Logic and Reasoning

propositional logic, first-order logic

CS 4100/5100
Foundations of AI

Announcements

- Assignment 1 out
  - Due September 27th, 6pm
- Piazza
- Blackboard
- Reading Responses

PROPOSITIONAL LOGIC

Knowledge-Based Agents

Understanding of the World

if it is raining then the ground is wet
if the sprinkler is on then the ground is wet
if the sprinkler is on then it isn't raining

Knowledge-Based Agents

Understanding of the World

if it is raining then the ground is wet
if the sprinkler is on then the ground is wet
if the sprinkler is on then it isn't raining

Percepts

the sprinkler is on

Knowledge-Based Agents

Understanding of the World

if it is raining then the ground is wet
if the sprinkler is on then the ground is wet
if the sprinkler is on then it isn't raining

Percepts

the sprinkler is on

Updated Understanding of the World

it isn't raining
the ground is wet

Propositional Logic

Understanding of the World

raining -> ground_wet

sprinkler -> ground_wet

sprinkler -> not raining

Propositional Logic

Understanding of the World

raining -> ground_wet
sprinkler -> ground_wet
sprinkler -> not raining

Percepts

sprinkler

Propositional Logic

Understanding of the World

raining -> ground_wet
sprinkler -> ground_wet
sprinkler -> not raining

Percepts

sprinkler

Updated Understanding of the World

not raining
ground_wet

Possible Worlds

sprinkler	raining	ground_wet	raining -> ground_wet	sprinkler -> ground_wet	sprinkler -> not raining
T	T	T	T	T	F
T	T	F	F	F	F
T	F	T	T	T	T
T	F	F	T	F	T
F	T	T	T	T	T
F	T	F	F	T	T
F	F	T	T	T	T
F	F	F	T	T	T

Possible Worlds – Perceive Sprinkler

sprinkler	raining	ground_wet	raining -> ground_wet	sprinkler -> ground_wet	sprinkler -> not raining
T	T	T	T	T	F
T	T	F	F	F	F
T	F	T	T	T	T
T	F	F	T	F	T
F	T	T	T	T	T
F	T	F	F	T	T
F	F	T	T	T	T
F	F	F	T	T	T

Possible Worlds

sprinkler	raining	ground_wet	raining -> ground_wet	sprinkler -> ground_wet	sprinkler -> not raining
T	T	T	T	T	F
T	T	F	F	F	F
T	F	T	T	T	T
T	F	F	T	F	T
F	T	T	T	T	T
F	T	F	F	T	T
F	F	T	T	T	T
F	F	F	T	T	T

Possible Worlds – perceive ground wet

sprinkler	raining	ground_wet	raining -> ground_wet	sprinkler -> ground_wet	sprinkler -> not raining
T	T	T	T	T	F
T	T	F	F	F	F
T	F	T	T	T	T
T	F	F	T	F	T
F	T	T	T	T	T
F	T	F	F	T	T
F	F	T	T	T	T
F	F	F	T	T	T

Theorem Proving with Logical Inference

- Faster than model checking
- Checking for **entailed** sentences through proofs

Tautology

- Sentences that are **necessarily** true

$$P \lor \sim P$$

- Sentences that must be true are **valid**

Deduction Theorem

A entails B iff the sentence A -> B is valid.

Satisfiability

A sentence X is **satisfiable** if there exists a model such that X is true.

A sentence X is **unsatisfiable** if there exists *no* model such that X is true.

