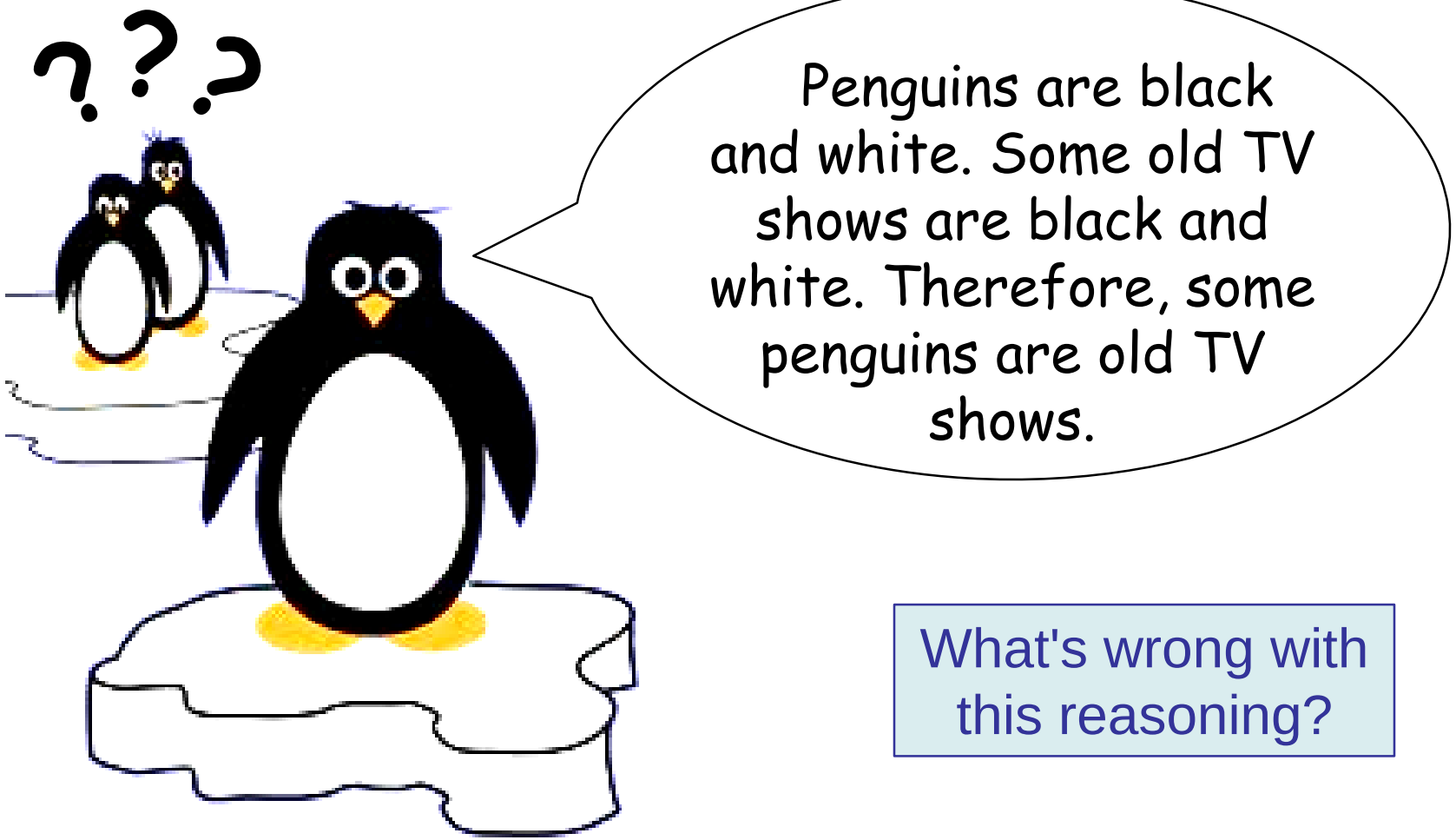
Logical Inference



What's wrong with this reasoning?

Logical Agents

- A logical agent: An agent that can represent knowledge of its world, its goals and the current situation by sentences in logic and decide what to do by inferring that a certain action or course of action is appropriate to achieve its goals.
- A logical agent consists of the following two components:
  - **Knowledge Base** (KB): Domain-specific information (represented as **logical sentences**)
  - **Inference Engine**: Domain-independent rules/procedures for deriving new sentences from the KB.
- The KB can be updated to add new information or adapt to environment changes.

