Chapter 7
Logical Agents

CS 461 – Artificial Intelligence
Pinar Duygulu
Bilkent University, Spring 2008

Slides are mostly adapted from AIMA and MIT Open Courseware
Outline

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - forward chaining
  - backward chaining
  - resolution
Introduction

- The **representation of knowledge** and the **reasoning processes** that bring knowledge to life are central to entire field of artificial intelligence.

- Knowledge and reasoning are important to artificial agents because they enable successful behaviors that would be very hard to achieve otherwise (no piece in chess can be on two different squares at the same time).

- Knowledge and reasoning also play a crucial role in dealing with partially observable environments (inferring hidden states in diagnosing diseases, natural language understanding).

- Knowledge also allows flexibility.
Knowledge bases

- Knowledge base = set of sentences in a formal language
- Each sentence is expressed in a knowledge representation language and represents some assertions about the world
- There should be a way to add new sentences to KB, and to query what is known
- Declarative approach to building an agent (or other system):
  - TELL it what it needs to know
  - Then it can ASK itself what to do - answers should follow from the KB
- Both tasks may involve inference – deriving new sentences from old
- In logical agents – when one ASKS a question to KB, the answer should follow from what has been TELLED
- Agents can be viewed at the knowledge level
  i.e., what they know, regardless of how implemented
- Or at the implementation level
  i.e., data structures in KB and algorithms that manipulate them
A simple knowledge-based agent

function KB-AGENT( percept) returns an action

static:  $KB$, 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, $t$))

    $t ← t + 1$

return action

• KB : maintain the background knowledge

• Each time the agent program is called it does three things
  – TELLS the KB what it perceives
  – ASK the KB what action it should perform
  – TELL the KB that the action is executed

• The agent must be able to:
  – Represent states, actions, etc.
  – Incorporate new percepts
  – Update internal representations of the world
  – Deduce hidden properties of the world
  – Deduce appropriate actions

• **Declarative** versus **procedural** approaches
Wumpus World PEAS description

• **Performance measure**
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow

• **Environment**
  - Squares adjacent to wumpus are smelly (stench)
  - 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:** Stench, Breeze, Glitter, Bump, Scream

• **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot
Wumpus world characterization

- **Fully Observable** No – only local perception
- **Deterministic** Yes – outcomes exactly specified
- **Episodic** No – sequential at the level of actions
- **Static** Yes – Wumpus and Pits do not move
- **Discrete** Yes
- **Single-agent?** Yes – Wumpus is essentially a natural feature
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Logic in general

- **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 truth of a sentence in a world

- E.g., the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x^2+y > \{}$ is not a sentence
  - $x+2 \geq y$ is true iff the number $x+2$ is no less than the number $y$
  - $x+2 \geq y$ is true in a world where $x = 7$, $y = 1$
  - $x+2 \geq y$ is false in a world where $x = 0$, $y = 6$

- Possible world – **model**
- $m$ is a model of $\alpha$ – the sentence $\alpha$ is true in model $m$
Entailment

- **Entailment** means that one thing **follows from** another:

\[
\text{KB} \models \alpha
\]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( KB \) is true
  - If \( \alpha \) true then \( KB \) must also be true

  - E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
  - E.g., \( x+y = 4 \) entails \( 4 = x+y \)
  - Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**
Models

- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated.

- 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* iff \( M(KB) \subseteq M(α) \)
  - E.g. \( KB = \text{Giants won and Reds won} \)
  - \( α = \text{Giants won} \)
Entailment in the wumpus world

• Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
Wumpus models

3 Boolean choices $\Rightarrow$ 8 possible models for the adjacent squares [1,2], [2,2] and [3,1] to contain pits
Wumpus models

Consider possible models for $KB$ assuming only pits

- $KB = \text{wumpus-world rules } + \text{ observations}$
- $KB$ is false in any model in which $[1,2]$ contains a pit, because there is no breeze in $[1,1]$
Wumpus models