Proof by Contradiction

KB entails A iff the sentence ~A ^ KB is **unsatisfiable**

Proof by Contradiction

Understanding of the World

if it is raining then the ground is wet
if the sprinkler is on then the ground is wet
if the sprinkler is on then it isn't raining

Percepts

the sprinkler is on

Claim: the ground is not wet

Proof by Contradiction

Understanding of the World

if it is raining then the ground is wet
if the sprinkler is on then the ground is wet
if the sprinkler is on then it isn't raining

Percepts

the sprinkler is on

Claim: the ground is not wet

Contradiction: if the sprinkler is on then the ground is wet

Proof by Contradiction

Understanding of the World

if it is raining then the ground is wet
if the sprinkler is on then the ground is wet
if the sprinkler is on then it isn't raining

Percepts

the sprinkler is on

Claim: the ground is not wet

Contradiction: if the sprinkler is on then the ground is wet

Conclusion: the ground is wet

Logical Equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

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

Modus Ponens

if *a* -> *b* and *a* is true, then *b* is true

KB0: raining -> ground_wet.

KB1: raining.

Modus Ponens

if *a* -> *b* and *a* is true, then *b* is true

KB0: raining -> ground_wet.

KB1: raining.

Conclusion: ground_wet.

Modus Tollens

if *a* -> *b* is true and *b* is false, then *a* is false.

KB0: raining -> ground_wet.

KB1: not ground_wet.

Modus Tollens

if *a* -> *b* is true and *b* is false, then *a* is false.

KB0: raining -> ground_wet.

KB1: not ground_wet.

Conclusion: not raining.

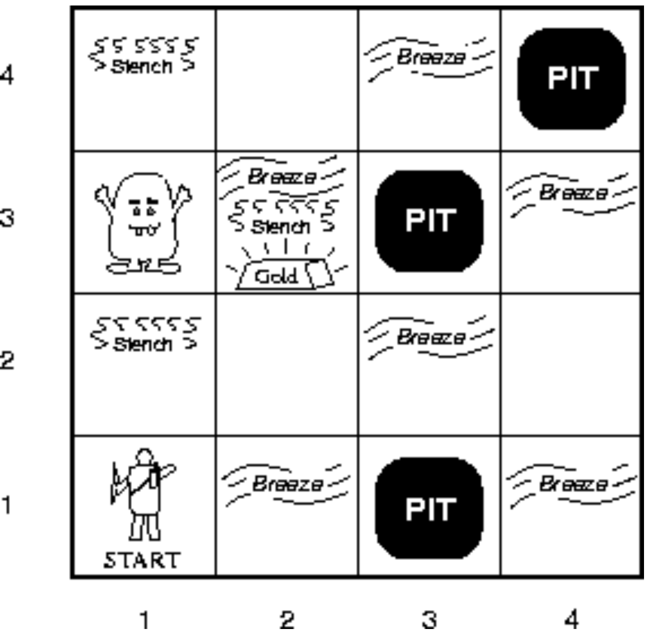
And-Elimination

if *a* ^ *b* is true, then *a* is true and *b* is true

KB: sprinkler and warm.

Conclusion: sprinkler.
warm.

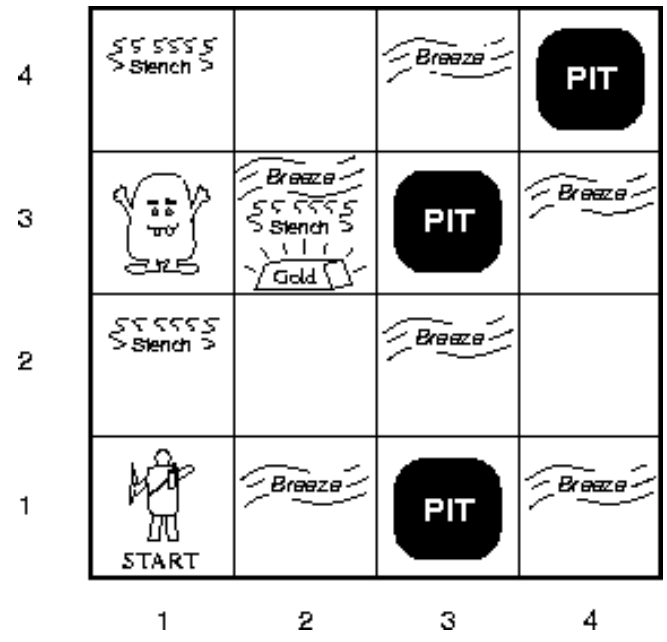
Back to Wumpus World



- **Environment**
  - 4x4 grid – agent starts at [1, 1]
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you face it
  - Shooting uses the only arrow
  - Grabbing picks up gold in the same square
  - Climbing exits the cave if at [1,1]
- **Actions**: Forward, TurnLeft, TurnRight, Grab, Shoot, Climb
- **Percepts**: Stench, Breeze, Glitter, Bump, Scream