Entailment and Inference

- **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.
- **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: Semantics

- A **model** in propositional logic is specified by the truth values for all the propositional symbols.
- For example, $\{P$=true, Q=false$\}$ is a model that makes PvQ true.

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*

Logical Theorem Proving

- 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: $P \vee \neg P$, $P \Rightarrow P$
- A sentence is **satisfiable** iff it is true in some models.
  - Examples: P, $P \wedge Q$
- A sentence is **unsatisfiable** iff it is false in all models.
  - Examples: $P \wedge \neg P$

Proof by Search

A search algorithm can be used to find the proof from an initial KB to a certain sentence:

- 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.
- Solution: The path.

Inference by Resolution

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

$$\frac{l_1 \vee \ldots \vee l_k, \quad m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$ (m is the negation of l_i)

(general form)

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

(m_j is the negation of l_i)

(easier to understand)

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

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

Resolution Algorithm

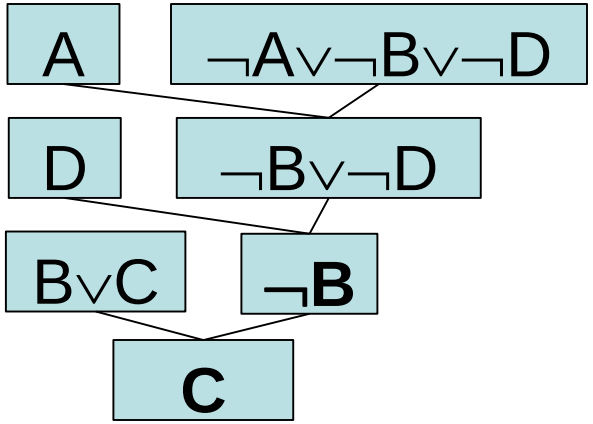
To prove that the KB entails a sentence α:

- 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

A small minesweeper example, with 4 unknown symbols:

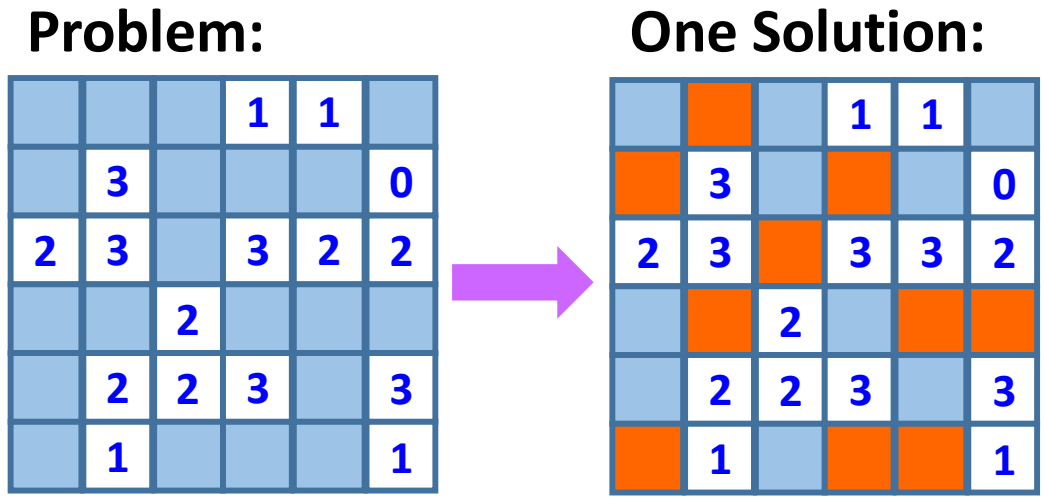
- From the basic rule, we know A and D are true..
- Regarding B and C, here are some sentences and corresponding CNFs:
  - $A \wedge D \Leftrightarrow \neg B$ $(\neg A \vee \neg B \vee \neg D,\quad A \vee B,\quad B \vee D)$
  - $C \Leftrightarrow \neg B$ $(\neg B \vee \neg C,\quad B \vee C)$
- Proof:



SAT Problems

- Given a set of logical variables and a sentence, the SAT (satisfiability) problem concerns finding a set of assignments so that the sentence evaluates to **True**.
- This is a special type of CSPs (Constraint Satisfaction Problems).
- Solving SAT problems corresponds to "model checking": Finding models that satisfy the KB.

