Logical Agents

Agents that are able to:
- Form representations of the world
- Use a process to derive new representations of the world
- Use these new representations to deduce what to do

Knowledge-based Agents

Knowledge and Reasoning: humans are very good at acquiring new information by combining raw knowledge, experience, with reasoning.

Examples:
- Diagnosis: e.g., a physician diagnoses a patient, i.e., it infers what disease he/she has, based on the knowledge he/she acquired as a student, textbooks, prior cases and also some reasoning process (patterns of association, or other process) that he/she may not be able to describe.
- Car repair diagnosis
- Common sense reasoning
- Inventions, new ideas

Knowledge-base Agents

Key issues:
- Representation of knowledge
- Reasoning processes

Knowledge base = set of sentences in a formal language representing facts about the world(*)

(*) called knowledge representation language

Knowledge bases

Key aspects:
- How to add sentences to the knowledge base
- How to query the knowledge base

Both tasks may involve inference: i.e. how to derive new sentences from old sentences.

Logical agents: inference must obey the fundamental requirement that when one asks a question to the knowledge base, the answer should follow from what has been told to the knowledge base previously. (In other words the inference process should not "make things" up…)

Outline

1. General principles of logic – main vehicle for representing knowledge
2. Wumpus World - a toy world; how a knowledge based agent operates
3. Propositional logic
4. Predicate logic
5. Satisfiability as an Encoding language
6. NP-Completeness – Worst case vs. practice
Logic in general

Logics are formal languages for representing information such that conclusions can be drawn. A logic involves:

- A language with a syntax for specifying what is a legal expression in the language; syntax defines well-formed sentences in the language.
- Semantics for associating elements of the language with elements of some subject matter. Semantics defines the "meaning" of sentences (link to the world); i.e., semantics defines the truth of a sentence with respect to each possible world.
- Inference rules for manipulating sentences in the language.

Arithmetic

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 \(x = 0\), \(y = 6\).

Systems as Constrained Featured Sets

Several systems -- biological, mechanical, electric, etc --- can be represented by appropriate sets of "features" with constraints among the features encoding physical or other laws relevant to the organism or device.

Reasoning can then be used among other purposes, to diagnose malfunctions in these systems; for example, features associated with "causes" can be inferred from features associated with "symptoms". This general approach is key to an important class of AI applications.

Simple Robot Domain

Consider a robot that is able to lift a block, if that block is liftable (i.e., not too heavy), and if the robot's battery power is adequate. If both of these conditions are satisfied, then when the robot tries to lift a block it is holding, its arm moves.

Feature 1 → BatIsOk (0 or 1)
Feature 2 → BlockLiftable (0 or 1)
Feature 3 → RobotMoves (0 or 1)

BatIsOk and BlockLiftable implies RobotMoves

Simple Robot Domain

Feature 1 → BatIsOk (0 or 1)
Feature 2 → BlockLiftable (0 or 1)
Feature 3 → RobotMoves (0 or 1)

We need a language to express the values of features and constraints among features, also inference mechanisms, i.e., principled ways of performing reasoning.
Binary valued featured descriptions

Consider the following description:
- The router can send packets to the edge system only if it supports the new address space. For the router to support the new address space it is necessary that the latest software release be installed. The router can send packets to the edge system if the latest software release is installed. The router does not support the new address space.

- Features:
  • Router
    → Feature 1 – router can send packets to the edge of system
  • Latest software release
    → Feature 2 – router supports the new address space
  • Feature 3 – latest software release is installed

How can we write these specifications in a formal language and reason about the system?

Truth of Sentence vs. Satisfaction of Constraints

Truth of a sentence vs. Satisfaction of Constraints - we can think of a logic sentence, e.g., an arithmetic sentence or a general logic sentence, as a constraint; the sentence is true if and only if the constraint is “satisfied”.

We will talk more about “constraint languages”, particular kinds of logics, and constraint solving as a form of logical reasoning.

Standard logics ➔ every sentence must be either true or false in each possible world – there is no "in between".

