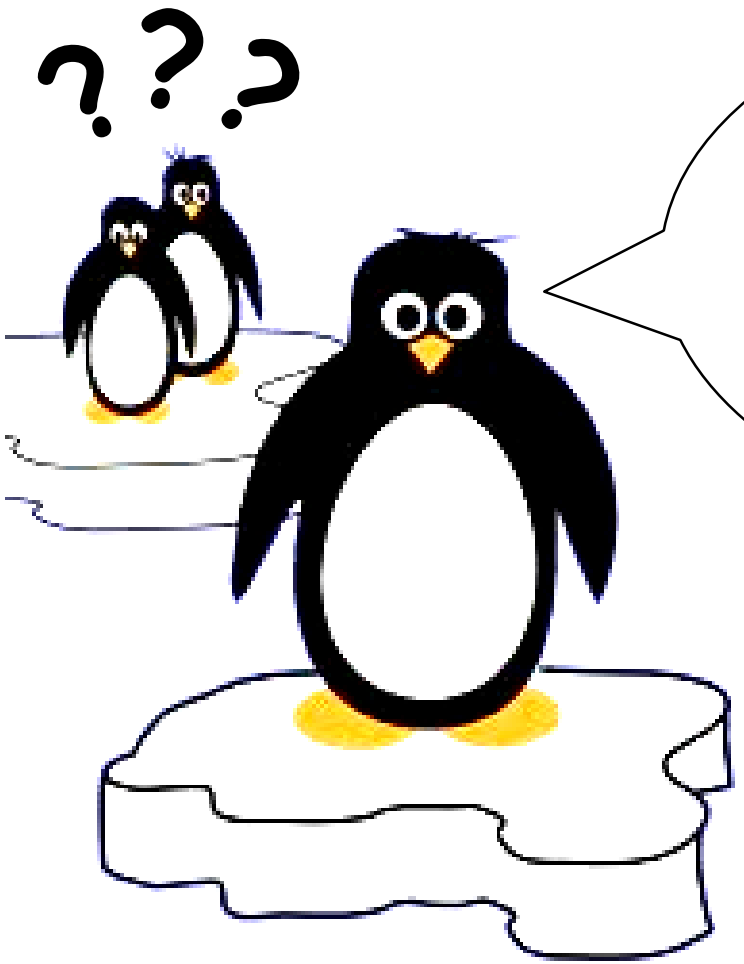
Logical Agents

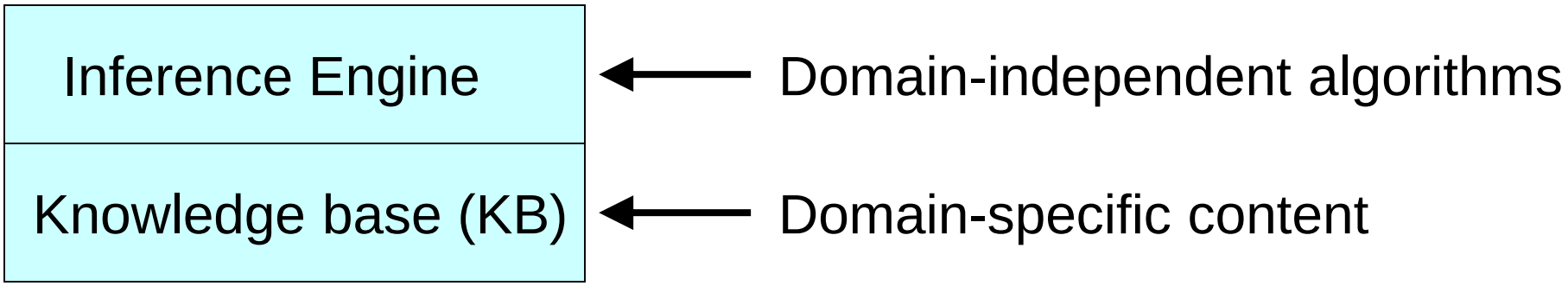


Penguins are black and white. Some old TV shows are black and white. Therefore, some penguins are old TV shows.

What's wrong with this reasoning?

Logical Agents

- A logical agent consists of the following two components:



- Logical agents are not limited to specific tasks; they can utilize the knowledge to solve different problems.
- The KB can be updated to add new information or adapt to environment changes.

The Knowledge Base

- The knowledge in a KB is encoded by a set of **sentences** in a knowledge representation language.
- **Axioms**: sentences that are taken as given without any derivation.
- **Inference**: The process of deriving new sentences from existing ones.
- Background knowledge: The initial content of a KB.

Knowledge-Based Agents

```
function KB-Agent(percept) returns an action
persistent:     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
```

- **Tell** the *KB* what is perceived.
- **Ask** the *KB* to get suitable actions. Extensive reasoning may be done in this step.
- **Tell** the *KB* that the action is taken (so that the *KB* can update the state).

Logic in General

Here we use arithmetic as an example …

- **Syntax** (how a sentence is formed):
  - $x+y=4$ is a sentence; $=xy4+$ is not
- **Semantics** (meaning of sentence):
  - $x+y=4$ is true if x and y add to 4
- **Model** (a model is a possible way the world is)
  - e.g., $(x=1, y=2)$, $(x=2, y=2)$, $(x=3, y=1)$, etc., are all models of a world specified by x and y.
- We say a model satisfies a sentence α or is a model of α if α is true for that model.
  - e.g., the sentence $x+y=4$ is satisfied by the model $(x=2, y=2)$ but not by the model $(x=1, y=2)$.
  - $M(\alpha)$ is the set of all the models of α.

Entailment

- **Entailment** is a relation between sentences based on semantics (true or false).
- $\alpha \models \beta$ (α entails β) means that whenever α is true, then β is also true.
- $\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$
- Example: $x=0$ entails $xy=0$. The former is stronger (or more limiting) on the possible models.

