Logic

Larry Holder
School of EECS
Washington State University

Knowledge-based Agent

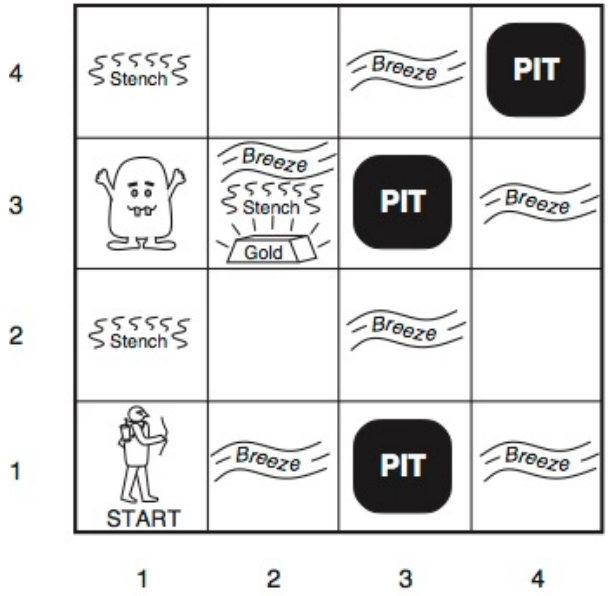
- Knowledge base
  - TELL agent about the environment
- Knowledge representation
  - First-order logic
  - Many others…
- Reasoning via inference
  - ASK agent how to achieve goal based on current knowledge

Knowledge-based Agent

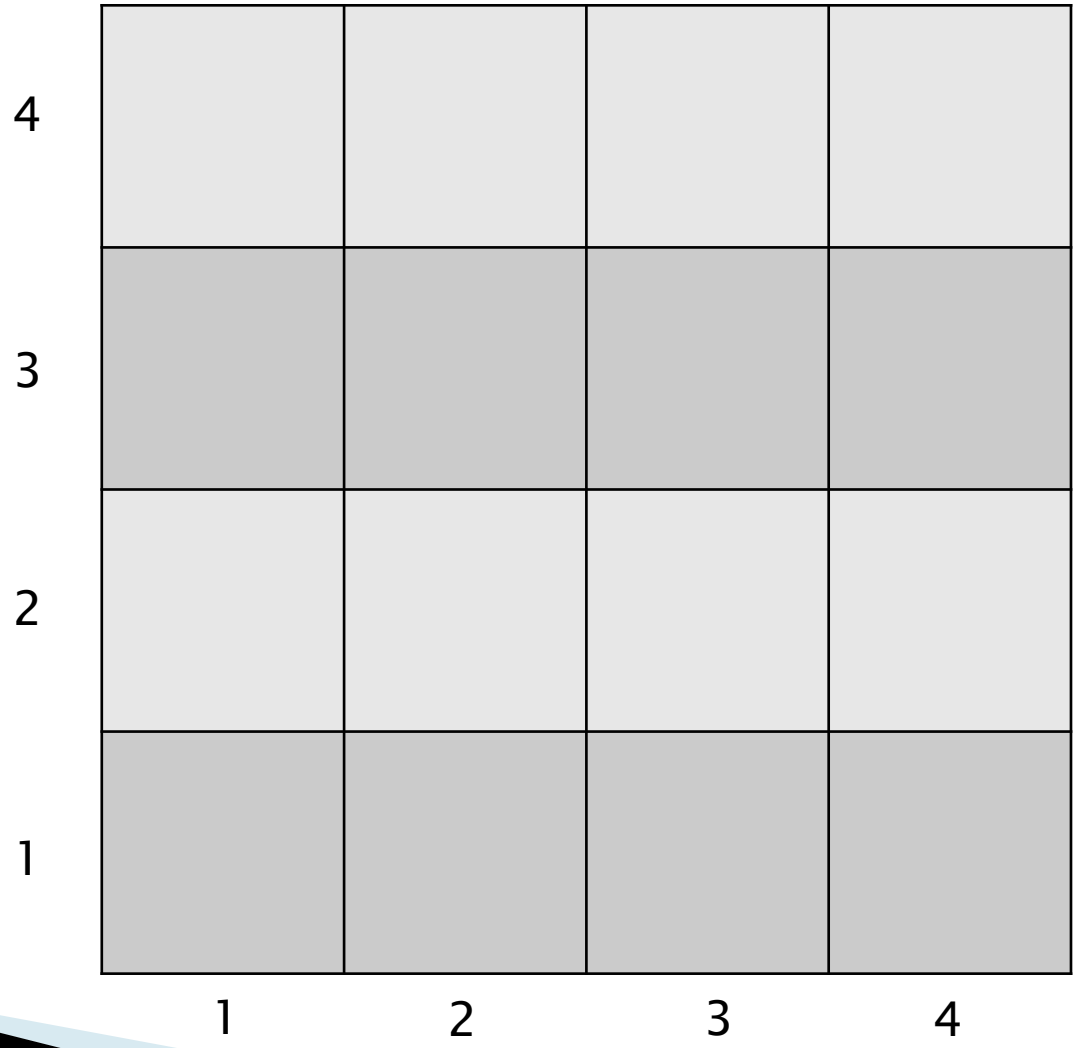
function KB-AGENT (*percept*) **returns** an action
  persistent: *KB*, a knowledge base
    t, a counter, initially 0, indicating time
  TELL (KB, MAKE-PERCEPT-SENTENCE (*percept*, *t*))
  action ← ASK (KB, MAKE-ACTION-QUERY *(t)*)
  TELL (KB, MAKE-ACTION-SENTENCE (*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

