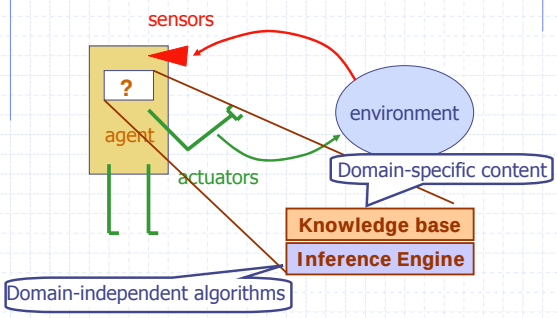
Logical Agents

Russell and Norvig: Chapter 7
CMSC 421 – Fall 2003

based on material from Marie desJardins, Jean-Claude Latombe, Stuart Russell

Knowledge-Based Agent



The Wumpus World

- The Wumpus computer game
- The agent explores a cave consisting of rooms connected by passageways.
- Lurking somewhere in the cave is the Wumpus, a beast that eats any agent that enters its room.
- Some rooms contain bottomless pits that trap any agent that wanders into the room.
- Occasionally, there is a heap of gold in a room.
- The goal is to collect the gold and exit the world without being eaten

History of "Hunt the Wumpus"

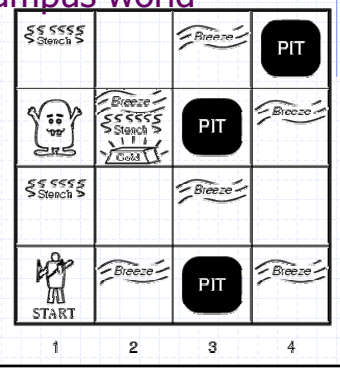
- WUMPUS /wuhm'p*s/ n. The central monster (and, in many versions, the name) of a famous family of very early computer games called "Hunt The Wumpus," dating back at least to 1972 (several years before ADVENT) on the Dartmouth Time-Sharing System. The wumpus lived somewhere in a cave with the topology of a dodecahedron's edge/vertex graph (later versions supported other topologies, including an icosahedron and Mobius strip). The player started somewhere at random in the cave with five "crooked arrows"; these could be shot through up to three connected rooms, and would kill the wumpus on a hit (later versions introduced the wounded wumpus, which got very angry). Unfortunately for players, the movement necessary to map the maze was made hazardous not merely by the wumpus (which would eat you if you stepped on him) but also by bottomless pits and colonies of super bats that would pick you up and drop you at a random location (later versions added "anaerobic termites" that ate arrows, bat migrations, and earthquakes that randomly changed pit locations).
- This game appears to have been the first to use a non-random graph-structured map (as opposed to a rectangular grid like the even older Star Trek games). In this respect, as in the dungeon-like setting and its terse, amusing messages, it prefigured ADVENT and Zork and was directly ancestral to both. (Zork acknowledged this heritage by including a super-bat colony.) Today, a port is distributed with SunOS and as freeware for the Mac. A C emulation of the original Basic game is in circulation as freeware on the net.

Wumpus PEAS description

- **Performance measure**:
  gold +1000, death -1000,
  -1 per step, -10 use arrow
- **Environment:**
  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 are facing it
  Shooting uses up the only arrow
  Grabbing picks up gold if in same square
  Releasing drops the gold in same square
- **Sensors:** Breeze, Glitter, Smell
- **Actuators:** Let turn, Right turn, Forward, Grab, Release, Shoot

A typical Wumpus world

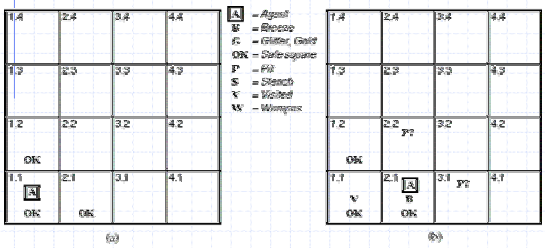
- The agent always starts in the field [1,1].
- The task of the agent is to find the gold, return to the field [1,1] and climb out of the cave.



