Problem 1. 4 pts.

To get the size of a tree, it makes sense to add 1 + (the size of each subtree). When the height of each subtree is zero, you’ll add one to indicate a leaf node.

\[ |t| = \begin{cases} 1 + |t_1| + |t_2| & \text{if } t =< t_1, t_2 > \\ 1 & \text{else} \end{cases} \]

To get the size of a tree, it makes sense to compute the max of each subtree and add 1.

\[ \text{height}(t) = \begin{cases} 1 + \max\{\text{height}(t_1), \text{height}(t_2)\} & \text{if } t =< t_1, t_2 > \\ 0 & \text{else} \end{cases} \]

Problem 2. 8 pts.

(a) To show that \( \mathcal{A} \) is a lattice, we need to show that (1) \((\mathbb{N} - \{0\}, \leq)\) is a poset, (2) the greatest lower bound of \( a \) and \( b \) in the ordering \( \leq \) exists, is unique and is represented by \( \gcd(a, b) \). (3) The least upper bound for \( a \) and \( b \) in the ordering \( \leq \) exists, is unique and is the result of the operation \( a \land b \).

1. To show that \((\mathbb{N} - \{0\}, \leq)\) is a poset, we need to show:
   - Reflexivity
     \( a \leq a \) is true because \( a1 = a \)
   - Anti-Symmetry
     \( a \leq b \) and \( b \leq a \) then \( a = b \)
     \( a \leq b \) means \( a \leq b \)
     \( b \leq a \) means \( b \leq a \)
     \( a \leq b \) and \( b \leq a \) iff \( a = b \)
   - Transitive
     if \( a \leq b \) then there exists \( n \) such that \( na = b \) and \( n \in \mathbb{N} - \{0\} \)
     if \( b \leq c \) then there exists \( m \) such that \( mb = c \) and \( m \in \mathbb{N} - \{0\} \)
     Therefore \( c = mb = mna \) so \( a \leq c \)

2. Now we need to show that \( \gcd \):
   - Exists
     \( \gcd(a, b) \) exists for any pair of input \( a, b \in \mathbb{N} - \{0\} \)
   - Is Unique
     For every \( a \) and \( b \) there is only once \( c \) such that \( c = \gcd(a, b) \)
   - Greatest Lower Bound
     By definition of \( \gcd \), \( c = \gcd(a, b) \) so \( c|a \) and \( c|b \) and thus no \( d \) exists such that \( c < d \) and \( d|a \) and \( d|b \). Therefore only one \( c \) exists and is the greatest lower bound.
3. Least Upper Bound:
   By definition of lcm, \( c = \text{lcm}(a, b) \) so \( c|a \) and \( c|b \) and thus no \( d \) exists such that \( c > d \) and \( d|a \) and \( d|b \). Therefore only one \( c \) exists and is the Least upper bound.

(b) Answer Courtesy Roman Bogdanowski.
We can show that \( \leq \) is a distributive lattice by showing that:

\[
\begin{align*}
  a \land (b \lor c) &= (a \land b) \lor (a \land c) \\
  a \lor (b \land c) &= (a \lor b) \land (a \lor c)
\end{align*}
\]

We can argue that this is true because of The Unique Prime Factorization Theorem. It states that every integer greater than 1 either is prime itself or is the product of prime numbers, and that this product is unique, up to the order of the factors. We can see that for all \( p, r, q \in \mathbb{N} - \{0\} \) if one of the numbers is 1 then both of the above equations hold true. If not, we can use the theorem to show that the above statements are true. By the theorem, let \( \{m_1, m_2, \ldots, m_n\} \) be distinct primes that occur in the prime factorization of \( p, q, r \). Then, we can decompose \( p, q, r \) as \( p = m_1^{x_1} m_2^{x_2} \ldots m_n^{x_n}, q = m_1^{y_1} m_2^{y_2} \ldots m_n^{y_n}, r = m_1^{z_1} m_2^{z_2} \ldots m_n^{z_n} \). So let’s show that the above statements are true.

\[
\begin{align*}
  a \land (b \lor c) &= (a \land b) \lor (a \land c) \\
  a \lor (b \land c) &= (a \lor b) \land (a \lor c)
\end{align*}
\]

Problems

Problem 3. 4 pts.
AnswerCourtesy Roman Bogdanowski.

(A) \( \phi ::= p \land \neg \phi \lor \phi \lor \phi \)

(B) \( \phi ::= p \lor \neg p \land \phi \lor \phi \land \phi \)

Lemma

Every propositional WFF \( \phi \) in the syntax of BNF definition (A) can be translated in linear time into an equivalent propositional WFF \( \varphi \) in the syntax of definition (B) such that \( |\varphi| < \frac{3}{2}|\phi| \).

Let’s define recursive algorithm Translate(\( \phi \)), that accepts WFF \( \psi \) in the syntax of BNF definition (B).

\[
\begin{align*}
  \text{if } \phi \in \{p \neq p\} \text{ then } \text{Translate}(\phi) \text{ returns } \phi \\
  \text{if } \phi = \phi_1 \land \phi_2, \text{ then } \text{Translate}(\phi) \text{ returns Translate}(\phi_1) \land \text{Translate}(\phi_2). \\
  \text{if } \phi = \phi_1 \lor \phi_2, \text{ then } \text{Translate}(\phi) \text{ returns Translate}(\phi_1) \lor \text{Translate}(\phi_2). \\
  \text{if } \phi = \neg \neg \phi, \text{ then } \text{Translate}(\phi) \text{ returns } \phi.
\end{align*}
\]
• if $\phi = \neg(\phi_1 \land \phi_2)$, then $\text{Translate}(\phi)$ returns $\text{Translate}(\neg \phi_1) \lor \text{Translate}(\neg \phi_2)$.

• if $\phi = \neg(\phi_1 \lor \phi_2)$, then $\text{Translate}(\phi)$ returns $\text{Translate}(\neg \phi_1) \land \text{Translate}(\neg \phi_2)$.

From the above, it is clear that $\text{Translate}(\phi)$ runs in linear time because it touches every node in the parse tree at most one time. It also clear that the output WFF will be equivalent to the input WFF, but will be in definition (B). Now let’s consider sizes of $\phi$ and $\text{Translate}(\phi)$. Translate() only modifies length of a WFF in a case when it adds or removes a negation(s). New negations are only added if there was at least one negation to start with. (last two cases). So let’s consider the worst case when we start with one negation and it gets 'push in,' so that $\phi$ has one negation and $\text{Translate}(\phi)$ has $N$ negations (where $N$ is the number of atoms in $\phi$ (and therefore in $\text{Translate}(\phi)$). If we have $N$ atoms, then we have that $\phi$ and therefore $\text{Translate}(\phi)$ has $N - 1$ connectives. So, in the worst case, $|\phi| = N + (N - 1) + 1 = 2N$ (we have $N$ atoms $N - 1$ connectives, 1 negation). And $|\text{Translate}(\phi)| = N + (N - 1) + N = 3N - 1$ (We have $N$ atoms, $N - 1$ connectives, $N$ negation). So, in the worst case $|\text{Translate}((\phi))| = 3N - 1 < \frac{3}{2}|\phi| = \frac{3}{2}2N = 3N$

Problem 4. 4 pts. An example:

$$\varphi = (T \rightarrow q) \land (T \rightarrow \neg q)$$

This is not valid for any $A$, therefore it is easy to see that Horn’s algorithm breaks.