Logical Reasoning: Entailment

Entailment means that one thing follows from another:

A Knowledge base $KB \models \alpha$ iff (if and only if) $\alpha$ is true in all worlds where $KB$ is true

- E.g., the KB containing “Giants won” and “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.

Example:
- \( x + y \geq 7 \), is true in all the models in which \( x \geq 7 - y \), assuming that we are dealing with real numbers, in particular \( x = 7 \) and \( y = 0 \) or \( x = 8 \) and \( y = 1 \), etc.
- Basically, each model corresponds to a different assignment of numbers to the variables and satisfying the constraint; note each assignment determines the truth or falsehood of the arithmetic sentence.

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 \( \alpha \) (i.e., models that assign true to \( \alpha \)).

Models

\[ KB \models \alpha \iff M(KB) \subseteq M(\alpha) \]
- E.g. \( KB = \text{Giants won and Reds won} \)
  \( \alpha = \text{Giants won} \)
- Other ways of talking about entailment:
  \( KB \models \alpha \)
  If \( \alpha \) is true, then KB must be true;
  (Informally – the truth of \( \alpha \) is contained in the truth of KB)

We can think of a knowledge base as a statement and we talk about a knowledge base entailing a sentence.

Wumpus World

Performance measure
- gold +1000,
- death -1000
- (falling into a pit or being eaten by the wumpus)
- -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

Sensors: Stench, Breeze, Glitter, Bump, Scream

Exploring a wumpus world

The knowledge base of the agent consists of the rules of the Wumpus world plus the percept nothing in [1,1].

None, none, none, none, none
Stench, Breeze, Glitter, Bump, Scream
Exploring a wumpus world

- Stench, Breeze, Glitter, Bump, Scream
- None, none, none, none, none
- A – Agent
- V – visited
- B – Breeze
- P – Pit in (2,2) or (3,1)

Difficult inference, because it combines knowledge gained at different times in different places; the inference is beyond the abilities of most animals.

Assumption: the agent turns and go to square (2,3)!

In each case where the agent draws a conclusion from the available information, that conclusion is guaranteed to be correct if the initial information is correct - fundamental property of logical reasoning!

How to build logical agents that can represent the necessary information and draw conclusions?

Entailment in the wumpus world

Knowledge Base in the Wumpus World

- Rules of the wumpus world + new percepts

Consider possible models for KB with respect to the cells (1,2), (2,2) and (3,1), with respect to the existence or non-existence of pits

3 Boolean choices ⇒ 8 possible models (enumerate all the models)

Wumpus models

- KB = wumpus-world rules + observations
- \( \alpha_1 = \text{"[1,2] has no pit"} \)
- In every model in which KB is true, \( \alpha_1 \) is True (proved by model checking)
Wumpus models

Models of the KB and α2

KB = wumpus-world rules + observations
α2 = “[3,2] has no pit”, this is only True in some of the models for which KB is True, therefore KB |= α2
Inference algorithm used to reason about α1 and α2.

Model Checking

Inference: Model Checking

Inference by Model checking –

we enumerate all the KB models and check if
α1 and α2 are True in all the models (which implies that we can only use it when we have a finite number of models).

Inference

KB |= α = sentence α can be derived from KB by procedure i

Soundness (or Truth preservation): i is sound if whenever KB |= α, it is also true that KB |= α; an unsound procedure can conclude statements that are not true.

Completeness: i is complete if whenever KB |= α, it is also true that KB |= α; a complete procedure is able to derive any sentence that is entailed. That is, the procedure will answer any question whose answer follows from what is known by the KB.

Note: first-order logic which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

3 - Propositional Logic

Syntax

Elements of the Language

Atoms:
– Truth Symbols: True and False;
– Propositional Symbols – (Strings of characters that begin with a capital letter: P, Q, R, ..., P1, P2, ColorOfBlock, etc)

Connectives:
– ¬ (negation)
– ∧ (and)
– ∨ (or)
– ⇒ (implication)
– ⇔ (biconditional)

Syntax

Syntax of Well Formed Formulas (wffs) or sentences

– Atomic sentences are wffs:
  True, False, and propositional symbol;
  Example: True; False; P, R, BlockIsRed; SeasonsWinter;
– Complex or compound wffs.