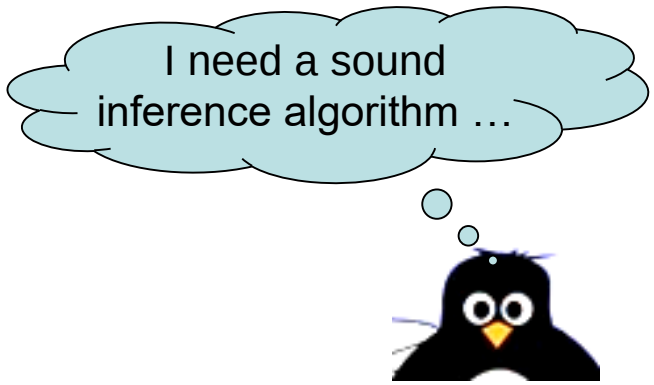
Entailment and Inference

- **Inference** is the process of deriving a new sentence from a set of known sentences.
- $KB \vdash_i \alpha$ means that α can be derived from KB using an inference algorithm i.
- **Sound** inference algorithm:

$$KB \vdash_i \alpha \Rightarrow KB \models \alpha$$

- **Complete** inference algorithm:

$$KB \models \alpha \Rightarrow KB \vdash_i \alpha$$



Propositional Logic: Syntax

- A very simple logic.
- The syntax defines allowable sentences.
- Atomic sentence: a propositional symbol that is either true or false

Sentence	→ *AtomicSentence* | *ComplexSentence*	
AtomicSentence	→ *True* | *False* | *P* | *Q* | *R* | …	
ComplexSentence	→ (*Sentence*) | [*Sentence*]	
	| ¬*Sentence*	negation
	| *Sentence* ∧ *Sentence*	conjunction
	| *Sentence* ∨ *Sentence*	disjunction
	| *Sentence* ⇒ *Sentence*	implication
	| *Sentence* ⇔ *Sentence*	biconditional

antecedent / premise ⇒ consequent / conclusion

Propositional Logic: Semantics

- The semantics is about how to determine whether a sentence is true or false (its truth value).
- A model in propositional logic is specified by the truth values for all the propositional symbols.
- The rules of determining truth values can be given by a truth table.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Toy Problem: The Wumpus World

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

4x4 grid of rooms

start position: (1,1) facing right

one wumpus, one gold piece, and unknown number of pits randomly located (not at (1,1))

Actuators

Left turn, Right turn, Forward

Grab: pick up the gold

Shoot (there is only one arrow)

Sensors

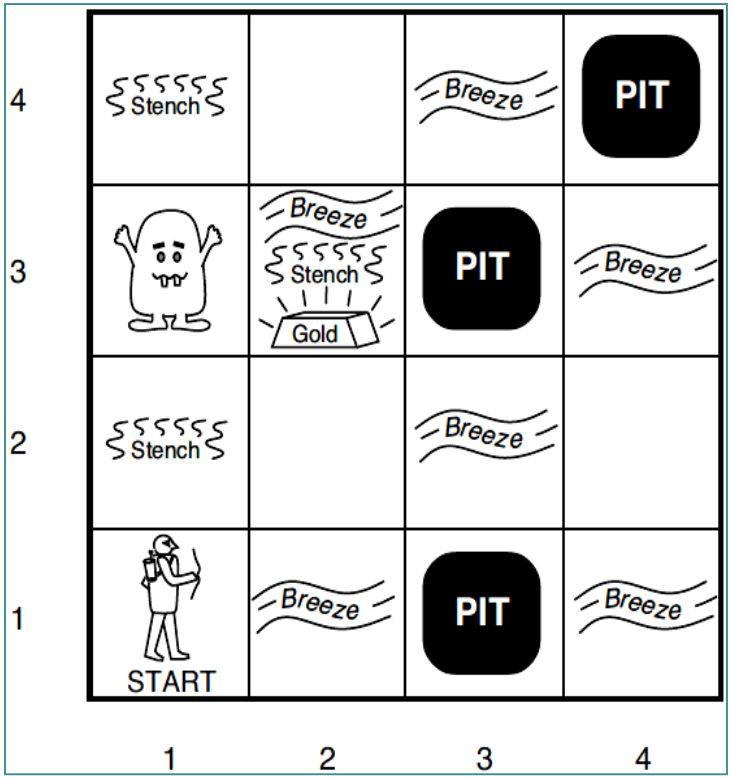
Stench: rooms adjacent to the wumpus

Breeze: rooms adjacent to the pits

Glitter: room containing gold

Bump: attempting to move facing a wall

Scream: when the wumpus is killed



The Wumpus World

Given the map (which is unknown to the agent), can the agent find a safe sequence of actions to retrieve the gold?

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

KB of the Wumpus World

- For simplicity, let's consider only pits and breezes, and the agent has only been at (1,1) and (2,1).
- From the rules of the wumpus world:

R_1: $\neg P_{1,1}$

R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

- What we know from the percepts:

R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

Inference by Model Enumeration

A truth table for all the relevant symbols and KB rules.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	false	false	false	false	false	false	true	true	true	true	false	false
false	false	false	false	false	false	true	true	true	false	true	false	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	true	false	false	false	false	false	true	true	false	true	true	false
false	true	false	false	false	false	true	true	true	true	true	true	true
false	true	false	false	false	true	false	true	true	true	true	true	true
false	true	false	false	false	true	true	true	true	true	true	true	true
false	true	false	false	true	false	false	true	false	false	true	true	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	true	true	true	true	true	true	false	true	true	false	true	false

- The KB entails a sentence α iff $M(KB) \subseteq M(\alpha)$.
- Using the truth table, we can prove that the KB entails $\neg P_{1,2}$.
- However, we can say nothing about $P_{2,2}$ or $\neg P_{2,2}$.

Logical Theorem Proving

- Proof by model enumeration is impractical for most problems.
- Logical theorem proving: A sequence of applications of (sound) inference rules to generate new sentences from existing ones in the KB.
- Two well-known inference rules:
  - Modus Ponens: $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$
  - And-Elimination: $\dfrac{\alpha \wedge \beta}{\alpha}$

$$\frac{\text{given}}{\text{inferred}}$$

- Logical equivalences (next slide) can also be used as inference rules.

Logical Equivalences

Two sentences are logically equivalent iff they are true in the same set of models:

$$\alpha \equiv \beta \text{ iff } \alpha \models \beta \text{ and } \beta \models \alpha$$

$$\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}$$

Validity and Satisfiability

- A sentence is **valid** iff it is true in all models.
  - Examples: $P \vee \neg P$, $P \Rightarrow P$
  - Connected to inference via the **deduction theorem**:

    $\alpha \models \beta$ iff $(\alpha \Rightarrow \beta)$ is valid
- A sentence is **satisfiable** iff it is true in some models.
  - Examples: P, $P \wedge Q$
  - Connected to inference via **proof by contradiction**:

    $\alpha \models \beta$ iff $(\alpha \wedge \neg\beta)$ is unsatisfiable (false in all models)

Example of Inference

Here we can try to apply inference rules to the wumpus world KB to prove $\neg P_{1,2}$, i.e., there is no pit in (1,2):

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$	R_2
$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$	biconditional elimination
$((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) \wedge (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1}))$	commutativitiy
$(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$	and-elimination
$\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$	contraposition
$\neg (P_{1,2} \vee P_{2,1})$	modus ponens + R_4
$\neg P_{1,2} \wedge \neg P_{2,1}$	de Morgan's
$\neg P_{1,2}$	and-elimination

Proof by Search

Q: How can a logical agent find the proof?

