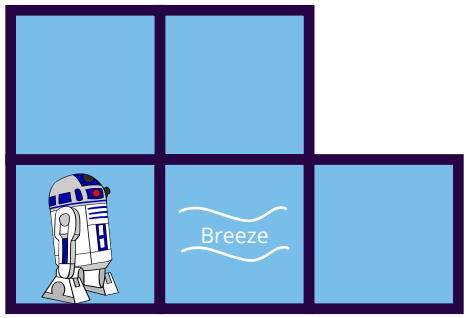


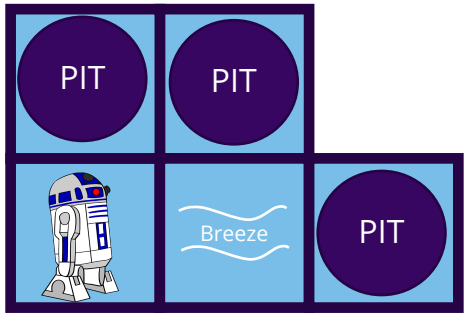






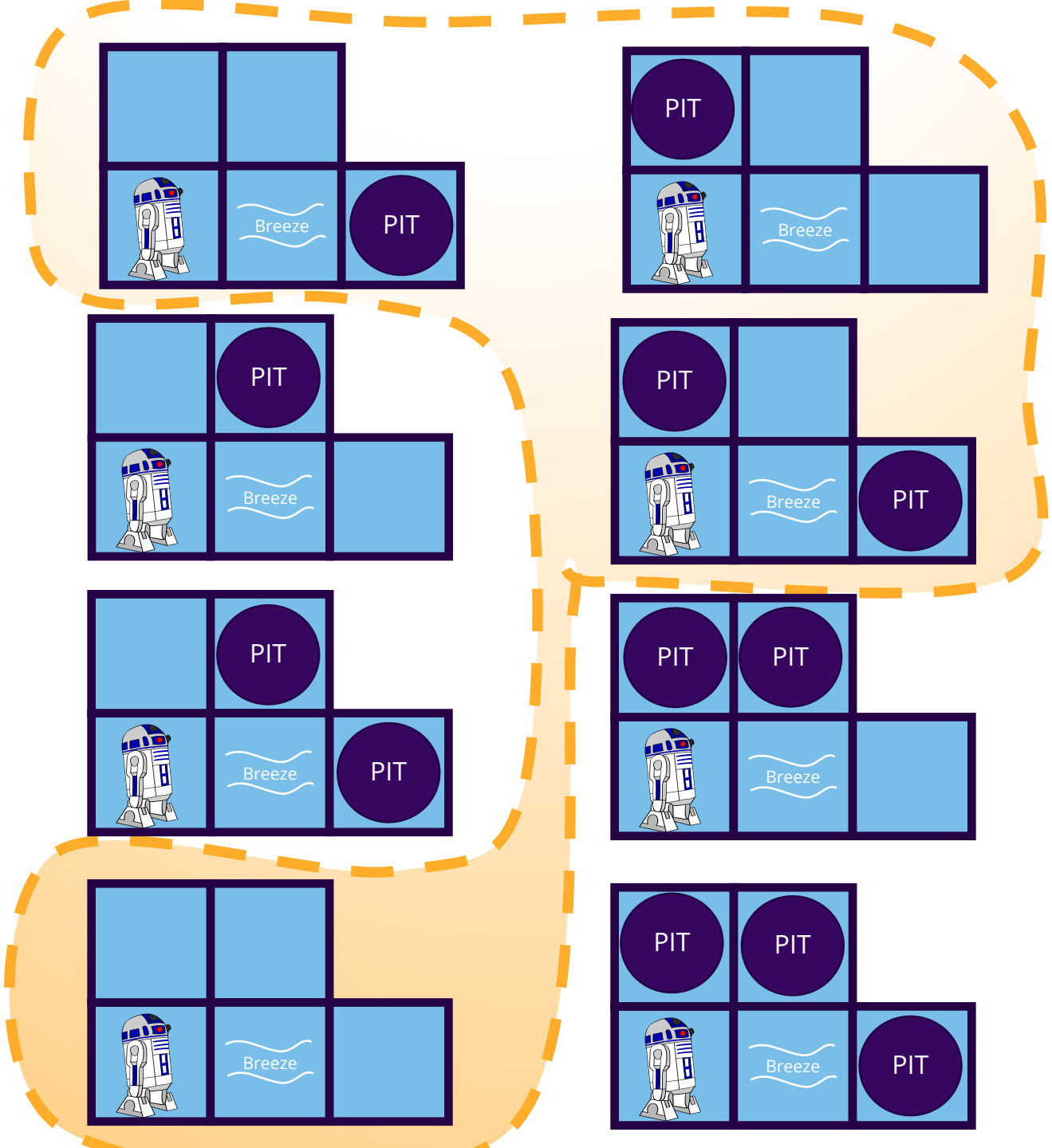






Possible Worlds

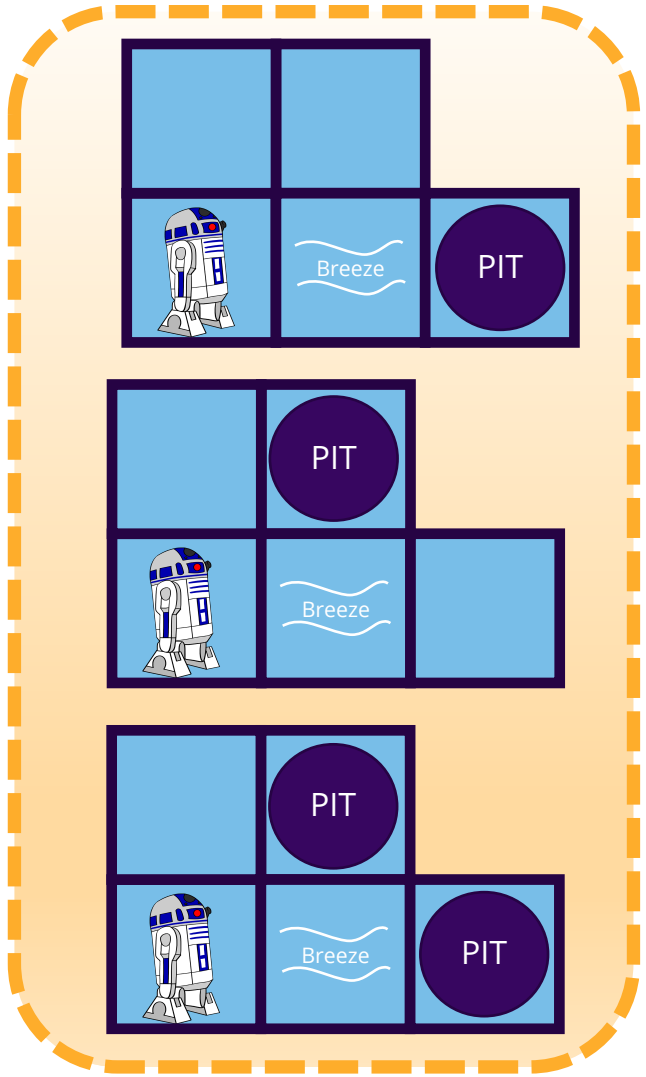
α_2 = “There is no pit in [2,2]”

