Logical Agents

Philipp Koehn

7 March 2017
The world is everything that is the case.

Wittgenstein, Tractatus
Outline

- Knowledge-based agents
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - forward chaining
  - backward chaining
  - resolution
knowledge-based agents
Knowledge-Based Agent

- Knowledge base = set of sentences in a **formal** language

- 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

- 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 \texttt{KB-AGENT} (\textit{percept}) returns an \textit{action}

static: \textit{KB}, a knowledge base
\hspace{1cm} \textit{t}, a counter, initially 0, indicating time

\texttt{TELL(\textit{KB}, MAKE-PERCEPT-SENTENCE( \textit{percept}, \textit{t}))}
\texttt{action} ← \texttt{ASK(\textit{KB}, MAKE-ACTION-QUERY(\textit{t}))}
\texttt{TELL(\textit{KB}, MAKE-ACTION-SENTENCE(\textit{action}, \textit{t}))}
\texttt{t} ← \texttt{t + 1}

return \textit{action}

- 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
example
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
  - 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

- **Actuators** Left turn, Right turn, Forward, Grab, Release, Shoot

- **Sensors** Breeze, Glitter, Smell
Wumpus World Characterization

- 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
Tight Spot

- Breeze in (1,2) and (2,1)  
  $\implies$ no safe actions

- Assuming pits uniformly distributed,  
  (2,2) has pit w/ prob 0.86, vs. 0.31
Tight Spot

- Smell in (1,1) \[\implies\] cannot move

- Can use a strategy of coercion: shoot straight ahead
  - wumpus was there \[\implies\] dead \[\implies\] safe
  - wumpus wasn’t there \[\implies\] safe
logic in general
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 \( x = 0, \ y = 6 \).
Entailment

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

  \[ KB \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( KB \) is true.

- E.g., the KB containing “the Ravens won” and “the Jays won” entails “the Ravens won or the Jays won”.

- E.g., \( x + y = 4 \) entails \( 4 = x + y \).

- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics.

- Note: brains process syntax (of some sort)
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 \( \alpha \) if \( \alpha \) is true in \( m \).

- \( M(\alpha) \) is the set of all models of \( \alpha \).

\[ \Rightarrow \ KB \models \alpha \text{ if and only if } M(KB) \subseteq M(\alpha) \]

- E.g. \( KB = \) Ravens won and Jays won

\[ \alpha = \text{Ravens won} \]
Entailment in the Wumpus World

- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for all ?, assuming only pits
- 3 Boolean choices $\Rightarrow$ 8 possible models
Possible Wumpus Models
Valid Wumpus Models

\[ KB = \text{wumpus-world rules + observations} \]
Entailment

\( KB = \text{wumpus-world rules + observations} \)

\( \alpha_1 = "[1,2] \text{ is safe}" , \ KB \vDash \alpha_1 , \text{proved by model checking} \)
$KB = \text{wumpus-world rules + observations}$
$KB = \text{wumpus-world rules + observations}$

$\alpha_2 = "[2,2] \text{ is safe", } KB \not\models \alpha_2$
Inference

- $KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

- Consequences of $KB$ are a haystack; $\alpha$ is a needle. Entailment = needle in haystack; inference = finding it

- **Soundness:** $i$ is sound if
  whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

- **Completeness:** $i$ is complete if
  whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

- That is, the procedure will answer any question whose answer follows from what is known by the $KB$. 
propositional logic
Propositional Logic: Syntax

- Propositional logic is the simplest logic—illustrates basic ideas

- The proposition symbols $P_1, P_2$ etc are sentences

- If $P$ is a sentence, $\neg P$ is a sentence (negation)

- If $P_1$ and $P_2$ are sentences, $P_1 \land P_2$ is a sentence (conjunction)

- If $P_1$ and $P_2$ are sentences, $P_1 \lor P_2$ is a sentence (disjunction)

- If $P_1$ and $P_2$ are sentences, $P_1 \implies P_2$ is a sentence (implication)

- If $P_1$ and $P_2$ are sentences, $P_1 \iff P_2$ is a sentence (biconditional)
Propositional Logic: Semantics