Acting Logically in Wumpus World

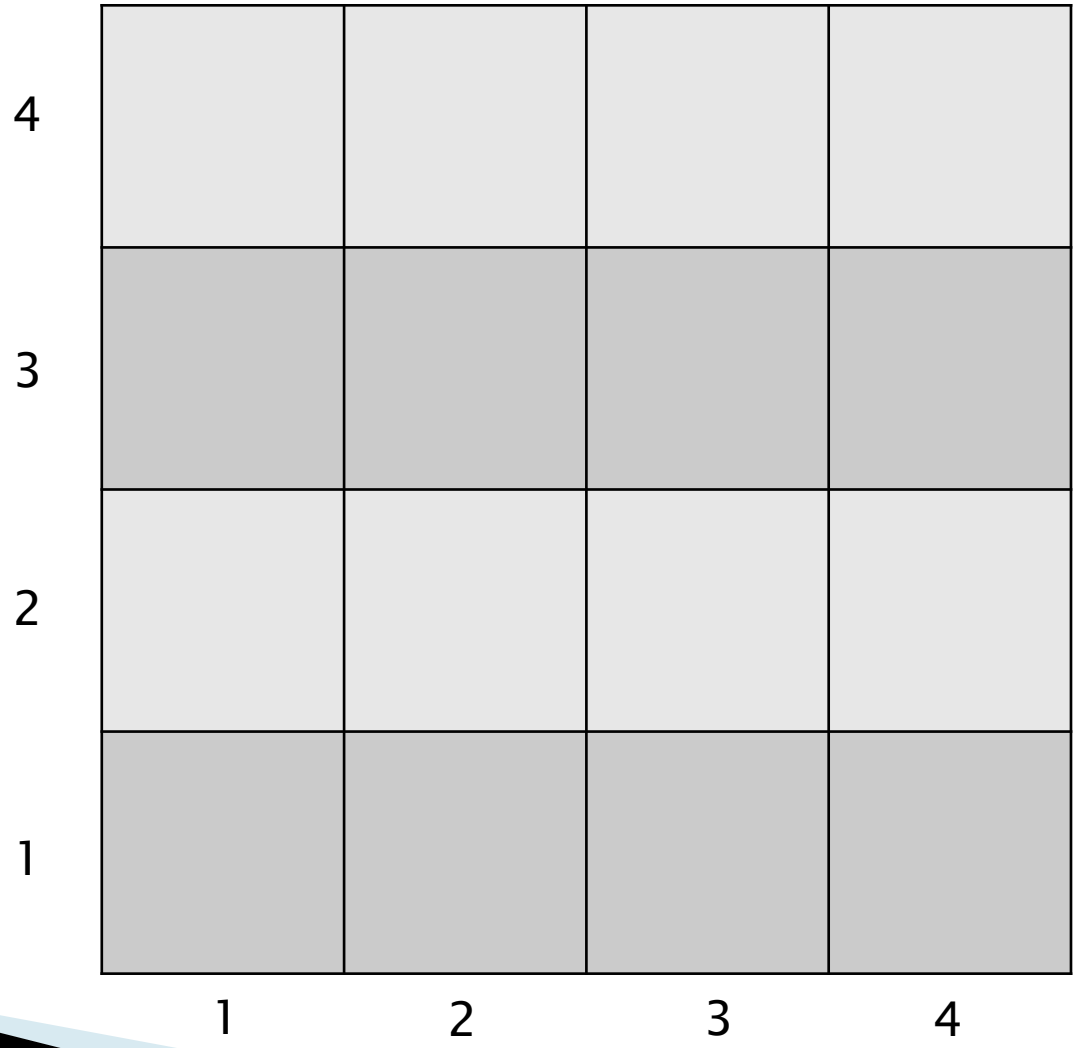
- Goals
  - Visit safe locations
  - Grab gold if present
  - If have gold or no more safe, unvisited locations, then move to [1,1] and Climb