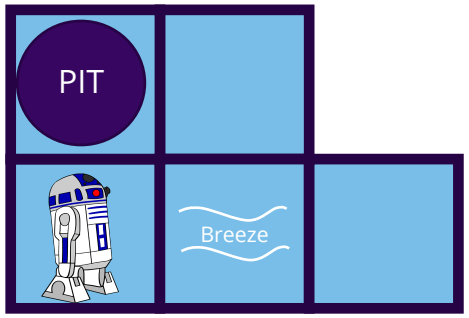


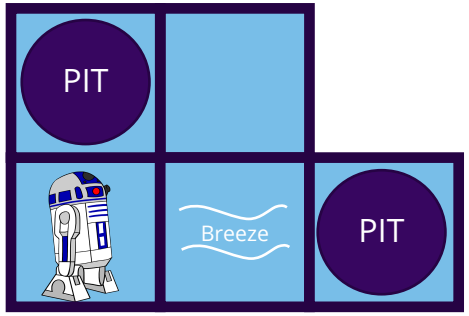
Possible Worlds

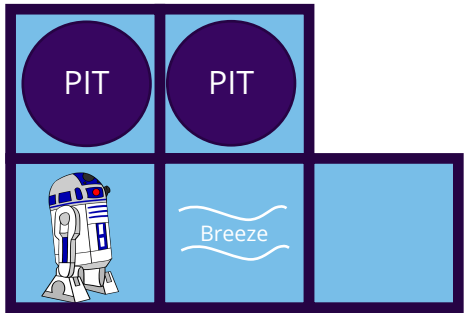
KB =

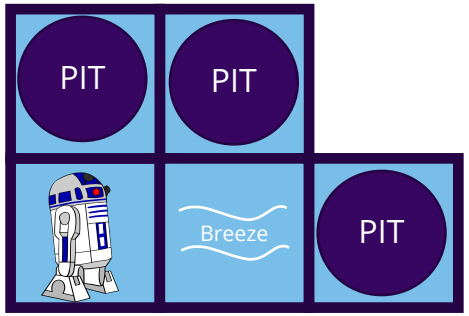
R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$









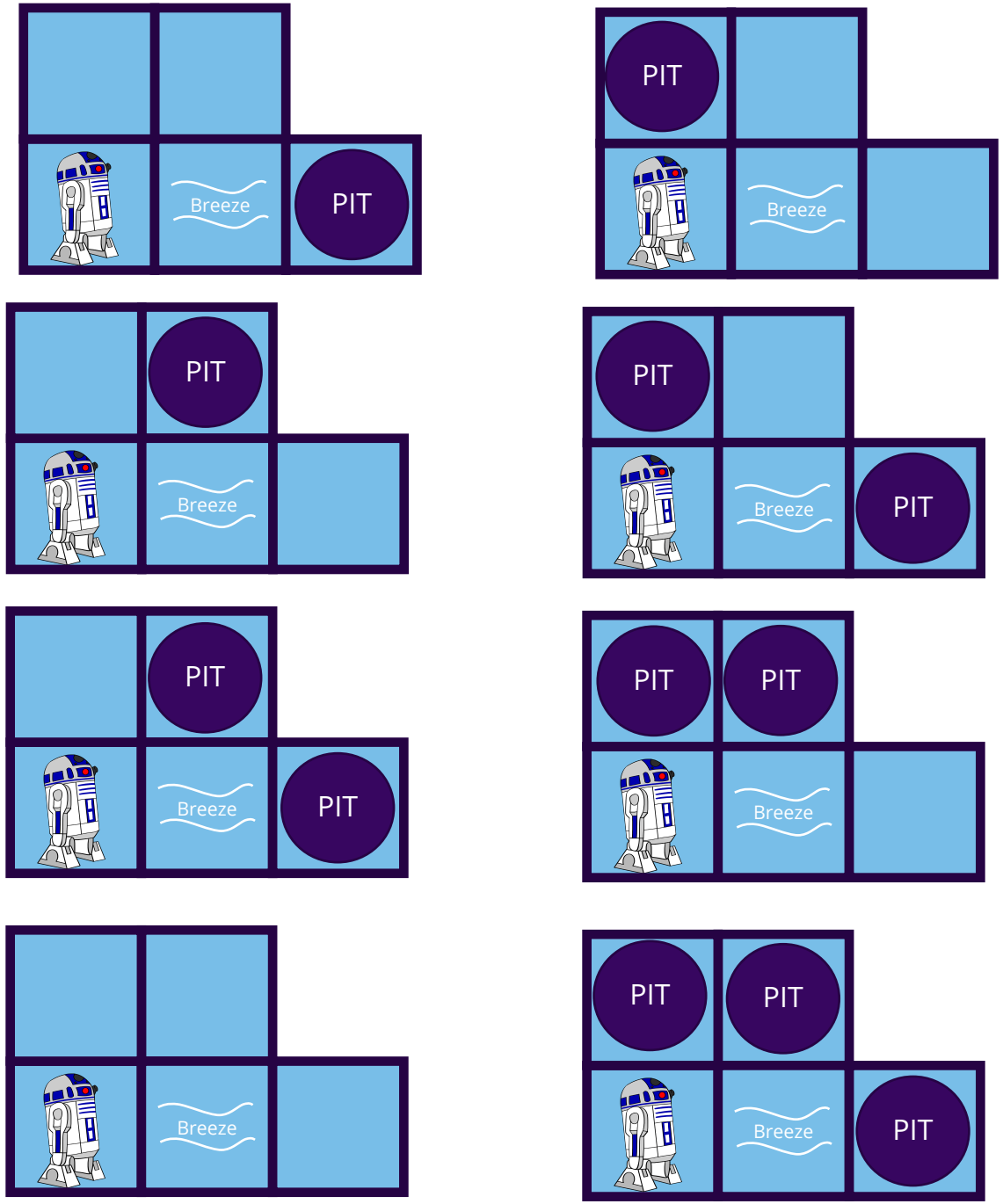


Possible Worlds

$\beta \models \alpha$ if and only if $M(\beta) \subseteq M(\alpha)$

"**β entails α** if and only if every model in which β is true, **α** is also true"

Does our KB entail that there is no pit in [1,2]?
$KB \models \alpha_1$ if and only if $M(KB) \subseteq M(\alpha_1)$



Possible Worlds

β ⊨ α if and only if **M(β) ⊆ M(α)**

"β entails **α** if and only if every model in which β is true, **α** is also true"

