Agent Based Systems

Paolo Turrini

www.dcs.warwick.ac.uk/~pturrini p.turrini@warwick.ac.uk

The plan for today

We have seen a way of modelling knowledge using set theory, both in single and multi agent situations

Today and tomorrow we look at how to model this from a point of view of a reasoning agent

- Knowledge Representation:
  - how to encode the state of the world in a logical language
  - worlds, actions, goals
  - expressivity versus complexity
- We focus on one agent only! (MAS languages: similar pattern)

KR is one of the most active sub-fields of AI, producing many useful results in many areas (e.g., automated medical diagnosis)

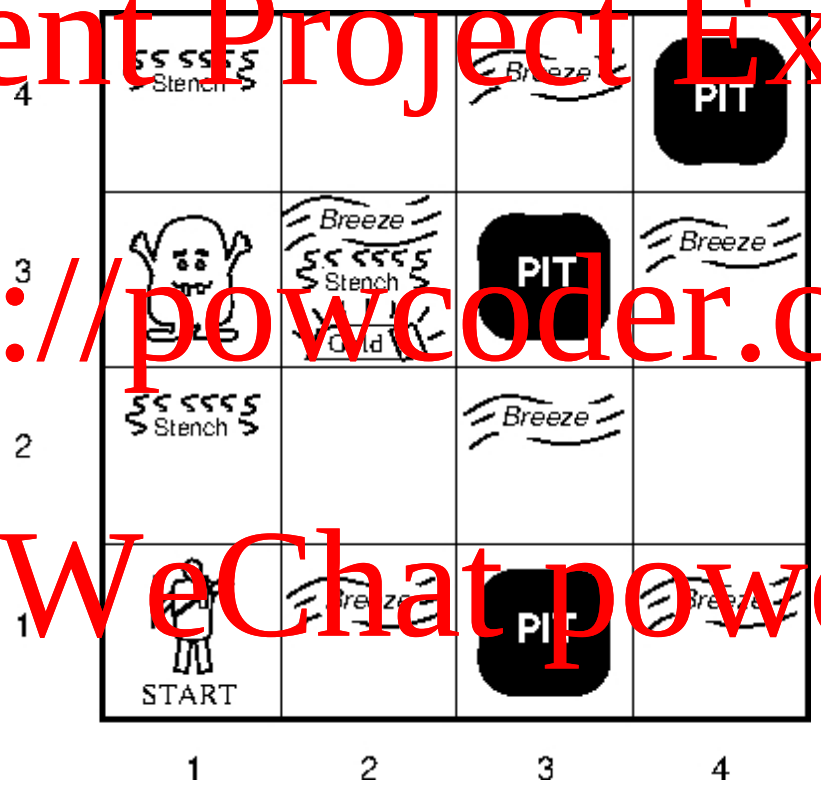
Knowledge Representation

An uncertain world

The main reference

- Stuart Russell and Peter Norvig
  Artificial Intelligence: a modern approach
  Chapters 7-9

The Wumpus World



The Wumpus World

Sensors Breeze, Glitter, Smell

Actuators Up, Down, Left, Right, Grab, Release, Shoot, Climb

Rewards 1000 escaping with gold, -1000 dying, -10 using arrow, -1 walking

Environment

- Squares adjacent to Wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills Wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square

Knowledge base

- A set of sentences representing what the agent thinks about the world.
  - 'I am in [2,1]'
  - 'I am out of arrows'
  - 'I smell Wumpus'
  - 'I'd better not go forward'
- We interpret it as what the agent **knows**, but it works just fine for what the agent **believes**.