A search algorithm can be used to find the proof:

- States: The KB, which grows during the search.
- Initial state: The initial KB.
- Actions: Possible applications of inference rules applied to the sentences in the KB; the sentences need to match the top half of the inference rule.
- Result: Each action generates a new sentence from the bottom half of the inference rule. The new sentence is added to the KB.
- Goal: A state containing the sentence we want to prove.

Inference by Resolution

The resolution rule is **sound and complete** for propositional logic:

$$\frac{l_1 \vee \ldots \vee l_k, \;\; m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

(m is the negation of l_i)

(general form)

$$\frac{l_1 \vee \ldots \vee l_k, \;\; m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

(m_j is the negation of l_i)

(easier to understand)

$$\frac{\alpha \vee \beta, \;\; \neg\beta}{\alpha} \quad \text{and} \quad \frac{\alpha \vee \beta, \;\; \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Conjunctive Normal Form (CNF)

The application of the resolution rule requires the KB to be in **conjunctive normal form** (CNF):

- CNF: The whole KB becomes the conjunction (AND) of a set of clauses.
- **clauses**: disjunctions (OR) of literals, or individual literals
- **literals**: atomic sentences or their negations

Converting a general sentence to CNF:

- biconditional: use biconditional elimination
- implication: use implication elimination
- negation of complex sentences: use De Morgan's Laws
- distribute $\vee$ over $\wedge$ when possible

Example of CNF

Now let us try to convert R_2 of the wumpus world KB to CNF:

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

$$\left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}\right) \wedge \left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})\right)$$

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

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

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

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

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

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

We end up with 3 CNF clauses.

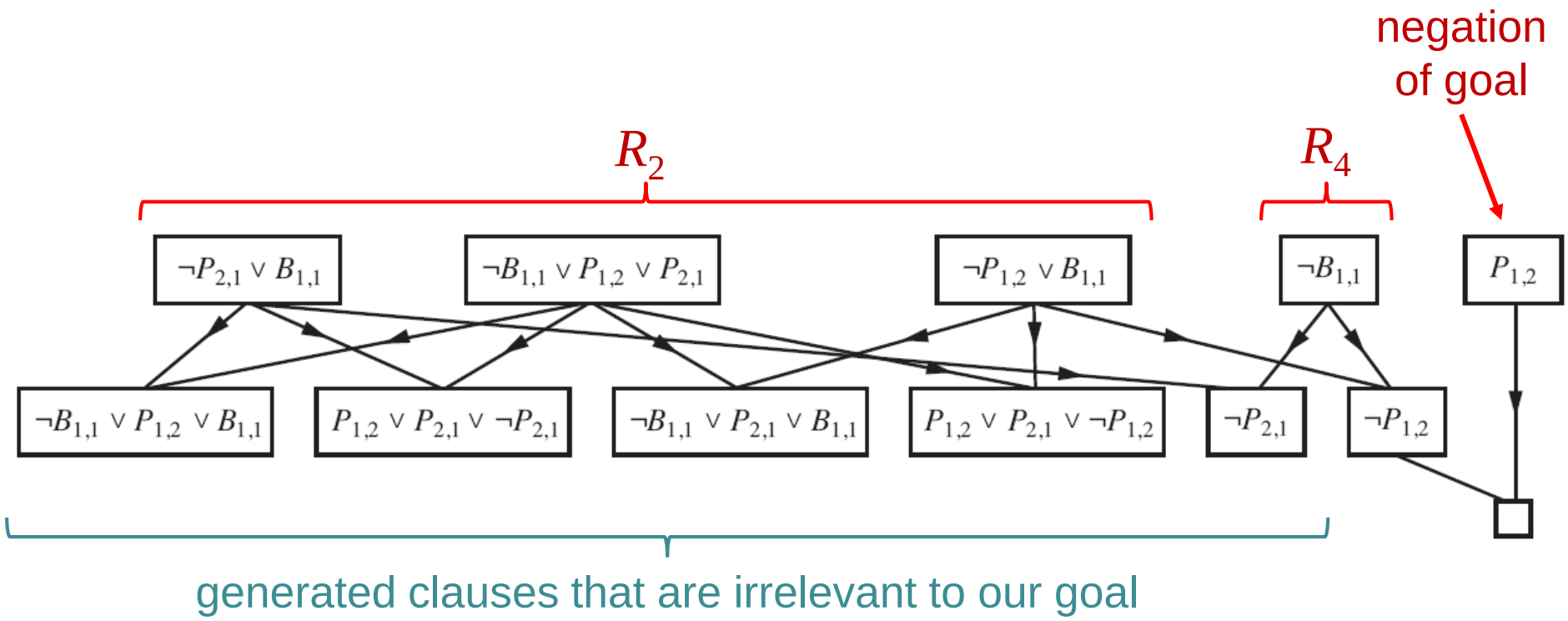
Resolution Algorithm

To prove that the KB entails a sentence α:

- Proof by contradiction: We prove that $(KB \wedge \neg\alpha)$ is **unsatisfiable** (always false).
- We repeatedly combine clause pairs with complementary literals to generate new clauses.
  - If we end up with an empty clause (false), KB entails α.
  - If no new clauses can be added, KB does not entail α.

Example Proof by Resolution

Here we try to see if the wumpus world KB entails $\neg P_{1,2}$.



Horn Clauses

If we limit the types of clauses, we can have more efficient algorithms than the general resolution algorithm.

- Definite clauses: Clauses with exactly one positive literal.
- Goal clauses: Clauses with no positive literal.
- Horn clauses: definite clauses or goal clauses (at most one positive literal).
- Horn clauses are closed under resolution.
- Inference with Horn clauses can be done through forward-chaining or backward-chaining, which have time complexity that is linear in KB size.
- Definite clauses are converted to implications:

$$\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_k \vee Q \quad \equiv \quad (P_1 \wedge P_2 \wedge \ldots \wedge P_k) \Rightarrow Q$$

Forward Chaining (FC)

Data-driven: starting with the known facts

The algorithm uses the following (q: the symbol being querried):

- *agenda*: initially the set of symbols known to be *true*
- *inferred*[p]: initially all *false*
- *count*[c]: the number of symbols in clause c's premise
- In each iteration, a symbol p is popped out of *agenda*. If *inferred*[p] is *false*:
  - If $p = q$, return *true*
  - If p is in c's premise, then decrement *count*[c] by one
  - If *count*[c] becomes zero, add c's conclusion to *agenda*
  - set *inferred*[p] to *true*
- return *false* (q is never reached)

