1 The Pigeon Hole Principle

For every natural number \( n \geq 2 \), the Pigeon Hole Principle (PHP) states: “If \( n \) pigeons sit in \( (n-1) \) holes, then some hole contains more than one pigeon.” We want to formalize PHP in propositional logic (PL). There are different ways of doing this, but perhaps the most natural is:

- Use propositional atom \( P_{i,j} \) to indicate that pigeon \( i \) is in hole \( j \), where \( 1 \leq i \leq n \) and \( 1 \leq j < n \).

With this formal representation of “pigeon \( i \) is in hole \( j \)” we can formalize PHP with the following PL formula \( \phi \):

\[
\phi \equiv \bigwedge_{1 \leq i \leq n} \left( \bigvee_{1 \leq j < n} P_{i,j} \right) \rightarrow \bigvee_{1 \leq i < k \leq n} \left( \bigvee_{1 \leq j < n} (P_{i,j} \wedge P_{k,j}) \right)
\]

where \( \bigwedge \) and \( \bigvee \) are shorthand notation to write long sequences of conjunctions and disjunctions, respectively. In particular, for the case \( n = 3 \), we get:

\[
\phi = (P_{1,1} \vee P_{1,2}) \wedge (P_{2,1} \vee P_{2,2}) \wedge (P_{3,1} \vee P_{3,2}) \rightarrow \\
(P_{1,1} \wedge P_{2,1}) \vee (P_{1,2} \wedge P_{2,2}) \vee (P_{1,1} \wedge P_{3,1}) \vee (P_{1,2} \wedge P_{3,2}) \vee (P_{2,1} \wedge P_{3,1}) \vee (P_{2,2} \wedge P_{3,2})
\]

**Exercise 1** Let \( f \) be a total function from \{1, \ldots, n\} to \{1, \ldots, n-1\}, and use the same propositional atoms \( P_{i,j} \) used in the formalization of PHP, to define PL formulas \( \varphi_1 \) and \( \varphi_2 \) such that:

1. \( \varphi_1 \) is satisfiable iff \( f \) is a total (possibly multivalued) function on \{1, \ldots, n\},
2. \( \varphi_2 \) is satisfiable iff \( f \) is not one-to-one.

Another way of formalizing PHP in PL is to define the formula \( \varphi' \equiv \varphi_1 \rightarrow \varphi_2 \).

**Exercise 2** This is an implementation exercise. The WFF \( \varphi \) that formalizes PHP is not only satisfiable, but also valid (or a tautology), i.e., all valuations of the atoms \( P_{i,j} \) should satisfy \( \varphi \). Use Isabelle, or any automated proof-assistant of your choice, to establish that \( \varphi \) is valid. Do the implementation for at least two cases, \( n = 3 \) and \( n = 4 \). Do you notice any difference in the execution times? How would you handle the case \( n = 10 \)?

2 A Two-Player Game: Tic-Tac-Toe

There are different ways, in different formal logics, of modeling Tic-Tac-Toe. If we only want to model the starting configuration and the winning configuration in the game, then propositional logic (PL) will do.
To make the game a little more interesting, consider Tic-Tac-Toe on a $K \times K$ board where $K \geq 3$. The game for $K = 3$ is the usual version. The game for $K = 4$ is shown in Figure 1, with a possible configuration of the board after 6 moves.

The first thing we need to do is to choose the propositional atoms for our modeling. For convenience, we write $[K]$ to denote the set of indices $\{1, 2, \ldots, K\}$. Here is a plausible choice:

- Use two-indexed propositional atoms, $P_{i,j}$ and $Q_{i,j}$ with $i, j \in [K]$, to identify the squares where $X$ and $O$ are located on the board. Specifically,
  
  $P_{i,j} = \begin{cases} 
  true & \text{if square } (i, j) \in [K] \times [K] \text{ contains } X, \\
  false & \text{if square } (i, j) \in [K] \times [K] \text{ does not contain } X,
  \end{cases}$

  $Q_{i,j} = \begin{cases} 
  true & \text{if square } (i, j) \in [K] \times [K] \text{ contains } O, \\
  false & \text{if square } (i, j) \in [K] \times [K] \text{ does not contain } O.
  \end{cases}$

The starting configuration is the configuration when no $X$ and no $O$ are yet placed on the board, which can be modeled by:

$$\varphi_{\text{start}} := (\bigwedge_{i,j \in [K]} \neg P_{i,j}) \land (\bigwedge_{i,j \in [K]} \neg Q_{i,j})$$

It should be clear that $\varphi_{\text{start}}$ is satisfied, i.e., made $true$, by the valuation that makes every $P_{i,j}$ and every $Q_{i,j}$ $false$.

We next model a winning configuration for $X$. But what is a winning configuration for $X$ when $K \geq 4$? There are four possible ways in which the $X$-player can win:

- $K$ occurrences of $X$ are placed in the same row of the board,
- $K$ occurrences of $X$ are placed in the same column of the board,
- $K$ occurrences of $X$ are placed along the first diagonal of the board,
- $K$ occurrences of $X$ are placed along the second diagonal of the board.
Note that the indeces of the first diagonal are \( \{(1,1), (2,2), \ldots, (K,K)\} \), while those of the second diagonal are \( \{(1,K), (2,K-1), \ldots, (K,1)\} \). We can thus model a winning configuration for the \( X \)-player with the formula:

\[
\varphi_{X\text{-win}} := \left( \bigvee_{i \in [K]} \bigwedge_{j \in [K]} P_{i,j} \right) \lor \left( \bigvee_{j \in [K]} \bigwedge_{i \in [K]} P_{i,j} \right) \lor \left( \bigwedge_{i \in [K]} P_{i,i} \right) \lor \left( \bigwedge_{i \in [K]} P_{i,K+1-i} \right)
\]

Exercise 3 The \( O \)-player in Tic-Tac-Toe wins by preventing the \( X \)-player from reaching a winning configuration. Define a propositional WFF \( \varphi_{O\text{-win}} \) which formally models a winning configuration for the \( O \)-player.

_Hint:_ A winning configuration for the \( O \)-player is not “symmetric” to a winning configuration for the \( X \)-player, _i.e._, the former cannot be obtained from the latter by replacing every \( P_{i,j} \) by \( Q_{i,j} \) in \( \varphi_{X\text{-win}} \).

Exercise 4 Generalize the notion of _winning configuration_ for the \( X \)-player (and similarly for the \( O \)-player) as follows:

- \( X \)-player wins iff \( K \) occurrences of \( X \) are placed in \( K \) contiguous squares on the board.

Call the resulting game Tic-Tac-Toe*.

Two squares of the board are _contiguous_ iff they have a side in common. For example, square \((2,3)\) and \((2,4)\) are contiguous. In this version of the game, \( K \) occurrences of \( X \) in the same row (or in the same column) is a winning configuration, just as it is in the usual Tic-Tac-Toe, but in contrast to the usual game, \( K \) occurrences of \( X \) along the first or the second diagonal is not a winning configuration.

Write a propositional WFF \( \psi_{X\text{-win}} \) which is satisfied iff \( \psi_{X\text{-win}} \) represents a winning configuration for the \( X \)-player in Tic-Tac-Toe*.