1 2001 Paper 5 Question 11

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2001p5q11.pdf

(a) Explain the meaning of the notation $A \models B$, where A and B denote formulae of propositional logic.

$A \models B$ is notation for "A entails B". This means that under any interpretation for which A is valid, B is also valid. Informally, "for every mapping from formulae / variables to truth values for which every formula in A is true; B is also true".

and this is semantic-entails

(b) For each of the following equivalences, state whether it holds or not, justifying each answer rigorously.

(i)

$$(P \land (Q \to R)) \to S \simeq (\neg P \lor \neg Q \lor S) \land (\neg P \lor \neg R \lor S)$$

The equivalence does not hold. Consider the interpretation
$I = \{P \mapsto \mathbf{t}, Q \mapsto \mathbf{t}, R \mapsto \mathbf{f}, S \mapsto \mathbf{f}\}$.

all correct but where did this interpretation come from?
i.e. what's the computer algorithm that answers the question?

$$\begin{aligned}(P \land (Q \to R)) \to S &= (\mathbf{t} \land (\mathbf{t} \to \mathbf{f})) \to \mathbf{f}\\ &= (\mathbf{t} \land \mathbf{f}) \to \mathbf{f}\\ &= \mathbf{f} \to \mathbf{f}\\ &= \mathbf{t}\end{aligned}$$

So under the interpretation I, the LHS formula holds

$$\begin{aligned}(\neg P \lor \neg Q \lor S) \land (\neg P \lor \neg R \lor S) &= (\neg \mathbf{t} \lor \neg \mathbf{t} \lor \mathbf{f}) \land (\neg \mathbf{t} \lor \neg \mathbf{f} \lor \mathbf{f})\\ &= (\mathbf{f} \lor \mathbf{f} \lor \mathbf{f}) \land (\mathbf{f} \lor \mathbf{t} \lor \mathbf{f})\\ &= \mathbf{f} \land \mathbf{t}\\ &= \mathbf{f}\end{aligned}$$

Therefore there exists an interpretation I which satisfies the LHS formula but does not satisfy the RHS formula. So the LHS formula does not entail the RHS formula. We can therefore conclude that the two formulae are not equivalent.

$$\begin{aligned}&(P \land (Q \to R)) \to S \not\models (\neg P \lor \neg Q \lor S) \land (\neg P \lor \neg R \lor S) \implies\\ &(P \land (Q \to R)) \to S \not\simeq (\neg P \lor \neg Q \lor S) \land (\neg P \lor \neg R \lor S)\end{aligned}$$

(ii)

$$(P \to Q) \to (Q \to P) \simeq (Q \to P)$$

This equivalence holds. It can be proved by algebraic manipulation:

$$\begin{aligned}(P \to Q) \to (Q \to P) &\simeq \neg(P \to Q) \lor (Q \to P)\\ &\simeq \neg(\neg P \lor Q) \lor \neg Q \lor P\\ &\simeq P \land \neg Q \lor \neg Q \lor P\\ &\simeq \neg Q \lor P \land (\mathbf{t} \lor \neg Q)\\ &\simeq \neg Q \lor P\\ &\simeq Q \to P\end{aligned}$$

is this a good way for a computer to do it? why (not)?

(iii)

$$\forall xy\,(P(x) \lor \neg P(y)) \simeq \forall xy\,(P(x) \leftrightarrow P(y))$$

\vdash is syntactic entailment -- what maths tells you is true (weaker)
    This is entailment where the logic system may or may not be sound and/or complete

\vDash is semantic entailment -- what IS true in the real world
    This is entailment where the logic system is both complete and sound

If everything that maths predicts is true, then your system is SOUND -- everything you predict is true.
If a system can predict everything which is true; then it is true.

You really want a sound model. Incomplete models can be useful -- they can be far simpler and if you can prove everything then it's much simpler.

After conversion to CNF/DNF and resolution; you get a canonical answer (unique)

Truth tables have a worse expected case than CNF. However, if you want specific counterexamples or to find specific counterexamples or know it's likely to fail-fast, then truth tables can be better.

