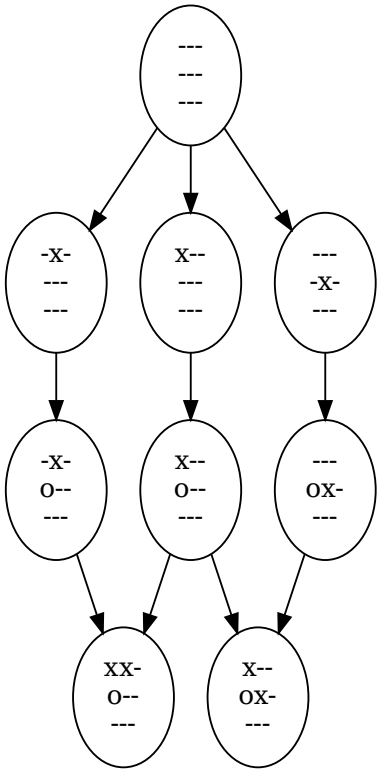
1. Erstelle den Spielzuggraphen für Tic Tac Toe.

   (a) Erstelle einen Algorithmus um alle möglichen Spielzüge in einem Tic Tac Toe Spiel darzustellen. Ausganglage ist ein leeres Spielfeld. Den ersten Zug führt immer *X* durch. Anschließend wird im Wechsel gespielt. [8 P]

   Stelle eine Spielkonfiguration über einen Knoten in einem Graph dar. Eine Kante zwischen zwei Knoten wird gezogen, wenn von einer Spielkonfiguration in eine andere übergegangen werden kann. Führe jede Spielkonfiguration bis zu einem Sieg von *X* oder *O* auf.

   In folgender Darstellung ist ein Auszug aus diesem Graphen dargestellt.



   Wähle eine geeignete Klassendarstellung auf Basis von `TicTacToeNode` für einen Knoten (Spielkonfiguration). Nutze die Graphenimplementierung aus der Lehrveranstaltung. Um von einem Knoten zu einem anderen eine Kante zu ziehen, nutze die Idee der Tiefen- oder Breitensuche.

   Erstelle die Klasse:

```
public class TicTacToeGraph<T> where T : TicTacToeNode
{
    /// <summary>Generates the tic tac toe play graph</summary>
    /// <returns>A graph representation of the play graph</returns>
    public Graph<T> Generate();
}

public class TicTacToeNode : IComparable<TicTacToeNode>
{

}
```

   (b) Erweitere beziehungsweise überschreibe in der Graph-Implementierung aus der Lehrveranstaltung die `ToString()`-Methode um für einen gegeben *gerichteten* Graphen die Dot-Darstellung [2 P]