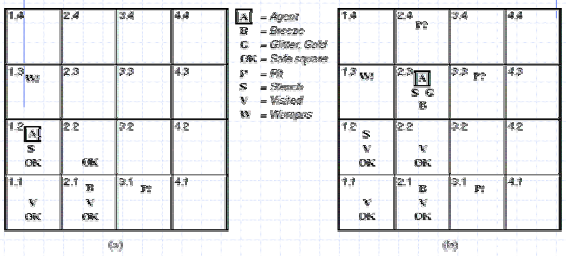
Wumpus World Characteristics

- Observable?
- Deterministic?
- Static?
- Discrete?
- Single-agent?

The Wumpus agent's first step



Later



World-wide web wumpuses

- http://www.cs.ucla.edu/~apulliam/wumpus/
- http://www.cc.gatech.edu/gvu/people/Phd/Reid.Harmon/htw/
- http://www.cs.berkeley.edu/~russell/code/doc/overview-AGENTS.html

Types of Knowledge

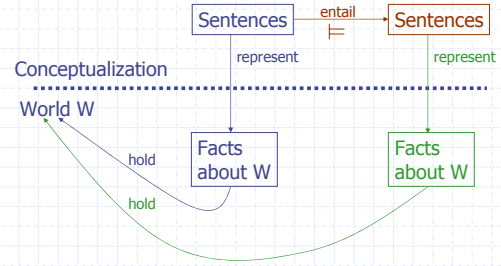
- Procedural, e.g.: functions
  Such knowledge can only be used in one way -- by executing it
- Declarative, e.g.: constraints and rules
  It can be used to perform many different sorts of inferences

Logics

Logics are formal languages for representing information such that conclusions can be drawn

- **Syntax:** defines the sentences in the language
- **Semantics:** define the "meaning" of sentences: i.e., define true of a sentence in a world
- Example: arithmetic

Connection World-Representation



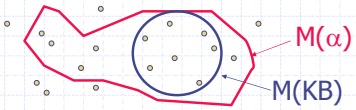
Entailment

- Entailment means that one thing follows from another, written KB ⊨ α
- A knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

⊨

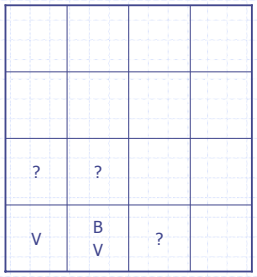
Models

- Models are formal definitions of possible states of the world
- We say m is a model of a sentence α if α is true in m
- M(α) is the set of all models of α
- Then KB⊨α if and only if M(KB) ⊆ M(α)

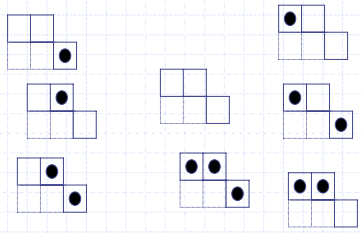


Entailment in the Wumpus World

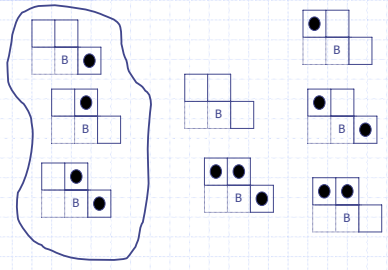
- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- What are possible worlds for ? – assume only possibilty pit or no pit.



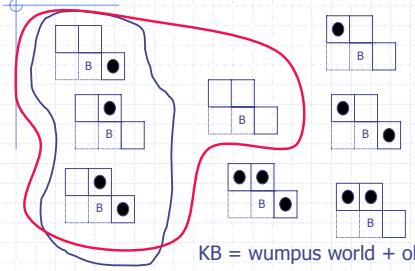
Wumpus Models



Wumpus Models



Wumpus Models

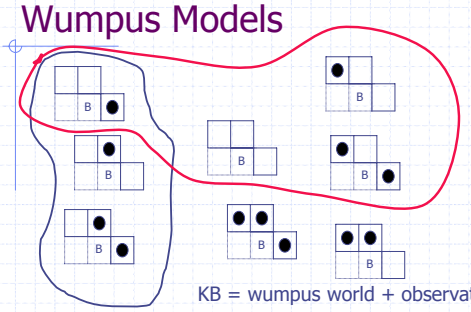


KB = wumpus world + observations

α_1="[1,2] is safe"

KB |= α_1

Wumpus Models



KB = wumpus world + observations

α_2="[2,2] is safe"

KB |= α_2 ??