Truth tables are also not seriously affected by complex connectives ie <=>
Reduction to normal forms are greatly affected by complex connectives

This equivalence holds. It can be proved algebraically:

$$\begin{aligned}
\forall xy(P(x) \leftrightarrow \neg P(y)) &\simeq \forall xy(P(x) \land P(y) \lor \neg P(x) \land \neg P(y)) && \text{by definition of } \leftrightarrow \\
&\simeq \forall xy((P(x) \lor \neg P(y)) \land (\neg P(x) \lor P(y))) && \text{using distributivity} \\
&\simeq \forall xy(P(x) \lor \neg P(y)) \land \forall xy(\neg P(x) \lor P(y)) && \text{using distributivity} \\
&\simeq \forall xy(P(x) \lor \neg P(y)) \land \forall xy(P(x) \lor \neg P(y)) && \text{by renaming variables} \\
&\simeq \forall xy(P(x) \lor \neg P(y)) && \text{by idempotence}
\end{aligned}$$

is this a good way for a computer to do it? why (not)?

2 2002 Paper 5 Question 11

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2002p5q11.pdf

(a) For each of the following formulae, state (with justification) whether it is satisfiable, valid or neither.

(i)

$$((Q \to R) \to Q) \land \neg Q$$

This formula is satisfiable, but not valid. The formula holds under the interpretation $\{Q \mapsto \mathbf{f}, R \mapsto \mathbf{f}\}$; so it is satisfiable. However, under the interpretation $\{Q \mapsto \mathbf{t}, R \mapsto \mathbf{f}\}$ the formula does not hold; so it is not valid. Therefore, the formula is satisfiable but not valid.

how does a computer answer this?

(ii)

$$((P \leftrightarrow Q) \leftrightarrow P) \leftrightarrow Q$$

This formula is also satisfiable, but not valid. It is true under the interpretation $\{P \mapsto \mathbf{t}, Q \mapsto \mathbf{t}\}$ and so is satisfiable; but false under the interpretation $\{P \mapsto \mathbf{f}, Q \mapsto \mathbf{t}\}$ so is not valid. Therefore the formula is satisfiable but not valid.

how does a computer answer this?

(iii)

$$\exists xy\,[P(x,y) \to \forall xy\; P(x,y)]$$

This formula is satisfiable but not valid. The formula holds under the interpretation $\mathcal{I} = (\{2\}, \{P(x,y) \mapsto x = y\})$ – so the formula is satisfiable. However, the formula does not hold under the interpretation $\mathcal{I} = (\mathbb{N}, \{P(x,y) \mapsto x = y\})$. Therefore, the formula is satisfiable but not valid.

how does a computer answer this?

(iv)

$$[\forall x\,(P(x) \to Q(x)) \land \exists x P(x)] \to \forall x\; Q(x)$$

This formula is satisfiable but not valid. It holds under the interpretation $\mathcal{I} = (\{2\}, \{P(x) \mapsto x = 2, Q(x) = x = 2\}$ so is satisfiable, but does not hold $\mathcal{I} = (\mathbb{N}, \{P(x) \mapsto x = 2, Q(x) \mapsto x = 2\})$ so is not valid. Therefore, the formula is satisfiable but not valid.

how does a computer answer this?

(b) Briefly outline the semantics of first-order logic, taking as an example the formula $\forall xy\; f(x,y) = f(y,x)$

A formula in first-order logic is an element of a first-order language $\mathcal{L}$. Formulas with an interpretation $\mathcal{I} = (D, I)$ are either true or false. A formula is satisfiable if there is at least one interpretation for which it evaluates to true. A formula is valid if it is true under every interpretation. An interpretation $\mathcal{I}$ consists of a pair of a domain (the set of values from which existentially or universally bound quantifiers may be drawn from) and a mapping from function and predicates to implementations.

Variables in formulae are either bound (by $\forall$ or $\exists$) or free. A valuation $\mathcal{V}$ is a mapping from free variables to elements in D.

The example is satisfiable since there exist interpretations for which it is true. However, it does not always hold and so it is not valid. The formula is true under the

equality must denote itself

interpretation $\mathcal{I} = (\mathbb{N}, \{f(x, y) \mapsto x + y\})$. However, it does not hold under the interpretation $\mathcal{I} = (\mathbb{N}, \{f(x, y) \mapsto x - y\})$.

The given example has no free values, so any function works as a valuation.

(c) Exhibit a model that satisfies both of the following formulae (a is a constant):

$$\forall x \; g(x) \neq a$$
$$\forall xy \, [g(x) = g(y) \to x = y]$$

$$\mathcal{I} = (\mathbb{N}, \{a \mapsto 0, g(x) \mapsto x + 1\})$$

This only holds for infinite domains.
For a finite domain, there aren't enough outputs!
Proof via pigeonhole principle

The sequent calculus can fail to terminate by repeatedly pulling examples