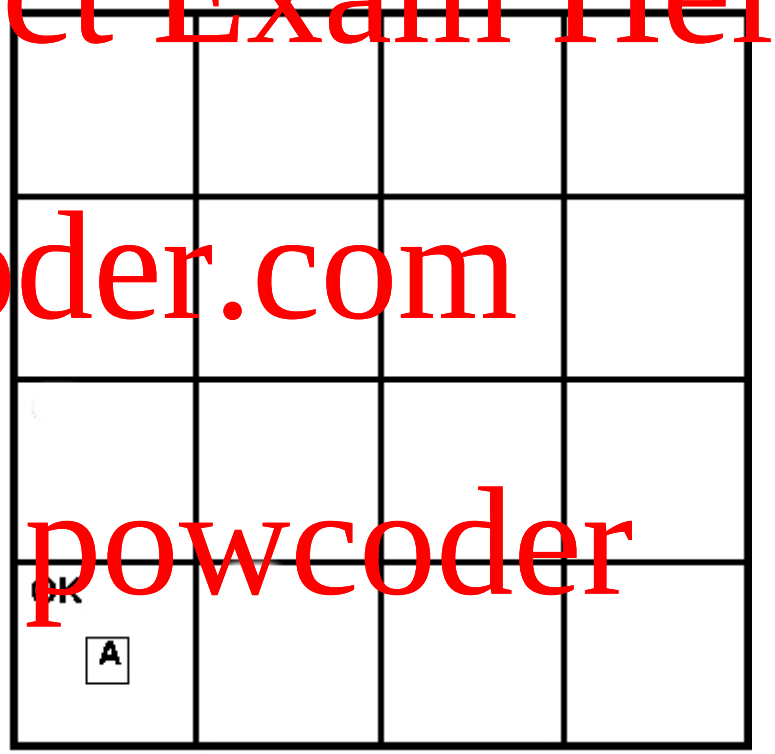
Updating the knowledge base

- What we TELL the knowledge base
- What we ASK the knowledge base


```
function KB-Agent( percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time
    Tell(KB, Make-Percept-Sentence( percept, t))
    action ← Ask(KB, Make-Action-Query(t))
    Tell(KB, Make-Action-Sentence(action, t))
    t ← t + 1
    return action
```

Rational explorations

- The starting state...

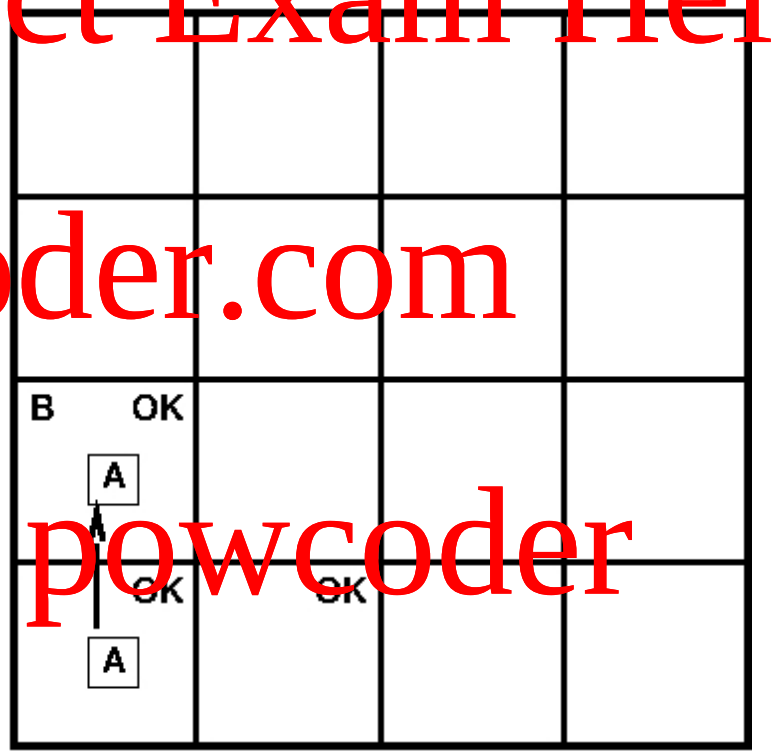


Rational explorations

- and what we know.