An example SAT problem based on Minesweeper:



SAT with Backtracking Search

- The representative algorithm of this approach is **DPLL** (Davis-Putnam-Logemann-Loveland).
- The KB consists of CNFs.
- A depth-first search (DFS) based procedure, with each step being a truth-value assignment to a symbol.
- Several specialized techniques/heuristics are used for acceleration.

SAT with Local Search

- Any local search metaheuristic can be applied.
- Local search for SAT is semi-decidable (unable to prove that no solution exists), but is empirically more efficient than complete search algorithms if a solution exists.
- Search starts from a complete set of random assignments.
- **WalkSAT** algorithm:
  - In each step, choose a clause that is currently false.
  - Do one of the following with a probability:
    - Flip one symbol in the clause to maximize the number of satisfied clauses (min-conflicts).
    - Flip one symbol in the clause randomly.

Propositional vs. First-Order Logic

- Propositional logic is based only on simple facts about the world, such as “John is a Student”.
- FOL is concerned with objects. Facts represent "attributes of objects" and "relations among objects".
- Example: "x is a student", "x is in NYCU", and "x is smart" are all attributes of x, which can represent any object.
  - The sentence "Every student in NYCU is smart" can be written in this logical expression

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

Representations in FOL

- Constant symbols (objects): *John*, *Mary*, *EC122*, etc.
- Predicate symbols (attributes/relations): These look like functions that give **truth values**.
  - Single argument: *IsStudent*(*John*), *Tall*(*Mary*), etc.
  - Multiple arguments: ≥(2,1), *Classmate*(*John*,*Mary*), *Product*(3,5,15), *Inside*(*John*, *EC122*), 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.

Syntax of FOL

Each sentence has a truth value.

Sentence → *AtomicSentence* | *ComplexSentence*

AtomicSentence → *Predicate* | *Predicate*(*Term*,…) | *Term*=*Term*

ComplexSentence → (*Sentence*) | [*Sentence*] | ¬*Sentence*

| *Sentence* ∧ *Sentence* | *Sentence* ∨ *Sentence*

| *Sentence* ⇒ *Sentence* | *Sentence* ⇔ *Sentence*

| *Quantifier Variable*,... *Sentence*

Term → *Function*(*Term*,…) | *Constant* | *Variable*

Quantifier → ∀ | ∃

Constant → *A* | X_1 | *John* | ...

Variable → *a* | *x* | *s* | ...

Predicate → *True* | *False* | *After* | *Loves* | *Raining* | ...

Function → *Mother* | *LeftLeg* | ...

Operator Precedence: ¬, =, ∧, ∨, ⇒, ⇔

Universal Quantification

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

Existential Quantification

- $\exists x\ P$
  - True iff P is true for x being some possible object in the model.
- Example: Someone in NYCU is smart.
  - $\exists x\ AtNYCU(x) \wedge Smart(x)$
- What is the problem with the following?
  - $\exists x\ AtNYCU(x) \Rightarrow Smart(x)$
  - 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:

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

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 (a definition)

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

- John has at least two brothers

$$\exists x\ \exists y\ Brother(x,John) \wedge Brother(y,John) \wedge \neg(x=y)$$

The equality symbol (=) means that two terms refer to the same object.

Inference in FOL

- Difference with propositional logic: variables
  - Symbols (sentences) in FOL that do not involve variables can be treated as propositional symbols. Example: $IsStudent(John)$, etc.
- Handle variables and quantifiers?
  - **Universal Instantiation**
  - **Existential Instantiation**
  - **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

$\text{Subst}(\{v/g\},\alpha)$ is the version of α with variable v replaced by g.

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

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

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

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.

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
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Jane)$	$\{x/Jane, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{x/Mother(John), y/John\}$

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.

FOL Inference with Resolution

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

if θ unifies l_i and $\neg m_j$.