Inference

- KB $|-_i \alpha$: sentence α can be derived from KB by procedure i
- **Soundness**: i is sound if whenever KB $|-_i \alpha$ it is also true that KB $\models \alpha$
- **Completeness**: i is complete if whenever KB $\models \alpha$ it is also true that KB $|-_i \alpha$

Examples of Logics

- Propositional calculus
  `A ∧ B ⇒ C`
- First-order predicate calculus
  `(∀x)(∃y) Mother(y,x)`
- Logic of Belief
  `B(John,Father(Zeus,Cronus))`

Symbols of PL

- Connectives: ¬, ∧, ∨, ⇒
- Propositional symbols, e.g., P, Q, R, …
- *True*, *False*

Syntax of PL

- sentence → atomic sentence | complex sentence
- atomic sentence → Propositional symbol, *True*, *False*
- Complex sentence → ¬sentence
  | (sentence ∧ sentence)
  | (sentence ∨ sentence)
  | (sentence ⇒ sentence)

Syntax of PL

- sentence → atomic sentence | complex sentence
- atomic sentence → Propositional symbol, *True*, *False*
- Complex sentence → ¬sentence
  | (sentence ∧ sentence)
  | (sentence ∨ sentence)
  | (sentence ⇒ sentence)

- Examples:
  - ((P ∧ Q) ⇒ R)
  - (A ⇒ B) ∨ (¬C)

Models in Propositional Logic

- Assignment of a truth value – true or false – to every atomic sentence
- Examples:
  - Let A, B, C, and D be the propositional symbols
  - is m = {A=true, B=false, C=false, D=true} a model?
  - is m' = {A=true, B=false, C=false} a model?
- How many models can be defined over n propositional symbols?

Semantics of PL

- It specifies how to determine the truth value of any sentence in a model m
- The truth value of *True* is *True*
- The truth value of *False* is *False*
- The truth value of each atomic sentence is given by m
- The truth value of every other sentence is obtained recursively by using truth tables

Truth Tables

A	B	¬ A	A ∧ B	A ∨ B	A ⇒ B
True	*True*	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*

About ⇒

- ODD(5) ⇒ CAPITAL(Japan,Tokyo)
- EVEN(5) ⇒ SMART(Sam)
- Read A ⇒ B as:
  "If A IS True, then I claim that B is True, otherwise I make no claim."

Example

Model: A= *True*, B= *False*, C= *False*, D= *True*

(¬A ∨ B ⇒ C) ⇒ D ∧ A

F
F
T
T
T

A Small Knowledge Base

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. ¬Car-OK

Sentences 1, 2, and 3 → Background knowledge

Sentences 4 and 5 → Observed knowledge

Wumpus world sentences

- P_{ij} is true if there is a pit in [i,j]
- B_{ij} is true if there is a breeze in [i,j]

  $\neg P_{11}$

  $\neg B_{11}$

  B_{21}
- "A square is breezy if and only if there is an adjacent pit" $B_{11} \Leftrightarrow P_{12} \vee P_{21}$

  $B_{21} \Leftrightarrow$???

Satisfiability of a KB

A KB is satisfiable iff it admits at least one model; otherwise it is unsatisfiable

KB1 = {P, ¬Q∧R} is satisfiable

KB2 = {¬P∨P} is satisfiable

valid sentence or tautology

KB3 = {P, ¬P} is unsatisfiable

Logical Equivalence

- Two sentences α and β are logically equivalent – written $\alpha \equiv \beta$ -- iff they have the same models, i.e.:

  $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

Logical Equivalence

- Two sentences α and β are logically equivalent – written $\alpha \equiv \beta$ -- iff they have the same models, i.e.:

  $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
- Examples:
  - $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$
  - $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
  - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
  - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$

Logical Equivalence

- Two sentences α and β are logically equivalent – written $\alpha \equiv \beta$ -- iff they have the same models, i.e.:

  $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
- Examples:
  - $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$
  - $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
  - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
  - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
- One can always replace a sentence by an equivalent one in a KB

