1. (10 marks) **Note updated problem statement**

Consider the encoding of tournament scheduling in SAT presented in Lecture 7 (B/W slide 11). Encode the following constraints as propositional formulas. You may introduce propositional atoms other than \( p_{x,y,z} \). In this case, formulate constraints between the new variables and \( p_{x,y,z} \), also as propositional formulas. For example, if you introduce a variable \( b_{1,4} \), which encodes whether Team 1 has a bye in Date 4, you need the following formula that expresses the connection between \( b_{1,4} \) and the \( p \) variables:

\[
(b_{1,4} \rightarrow \neg p_{1,2,4} \land \neg p_{1,3,4} \land \cdots \land \neg p_{1,9,4} \land \neg p_{2,1,4} \land \neg p_{3,1,4} \land \cdots \land \neg p_{9,1,4}) \land \\
(\neg p_{1,2,4} \land \neg p_{1,3,4} \land \cdots \land \neg p_{1,9,4} \land \neg p_{2,1,4} \land \neg p_{3,1,4} \land \cdots \land \neg p_{9,1,4} \rightarrow b_{1,4})
\]

As in the previous formula, you may use the \( \cdots \) notation, if the meaning is clear.

- UNC (Team 1) plays Duke (Team 2) in the last date and in Date 11.

**Solution.** The slides mention that \( p_{x,y,z} \) encodes that Team \( x \) has a home game against Team \( y \) in Date \( z \). Thus, the requirement that UNC plays Duke in Date 11 needs to consider both possibilities for the venue: UNC plays home or Duke plays home. This is encoded by:

\[
p_{1,2,11} \lor p_{2,1,11}
\]

Similarly, the two options need to be considered for the last date. Overall, the following formula captures the requirement:

\[
(p_{1,2,11} \lor p_{2,1,11}) \land (p_{1,2,18} \lor p_{2,1,18})
\]

- The following pairings must occur at least once in Dates 11 to 18: Duke (Team 2) – GT (Team 3), Duke (Team 2) – Wake (Team 4), GT (Team 3) – UNC (Team 1), UNC (Team 1) – Wake (Team 4).

**Solution.** The fact that a pairing occurs at least once can be represented by a disjunction of all possibilities, again considering both options for the venue. Overall, the following conjunction results:

\[
(p_{2,3,11} \lor p_{3,2,11} \lor p_{2,3,12} \lor p_{3,2,12} \lor \cdots \lor p_{2,3,18} \lor p_{3,2,18}) \land \\
(p_{2,4,11} \lor p_{4,2,11} \lor p_{2,4,12} \lor p_{4,2,12} \lor \cdots \lor p_{2,4,18} \lor p_{4,2,18}) \land \\
(p_{3,1,11} \lor p_{1,3,11} \lor p_{3,1,12} \lor p_{1,3,12} \lor \cdots \lor p_{3,1,18} \lor p_{1,3,18}) \land \\
(p_{1,4,11} \lor p_{4,1,11} \lor p_{1,4,12} \lor p_{4,1,12} \lor \cdots \lor p_{1,4,18} \lor p_{4,1,18})
\]

- No team can play away on both last dates.
Solution. A team $x$ plays away on a date, if there is some other team that plays home against $x$ on that date. Thus, one way of expressing the constraint is:

$$\neg\left(\left((p_{2,1,17} \lor p_{3,1,17} \lor p_{4,1,17} \lor \cdots \lor p_{9,1,17})\right) \land \left(p_{2,1,18} \lor p_{3,1,18} \lor p_{4,1,18} \lor \cdots \lor p_{9,1,18}\right)\right)\land$$

$$\neg\left(\left((p_{1,2,17} \lor p_{3,2,17} \lor p_{4,2,17} \lor \cdots \lor p_{9,2,17})\right) \land \left(p_{1,2,18} \lor p_{3,2,18} \lor p_{4,2,18} \lor \cdots \lor p_{9,2,18}\right)\right)$$

$$\land \cdots \land$$

$$\neg\left(\left((p_{1,9,17} \lor p_{2,9,17} \lor p_{3,9,17} \lor \cdots \lor p_{8,9,17})\right) \land \left(p_{1,9,18} \lor p_{2,9,18} \lor p_{3,9,18} \lor \cdots \lor p_{8,9,18}\right)\right)$$

