# Logic

School of EECS
Washington State University

# Knowledge-based Agent

- Knowledge Base (KB) and Knowledge Engineering
  - TELL agent about the environment
- Knowledge representation and logic formalisms
  - **Propositional logic**
  - **First-order logic**
  - Temporal logics, modal logics (esp. in DAI context!)
  - Many others…

- Reasoning via logical inference
  - ASK agent how to achieve goal based on current knowledge about the task and the world

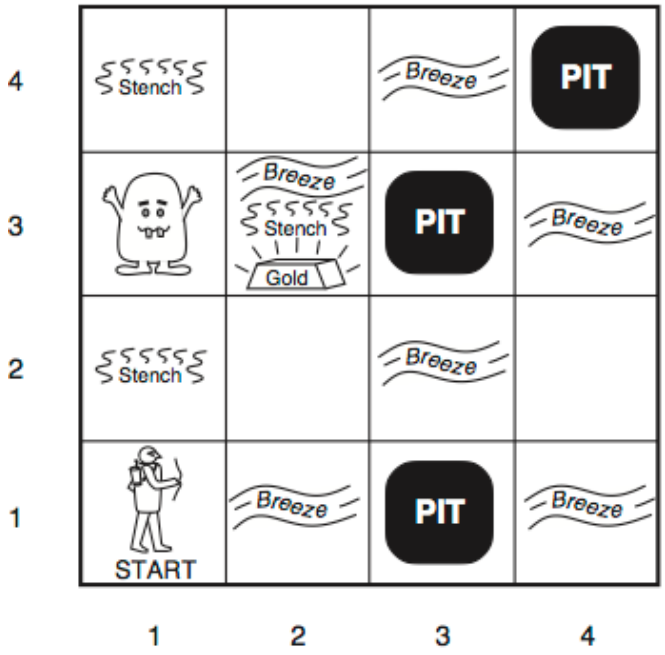
# Knowledge-based Agent

**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*))
  $t \leftarrow t + 1$
  **return** *action*

# Wumpus World (PEAS)

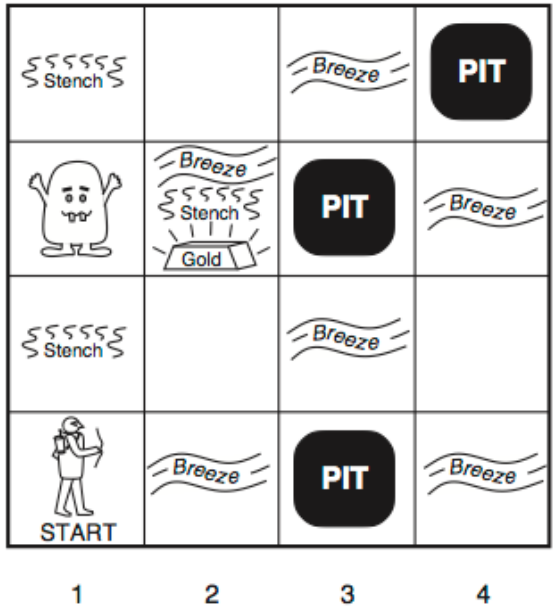
- Performance measure
  - +1000 for leaving cave with gold
  - −1000 for falling in pit or being eaten by Wumpus
  - −1 for each action taken

    (here, action = move or turn)
  - −10 for using the arrow
  - Game ends when agent either gets killed or leaves the cave



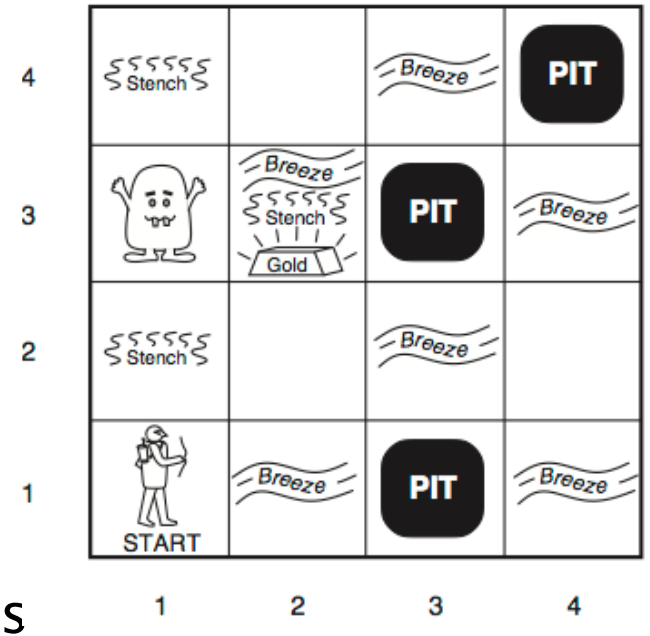
# Wumpus World (PEAS)

- **Environment**
  - 4x4 grid of rooms
  - Agent starts in square [1,1] facing right
  - Location of Wumpus and gold chosen at random
    - any other than [1,1]
  - Each square other than [1,1] has a 0.2 probability of containing a pit
  - What if… (location of pits vs. location of gold/Wumpus)



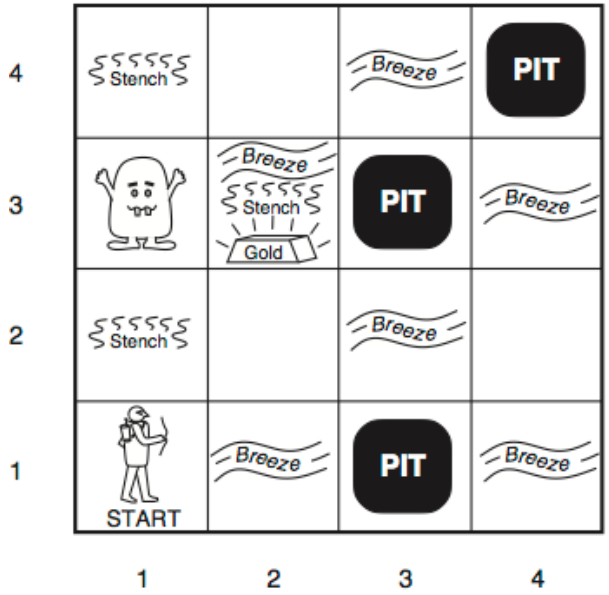
# Wumpus World (PEAS)

- Actuators
  - Forward
  - TurnLeft by 90°
  - TurnRight by 90°
  - Grab picks up gold if agent in gold location
  - Shoot shoots arrow in direction agent is facing
    - Arrow continues straight until hits Wumpus or wall
  - Climb leaves cave if agent in [1,1]



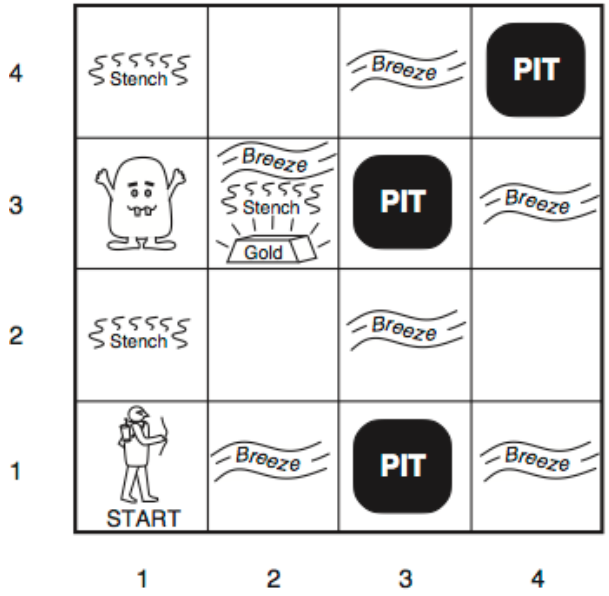
# Wumpus World (PEAS)

- Sensors (Boolean)
  - Stench if wumpus in directly (not diagonally) adjacent square
  - Breeze if pit in directly adjacent square
  - Glitter if gold in agent's current square
  - Bump if agent walks into a wall
  - Scream if wumpus is killed

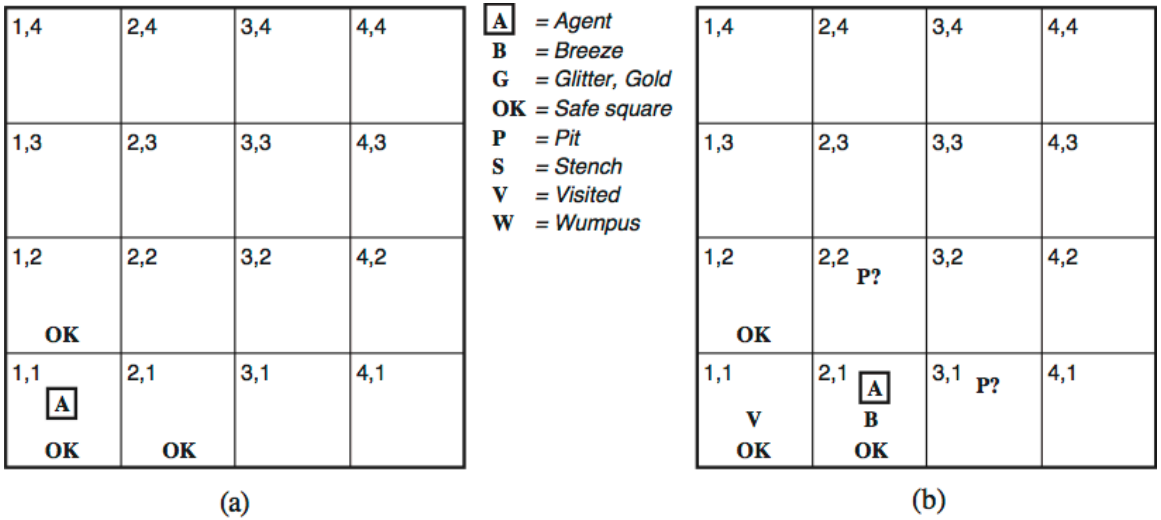


# Acting Logically in Wumpus World

- **Goals**
  - Visit safe locations
  - Grab gold if present
  - If have gold or no more safe spots / unvisited locations, then move to [1,1] and Climb out



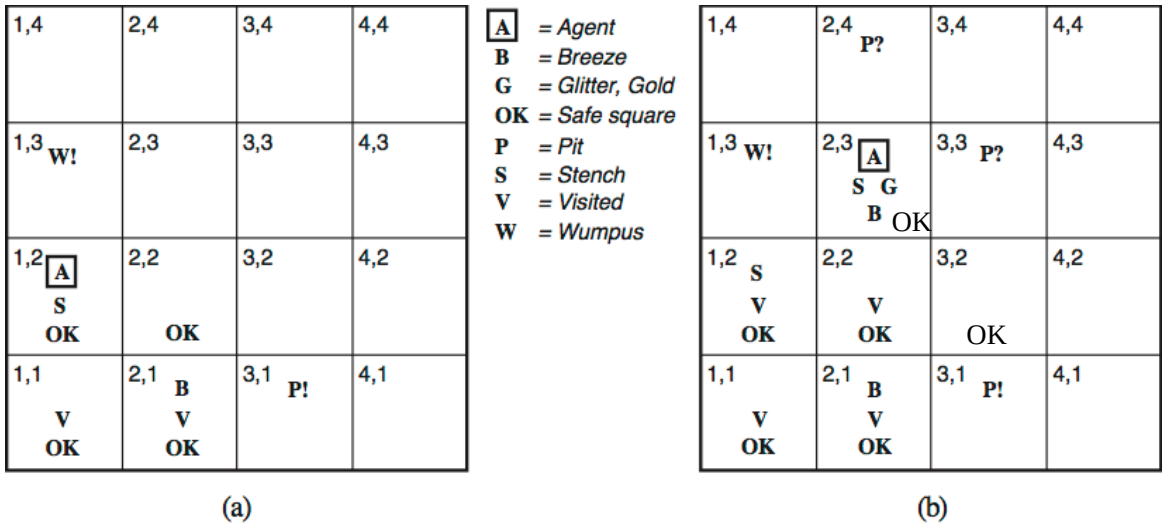
# Acting Logically in Wumpus World




(a) (b)