Wumpus World: Proposition Symbols

- World Representation:
  - $P_{x,y}$
  - $W_{x,y}$
- Agent Perception:
  - $S_{x,y}$
  - $B_{x,y}$



Wumpus Inference Example

1. $\sim P_{1,1}$ *%percept*
2. $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$ *%rule*
3. $B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$ *%rule*
4. $\sim B_{1,1}$ *%percept*
5. $B_{2,1}$ *%percept*

Prove that there is no pit in [1,2].

Logical Equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

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

Resolution

Applies to two **clauses** in which there are **complementary literals**

A v B v C

~C v D v E

Resolution

Applies to two **clauses** in which there are **complementary literals**

A v B v **C**

~C v D v E

Resolution

Applies to two **clauses** in which there are **complementary literals**

A v B v ~~C~~

~~~C~~ v D v E

A v B v D v E
~~~

Proof by Resolution

1. P v Q
2. ~P v R
3. ~Q v R

Prove R.

Resolution

- Resolution on its own is enough for inferring all sentences from a knowledge base.
- …but it's only good for disjunctive clauses

Resolution

- Resolution on its own is enough for inferring all sentences from a knowledge base.
- …but it’s only good for conjunctions.
- Every sentence can be converted to **conjunctive normal form**.

Conjunctive Normal Form

- A sentence expressed purely as a conjunction of disjunctive clauses.

(A v B v C) ^ (D v E v ~A) ^ (A v C v E)

Logical Equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

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

Activity: Converting to CNF

1. P v Q -> R ^ S ?

2. $B_{1,1}$ <-> ($P_{1,2}$ v $P_{2,1}$) ?

Activity: Unicorns

- If the unicorn is mythical, then it is immortal. But if the unicorn is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal then it is horned. The unicorn is magical if it is horned.

- Can you prove the unicorn is mythical? Magical? Horned?

Horn Clauses: A Special Case

- Horn clause: a clause with at most one positive literal
  - ~A v ~B v ~C v ~D

- Definite clause: a Horn clause with exactly one positive literal
  - ~A v ~B v C v ~D

Chaining

- Horn clauses are **closed under resolution**

(~A v B v ~C)   (~B v ~D v ~E v F)

~A v ~C v ~D v ~E v F

Forward Chaining

- Start with known facts and derive new knowledge to add to the knowledge base
- Agent can derive conclusions from incoming percepts
- Data-driven approach

Forward Chaining

- Horn clauses:
  - C1: $\sim P_1 \vee \sim P_2 \vee P_4$ $(P_1 \wedge P_2 \rightarrow P_4)$
  - C2: $\sim P_4 \vee P_5$ $(P_4 \rightarrow P_5)$

- Facts:
  - P_1, P_2

Forward Chaining

- Horn clauses:
  - C1: $\sim P_1 \vee \sim P_2 \vee P_4$ $(P_1 \wedge P_2 \rightarrow P_4)$
  - C2: $\sim P_4 \vee P_5$ $(P_4 \rightarrow P_5)$
- Facts:
  - P_1, P_2
- Percepts P_1 and P_2 resolve with C1 to get P_4

Forward Chaining

- Horn clauses:
  - C1: $\sim P_1 \lor \sim P_2 \lor P_4$ $(P_1 \land P_2 \rightarrow P_4)$
  - C2: $\sim P_4 \lor P_5$ $(P_4 \rightarrow P_5)$
- Facts:
  - P_1, P_2
- Percepts P_1 and P_2 resolve with C1 to get P_4
- Resolve P_4 with C2 to get P_5