• Consider $\alpha_1 = \text{“[1,2] is safe” = “There is no pit in [1,2]”}
• In every model $KB$ is true $\alpha_1$ is also true
• $KB \models \alpha_1$, proved by model checking
• We can conclude that there is no pit in [1,2]
Wumpus models

- Consider $\alpha_2 = \text{"[2,2] is safe" = "There is no pit in [2,2]"}$
- In some models in which $KB$ is true $\alpha_2$ is false
- $KB \not\models \alpha_2$
- We cannot conclude that there is no pit in [2,2]
Inference

- $KB \vdash_i \alpha$ = sentence $\alpha$ can be derived from $KB$ by a procedure $i$ (an inference algorithm)

- **Soundness**: $i$ is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- An inference algorithm that derives only entailed sentences is sound or truth preserving (model checking is a sound procedure)
- **Completeness**: $i$ is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
- An inference algorithm is complete if it can derive any sentence that is entailed

- Think set of all consequences of KB as a haystack and $\alpha$ as a needle. Entailment is like the needle being in the haystack, and inference is like finding it
- An unsound inference procedure essentially makes things up as it goes along – it announces the discovery of nonexistent needles
- For completeness, a systematic examination can always decide whether the needle is in the haystack which is finite

- If $KB$ is true in the real world then any sentence $\alpha$ derived from $KB$ by a sound inference procedure is also true in real world
  - The conclusions of the reasoning process are guaranteed to be true in any world in which the premises are true
Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas

- Atomic sentences: consists of proposition symbols $P_1$, $P_2$

- Complex sentences: constructed from atomic sentences using logical connectives
  - If $S$ is a sentence, $\neg S$ is a sentence (negation)
  - If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence (conjunction)
  - If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence (disjunction)
  - If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)
Precedence

- Use parentheses to specify the precedence
- Otherwise the precedence from highest to lowest is: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- $A \Rightarrow B \Rightarrow C$ is not allowed
Propositional logic: Semantics

Semantics defines the rules for determining the truth of a sentence with respect to a particular model.

Each model specifies true/false for each proposition symbol.

E.g. \( P_{1,2} \quad P_{2,2} \quad P_{3,1} \)

false true false

\textit{True} is true in every model, \textit{False} is false in every model.

The truth value of every other proposition symbol must be specified directly in the model.

For the complex sentences,

Rules for evaluating truth with respect to a model \( m \):

\( \neg S \) is true iff \( S \) is false

\( S_1 \land S_2 \) is true iff \( S_1 \) is true and \( S_2 \) is true

\( S_1 \lor S_2 \) is true iff \( S_1 \) is true or \( S_2 \) is true

\( S_1 \Rightarrow S_2 \) is true iff \( S_1 \) is false or \( S_2 \) is true

i.e., is false iff \( S_1 \) is true and \( S_2 \) is false

\( S_1 \iff S_2 \) is true iff \( S_1 \Rightarrow S_2 \) is true and \( S_2 \Rightarrow S_1 \) is true

Important shorthand:

\( S_1 \Rightarrow S_2 \equiv \neg S_1 \lor S_2 \)

\( S_1 \iff S_2 \equiv S_1 \Rightarrow S_2 \land S_2 \Rightarrow S_1 \)
Truth tables for connectives

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Simple recursive process evaluates an arbitrary sentence, e.g.,

\[ \neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = \text{true} \land (\text{true} \lor \text{false}) = \text{true} \land \text{true} = \text{true} \]

**Implication:** if P is true then I am claiming that Q is true, otherwise I am making no claim.
The sentence is false, if P is true but Q is false.