- $Percept_1$ = [None,None,None,None,None]
  - [1,2] and [2,1] safe
- Action = Forward
- $Percept_2$ = [None,Breeze,None,None,None]
- Either [2,2] or [3,1] or both has a pit
- Execute TurnLeft, TurnLeft, Forward, TurnRight, Forward

# Acting Logically in Wumpus World




- $Percept_7$ = [Stench,None,None,None,None]
  - Wumpus in [1,3]
  - No pit in [2,2] (safe), so pit in [3,1]
- Could Shoot, but <TurnRight,Forward> to [2,2]
- $Percept_9$ = [None,None,None,None,None]
  - [3,2] and [2,3] are safe
- <TurnLeft,Forward> to [2,3]
- $Percept_{11}$ = [Stench,Breeze,Glitter,None,None]
- Grab gold, head home, and Climb (score: 1000 – 17 = 983)

# Intro to Logic

- A knowledge base (KB) consists of "sentences"
- Syntax specifies a well-formed sentence
- Semantics specifies the meaning of a sentence
- A model **m** specifies whether each sentence is true or false
  - There can be many (possibly infinitely many) models
  - E.g., Model $m_1$ may say "wumpus in [2,3]" is true, $m_1$(wumpus in [2,3])
  - E.g., Model $m_2$ may say "wumpus in [2,3]" is false, or "wumpus not in [2,3]", $m_2$(wumpus not in [2,3])
  - Model $m_3$ may agree with $m_1$ on above but differ on smt. else
- **m(α)** says "**m** satisfies **α**" or "**m** is a model of **α**"

# Logical Entailment

- Entailment between sentences implies that one sentence follows logically from another
- $\alpha \models \beta$ means $\alpha$ entails $\beta$
  - Or, for every model in which $\alpha$ is true, $\beta$ is also true
  - $\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$

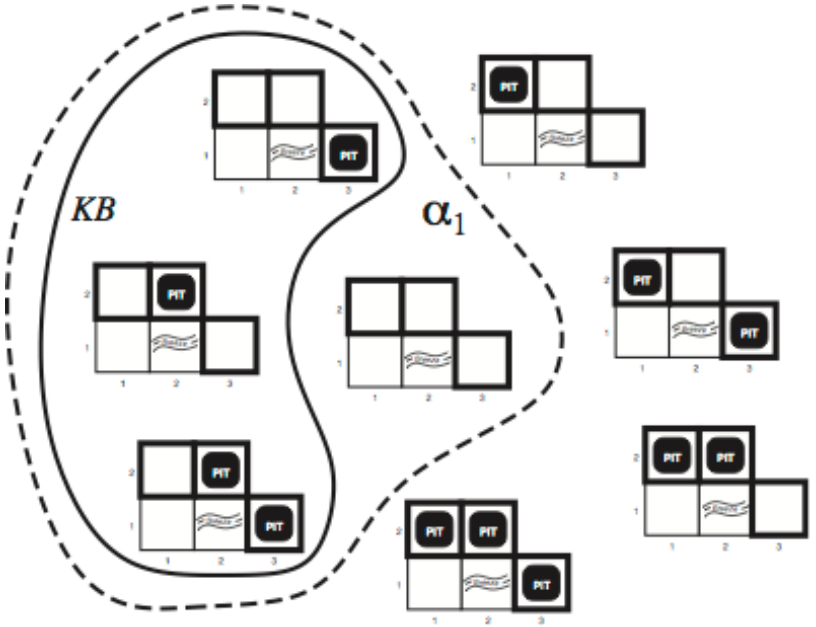
  where $M(\alpha)$ is the set of all models in which $\alpha$ is true
- Example: $Pit([3,1]) \models Breeze([2,1])$

# Logical Entailment (continued)

- E.g., KB = {¬Breeze([1,1]), Breeze([2,1]), *rules of wumpus world* }

- Pit([1,2]) ?, Pit([2,2] ?, Pit([3,1]) ?
- $\alpha_1$ = ¬Pit([1,2]), KB ⊨ $\alpha_1$, M(KB) ⊆ M($\alpha_1$)

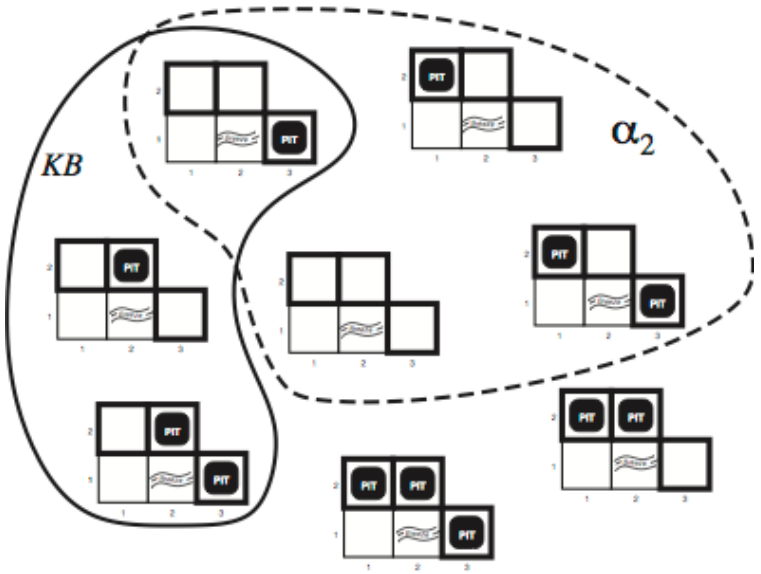
| | | | |
|---|---|---|---|
| 1,4 | 2,4 | 3,4 | 4,4 |
| 1,3 | 2,3 | 3,3 | 4,3 |
| 1,2 OK | 2,2 P? | 3,2 | 4,2 |
| 1,1 V OK | 2,1 A B OK | 3,1 P? | 4,1 |



# Logic: Entailed vs. Not Entailed

- E.g., KB = {¬Breeze([1,1]), Breeze([2,1]), *rules of wumpus world*}
- $\alpha_2$ = ¬Pit([2,2]), KB ⊭ $\alpha_2$, M(KB) ⊈ M($\alpha_2$)

| | | | |
|---|---|---|---|
| 1,4 | 2,4 | 3,4 | 4,4 |
| 1,3 | 2,3 | 3,3 | 4,3 |
| 1,2 OK | 2,2 P? | 3,2 | 4,2 |
| 1,1 V OK | 2,1 A B OK | 3,1 P? | 4,1 |

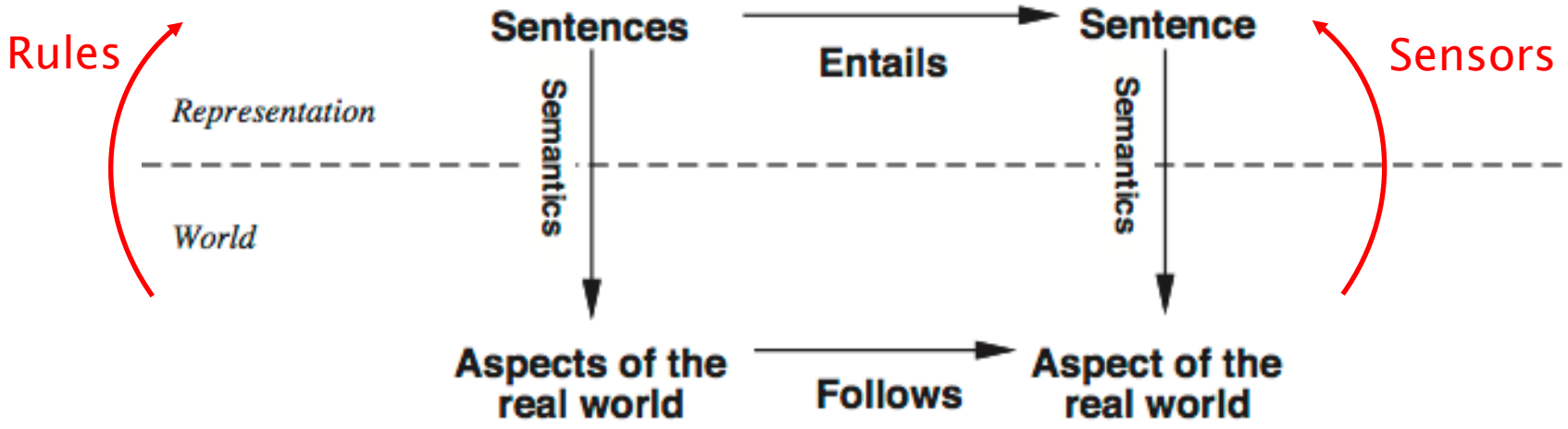


# Logical Inference

- Logical inference is the process by which we infer one sentence is true from others
- E.g., model checking (previous two slides)
  - Enumerate all possible models in which KB is true and check if α is also true
- KB $\vdash_i$ α means that α can be derived from KB using inference algorithm **i**
- Inference algorithm **i** is ***sound*** or truth-preserving if everything derived is also entailed
- Inference algorithm **i** is ***complete*** if it can derive everything that is entailed

# Logic in the Real World

- Grounding is the connection between logic and the "real world"
- From an engineering / practitioner standpoint, grounding (together w/ design & implementation complexity) is among most important aspect of AI models and theories



# Propositional Logic

- Syntax
  - Atomic sentences consist of a single propositional symbol, which can be true or false

Sentence → AtomicSentence | ComplexSentence
AtomicSentence → True | False | P | Q | R | North | $W_{1,3}$ | ...
ComplexSentence → (Sentence) | [Sentence]
| ¬ Sentence "not"
| Sentence ∧ Sentence "and"
| Sentence ∨ Sentence "or"
| Sentence ⇒ Sentence "implies"
| Sentence ⇔ Sentence "if and only if"

Operator Precedence: ¬, ∧, ∨, ⇒, ⇔

# Propositional Logic: Syntax

- Syntax
  - Not ($\neg$) is a negation
  - Literal is either an atomic sentence (positive literal) or a negated atomic sentence (negative literal)
  - And ($\wedge$) is a conjunction; its parts are conjuncts
  - Or ($\vee$) is a disjunction; its parts are disjuncts
  - Implies ($\Rightarrow$) is an implication
    - Its lefthand side is the antecedent or premise
    - Its righthand side is the consequent or conclusion
  - If and only if ($\Leftrightarrow$) is a biconditional

# Propositional Logic: Semantics

- **Semantics**
  - How to determine the truth value (true or false) of every proposition in a model
- True is true in every model
- False is false in every model
- Truth values of every other proposition must be specified directly in the model
  - E.g., $m_1 = \{W_{1,3}=\text{true}, P_{3,1}=\text{true}, P_{2,2}=\text{false}, \ldots\}$