Backward Chaining

- Goal-driven reasoning
- Work backwards to see if query is true
- If inconclusive, query is false
- Efficient: only touches relevant facts or rules

Backward Chaining

- Horn clauses:
  - C1: $\sim P_1 \lor \sim P_2 \lor P_4$ $(P_1 \land P_2 \rightarrow P_4)$
  - C2: $\sim P_4 \lor P_5$ $(P_4 \rightarrow P_5)$
- Facts:
  - P_1, P_2
- Goal: P_5
- Subgoal: prove P_4

Backward Chaining

- Horn clauses:
  - C1: $\sim P_1 \vee \sim P_2 \vee P_4$ $(P_1 \wedge P_2 \rightarrow P_4)$
  - C2: $\sim P_4 \vee P_5$ $(P_4 \rightarrow P_5)$
- Facts:
  - P_1, P_2
- Goal: P_5
- Subgoal: prove P_4
  - Sub-sub goal: prove P_2
  - Sub-sub goal: prove P_1

FIRST-ORDER LOGIC

More Flexibility

- Objects
- Relations
- Functions (special kind of relation)

Some examples…

- “Squares neighboring the wumpus are smelly.”
  - Objects: Wumpus, squares
  - Relations: Smelly (property), neighboring

Some examples…

- “The father of Gillian is John.”
  - Objects: Gillian, John
  - Relations: father (also a function)
- “John is an engineer.”
  - Objects: John
  - Relations: engineer (property)

Some examples…

- "Foundations of AI is a fun class!"
  - Objects: ?
  - Relations: ?

- "Boston is cold in the winter and warm in the summer."
  - Objects: ?
  - Relations: ?

Ontological Commitments

- What is the nature of reality?
  - Objects with relationships that do not change with time
  - Relationships are true or false (or no opinion)
- Other kinds of languages
  - Temporal logic
  - Fuzzy logic
  - Higher order logic
  - Probability theory

First Order Logic - Syntax

- Constants
  - john, gillian, mary
- Predicates
  - president(america, obama)
- Functions
  - father(gillian) = john
- Variables
  - X, Y, Z…
- Connectives
  - ^ v ~ ->
- Quantifiers
  - ∀, ∃

Converting English to First Order Logic

- Stephen and Jeremy are friends.
- Sarah is a computer scientist.
- If a person is a computer scientist, then Stephen is friends with them.

Converting English to First Order Logic

- Stephen and Jeremy are friends.
  - friends(stephen, jeremy).
- Sarah is a computer scientist.

- If a person is a computer scientist, then Stephen is friends with them.

Converting English to First Order Logic

- Stephen and Jeremy are friends.
  - friends(stephen, jeremy).
- Sarah is a computer scientist.
  - computerscientist(sarah).
- If a person is a computer scientist, then Stephen is friends with them.

Converting English to First Order Logic

- Stephen and Jeremy are friends.
  - friends(stephen, jeremy).
- Sarah is a computer scientist.
  - computerscientist(sarah).
- If a person is a computer scientist, then Stephen is friends with them.
  - computerscientist(X) -> friends(stephen, X)

Converting English to First Order Logic

- The enemy of my enemy is my friend.
- All dogs go to heaven.
- There is a nice person in class.

Converting English to First Order Logic

- The enemy of my enemy is my friend.
  - enemy(X, Y) ^ enemy(Y, Z) -> friend(X, Z)
- All dogs go to heaven.
- There is a nice person in class.

Converting English to First Order Logic

- The enemy of my enemy is my friend.
  - enemy(X, Y) ^ enemy(Y, Z) -> friend(X, Z)
- All dogs go to heaven.
  - ∀x dog(x) -> afterlife(heaven, x)
- There is a nice person in class.