Example of Forward-Chaining

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

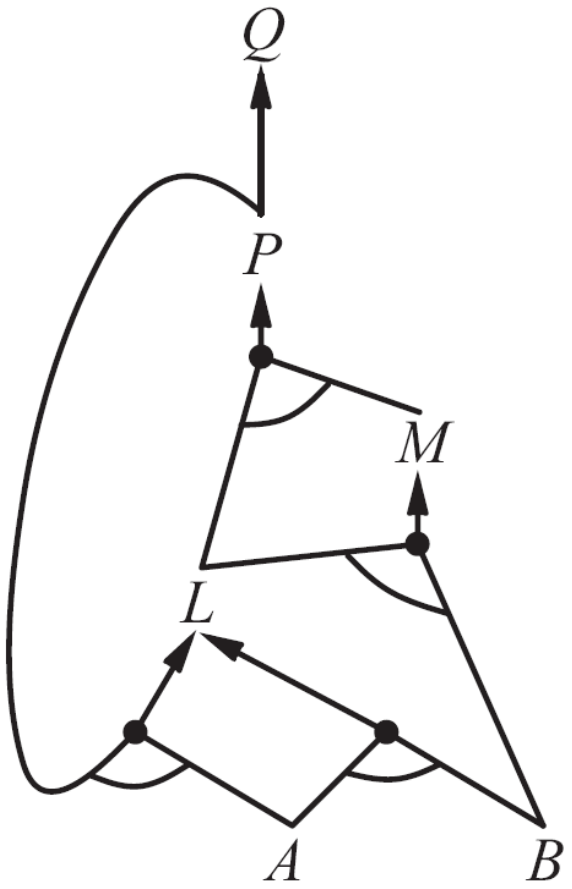
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$$A$$

$$B$$



This is an AND-OR graph. Multiple links joined by an arc is conjunction (AND) and multiple links joined without an arc is disjunction (OR).

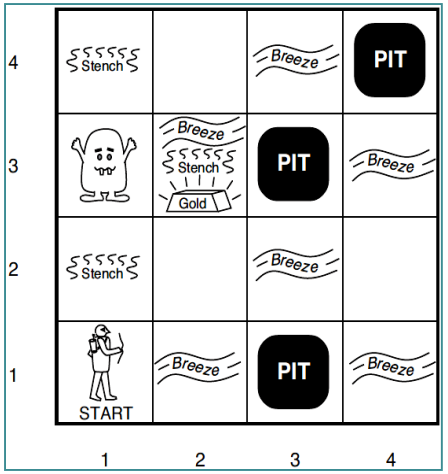
Backward-Chaining (BC)

- Goal-driven: starting with the goal (querried symbol q)
- Working backward: To prove q, find implications in the KB whose conclusion is q.
  - The problem becomes to prove the premises of one of such implications.
  - If the premises of one of such implications are all true, then q is true.
- In practice, BC is much faster than FC because it can skip irrelevant information.
- We will get to the BC algorithm when talking about first-order logic.

First-Order Logic (FOL)

Why FOL?

- Example #1: Try to write down all the sentences (rules in KB) that represent "a pit causes breeze in adjacent squares."
  - How many sentences do we need?
  - How many sentences do we need if the Wumpus world is 100x100 and there are 1000 pits?
- Example #2: How do you represent "Every student in NCTU is smart" in propositional logic?



Why FOL?

- Propositional logic is based on facts about the world, such as $P_{2,2}$. However, we can not have any connection between the two facts $P_{2,2}$ and $P_{3,2}$ even though they are very similar.
  - Propositional logic has very limited expressive power.
- FOL is concerned with objects. Facts represent "attributes of objects" and "relations among objects".
- Example: "x is a student", "x is in NCTU", and "x is smart" are all attributes of x, which can represent any object.
  - The sentence "Every student in NCTU is smart" can be written in this logical expression

    (x is a student) $\wedge$ (x is in NCTU) $\Rightarrow$ (x is smart)

Representations in FOL

- Constant symbols (objects): *John*, *Mary*, *Wumpus*, etc.
- Predicate symbols (attributes/relations): These look like functions that give **truth values**.
  - Single argument: *Student*(*John*), *Red*(*Wumpus*), etc.
  - Multiple arguments: ≥(2,1), *Classmate*(*John*,*Mary*), *Product*(3,5,15), *Inside*(*John*, *ED117*), etc.
- Function symbols (functions): These give **objects** defined according to their relations to other objects: *Mother*(*Mary*), *Sqrt*(100), *LeftLeg*(*John*), *Sum*(3,5), etc.

Models in FOL

A model in FOL includes the following:

- **Domain**: A non-empty set of objects.
- **Relations** among the objects: A relation is defined by the set of all the **tuples** of objects that are related.
  - Example: Let the relation *Classmate* include the tuples ⟨*John*,*Mary*⟩ and ⟨*Mary*,*John*⟩.
    - Given the relation, the predicates *Classmate*(*John*,*Mary*) and *Classmate*(*Mary*,*John*) are true.
  - Unary relation: All the objects that satisfy a certain attribute. For example, the relation *Student* includes ⟨*John*⟩ and ⟨*Mary*⟩, but not ⟨*John's dog*⟩.
    - Given the relation, the predicates *Student*(*John*) and *Student*(*Mary*) are true, but the predicate *Student*(*John's dog*) is false.

Models in FOL

A model in FOL includes the following:

- When a relation can be made into a **function**: In this relation, an object can only be related to only one other particular object.
  - Example: Let the relation *Head* include the tuples ⟨*John, John's head*⟩ and ⟨*Mary, Mary's head*⟩.
    - Given the relation, we can have a function *Head*, where *Head*(*John*) is John's head and *Head*(*Mary*) is Mary's head.
  - Example: The relation *Classmate* can not become a function.
- **Interpretation**: This links the symbols with the actual objects and relations in the model.