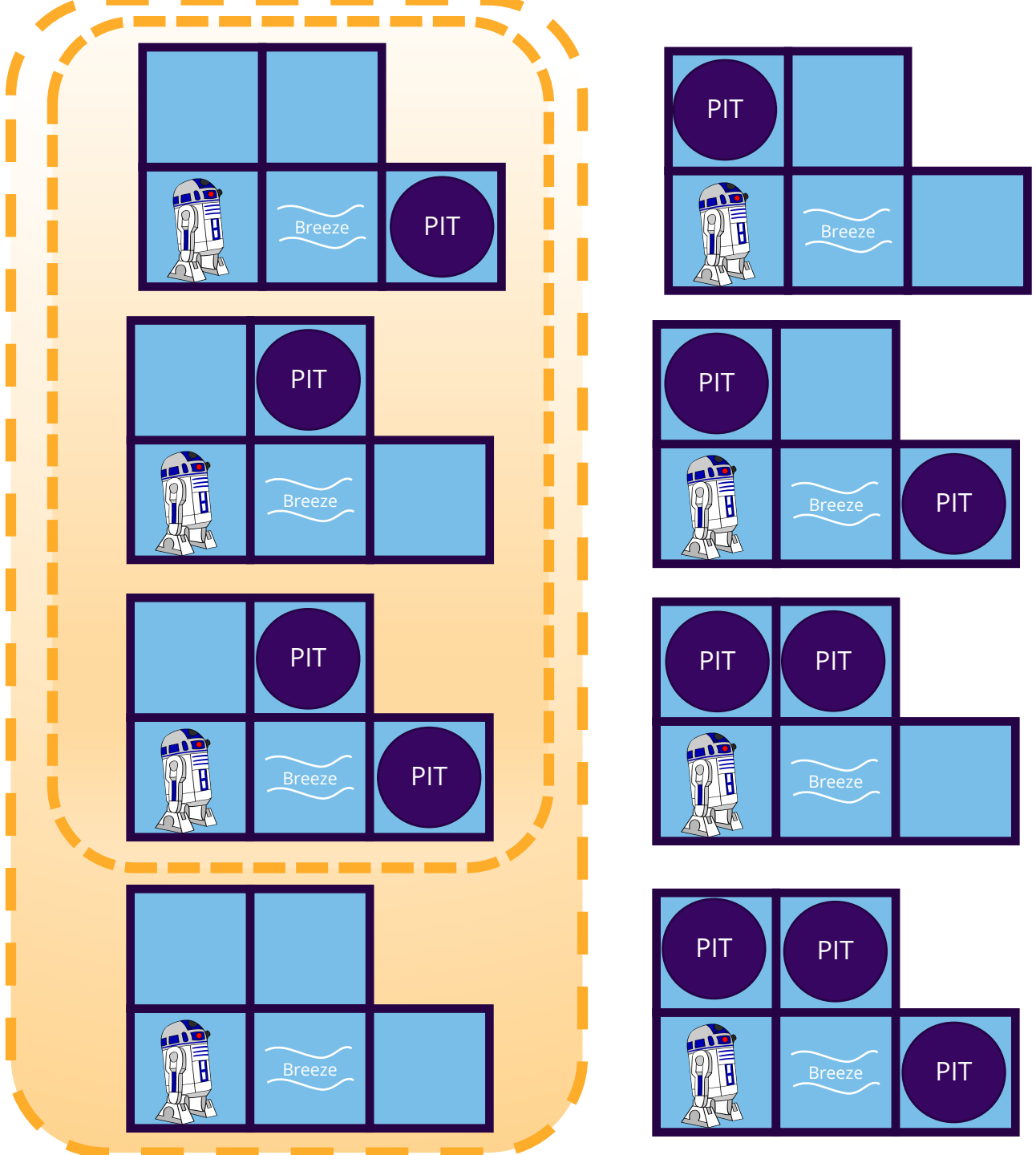
Does our KB entail that there is no pit in [1,2]?
KB⊨α_1 if and only if **M(KB) ⊆ M(α_1)**

KB =

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

α_1 = "There is no pit in [1,2]"

KB⊨α_1



Possible Worlds

$\beta \models \alpha$ if and only if $M(\beta) \subseteq M(\alpha)$

"β entails α if and only if every model in which β is true, α is also true"

Does our KB entail that there is **no pit in [2,2]**?

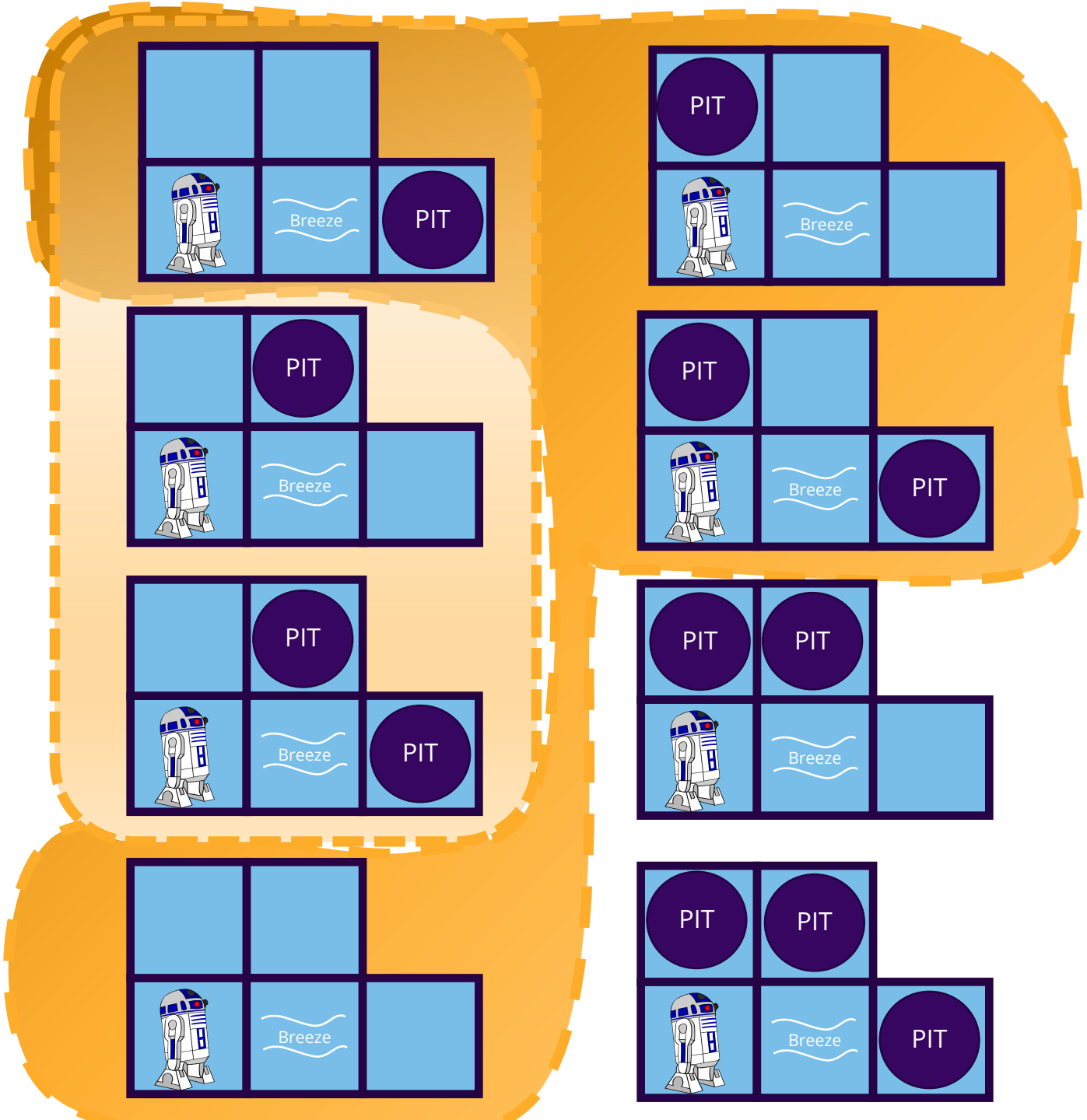
$KB \models \alpha_2$ if and only if $M(KB) \subseteq M(\alpha_2)$

KB =

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

α_2 = "There is no pit in [2,2]"

KB does not entail α_2 in some models where KB is true α_2 is false



Entailment and Inference

Entailment can be applied to derive conclusions, which is the process of **logical inference.**

We can think about the consequences of a KB as a large set of additional sentences that are entailed given the sentences that have been added to the KB.

We would like to design **inference algorithms** to enumerate these sentences.

When an inference algorithm **i** allows us to conclude that **α** is true, then we write

$$\mathbf{KB} \vdash_{\mathbf{i}} \alpha$$

"**α** is derived from **KB** by **i**"

Soundness and Completeness

- **Sound** – the inference algorithm should **only** derive entailed sentences.
- **Complete** – an inference algorithm is complete if it can derive **all** sentences that are entailed

Theorem Proving

Review: Propositional Logic

Complex sentences are constructed from simpler ones using parentheses and **logical connectives**.