Converting English to First Order Logic

- The enemy of my enemy is my friend.
  - enemy(X, Y) ^ enemy(Y, Z) -> friend(X, Z)
- All dogs go to heaven.
  - ∀x dog(x) -> afterlife(heaven, x)
- There is a nice person in class.
  - ∃x classmate(x) ^ nice(x)

reasoning with first order logic

PROLOG

Prolog

- Logic programming language

- Use cases:
  - Expert systems
  - Natural language processing

- Backward chaining

Programming with Prolog

- Facts
  - monster(zombie).
  - connected(hallway, kitchen).
  - sleepy(student).
  - likes(peanuts, elephant).

- Rules
  - common_interest(X, Y) :- likes(Z, X), likes(Z, Y).
  - scary(X) :- monster(X).

Unification

- Look through knowledge base for sentence that matches the query, unify variables

- Find the unifier (θ) of unify(a, b)

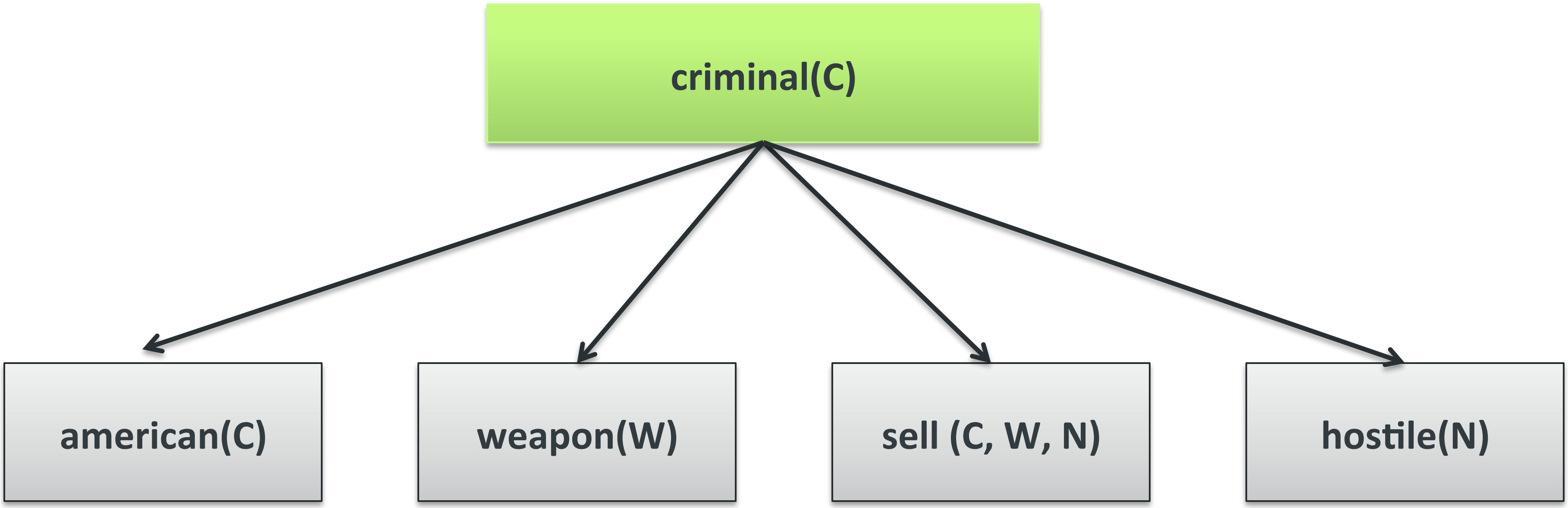
a	b	θ
knows(john, X)	knows(Y, elizabeth)	X/elizabeth, Y/john
knows(X, Y)	knows(sarah, Y)	X/sarah, Y ungrounded
knows(john, X)	knows(sarah, Y)	fail

Derivation Trees

- 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, an American.

- Is Colonel West a criminal?

