using cAlgo.API;  
using cAlgo.API.Indicators;  
using cAlgo.API.Internals;  
using System;  
using System.Collections.Generic;  
  
namespace cAlgo.Indicators  
{  
    //  [Indicator(TimeZone = TimeZones.UTC, AccessRights = AccessRights.None)]  
    [Indicator(IsOverlay = true, TimeZone = **TimeZones**.**UTC**, AccessRights = AccessRights.FullAccess)]  
    public class ATRBasedStoplossTakeProfit : Indicator  
    {  
        [Parameter("ATR Lookback Period", DefaultValue = 30, Group = "Main", MinValue = 1)]  
        public int ATRLookbackPeriod { get; set; }  
  
        [Parameter("ATR Length", DefaultValue = 14, Group = "Main", MinValue = 1)]  
        public int ATRLength { get; set; }  
  
        [Parameter("Stop Loss Take Profit Type", DefaultValue = "ATR Surge Average", Group = "Main")]  
        public StopLossTakeProfitType StopLossTakeProfitTypeParameter { get; set; }  
  
        [Parameter("ATR Profit Multiplier", DefaultValue = 4, Group = "Main", MinValue = 0.1)]  
        public double ATRProfitMultiplier { get; set; }  
  
        [Parameter("ATR Loss Multiplier", DefaultValue = 1, Group = "Main", MinValue = 0.1)]  
        public double ATRLossMultiplier { get; set; }  
  
        [Parameter("MA Length", DefaultValue = 8, Group = "Main", MinValue = 1)]  
        public int MALength { get; set; }  
  
  
        [Output("Moving Average", LineColor = "blue",Thickness =2)]  
        public IndicatorDataSeries MovingAverage { get; set; }  
  
  
        [Output("Long Take Profit", LineColor = "Lime", Thickness = 2)]  
        public IndicatorDataSeries LongTakeProfit { get; set; }  
  
        [Output("Short Take Profit", LineColor = "Maroon", Thickness = 2)]  
        public IndicatorDataSeries ShortTakeProfit { get; set; }  
  
        [Output("Short Stop Loss", LineColor = "Fuchsia", Thickness = 2)]  
        public IndicatorDataSeries ShortStopLoss { get; set; }  
  
        [Output("Long Stop Loss", LineColor = "Orange", Thickness = 2)]  
        public IndicatorDataSeries LongStopLoss { get; set; }  
  
  
  
        private AverageTrueRange \_atr;  
        private IndicatorDataSeries tnatrSim;  
        private IndicatorDataSeries natrSim;  
        private IndicatorDataSeries natr;  
        private IndicatorDataSeries natrSH;  
        private bool surgeReset;  
  
        private List<double> \_natrSHArray;  
        private List<double> \_tnatrSHArray;  
        private double \_psnatr;  
        private double \_lsnatr;  
        private double \_prftpcnt;  
        private double \_losspcnt;  
  
        protected override void Initialize()  
        {  
            bool v = System.Diagnostics.Debugger.Launch();  
            \_atr = Indicators.AverageTrueRange(ATRLength,MovingAverageType.Simple);  
            \_natrSHArray = new List<double>();  
            \_tnatrSHArray = new List<double>();  
            \_psnatr = 0;  
            \_lsnatr = 0;  
            \_prftpcnt = 0;  
            \_losspcnt = 0;  
            tnatrSim = CreateDataSeries();  
            natr = CreateDataSeries();  
            natrSim = CreateDataSeries();  
            natrSH = CreateDataSeries();  
        }  
  
        public override void Calculate(int index)  
        {  
            tnatrSim[index] = 100 \* \_atr.Result.LastValue / Bars.LastBar.Close;  
            natr[index] = \_atr.Result.LastValue;  
            natrSim[index] = Indicators.AverageTrueRange(1,MovingAverageType.Simple).Result.LastValue;  
            natrSH[index] = GetHighest(natrSim.LastValue, ATRLookbackPeriod, natrSim.Count);  
            int Value = 0;  
            bool v = false;  
            v = natrSH.LastValue > natrSH[1];  
            Value = GetBarsSince(v, natrSH.Count);  
            surgeReset = Value > 5;  
              
  
            double natrSurgeH = 0;  
            natrSurgeH = natrSH[index] > natrSH[1] ? 1 : surgeReset ? 0 : natrSurgeH;  
  
            bool freqnatrSurge = IsCrossover(natrSurgeH, 0);  
            double simVal = 0;  
            simVal = freqnatrSurge ? natrSim[index] : simVal;  
  
            if (freqnatrSurge)  
                \_natrSHArray.Add(natrSH[index]);  
  
            double natrSHAvg = \_natrSHArray.Count > 0 ? GetAverage(\_natrSHArray) : double.NaN;  
  
            bool natrSurge = freqnatrSurge && tnatrSim[index] > 1;  
  
            double natrP = StopLossTakeProfitTypeParameter == StopLossTakeProfitType.ATRSurgeAverage ? natrSHAvg \* ATRProfitMultiplier : natr[index] \* ATRProfitMultiplier;  
            double natrL = StopLossTakeProfitTypeParameter == StopLossTakeProfitType.ATRSurgeAverage ? natrSHAvg \* ATRLossMultiplier : natr[index] \* ATRLossMultiplier;  
  