Proof Methods

- Applications of inference rules
  - Legitimate (sound) generation of new sentences from old
  - Proof = a sequence of inference rule applications
    can use inference rules as operators in a standard search alg
  - Typically requires translation of senetences into a normal form
- Model checking
  - Truth table enumeration (exponential in n)
  - Improved backtracking
  - Heuristic search in model space (sound but incomplete)
    e.g. min-conflicts like hill-climbing algorithms

Inference Rule: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK

Example: Modus Ponens

$$\{ \alpha \Rightarrow \beta , \alpha \} \vdash \beta$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK
Headlights-Work

Forward and backward chaining

- Horn Form (restricted)
  KB = conjunction of Horn clauses
- Horn clause =
  - Propositional symbol or
  - (conjunction of symbols) ⇒ symbol
- Modus Ponens complete for Horn KBs
- Can be used with forward chaining or backward chaining. Algorithms natural and run in linear time

Forward Chaining

- Idea: Fire any rule whose premises are satisfied in the KB and add its conclusion to the KB, until query is found

FC example

- P ⇒ Q
- L ^ M ⇒ P
- B ^ L ⇒ M
- A ^ P ⇒ L
- A ^ B ⇒ L
- A
- B

Backward chaining

- Idea: work backward from the query q to prove q using BC
  - Check if q is known already, or
  - Prove by BC all premises of some rul concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - Has already been proved true
  - Has already failed

BC example

- P ⇒ Q
- L ^ M ⇒ P
- B ^ L ⇒ M
- A ^ P ⇒ L
- A ^ B ⇒ L
- A
- B

Forward vs. backward chaining

- FC is data-driven
  - Automatic, unconscious processing
  - E.g. object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving
  - E.g. Where are my keys? How do I get to my next class
  - Complexity of BC can be much less than linear in the size of the KB

General Problem:

- Given:
  - KB: a set of sentence, not necessarily horn
  - α: a sentence
- Answer:
  - KB $\models \alpha$?

Deduction vs. Satisfiability Test

KB $\models \alpha$ iff {KB,$\neg\alpha$} is unsatisfiable

Hence:
- Deciding whether a set of sentences entails another sentence
- Testing whether a set of sentences is unsatisfiable

are closely related problems

Computational Approaches

- Enumeration of models (model checking)
- Construction of a proof (inference rules)

Enumeration of Models

P: Set of propositional symbols in {KB,$\neg\alpha$}
n: Size of P

ENTAILS?(KB,α)
  For each of the 2^n models on P do
    If it is a model of {KB,$\neg\alpha$} then return no
  Return yes

Satisfiability Test as CSP

- Each propositional symbol is a variable
- The domain of each variable is {True, False}
- Each sentence in {KB,$\neg\alpha$} is a constraint on the value(s) taken by one or several variables
- Recursive backtracking CSP techniques and heuristics are applicable

Construction of a Proof

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. ¬Engine-Starts ∨ Flat-Tire ← (3+8)
13. Flat-Tire ← (11+12)

Construction of a Proof

- What do we need?
- A complete set of sound inference rules
- A complete search algorithm to decide which rule to apply next and to which sentences

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlight-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. ¬Engine-Starts ∨ Flat-Tire ← (3+8)
13. Flat-Tire ← (11+12)

Complementary Literals

- A literal is a either an atomic sentence or the negated atomic sentence, e.g.:
  P or ¬P
- Two literals are complementary if one is the negation of the other, e.g.:
  P and ¬P

Unit Resolution Rule

- Given two sentences:
  $L_1 \vee \ldots \vee L_p$ and M
  where $L_i, \ldots, L_p$ and M are all literals, and M and L_i are complementary literals
- Infer:
  $L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_p$

Examples

From:
¬Engine-Starts ∨ Car-OK
Engine-Starts

Infer:
Car-OK

Modus ponens

Engine-Starts ⇒ Car-OK

From:
¬Engine-Starts ∨ Car-OK
¬Car-OK

Infer:
¬Engine-Starts

Modus tolens

Another Example

1. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
2. Engine-Starts
3. ¬Flat-Tire
4. Flat-Tire ∨ Car-OK
5. Car-OK

Detection of Unsatisfiability

1. Car-OK
2. ¬Car-OK

3. False

Soundness of Unit Resolution

- Let m be a model of:
  $L_1 \vee \ldots \vee L_p$ and M
  where M and L_i are complementary literals
- L_i must be False in m, hence
  $L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_p$
  must be True

Shortcoming of Unit Resolution

From:

- ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
- Engine-Starts ∨ Empty-Gas-Tank

we can infer nothing!