Derivation Trees

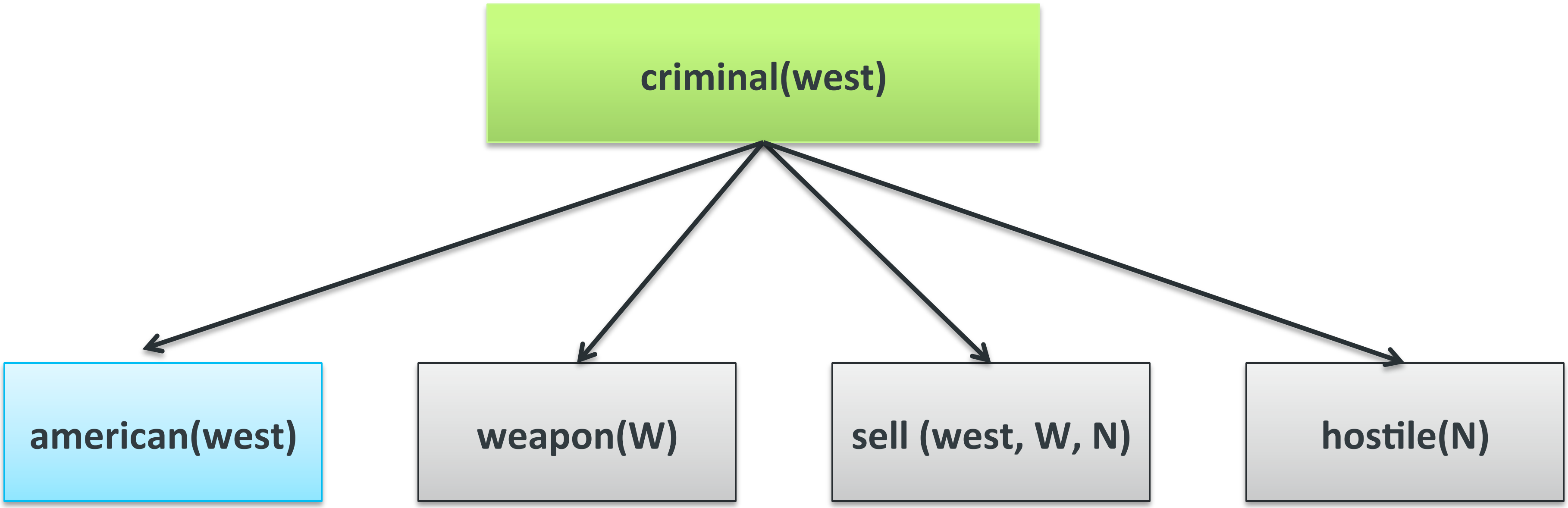


Unification: {}

Goal: criminal(west).

Rule: criminal(C) :- american(C), weapon(W), sell(C, W, N), hostile(N).