**Biconditional:** True whenever both P→Q and Q→P is true.
(e.g. a square is breezy if and only if adjacent square has a pit: implication requires the presence of pit if there is a breeze, biconditional also requires the absence of pit if there is no breeze)
Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

Knowledge base includes:

- $R_1: \neg P_{1,1}$  \text{ No pit in [1,1]}
- $R_2: \neg B_{1,1}$  \text{ No breeze in [1.1]}
- $R_3: B_{2,1}$  \text{ Breeze in [2,1]}

- "Pits cause breezes in adjacent squares"
  
  - $R_4: B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
  - $R_5: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

$$KB = R_1 \land R_2 \land R_3 \land R_4 \land R_5$$
Inference

- Decide whether $KB \models \alpha$
- First method: enumerate the models and check that $\alpha$ is true in every model in which $KB$ is true
- $B_{1,1}, B_{2,1}, P_{1,1}, P_{1,2}, P_{2,1}, P_{3,1}$
- 7 symbols: $2^7 = 128$ possible models
Truth tables for inference

R1: \( \neg P_{1,1} \)

R2: \( \neg B_{1,1} \)

R3: \( B_{2,1} \)

R4: \( B_{1,1} \leftrightarrow (P_{1,2} \lor P_{2,1}) \)

R5: \( B_{2,1} \leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

\( KB = R1 \land R2 \land R3 \land R4 \land R5 \)

\( \alpha_1 = \neg P_{1,2} \)

\( \alpha_2 = P_{2,2} \)

\( \alpha_1 \) is true in all models that KB is true

\( \alpha_2 \) is true only in two models that KB is true, but false in the other one
Inference by enumeration