Given w1 and w2 wffs:
– ¬ w1 (negation)
– w1 ∧ w2 (conjunction)
– w1 ∨ w2 (disjunction)
– w1 ⇒ w2 (implication; w1 is the antecedent; w2 is the consequent)
– w1 ⇔ w2 (biconditional)
Propositional logic: Examples

Examples of wffs

- \( P \land Q \)
- \( (P \lor Q) \Rightarrow R \)
- \( P \land Q \Rightarrow P \)
- \( (P \Rightarrow Q) \lor (Q \Rightarrow \neg P) \)
- \( P \land (Q \Rightarrow R) \)
- \( P \land Q \Rightarrow P \land Q \), this is not a wff.

Note 1: atoms or negated atoms are called literals; examples \( P \) and \( \neg P \) are literals.
Note 2: parentheses are important to ensure that the syntax is unambiguous. Quite often parentheses are omitted. The order of precedence in propositional logic is (from highest to lowest): \( \neg \), \( \land \), \( \lor \), \( \Rightarrow \), \( \Leftrightarrow \).

While an inference process operates on "syntax" --- internal physical configurations such as bits in registers or patterns of electrical blips in brains --- the process corresponds to the real-world via semantics --- the process of assigning truth or falsehood to every sentence in a possible world.

Propositional Logic: Syntax vs. Semantics

Semantics has to do with "meaning":
- it associates the elements of a logical language with the elements of a domain of discourse;
- Propositional Logic - we associate atoms with propositions about the world (therefore propositional logic);

Propositional Logic: Semantics

Example:
- We might associate the atom BlockIsRed with the proposition: "The block is Red". But we could also associate it with the proposition: "The block is Black" even though this would be quite confusing. BlockIsRed has value True just in the case the block is red; otherwise BlockIsRed is False.
- Which ones are propositions?
  - Cornell University is Ithaca NY;
  - \( 1 + 1 = 2 \);
  - What time is it?
  - \( 2 \times 3 = 10 \);
  - Watch your step!

Propositional Logic: Semantics

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

\[ \begin{align*}
\neg S & \text{ is true iff } S \text{ is false} \\
S \lor S_1 & \text{ is true iff } S \text{ is true or } S_1 \text{ is true} \\
S \Rightarrow S_1 & \text{ is true iff } S \text{ is false} \text{ or } S_1 \text{ is true} \\
S \Leftrightarrow S_1 & \text{ is true iff } S \text{ is true and } S_1 \text{ is true or } S \text{ is false and } S_1 \text{ is false}
\end{align*} \]

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

\[ \neg (P_1 \lor (P_2 \land P_3)) \Rightarrow \neg \neg (true \lor false) \Rightarrow true \lor false \Rightarrow true \Rightarrow true \]
Propositional Logic: Semantics

Truth table for connectives

Given the values of atoms under some interpretation, we can use a truth table to compute the value for any wff under that same interpretation; the truth table establishes the semantics (meaning) of the propositional connectives.

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>(~P)</th>
<th>P &amp; 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>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

We can use the truth table to compute the value of any wff given the values of the constituent atoms in the wff.