Logical Connective	Meaning
¬ (not)	$\neg \mathbf{W_{1,3}}$ is the negation of $\mathbf{W_{1,3}}$
∧ (and)	$\mathbf{W_{1,3}} \wedge \mathbf{P_{3,1}}$ is called a conjunction
∨ (or)	$\mathbf{W_{1,3}} \vee \mathbf{P_{3,1}}$ is called a disjunction
⟹ (implies)	$\mathbf{W_{1,3}} \Longrightarrow \mathbf{S_{1,2}}$ is called an implication. $\mathbf{W_{1,3}}$ is its **premise or antecede** and $\mathbf{S_{1,2}}$ is its **conclusion or consequence**
⇔ (if and only if)	$\mathbf{W_{1,3}} \Leftrightarrow \neg \mathbf{W_{3,4}}$ is called an biconditional

Logical Equivalence

Two sentences **α** and β are **logically equivalent** if they are true in the same set of models. We write this as

$$\alpha \equiv \beta$$

An alternate definition of logical equivalence is that two sentences **α** and β are logically equivalent if any only if they entail each other.

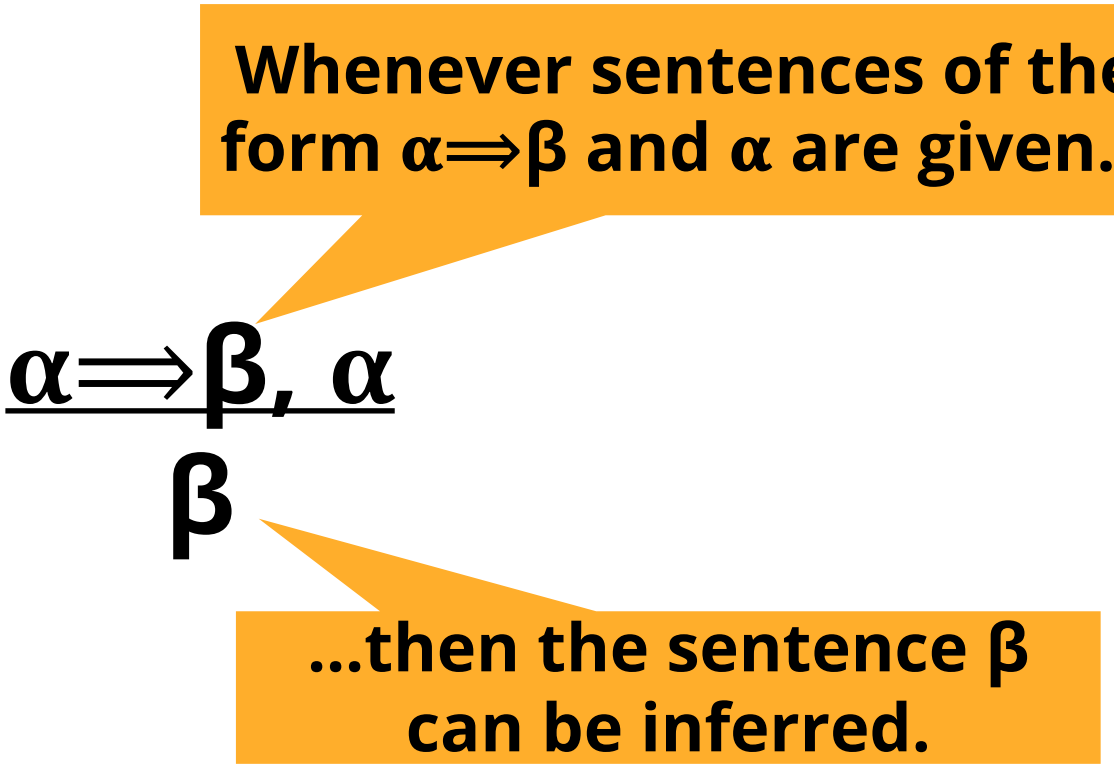
$$\alpha \equiv \beta \quad \text{if and only if} \quad \alpha \models \beta \text{ and } \beta \models \alpha.$$

Logical Equivalence

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	Commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	Commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	Associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	Associativity of $\vee$
$\neg (\neg \alpha)$	$\equiv$	α	Double-negation elimination
$(\alpha \Longrightarrow \beta)$	$\equiv$	$(\neg\beta \Longrightarrow \neg\alpha)$	Contraposition
$(\alpha \Longrightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	Implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Longrightarrow \beta) \wedge (\beta \Longrightarrow\alpha))$	Biconditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)))$	Distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma)))$	Distributivity of $\vee$ over $\wedge$

Inference Rules

Inference Rules can be used to derive proofs. Here's the notation:



Inference Rules

Modus Ponens:

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

Biconditional Elimination:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

And Elimination:

$$\frac{\alpha \wedge \beta}{\alpha}$$

Biconditional Elimination:

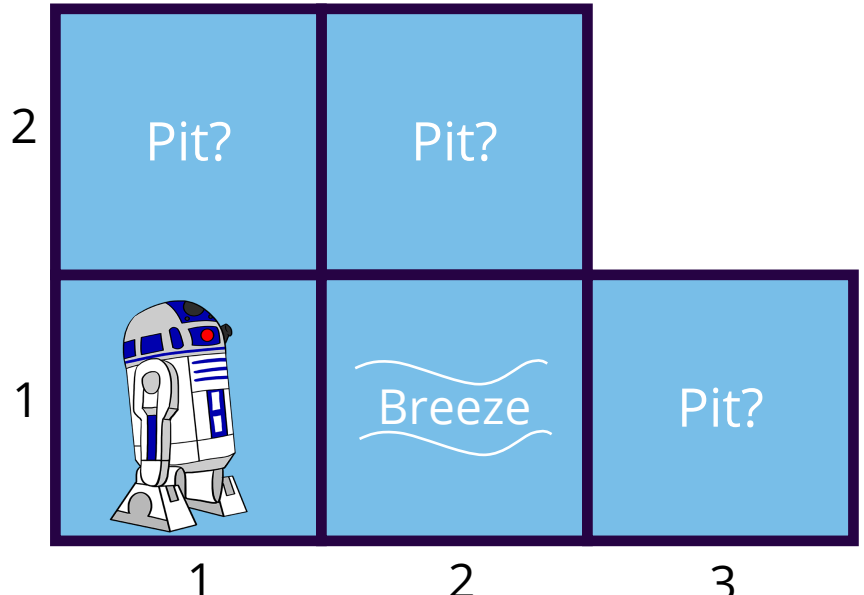
$$\frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

All of the logical equivalence rules can be re-written as inference rules.

Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

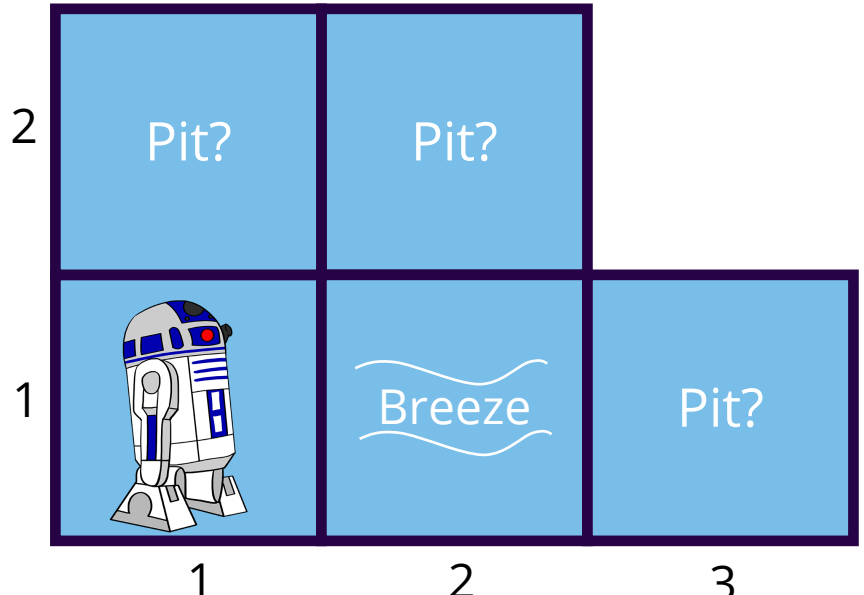
Biconditional Elimination:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

Apply biconditional Elimination to R2 to get R6.