Models in FOL

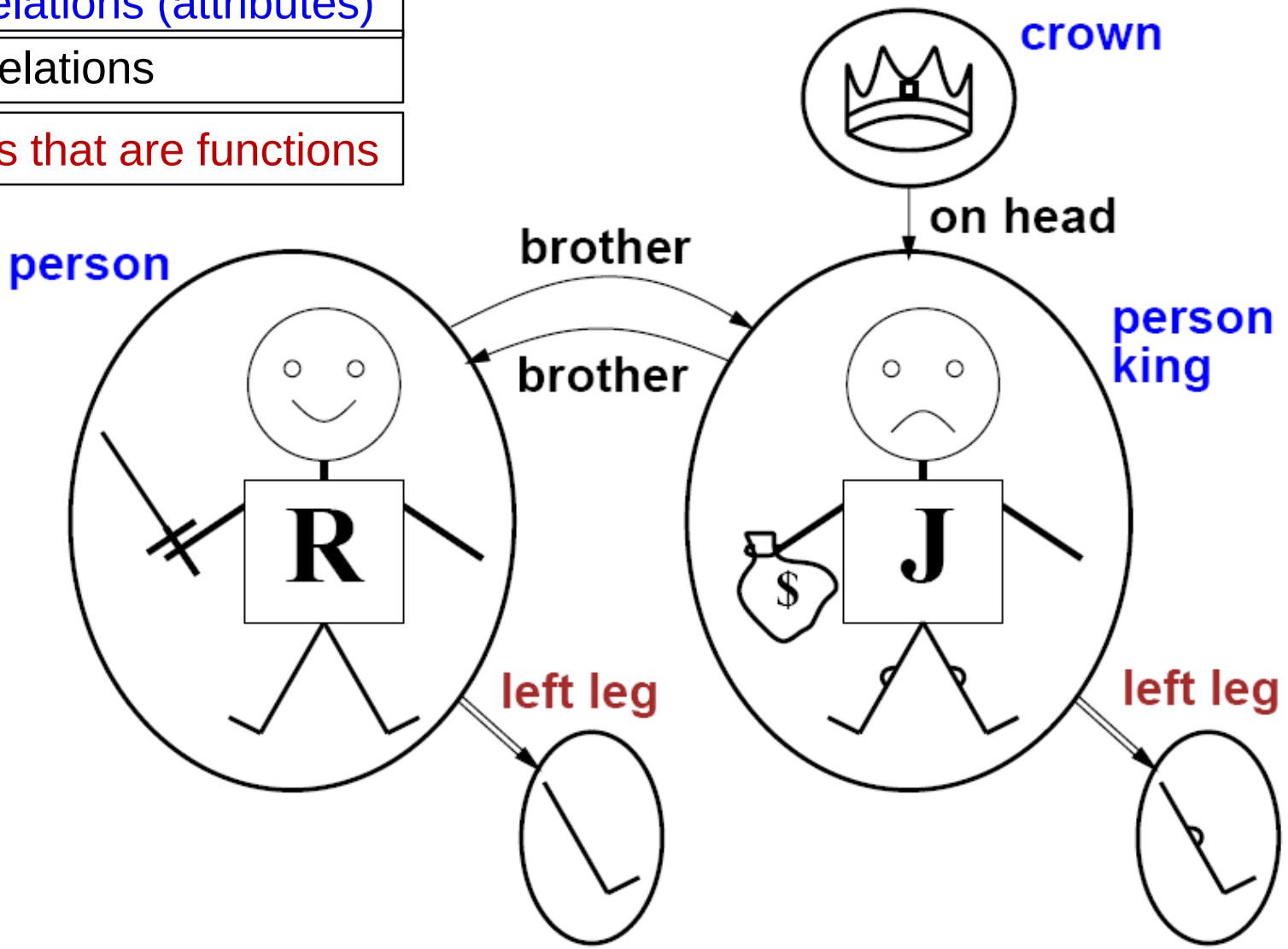
More about interpretation:

- There are multiple possible interpretations for linking the objects/relations and symbols. The truth value of a sentence depends on the interpretation.
- Example:
  - *Classmate*(*John*,*Mary*) is **true** if *John* refers to John the student and *Mary* refers to Mary the student.
  - *Classmate*(*John*,*Mary*) is **false** if *John* refers to John the student and *Mary* refers to John's cat.
- The actual object referred to by a term is called the term's referent.

Models in FOL

An example model in the textbook:

unary relations (attributes)
binary relations
relations that are functions



Syntax of FOL

$Sentence \rightarrow AtomicSentence \mid ComplexSentence$

$AtomicSentence \rightarrow Predicate \mid Predicate(Term,\ldots) \mid Term = Term$

$ComplexSentence \rightarrow (Sentence) \mid [Sentence] \mid \neg Sentence$

$\mid Sentence \wedge Sentence \quad \mid Sentence \vee Sentence$

$\mid Sentence \Rightarrow Sentence \quad \mid Sentence \Leftrightarrow Sentence$

$\mid Quantifier\ Variable,\ldots\ Sentence$

Each sentence has a truth value.

$Term \rightarrow Function(Term,\ldots) \mid Constant \mid Variable$

$Quantifier \rightarrow \forall \mid \exists$

$Constant \rightarrow A \mid X_1 \mid John \mid \ldots$

$Variable \rightarrow a \mid x \mid s \mid \ldots$

$Predicate \rightarrow True \mid False \mid After \mid Loves \mid Raining \mid \ldots$

$Function \rightarrow Mother \mid LeftLeg \mid \ldots$

Operator Precedence: $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Universal Quantification

- $\forall x\ P$
  - True iff P is true for x being every possible object in the model.
- Example: Everyone in NCTU is smart.
  - $\forall x\ AtNCTU(x) \Rightarrow Smart(x)$
  - Any problem with this? Consider the domain of the model.
- Example: Every student in NCTU is smart.
  - $\forall x\ AtNCTU(x) \wedge Student(x) \Rightarrow Smart(x)$
- What is the meaning of the following?
  - $\forall x\ AtNCTU(x) \wedge Student(x) \wedge Smart(x)$

Existential Quantification

- $\exists x\ P$
  - True iff P is true for x being some possible object in the model.
- Example: Someone in NTHU is smart.
  - $\exists x\ AtNTHU(x) \wedge Smart(x)$
- What is the problem with the following?
  - $\exists x\ AtNTHU(x) \Rightarrow Smart(x)$
  - Normally we do not use implication as the main connective with existential quantification.

Quantifier Duality

De Morgan's rules:

- $\exists x \neg P \equiv \neg \forall x P$
- $\exists x P \equiv \neg \forall x \neg P$
- $\forall x \neg P \equiv \neg \exists x P$
- $\forall x P \equiv \neg \exists x \neg P$

Examples:

- $\forall x \, Likes(x, IceCream) \equiv \neg \exists x \neg Likes(x, IceCream)$
- $\exists x \, Likes(x, Broccoli) \equiv \neg \forall x \neg Likes(x, Broccoli)$

Nesting Quantifiers

Try to link the sentences with their meanings:

- $\forall x\ \forall y\ Loves(x,y)$
- $\exists x\ \exists y\ Loves(x,y)$
- $\exists x\ \forall y\ Loves(x,y)$
- $\forall x\ \exists y\ Loves(x,y)$
- $\forall y\ \exists x\ Loves(x,y)$
- $\exists y\ \forall x\ Loves(x,y)$

?

Everyone loves someone.

Someone loves everyone.

Everyone is loved by someone.

Someone is loved by everyone.

Someone is loved by someone.

Everyone loves everyone.