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

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

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

Resolution Example (Criminal KB)

KB in CNF (procedures to convert the sentences to CNF skipped):

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: $Enemy(Nono, America)$

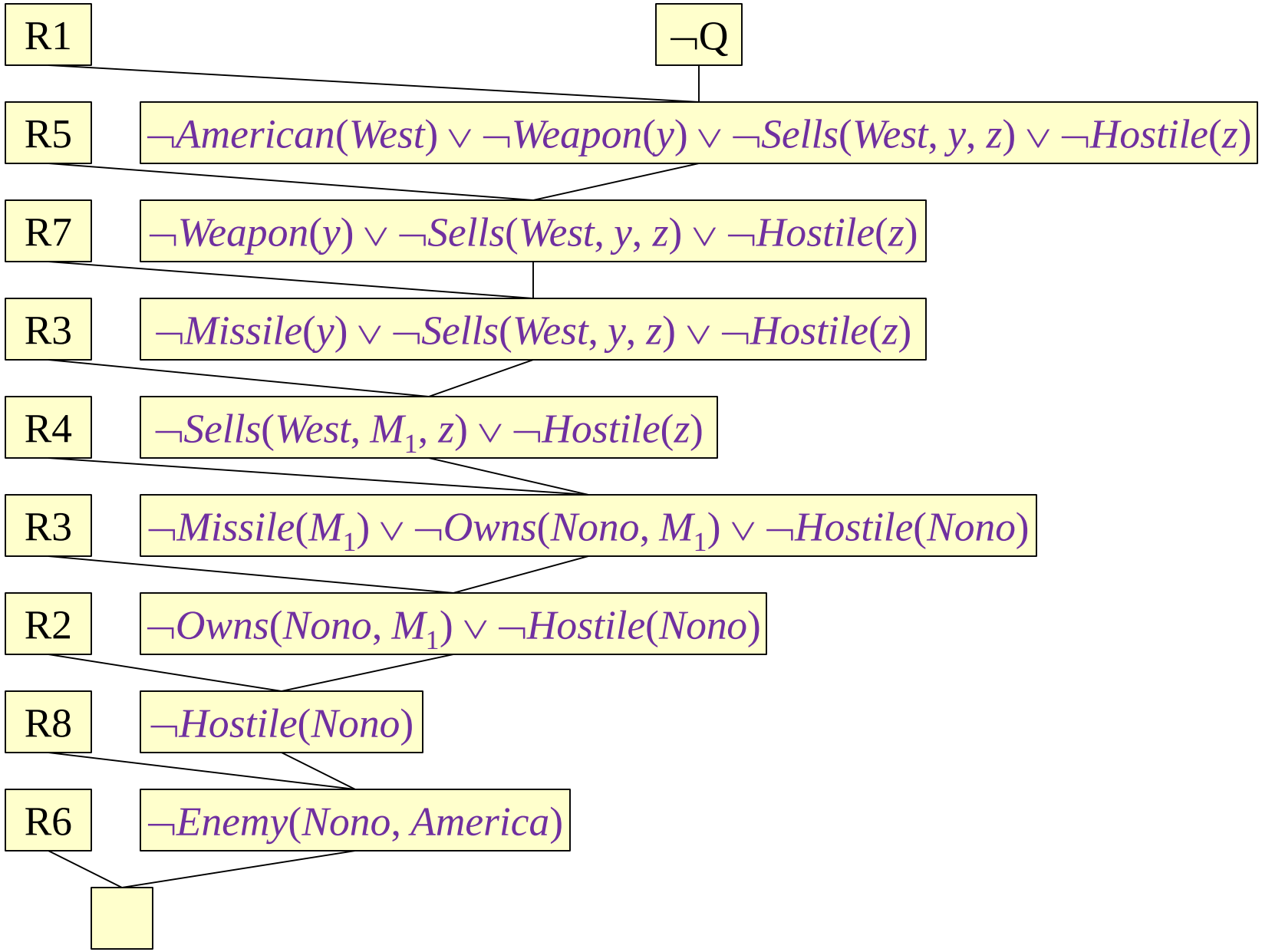
R7: $\neg Missile(x) \vee Weapon(x)$

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).
  - Resolution is the most widely used engine.
  - 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.

Logical Programming

- A snapshot of **Prolog**:

```
mother_child(trude, sally).
father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).
sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).
parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).

?- sibling(sally, erica).
Yes
```

- The use of Prolog in real-world applications has been quite limited. Problems include limits of the language itself, efficiency, and portability.
- Its usefulness has been demonstrated in some specialized fields. An example is that Prolog is part of IBM Watson machine that won the *Jeopardy!* game in 2011.