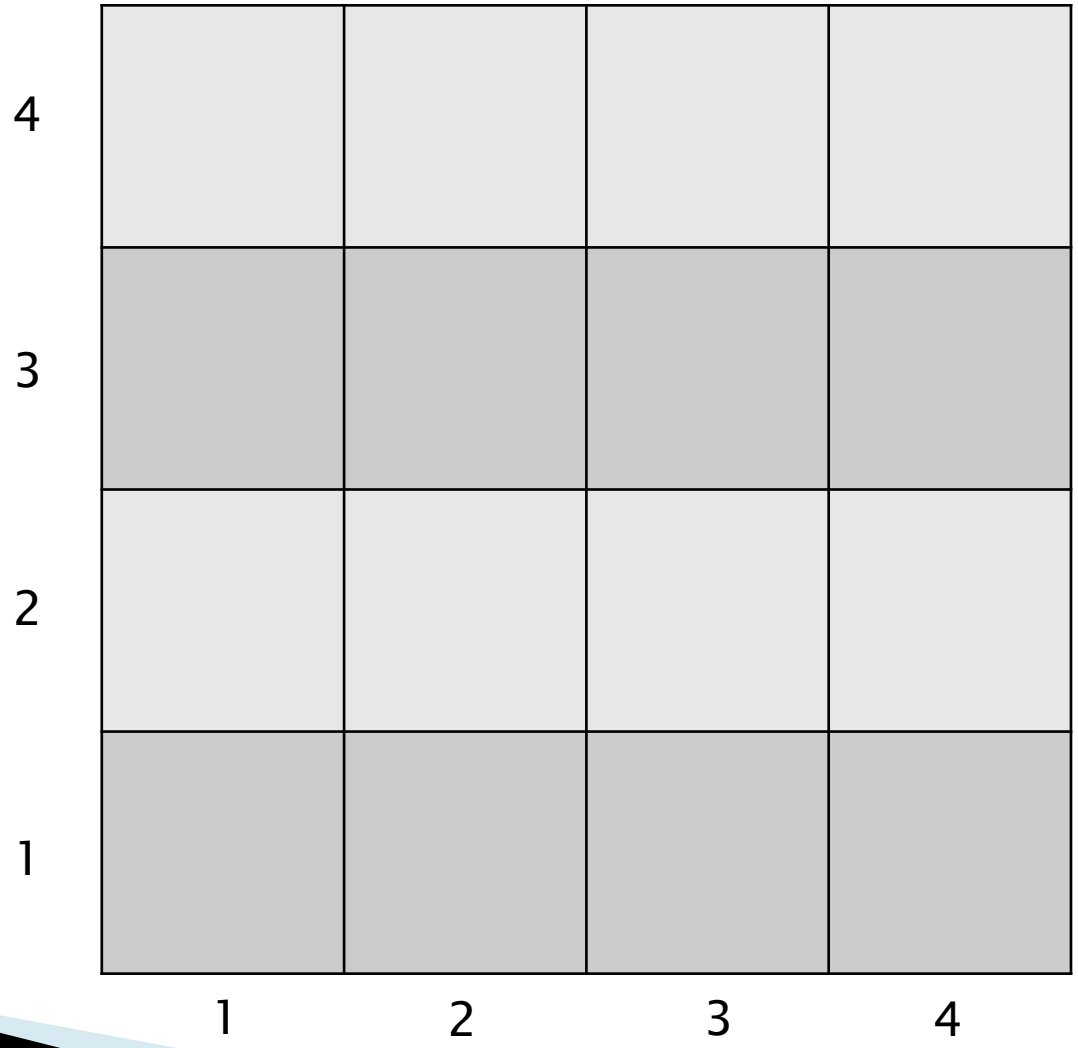
Acting Logically in Wumpus World



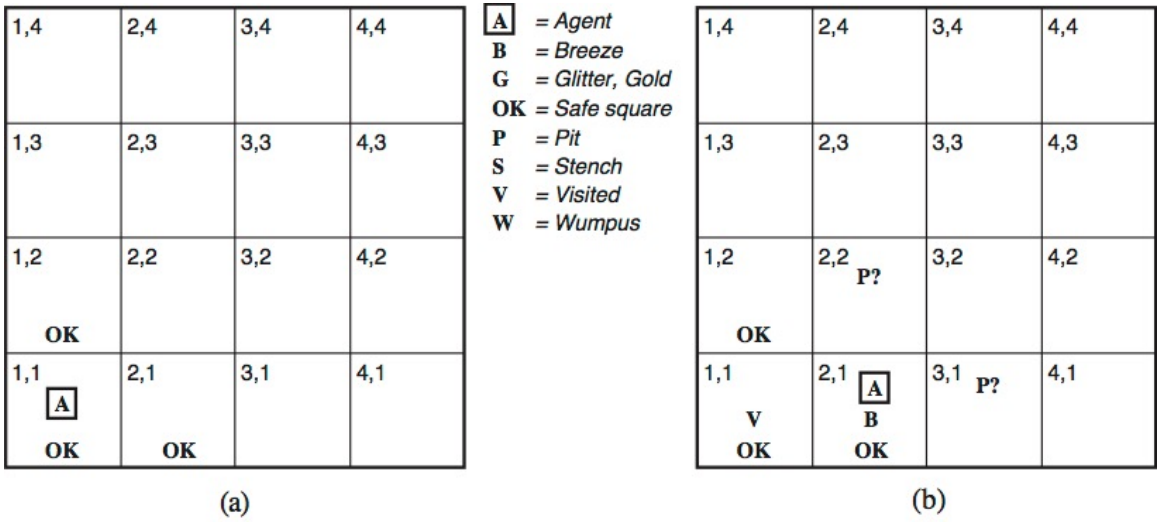
Acting Logically in Wumpus World



Acting Logically in Wumpus World

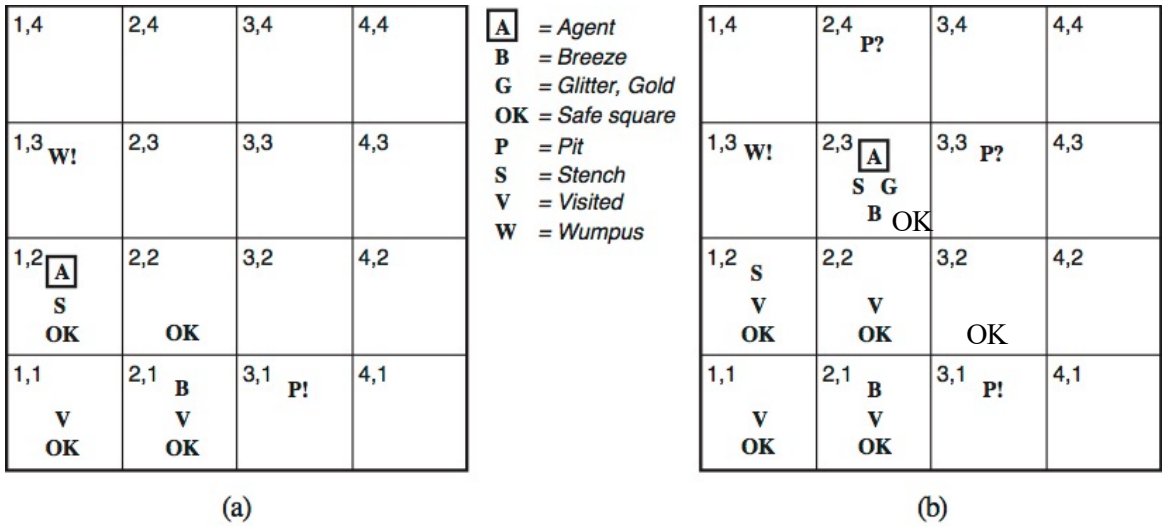


Acting Logically in Wumpus World



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

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,GoForward> to [2,2]
- $Percept_9$ = [None,None,None,None,None]
  - [3,2] and [2,3] are safe
- <TurnLeft,GoForward> to [2,3]
- $Percept_{11}$ = [Stench,Breeze,Glitter,None,None]
- Grab gold, head home, and Climb (score: 1000 – 17 = 983)

Logic

- A knowledge base (KB) consists of "sentences"
- Syntax specifies a well-formed sentence
- Semantics specifies the meaning of a sentence
- Example
  - Syntax: Wumpus(2,2)
  - Semantics: Wumpus is in location (2,2)

