Assignment8 – Reasoning for Propositional Logic

Problem 8.1 (Calculi Comparison)

Prove (or disprove) the validity of the following formulae in i) Natural Deduction ii) Tableau and iii) Resolution:

1. $P \wedge Q \Rightarrow (P \vee Q)$

Solution:

1. ND:

(1)	1	$P \wedge Q$	Assumption
(2)	1	P	$\wedge E_l$ (on 1)
(3)	1	$P \vee Q$	$\vee I_l$ (on 2)
(4)		$P \wedge Q \Rightarrow (P \vee Q)$	$\Rightarrow I$ (on 1 and 3)

2. Tableau:

(1)	$(P \wedge Q \Rightarrow (P \vee Q))^F$	
(2)	$(P \wedge Q)^T$	(from 1)
(3)	$(P \vee Q)^F$	(from 1)
(4)	P^T	(from 2)
(5)	Q^T	(from 2)
(6)	P^F	(from 3)

3. Resolution: $P \wedge Q \Rightarrow (P \vee Q)$: We negate and build a CNF:

$$\begin{aligned} &P \wedge Q \wedge \neg(P \vee Q) \\ \equiv &P \wedge Q \wedge \neg P \wedge \neg Q \end{aligned}$$

yielding clauses $\{P^T\}, \{Q^T\}, \{P^F\}, \{Q^F\}$

---

2. $(A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C) \Rightarrow C$

Solution:

1. ND:

(1)	1	$(A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C)$	Assumption
(2)	1	$A \vee B$	$\wedge E_l$ (on 1)
(3)	1	$(A \Rightarrow C) \wedge (B \Rightarrow C)$	$\wedge E_r$ (on 1)
(4)	1	$A \Rightarrow C$	$\wedge E_l$ (on 3)
(5)	1	$B \Rightarrow C$	$\wedge E_r$ (on 3)
(6)	1,6	A	Assumption
(7)	1,6	C	$\Rightarrow E$ (on 4 and 6)
(8)	1,8	B	Assumption
(9)	1,8	C	$\Rightarrow E$ (on 5 and 8)
(10)	1	C	$\vee E$ (on 2, 7 and 9)
(11)		$(A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C) \Rightarrow C$	$\Rightarrow I$ (on 1 and 10)

2. Tableau:

<table>
<tr><td>(1)</td><td colspan="2">$((A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C) \Rightarrow C)^{\mathsf{F}}$</td><td></td></tr>
<tr><td>(2)
(3)</td><td colspan="2">$((A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C))^{\mathsf{T}}$
C^{F}</td><td>(from 1)
(from 1)</td></tr>
<tr><td>(4)
(5)
(6)</td><td colspan="2">$(A \vee B)^{\mathsf{T}}$
$(A \Rightarrow C)^{\mathsf{T}}$
$(B \Rightarrow C)^{\mathsf{T}}$</td><td>(from 2)
(from 2)
(from 2)</td></tr>
<tr><td>(7)</td><td>A^{T}
A^{T} | C^{T} ‖ (split on 5)</td><td>B^{T}
B^{F} | C^{T} ‖ (split on 6)</td><td>(split on 4)</td></tr>
</table>

3. Resolution:
$(A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C) \Rightarrow C$: We negate and build a CNF:

$$
\begin{aligned}
&(A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C) \wedge \neg C \\
\equiv &(A \vee B) \wedge (\neg A \vee C) \wedge (\neg B \vee C) \wedge \neg C
\end{aligned}
$$

yielding clauses $\{A^{\mathsf{T}}, B^{\mathsf{T}}\}, \{A^{\mathsf{F}}, C^{\mathsf{T}}\}, \{B^{\mathsf{F}}, C^{\mathsf{T}}\}, \{C^{\mathsf{F}}\}$.
Resolving yields:

$$
\begin{aligned}
\{A^{\mathsf{F}}, C^{\mathsf{T}}\} + \{C^{\mathsf{F}}\} &\Longrightarrow \{A^{\mathsf{F}}\} \\
\{B^{\mathsf{F}}, C^{\mathsf{T}}\} + \{C^{\mathsf{F}}\} &\Longrightarrow \{B^{\mathsf{F}}\} \\
\{A^{\mathsf{T}}, B^{\mathsf{T}}\} + \{A^{\mathsf{F}}\} &\Longrightarrow \{B^{\mathsf{F}}\} \\
\{B^{\mathsf{T}}\} + \{B^{\mathsf{F}}\} &\Longrightarrow \emptyset
\end{aligned}
$$

---

3. $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$

Solution:

1. ND:

(1)		$P \vee \neg P$	TND
(2)	2	P	Assumption
(3)	2,3	$(P \Rightarrow Q) \Rightarrow P$	Assumption
(4)	2	$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	$\Rightarrow I$ (on 3 and 2)
(5)	5	$\neg P$	Assumption
(6)	5,6	$(P \Rightarrow Q) \Rightarrow P)$	Assumption
(7)	5,6,7	P	Assumption
(8)	5,6,7	F	FI (on 5 and 7)
(9)	5,6,7	Q	FE (on 8)
(10)	5,6	$P \Rightarrow Q$	$\Rightarrow I$ (on 7 and 9)
(11)	5,6	P	$\Rightarrow E$ (on 6 and 10)
(12)	5	$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	$\Rightarrow I$ (on 6 and 11)
(13)		$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	$\vee E$ (on 1, 4 and 12)