- Depth-first enumeration of all models is sound and complete

```plaintext
function TT-ENTAILS?(KB, α) returns true or false
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model) and
                             TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

- For $n$ symbols, time complexity is $O(2^n)$, space complexity is $O(n)$
Logical equivalence

- Two sentences are logically equivalent iff they are true in the same models:
- \( \alpha \equiv \beta \) iff \( \alpha \models \beta \) and \( \beta \models \alpha \)

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of} \; \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of} \; \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of} \; \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of} \; \lor \\
\neg(\neg \alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of} \; \land \; \text{over} \; \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of} \; \lor \; \text{over} \; \land
\end{align*}
\]
Validity and satisfiability

A sentence is valid if it is true in all models,
e.g., True, A ∨ ¬A, A ⇒ A, (A ∧ (A ⇒ B)) ⇒ B

Valid sentences are tautologies
Every valid sentence is equivalent to True

Validity is connected to inference via the Deduction Theorem:

\( KB \models \alpha \) if and only if \((KB \Rightarrow \alpha)\) is valid
Every valid implication sentence describes a legitimate inference

A sentence is satisfiable if it is true in some model
e.g., A v B, C

If a sentence is true in a model m, then we say m satisfies the sentence, or a model of the sentence

A sentence is unsatisfiable if it is true in no models
e.g., A ∧ ¬A

α is valid iff ¬α is unsatisfiable, α is satisfiable iff ¬α is not valid

Satisfiability is connected to inference via the following:

\( KB \models \alpha \) if and only if \((KB \land \lnot \alpha)\) is unsatisfiable
### Examples

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Valid?</th>
<th>Interpretation that make sentence’s truth value = f</th>
</tr>
</thead>
<tbody>
<tr>
<td>smoke → smoke</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>smoke ∨ ¬smoke</td>
<td></td>
<td></td>
</tr>
<tr>
<td>smoke → fire</td>
<td>satisfiable, not valid</td>
<td>smoke = t, fire = f</td>
</tr>
<tr>
<td>(s → f) → (¬ s → ¬ f)</td>
<td>satisfiable, not valid</td>
<td>s = f, f = t</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s → f = t, ¬ s → ¬ f = f</td>
</tr>
<tr>
<td>contrapositive</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>(s → f) → (¬ f → ¬ s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b ∨ d ∨ (b → d)</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>b ∨ d ∨ ¬b ∨ d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Satisfiability

• Related to constraint satisfaction
• Given a sentence S, try to find an interpretation \( i \) where S is true
• Analogous to finding an assignment of values to variables such that the constraint hold
• Example problem: scheduling nurses in a hospital
  – Propositional variables represent for example that Nurse1 is working on Tuesday at 2
  – Constraints on the schedule are represented using logical expressions over the variables
• Brute force method: enumerate all interpretations and check
Example Problem

Imagine we knew that:

- If today is sunny, then Tomas will be happy \((S \rightarrow H)\)
- If Tomas is happy, the lecture will be good \((H \rightarrow G)\)
- Today is sunny \((S)\)

Should we conclude that the lecture will be good?
Checking Interpretations

• Start by figuring out what set of interpretations make our original sentences true.

• Then, if G is true in all those interpretations, it must be OK to conclude it from the sentences we started out with (our knowledge base).

• In a universe with only three variables, there are 8 possible interpretations in total.
Checking Interpretations

• Only one of these interpretations makes all the sentences in our knowledge base true:
• $S = \text{true}, \ H = \text{true}, \ G = \text{true}$.
Checking Interpretations

- it's easy enough to check that G is true in that interpretation, so it seems like it must be reasonable to draw the conclusion that the lecture will be good.
Computing entailment

A knowledge base (KB) \textit{entails} a sentence S iff every interpretation that makes KB true also makes S true.

- enumerate all interpretations
- select those in which all elements of KB are true
- check to see if S is true in all of those interpretations
Entailment and Proof

A proof is a way to test whether a KB entails a sentence, without enumerating all possible interpretations.
Proof

- Proof is a sequence of sentences
- First ones are premises (KB)
- Then, you can write down on the next line the result of applying an inference rule to previous lines
- When S is on a line, you have proved S from KB