- Each model specifies true/false for each proposition symbol

  E.g. \( P_{1,2} \quad P_{2,2} \quad P_{3,1} \)
  
  \( \begin{array}{c}
  true \\
  true \\
  false 
  \end{array} \)

  (with these symbols, 8 possible models, can be enumerated automatically)

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

  \[ \neg P \quad \text{is true iff} \quad P \quad \text{is false} \]
  \[ P_1 \land P_2 \quad \text{is true iff} \quad P_1 \quad \text{is true} \quad \text{and} \quad P_2 \quad \text{is true} \]
  \[ P_1 \lor P_2 \quad \text{is true iff} \quad P_1 \quad \text{is true} \quad \text{or} \quad P_2 \quad \text{is true} \]
  \[ P_1 \rightarrow P_2 \quad \text{is true iff} \quad P_1 \quad \text{is false} \quad \text{or} \quad P_2 \quad \text{is true} \]
  \[ \text{i.e., is false iff} \quad P_1 \quad \text{is true} \quad \text{and} \quad P_2 \quad \text{is false} \]
  \[ P_1 \iff P_2 \quad \text{is true iff} \quad P_1 \quad \text{is true} \quad \text{and} \quad P_2 \quad \text{is false} \]

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

  \[ \neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (false \lor true) = true \land true = true \]
# Truth Tables for Connectives

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
<th>$P \Leftrightarrow Q$</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>
Wumpus World Sentences

- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
  - observation $R_1 : \neg P_{1,1}$

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

- "Pits cause breezes in adjacent squares"
  - rule $R_2 : B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
  - rule $R_3 : B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
  - observation $R_4 : \neg B_{1,1}$
  - observation $R_5 : B_{2,1}$

- What can we infer about $P_{1,2}, P_{2,1}, P_{2,2}$, etc.?
## Truth Tables for Inference

<table>
<thead>
<tr>
<th></th>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
<th>$R_5$</th>
<th>$KB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$false$</td>
</tr>
<tr>
<td>$false$ :</td>
<td>$true$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$false$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$false$</td>
<td>$false$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</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>$false$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$ :</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$false$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$true$  :</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
</tbody>
</table>

- Enumerate rows (different assignments to symbols $P_{i,j}$)
- Check if rules are satisfied ($R_i$)
- Valid model ($KB$) if all rules satisfied
Inference by Enumeration

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

```plaintext
function TT-ENTAILS?(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic
  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))

• O(2^n) for n symbols; problem is co-NP-complete
```
equivalence, validity, satisfiability
Logical Equivalence

- Two sentences are logically equivalent iff true in same models:
  \[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

<table>
<thead>
<tr>
<th>Expression</th>
<th>Equivalent Expression</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\alpha \land \beta))</td>
<td>((\beta \land \alpha))  commutativity of \land</td>
<td></td>
</tr>
<tr>
<td>((\alpha \lor \beta))</td>
<td>((\beta \lor \alpha))  commutativity of \lor</td>
<td></td>
</tr>
<tr>
<td>(((\alpha \land \beta) \land \gamma))</td>
<td>((\alpha \land (\beta \land \gamma))) associativity of \land</td>
<td></td>
</tr>
<tr>
<td>(((\alpha \lor \beta) \lor \gamma))</td>
<td>((\alpha \lor (\beta \lor \gamma))) associativity of \lor</td>
<td></td>
</tr>
<tr>
<td>(\neg(\neg \alpha))</td>
<td>(\alpha)  double-negation elimination</td>
<td></td>
</tr>
<tr>
<td>((\alpha \implies \beta))</td>
<td>((\neg \beta \implies \neg \alpha)) contraposition</td>
<td></td>
</tr>
<tr>
<td>((\alpha \implies \beta))</td>
<td>((\neg \alpha \lor \beta))  implication elimination</td>
<td></td>
</tr>
<tr>
<td>((\alpha \iff \beta))</td>
<td>(((\alpha \implies \beta) \land (\beta \implies \alpha))) biconditional elimination</td>
<td></td>
</tr>
<tr>
<td>(\neg(\alpha \land \beta))</td>
<td>((\neg \alpha \lor \neg \beta))  De Morgan</td>
<td></td>
</tr>
<tr>
<td>(\neg(\alpha \lor \beta))</td>
<td>((\neg \alpha \land \neg \beta))  De Morgan</td>
<td></td>
</tr>
<tr>
<td>((\alpha \land (\beta \lor \gamma)))</td>
<td>(((\alpha \land \beta) \lor (\alpha \land \gamma))) distributivity of \land over \lor</td>
<td></td>
</tr>
<tr>
<td>((\alpha \lor (\beta \land \gamma)))</td>
<td>(((\alpha \lor \beta) \land (\alpha \lor \gamma))) distributivity of \lor over \land</td>
<td></td>
</tr>
</tbody>
</table>
Validity and Satisfiability