- Dates 1 and 8 are mirrored. This means two teams play each other in Date 1, iff they play each other in Date 8.
Solution. Mirroring can be enforced by stating the double-implication of the corresponding \( p \) variables:

\[
((p_{1,2,1} \rightarrow p_{1,2,8}) \land (p_{2,1,8} \rightarrow p_{1,2,1}) \land \\
(p_{1,3,1} \rightarrow p_{3,1,8}) \land (p_{3,1,8} \rightarrow p_{1,3,1}) \land \\
\vdots \\
(p_{1,9,1} \rightarrow p_{9,1,8}) \land (p_{9,1,8} \rightarrow p_{1,9,1})
\]

\[
\land \\
((p_{2,1,1} \rightarrow p_{1,2,8}) \land (p_{1,2,8} \rightarrow p_{2,1,1}) \land \\
(p_{2,3,1} \rightarrow p_{3,2,8}) \land (p_{3,2,8} \rightarrow p_{2,3,1}) \land \\
\vdots \\
(p_{2,9,1} \rightarrow p_{9,2,8}) \land (p_{9,2,8} \rightarrow p_{2,9,1})
\]

\[
\land \\
\vdots \\
\land \\
((p_{9,1,1} \rightarrow p_{1,9,8}) \land (p_{1,9,8} \rightarrow p_{9,1,1}) \land \\
(p_{9,2,1} \rightarrow p_{2,9,8}) \land (p_{2,9,8} \rightarrow p_{9,2,1}) \land \\
\vdots \\
(p_{8,9,1} \rightarrow p_{9,8,8}) \land (p_{9,8,8} \rightarrow p_{8,9,1})
\]

2. (10 marks) (a.k.a. exercise 8 in the lecture notes)
We inductively define the set of binary trees as follows:

\[
\text{Tree} = \bullet \vert \text{Tree Tree}
\]

Define (paper and Coq) a function that doubles all of the leaves in a tree. For example, this function should behave as follows:

\[
\begin{align*}
\text{double}(\bullet) &= \bullet \bullet \\
\text{double}(\bullet \bullet) &= \bullet \bullet \bullet \bullet \\
\text{double}(\bullet \bullet \bullet) &= \bullet \bullet \bullet \bullet \bullet \bullet
\end{align*}
\]

Solution. Here is the \texttt{double} function:

\[
double(t) = \begin{cases} \\
\text{double}(t_l) \text{ double}(t_r) & \text{when } t = t_l \; t_r \\
\text{when } t = \bullet
\end{cases}
\]
And here is the Coq implementation:

```coq
Fixpoint double (t : Tree) : Tree :=
  match t with
  | Leaf => Node Leaf Leaf
  | Node tl tr => Node (double tl) (double tr)
  end.
```

3. (10 marks) (a.k.a. exercise 10 in the lecture notes)

Using the `double` function you defined for the previous problem, and the following definition for `leaves`:

\[
\text{leaves}(t) \equiv \begin{cases} 
1 & \text{when } t = \bullet \\
\text{leaves}(t_l) + \text{leaves}(t_r) & \text{when } t = t_l \xrightarrow{} t_r
\end{cases}
\]

Please prove:

\[
\forall t \; (\text{leaves}(\text{double}(t)) = \text{leaves}(t) + \text{leaves}(t))
\]

**Solution.** By induction on the structure of \( t \) using induction hypothesis

\[
\text{IH}(t) \equiv \text{leaves}(\text{double}(t)) = \text{leaves}(t) + \text{leaves}(t)
\]

- **Case 1**: \( t = \bullet \). We have

  \[
  \text{leaves}(\text{double}(\bullet)) = \text{leaves}(\bullet \xrightarrow{} \bullet) = \text{leaves}(\bullet) + \text{leaves}(\bullet)
  \]

  This is enough to prove case 1.