Implication (p \rightarrow q)

Implication plays an important role in reasoning a variety of terminology is used to refer to implication:
- conditional statement;
- if \( p \) then \( q \);
- \( p \rightarrow q \);
- \( p \) is sufficient for \( q \);
- \( p \) when \( q \);
- \( q \) whenever \( p \);
- a necessary condition for \( p \) is \( q \);
- \( p \) implies \( q \);
- \( p \) only if \( q \);
- \( p \) if and only if \( q \);
- \( p \rightarrow q \)

(*) assuming the statement true, for \( p \) to be true, \( q \) has to be true

Notes: Bi-conditionals (p \iff q)

Logical equivalence

Two sentences are logically equivalent if they have the same truth value in all the interpretations, i.e., \( \alpha \equiv \beta \) iff \( \alpha \models \beta \) and \( \beta \models \alpha \)

| (\( a \land \beta \) \land \( b \land \alpha \)) = \( (\beta \land \alpha) \) | commutativity of \( \land \) |
| (\( a \land \beta \) \land \( \alpha \land \gamma \)) = \( \alpha \land (\beta \land \gamma) \) | associativity of \( \land \) |
| (\( a \land \beta \) \land \( \alpha \lor \gamma \)) = \( \alpha \lor (\beta \land \gamma) \) | associativity of \( \lor \) |
| \( \sim (a \land \beta) \) = \( \sim (a \land \beta) \) | double negation elimination |
| \( (a \rightarrow \beta) \equiv (\sim (a \land \beta)) \) | contraposition |
| \( (a \rightarrow \beta) \equiv (\sim (a \land \beta)) \) | implication elimination |
| \( \sim (a \rightarrow \beta) \equiv (a \land \beta) \) | de Morgan |
| \( \sim (a \lor \beta) \equiv (a \land \beta) \) | de Morgan |
| \( \sim (a \lor \beta) \equiv (a \land \beta) \) | distributivity of \( \land \) over \( \lor \) |
| \( \sim (a \lor \beta) \equiv (a \land \beta) \) | distributivity of \( \lor \) over \( \land \) |

Note: logical equivalence (or iff) allows us to make statements about PL, pretty much like we use \( = \) in ordinary mathematics.

Propositional Logic: Semantics

Truth table for connectives

We can use the truth table to compute the value of any wff given the values of the constituent atoms in the wff.

Example:

Suppose \( P \) and \( Q \) are False and \( R \) has value True. Given this interpretation, what is the truth value of \( (P \& Q) \rightarrow R \lor (\sim P ?)

If an agent describes its world using \( n \) features (corresponding to propositions), and these features are represented in the agent’s model of the world by a corresponding set of \( n \) atoms, then there are \( 2^n \) different ways its world can be. Why? Each of the ways the world can be corresponds to an interpretation. Therefore there are \( 2^n \) interpretations.

Given an interpretation (i.e., the truth values for the \( n \) atoms) the agent can use the truth table to find the values of any wffs.
Translating sentences in natural language (e.g., English) into logical expressions is an essential part of specifying hardware and software systems.

Example:
- The automated reply cannot be sent when the file system is full;
  - p - the automated reply can be sent;
  - q - the file system is full;
- \( q \implies \neg p \)

The KB with the system specifications should not contain conflicting requirements; i.e., the KB should be consistent: there must be at least a model that makes the system true.

Example:
- Constraints:
  - Feature 1 (P) (router supports the new address space) and Feature 2 (Q) (latest software release is installed);
  - If Feature 2: \( Q \) (latest software release is installed) then Feature 1: \( P \) (router supports the new address space);
- Feature 3 (R) (latest software release is installed) and Feature 4 (S) (router can send packets to the edge system);
  - If Feature 3: \( R \) (latest software release is installed) then Feature 4: \( S \) (router can send packets to the edge system);
- \( R \implies S \)

Propositional Logic:

Satisfiability and Models

An interpretation satisfies a wff, if the wff is assigned the value True, under that interpretation. An interpretation that satisfies a wff is called a model of that wff.

Given an interpretation (i.e., the truth values for the n atoms) the agent can use the truth table to find the values of any wffs.

Examples:
1 - \( P \land \neg P \)
2 - \( P \lor Q \lor \neg Q \lor \neg P \)

Propositional Logic:

Inconsistency (Unsatisfiability) and Validity

An inconsistent or unsatisfiable set of Wffs

It is possible that an interpretation satisfies a set of wffs. In that case we say that the set of wffs is unsatisfiable or inconsistent.

