REASONING

KNOWLEDGE BASED AGENTS

- Humans reason based on existing knowledge and draw conclusions.
- Knowledge Based Agent
  - Knowledge Base
    - Facts (Sentences)
  - Inference
- KBA Program
  - Uses the KB to
    - Represent background knowledge
    - Store perceived environment (Tell)
    - Derive actions (Ask)
    - Store executed actions

LOGIC

- KB has sentences
- Syntax
  - Well formed sentences
  - Symbols / Atomics Formulas
    - T, F, A, B, C etc.
  - Connectives
- Semantics
  - Meaning of sentence
  - Defines truth w.r.t to each possible state
  - Model: Assignment of truth values to symbol

PROPOSITIONAL LOGIC

- Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false.

- It is a technique of knowledge representation in logical and mathematical form.

- Example:
- a) It is Sunday.
- b) The Sun rises from West (False proposition)
- c) 3+3= 7(False proposition)
- d) 5 is a prime number.

Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and logical connectives.

Propositional Syntax

1. Atomic Propositions
2. Compound propositions

Logical Connectives

Logical connectives are logical symbols that connect propositional symbols in order to reason in a more complex way about the world.

Propositional Logic Connectives

Connective symbols	Word	Technical term	Example
$\wedge$	AND	Conjunction	$A \wedge B$
$\vee$	OR	Disjunction	$A \vee B$
$\rightarrow$	Implies	Implication	$A \rightarrow B$
$\Leftrightarrow$	If and only if	Biconditional	$A \Leftrightarrow B$
$\neg$ or $\sim$	Not	Negation	$\neg A$ or $\neg B$

PROPOSITIONAL LOGIC

- Truth Tables for Logic connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Inference Rules

- Model Checking is not an efficient algorithm because it has to consider every possible model before giving the answer (a reminder: a query R is true if under all the models (truth assignments) where the KB is true, R is true as well).
- Inference rules allow us to generate new information based on existing knowledge without considering every possible model.

PROPOSITIONAL LOGIC

- Theorem Proving
  - Applying inference rules directly to the sentences in KB to construct proof. Larger the number of models, length of proof is short
  - Hence better than model checking.

Following is the logical equivalence

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

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

PROPOSITIONAL LOGIC

- Inference and Proofs

  - Modus Ponens

  The type of inference rule we use in this example is Modus Ponens, which is a fancy way of saying that if we know an implication and its antecedent to be true, then the consequent is true as well.

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

  - And- Elimination

  If an And proposition is true, then any one atomic proposition within it is true as well.

$$\frac{\alpha \wedge \beta}{\alpha}$$

- Implication Elimination
- An implication is equivalent to an Or relation between the negated antecedent and the consequent.
- P → Q can be represented as ¬P ∨ Q

- Biconditional Elimination

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Resolution Refutation in Propositional Logic

- Resolution is one kind of proof technique that works this way -
- (i) select two clauses that contain conflicting terms (ii) combine those two clauses and (iii) cancel out the conflicting

terms.

The Resolution Refutation method was devised by Alan Robinson in 1965. He showed that the Resolution Refutation method is complete for deriving the null clause.

A proof by resolution method is a *proof by contradiction*. We start with the set of premises, *negate* the goal and add it as another clause, and show that it *leads to a contradiction* (something that is false or not possible).

The Resolution Rule

The single rule used in the refutation method, called the resolution rule, takes two clauses that have a complimentary literal as follows.

From: $R_1 \vee R_2 \ldots \vee R_k \vee Q$

And: $P_1 \vee P_2 \ldots \vee P_m \vee \neg Q$

Infer: $R_1 \vee R_2 \ldots \vee R_k \vee P_1 \vee P_2 \ldots \vee P_m$

Both the literal Q and its negation $\neg Q$ are removed and the remaining literals are combined to form a new clause, called the *resolvent*. With the smaller clause the rule becomes,

From: $R \vee Q$

And: $P \vee \neg Q$

Infer: $R \vee P$

- A Clause is a disjunction of literals (a propositional symbol or a negation of a propositional symbol, such as P, ¬P).
- A disjunction consists of propositions that are connected with an Or logical connective (P ∨ Q ∨ R).
- A conjunction, on the other hand, consists of propositions that are connected with an And logical connective (P ∧ Q ∧ R).

- Clauses allow us to convert any logical statement into a Conjunctive Normal Form (CNF), which is a conjunction of clauses, for example: (A ∨ B ∨ C) ∧ (D ∨ ¬E) ∧ (F ∨ G).
- Using logical equivalence, construct the CNF.

- Knowledge Base (KB)
- The knowledge base is a set of sentences known by a knowledge-based agent. This is knowledge that the AI is provided about the world in the form of propositional logic sentences that can be used to make additional inferences about the world.

- Entailment ($\models$)

- If $\alpha \models \beta$ (α entails β), then in any world where α is true, β is true, too.

Resolution

- Resolving a literal and its negation, i.e. ¬P and P, gives the empty clause (). The empty clause is always false, and this makes sense because it is impossible that both P and ¬P are true. This fact is used by the resolution algorithm.
- Proof by contradiction is a tool used often in computer science. If our knowledge base is true, and it contradicts ¬α, it means that ¬α is false, and, therefore, α must be true.

More technically, the algorithm would perform the following actions:

- To determine if KB ⊨ α:
- Convert (KB ∧ ¬α) to Conjunctive Normal Form.
- Keep checking to see if we can use resolution to produce a new clause.
- If we ever produce the empty clause (equivalent to False), it means, We have arrived at a contradiction, thus proving that KB ⊨ α.
- However, if contradiction is not achieved and no more clauses can be inferred, there is no entailment.