- **Case 2**: \( t = t_l \xrightarrow{} t_r \). We can assume the induction hypotheses:

  \[
  \text{leaves}(\text{double}(t_l)) = \text{leaves}(t_l) + \text{leaves}(t_l)
  \]
  \[
  \text{leaves}(\text{double}(t_r)) = \text{leaves}(t_r) + \text{leaves}(t_r)
  \]

  We therefore have:

  \[
  \begin{align*}
  \text{leaves}(\text{double}(t_l \xrightarrow{} t_r)) &= \text{leaves}(\text{double}(t_l) \xrightarrow{} \text{double}(t_r)) \\
  &= \text{leaves}(\text{double}(t_l)) + \text{leaves}(\text{double}(t_r)) \\
  &= \text{leaves}(t_l) + \text{leaves}(t_l) + \text{leaves}(t_r) + \text{leaves}(t_r) \\
  &= (\text{leaves}(t_l) + \text{leaves}(t_r)) + (\text{leaves}(t_l) + \text{leaves}(t_r)) \\
  &= \text{leaves}(t_l \xrightarrow{} t_r) + \text{leaves}(t_l \xrightarrow{} t_r)
  \end{align*}
  \]

  This is enough to prove case 2.
Thus by structural induction we have proved that
\[ \forall t \, (\text{leaves(double}(t)) = \text{leaves}(t) + \text{leaves}(t)) \]

4. (10 marks)
Now we define the function \( \text{nodes} \) as follows:
\[
\text{nodes}(t) \equiv \begin{cases} 
0 & \text{when } t = \bullet \\
1 + \text{nodes}(t_l) + \text{nodes}(t_r) & \text{when } t = t_l \quad t_r 
\end{cases}
\]
Please prove:
\[ \forall t \, (\text{nodes(double}(t)) = \text{nodes}(t) + \text{leaves}(t)) \]

**Solution.** By induction on the structure of \( t \) using induction hypothesis
\[ \text{IH}(t) \equiv \text{nodes(double}(t)) = \text{nodes}(t) + \text{leaves}(t) \]

- **Case 1:** \( (t = \bullet) \). We have
  \[ \text{nodes(double}(\bullet)) = \text{nodes}(\bullet \bullet) = 1 + \text{nodes}(\bullet) + \text{nodes}(\bullet) = 1 + 0 + 0 = 0 + 1 = \text{nodes}(\bullet) + \text{leaves}(\bullet) \]
  This is enough to prove case 1.
- **Case 2:** \( (t = t_l \quad t_r) \). We can assume the induction hypotheses:
  \[ \text{nodes(double}(t_l)) = \text{nodes}(t_l) + \text{leaves}(t_l) \]
  \[ \text{nodes(double}(t_r)) = \text{nodes}(t_r) + \text{leaves}(t_r) \]
  We then have:
  \[
  \text{nodes(double}(t_l \quad t_r)) = \text{nodes}(\text{double}(t_l) \quad \text{double}(t_r)) = 1 + \text{nodes(double}(t_l)) + \text{nodes(double}(t_r)) = 1 + \text{nodes}(t_l) + \text{leaves}(t_l) + \text{nodes}(t_r) + \text{leaves}(t_r) = (1 + \text{nodes}(t_l) + \text{nodes}(t_r)) + (\text{leaves}(t_l) + \text{leaves}(t_r)) = \text{nodes}(t_l \quad t_r) + \text{leaves}(t_l \quad t_r) \]
  Thus by structural induction we have proved that
  \[ \forall t \, (\text{nodes(double}(t)) = \text{nodes}(t) + \text{leaves}(t)) \]
5. (15 marks) (a.k.a. exercise 11 in the lecture notes)

Suppose we attempt to define streams inductively via the rule

\[ \frac{n : \text{nat} \quad s : \text{stream}}{n \cdot s : \text{stream}} \]

Prove that in that case, stream is empty; that is,

\[ \neg \exists s : \text{stream}(\top) \]

Note that this is (deMorgan-) equivalent to:

\[ \forall s : \text{stream}(\bot) \]

If you set this up correctly, the proof should be very short.

**Solution.** By induction on the structure of \( s \). We use \( \text{IH}(s) = \bot \).

- Case 1: \((s = n \cdot s')\). We can assume \( \text{IH}(s') \), that is, \( \bot \). Since we are trying to prove \( \text{IH}(s) \), that is, \( \bot \), we are done.

Thus by we have proved \( \forall s : \text{stream}(\bot) \) by induction.