# Propositional Logic

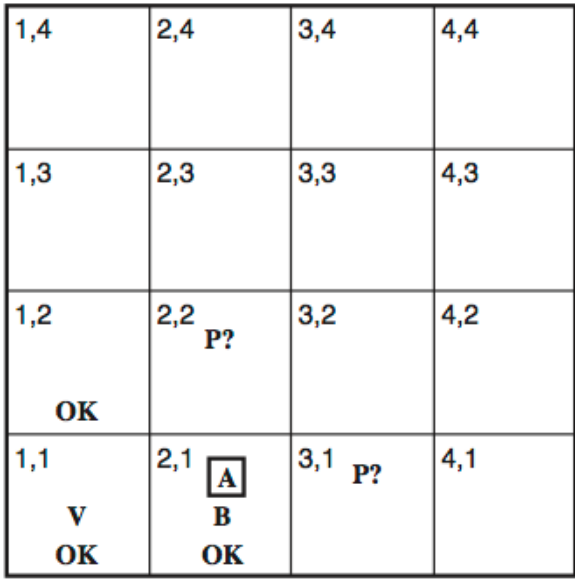
- Semantics for complex sentences in model **m**
  - ¬P is true iff P is false in m
  - P ∧ Q is true iff both P and Q are true in m
  - P ∨ Q is true iff either P or Q is true in m
  - P ⇒ Q is true unless P is true and Q is false in m
  - P ⇔ Q is true iff P and Q are both true or both false in m

Truth Table

| P | Q | ¬P | P ∧ Q | P ∨ Q | P ⇒ Q | P ⇔ 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 |

# Propositional Wumpus World KB

- $P_{x,y}$ is true if there is a pit in [x,y]
- $W_{x,y}$ is true if there is a wumpus in [x,y], alive or dead
- $B_{x,y}$ is true if the agent perceives a breeze in [x,y]
- $S_{x,y}$ is true if the agent perceives a stench in [x,y]




$R_1$: $\neg P_{1,1}$
$R_2$: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$R_3$: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
$R_4$: $\neg B_{1,1}$
$R_5$: $B_{2,1}$

# Simple Propositional Inference

**function** TT-ENTAILS? (*KB*, $\alpha$) **returns** *true* or *false*  // $KB \models \alpha$ ?
  *symbols* ← a list of the proposition symbols in *KB* and $\alpha$
  **return** TT-CHECK-ALL (*KB*, $\alpha$, *symbols*, {})

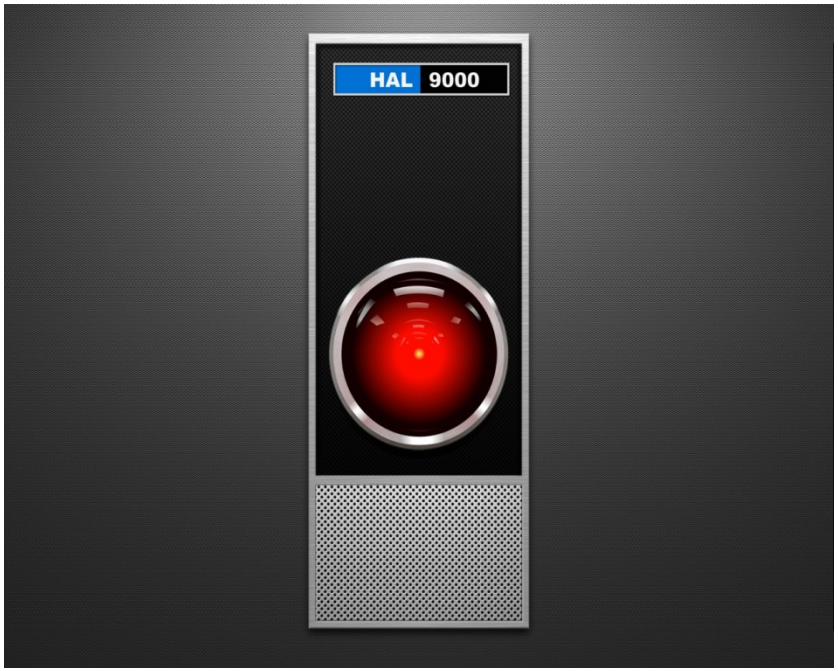
**function** TT-CHECK-ALL (*KB*, $\alpha$, *symbols*, *model*) **returns** *true* or *false*
  **if** EMPTY? (*symbols*) **then**
    **if** PL-TRUE? (*KB*, *model*) **then return** PL-TRUE? ($\alpha$, *model*)
    **else return** *true*
  **else do**
    *P* ← FIRST (*symbols*)
    *rest* ← REST (*symbols*)
    **return** (TT-CHECK-ALL (*KB*, $\alpha$, *rest*, *model* $\cup$ {*P* = *true*}) **and**
          TT-CHECK-ALL (*KB*, $\alpha$, *rest*, *model* $\cup$ {*P* = *false*}))

if KB false, return true?

and?

PL-TRUE? (*s*, *m*) returns true if sentence *s* true in model *m*.
TT-ENTAILS? sound and complete, but $O(2^n)$ time complexity.

# What Happened to HAL?




"2001: A Space Odyssey" (1968)

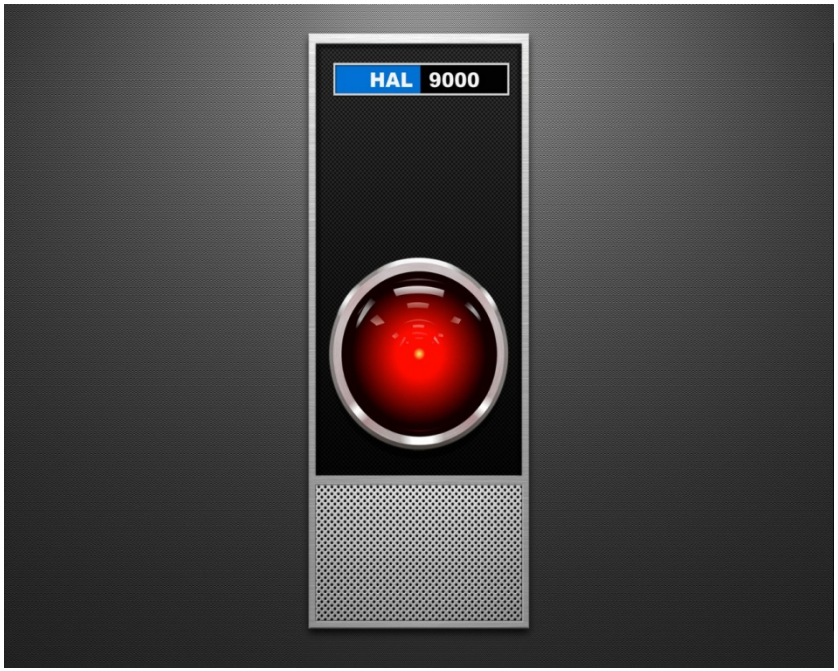
# Propositional Theorem Proving

- Theorem proving
  - Applying rules of inference on sentences in KB to derive sentence $\alpha$
- A sentence is valid if it is true in all models
- **Deduction Theorem:**
  - For any sentences $\alpha$ and $\beta$, $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid

# Propositional Theorem Proving

- A sentence is satisfiable if it is true in some model
  - $\alpha \models \beta$ if and only if the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable
- Proof by refutation or contradiction
  - Prove $\beta$ from $\alpha$ by checking if $(\alpha \wedge \neg\beta)$ is unsatisfiable
  - I.e., assume $\beta$ to be false and show this leads to a contradiction with $\alpha$
  - Also implies that an inconsistent $\alpha$ can prove anything

# What Really Happened to HAL?




"2010" (1984)

# Logical Equivalences

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

# Inference Rules

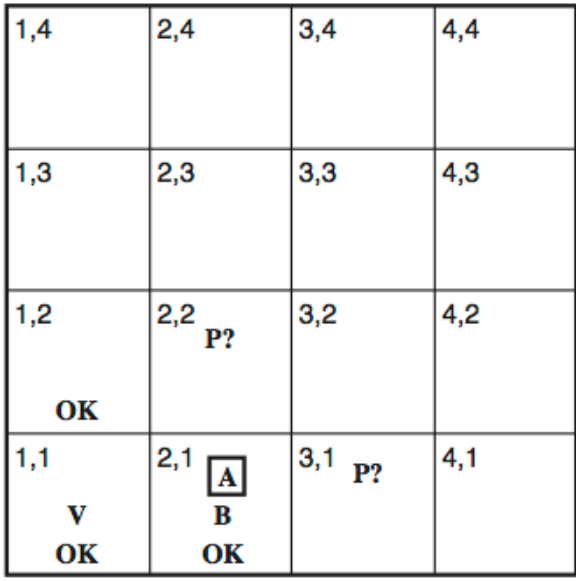
- Notation: $\dfrac{\text{sentences given}}{\text{sentences inferred}}$
- Modus Ponens: $\dfrac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$
- And-Elimination: $\dfrac{\alpha \wedge \beta}{\alpha}$
- Biconditional elimination:

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

# Sample Proof

$R_1$: $\neg P_{1,1}$
$R_2$: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$R_3$: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
$R_4$: $\neg B_{1,1}$
$R_5$: $B_{2,1}$

**Prove**: $\neg P_{1,2}$




Apply biconditional elimination to $R_2$
$R_6$: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Apply And-Elimination to $R_6$
$R_7$: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Apply Contraposition to $R_7$
$R_8$: $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$

Apply Modus Ponens to $R_8$ and $R_4$
$R_9$: $\neg(P_{1,2} \vee P_{2,1})$

Apply De Morgan's rule to $R_9$
$R_{10}$: $\neg P_{1,2} \wedge \neg P_{2,1}$

Apply And-Elimination to $R_{10}$
$\neg P_{1,2}$ (**done**)

# Proof by Search

- Use search to perform propositional inference
- Initial State: KB
- Actions: Apply inference rules to sentences matching top of rule
- Result: Add sentences in bottom of rule to KB
- Goal: KB contains sentence to be proved
- Sound?
- Complete?
- Efficient?