R6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Monotonicity: if **KB**$\models\alpha$ then **KB** $\wedge \beta \models\alpha$
We can safely add to the KB, without invalidating anything else that we inferred.



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

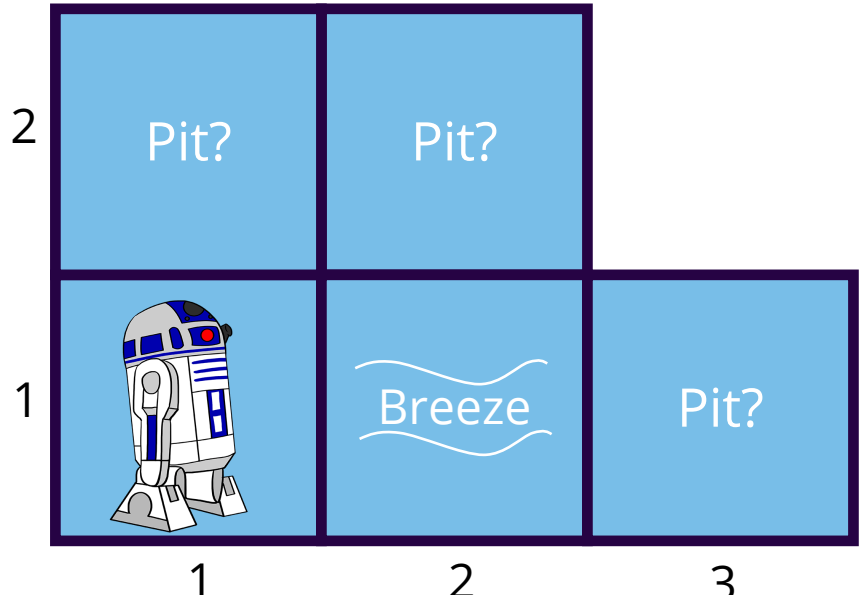
R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

Apply And-Elimination to R6 to get R7.

R7: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1}))$

And Elimination:

$$\frac{\alpha \wedge \beta}{\alpha}$$



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

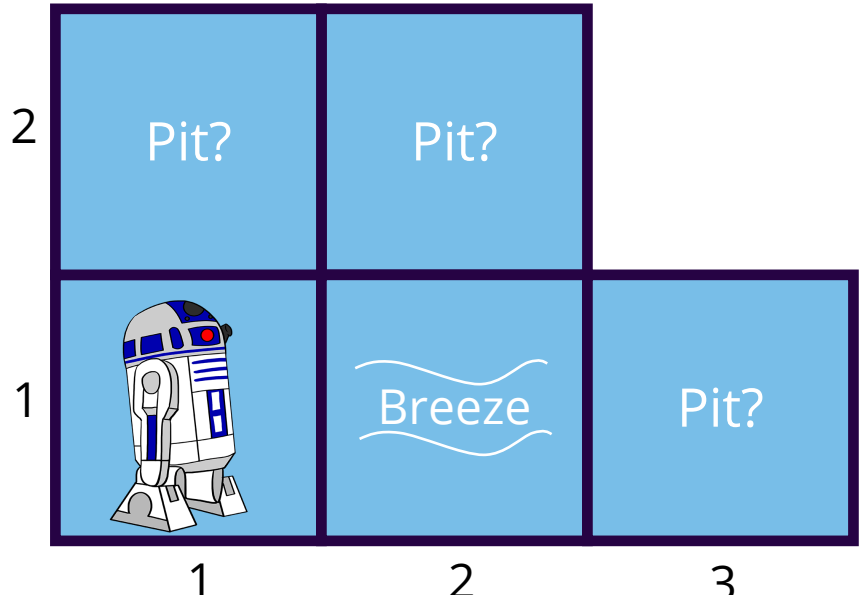
R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

Logical equivalence for contrapositives applied to R7 gives R8.

R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$

Logical equivalence for contrapositives:

$$\frac{(\alpha \Longrightarrow \beta)}{(\neg \beta \Longrightarrow \neg \alpha)}$$



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

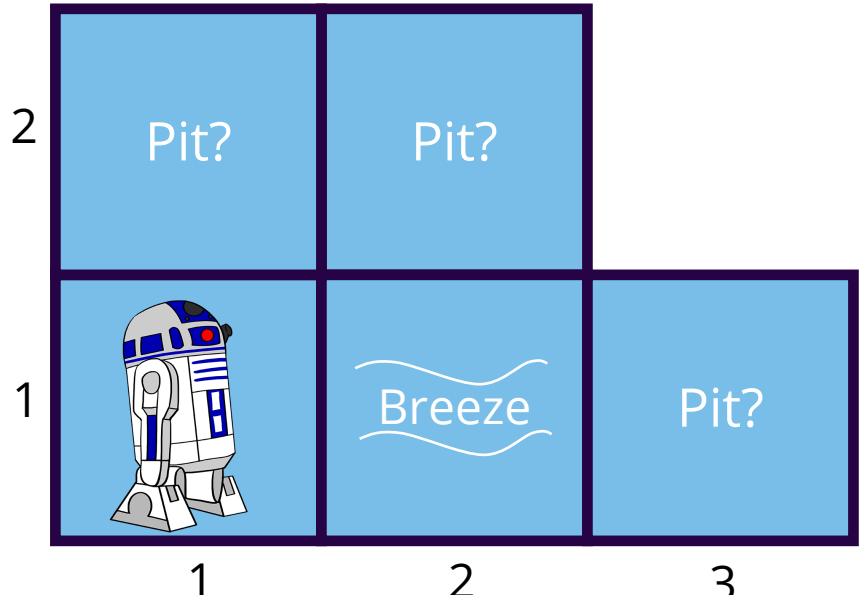
R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$

Apply Modus Ponens to R4 and R8 to get:

R9: $\neg (P_{1,2} \vee P_{2,1})$

Modus Ponens:

$$\frac{\alpha \Longrightarrow \beta,\ \alpha}{\beta}$$



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$

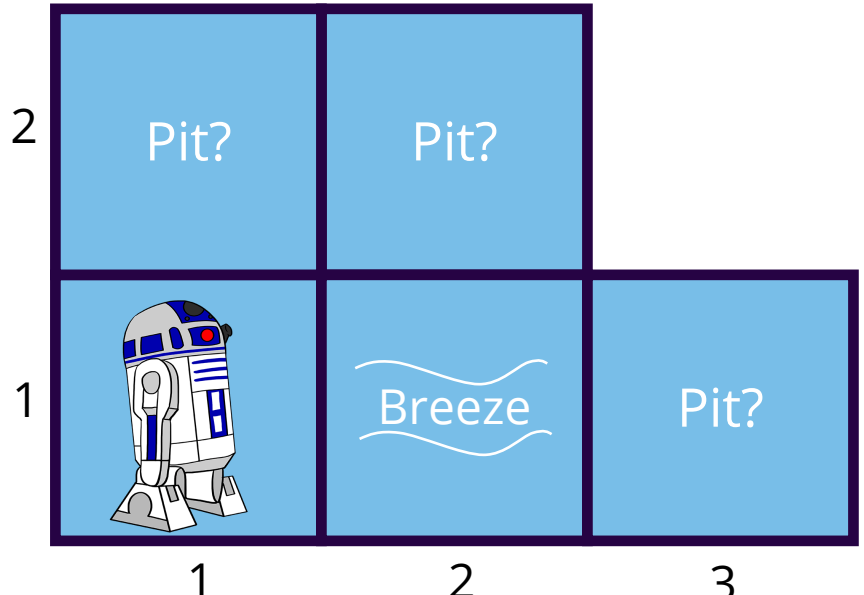
R9: $\neg (P_{1,2} \vee P_{2,1}))$

Apply De Morgan's Rule to R9:

R10: $\neg P_{1,2} \wedge \neg P_{2,1}$

De Morgan's Rule

$$\frac{\neg(\alpha \vee \beta)}{(\neg\alpha \wedge \neg\beta)}$$



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$

R9: $\neg (P_{1,2} \vee P_{2,1}))$

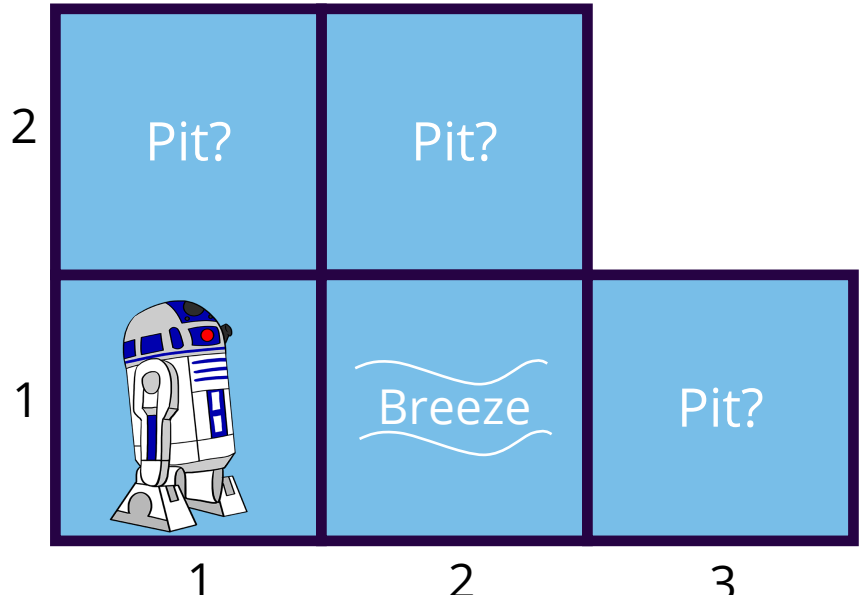
R10: $\neg P_{1,2} \wedge \neg P_{2,1}$

R11: $\neg P_{1,2}$

R12: $\neg P_{2,1}$

And Elimination:

$$\frac{\alpha \wedge \beta}{\alpha}$$



Inference Example

Our KB

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

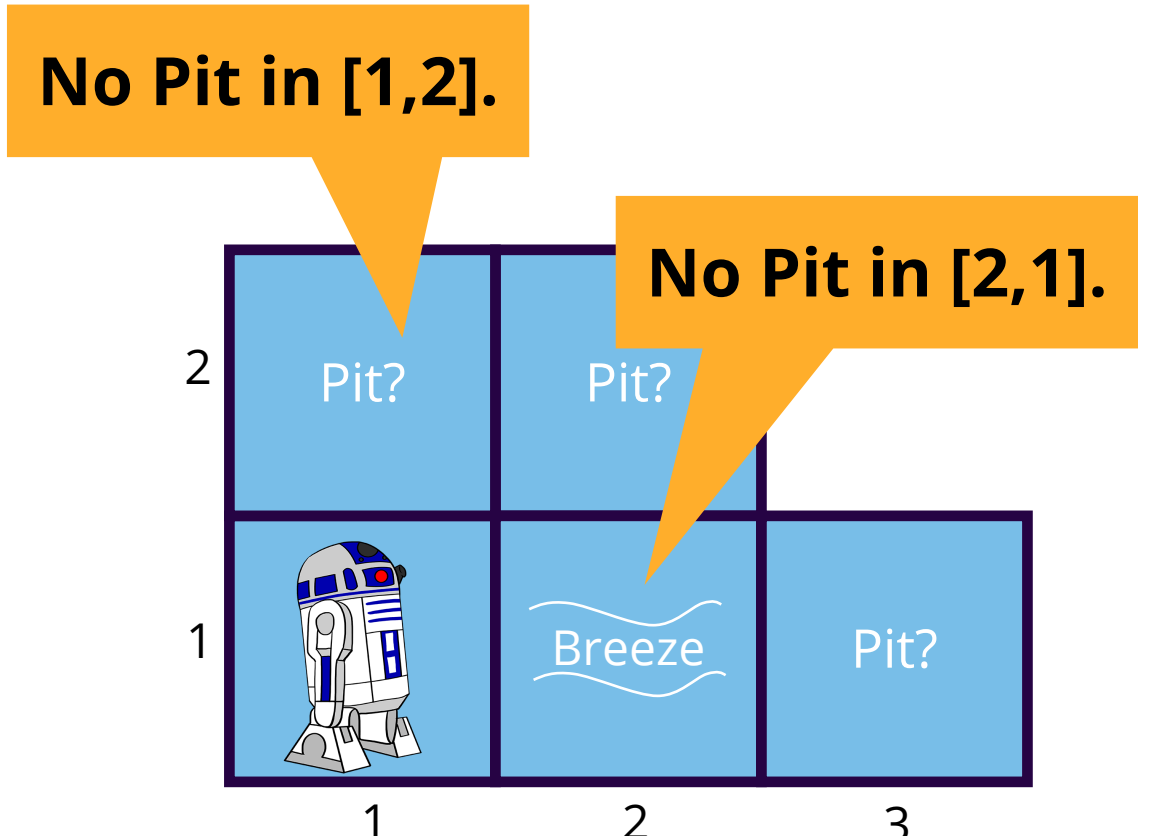
R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$

R9: $\neg (P_{1,2} \vee P_{2,1}))$

R10: $\neg P_{1,2} \wedge \neg P_{2,1}$

R11: $\neg P_{1,2}$

R12: $\neg P_{2,1}$



Search for a Proof

We can use search algorithms to find a sequence of steps that constitutes a proof!

- **Initial State:** the initial knowledge base.
- **Actions:** All inference rules applied to all sentences that match the top half of the inference rule.
- **Result:** Add the sentence on the bottom half to the KB.
- **Goal:** The goal is a state that contains the sentence that we are trying to prove.

Initial State

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$

Action

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$$

Result

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R6: $B_{2,1}$
R7: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$

Goal

$\neg P_{1,2}$

Monotonicity implies Soundness

Monotonicity says that the set of entailed sentence can only increase as information is assed to the KB.

if $\mathbf{KB} \models \alpha$ then $\mathbf{KB} \wedge \beta \models \alpha$

If we add additional information to the KB, it doesn't invalidate any other info that we've already inferred.

Therefore it is **sound** it will only derive entailed sentences.