Examples:
1 - \( P \land \neg P \)
2 - \( P \lor Q \lor \neg Q \lor \neg P \)

Valid or Tautology of a set of Wffs

If a wff is True under all the interpretations of its constituents atoms, we say that the wff is valid or is a tautology.

Examples:
1 - \( P \land Q \lor \neg P \lor Q \lor \neg P \lor \neg Q \land \neg Q \land P \land \neg P \)
2 - \( P \lor Q \lor \neg Q \lor \neg P \)

Validity of the KB

BatIsOk \& BlockLiftable \implies RobotMove

If we have this rule in our KB, (therefore we want it to be true) interpretations that assign the value True to BatIsOk and BlockLiftable and False to RobotMove can be ruled out as Models.
Inference

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 $[i,j]$.
Sentence 1 (R1): $\neg P_{1,1}$
Sentence 2 (R2): $\neg B_{1,1}$
Sentence 3 (R3): $B_{2,1}$
Sentence 4 (R4): $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
Sentence 5 (R5): $B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

Inference by enumeration

- The goal of logical inference is to decide whether $\text{KB} \models \alpha$, for some sentence $\alpha$.
- For example, given the rules of the Wumpus World is $P_{1,2}$ entailed?

Relevant propositional symbols:
- $R_1: \neg P_{1,1}$
- $R_2: \neg B_{1,1}$
- $R_3: B_{2,1}$
- $R_4: B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
- $R_5: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

Inference by enumeration → we have 7 symbols therefore $2^7$ models.

Is $P_{1,2}$ Entailed from KB?
Is $P_{2,2}$ Entailed from KB?

Given $R_1, R_2, R_3, R_4, R_5$

There are only 3 models for the KB: i.e., for which $R_1, R_2, R_3, R_4, R_5$ are True;
In all of them $P_{1,2}$ is false, so there is not pit in $[1,2]$ – the KB entails $\neg P_{1,2}$; on the other hand $P_{2,2}$ is true in two of the three models and false in the other one – so at this point we cannot tell whether $P_{2,2}$ is true or not.
Inference by enumeration

\[ \text{TT-Entails} \] – Truth Table enumeration algorithm for deciding propositional entailment;

Validity and Satisfiability

A sentence is valid (or is a tautology) if it is true in all models,
\( \text{e.g., True, } A \lor \neg A, A \Rightarrow A, (A \land (A \Rightarrow B)) \Rightarrow B \)

Validity is connected to inference via the Deduction Theorem:
\( \text{KB} \models \alpha \text{ if and only if (KB} \Rightarrow \alpha \text{) is valid} \)

A sentence is satisfiable if it is true in some model
\( \text{e.g., } A \lor B, C \)

A sentence is unsatisfiable if it is true in no models
\( \text{e.g., } A \land \neg A \)

Satisfiability is connected to inference via the following:
\( \text{KB} \models \alpha \text{ if and only if (KB} \land \neg \alpha \text{) is unsatisfiable (Reductio ad absurdum; Proof by refutation or Proof by contradiction)} \)

Proof methods

Proof methods divide into (roughly) two kinds:

- Application of inference rules
  - Legitimate (sound) generation of new sentences from old
  - 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)

Rules of Inference and Proofs

The sequence of wffs \( w_1, w_2, ..., w_n \) is called a proof (or deduction) of \( w_n \) from a set of wffs \( \Delta \) if each \( w_i \) in the sequence is either in \( \Delta \) or can be inferred from a wff (or wffs) earlier in the sequence by using a valid rule of inference.

If there is a proof of \( w_n \) from \( \Delta \), we say that \( w_n \) is a theorem of the set \( \Delta \)
\( \Delta \models w_n \)

The concept of proof is relative to a particular set of inference rules used. If we denote the set of inference rules used by \( R \), we can write the fact that \( w_n \) can be derived from \( \Delta \) using the set of inference rules in \( R \)
\( \Delta \models_R w_n \)
Propositional logic: Rules of Inference or Methods of Proof

How to produce additional wffs (sentences) from other ones? What steps can we perform to show that a conclusion follows logically from a set of hypotheses?