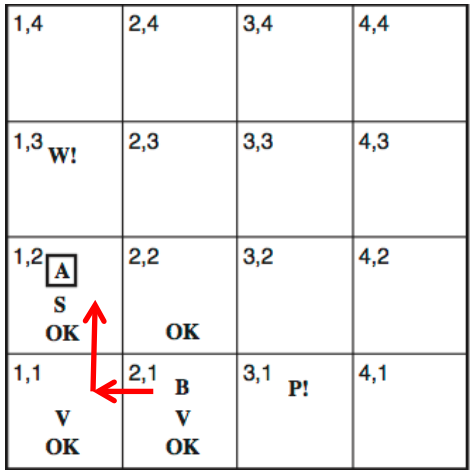
# Resolution

- $\{(A \Rightarrow B), A\} \models B$ *"modus ponens"*
- $\{(\neg A \vee B), A\} \models B$ *"unit resolution"*

- $(A \wedge B \wedge C \Rightarrow D$
  - $\neg(A \wedge B \wedge C) \vee D$
  - $\neg A \vee \neg B \vee \neg C \vee D$
- *Full resolution*
  - $\{(\neg A \vee \neg B \vee \neg C \vee D), (A \vee E)\} \models \neg B \vee \neg C \vee D \vee E$

- Example
  - $(A \Rightarrow C), (B \Rightarrow C), (A \vee B)$
  - $(\neg A \vee C), (\neg B \vee C), (A \vee B)$
  - $(B \vee C), (\neg B \vee C)$
  - $(C \vee C) \models C$

# Proof by Resolution




- Example (cont.)
  - Agent moves from [2,1] to [1,2]

Add percept information to KB

$R_{11}$: $\neg B_{1,2}$

$R_{12}$: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

By earlier process, can eliminate pits in [2,2] and [1,3]

$R_{13}$: $\neg P_{2,2}$

$R_{14}$: $\neg P_{1,3}$

Apply Biconditional Elimination to $R_3$, then Modus Ponens with $R_5$

$R_{15}$: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

Apply **Resolution** to $R_{13}$ and $R_{15}$

$R_{16}$: $P_{1,1} \vee P_{3,1}$

Apply **Resolution** to $R_1$ and $R_{16}$

$R_{17}$: $P_{3,1}$

# Unit Resolution

- Given literals $l_1,\ldots,l_k$ and m, where $l_i$ and m are complementary
- Unit resolution inference rule:

$$\frac{l_1 \vee \cdots \vee l_k, \quad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

- Clause is a disjunction of literals
  - $l_1 \vee \ldots \vee l_k$ is a clause
  - m is a unit clause

# Full Resolution

- Given literals $l_1,\ldots,l_k$ and $m_1,\ldots,m_n$, where $l_i$ and $m_j$ are complementary
- Resolution inference rule:

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

- Resolution is sound and complete!
- Requires sentences to be clauses
- Luckily, every sentence in propositional logic can be expressed as a conjunction of clauses

# Conjunctive Normal Form (CNF)

- Procedure for converting propositional logic sentence to CNF
  1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
  2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
  3. Move $\neg$ inward to appear only in literals
     - $\neg(\neg\alpha) \equiv \alpha$ (double-negation 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)
  4. Apply distributivity of $\vee$ over $\wedge$
     - $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

# CNF Conversion Example

- $W_{1,3} \Leftrightarrow S_{1,4} \wedge S_{1,2} \wedge S_{2,3}$
- Step 1: Eliminate $\Leftrightarrow$
  - $(W_{1,3} \Rightarrow S_{1,4} \wedge S_{1,2} \wedge S_{2,3}) \wedge (S_{1,4} \wedge S_{1,2} \wedge S_{2,3} \Rightarrow W_{1,3})$
- Step 2: Eliminate $\Rightarrow$
  - $(\neg W_{1,3} \vee (S_{1,4} \wedge S_{1,2} \wedge S_{2,3})) \wedge (\neg(S_{1,4} \wedge S_{1,2} \wedge S_{2,3}) \vee W_{1,3})$
- Step 3: Move $\neg$ inward
  - $(\neg W_{1,3} \vee (S_{1,4} \wedge S_{1,2} \wedge S_{2,3})) \wedge (\neg S_{1,4} \vee \neg S_{1,2} \vee \neg S_{2,3} \vee W_{1,3})$
- Step 4: Apply distributivity of $\vee$ over $\wedge$
  - $(\neg W_{1,3} \vee S_{1,4}) \wedge (\neg W_{1,3} \vee S_{1,2}) \wedge (\neg W_{1,3} \vee S_{2,3}) \wedge (\neg S_{1,4} \vee \neg S_{1,2} \vee \neg S_{2,3} \vee W_{1,3})$

# Another Resolution Example

- From previous conversion plus And-Elimination:
  - C1: $(\neg W_{1,3} \vee S_{1,4})$
  - C2: $(\neg W_{1,3} \vee S_{1,2})$
  - C3: $(\neg W_{1,3} \vee S_{2,3})$
  - C4: $(\neg S_{1,4} \vee \neg S_{1,2} \vee \neg S_{2,3} \vee W_{1,3})$
- Assume we have observed the surrounding stenches:
  - C5: $S_{1,4}$
  - C6: $S_{1,2}$
  - C7: $S_{2,3}$
- Resolving C4 with C5:
  - C8: $(\neg S_{1,2} \vee \neg S_{2,3} \vee W_{1,3})$
- Resolving C8 with C6:
  - C9: $(\neg S_{2,3} \vee W_{1,3})$
- Resolving C9 with C7:
  - $W_{1,3}$

| | | | |
|---|---|---|---|
| 1,4 S | 2,4 | 3,4 | 4,4 |
| 1,3 W! | 2,3 S | 3,3 | 4,3 |
| 1,2 A S OK | 2,2 OK | 3,2 | 4,2 |
| 1,1 V OK | 2,1 B V OK | 3,1 P! | 4,1 |

# Propositional Logic Resolution

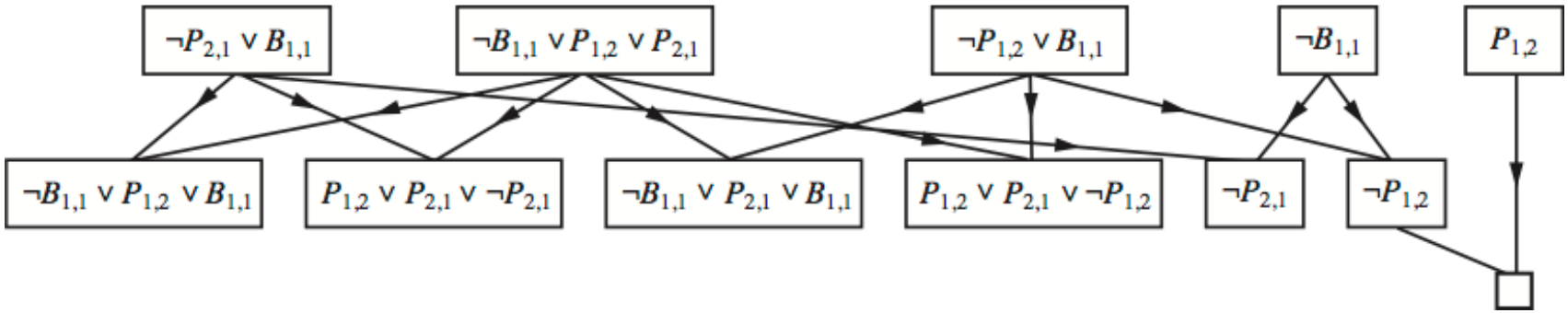
**function** PL-RESOLUTION? ($KB$, $\alpha$) **returns** *true* or *false*
  *clauses* ← convert ($KB \wedge \neg\alpha$) to CNF
  *new* ← {}
  **loop do**
    **for each** pair of clauses $C_i$, $C_j$ **in** *clauses* **do**
      *resolvents* ← PL-RESOLVE ($C_i$, $C_j$)
      **if** *resolvents* contains the empty clause **then return** *true*
      *new* ← *new* ∪ *resolvents*
    **if** *new* ⊆ *clauses* **then return** *false*
    *clauses* ← *clauses* ∪ *new*

- Proof by contradiction
- Empty clause result of resolving A with ¬A
- If iteration produces no new clauses, then KB ⊭ α

# Example

| | | | |
|---|---|---|---|
| 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 |

- $KB = R_2 \wedge R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



# Efficiency of PL-Resolution

- Branching factor $|KB|^2$
- Many heuristics apply
  - E.g., prefer unit clauses
- Local search works surprisingly well (e.g., WALKSAT)
- Horn clause is a clause with at most one positive literal
  - E.g., $(A \wedge B \wedge C \Rightarrow D) \equiv (\neg A \vee \neg B \vee \neg C \vee D)$
  - Entailment with Horn clauses is linear in $|KB|$

# PL-Based Agent

- KB includes initial state and axioms
  - $\neg W_{1,1} \wedge \neg P_{1,1} \wedge L_{1,1} \wedge \text{FacingEast} \wedge \text{HaveArrow} \wedge \neg\text{HaveGold} \wedge \text{AgentAlive} \wedge \text{InCave} \wedge \text{WumpusAlive}$
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $S_{1,1} \Leftrightarrow (W_{1,2} \vee W_{2,1})$
  - …
  - At most one wumpus
    - $\neg(W_{1,1} \wedge W_{1,2})$, $\neg(W_{1,1} \wedge W_{1,3})$, …
- Would prefer
  - $B_{x,y} \Leftrightarrow (P_{x-1,y} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x,y+1})$
  - First-order logic

# PL-Based Agent

- Movement
  - E.g., TurnLeft
    - Add ¬FacingEast and FacingNorth to KB
    - Remove FacingEast; add FacingNorth
  - E.g., Forward
    - Remove $L_{1,1}$; add $L_{2,1}$
    - Bump?
  - E.g., Shoot
    - Remove HaveArrow, add ¬HaveArrow
    - WumpusAlive?
- Monotonic logic
  - Set of entailed sentences can only increase
- History? (e.g., $L_{1,1}$ at t=1)

# PL-Based Agent

- Frame problem
  - Describing what changes and what stays the same after taking an action
- Frame axiom
  - E.g., $\text{HaveArrow}^{t+1} \Leftrightarrow \text{HaveArrow}^{t} \wedge \neg \text{Shoot}^{t}$
  - Requires sets of axioms for each time transition
  - Or, first-order-ness

# PL-Based Agent

- Intermediate information
  - $OK_{2,1}$
  - $\neg Visited_{2,1}$
- Goals
  - $HaveGold \wedge \neg InCave$
  - $OK_{x,y} \Rightarrow Visited_{x,y}$
  - $\neg WumpusAlive$
- In general, pure PL-based agent cumbersome
- Need hybrid logic/search-based agent

**function** HYBRID-WUMPUS-AGENT(*percept*) **returns** an *action*
**inputs**: *percept*, a list, [*stench*,*breeze*,*glitter*,*bump*,*scream*]
**persistent**: $KB$, a knowledge base, initially the atemporal "wumpus physics"
$t$, a counter, initially 0, indicating time
*plan*, an action sequence, initially empty

TELL($KB$, MAKE-PERCEPT-SENTENCE(*percept*, $t$))
TELL the $KB$ the temporal "physics" sentences for time $t$
$safe \leftarrow \{[x, y] : \text{ASK}(KB, OK^t_{x,y}) = true\}$
**if** ASK($KB$, $Glitter^t$) = *true* **then**
*plan* $\leftarrow$ [*Grab*] + PLAN-ROUTE(*current*, {[1,1]}, *safe*) + [*Climb*]
**if** *plan* is empty **then**
$unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^{t'}_{x,y}) = false \text{ for all } t' \leq t\}$
*plan* $\leftarrow$ PLAN-ROUTE(*current*, *unvisited* $\cap$ *safe*, *safe*)
**if** *plan* is empty and ASK($KB$, $HaveArrow^t$) = *true* **then**
$possible\_wumpus \leftarrow \{[x, y] : \text{ASK}(KB, \neg\, W_{x,y}) = false\}$
*plan* $\leftarrow$ PLAN-SHOT(*current*, *possible_wumpus*, *safe*)
**if** *plan* is empty **then** // no choice but to take a risk
$not\_unsafe \leftarrow \{[x, y] : \text{ASK}(KB, \neg\, OK^t_{x,y}) = false\}$
*plan* $\leftarrow$ PLAN-ROUTE(*current*, *unvisited* $\cap$ *not_unsafe*, *safe*)
**if** *plan* is empty **then**
*plan* $\leftarrow$ PLAN-ROUTE(*current*, {[1, 1]}, *safe*) + [*Climb*]
*action* $\leftarrow$ POP(*plan*)
TELL($KB$, MAKE-ACTION-SENTENCE(*action*, $t$))
$t \leftarrow t + 1$
**return** *action*