2. Tableau:

(1)	$(((P \Rightarrow Q) \Rightarrow P) \Rightarrow P)^{\mathsf{F}}$	
(2)	$((P \Rightarrow Q) \Rightarrow P)^{\mathsf{T}}$	(from 1)
(3)	P^{F}	(from 1)
(4)	$(P \Rightarrow Q)^{\mathsf{F}}$ / (5) P^{T} $\|$ (from 4) $\;\big|\;$ P^{T}	(split on 2)

3. Resolution:
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$: We negate and build a CNF:

$$
\begin{aligned}
&((P \Rightarrow Q) \Rightarrow P) \wedge \neg P \\
\equiv&(\neg(P \Rightarrow Q) \vee P) \wedge \neg P \\
\equiv&(P \wedge \neg Q \vee P) \wedge \neg P \\
\equiv&(P \vee P) \wedge (\neg Q \vee P) \wedge \neg P
\end{aligned}
$$

yielding clauses $\{P^{\mathsf{T}}\}, \{Q^{\mathsf{F}}, P^{\mathsf{T}}\}, \{P^{\mathsf{F}}\}$.

---

4. Can you identify any advantages or disadvantage of the calculi, and in which situations?

Problem 8.2 (Equivalence of CSP and SAT)
We consider

- CSPs $\langle V, D, C\rangle$ with finite domains as before
- Boolean satisfiability problems $\langle V, A\rangle$ where V is a set of propositional variables and A is a propositional formula over V.

We will show that these problem classes are equivalent by reducing their instances to each other.

1. Given a **SAT** instance $P = \langle V, A \rangle$, define a CSP instance $P' = \langle V', D', C' \rangle$ and two bijections:

   - f mapping satisfying assignments of P to solutions of P',
   - and f' the inverse of f.

   We already know that constraint networks are equivalent to higher-order CSPs. Therefore, it is sufficient to give a higher-order CSP.

   Solution: We define P' by $V' = V$, $D_v = \{\mathtt{T}, \mathtt{F}\}$ for every $v \in V$, and $C = \{A\}$, i.e., C contains the single higher-order constraint that holds if an assignment to V' (seen as an propositional assignment to V) satisfies A.
   f and f' are the identity.

   ---

2. Given a CSP instance $\langle V, D, C \rangle$, define a **SAT** instance (V', A') and bijections as above.

   Solution: We define P' as follows. V' contains variables p'_{va} for every $v \in V$ and $a \in D_v$. The intuition behind p'_{va} is that v has value a.
   A' is the conjunction of the following formulas:

   - for all $v \in V$ with $D_v = \{a_1, \ldots, a_n\}$, the formula $p'_{va_1} \vee \ldots \vee p'_{va_n}$ (i.e., v must have at least one value)
   - for all $v \in V$, and $a, b \in D_v$ with $a \neq b$, the formula $p'_{va} \Rightarrow \neg p'_{vb}$ (i.e., v can have at most one value)
   - for all C_{vw} and $(a,b) \notin C_{vw}$, the formula $\neg(p'_{va} \wedge p'_{wb})$ (i.e., every constraint must be satisfied)

   The bijection f maps a solution α of P to a A'-satisfying propositional assignment φ for V' as follows: for all v, a, we put $\varphi(p'_{va}) = \mathtt{T}$ if $\alpha(v) = a$ and $\varphi(p'_{va}) = \mathtt{F}$ otherwise.
   The inverse bijection f' maps an A'-satisfying assignment φ to a solution α of P as follows: for all v we put $\alpha(v) = a$ where a is the unique value for which $\varphi(p'_{va}) = \mathtt{T}$.

Problem 8.3 (Satisfiability and Validity)

Consider propositional logic with propositional variables $\{P, Q, R\}$. For each of the following statements, give a counter-example that refutes it:

1. The formula $(P \wedge Q \vee Q \wedge R) \Rightarrow (\neg P \vee \neg R)$ is satisfied by all assignments.

   Solution: A falsifying assignment is $P = Q = R = 1$.

   ---

2. If a formula F cannot be proved in the natural deduction calculus, then $\neg F$ is valid.

Solution: P cannot be proved but is satisfiable thus $\neg F$ is not valid.

---

3. If, for two formulas F, G, all assignments satisfy $F \Rightarrow G$ and no assignment satisfies F, then no assignment satisfies G.

Solution: Let $F = P \wedge \neg P$ and $G = P$. Then every assignments satisfies $F \Rightarrow G$, none satisfies F, and G is satisfiable.

Problem 8.4 (SAT Solver)

We want to implement a basic SAT solver. To simplify the I/O, we assume the following:

- The propositional variables are numbered $P_1, P_2, \ldots$.
- The input formula A is already in CNF and represented as a list of lists of integers such that:
  - the main list $[C_1, \ldots, C_m]$ represents A as a list of conjuncts, i.e., $A = C_1 \wedge \ldots \wedge C_m$.
  - each $C_i = [L_1, \ldots, L_{n_i}]$ represents one of the disjunctions, i.e., $C_i = L_1 \vee \ldots \vee L_{n_i}$
  - each integer L represent a literal, with negative numbers representing negations.

For example, $[[1, 2, -3], [2, -5]]$ represents the formula $(P_1 \vee P_2 \vee \neg P_3) \wedge (P_2 \vee \neg P_5)$.

1. Implement a naive SAT solver, e.g., by checking all assignments.
   Run an experiment where you generate large formulas with many variables and test how well your implementation scales. Think about what kind of formulas make your code take the longest and use those, e.g., unsatisfiable formulas take longer if they require testing all assignments.
2. Implement optimizations and repeat your experiments.
   For example, you can implement DPLL or clause learning.