Example

Modus Ponens

\[ P \]

\[ P \Rightarrow Q \]

\[ \therefore \]

\[ Q \]

The hypotheses are written in a column and the conclusions below the bar. The symbol \[ \therefore \] denotes “therefore”. Given the hypotheses, the conclusion follows.

The basis for this rule of inference is the tautology \[ (P \land (P \Rightarrow Q)) \Rightarrow Q \] .

Proof Tree

Proofs can also be based on partial orders – we can represent them using a tree structure:

- Each node in the proof tree is labeled by a wff, corresponding to a wff in \( \Delta \) or be inferable from its parents in the tree using one of the rules of inference;
- The labeled tree is a proof of the label of the root node.

Example:

Given the set of wffs:

\[ P, R, P \Rightarrow Q \]

Give a proof of \( Q \land R \)

Tree Proof

\[ P, P \Rightarrow Q, R, Q \Rightarrow R \]

Propositional Logic: Entailment

Let a Knowledge Base (KB) be a set of wffs and \( \alpha \) be a wff:

Entailment means that one thing follows from another:

\[ KB \models \alpha \]

A Knowledge base KB logically entails a wff \( \alpha \), or \( \alpha \) logically follows from KB, or \( \alpha \) is a logical consequence of KB, if and only if it is true in all worlds where KB is true:

- \( \{P\} \models P \)
- \( \{P, P \Rightarrow Q\} \models Q \)
- False \( \models \) (where \( W \) is any wff)
- \( \models Q \Rightarrow P \)
- \( \models x = k \land x \neq k\)

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

Entailment vs. Inference

When inference rules are sound and complete we can determine whether one wff follows from a knowledge base, KB, by searching for a proof instead of using the truth table;

When the set of rules are sound, if we find a proof of \( \alpha \) from KB, we know that \( \alpha \) logically follows from KB.

When the set of inference rules are complete we know that we will eventually be able to prove that \( \alpha \) follows from KB by using a complete search procedure for the proof.

Using proof methods instead of truth table methods, usually gives great computational advantage – however, to determine whether or not a wff logically follows from a set of wffs is in general an NP-Complete problem.

There are special cases that are tractable.
Resolution in Propositional Logic

**Resolution (for CNF)**

Very important inference rule – several other inference rules can be seen as special cases of resolution.

\[
\begin{align*}
P \lor Q \\
\therefore Q \lor R
\end{align*}
\]

Soundness of rule (validity of rule):

\[
[(P \lor Q) \land (\neg P \lor R)] \Rightarrow (Q \lor R)
\]

Resolution for CNF – applied to a special type of wffs: conjunction of clauses.

**Literal** – either an atom (e.g., \( P \)) or its negation (e.g., \( \neg P \)).

**Clause** – disjunction of literals (e.g., \( P \lor Q \lor \neg R \)).

Note: Sometimes we use the notation of a set for a clause: e.g. \( \{ P, Q, \neg R \} \) corresponds to the clause \( (P \lor Q \lor \neg R) \); the empty clause (sometimes written as Nil or \( \{ \} \)) is equivalent to False.

**CNF**

Conjunctive Normal Form (CNF)

A wff is in CNF format when it is a conjunction of disjunctions of literals.

\( (\neg P \lor Q \lor R) \land (S \lor P \lor T \lor \neg R) \land (Q \lor S) \)

Resolution for CNF – applied to wffs in CNF format.