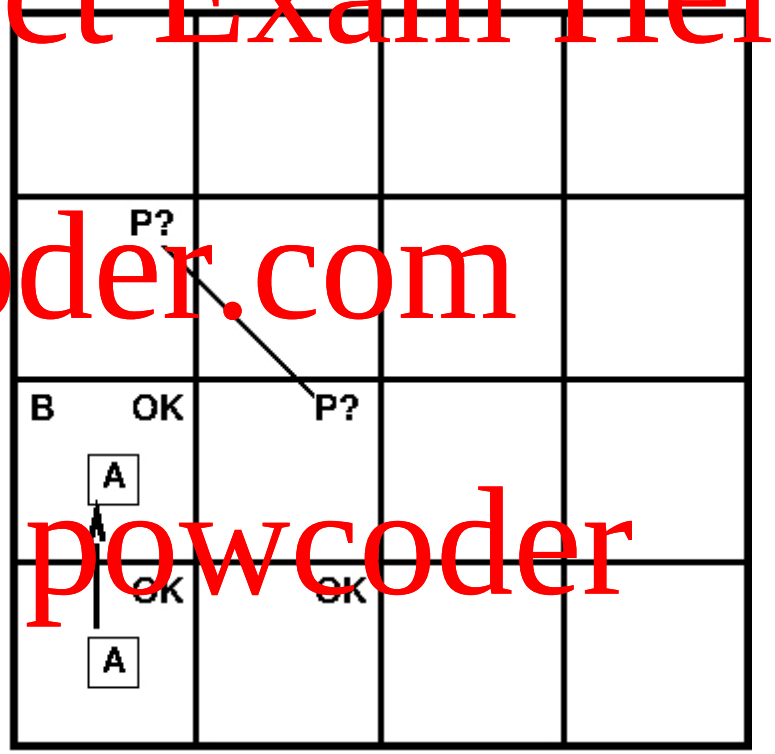
Rational explorations

- B stands for Breeze



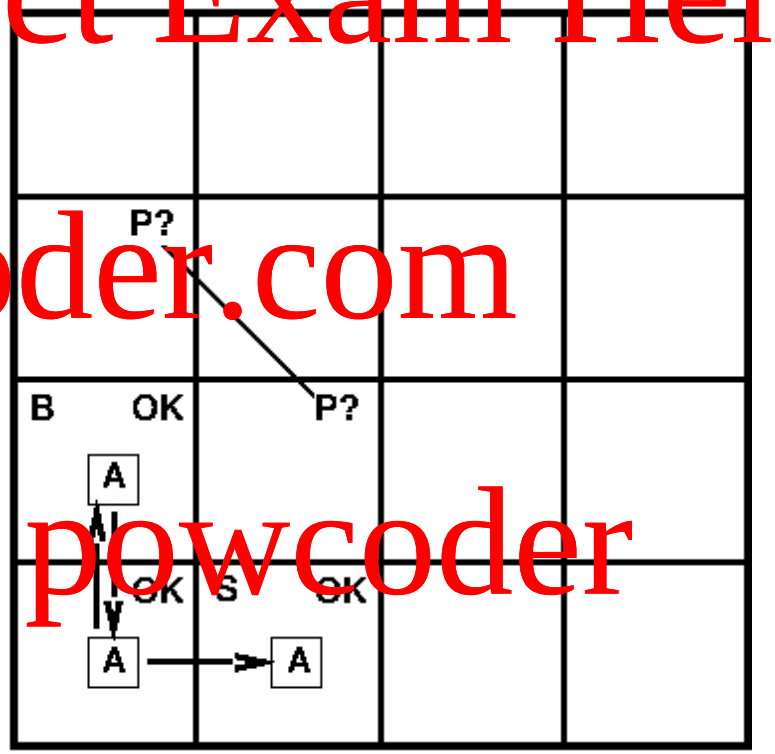
Rational explorations

- Where is the pit?
- We are ruling out one square!



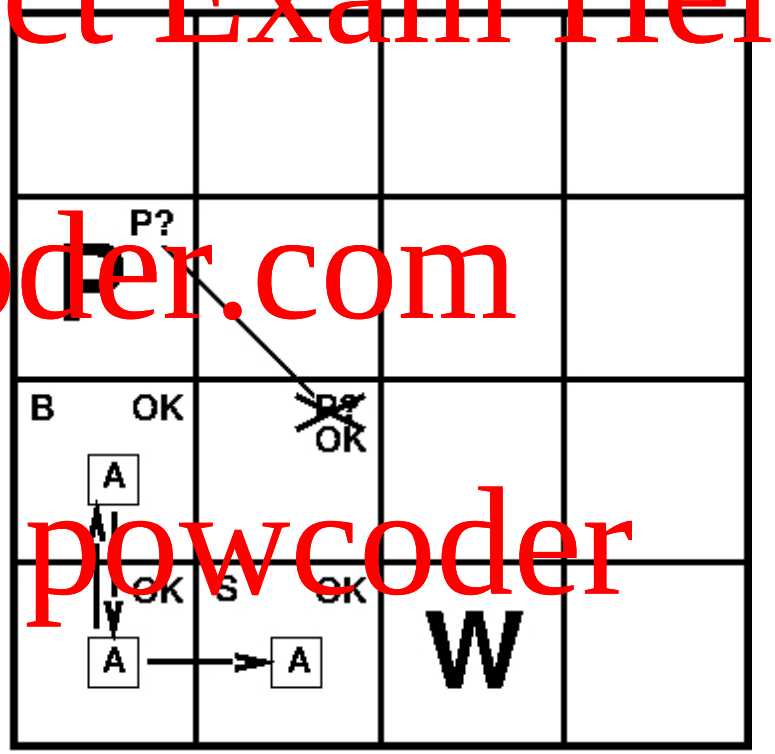
Rational explorations

- S stands for smell
- What do we know?