Notes:

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is **not** the same as $\forall y\ \exists x$

Fun with FOL Sentences

Brothers are siblings

$$\forall x\ \forall y\ Brother(x, y) \Rightarrow Sibling(x, y)$$

"Sibling" is symmetric

$$\forall x\ \forall y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$$

One's mother is one's female parent

$$\forall x\ \forall y\ Mother(x, y) \Leftrightarrow Female(x) \wedge Parent(x, y)$$

A first cousin is a child of a parent's sibling

$$\forall x\ \forall y\ [\ FirstCousin(x, y) \Leftrightarrow$$

$$\exists p,q\ Parent(p, x) \wedge Parent(q, y) \wedge Sibling(p, q)\ ]$$

Equality in FOL

- The equality symbol (=) means that two terms refer to the same object.
- Example: "John has at least two brothers"
  - $\exists x\, \exists y\, Brother(x,John) \wedge Brother(y,John) \wedge \neg(x=y)$
  - Not sufficient: $\exists x\, \exists y\, Brother(x,John) \wedge Brother(y,John)$
- Example: definition of (full) sibling

$$\forall x\, \forall y\, FullSibling(x, y) \quad \Leftrightarrow$$

$$\neg(x=y) \wedge$$

$$[\, \exists m, f \quad \neg(m=f) \wedge Parent(m, x) \wedge Parent(m, y) \wedge Parent(f, x) \wedge Parent(f, y)\,]$$

Definitions, Axioms, Theorems

- We have used sentences in the following form to "define" new predicates:
  - $\forall x,\ldots\ P(x,\ldots) \Leftrightarrow \ldots$
  - This way we can enrich the representation from a basic set of predicates.
- Example: From a basic set of *Child*, *Female*, and *Spouse*, we can define other relations in the kinship domain: *Parent*, *Male*, etc.
- Axioms: Sentences in the KB that are not entailed by other sentences. Axioms include definitions.
- Theorems: Useful facts that are entailed by the KB.
  - Example: $\forall x\ \forall y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$

Exercises: Writing FOL Sentences

Starting from the basic set of three predicates *Male*, *Parent*, and *Spouse*, give definitions of these predicates:

- *Female*: $\text{Female}(x) \Leftrightarrow \neg\text{Male}(x)$
- *Child*:
- *Father*:
- *Sister*:
- *Grandfather*:
- *Uncle*: $\text{Uncle}(x,y) \Leftrightarrow \exists z\ \text{Brother}(x,z) \wedge \text{Parent}(z,y)$
- *Niece*:

Exercises: Writing FOL Sentences

- John has exactly two brothers.

$$\exists x,y\ \text{Brother}(\text{John},x) \wedge \text{Brother}(\text{John},y) \wedge (x \neq y) \wedge [\forall z\ \text{Brother}(\text{John},z) \Rightarrow (x=z) \vee (y=z)]$$

- Each of John's classmates has at least one sibling.

$$\forall x\ \text{Classmate}(\text{John},x) \Rightarrow [\exists y\ \text{Sibling}(y,x)]$$

- One of Mary's classmates is John's brother.

$$\exists x\ \text{Classmate}(\text{Mary},x) \wedge \text{Brother}(\text{John},x)$$

Assertions and Queries for a FOL KB

- Assertions: Sentences added to the KB. Examples:
  - $Tell(KB, King(John))$
  - $Tell(KB, Person(Richard))$
  - $Tell(KB, \forall x\ King(x) \Rightarrow Person(x))$
- Queries (inference required here)
  - $Ask(KB, King(John))$
  - $Ask(KB, Person(John))$
  - $Ask(KB, \exists x\ Person(x))$
  - $AskVars(KB, Person(x))$

Which x in the domain makes the sentence $Person(x)$ true?

- **Substitution** (binding list): The response to *AskVars* that lists the possible variable assignments that make the query true.
  - Example for the last query: $\{x/John\}$ and $\{x/Richard\}$

Inference in FOL

- Difference with propositional logic: variables
  - Symbols (sentences) in FOL that do not involve variables can be treated as propositional symbols. Example: $Student(John)$, $Red(Wumpus)$, etc.
- How do we handle variables (and quantifiers)?
  - **Propositionalization**
  - **Unification**

Universal Instantiation (UI)

- Instantiation means to replace a variable in a sentence with a ground term (a term with no variables).
- Every instantiation of a universally quantified sentence is entailed by it:

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

α: sentence
v: variable
g: ground term

$\text{Subst}(\theta,\alpha)$ is the substituted version of α by θ.

Example: We can infer $Likes(John, IceCream)$

from $\forall x \ Likes(x, IceCream)$

with the substitution $\{x/John\}$

Universal Instantiation (UI)

- Universal instantiation can be applied multiple times to add new sentences to the KB.
  - There are infinitely many possible new sentences if the KB contains functions (see example below).
- Example:

  The sentence $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ entails

  $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$
  etc.

Existential Instantiation (EI)

- A new (previously unused) constant symbol is used to represent an instance:

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

α: sentence
v: variable
k: new symbol

- This new constant is called a Skolem constant.
- Existential instantiation can be applied once to replace a sentence with existential quantification.
- Example:

The sentence $\exists x \ King(x) \wedge Greedy(x)$
becomes $King(C_1) \wedge Greedy(C_1)$

The sentence $\exists x \ Crown(x) \wedge OnHead(x, John)$
becomes $Crown(W_2) \wedge OnHead(W_2, John)$

Propositionalization

- Idea: Convert all the sentences in a FOL KB and the query to ground sentences (no variables).
- We can then apply inference rules in propositional logic to get our answers.
- Example: Consider the KB with the following sentences:

$$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

Propositionalization

- For the universally quantified sentence, we need to generate corresponding ground sentences using all the possible ground terms:

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$

- Problem: When there is a function symbol, there are an infinite number of ground terms. Example:

$$Father(John)$$
$$Father(Father(John))$$
$$Father(Father(Father(John)))$$

etc.

Propositionalization

- Herbrand's theorem (1930): If a sentence is entailed by an FOL KB, it is entailed by a finite subset of the propositional KB.
- Solution: We can iteratively increase the depth of nested function terms until the entailment is proved.
- Problem: If the sentence is not entailed, the process can go on forever.
- Inference in FOL via propositionalization is complete (all entailed sentences can be found).
- Entailment in FOL is semidecidable if the KB contains functions; non-entailment can not be proved.

Inefficiency of Propositionalization

- Propositionalization can generate many irrelevant sentences. Example: The goal is to query the KB about

  $Evil(John)$

- However, the following is also generated, which is irrelevant:

  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

  etc.