What does this mean? Not Complete!!

Full Resolution Rule

- Given two sentences:
  $L_1 \vee \ldots \vee L_p$ and $M_1 \vee \ldots \vee M_q$
  where $L_1,\ldots, L_p, M_1,\ldots, M_q$ are all literals,
  and L_i and M_j are complementary literals
- Infer:

$L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_k \vee M_1 \vee \ldots \vee M_{j-1} \vee M_{j+1} \vee \ldots \vee M_k$

in which only one copy of each literal is retained (factoring)

Example

From:

¬Engine-Starts v Flat-Tire v Car-OK

Engine-Starts v Empty-Gas-Tank

Infer:

Flat-Tire v Car-OK v Empty-Gas-Tank

Example

From:

$\neg P \vee Q \quad (\equiv P \Rightarrow Q)$

$\neg Q \vee R \quad (\equiv Q \Rightarrow R)$

Infer:

$\neg P \vee R \quad (\equiv P \Rightarrow R)$

Not All Inferences are Useful!

From:

¬Engine-Starts ∨ Flat-Tire ∨ Car-OK

Engine-Starts ∨ ¬Flat-Tire

Infer:

¬Flat-Tire ∨ Flat-Tire ∨ Car-OK

Not All Inferences are Useful!

From:

$\neg$Engine-Starts $\vee$ Flat-Tire $\vee$ Car-OK

Engine-Starts $\vee$ $\neg$Flat-Tire

Infer:

$\neg$Flat-Tire $\vee$ Flat-Tire $\vee$ Car-OK

tautology

Not All Inferences are Useful!

From:

$\neg$Engine-Starts $\vee$ Flat-Tire $\vee$ Car-OK

Engine-Starts $\vee$ $\neg$Flat-Tire

Infer:

$\neg$Flat-Tire $\vee$ Flat-Tire $\vee$ Car-OK $\equiv$ *True*

tautology

Full Resolution Rule

- Given two sentences:
  $L_1 \vee \ldots \vee L_p$ and $M_1 \vee \ldots \vee M_q$
- Infer:
  $L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_k \vee M_1 \vee \ldots \vee M_{j-1} \vee M_{j+1} \vee \ldots \vee M_k$
- Resolution is sound and complete for proposition logic

Conversion to CNF

Example:

$B_{11} \Leftrightarrow (P_{12} \vee P_{21})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
   $(B_{11} \Rightarrow ((P_{12} \vee P_{21})) \wedge ((P_{12} \vee P_{21}) \Rightarrow B_{11}))$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg(P_{12} \vee P_{21}) \vee B_{11}))$
3. Move $\neg$ in using deMorgan's rules and double negation
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \wedge \neg P_{21}) \vee B_{11}))$
4. Distribute $\vee$ over $\wedge$
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$

Set of clauses:

$\{(\neg B_{11} \vee P_{12} \vee P_{21}), (\neg P_{12} \vee B_{11}), (\neg P_{21} \vee B_{11})\}$

Your Turn: Conversion to CNF

Example:

$(A \vee \neg B) \Rightarrow (C \wedge D)$

1. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$
   $\neg(A \vee \neg B) \vee (C \wedge D)$
2. Reduce scope of $\neg$
   $(\neg A \wedge B) \vee (C \wedge D)$
3. Distribute $\vee$ over $\wedge$
   $(\neg A \vee (C \wedge D)) \wedge (B \vee (C \wedge D))$
   $(\neg A \vee C) \wedge (\neg A \vee D) \wedge (B \vee C) \wedge (B \vee D)$

Set of clauses:

$\{\neg A \vee C, \neg A \vee D, B \vee C, B \vee D\}$

Resolution Refutation Algorithm

RESOLUTION-REFUTATION(KB,α)
clauses $\leftarrow$ set of clauses obtained from KB and $\neg\alpha$
new $\leftarrow$ {}
Repeat:
  For each C, C' in clauses do
    res $\leftarrow$ RESOLVE(C,C')
    If res contains the empty clause then return yes
    new $\leftarrow$ new U res
  If new $\subseteq$ clauses then return no
  clauses $\leftarrow$ clauses U new