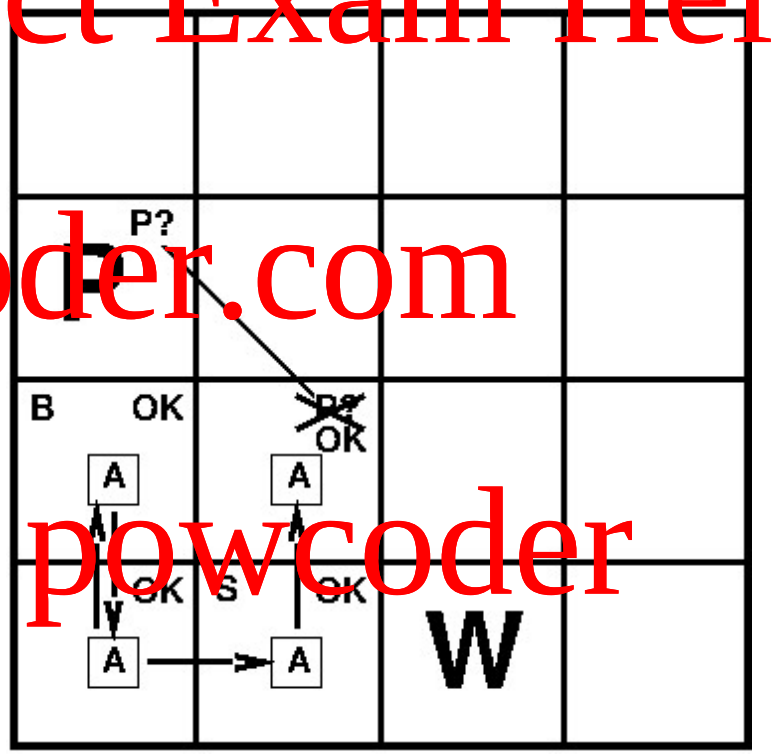
Rational explorations

- Logic is the key!