Logical Inference

- Logical inference is the process of inferring one sentence is true from others
- Inference should be sound or truth-preserving
  - Everything inferred is true
- Inference should be complete
  - Everything that is true can be inferred

Different 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) ^ Adjacent(x,y,w,z) ⇒ Stench(w,z)
- **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) ^ Adjacent(x,y,w,z) ⇒ Stench(w,z)])
- **Probabilistic logic** = propositional logic with a degree of belief for each fact
  - E.g., P(Wumpus(1,3)) = 0.067

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))

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 Syntax

- Not (¬) is a negation
- Literal is either an atomic sentence (positive literal) or a negated atomic sentence (negative literal)
  - ¬Breeze(1,1), Breeze(2,1)
- And (∧) is a conjunction; its parts are conjuncts
- Or (∨) is a disjunction; its parts are disjuncts
- Implies (⇒) is an implication
  - Pit(2,2) ⇒ Breeze(1,2)
  - Lefthand side is the antecedent or premise
  - Righthand side is the consequent or conclusion
- If and only if (⇔) is a biconditional

FOL Semantics

- How to determine the truth value (true or false) of every sentence
- True is always true
- False is always false
- Truth values of every other sentence must be specified directly or inferred
  - E.g., Wumpus(2,2) is true, Wumpus(3,3) is false

FOL Semantics

- Semantics for complex sentences
  - $\neg P$ is true iff P is false
  - $P \wedge Q$ is true iff both P and Q are true
  - $P \vee Q$ is true iff either P or Q is true
  - $P \Rightarrow Q$ is true unless P is true and Q is false
  - $P \Leftrightarrow Q$ is true iff P and Q are both true or both false

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

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(GoForward,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
  - $\text{Sibling}(x,y) \Leftrightarrow \text{Parent}(p,x) \wedge \text{Parent}(p,y)$
  - $\text{Sibling}(x,y) \Leftrightarrow \text{Parent}(p,x) \wedge \text{Parent}(p,y) \wedge \neg(x = y)$
  - $\forall x,y\ \text{Sibling}(x,y) \Leftrightarrow \exists p\ \text{Parent}(p,x) \wedge \text{Parent}(p,y) \wedge \neg(x = y)$

Closed-World Assumption

- 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

- TELL (KB, α)
  - TELL (KB, Percept([st,br,Glitter,bu,sc],5))
- ASK (KB, β)
  - 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)
  - ASK (KB, ∃a Action(a,5) ^ Answer(a))

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
  - GoForward, 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
  - $\forall t\ HaveArrow(t+1) \Leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot,t))$
  - $\forall t\ HaveGold(t+1) \Leftrightarrow (HaveGold(t) \vee (Glitter(t) \wedge Action(Grab,t)))$
  - ...

FOL for the Wumpus World

- How do we express that there is only one wumpus?
  - $\forall x,y\ (Wumpus(x,y) \Rightarrow \neg(\exists w,z\ Wumpus(w,z) \wedge (\neg(w = x) \vee \neg(z=y))))$
- Only one arrow?

- Only one gold?
- At least one pit?

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 SEARCH(*problem*) // *Any search algorithm*

Inference in First-Order Logic

- Now that we have FOL, how can we perform sound, complete and efficient inference?
- Approaches
  - Generalized modus ponens
  - Forward and backward chaining
  - Resolution
- State of the art

Inference in First-Order Logic

- Carefully…

Monty Python and the Holy Grail (1975)

Inference Rules

- Notation

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

- Modus Ponens

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

Generalized Modus Ponens

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

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

  - where SUBST(θ,p_i') = SUBST(θ,p_i) for all i
- Find θ 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) = θ where SUBST(θ,p) = SUBST(θ,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

- Standarize variables 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}

Unification

- 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

Unification Examples

Term 1	Term 2	Substitution (or fail)
Glitter(x,y)	Glitter(3,3)	
Adjacent(x,2,2,y)	Adjacent(2,y,x,3)	
At(w,x,y)	At(Wumpus,u,3)	
At(Agent,x,Row(Wumpus))	At(z,3,Row(w))	
At(w,Column(w),y)	At(Agent,Column(Wumpus),3)	

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., $\forall x,y \neg A(x) \vee \neg B(y) \vee C(x,y) \equiv \forall x,y\ A(x) \wedge B(y) \Rightarrow C(x,y)$