- If inference rules are sound, then any S you can prove from KB is entailed by KB

- If inference rules are complete, then any S that is entailed by KB can be proved from KB
Logical equivalence

\[(\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of } \land\]
\[(\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor\]
\[((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land\]
\[((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor\]
\[\neg(\neg \alpha) \equiv \alpha \quad \text{double-negation elimination}\]
\[(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition}\]
\[(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination}\]
\[(\alpha \iff \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}\]
\[\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan}\]
\[\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan}\]
\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor\]
\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land\]
Natural deduction

\[ \alpha \leftrightarrow \beta \]

\[ \frac{(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)}{\alpha \leftrightarrow \beta} \]

Biconditional Elimination

Some inference rules:

\[ \alpha \rightarrow \beta \]

\[ \frac{\alpha}{\beta} \]

Modus ponens

\[ \alpha \rightarrow \beta \]

\[ \frac{\neg \beta}{\neg \alpha} \]

Modus tollens

\[ \alpha \]

\[ \frac{\beta}{\alpha \land \beta} \]

And-introduction

\[ \alpha \land \beta \]

\[ \frac{\alpha}{\alpha} \]

And-elimination
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \land Q$</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>$P \rightarrow R$</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>$(Q \land R) \rightarrow S$</td>
<td>Given</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P \land Q</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>P \rightarrow R</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>(Q \land R) \rightarrow S</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>P</td>
<td>1 And-Elim</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \land Q$</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>$P \rightarrow R$</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>$(Q \land R) \rightarrow S$</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>$P$</td>
<td>1 And-Elim</td>
</tr>
<tr>
<td>5</td>
<td>$R$</td>
<td>4, 2 Modus Ponens</td>
</tr>
</tbody>
</table>
Example

Prove $S$

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \land Q$</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>$P \rightarrow R$</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>$(Q \land R) \rightarrow S$</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>$P$</td>
<td>1 And-Elim</td>
</tr>
<tr>
<td>5</td>
<td>$R$</td>
<td>4,2 Modus Ponens</td>
</tr>
<tr>
<td>6</td>
<td>$Q$</td>
<td>1 And-Elim</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \land Q$</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>$P \rightarrow R$</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>$(Q \land R) \rightarrow S$</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>$P$</td>
<td>1 And-Elim</td>
</tr>
<tr>
<td>5</td>
<td>$R$</td>
<td>4, 2 Modus Ponens</td>
</tr>
<tr>
<td>6</td>
<td>$Q$</td>
<td>1 And-Elim</td>
</tr>
<tr>
<td>7</td>
<td>$Q \land R$</td>
<td>5, 6 And-Intro</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P ∧ Q</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>P → R</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>(Q ∧ R) → S</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>P</td>
<td>1 And-Elim</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>4,2 Modus Ponens</td>
</tr>
<tr>
<td>6</td>
<td>Q</td>
<td>1 And-Elim</td>
</tr>
<tr>
<td>7</td>
<td>Q ∧ R</td>
<td>5,6 And-Intro</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>7,3 Modus Ponens</td>
</tr>
</tbody>
</table>
Example from Wumpus World

R1: \( \neg P_{1,1} \)
R2: \( \neg B_{1,1} \)
R3: \( B_{2,1} \)
R4: \( B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1}) \)
R5: \( B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

\( KB = R1 \land R2 \land R3 \land R4 \land R5 \)

Prove \( \alpha_1 = \neg P_{1,2} \)
Example from Wumpus World

R1: ¬P_{1,1}
R2: ¬B_{1,1}
R3: B_{2,1}
R4: B_{1,1} ⇔ (P_{1,2} ∨ P_{2,1})
R5: B_{2,1} ⇔ (P_{1,1} ∨ P_{2,2} ∨ P_{3,1})
R6: B_{1,1} ⇔ (B_{1,1} ⇒ (P_{1,2} ∨ P_{2,1})) ∧ ((P_{1,2} ∨ P_{2,1}) ⇒ B_{1,1})  Biconditional elimination
Example from Wumpus World

R1: \( \neg P_{1,1} \)

R2: \( \neg B_{1,1} \)

R3: \( B_{2,1} \)

R4: \( B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \)

R5: \( B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

R6: \( B_{1,1} \iff (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)  Biconditional elimination

R7: \( ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)  And Elimination
Example from Wumpus World

R1: \( \neg P_{1,1} \)
R2: \( \neg B_{1,1} \)