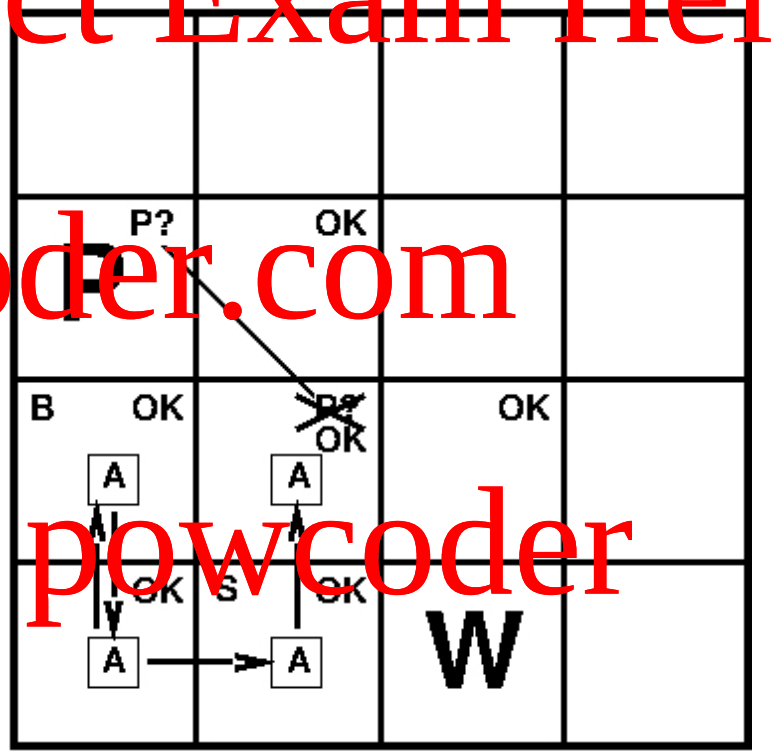
Rational explorations

- The further we go the more we know



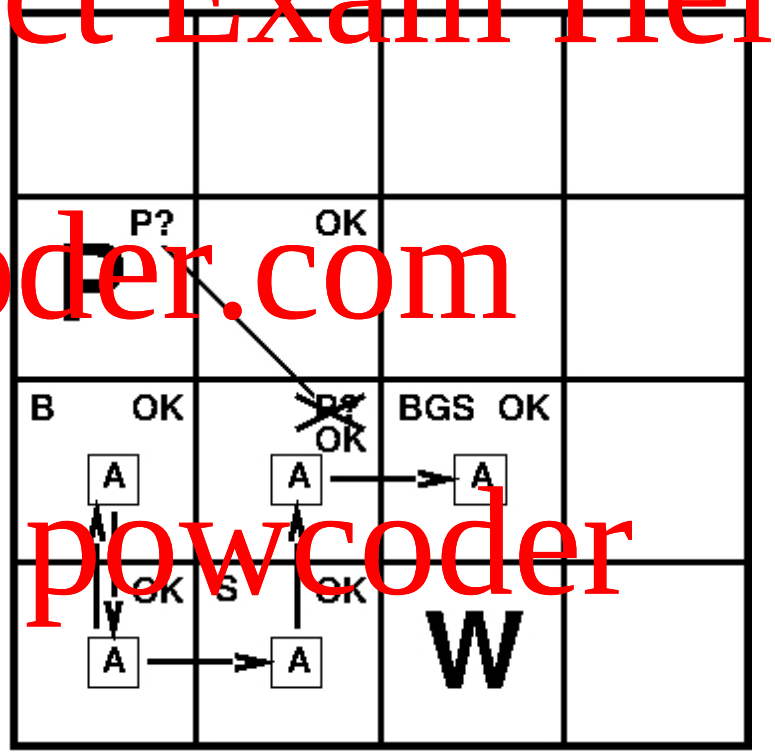
Rational explorations

- The further we go the more we know



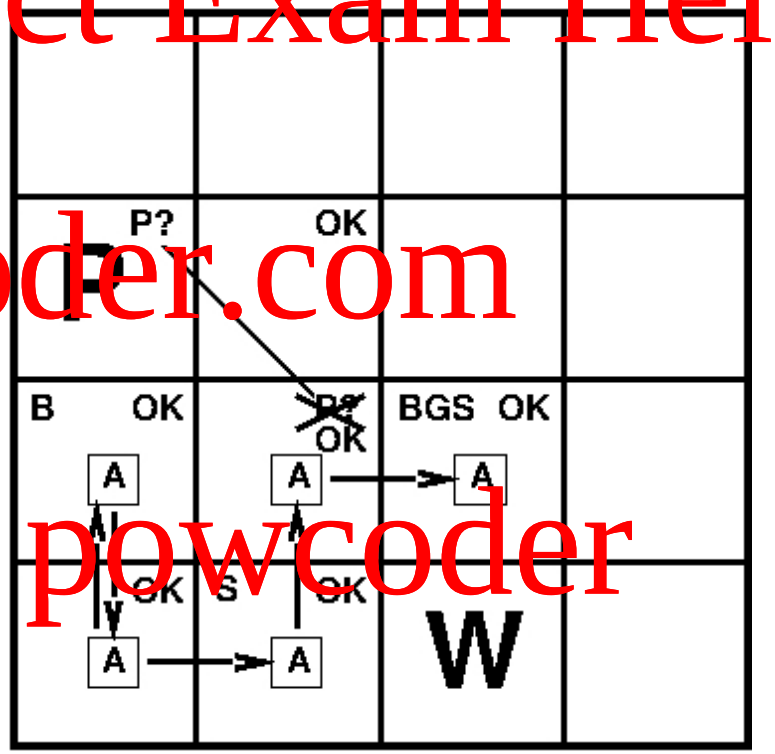
Rational explorations

- Gold!



Rational explorations

- We know the way out
- Game over



Representing the Wumpus World

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

Representing the Wumpus World

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

Representing the Wumpus World

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

Representing the Wumpus World

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

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

"A square is breezy **if and only if** there is an adjacent pit"

Expressivity: at what cost?

- OK if we were only dealing with finite objects
- But even then we would have to enumerate all the possibilities

Expressivity: at what cost?

- OK if we were only dealing with finite objects
- But even then we would have to enumerate all the possibilities

Propositional Logic lacks expressive power

- Massive increase of expressivity: we can use existential $\exists$ and universal $\forall$ quantifiers before sentences!
- But there are costs, e.g., decidability: we don't always have a way to establish whether a sentences is true or false.
- We will see how to exploit the gains while limiting the costs

Recall: Universal Instantiation

Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v\ \alpha}{\alpha(\{v/g\})}$$

for any variable v and ground term g

Recall: Universal Instantiation

Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v \ \alpha}{\alpha(\{v/g\})}$$

for any variable v and ground term g

E.g., $\forall x \ \ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields

Recall: Universal Instantiation

Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v \ \alpha}{\alpha(\{v/g\})}$$

for any variable v and ground term g

E.g., $\forall x \ \ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields

$$King(Joffrey) \wedge Greedy(Joffrey) \Rightarrow Evil(Joffrey)$$
$$King(Aerys) \wedge Greedy(Aerys) \Rightarrow Evil(Aerys)$$
$$King(Father(Joffrey)) \wedge Greedy(Father(Joffrey)) \Rightarrow Evil(Father(Joffrey))$$
$$\vdots$$

Recall: Existential instantiation (EI)

For any sentence α, variable v, and constant symbol k **that does not appear elsewhere in the knowledge base**:

$$\frac{\exists v \ \alpha}{\alpha(\{v/g\})}$$

Recall: Existential instantiation (EI)

For any sentence α, variable v, and constant symbol k **that does not appear elsewhere in the knowledge base**:

$$\frac{\exists v \ \alpha}{\alpha(\{v/k\})}$$

E.g., $\exists x \ \ Crown(x) \wedge OnHead(x, John)$ yields

Recall: Existential instantiation (EI)