- A sentence is **valid** if it is true in all models,
  e.g., $\text{True}$, $A \lor \neg A$, $A \implies A$, $(A \land (A \implies B)) \implies B$

- Validity is connected to inference via the **Deduction Theorem**:
  $KB \models \alpha$ if and only if $(KB \implies \alpha)$ is valid

- A sentence is **satisfiable** if it is true in some model
  e.g., $A \lor B$, $C$

- A sentence is **unsatisfiable** if it is true in no models
  e.g., $A \land \neg A$

- Satisfiability is connected to inference via the following:
  $KB \models \alpha$ if and only if $(KB \land \neg \alpha)$ is unsatisfiable
  i.e., prove $\alpha$ by **reductio ad absurdum**
inference
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 alg.
    - Typically require translation of sentences into a normal form

- **Model checking**
  - truth table enumeration (always exponential in $n$)
  - improved backtracking
  - heuristic search in model space (sound but incomplete)
    - e.g., min-conflicts-like hill-climbing algorithms
Forward and Backward Chaining

- **Horn Form** (restricted)
  
  \[ KB = \text{conjunction of Horn clauses} \]

- Horn clause =
  
  - proposition symbol; or
  - (conjunction of symbols) \implies symbol

  \[ e.g., C \land (B \implies A) \land (C \land D \implies B) \]

- **Modus Ponens** (for Horn Form): complete for Horn KBs

  \[
  \frac{\alpha_1, \ldots, \alpha_n, \alpha_1 \land \cdots \land \alpha_n \implies \beta}{\beta}
  \]

- Can be used with forward chaining or backward chaining

- These algorithms are very natural and run in \textit{linear} time
Example

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

$P \implies Q$
$L \land M \implies P$
$B \land L \implies M$
$A \land P \implies L$
$A \land B \implies L$
$A$
$B
forward chaining
Forward Chaining

- Start with given proposition symbols (atomic sentence)
  e.g., $A$ and $B$

- Iteratively try to infer truth of additional proposition symbols
  e.g., $A \land B \implies C$, therefore we establish $C$ is true

- Continue until
  - no more inference can be carried out, or
  - goal is reached
Forward Chaining Example

• Given

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

• Agenda: \( A, B \)

• Annotate horn clauses with number of premises
Forward Chaining Example

- Process agenda item $A$
- Decrease count for horn clauses in which $A$ is premise
Forward Chaining Example

- Process agenda item $B$
- Decrease count for horn clauses in which $B$ is premise
- $A \land B \implies L$ has now fulfilled premise
- Add $L$ to agenda
Forward Chaining Example

- Process agenda item $L$

- Decrease count for horn clauses in which $L$ is premise

- $B \land L \implies M$ has now fulfilled premise

- Add $M$ to agenda
Forward Chaining Example

- Process agenda item $M$
- Decrease count for horn clauses in which $M$ is premise
- $L \land M \implies P$ has now fulfilled premise
- Add $P$ to agenda
Forward Chaining Example

- Process agenda item $P$
- Decrease count for horn clauses in which $P$ is premise
- $P \implies Q$ has now fulfilled premise
- Add $Q$ to agenda
- $A \land P \implies L$ has now fulfilled premise
Forward Chaining Example

- Process agenda item $P$
- Decrease count for horn clauses in which $P$ is premise
- $P \implies Q$ has now fulfilled premise
- Add $Q$ to agenda
- $A \land P \implies L$ has now fulfilled premise
- But $L$ is already inferred
Forward Chaining Example