Proof by contradiction, i.e., show {KB,$\neg\alpha$} is unsatisfiable

Example

1. ¬Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire ← **negated goal**
10. ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts 2,5
11. ¬Battery-OK ∨ Empty-Gas-Tank ∨ Engine-Starts 2,6
12. ¬Battery-OK ∨ ¬Starter-OK ∨ Engine-Starts 2,7
13. ¬Engine-Starts ∨ Flat-Tire 3,8
14. ¬Engine-Starts ∨ Car-OK 3,9

Resolution Refutation Ex.

1. ¬A ∨ ¬B ∨ C
2. ¬B ∨ ¬D ∨ E ∨ F
3. ¬E ∨ F ∨ C
4. D
5. B
6. G
7. ¬E
8. ¬C
9. ¬F

task: Prove F

Resolution Refutation Search Strategies

- Ordering Strategies
  - define levels
  - DFS, BFS
- Refinement Strategies
- Simplifications

Refinement Strategy

- **Set-of-support heuristic**:
  At least one ancestor of every inferred clause comes from α

Example (Set-of-Support)

1. ¬A ∨ ¬B ∨ C
2. ¬B ∨ ¬D ∨ E ∨ F
3. ¬E ∨ F ∨ C
4. D
5. B
6. G
7. ¬E
8. ¬C
9. ¬F

Example (Set-of-Support)

1. ¬A ∨ ¬B ∨ C
2. ¬B ∨ ¬D ∨ E ∨ F
3. ¬E ∨ F ∨ C
4. D
5. B
6. G
7. ¬E
8. ¬C
9. ¬F
10. ¬B ∨ ¬D ∨ E 9,2
11. ¬D ∨ E 10.5
12. E 11,4
13. false 12,7

Example (Set-of-Support)

1. ¬Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. Headlight-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire

Example (Set-of-Support)

1. ¬Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. Headlight-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire
10. ¬Engine-Starts ∨ Car-OK
11. ¬Engine-Starts
12. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank
13. ¬Starter-OK ∨ Empty-Gas-Tank
14. Empty-Gas-Tank
15. *False*

Note the goal-directed flavor

Resolution Heuristics

- **Shortest-clause heuristic:** Generate a clause with the fewest literals first

Example (Shortest-Clause)

1. ¬Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. Headlight-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire

Example (Shortest-Clause)

1. ¬Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. Headlight-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire
10. ¬Engine-Starts ∨ Car-OK
11. ¬Engine-Starts
12. ¬Bulbs-OK ∨ Headlights-Work
13. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank
14. ¬Starter-OK ∨ Empty-Gas-Tank
15. Empty-Gas-Tank
16. *False*

Resolution Heuristics

- Simplifications heuristics:
  - Remove any clause containing two complementary literals (tautology)
  - Remove any clause C that contains all the literals of another clause C'
  - If a symbol always appears with the same "sign", remove all the clauses that contain it (pure symbol)

Example (Pure Literal)

1. ~~Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work~~
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-Starts
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. ~~Headlights-Work~~
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire

Review: 2 Important Properties

- #1: If KB |- Q then KB |= Q
  - If Q is derived from a set of sentences KB using a given set of rules, then Q is entailed by KB.
  - Hence, inference produces only real entailments, or any sentence that follows deductively from the premises is valid.

Soundness

- #2: If KB |= Q then KB |- Q
  - If Q is entailed by a set of sentences KB, then Q can be derived from KB using the rules of inference.
  - Hence, inference produces all entailments, or all valid sentences can be proved from the premises.

Completeness

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - Syntax: formal structure of sentences
  - Semantics: truth of sentences wrt models
  - Entailment: necessary truth of one sentence given another
  - Inference: deriving sentences from other sentences
  - Soundness: derivations produce only entailed sentences
  - Completeness: derivations can produce all entailed sentences
- FC and BC are linear-time, complete for Horn clauses
- Resolution is complete for propositional logic

Something to Think About

What is the difference between:

$\Rightarrow$

$\vdash$

$\models$