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

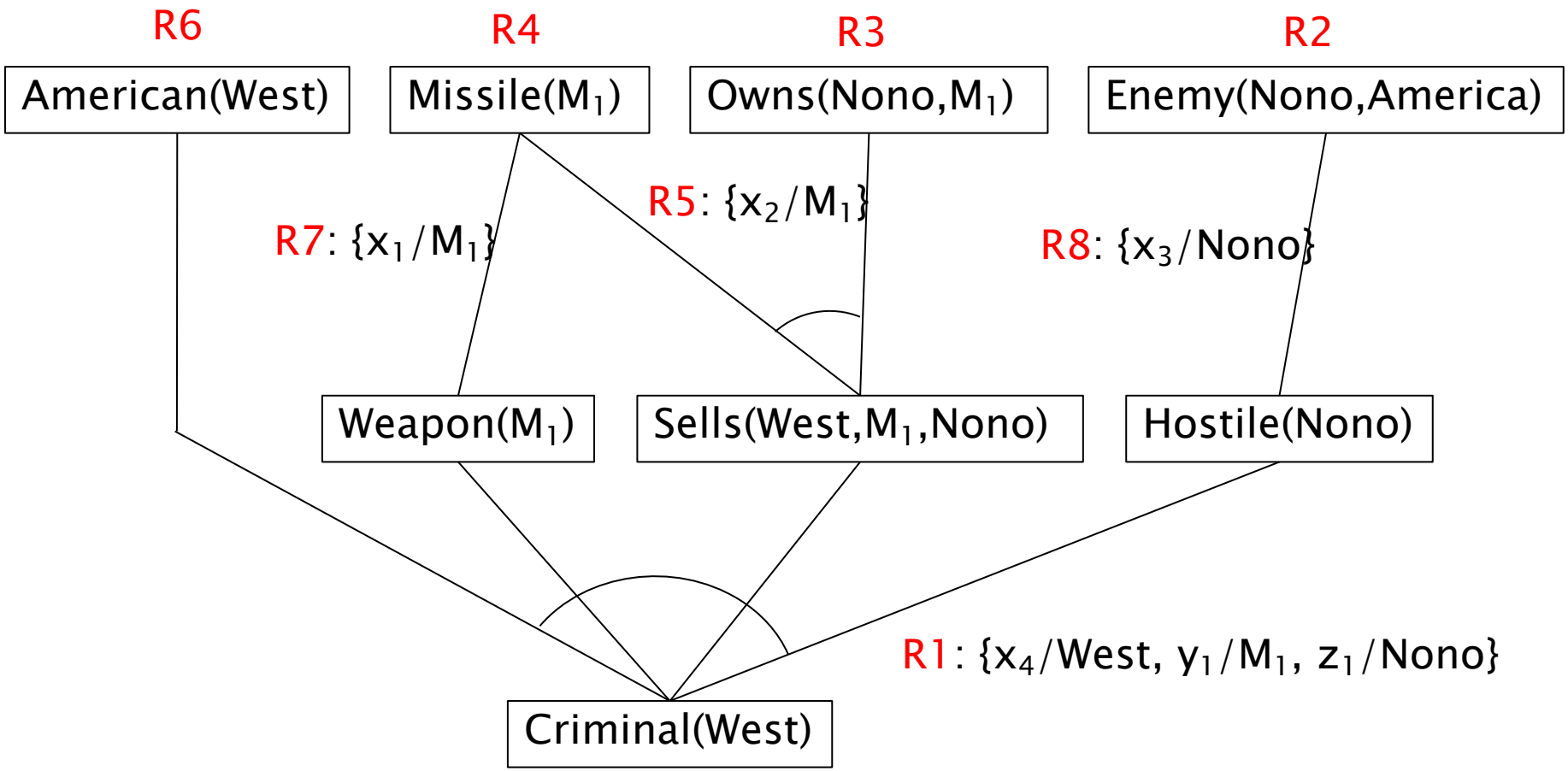
- "… it is a crime for an American to sell weapons to hostile nations."
  - R1: $\forall x,y,z\ \text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x,y,z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- "… Nono, an enemy of America, …"
  - R2: $\text{Enemy}(\text{Nono},\text{America})$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