6. (15 marks) (a.k.a. exercise 5 in the lecture notes)

Give a series of rules and a complete and invertible set of objects satisfying those rules that is neither the least (inductive) nor greatest (coinductive).

**Solution.** For the rules, use the following:

\[ \frac{t : \tau}{Y(t) : \tau} \quad \frac{t : \tau}{Z(t) : \tau} \]

Define the set \( S \) as follows:

\[ S \equiv \{ Y(Y(Y(\ldots))), \text{(call this element } Y_\omega) \]

\[ Z(Y_\omega), \quad Z(Z(Y_\omega), \quad Z(Z(Z(Y_\omega))), \quad \ldots \]

\[ Y(Z(Y_\omega)), \quad Y(Y(Z(Y_\omega))), \quad Y(Y(Y(Z(Y_\omega)))), \quad \ldots \]

\[ Z(Y(Z(Y_\omega))), \quad Z(Y(Y(Z(Y_\omega)))), \quad \ldots \]

\[ Y(Y(Z(Z(Y_\omega)))), \quad \ldots \]

\[ \ldots \} \]

In other words, \( S \) contains the infinite-Y element \( Y_\omega \), plus all finite sequences of \( Z \) and \( Y \) (including the empty sequence), followed by a single
Z, followed by $Y_\omega$. The set $S$ is complete since given an element $s \in S$, applying the constructors $Y$ or $Z$ one time still leaves you in $S$ (we just made the finite sequence at the beginning one element longer, except in the case where we added an extra $Y$ to the front of $Y_\omega$, which just gives us $Y_\omega$ again). The set $S$ is invertible since every element is made from the rules $Y\tau$ and $Z\tau$. The set $S$ is not the least set since the least complete and invertible set satisfying the rules $Y\tau$ and $Z\tau$ is empty (see problem 5). The set $S$ is not the greatest complete and invertible set since it does not contain, among other things, the infinite-$Z$ element:

$$Z_\omega \equiv Z(Z(Z(\ldots))) \notin S$$

7. (20 marks)

Please prove

$$\forall t \ (t = \overline{t} \Rightarrow \bot)$$

You can assume that generators are injective; that is, from

$$\overline{t_{11}} \overline{t_{12}} = \overline{t_{21}} \overline{t_{22}}$$

you may conclude

$$t_{11} = t_{21} \quad \text{and} \quad t_{12} = t_{22}$$

You may also assume that

$$\forall t_1 \forall t_2 (\bullet = \overline{t_1} \overline{t_2} \Rightarrow \bot)$$

**Solution.** We will first prove the related fact:

$$\forall t \forall t' \ (t = \overline{t'} \Rightarrow \bot)$$

By induction on the structure of $t$ using the induction hypothesis:

$$\mathcal{H}'(t) \equiv \forall t' \ (t = \overline{t'} \Rightarrow \bot)$$

- **Case 1:** ($t = \bullet$). We want to prove $\mathcal{H}'(\bullet)$, that is,

$$\forall t' \ (\bullet = \overline{t'} \Rightarrow \bot)$$

This is automatic from the given assumption

$$\forall t_1 \forall t_2 ((\bullet = \overline{t_1} \overline{t_2}) \Rightarrow \bot)$$

So we are done with case 1.
Case 2: \((t = t_l \wedge t_r)\). We assume \(IH'(t_l)\) and \(IH'(t_r)\). We will only need \(IH'(t_l)\), i.e.:

\[\forall t' \; (t_l = t_l' \Rightarrow \bot)\]

Now assume \(t = t_l \wedge t_r\); that is,

\[t_l \wedge t_r = t_l \; t_r \; t_l \; t_r\]

Since generators are injective we have

\[t_l = t_l \wedge t_r \quad \text{and} \quad t_r = t_l \wedge t_r\]

By \(IH'(t_l)\), we can conclude \(\bot\) by setting \(t' = t_r\). Thus we are done with case 2.

We have therefore proved

\[\forall t \forall t' \; (t = t_l \wedge t_r' \Rightarrow \bot)\]

by structural induction.

It is clear that this is strictly stronger than the original goal of

\[\forall t \; (t = t_l \Rightarrow \bot)\]

By simply setting \(t' = t\).