For any sentence α, variable v, and constant symbol k **that does not appear elsewhere in the knowledge base**:

$$\frac{\exists v \ \alpha}{\alpha(\{v/k\})}$$

E.g., $\exists x \ \ Crown(x) \wedge OnHead(x, John)$ yields

$Crown(C_1) \wedge OnHead(C_1, John)$

Recall: Existential instantiation (EI)

For any sentence α, variable v, and constant symbol k **that does not appear elsewhere in the knowledge base**:

$$\frac{\exists v \ \alpha}{\alpha(\{v/g\})}$$

E.g., $\exists x \ \ Crown(x) \wedge OnHead(x, John)$ yields

$$Crown(C_1) \wedge OnHead(C_1, John)$$

provided C_1 is a **new** constant symbol, called a Skolem constant

Recall: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

Recall: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence;

Recall: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence;

the new KB is **not** equivalent to the old,

Recall: Existential instantiation contd.

UI can be applied several times to **add** new sentences;

the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence;

the new KB is **not** equivalent to the old,

but is satisfiable iff the old KB was satisfiable

KB with FOL

- We can encode the KB at each particular time point using FOL


```
function KB-Agent( percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time
  Tell(KB, Make-Percept-Sentence( percept, t))
  action ← Ask(KB, Make-Action-Query(t))
  Tell(KB, Make-Action-Sentence(action, t))
  t ← t + 1
  return action
```

Representing the Wumpus World

- You already know how to describe the WW in first order logic

Representing the Wumpus World

- You already know how to describe the WW in first order logic
  - Percept (at given time), e.g., $Percept([Stench, Breeze, Glitter], 5)$ or $Percept([None, Breeze, None], 3)$

Representing the Wumpus World

- You already know how to describe the WW in first order logic
  - Percept (at given time), e.g., $Percept([Stench, Breeze, Glitter], 5)$ or $Percept([None, Breeze, None], 3)$
  - Starting Knowledge Base, e.g., $\neg AtGold(0)$

Representing the Wumpus World

- You already know how to describe the WW in first order logic
  - Percept (at given time), e.g., $Percept([Stench, Breeze, Glitter], 5)$ or $Percept([None, Breeze, None], 3)$
  - Starting Knowledge Base e.g., $\neg AtGold(0)$
  - Axioms to generate new knowledge from percepts, e.g., $\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Representing the Wumpus World

- You already know how to describe the WW in first order logic
  - Percept (at given time), e.g., $Percept([Stench, Breeze, Glitter], 5)$ or $Percept([None, Breeze, None], 3)$
  - Starting Knowledge Base e.g., $\neg AtGold(0)$
  - Axioms to generate new knowledge from percepts, e.g., $\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$
  - Axioms to generate actions (plans) from KB, e.g., $\forall\, t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$

Representing the Wumpus World

- You already know how to describe the WW in first order logic
  - Percept (at given time), e.g., $Percept([Stench, Breeze, Glitter], 5)$ or $Percept([None, Breeze, None], 3)$
  - Starting Knowledge Base e.g., $\neg AtGold(0)$
  - Axioms to generate new knowledge from percepts, e.g., $\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$
  - Axioms to generate actions (plans) from KB, e.g., $\forall\, t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$
  - Axioms from knowledge to knowledge, e.g., $\forall\, t \;\; AtGold(t) \wedge Action(Grab, t) \;\Rightarrow\; Holding(Gold, t+1)$

Perception $\forall s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Representing the Wumpus World

Perception $\forall s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Location $At(Agent, s, t)$

Representing the Wumpus World

Perception $\forall s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Location $At(Agent, s, t)$

Decision-making $\forall t \;\; AtGold(t) \Rightarrow Action(Grab, t)$

Representing the Wumpus World

Perception $\forall s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Location $At(Agent, s, t)$

Decision-making $\forall t \;\; AtGold(t) \Rightarrow Action(Grab, t)$

Internal reflection $\forall t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$,
do we have gold already? (notice we cannot observe if we are holding gold: we need to track it)

Representing the Wumpus World

Adjacent squares

$$\forall x, y, a, b \;\; Adjacent([x, y], [a, b]) \;\Leftrightarrow$$
$$(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1))$$

Representing the Wumpus World

Adjacent squares

$$\forall x, y, a, b \;\; Adjacent([x, y], [a, b]) \;\; \Leftrightarrow$$
$$(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1))$$

"A square is breezy **if and only if** there is an adjacent pit"

$$\forall s \;\; , Breezy(s) \;\; \Leftrightarrow \;\; \exists r \, (Adjacent(r, s) \wedge Pit(r))$$