# Hybrid Wumpus Agent (cont.)

```
function PLAN-ROUTE(current, goals, allowed) returns an action sequence
  inputs: current, the agent's current position
          goals, a set of squares; try to plan a route to one of them
          allowed, a set of squares that can form part of the route

  problem ← ROUTE-PROBLEM(current, goals, allowed)
  return A*-GRAPH-SEARCH(problem)
```

# First-Order Logic

- Propositional logic insufficient to express even commonsense knowledge
- First-order logic (FOL) or First-order predicate calculus (FOPC)
- Borrowing from elements of natural language
  - Objects: nouns, noun phrases (e.g., wumpus, pit)
  - Relations: verbs, verb phrases (e.g., shoot)
    - Properties: adjectives (e.g., smelly)
    - Functions: map input to single output (e.g., location(wumpus))

# Ontological View of Logics

- **Propositional logic** assumes world consists of facts that are either true, false or unknown
  - E.g., wumpus(1,3) ⇒ stench(1,2)
- **First-order logic** assumes world consists of facts, objects and relations that are either true, false or unknown
  - E.g., wumpus(X,Y) ⇒ stench(X,Y-1)
- **Temporal logic** = FOL where facts hold at particular times
  - E.g., before(action(shoot), percept(scream))
- **Higher-order logic** assumes world includes first-order relations as objects
  - E.g., know([wumpus(X,Y) ⇒ stench(X,Y-1)])
- **Probabilistic logic** = propositional logic with a degree of belief for each fact
  - E.g., P(wumpus(1,3)) = 0.067

# FOL Syntax

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 | B | Wumpus | 1 | 2 | ...
Variable → a | x | s | ...
Predicate → True | False | Adjacent | At | Alive | ...
Function → Location | RightOf | ...

Operator Precedence: ¬, =, ∧, ∨, ⇒, ⇔

# FOL Semantics

- Constant symbols stand for objects
- Predicate symbols stand for relations
- Function symbols stand for functions
- R&N convention: Above symbols begin with uppercase letters
  - E.g., Wumpus, Adjacent, RightOf
- Arity is the number of arguments to a predicate or function
  - E.g., Adjacent ($loc_1$, $loc_2$), RightOf (location)

# FOL Semantics

- Terms represent objects with constants, variables or functions
- Note: Functions do not return an object, but represent that object
  - E.g., Action(Forward,t) $\wedge$ Orientation(Agent, Right, t) $\wedge$ At(Agent, loc, t) $\Rightarrow$ At(Agent, RightOf(loc), t+1)
- R&N convention: variables begin with lowercase letters

# Quantifiers

- Express properties of collections of objects
- Universal quantification ($\forall$)
  - A statement is true for all objects represented by quantified variables
  - E.g., $\forall$ x,y At(Wumpus,x,y) $\Rightarrow$ Stench(x+1,y)
  - Same as $\forall$ x,y At(Wumpus,x,y) $\wedge$ Stench(x+1,y) ?
  - Same as $\forall$ x,y $\neg$At(Wumpus,x,y) $\vee$ Stench(x+1,y) ?
- $\forall x\ P(x) \equiv P(A) \wedge P(B) \wedge P(Wumpus) \wedge \ldots$

# Quantifiers

- Existential quantification (∃)
  - There exists at least one set of objects, represented by quantified variables, for which a statement is true
  - E.g., ∃ w,x,y At(w,x,y) ∧ Wumpus(w)
  - Same as ∃ w,x,y At(w,x,y) ⇒ Wumpus(w) ?
- ∃x P(x) ≡ P(A) ∨ P(B) ∨ P(Wumpus) ∨ ...

# Properties of Quantifiers

- Nested quantifiers
- $\forall x\ \forall y$ same as $\forall y\ \forall x$ same as $\forall x,y$
- $\exists x\ \exists y$ same as $\exists y\ \exists x$ same as $\exists x,y$
- $\exists x\ \forall y$ same as $\forall y\ \exists x$ ?
  - $\exists x\ \forall y$ Likes(x,y) ?
  - $\forall y\ \exists x$ Likes(x,y) ?
  - $\forall x\ \exists y$ Likes(x,y) ?
  - $\exists y\ \forall x$ Likes(x,y) ?

# Properties of Quantifiers

- Negation and quantifiers
- $\exists x\ P(x) \equiv \neg\forall x\ \neg P(x)$
  - "If P is true for some x, then P can't be false for all x"
- $\forall x\ P(x) \equiv \neg\exists x\ \neg P(x)$
  - "If P is true for all x, then there can't be an x for which P is false."
- $\forall x\ \neg P(x) \equiv \neg\exists x\ P(x)$
  - "If P is false for all x, then there can't be an x for which P is true."
- $\neg\forall x\ P(x) \equiv \exists x\ \neg P(x)$
  - "If P is not true for all x, then there must be an x for which P is false."

# Equality

- Equality symbol (Term1 = Term1) means Term1 and Term2 refer to the same object
  - E.g., RightOf(Location(1,1)) = Location(2,1)
- Useful for constraining two terms to be different
- E.g., Sibling
  - Sibling(x,y) $\Leftrightarrow$ Parent(p,x) $\wedge$ Parent(p,y)
  - Sibling(x,y) $\Leftrightarrow$ Parent(p,x) $\wedge$ Parent(p,y) $\wedge$ $\neg$(x = y)
  - $\forall$x,y Sibling(x,y) $\Leftrightarrow$ $\exists$p Parent(p,x) $\wedge$ Parent(p,y) $\wedge$ $\neg$(x = y)

# Closed-World Assumption

- How do we express that there is only one wumpus?
  - $\text{Wumpus}(x,y) \Rightarrow \neg\text{Wumpus}(w,z) \wedge (\neg(w = x) \vee \neg(z=y))$
- How about one arrow? One gold? At least one pit?
- Closed-world assumption
  - Atomic sentences not known to be true are assumed false
- Unique-names assumption
  - Every constant symbol refers to a distinct object
- Domain closure
  - If not named by a constant symbol, then doesn't exist

# Using FOL

- Carefully…

Monty Python and the Holy Grail (1975)

# Using FOL

- TELL (KB, α)
- ASK (KB, β)
- TELL (KB, Percept([st,br,Glitter,bu,sc],5))
- ASK (KB, ∃a Action(a,5))
- I.e., does KB entail any particular actions at time 5?
- Answer: Yes, {a/Grab} ← substitution (binding list)
- ASKVARS (KB, α)
  - Returns answers (variable bindings) that make α true
  - Or, use Answer literal (later)

# FOL for the Wumpus World

- Percepts
  - Percept(p,t) = predicate that is true if percept p observed at time t
  - Percept is a list of five terms
  - E.g., Percept([Stench,Breeze,Glitter,None,None],5)
- Actions
  - Forward, TurnLeft, TurnRight, Grab, Shoot, Climb
- AskVars (∃a BestAction(a,5)) → {a/Grab}

# FOL for the Wumpus World

- "Perception"
  - ∀ t,s,g,m,c Percept([s,Breeze,g,m,c],t) ⇒ Breeze(t)
  - ∀ t,s,b,m,c Percept([s,b,Glitter,m,c],t) ⇒ Glitter(t)
- Reflex agent
  - ∀ t Glitter(t) ⇒ BestAction(Grab,t)

# FOL for the Wumpus World

- Location list term [x,y] (e.g., [1,2])
  - Pit(s) or Pit([x,y])
  - At(Wumpus,[x,y],t)
  - At(Agent,[1,1],1)
- Definition of Breezy(s), where s is a location
  - $\forall s$ Breezy(s) $\Leftrightarrow \exists r$ Adjacent(s,r) $\wedge$ Pit(r)
- Definition of Adjacent
  - $\forall x,y,a,b$ 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))

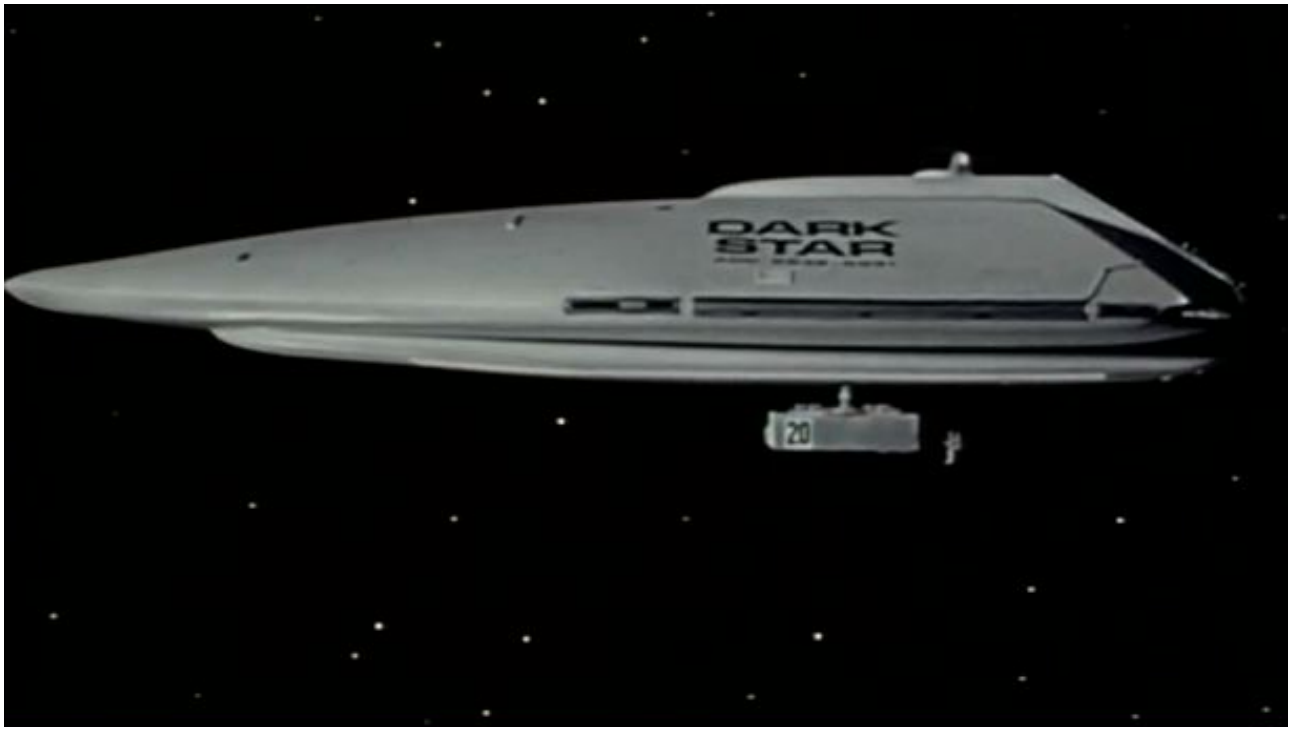
# FOL for the Wumpus World

- Movement
- Wumpus never moves
  - $\forall t$ At(Wumpus,[1,3],t)
- Nothing can be in two places at once
  - $\forall x,s_1,s_2,t$ At(x, $s_1$,t) $\wedge$ At(x, $s_2$,t) $\Rightarrow$ $s_1 = s_2$
- Successor-state axioms for each action
  - Describes what's true before and after action
  - $\forall t$ HaveArrow(t+1) $\Leftrightarrow$ (HaveArrow(t) $\wedge$ $\neg$Action(Shoot,t))
  - $\forall t$ HaveGold(t+1) $\Leftrightarrow$ (HaveGold(t) $\vee$ (Glitter(t) $\wedge$ Action(Grab,t)))
  - ...