Derivation Trees

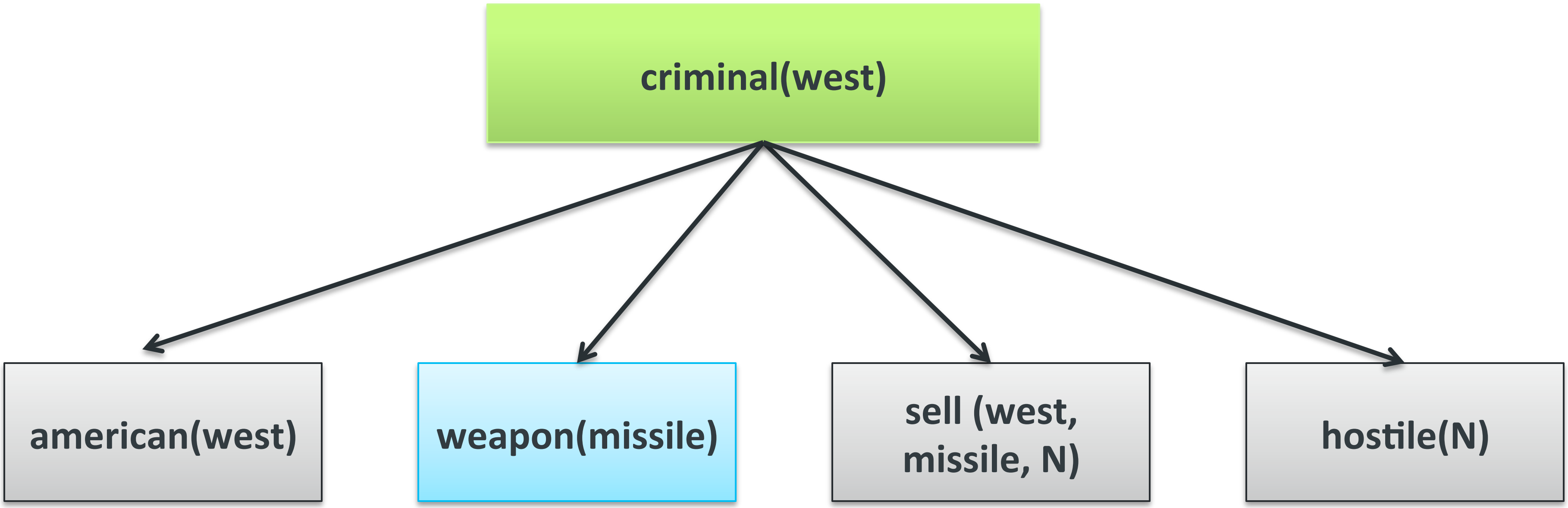


Unification: {C/west}

Goal: criminal(west).

Rule: criminal(C) :- american(C), weapon(W), sell(C, W, N), hostile(N).