R3: \( B_{2,1} \)
R4: \( B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \)
R5: \( B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)
R6: \( B_{1,1} \iff (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \) Biconditional elimination
R7: \( ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \) And Elimination
R8: \( (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \) Equivalence for contrapositives
Example from Wumpus World

R1: ¬P_{1,1}
R2: ¬B_{1,1}
R3: B_{2,1}
R4: B_{1,1} ⇔ (P_{1,2} ∨ P_{2,1})
R5: B_{2,1} ⇔ (P_{1,1} ∨ P_{2,2} ∨ P_{3,1})
R6: B_{1,1} ⇔ (B_{1,1} ⇒ (P_{1,2} ∨ P_{2,1})) ∧ ((P_{1,2} ∨ P_{2,1}) ⇒ B_{1,1}) Biconditional elimination
R7: ((P_{1,2} ∨ P_{2,1}) ⇒ B_{1,1}) And Elimination
R8: (¬B_{1,1} ⇒ ¬(P_{1,2} ∨ P_{2,1})) Equivalence for contrapositives
R9: ¬(P_{1,2} ∨ P_{2,1}) Modus Ponens with R2 and R8
Example from Wumpus World

R1: \(\neg P_{1,1}\)
R2: \(\neg B_{1,1}\)
R3: \(B_{2,1}\)
R4: \(B_{1,1} \iff (P_{1,2} \lor P_{2,1})\)
R5: \(B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})\)
R6: \(B_{1,1} \iff (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})\)  Biconditional elimination
R7: \(((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})\)  And Elimination
R8: \(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})\)  Equivalence for contrapositives
R9: \(\neg (P_{1,2} \lor P_{2,1})\)  Modus Ponens with R2 and R8
R10: \(\neg P_{1,2} \land \neg P_{2,1}\)  De Morgan’s Rule
Example from Wumpus World

R1: $\neg P_{1,1}$
R2: $\neg B_{1,1}$
R3: $B_{2,1}$
R4: $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
R5: $B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
R6: $B_{1,1} \iff (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
R7: $((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1}))$
R9: $\neg (P_{1,2} \lor P_{2,1})$
R10: $\neg P_{1,2} \land \neg P_{2,1}$
R11: $\neg P_{1,2}$
Monotonicity

- The set of entailed sentences can only increase as information is added to the knowledge base

- If
  - $\text{KB} \models \alpha$
  - Then
  - $\text{KB} \land \beta \models \alpha$
Proof systems

- There are many natural deduction systems; they are typically “proof checkers”, sound but not complete.
- Natural deduction uses lots of inference rules which introduces a large branching factor in the search for a proof.
- In general, you need to do “proof by cases” which introduces even more branching.

Prove R

<table>
<thead>
<tr>
<th></th>
<th>P v Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Q → R</td>
</tr>
<tr>
<td>3</td>
<td>P → R</td>
</tr>
</tbody>
</table>
Resolution

R1: \( \neg P_{1,1} \)
R2: \( \neg B_{1,1} \)
R3: \( B_{2,1} \)
R4: \( B_{1,1} \leftrightarrow (P_{1,2} \lor P_{2,1}) \)
R5: \( B_{2,1} \leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

....
R11: \( \neg B_{1,2} \)
R12: \( B_{1,2} \leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{1,3}) \)
R13: \( \neg P_{2,2} \)
R14: \( \neg P_{1,3} \)
R15: \( (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)  
\hspace{1cm} \text{biconditional elimination on R3, followed by a Modus Ponens with R5}
R16: \( (P_{1,1} \lor P_{3,1}) \)  
\hspace{1cm} \text{Resolution with \( \neg P_{2,2} \) in R13}

If there is a pit in one of \([1,1]\), \([2,2]\) and \([3,1]\) and it is not in \([2,2]\) then it is in \([1,1]\) or \([3,1]\)

R17: \( P_{3,1} \)  
\hspace{1cm} \text{Resolve with \( \neg P_{1,1} \) in R1}
Resolution

- Resolution rule:
  \[
  \alpha \lor \beta \\
  \neg \beta \lor \gamma \\
  \hline
  \alpha \lor \gamma
  \]

- Single inference rule is a sound and complete proof system

- Requires all sentences to be converted to conjunctive normal form
Conjunctive Normal form

- Conjunctive normal form (CNF) formulas:

\[(A \lor B \lor \neg C) \land (B \lor D) \land (\neg A) \land (B \lor C)\]

- \((A \lor B \lor \neg C)\) is a clause, which is a disjunction of literals.
- \(A, B, \) and \(\neg C\) are literals, each of which is a variable or the negation of a variable.
- Each clause is a requirement that must be satisfied and can be satisfied in multiple ways.
- Every sentence in propositional logic can be written in CNF.
Converting to CNF

1. Eliminate arrows using definitions
2. Drive in negations using De Morgan’s Laws
   \[ \neg(\phi \lor \varphi) \equiv \neg\phi \land \neg\varphi \]
   \[ \neg(\phi \land \varphi) \equiv \neg\phi \lor \neg\varphi \]
3. Distribute or over and
   \[ A \lor (B \land C) \equiv (A \lor B) \land (A \lor C) \]
4. Every sentence can be converted to CNF, but it may grow exponentially in size
CNF Conversion Example

\[(A \lor B) \rightarrow (C \rightarrow D)\]

1. Eliminate arrows
\[\neg(A \lor B) \lor (\neg C \lor D)\]

2. Drive in negations
\[(\neg A \land \neg B) \lor (\neg C \lor D)\]

3. Distribute
\[(\neg A \lor \neg C \lor D) \land (\neg B \lor \neg C \lor D)\]
Resolution

- Resolution rule:
  \[ \alpha \lor \beta \\
  \neg \beta \lor \gamma \\
  \alpha \lor \gamma \]

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can’t apply any more
  - Resolution refutation is sound and complete
    - If we derive a contradiction, then the conclusion follows from the axioms
    - If we can’t apply any more, then the conclusion cannot be proved from the axioms.
Example

Prove $R$

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \lor Q$</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>$\neg P \lor R$</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>$\neg Q \lor R$</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>$\neg R$</td>
<td>Negated conclusion</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P \lor Q</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>\neg P \lor R</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>\neg Q \lor R</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>\neg R</td>
<td>Negated conclusion</td>
</tr>
<tr>
<td>5</td>
<td>Q \lor R</td>
<td>1,2</td>
</tr>
<tr>
<td>6</td>
<td>\neg P</td>
<td>2,4</td>
</tr>
<tr>
<td>7</td>
<td>\neg Q</td>
<td>3,4</td>
</tr>
<tr>
<td>8</td>
<td>R</td>
<td>5,7</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>4,8</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P v Q</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>\neg P v R</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>\neg Q v R</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>\neg R</td>
<td>Negated conclusion</td>
</tr>
<tr>
<td>5</td>
<td>Q v R</td>
<td>1, 2</td>
</tr>
<tr>
<td>6</td>
<td>\neg P</td>
<td>2, 4</td>
</tr>
<tr>
<td>7</td>
<td>\neg Q</td>
<td>3, 4</td>
</tr>
<tr>
<td>8</td>
<td>R</td>
<td>5, 7</td>
</tr>
<tr>
<td>9</td>
<td>.</td>
<td>4, 8</td>
</tr>
</tbody>
</table>
The power of false

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>\neg P</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>\neg Z</td>
<td>Negated conclusion</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>1,2</td>
</tr>
</tbody>
</table>

Note that \((P \land \neg P) \rightarrow Z\) is **valid**

Any conclusion follows from a contradiction – and so strict logic systems are very brittle.
Example

Prove R

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(P → Q) → Q</td>
</tr>
<tr>
<td>2</td>
<td>(P → P) → R</td>
</tr>
<tr>
<td>3</td>
<td>(R → S) → \neg(S → Q)</td>
</tr>
</tbody>
</table>

Convert to CNF

- \neg(\neg P v Q) v Q
- (P \land \neg Q) v Q
- (P v Q) \land (\neg Q v Q)
- (P v Q)

- \neg(\neg P v P) v R
- (P \land \neg P) v R
- (P v R) \land (\neg P v R)

- \neg(R v S) v \neg(\neg S v Q)
- (R \land \neg S) v (S \land \neg Q)
- (R v S) \land (\neg S v S) \land (R v \neg Q) \land (\neg S v \neg Q)
- (R v S) \land (R v \neg Q) \land (\neg S v \neg Q)
## Example

Prove $R$

<table>
<thead>
<tr>
<th></th>
<th>$P \lor Q$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \lor R$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\neg P \lor R$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$R \lor S$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$R \lor \neg Q$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$\neg S \lor \neg Q$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$\neg R$</td>
<td>$\text{Neg}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$S$</th>
<th></th>
<th>4,7</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>$\neg Q$</td>
<td></td>
<td>6,8</td>
</tr>
<tr>
<td>9</td>
<td>$P$</td>
<td></td>
<td>1,9</td>
</tr>
<tr>
<td>10</td>
<td>$R$</td>
<td></td>
<td>3,10</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>7,11</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

CS461 Artificial Intelligence © Pinar Duygulu
Resolution

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals

clauses

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

- **Resolution** inference rule (for CNF):

\[
\begin{array}{c}
\overline{l_i \lor \ldots \lor l_k, \quad m_1 \lor \ldots \lor m_n} \\
\overline{l_i \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n}
\end{array}
\]

where \(l_i\) and \(m_j\) are complementary literals.

*E.g.,* \(P_{1,3} \lor P_{2,2}, \neg P_{2,2}\) \(\overline{P_{1,3}}\)

- Resolution is sound and complete for propositional logic
Conversion to CNF

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]

3. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)\).
   \[ (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \).
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

3. Move \( \neg \) inwards using de Morgan's rules and double-negation:
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1}) \]

4. Apply distributivity law (\( \land \) over \( \lor \)) and flatten:
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]
Resolution algorithm

- Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