# Inference in First-Order Logic

- Now that we have FOL, how can we perform sound, complete and efficient inference?
- Approaches
  - Convert to propositional logic
  - Generalized Modus Ponens
  - Forward and backward chaining
  - Logic programming (Prolog)
  - Resolution
- State of the art

# Note of Caution: Bomb #20




Dark Star (1974)

# Propositionalization

- Convert FOL problem to PL problem
- Main challenge: Remove quantifiers
- Universal instantiation

$$\frac{\forall v \quad \alpha}{\mathrm{SUBST}(\{v/g\},\alpha)}$$

- Substitute every ground term g for any variable v in α
- E.g., ∀s,t,r At(Wumpus,s,t) ∧ Adjacent(s,r) ⇒ Stench(r)
  - At(Wumpus,[1,3],1) ∧ Adjacent([1,3],[2,3]) ⇒ Stench([2,3])
  - At(Wumpus,[2,2],1) ∧ Adjacent([2,2],[2,3]) ⇒ Stench([2,3])
  - …

# Propositionalization

- Existential instantiation

$$\frac{\exists v \quad \alpha}{\mathrm{SUBST}(\{v/k\}, \alpha)}$$

  - Substitute new constant symbol k for variable v in α
- E.g., ∃s,t At(Wumpus,s,t)
  - At(Wumpus,S1,T1)
  - S1 and T1 are new constant symbols
- After removing quantifiers, perform PL inference
- Complete, but inefficient
- FOL is semi-decidable
  - Some algorithms can say yes to every entailed sentence, but no algorithm can say no to every non-entailed sentence

# Generalized Modus Ponens

- Substitution (binding) $\theta = \{x/y\}$
  - Replace all occurrences of x with y
  - E.g., $\alpha$ = At(Wumpus,s,t), $\theta = \{s/[1,3], t/5\}$
    - $\alpha\theta$ = At(Wumpus,[1,3],5)
- Generalized Modus Ponens

$$\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

  - where SUBST($\theta$,$p_i$') = SUBST($\theta$,$p_i$) for all i
- Find $\theta$ via unification

# Generalized Modus Ponens

- Example
- $\forall s,r\ Pit(s) \wedge Adjacent(s,r) \Rightarrow Breeze(r)$
- Pit([3,1]), Adjacent([3,1],[2,1])
- $p_1 = Pit(s)$, $p_2 = Adjacent(s,r)$, $q = Breeze(r)$
- $p_1' = Pit([3,1])$, $p_2' = Adjacent([3,1],[2,1])$
- $\theta = \{s/[3,1], r/[2,1]\}$
- $SUBST(\theta,q) = Breeze([2,1])$

# Unification

- Unification determines if two sentences match given some substitution (unifier)
- UNIFY(p,q) = $\theta$ where SUBST($\theta$,p) = SUBST($\theta$,q)
- Examples
  - UNIFY (At(Wumpus,s,t), At(Wumpus,[1,3],5)) = {s/[1,3], t/5}
  - UNIFY (At(Wumpus,s,t), At(Wumpus,r,5)) = {s/r, t/5}
  - UNIFY (At(Wumpus,s,t), At(Wumpus,AgentLoc(t),5)) = {s/AgentLoc(t), t/5} = {s/AgentLoc(5), t/5}

# Unification Issues

- Standardizing apart
  - Use unique variable names in each sentence
  - UNIFY (At(x,[1,3],t), At(Wumpus,x,t)) = failure
  - UNIFY (At($x_{17}$,[1,3],t), At(Wumpus,$x_{21}$,5)) = {$x_{17}$/Wumpus, $x_{21}$/[1,3], t/5}
- Most General Unifier (MGU)
  - Unifier returned by Unify should place the least possible restrictions on variables
  - UNIFY (Adjacent(r,s),Adjacent([1,3],x)) = {r/[1,3], s/[2,3], x/[2,3]} works
  - But so does {r/[1,3], s/x} (more general)

# Unification Algorithm

- Recursively compare two expressions
- Build up substitutions along the way
- Details
  - Compound expression of the form F(A,B)
  - If x = F(A,B), then x.OP = F, x.ARGS = [A,B] (a list)
  - List decomposed using First and Rest
  - If x = [A,B,C], then x.FIRST = A and x.REST = [B,C]

# Unification

**function** UNIFY ($x$, $y$, $\theta$) **returns** a substitution to make $x$ and $y$ identical
  **inputs**: $x$, a variable, constant, list, or compound expression
    $y$, a variable, constant, list, or compound expression
    $\theta$, the substitution built up so far (optional, defaults to empty)
  **if** $\theta$ = failure **then return** failure
  **else if** $x = y$ **then return** $\theta$
  **else if** VARIABLE?($x$) **then return** UNIFY-VAR($x$,$y$,$\theta$)
  **else if** VARIABLE?($y$) **then return** UNIFY-VAR($y$,$x$,$\theta$)
  **else if** COMPOUND?($x$) **and** COMPOUND?($y$) **then**
    **return** UNIFY($x$.ARGS, $y$.ARGS, UNIFY($x$.OP, $y$.OP, $\theta$))
  **else if** LIST?($x$) **and** LIST?($y$) **then**
    **return** UNIFY($x$.REST, $y$.REST, UNIFY($x$.FIRST, $y$.FIRST, $\theta$))
  **else return** failure

# Unification

**function** UNIFY-VAR (*var*, *x*, $\theta$) **returns** a substitution
  **if** {*var* / *val*} $\in$ $\theta$ **then return** UNIFY(*val*, *x*, $\theta$)
  **else if** {*x* / *val*} $\in$ $\theta$ **then return** UNIFY(*var*, *val*, $\theta$)
  **else if** OCCUR-CHECK?(*var*, *x*) **then return** failure
  **else return** add {*var* / *x*} to $\theta$

- Occur check
  - When matching variable and term, check if variable occurs in term
  - If so, failure (e.g., P(x) does not unify with P(P(x))
  - Makes UNIFY quadratic in size of expression
  - Some inference systems omit occur check

# Forward Chaining

- Start with atomic sentences in KB
- Apply Modus Ponens where possible to infer new atomic sentences
- Continue until goal is proven or no new inferences can be made
- Assume first-order definite clauses for now
  - Disjunction of literals with exactly one positive literal
  - E.g., $\neg A(x) \vee \neg B(y) \vee C(x,y) \equiv A(x) \wedge B(y) \Rightarrow C(x,y)$

# Example

- 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 Col. West is a criminal

# Example

- "… it is a crime for an American to sell weapons to hostile nations."
  - R1: American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x,y,z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x)
- "… Nono, an enemy of America, …"
  - R2: Enemy(Nono,America)
- "… Nono … has some missiles"
  - $\exists$x Owns(Nono,x) $\wedge$ Missile(x)
  - R3: Owns(Nono,$M_1$)
  - R4: Missle($M_1$)

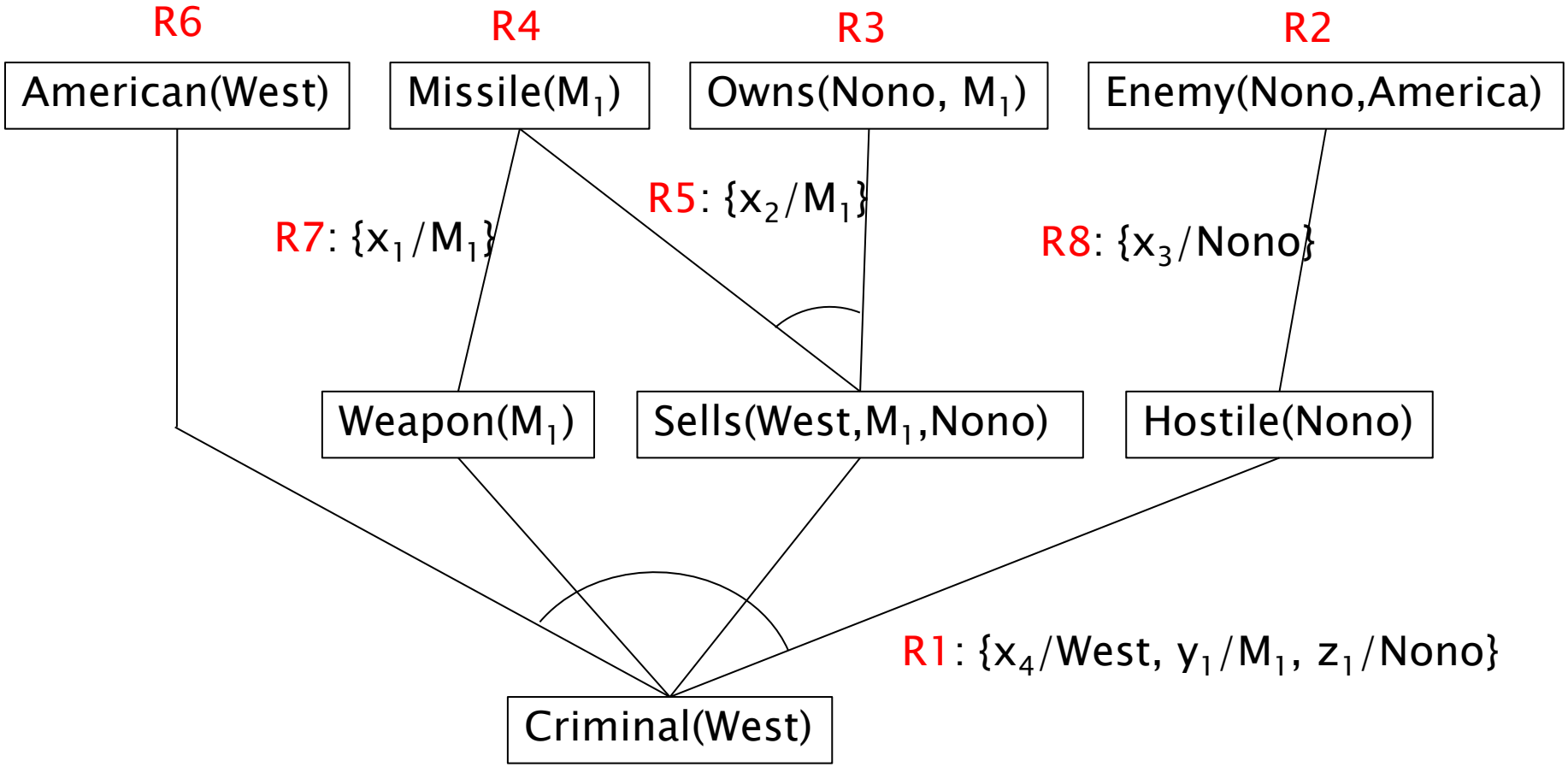
} Existential Instantiation

# Example

- "… all of its missiles were sold to it by Colonel West"
  - R5: Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)
- "… Colonel West, who is American."
  - R6: American(West)
- A few more rules…
  - R7: Missile(x) $\Rightarrow$ Weapon(x)
  - R8: Enemy(x,America) $\Rightarrow$ Hostile(x)