Derivation Trees

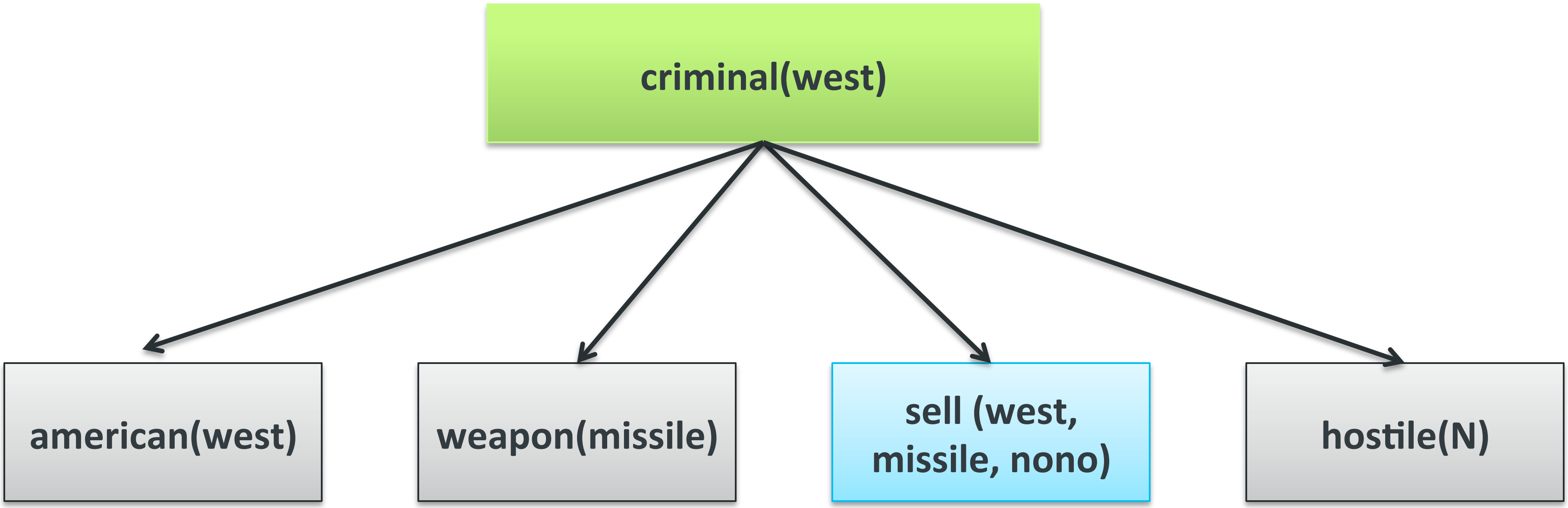


Unification: {C/west, W/missile}

Goal: criminal(west).

Rule: criminal(C) :- american(C), weapon(W), sell(C, W, N), hostile(N).

Derivation Trees

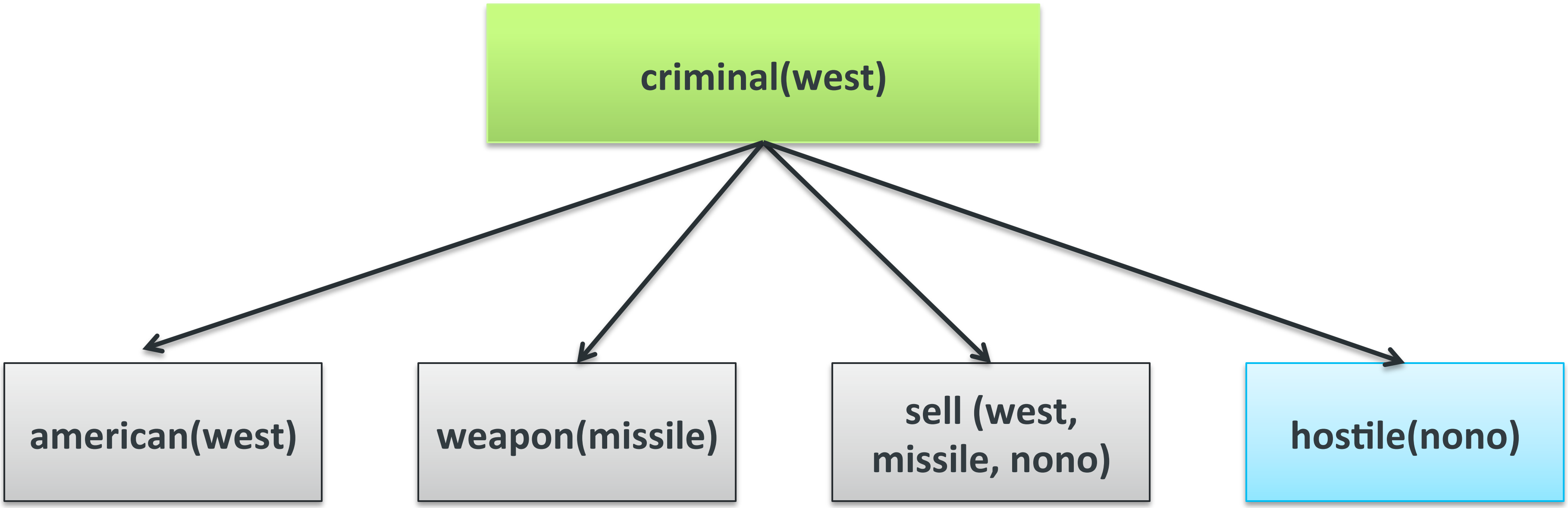


Unification: {C/west, W/missile, N/nono}

Goal: criminal(west).

Rule: criminal(C) :- american(C), weapon(W), sell(C, W, N), hostile(N).

Derivation Trees



Unification: {C/west, W/missile, N/nono}

Goal: criminal(west).

Rule: criminal(C) :- american(C), weapon(W), sell(C, W, N), hostile(N).

More Prolog...

- Arithmetic
- Lists

- There is a great tutorial linked in the assignment!

Assignment 1

- Make an adventure game in prolog