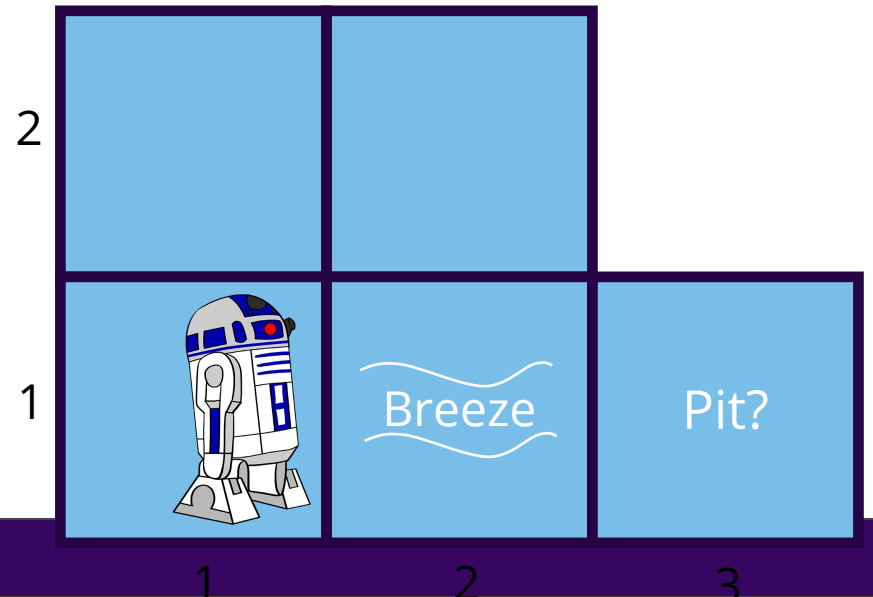
Proof by Resolution

But is it **complete?** Can this method derive **all** sentences that are entailed? If we used iterative deepening search, then we would find any reachable goal.

But what if we were missing an important inference rule? E.g. what if we forgot to include biconditional elimination?

There is a single inference rule, **resolution**, which yields a complete inference algorithm when coupled with any complete search algorithm.

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$
R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$
R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$
R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$
R9: $\neg (P_{1,2} \vee P_{2,1}))$
R10: $\neg P_{1,2} \wedge \neg P_{2,1}$
R11: $\neg P_{1,2}$
R12: $\neg P_{2,1}$



R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$
R6: $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$
R7: $((P_{1,2} \vee P_{2,1}) \Longrightarrow B_{1,1})$
R8: $(\neg B_{1,1} \Longrightarrow \neg (P_{1,2} \vee P_{2,1}))$
R9: $\neg (P_{1,2} \vee P_{2,1}))$
R10: $\neg P_{1,2} \wedge \neg P_{2,1}$
R11: $\neg P_{1,2}$
R12: $\neg P_{2,1}$

From Percept

R13: $\neg B_{1,2}$

From Wampa World Rules

R14: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

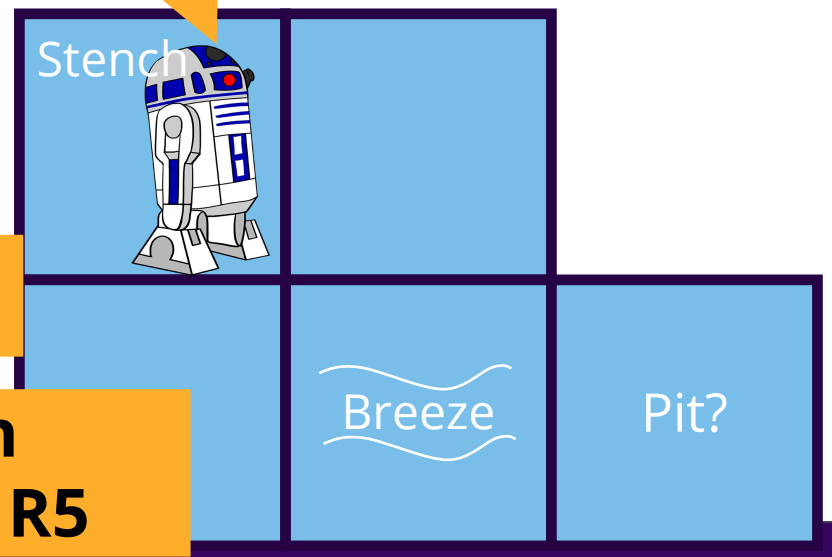
R15: $\neg P_{2,2}$
R16: $\neg P_{1,3}$

Using same process as before

R17: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

Biconditional Elimination + Modus Ponens on R3 and R5

Stench, but no Breeze



R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$
R6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
R7: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$
R9: $\neg (P_{1,2} \vee P_{2,1}))$
R10: $\neg P_{1,2} \wedge \neg P_{2,1}$
R11: $\neg P_{1,2}$
R12: $\neg P_{2,1}$

R13: $\neg B_{1,2}$
R14: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{\ldots})$
R15: $\neg P_{2,2}$
R16: $\neg P_{1,3}$
R17: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

R18: $(P_{1,1} \vee P_{3,1})$

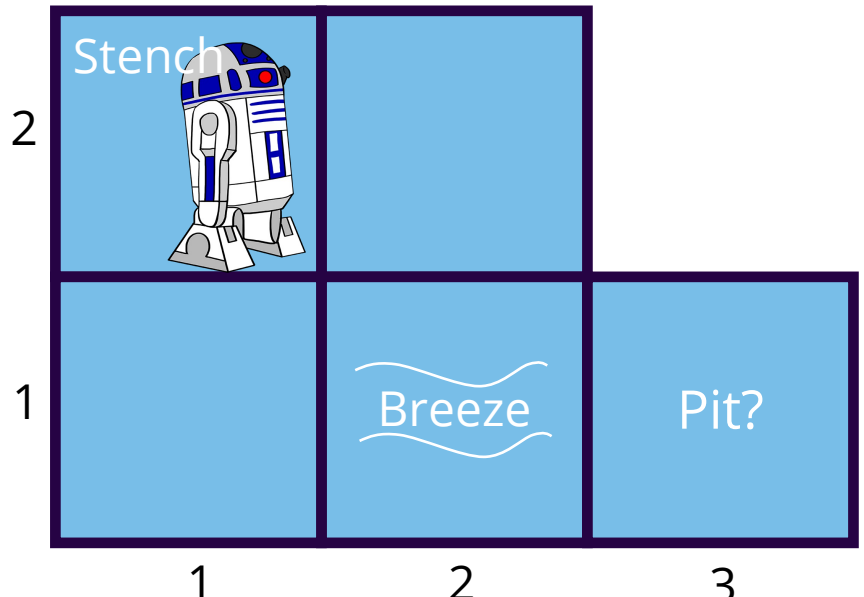
First application of **Resolution** using R17 and R15**:**

$$\frac{(P_{1,1} \vee \mathbf{P_{2,2}} \vee P_{3,1}) \wedge \mathbf{\neg P_{2,2}}}{(P_{1,1} \vee P_{3,1})}$$

And there's no pit in [2,2]

There must be a pit in either [1,1] or [2,2] or [3,1]

Resolution: There must be a pit in either [1,1] or [3,1]



And there's no pit in [1,1]

R1: $\neg P_{1,1}$
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R4: $\neg B_{1,1}$
R5: $B_{2,1}$
R6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
R7: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$
R9: $\neg (P_{1,2} \vee P_{2,1}))$
R10: $\neg P_{1,2} \wedge \neg P_{2,1}$
R11: $\neg P_{1,2}$
R12: $\neg P_{2,1}$

First application of **Resolution** using R18 and R1:

$$\frac{(P_{1,1} \vee P_{3,1}) \wedge \neg P_{1,1}}{P_{3,1}}$$

R13: $\neg B_{1,2}$
R14: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
R15: $\neg P_{2,2}$
R16: $\neg P_{1,3}$
R17: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