Representing the Wumpus World

- We can go on and describe plans, causal rules, etc.
- But let's do some reasoning now

'Joffrey Baratheon is a king'

'Jon Snow is a person'

Facts and knowledge bases

'Jon Snow is a king'

$Tell(KB, King(Joffrey))$

Telling and Asking

$Tell(KB, King(Joffrey))$

$Tell(KB, Person(Jon))$

Telling and Asking

$Tell(KB, King(Joffrey))$

$Tell(KB, Person(Jon))$

$Tell(KB, \forall x\ King(x) \Rightarrow Person(x))$

Telling and Asking

$Tell(KB, King(Joffrey))$

$Tell(KB, Person(Jon))$

$Tell(KB, \forall x\, King(x) \Rightarrow Person(x))$

$Ask(KB, \exists x Person(x))$ is there a person?

Telling and Asking

$Tell(KB, King(Joffrey))$

$Tell(KB, Person(Jon))$

$Tell(KB, \forall x\ King(x) \Rightarrow Person(x))$

$Ask(KB, \exists x Person(x))$ is there a person?

$Askvar(KB, Person(x))$ who is a person?

Telling and Asking

$Tell(KB, King(Joffrey))$

$Tell(KB, Person(Jon))$

$Tell(KB, \forall x\ King(x) \Rightarrow Person(x))$

$Ask(KB, \exists x Person(x))$ is there a person?

$Askvar(KB, Person(x))$ who is a person?

Askvar returns a list of **substitutions**: $\{x/Joffrey\}, \{x/Jon\}$

Substitutions

Definition

Given a sentence S and a substitution σ,

Substitutions

Definition

Given a sentence S and a substitution σ,

$S\sigma$ denotes the result of plugging σ into S; e.g.,

Substitutions

Definition

Given a sentence S and a substitution σ,

$S\sigma$ denotes the result of plugging σ into S; e.g.,

$S = Smarter(x, y)$

Substitutions

Definition

Given a sentence S and a substitution σ,

$S\sigma$ denotes the result of plugging σ into S; e.g.,

$S = Smarter(x, y)$

$\sigma = \{x/Tyrion, y/Joffrey\}$

Substitutions

Definition

Given a sentence S and a substitution σ,

$S\sigma$ denotes the result of plugging σ into S; e.g.,

$S = Smarter(x, y)$

$\sigma = \{x/Tyrion, y/Joffrey\}$

$S\sigma = Smarter(Tyrion, Joffrey)$

Substitutions

Definition

Given a sentence S and a substitution σ,

$S\sigma$ denotes the result of plugging σ into S; e.g.,

$S = Smarter(x, y)$

$\sigma = \{x/Tyrion, y/Joffrey\}$

$S\sigma = Smarter(Tyrion, Joffrey)$

$Askvar(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Unification

$$\forall x \;\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

$$King(Joffrey)$$

$$\forall y \;\; Greedy(y)$$

Unification

$$\forall x \;\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

$$King(Joffrey)$$

$$\forall y \;\; Greedy(y)$$

We can get the inference immediately if we can find a substitution matching the premises of the implication to the known facts.

Unification

$$\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

$$King(Joffrey)$$

$$\forall y \; Greedy(y)$$

We can get the inference immediately if we can find a substitution matching the premises of the implication to the known facts.

$\theta = \{x/Joffrey, y/Joffrey\}$ works

UNIFY(α, β) returns θ if $\alpha\theta = \beta\theta$

Unification

$\textsc{Unify}(\alpha, \beta)$ returns θ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(Joffrey, x)$	$Knows(Joffrey, Sansa)$	
$Knows(Joffrey, x)$	$Knows(y, Sansa)$	
$Knows(Joffrey, x)$	$Knows(y, Mother(Joffrey))$	
$Knows(Jon, x)$	$Knows(x, Mother(Jon))$	

Unification

$\textsc{Unify}(\alpha, \beta)$ returns θ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(Joffrey, x)$	$Knows(Joffrey, Sansa)$	$\{x/Sansa\}$
$Knows(Joffrey, x)$	$Knows(y, Sansa)$	
$Knows(Joffrey, x)$	$Knows(y, Mother(Joffrey))$	
$Knows(Jon, x)$	$Knows(x, Mother(Jon))$	

Unification

$\textsc{Unify}(\alpha, \beta)$ returns θ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(Joffrey, x)$	$Knows(Joffrey, Sansa)$	$\{x/Sansa\}$
$Knows(Joffrey, x)$	$Knows(y, Sansa)$	$\{x/Sansa, y/Joffrey\}$
$Knows(Joffrey, x)$	$Knows(y, Mother(Joffrey))$	
$Knows(Jon, x)$	$Knows(x, Mother(Jon))$	