Unification

- This is the process of making two sentences identical through substitution.
- Example: *Greedy*(*John*) and *Greedy*(*x*) are unified by the substitution $\{x/John\}$.
- **Unifier**: $\text{Unify}(p, q) = \theta \quad \text{where} \quad \text{Subst}(\theta, p) = \text{Subst}(\theta, q)$

p	q	θ
Knows(*John*, *x*)	*Knows*(*John*, *Jane*)	$\{x/Jane\}$
Knows(*John*, *x*)	*Knows*(*y*, *Jane*)	$\{x/Jane, y/John\}$
Knows(*John*, *x*)	*Knows*(*y*, *Mother*(*y*))	$\{x/Mother(John), y/John\}$
Knows(*John*, *x*)	*Knows*(*x*, *Jane*)	???

The last case requires "standardizing apart": Renaming variables to avoid name clashes.

Generalized Modus Ponens

- Example: Consider the sentence

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

If we know $King(John)$, then the substitution $\{x/John,\ y/John\}$ makes the premise of the implication true.

$\Rightarrow$ The consequence of the implication ($Evil(John)$) is true.

- **Generalized Modus Ponens:** If there is a substitution θ such that $\text{Subst}(\theta, p_i') = \text{Subst}(\theta, p_i)$ for all i

then we have $$\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{Subst}(\theta, q)}$$

- Note: Variables with no quantification are assumed to be universally quantified.

Example KB

Given: The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Colonel West is a criminal

Example KB in FOL Definite Clauses

… it is a crime for an American to sell weapons to hostile nations:

R1: $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$

Nono …has some missiles

$\exists x\ Owns(Nono, x) \land Missile(x)$

(**EI**)→ R2: $Owns(Nono, M_1)$ and R3: $Missile(M_1)$

… all of its missiles were sold to it by Colonel West

R4: $Owns(Nono, x) \land Missile(x) \Rightarrow Sells(West\ ,\ x\ ,\ Nono)$

West, who is American

R5: $American(West)$

The country Nono, an enemy of America

R6: $Enemy(Nono, America)$

Missiles are weapons:

R7: $Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile"

R8: $Enemy(x, America) \Rightarrow Hostile(x)$

Forward Chaining for FOL

- New sentences are iteratively added to the KB.
- Termination:
  - Success: A sentence unifies with the query (the unifier is returned).
  - Failure: No new sentence can be generated.
  - Can loop forever if the KB contains functions (FOL entailment with definite clauses is semidecidable).
- In each iteration, every KB rule is checked:
  - If GMP can be applied to the rule (i.e., there exists some substitution that makes the premises true), then add the consequence to the KB if it is new (i.e., it does not unify with any existing sentence).

Forward Chaining Example

Iteration #1:

- Unify(R2∧R3, premise(R4)) =

  → R9: *Sells*(*West*, M_1, *Nono*)

- Unify(R3, premise(R7)) =

  → R10: *Weapon*(M_1)

- Unify(R6, premise(R8)) =

  → R11: *Hostile*(*Nono*)

Iteration #2:

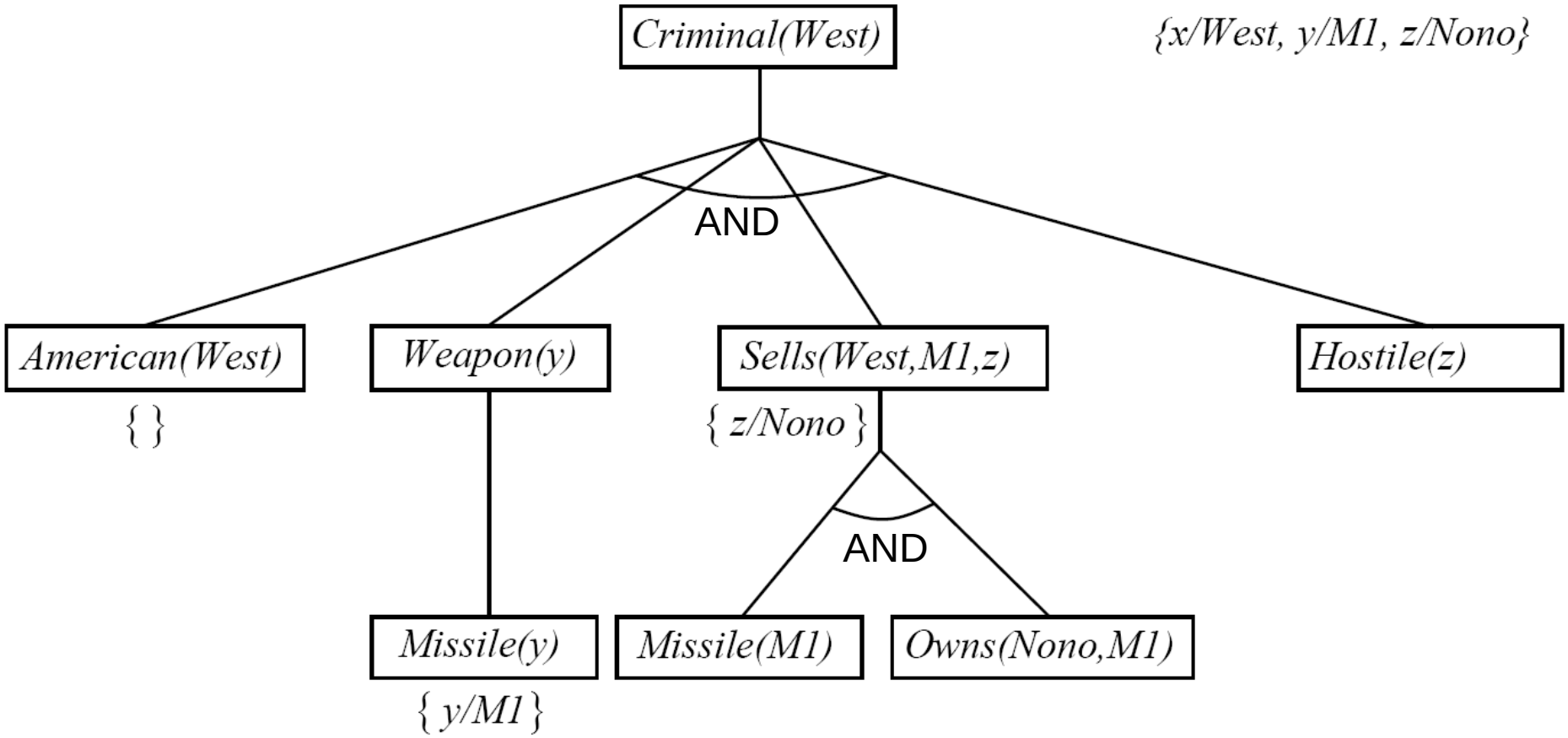
- Unify(R5∧R10∧R9∧R11, premise(R1)) =

  → R12: *Criminal*(*West*)

Backward Chaining for FOL

- Depth-first search starting from the goal.
  - OR-step: Find all the rules whose consequences unify with the goal.
  - AND-step: For such a (potentially useful) rule, try to prove each conjunct of its premise as a sub-goal.
- Each line of search terminates if it unifies with a known fact.
- The required substitution is kept tracked of during the search.
- Difficulties: repeated states, incompleteness.