Example

- Part(I) represents the English meanings for the clauses,
- Part(II) represents the propositional logic statements for given english sentences,
- Part(III) represents the Conjunctive Normal Form (CNF) of Part(II) and
- Part(IV) shows the statements to prove

Part(I) : English Sentences

- (1) If it is sunny and warm day you will enjoy.
- (2) If it is warm and pleasant day you will do strawberry picking
- (3) If it is raining then no strawberry picking.
- (4) If it is raining you will get wet.
- (5) It is warm day
- (6) It is raining
- (7) It is sunny

Part(II) : Propositional Statements

- (1) enjoy ← sunny ∧ warm
- (2) strawberry_picking ← warm ∧ pleasant
- (3) ~strawberry_picking ← raining
- (4) wet ← raining
- (5) warm
- (6) raining
- (7) sunny

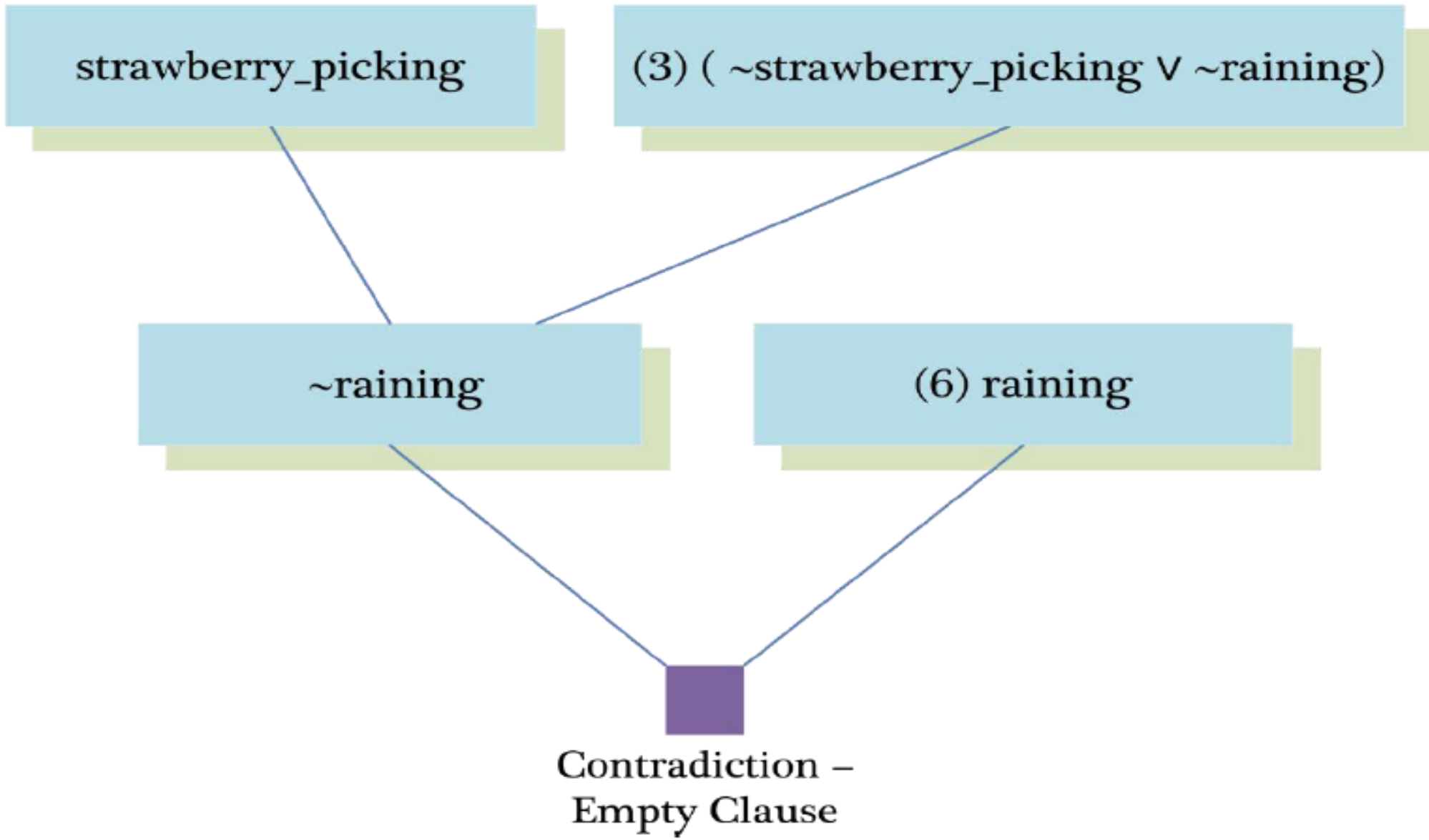
Part(III) : CNF of Part(II)

- (1) (enjoy V~sunnyV~warm)
- (2) (strawberry_picking V~warmV~pleasant)
- (3) (~strawberry_picking V~raining)
- (4) (wet V~raining)
- (5) (warm)
- (6) (raining)
- (7) (sunny)

Part(IV) : Other statements to prove by Refutation

Goal : You are not doing strawberry picking.

- Prove : ~strawberry_picking
- Assume : strawberry_picking (negate the goal and add it to given clauses).


strawberry_picking
(3) (~strawberry_picking ∨ ~raining)
~raining
(6) raining
Contradiction –
Empty Clause