Unification

$\textsc{Unify}(\alpha, \beta)$ returns θ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(Joffrey, x)$	$Knows(Joffrey, Sansa)$	$\{x/Sansa\}$
$Knows(Joffrey, x)$	$Knows(y, Sansa)$	$\{x/Sansa, y/Joffrey\}$
$Knows(Joffrey, x)$	$Knows(y, Mother(Joffrey))$	$\{y/Joffrey, x/Mother(Joffrey)\}$
$Knows(Jon, x)$	$Knows(x, Mother(Jon))$	

Unification

$\textsc{Unify}(\alpha, \beta)$ returns θ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(Joffrey, x)$	$Knows(Joffrey, Sansa)$	$\{x/Sansa\}$
$Knows(Joffrey, x)$	$Knows(y, Sansa)$	$\{x/Sansa, y/Joffrey\}$
$Knows(Joffrey, x)$	$Knows(y, Mother(Joffrey))$	$\{y/Joffrey, x/Mother(Joffrey)\}$
$Knows(Jon, x)$	$Knows(x, Mother(Jon))$	$fail$

Standardising apart

$Knows(Jon, x)$ & $Knows(x, Mother(Jon))$ fails

Standardising apart

$Knows(Jon, x)$ & $Knows(x, Mother(Jon))$ fails

Standardising apart eliminates overlap of variables, e.g.,

$Knows(z_{17}, Mother(Jon))$

Generalized Modus Ponens (GMP)

Definite clause:

disjunction of literals, **exactly** one of which positive

Generalized Modus Ponens (GMP)

Definite clause:

disjunction of literals, **exactly** one of which positive

eg., $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

Generalized Modus Ponens (GMP)

Definite clause:

disjunction of literals, **exactly** one of which positive

e.g., $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$ where $p_i'\theta = p_i\theta$ for all i

Generalized Modus Ponens (GMP)

Definite clause:

disjunction of literals, **exactly** one of which positive

eg., $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \quad \text{where } p_i'\theta = p_i\theta \text{ for all } i$$

Assuming all variables are universally quantified...

Generalized Modus Ponens (GMP)

Definite clause:

disjunction of literals, **exactly** one of which positive

eg., $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \quad \text{where } p_i'\theta = p_i\theta \text{ for all } i$$

Assuming all variables are universally quantified...

p_1' is $King(Joffrey)$	p_1 is $King(x)$
p_2' is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/Joffrey, y/Joffrey\}$	q is $Evil(x)$
$q\theta$ is $Evil(Joffrey)$	

Soundness of GMP

Need to show that

$$p_1', \ldots, p_n', \quad (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

provided that $p_i'\theta = p_i\theta$ for all i

Lemma: If φ is definite clause, then $\varphi \models \varphi\theta$ by Universal Instantiation.

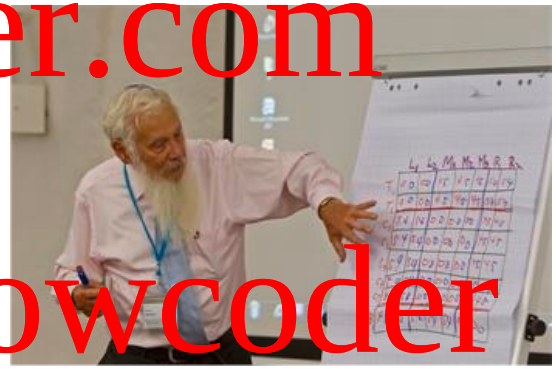
1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1', \ldots, p_n \models p_1' \wedge \ldots \wedge p_n \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

In this lecture

- How to describe the world in logic
- Moving as a way to gather new facts
- Generalised modus ponens

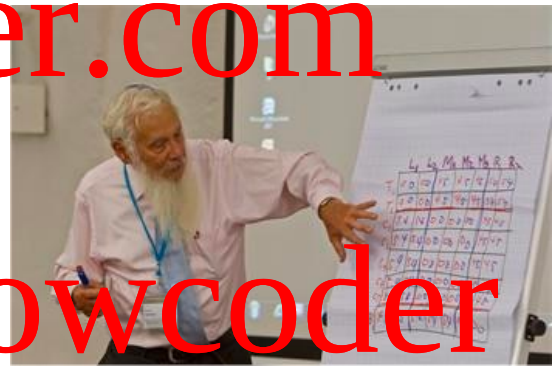
In this lecture

- How to describe the world in logic
- Moving as a way to gather new facts
- Generalised modus ponens



In this lecture

- How to describe the world in logic
- Moving as a way to gather new facts
- Generalised modus ponens



In this lecture

- How to describe the world in logic
- Moving as a way to gather new facts
- Generalised modus ponens