- Note: All variables universally quantified

# Example: Forward Chaining



**function** FOL-FC-ASK (*KB*, $\alpha$) returns a substitution or *false*

**inputs**: *KB*, the knowledge base, a set of first-order definite clauses
$\alpha$, the query, an atomic sentence

**local variables**: *new*, the new sentences inferred at each iteration

**repeat until** *new* is empty
*new* ← { }
**for each** *rule* **in** *KB* **do**
($p_1 \wedge \ldots \wedge p_n \Rightarrow q$) ← STANDARDIZE-VARIABLES(*rule*)
**for each** $\theta$ such that SUBST($\theta$, $p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta$, $p_1' \wedge \ldots \wedge p_n'$)
for some $p_1'$, …, $p_n'$ in *KB*
*q'* ← SUBST($\theta$, *q*)
**if** *q'* does not unify with some sentence already in *KB* or *new* **then**
add *q'* to *new*
$\phi$ = UNIFY(*q'*, $\alpha$)
**if** $\phi$ is not *fail* **then return** $\phi$
add *new* to *KB*
**return** *false*

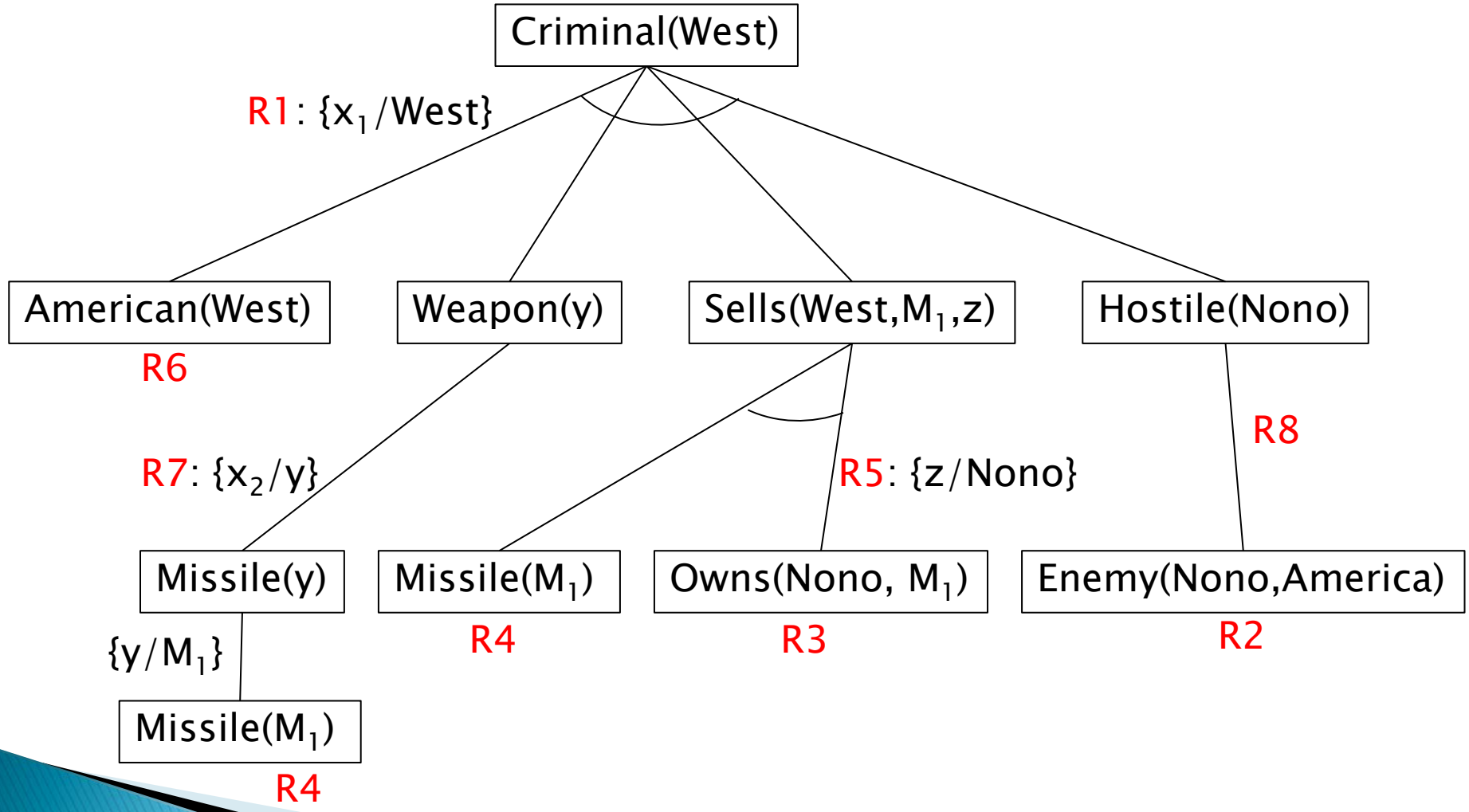
# Forward Chaining

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all known facts
    - Conjunct ordering
    - R5: Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)
  - Recheck every rule on every iteration
    - Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration t-1
    - Incremental forward chaining
  - Irrelevant facts (e.g., Enemy(Wumpus,America))

# Backward Chaining

- Work backwards from the goal
- For rules concluding goal, add premises as new goals
- Continue until all open goals supported by known facts
- Again, assume first-order definite clauses for now

# Example: Backward Chaining



# Backward Chaining

**function** FOL-BC-ASK($KB$, $goals$, $\theta$) **returns** a set of substitutions

**inputs:** $KB$, a knowledge base

$goals$, a list of conjuncts forming a query ($\theta$ already applied)

$\theta$, the current substitution, initially the empty substitution $\{\}$

**local variables:** $answers$, a set of substitutions, initially empty

**if** $goals$ is empty **then return** $\{\theta\}$

$q' \leftarrow$ SUBST($\theta$, FIRST($goals$))

**for each** sentence $r$ **in** $KB$

where STANDARDIZE-APART($r$) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$

and $\theta' \leftarrow$ UNIFY($q$, $q'$) succeeds

$new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$

$answers \leftarrow$ FOL-BC-ASK($KB$, $new\_goals$, COMPOSE($\theta', \theta$)) $\cup$ $answers$

**return** $answers$

# Backward Chaining

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all **open goals**
    - More constraints
    - R5: Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)
  - Recheck every rule on every iteration
    - Yes, but only those whose consequent unifies with an open goal
  - Irrelevant facts (e.g., Enemy(Wumpus,America))
    - Excluded

# Logic Programming

- A program is a set of logic statements defining the constraints of the problem
- Theorem-proving used to "run" the program
- Prolog is a logic programming language
  - Closed-world assumption
  - Supports arithmetic "X is 1+2" binds X to 3
  - Allows assertion and retraction of sentences
  - No occur check
  - Depth-first backward chaining (incomplete)

# Prolog

- Uses uppercase for variables
- Uses lowercase for constants, functions and predicates
- Uses comma for AND
- Uses ":-" for implication (reversed)
- E.g., "$A \wedge B \Rightarrow C$" written as "C :- A, B."
- sells(west,X,nono) :- missile(X), owns(nono,X).
- Gnu Prolog (www.gprolog.org)

# FOL Resolution

- Resolution using refutation (proof by contradiction) is sound and complete
  - $(\neg A(x) \vee B(x), A(Wumpus)) \vdash B(x) \{x/Wumpus\}$
- Convert FOL to clausal form (CNF)
- Efficient? Resolution strategies

# Convert FOL to CNF

- Conjunctive Normal Form (CNF)
  - Conjunction of clauses
  - Each clause is a disjunction of literals
  - Variables assumed to be universally quantified
- Example
  - $\forall x,y,z\ \text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x,y,z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
  - $\neg\text{American}(x) \vee \neg\text{Weapon}(y) \vee \neg\text{Sells}(x,y,z) \vee \neg\text{Hostile}(x) \vee \text{Criminal}(x)$
- Same as propositional logic, but need to eliminate existential quantifiers

# Convert FOL to CNF

- Step 1: Eliminate implications $\Rightarrow$
  - From: $\forall x\ A(x) \wedge B(x) \Rightarrow C(x)$
  - To: $\forall x\ \neg A(x) \vee \neg B(x) \vee C(x)$
- Step 2: Move $\neg$ inwards
  - $\neg \forall x\ A(x)$ becomes $\exists x\ \neg A(x)$
  - $\neg\ \exists x\ A(x)$ becomes $\forall x\ \neg A(x)$
- Step 3: Standardize variables
  - From: $(\forall x\ A(x)) \wedge (\forall x\ B(x))$
  - To: $(\forall x_1\ A(x_1)) \wedge (\forall x_2\ B(x_2))$

# Convert FOL to CNF

- Step 4: Skolemize (Skolemization)
  - Eliminate existential quantifiers by replacing them with a new constant or function
  - Skolem constant, Skolem function
  - Arguments of the Skolem function are all the universally quantified variables in whose scope the existential quantifier appears
  - From: $\forall x,y\ \exists z\ P(x,y,z)$
  - To: $\forall x,y\ P(x,y,F1(x,y))$

Thoralf Skolem (1887-1963)
Norwegian mathematician

# Convert FOL to CNF

- Step 5: Drop universal quantifiers
  - All remaining variables universally quantified
  - So, just drop the $\forall x,y,\ldots$
- Step 6: Distribute $\vee$ over $\wedge$
  - From: $(A(x) \wedge B(x)) \vee C(x)$
  - To: $(A(x) \vee C(x)) \wedge (B(x) \vee C(x))$

# Example (FOL → CNF)

- What is a brick?
  - A brick is on something that is not a pyramid
  - There is nothing that a brick is on and that is on the brick as well
  - There is nothing that is not a brick and also is the same thing as a brick.

$$\forall x\,[Brick(x) \Rightarrow (\exists y\,[On(x,y) \wedge \neg Pyramid(y)] \wedge \neg\exists y\,[On(x,y) \wedge On(y,x)] \wedge \forall y\,[\neg Brick(y) \Rightarrow \neg Equal(x,y)])]$$

# Example (FOL → CNF)

- Step 1: Eliminate implications

$$\forall x\,[\neg Brick(x) \vee (\exists y\,[On(x,y) \wedge \neg Pyramid(y)] \wedge \neg\exists y\,[On(x,y) \wedge On(y,x)] \wedge \forall y\,[\neg\neg Brick(y) \vee \neg Equal(x,y)])]$$

# Example (FOL → CNF)

- Step 2: Move ¬ inwards

$$\forall x\,[\neg Brick(x) \vee (\exists y\,[On(x,y) \wedge \neg Pyramid(y)] \wedge \forall y\,\neg[On(x,y) \wedge On(y,x)] \wedge \forall y\,[Brick(y) \vee \neg Equal(x,y)])]$$

$$\forall x\,[\neg Brick(x) \vee (\exists y\,[On(x,y) \wedge \neg Pyramid(y)] \wedge \forall y\,[\neg On(x,y) \vee \neg On(y,x)] \wedge \forall y\,[Brick(y) \vee \neg Equal(x,y)])]$$

# Example (FOL → CNF)

- Step 3: Standardize variables

$$\forall x\,[\neg Brick(x) \vee (\exists y\,[On(x,y) \wedge \neg Pyramid(y)] \wedge \forall a\,[\neg On(x,a) \vee \neg On(a,x)] \wedge \forall b\,[Brick(b) \vee \neg Equal(x,b)])]$$