- Process agenda item $Q$
- $Q$ is inferred
- Done
Forward Chaining Algorithm

**function** PL-FC-ENTAILS?(KB, q) **returns** true or false

**inputs:** KB, the knowledge base, a set of propositional Horn clauses
q, the query, a proposition symbol

**local variables:** count, a table, indexed by clause, init. number of premises
inferred, a table, indexed by symbol, each entry initially false
agenda, a list of symbols, initially the symbols known in KB

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)
    return false
backward chaining
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

- $A$ and $B$ are known to be true
- $Q$ needs to be proven
Backward Chaining Example

- Current goal: $Q$
- $Q$ can be inferred by $P \implies Q$
- $P$ needs to be proven
Backward Chaining Example

- Current goal: $P$

- $P$ can be inferred by $L \land M \implies P$

- $L$ and $M$ need to be proven
Backward Chaining Example

- Current goal: $L$
- $L$ can be inferred by $A \land P \implies L$
- $A$ is already true
- $P$ is already a goal

$\implies$ repeated subgoal
Backward Chaining Example

- Current goal: $L$
Backward Chaining Example

- Current goal: \( L \)
- \( L \) can be inferred by \( A \land B \implies L \)
- Both are true
Backward Chaining Example

- Current goal: $L$
- $L$ can be inferred by $A \land B \implies L$
- Both are true

$\implies L$ is true
Backward Chaining Example

- Current goal: $M$
Backward Chaining Example

- Current goal: $M$
- $M$ can be inferred by $B \land L \implies M$
Backward Chaining Example

- Current goal: $M$
- $M$ can be inferred by $B \land L \implies M$
- Both are true

$\Rightarrow M$ is true
Backward Chaining Example

- Current goal: \( P \)
- \( P \) can be inferred by \( L \land M \implies P \)
- Both are true

\( \implies P \) is true
Backward Chaining Example

- Current goal: $Q$
- $Q$ can be inferred by $P \implies Q$
- $P$ is true

$\Rightarrow Q$ is true
Forward vs. Backward Chaining

- FC is *data-driven*, cf. 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
resolution
Resolution

- Conjunctive Normal Form (CNF—universal)
  - conjunction of disjunctions of literals
    - E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

- Resolution inference rule (for CNF): complete for propositional logic
  - \(\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n\)
  - \(\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n\)
  - where \(\ell_i\) and \(m_j\) are complementary literals. E.g.,
    - \(P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}\)
    - \(\vdash P_{1,3}\)

- Resolution is sound and complete for propositional logic
Wampus World

- Rules such as: “If breeze, then a pit adjacent.”

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

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

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \( (\alpha \implies \beta) \land (\beta \implies \alpha) \).

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

2. Eliminate \( \implies \), replacing \( \alpha \implies \beta \) 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 (\( \lor \) over \( \land \)) 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 Example

- \( KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \)
  reformulated as:
  \((\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})\)

- Observation: \( \neg B_{1,1} \)

- Goal: disprove: \( \alpha = \neg P_{1,2} \)

- Resolution
  \[
  \begin{array}{c}
  \neg P_{1,2} \lor B_{1,1} \\
  \neg B_{1,1} \\
  \hline
  \neg P_{1,2}
  \end{array}
  \]

- Resolution
  \[
  \begin{array}{c}
  \neg P_{1,2} \\
  P_{1,2} \\
  \hline
  \text{false}
  \end{array}
  \]
• In practice: all resolvable pairs of clauses are combined
Resolution Algorithm

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

```plaintext
function PL-RESOLUTION(KB, \alpha) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
             \alpha, the query, a sentence in propositional logic
    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 ∪ resolvents
            if new ⊆ clauses then return false
        clauses ← clauses ∪ new
    end
```

• Logical agent for Wumpus world explores actions
  – observe glitter → done
  – unexplored safe spot → plan route to it
  – if Wampus in possible spot → shoot arrow
  – take a risk to go possibly risky spot

• Propositional logic to infer state of the world

• Heuristic search to decide which action to take
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 with respect to 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, inference to determine state of the world, etc.

- Forward, backward chaining are linear-time, complete for Horn clauses.

- Resolution is complete for propositional logic.