Resolution: Notes

1 – Rule of Inference: Chaining

\[
\begin{align*}
P \lor Q \\
\therefore R \lor Q
\end{align*}
\]

Rule of Inference: Chaining

2 – Rule of Inference: Modus Ponens

\[
\begin{align*}
P \lor Q \\
\therefore Q
\end{align*}
\]

Rule of Inference: Modus Ponens

3 – Unit Resolution

\[
\begin{align*}
P \lor Q \\
\therefore \neg P \lor Q
\end{align*}
\]

Rule of Inference: Chaining

4 – No duplications in the resolvent set

\[
\begin{align*}
P \lor Q \lor R \lor S \\
\therefore Q \lor R \lor S \lor W
\end{align*}
\]

Resolution: Notes

5 – Resolving one pair at a time

\[
\begin{align*}
P \lor Q \lor R \lor S \\
\therefore Q \lor R \lor S \lor W
\end{align*}
\]

Resolution: Notes
Resolution:

Notes

6 – Same atom with opposite signs

\[ \{ \neg P \} \]

\[ \vdash \neg \neg P \]

\[ \vdash \neg P \]

False – any set of wffs containing two contradictory clauses is unsatisfiable. However, a clause \( \{ P, \neg P \} \) is True.

Soundness of Resolution:

Validity of the Resolution Inference Rule

\[ (P \lor Q) \lor (P \lor R) \implies (Q \lor R) \]

Conversion to CNF

\[ P \iff (Q \lor R) \]

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

\[ (P \implies (Q \lor R)) \land ((Q \lor R) \implies P) \]

2. Eliminate \( \implies \), replacing \( \alpha \implies \beta \) with \( \neg \alpha \lor \beta \).

\[ (\neg P \lor Q \lor R) \land (\neg Q \lor \neg R \lor P) \]

3. Move \( \neg \) inwards using de Morgan's rules and double-negation:

\[ (\neg P \lor Q \lor R) \land ((\neg Q \lor \neg R) \lor P) \]

4. Apply distributivity law (\( \land \) over \( \lor \) ) and flatten:

\[ (\neg P \lor Q \lor R) \land (\neg Q \lor P) \land (\neg Q \lor P) \]

Resolution:

Wumpus World

\[ P_{31} \lor P_{2,2}, \neg P_{2,2}, \]

\[ \vdash P_{31} \]

Resolution:

Robot Domain

\[ KB \]

\[ \vdash \neg \text{BlockLiftable} \]

Example:

\[ \text{RobotMoves} \land \text{BatIsOk} \land \text{BlockLiftable} \implies \text{RobotMoves} \]

\[ KB' \]

\[ \vdash \neg \text{BlockLiftable} \]

\[ KB'' \]

Show that KB

\[ \vdash \neg \text{BlockLiftable} \]

\[ \text{Nil} \]
**Propositional Logic: Proof by refutation or contradiction:**

Satisfiability is connected to inference via the following:

\[ KB \models \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]

One assumes \( \neg \alpha \) and shows that this leads to a contradiction with the facts in KB.

**Resolution Refutation**

Resolution is sound - but resolution is not complete - e.g., \( (P \lor R) \models (P \lor R) \) but we cannot use resolution directly to decide all logical entailments.

Resolution is Refutation Complete:

- We can show that a particular \( \psi \) is entailed from a given KB how?
- We can show that a particular \( \psi \) is entailable from a given KB if we can show that \( \neg \psi \) is not.
- We can show that a particular \( \psi \) is entailable from a given KB if we can show that \( \neg \psi \) is not.

To show that \( (P \lor R) \models \neg \psi \), we prove that \( (P \lor R) \neg \psi \) is unsatisfiable.

**Resolution**

Resolution is refutation complete (Completeness of resolution refutation):

- If \( KB \models \psi \), the resolution refutation procedure, i.e., applying resolution on KB', will produce the empty clause.
- Decidability of propositional calculus by resolution refutation:
  - If KB is a set of finite clauses and if \( KB \models \psi \), then the resolution refutation procedure will terminate without producing the empty clause.

Ground Resolution Theorem

- If a set of clauses is not satisfiable, then resolution closure of those clauses contains the empty clause.
- In general, resolution for propositional logic is exponential.

The resolution closure of a set of clauses \( \psi \) in CNF, \( RC(\psi) \), is the set of all clauses derivable by repeated application of the resolution rule to clauses in \( \psi \) or their derivatives.