} Existential Instantiation

Example

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

Example: Forward Chaining



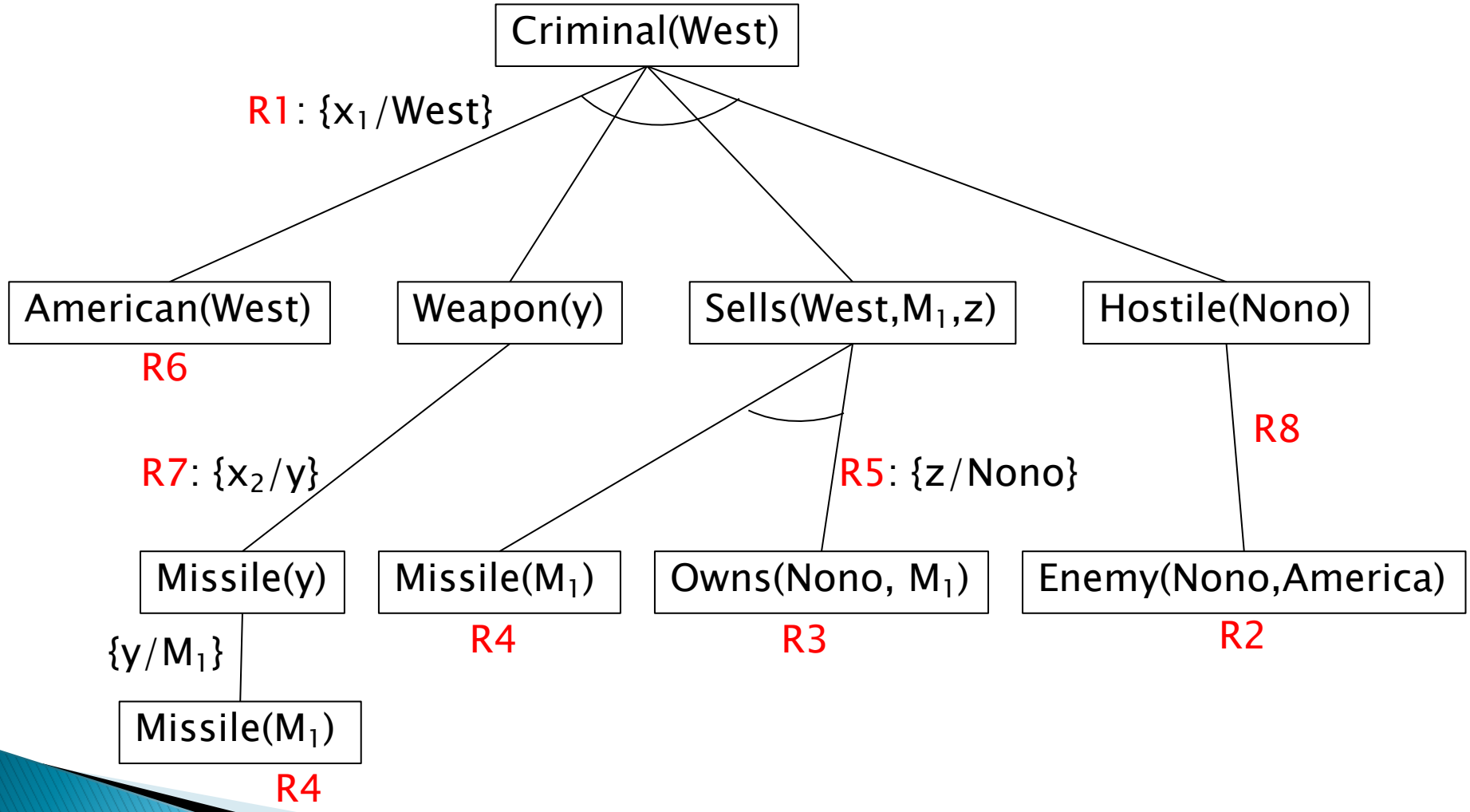
Forward Chaining

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all known facts
    - Conjunct ordering
    - R5: $\forall x$ 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

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all **open goals**
    - More constraints
    - R5: $\forall x\ Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
  - 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 in Prolog

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

- $l's$ and $m's$ are literals
- A clause is a disjunction of literals
- Resolution takes two clauses and infers a new clause

FOL Resolution

- Resolution using refutation (proof by contradiction) is sound and complete
  - KB = {¬A(x) ∨ B(x), A(Wumpus)}
  - Prove: B(wumpus)
  - Add negated goal to KB: ¬B(wumpus)
  - Search for contradition using resolution
    - If result ever empty clause, then proven
  - Resolve original clauses: B(wumpus), θ = {x/wumpus}
  - Resolve B(wumpus) and ¬B(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
  - $\forall x,y,z\ American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
  - $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(x) \vee Criminal(x)$