Backward Chaining Example



FOL Inference with Resolution

$$\frac{l_1 \vee \ldots \vee l_k, \quad m_1 \vee \ldots \vee m_n}{\text{Subst}(\theta, \; l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)}$$

if θ unifies l_i and $\neg m_j$.

Example: $$\frac{\neg Rich(x) \vee Unhappy(x), \quad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Convert FOL Sentences to CNF

- Example sentence:

$$\forall x\,[\forall y\,Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\,Loves(y,x)]$$

- Eliminate biconditionals and implications

$$\forall x\,[\forall y\,\neg Animal(y) \vee Loves(x,y)] \Rightarrow [\exists y\,Loves(y,x)]$$
$$\forall x\,[\neg\forall y\,\neg Animal(y) \vee Loves(x,y)] \vee [\exists y\,Loves(y,x)]$$

- Move negations inward (de Morgan's)

$$\forall x\,[\exists y\ \ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\,Loves(y,x)]$$
$$\forall x\,[\exists y\,Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\,Loves(y,x)]$$

- Standardize variables (different for each quantifier)

$$\forall x\,[\exists y\,Animal(y) \wedge \neg Loves(x,y)] \vee [\exists \mathbf{z}\,Loves(\mathbf{z},x)]$$

Convert FOL Sentences to CNF

- Skolemize: (more general than existential instantiation) replace each existentially quantified variable with a **Skolem function** of all the enclosing universally quantified variables

  $\forall x\,[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

  (Use regular EI for existentially quantified variables that are not enclosed by universal quantification.)

- Drop universal quantifiers

  $[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

- Distribute $\wedge$ over $\vee$

$[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$

Resolution Example (Criminal KB)

KB in CNF:

R1: $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$

R2: $Owns(Nono, M_1)$

R3: $Missile(M_1)$

R4: $\neg Owns(Nono, x) \vee \neg Missile(x) \vee Sells(West, x, Nono)$

R5: $American(West)$

R6: $Enemy(Nono, America)$

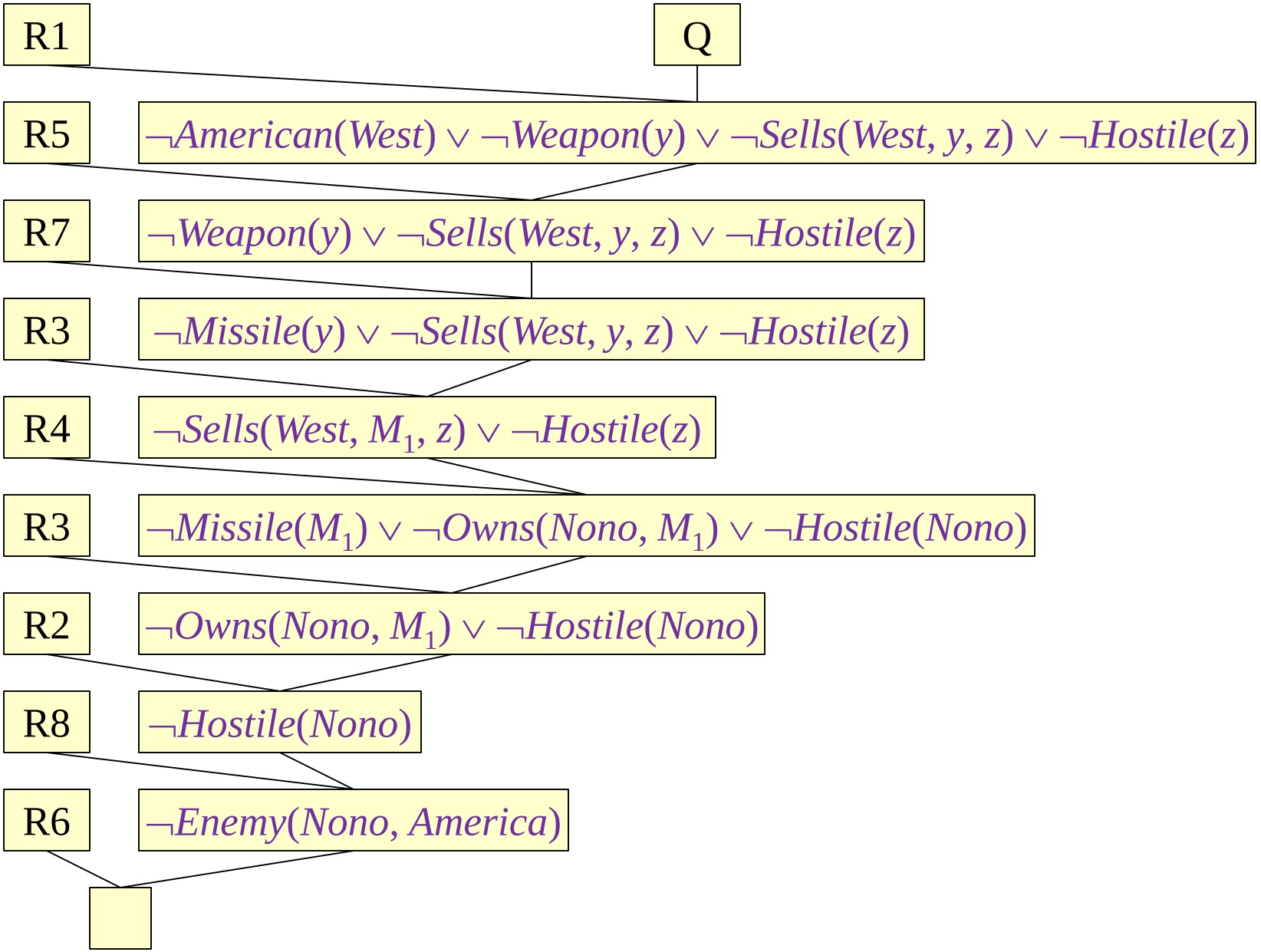
R7: $\neg Missile(x) \vee Weapon(x)$

R8: $\neg Enemy(x, America) \vee Hostile(x)$

Query: Q: $Criminal(West)$

Goal: To prove that $(KB \wedge \neg Q)$ is invalid (always false)

Resolution Example (Criminal KB)



What Next?

- Second-order logic: "Quantification over predicates (sets of objects)" and "Predicates defined on predicates", and some more.
- Many other different types of logics.
- Automated theorem proving:
  - Based on FOL (mostly).
  - Can have human-in-the-loop.
  - Many available systems today.
  - Have found proofs for some open mathematical problems.
  - Hardware and software verifications are the most practically important applications.