**Resolution example: Wumpus World**

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \quad \alpha = \neg P_{1,2} \]

**Resolution algorithm**

Any complete search algorithm applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic – resolution can always be used to either confirm or refute a sentence – refutation completeness (Given A, it's true we cannot use resolution to derive A OR B, but we can use resolution to answer the question of whether A OR B is true.)

**Resolution example: Wumpus World**

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \quad \alpha = \neg P_{1,2} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]

\[ KB = (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \land \neg B_{1,1} \]
Resolution Refutation – Ordering Search Strategies

Original clauses – 0th level resolvents
- Depth first strategy →
  • Produce a 1st level resolvent;
  • Resolve the 1st level resolvent with a 0th level resolvent to produce a 2nd level resolvent, etc.
  • With a depth bound, we can use a backtrack search strategy;
- Breadth first strategy →
  • Generate all 1st level resolvents, then all 2nd level resolvents, etc.

Refinement Resolution Strategies

Definitions:
A clause γ₂ is a descendant of a clause γ₁ iff:
- It is a resolvent of γ₁ with some other clause;
- Or is a resolvent of a descendant of γ₁ with some other clause;
If γ₁ is a descendant of γ₂, γ₁ is an ancestor of γ₂;

Set-of-support – set of clauses that are either clauses coming from the negation of the theorem to be proved or descendants of those clauses;
Set-of-support Strategy – it allows only refutations in which one of the clauses being resolved is in the set of support;
Set-of-support Strategy is refutation complete.

Refinement Strategies

Ancestry-filtered strategy – allows only resolutions in which at least one member of the clauses being resolved either is a member of the original set of clauses or is an ancestor of the other clause being resolved;

The ancestry-filtered strategy is refutation complete.

Horn Clauses

Definition:
A Horn clause is a clause that has at most one positive literal.
Examples:
- P; P ∨ ¬Q; ¬P ∨ ¬Q; ¬P ∨ ¬Q ∨ R;

Types of Horn Clauses:
- Fact – single atom – e.g., P;
- Rule – implication, whose antecedent is a conjunction of positive literals and whose consequent consists of a single positive literal – e.g., P → Q ⇒ R;
- Set of negative literals - in implication form, the antecedent is a conjunction of positive literals and the consequent is empty.
  e.g., P → Q ⇒ ¬P ∨ ¬Q;

Inference with propositional Horn clauses can be done in linear time ☺!
Forward chaining

HORN (Expert Systems and Logic Programming)

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

\[
\alpha_1, \ldots, \alpha_n, \alpha_1 \& \ldots \& \alpha_n \Rightarrow \beta
\]

Can be used with forward chaining

Deciding entailment with Horn clauses can be done in linear time, in the size of the KB

Forward Chaining: Diagnosis systems

Example: diagnostic system
- If the engine is getting gas and the engine turns over
  THEN the problem is spark plugs
- If the engine does not turn over and the lights do not come on
  THEN the problem is battery or cables
- If the engine does not turn over and the lights come on
  THEN the problem is starter motor
- If there is gas in the fuel tank and there is gas in the carburetor
  THEN the engine is getting gas

Forward chaining

(Data driven reasoning)

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

Forward chaining algorithm

```
while query is not empty do
    p₁, p₂, …, pₙ := [query] \& [premises]
    for each Horn clause c \& \Rightarrow \beta in KB do
        if \& c \& satisfies \& p₁, p₂, …, pₙ do
            new KB := KB \& \beta
            break
    end
end
```

Forward chaining is sound and complete for Horn KB

Forward chaining example

```
P \Rightarrow Q
L \& M \Rightarrow P
B \& L \Rightarrow M
A \& P \Rightarrow L
A \& \beta \Rightarrow L
A
B
```

AND-OR graph
Forward chaining example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \land E \Rightarrow L \]
\[ A \]
\[ B \]
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 $m$
4. Hence $m$ is a model of $KB$
5. If $KB \vdash q$, $q$ 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
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