# Example (FOL → CNF)

- Step 4: Skolemization

$$\forall x\,[\neg Brick(x) \vee [On(x,F(x)) \wedge \neg Pyramid(F(x))] \wedge \forall a\,[\neg On(x,a) \vee \neg On(a,x)] \wedge \forall b\,[Brick(b) \vee \neg Equal(x,b)])]$$

- Step 5: Drop universal quantifiers

$$\neg Brick(x) \vee [On(x,F(x)) \wedge \neg Pyramid(F(x))] \wedge [\neg On(x,a) \vee \neg On(a,x)] \wedge [Brick(b) \vee \neg Equal(x,b)])]$$

# Example (FOL → CNF)

- Step 6: Distribute $\vee$ over $\wedge$

$$(\neg Brick(x) \vee On(x,F(x))) \wedge$$
$$(\neg Brick(x) \vee \neg Pyramid(F(x))) \wedge$$
$$(\neg Brick(x) \vee \neg On(x,a) \vee \neg On(a,x)) \wedge$$
$$(\neg Brick(x) \vee Brick(b) \vee \neg Equal(x,b))$$

# Resolution Inference Rule

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

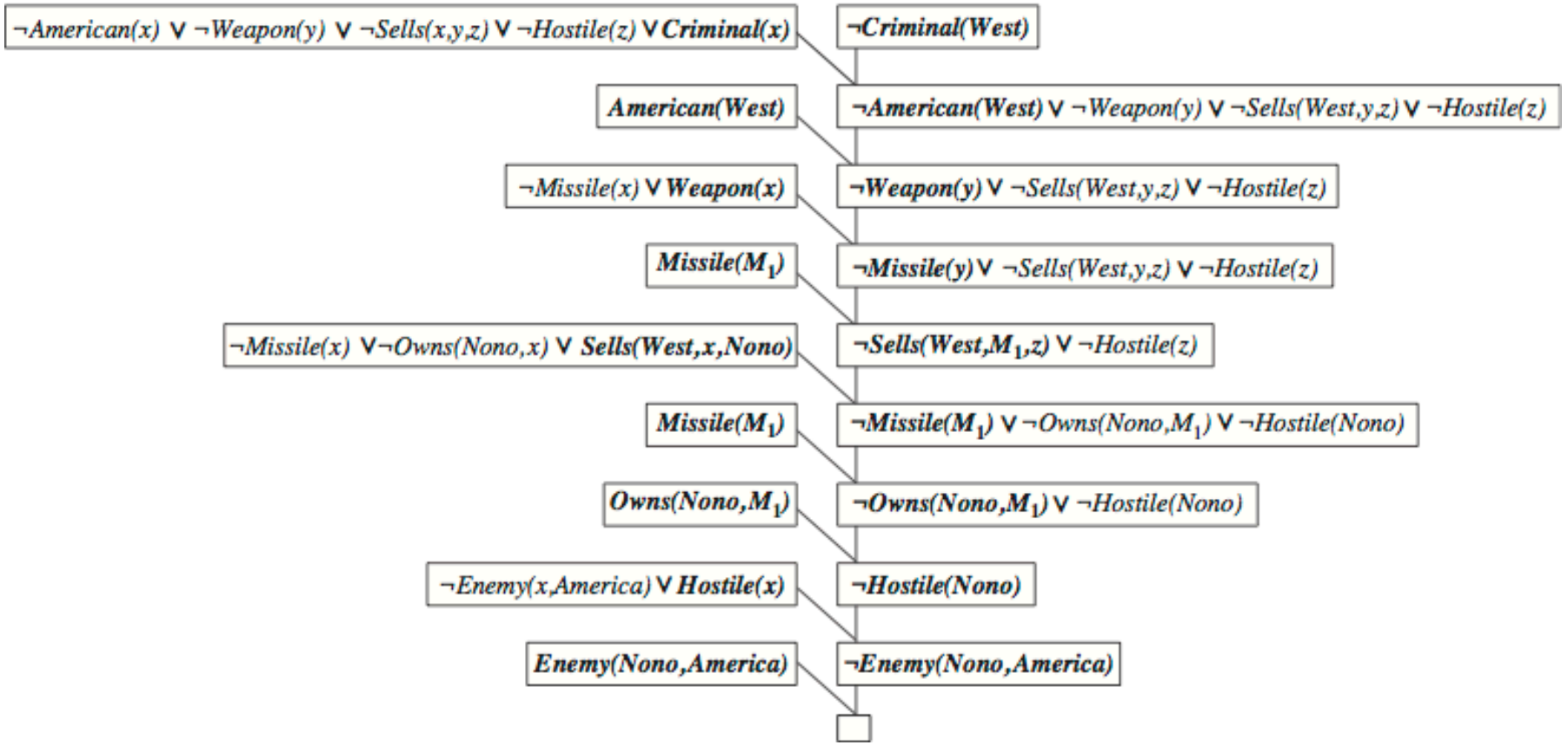
where $\text{UNIFY}(l_i, \neg m_j) = \theta$

- Binary resolution resolves one pair of complementary literals
- Full resolution can resolve multiple pairs of complementary literals
- Resolution plus proof by refutation is sound and complete

# Example Proof: Criminal(West)

- CNF
  - $\neg$American(x) $\vee$ $\neg$Weapon(y) $\vee$ $\neg$Sells(x,y,z) $\vee$ $\neg$Hostile(z) $\vee$ Criminal(x)
  - $\neg$Missile(x) $\vee$ $\neg$Owns(Nono,x) $\vee$ Sells(West,x,Nono)
  - $\neg$Missile(x) $\vee$ Weapon(x)
  - $\neg$Enemy(x,America) $\vee$ Hostile(x)
  - Enemy(Nono,America)
  - Owns(Nono,$M_1$)
  - Missile($M_1$)
  - American(West)
- Prove: Criminal(West)
  - Add $\neg$Criminal(West) to KB and derive empty clause

# Example Proof: Criminal(West)



# Is There a Criminal?

- Prove: $\exists c$ Criminal(c)
  - Add $\neg\exists c$ Criminal(c) to KB
  - I.e., add $\neg$Criminal(c) to KB
- Generated clauses
  - $\neg$American(c) $\vee$ $\neg$Weapon(y) $\vee$ $\neg$Sells(c,y,z) $\vee$ $\neg$Hostile(z)
  - $\neg$Weapon(y) $\vee$ $\neg$Sells(West,y,z) $\vee$ $\neg$Hostile(z)
  - $\neg$Missile(y) $\vee$ $\neg$Sells(West,y,z) $\vee$ $\neg$Hostile(z)
  - $\neg$Sells(West,$M_1$,z) $\vee$ $\neg$Hostile(z)
  - $\neg$Missile($M_1$) $\vee$ $\neg$Owns(Nono,$M_1$) $\vee$ $\neg$Hostile(Nono)
  - $\neg$Owns(Nono,$M_1$) $\vee$ $\neg$Hostile(Nono)
  - $\neg$Hostile(Nono)
  - $\neg$Enemy(Nono,America)
  - $\square$

# Answer Literal

- Add clause with negated goal and answer literal to KB
- Search for clause containing only answer literal
- Prove: $\exists x,y,z$ Goal(x,y,z)
- Add [$\neg$Goal(x,y,z) $\vee$ Answer(x,y,z)] to KB
- Final clause Answer(x,y,z) will have variables bound to answers

# Who is the Criminal?

- Prove: $\exists$c Criminal(c) and retrieve c
  - Add [$\neg$Criminal(c) $\vee$ Answer(c)] to KB
- Generated clauses
  - $\neg$American(c) $\vee$ $\neg$Weapon(y) $\vee$ $\neg$Sells(c,y,z) $\vee$ $\neg$Hostile(z) $\vee$ Answer(c)
  - $\neg$Weapon(y) $\vee$ $\neg$Sells(West,y,z) $\vee$ $\neg$Hostile(z) $\vee$ Answer(West)
  - $\neg$Missile(y) $\vee$ $\neg$Sells(West,y,z) $\vee$ $\neg$Hostile(z) $\vee$ Answer(West)
  - $\neg$Sells(West,$M_1$,z) $\vee$ $\neg$Hostile(z) $\vee$ Answer(West)
  - $\neg$Missile($M_1$) $\vee$ $\neg$Owns(Nono,$M_1$) $\vee$ $\neg$Hostile(Nono) $\vee$ Answer(West)
  - $\neg$Owns(Nono,$M_1$) $\vee$ $\neg$Hostile(Nono) $\vee$ Answer(West)
  - $\neg$Hostile(Nono) $\vee$ Answer(West)
  - $\neg$Enemy(Nono,America) $\vee$ Answer(West)
  - Answer(West)

# Equality

- Handle (x = y) terms in resolution theorem proving
- Let SUB(x,y,m) mean to replace x with y everywhere x occurs within m
- Demodulation

$$\frac{x = y, \quad m_1 \vee \cdots \vee m_n}{\mathrm{SUB}(\mathrm{SUBST}(\theta, x), \mathrm{SUBST}(\theta, y), m_1 \vee \cdots \vee m_n)}$$

$$\text{where } \mathrm{UNIFY}(x, z) = \theta \text{ and } z \text{ appears in literal } m_i$$

- Example
  - Given: Father(Father(x)) = Grandfather(x)
    Birthdate(Father(Father(Bob)),1941)
  - Infer: Birthdate(Grandfather(Bob),1941)

# Equality

- Paramodulation

$$\frac{l_1 \vee \ldots \vee l_k \vee x = y, \quad m_1 \vee \cdots \vee m_n}{\mathrm{SUB}(\mathrm{SUBST}(\theta, x), \mathrm{SUBST}(\theta, y), \mathrm{SUBST}(\theta, l_1 \vee \ldots \vee l_k \vee m_1 \vee \cdots \vee m_n))}$$

where $\mathrm{UNIFY}(x, z) = \theta$ and $z$ appears in literal $m_i$

- Paramodulation is complete for FOL with equality

# Strategies for Efficient Resolution

- Unit preference
  - Prefer resolutions where one clause contains a single literal (unit clause)
  - New sentence always shorter
  - Incomplete (but complete for Horn clauses)
- Set of support
  - Subset of clauses, one of which must always be used for resolution
  - New clauses added to set of support
  - Complete if remaining clauses satisfiable
  - Initially, SoS = negated goal

# Strategies for Efficient Resolution

- Input resolution
  - Every resolution combines sentence from (KB+goal) and some other sentence
  - E.g., Criminal(West) proof
  - Incomplete (but complete for Horn clauses)
- Linear resolution
  - Input resolution, but also allowing ancestor sentences
  - Complete
- Subsumption
  - Eliminate sentences more specific than (subsumed by) others
  - E.g., if P(x) in KB, then don't add P(A)

# Theorem Proving: State of the Art

- Vampire (www.vprover.org)
- E (www.eprover.org)
- iProver (www.cs.man.ac.uk/~korovink/iprover)
- Conference on Automated Deduction (CADE) ATP System Competition (CASC)
  - www.cs.miami.edu/~tptp/CASC/
- Applications
  - Mathematical theorem proving
  - Hardware and software verification and synthesis

# Summary

- Knowledge-based agent
- Logic
- Propositional logic
- First-order logic
- Inference
  - Unification
  - Resolution
  - Proof by contradiction