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: $\exists x\ P(x)$, To: $P(F1)$
  - 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 ∀x,y,…
- Step 6: Distribute ∨ over ∧
  - From: (A(x) ∧ B(x)) ∨ C(x)
  - To: (A(x) ∨ C(x)) ∧ (B(x) ∨ 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)

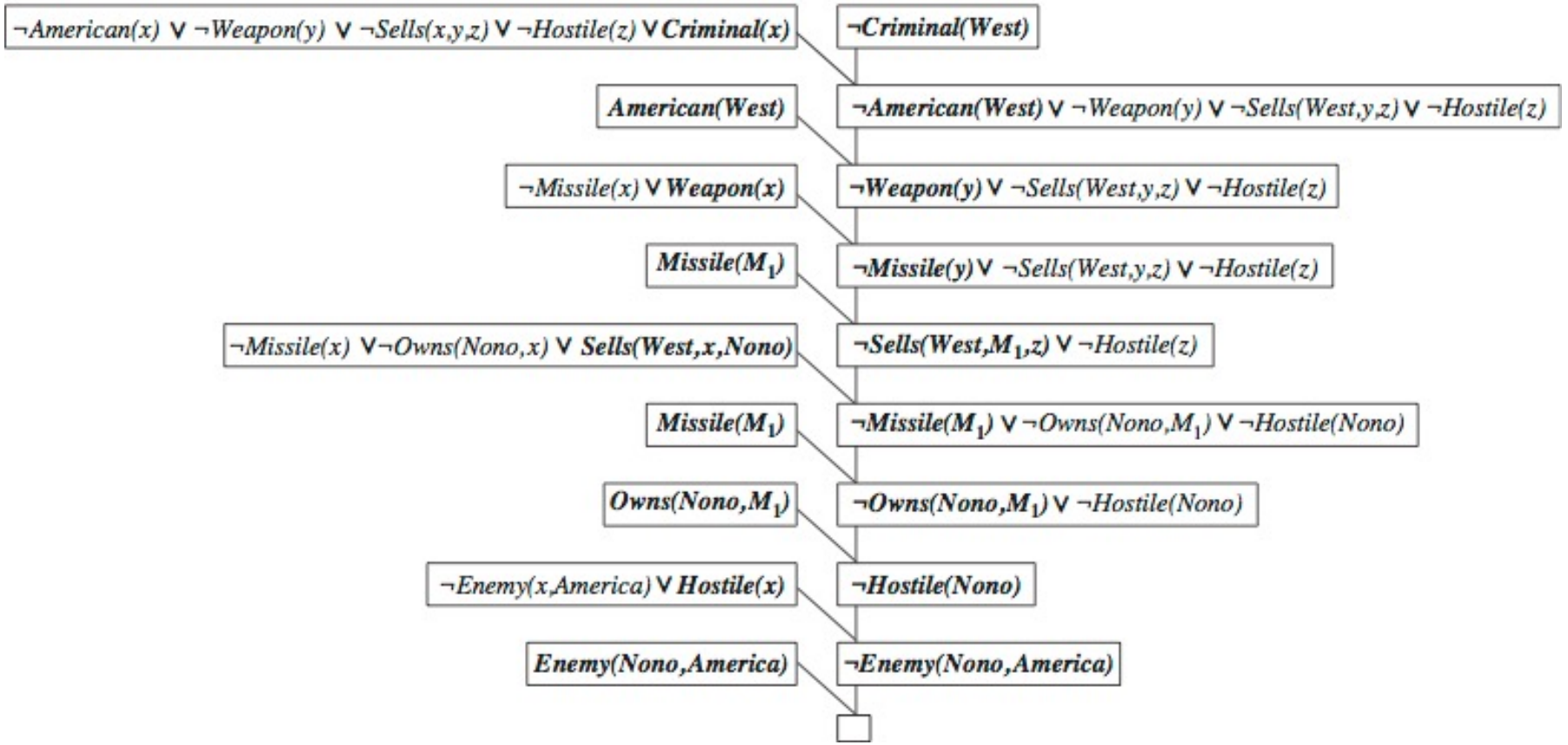
- Step 6: Distribute $\vee$ over $\wedge$

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

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)

Another Example

- Willy is a wumpus. Swiper is not a wumpus, and Swiper is not a dog. If something is neither a wumpus nor a dog, then it is a fox. Foxes jump over Wumpuses.
- Prove that Swiper jumps over Willy.

Another Example (cont.)

Another Example (cont.)

Theorem Proving: State of the Art

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

Summary

- Knowledge-based (logical) agent
- First-order logic
- Inference
  - Resolution proof by refutation