```python
function PL-Resoluation(KB, \alpha) returns true or false

    clauses ← the set of clauses in the CNF representation of $KB \land \neg \alpha$
    new ← {}
    loop do
        for each $C_i, C_j$ in clauses do
            resolvents ← PL-Resolve($C_i, C_j$)
            if resolvents contains the empty clause then return true
            new ← new \cup resolvents
        if new ⊆ clauses then return false
        clauses ← clauses \cup new
```
Resolution example

- \( KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \quad \alpha = \neg P_{1,2} \)
Forward and backward chaining

- **Horn Form** (restricted)
  - **KB** = conjunction of Horn clauses
  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) ⇒ symbol
  - E.g., C ∧ (B ⇒ A) ∧ (C ∧ D ⇒ B)
- **Modus Ponens** (for Horn Form): complete for Horn KBs
  \[ \alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta \]
  \[ \beta \]

- Can be used with **forward chaining** or **backward chaining**.
- These algorithms are very natural and run in **linear** time.
Forward chaining

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

$$\begin{align*}
P & \implies Q \\
L \land M & \implies P \\
B \land L & \implies M \\
A \land P & \implies L \\
A \land B & \implies L \\
A & \\
B &
\end{align*}$$
Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
    local variables: count, a table, indexed by clause, initially the number of premises
                    inferred, a table, indexed by symbol, each entry initially false
                    agenda, a list of symbols, initially the symbols known to be true

    while agenda is not empty do
        p ← POP(agenda)
        unless inferred[p] do
            inferred[p] ← true
            for each Horn clause c in whose premise p appears do
                decrement count[c]
                if count[c] = 0 then do
                    if HEAD[c] = q then return true
                    PUSH(HEAD[c], agenda)
                end do
            end for
        end unless
    end while
    return false
```

- For
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]

Diagram:

- Node labeled with \( P \) connected to 1.
- Node labeled with \( L \) connected to 2.
- Node labeled with \( M \) connected to 2.
- Node labeled with \( B \) and \( A \) connected to 2.
- Node labeled with \( Q \) at the top.
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Proof of completeness

- FC derives every atomic sentence that is entailed by $KB$
  1. FC reaches a fixed point where no new atomic sentences are derived
  2. Consider the final state as a model $m$, assigning true/false to symbols
  3. Every clause in the original $KB$ is true in $m$
     \[ a_1 \land \ldots \land a_k \Rightarrow b \]
  4. Hence $m$ is a model of $KB$
  5. If $KB \models q$, $q$ is true in every model of $KB$, including $m$
Backward chaining

Idea: work backwards from the query \( q \):

- to prove \( q \) by BC,
  - check if \( q \) is known already, or
  - prove by BC all premises of some rule concluding \( q \)

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

1. has already been proved true, or
2. has already failed
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[
P \implies Q
\]
\[
L \land M \implies P
\]
\[
B \land L \implies M
\]
\[
A \land P \implies L
\]
\[
A \land B \implies L
\]
\[
A
\]
\[
B
\]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
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 into a PhD program?

- Complexity of BC can be much less than linear in size of KB
Proof methods

- Proof methods divide into (roughly) two kinds:
  
  - Application 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 algorithm
    - Typically require transformation of sentences into a normal form
  
  - Model checking
    - truth table enumeration (always exponential in $n$)
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms
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

• Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

• Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses

• Propositional logic lacks expressive power