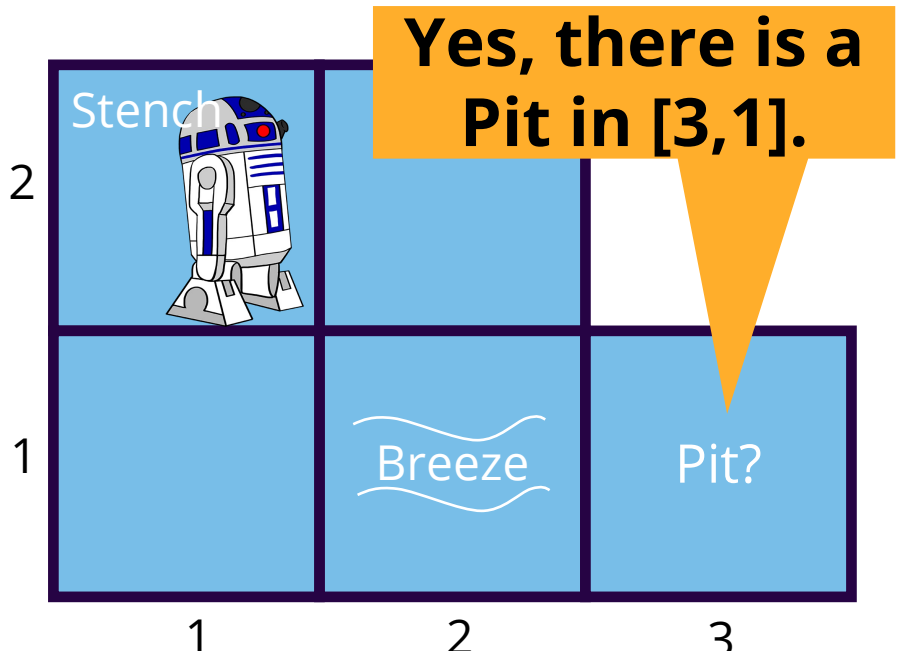
R18: $(P_{1,1} \vee P_{3,1})$

There must be a pit in [1,1] or a pit in [3,1]

R19: $P_{3,1}$

Resolution: There must be a pit in [3,1]

Yes, there is a Pit in [3,1].



Resolution

If ℓ_i and m are complementary literals, which means that one is the negation of another, then we can eliminate them via **unit resolution.**

$$\frac{(\ell_1 \vee \ldots \vee \ell_k), \quad m}{(\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k)}$$

Resolution

If ℓ_i and m are complementary literals, which means that one is the negation of another, then we can eliminate them via **unit resolution.**

$$\frac{(\ell_1 \vee \ldots \vee \ell_k), \quad m}{(\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k)}$$

Full **resolution rule** for complementary literals ℓ_i and m_j

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \quad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ell_{i-1} \vee \ldots \vee \ell_{i+1} \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee \ldots \vee m_{j+1} \vee m_k}$$

Resolution

If ℓ_i and m are complementary literals, which means that one is the negation of another, then we can eliminate them via **un**

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \quad m}{\ell_1 \vee}$$

The resolution rule only applies to clauses (== disjunctions of literals).

Answer: Every sentence can is equivalent to a conjunction of clauses.

ementary literal

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ell_{i-1} \vee \ldots \vee \ell_{i+1} \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee \ldots \vee m_{j+1} \vee m_k}$$

Our KB has full set of logical sentences.

How can it lead to a complete inference procedure for all of propositional logic?

Conjunctive Normal Form (CNF)

Every sentence of propositional logic is logically equivalent to a conjunction of clauses. Clauses are disjunctions of literals.

1. Eliminate $\Leftrightarrow$ by replacing $\boldsymbol{\alpha \Leftrightarrow \beta}$ with $\boldsymbol{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$
2. Eliminate $\Rightarrow$ by replacing $\boldsymbol{\alpha \Rightarrow \beta}$ with $\boldsymbol{\neg \alpha \vee \beta}$
3. Move $\neg$ inwards (eliminate double negation, and apply De Morgan's rules)
4. Now the sentence will contain nested $\wedge$ and $\vee$ operators applied to literals. Apply distributivity law to distribute $\vee$ over $\wedge$ wherever possible.

The sentence is now in CNF!

Resolution Algorithm

Inference procedures based on resolution work using the principle of **proof by contradiction.**

To show **KB⊨α** then we can show that **KB ∧ ¬α** is **unsatisfiable.**

Here's how:

- First, convert **(KB ∧ ¬α)** into **CNF**
- Then apply the **resolution rule** to the resulting clauses.
- Resolve every pair that contain **complementary literals** to produce **a new clause**
- If that new clause isn't already present, then **add it to the set**.
- Continue the process until one of two things happens:

1. There are **no new clauses** that can be added. In this case, the KB does not entail **α**
2. Two clauses resolve to yield the **empty clause,** in which case the KB does entail **α**

Resolution Algorithm

Inference procedures based on resolution work using the principle of **proof by contradiction.**

To show **KB⊨α** then we can show that **KB ∧ ¬α** is **unsatisfiable.**

Here's how:

- First, convert **(KB ∧ ¬α)** into **CNF**
- Then apply the **resolution rule** to the res
- Resolve every pair that contain **compleme**
- If that new clause isn't already present, then **add**
- Continue the process until one of two things ha

1. There are **no new clauses** that can be adde . In this case, the KB does not entail **α**
2. Two clauses resolve to yield the **empty clause,** in which case the KB does entail **α**

The empty clause arises from resolving two contradictory unit clauses like P and ¬P.

Resolution Algorithm

```
function PL-RESOLUTION(KB, α)
  inputs: KB, the knowledge base,
          α, the query, a sentence i

  clauses ← the set of clauses in the CNF r          ion of KB ∧ ¬α
  new ← { }
  while true do
      for each pair of clauses C_i, C_j in clauses do
          resolvents ← PL-RESOLVE(C_i, C_j)
          if resolvents contains the empty clause then return true
          new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

We are comparing every pair of clauses, potentially multiple times.

Horn Clauses and Definite Clauses

The resolution algorithm is **complete!** This makes is a very important inference method.

However, we don't always need the full power of resolution. Some KBs are formulated in a **more restrictive** form which enables them to use **more efficient** inference algorithms.

- **Definite clause –** disjunction of literals where **exactly one is positive**
- **Horn clause** – disjunction of literals where **at most one is positive**
  - **This includes definite clauses**
  - **And clauses with no positive literals**

Horn clauses are closed under resolution. If you apply resolution to a horn clause, you get back a horn clause.

KBs with only definite clauses

1. Every definite clause can be written as an implication whose premise is a conjunction of positive literals whose conclusion is a positive literal.

   $\neg L_{1,1} \vee \neg \textbf{Breeze} \vee B_{1,1}$ can be written as the implication $L_{1,1} \wedge \textbf{Breeze} \Rightarrow B_{1,1}$

2. Inference with Horn clauses can be done through the forward-chaining and backward-chaining algorithms
3. Deciding entailment with Horn clauses can be done in **linear time** with respect to the size of the KB.

Forward Chaining Algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
    p ← POP(queue)
    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 queue
  return false
```

Definite clause – disjunction of literals where **exactly one is positive** like
¬ $L_{1,1}$ V ¬ Breeze V $B_{1,1}$

Can be rewritten as
L1,1 ∧ Breeze ⟹ B1,

Track how many symbols in the premise of
L1,1 ∧ Breeze ⟹ B
are true.

If count of clause
L1,1 ∧ Breeze ⟹ B1,1
is 0, then we have seen all the elements in premise so conclusion must be true.

Forward Chaining and Backward Chaining

Forward Chaining is a kind of **data-driven reasoning.** An agent can start with known data, adding premises as new percepts come in, and then incrementally apply the forward chaining algorithm to derive new conclusions. It doesn't have to have a specific query in mind to do so.

Backward chaining works backwards from a query. If the query is known to be true, then no work is needed. Otherwise, the algorithm finds implications in the KB whose conclusions is the query. If all the premises of one of those can be proved to be true (via backward chaining), then the query is true.

Backward chaining is a form of goal-directed reasoning.

CIS 4210/5210:
ARTIFICIAL INTELLIGENCE

Logical Agents