            \_psnatr = natrSurge ? natrP : \_psnatr;  
            \_lsnatr = natrSurge ? natrL : \_lsnatr;  
  
            // The MA  
              
            //Result[index] = ((Bars.OpenPrices[index] + Bars.HighPrices[index] + Bars.LowPrices[index] + Bars.ClosePrices[index]) / 4.0);  
            double theMA = Indicators.SimpleMovingAverage(Bars.MedianPrices, MALength).Result.LastValue;  
            double theMAPlus = theMA + \_psnatr;  
            double theMANeg = theMA - \_psnatr;  
            double theplusstop = theMA + \_lsnatr;  
            double thenegstop = theMA - \_lsnatr;  
  
            MovingAverage[index] = theMA;  
            LongTakeProfit[index] = theMAPlus;  
            ShortTakeProfit[index] = theMANeg;  
            ShortStopLoss[index] = theplusstop;  
            LongStopLoss[index] = thenegstop;  
  
             
            /\*  
            Chart.DrawText("MovingAverage", Bars.LastBar.OpenTime, theMA, Bars.LastBar.OpenTime.AddMinutes(1), theMA, Color.Blue);  
            Chart.DrawLine("LongTakeProfit", Bars.LastBar.OpenTime, theMAPlus, Bars.LastBar.OpenTime.AddMinutes(1), theMAPlus, Color.Lime);  
            Chart.DrawLine("ShortTakeProfit", Bars.LastBar.OpenTime, theMANeg, Bars.LastBar.OpenTime.AddMinutes(1), theMANeg, Color.Maroon);  
            Chart.DrawLine("ShortStopLoss", Bars.LastBar.OpenTime, theplusstop, Bars.LastBar.OpenTime.AddMinutes(1), theplusstop, Color.Fuchsia);  
            Chart.DrawLine("LongStopLoss", Bars.LastBar.OpenTime, thenegstop, Bars.LastBar.OpenTime.AddMinutes(1), thenegstop, Color.Orange);  
            \*/  
  
            bool natrSDump = natrSurge && Bars.LastBar.Open > Bars.LastBar.Close;  
  
            // Table  
            string tablePosition = "BottomRight";  
            switch (tablePosition)  
            {  
                case "TopRight":  
                    tablePosition = "TopRight";  
                    break;  
                case "TopLeft":  
                    tablePosition = "TopLeft";  
                    break;  
                case "BottomLeft":  
                    tablePosition = "BottomLeft";  
                    break;  
                case "BottomRight":  
                    tablePosition = "BottomRight";  
                    break;  
            }  
  
            double tnatr = 100 \* Indicators.AverageTrueRange(ATRLength,MovingAverageType.Simple).Result.LastValue / Bars.LastBar.Close;  
            double tnatrSH = GetHighest(tnatrSim[index], ATRLookbackPeriod, index);  
  
            if (freqnatrSurge)  
                \_tnatrSHArray.Add(tnatrSH);  
  
            double tnatrSHAvg = \_tnatrSHArray.Count > 0 ? GetAverage(\_tnatrSHArray) : double.NaN;  
  
            double tnatrP = StopLossTakeProfitTypeParameter == StopLossTakeProfitType.ATRSurgeAverage ? tnatrSHAvg \* ATRProfitMultiplier : tnatr \* ATRProfitMultiplier;  
            double tnatrL = StopLossTakeProfitTypeParameter == StopLossTakeProfitType.ATRSurgeAverage ? tnatrSHAvg \* ATRLossMultiplier : tnatr \* ATRLossMultiplier;  
  
            \_prftpcnt = natrSurge ? tnatrP : \_prftpcnt;  
            \_losspcnt = natrSurge ? tnatrL : \_losspcnt;  
  
            double risk = 0.3;  
            double fund = (risk / \_losspcnt) \* 99.85;  
            double profit = fund \* (1 + (\_prftpcnt / 100)) - fund;  
  
            Color lCol = Color.FromArgb(60, Color.Red);  
  
            // Draw table  
            DrawTable("Fund", "$" + Math.Round(fund, 2), Color.Black, Color.White);  
            DrawTable("Profit", \_prftpcnt.ToString("P2") + "/" + "$" + Math.Round(profit, 2), Color.Black, Color.White);  
            DrawTable("ATR", natr.ToString(), Color.Black, Color.White);  
            DrawTable("Loss", \_losspcnt.ToString("P2") + "/" + "$" + Math.Round(risk, 2), lCol, Color.White);  
        }  
  
        private double GetHighest(double value, int period, int index)  
        {  
            double highest = double.MinValue;  
            for (int i = 0; i < period; i++)  
            {  
                if (index - i >= 0 && MarketSeries.Close[index - i] > highest)  
                    highest = MarketSeries.Close[index - i];  
            }  
            return highest;  
        }  
  
        private int GetBarsSince(bool condition, int index)  
        {  
            int count = 0;  
            for (int i = 0; i < index; i++)  
            {  
                if (condition)  
                    break;  
                count++;  
            }  
            return count;  
        }  
  
        private bool IsCrossover(double current, double previous)  
        {  
            return current > 0 && previous <= 0;  
        }  
  
        private double GetAverage(List<double> values)  
        {  
            if (values.Count == 0) return double.NaN;  
            double sum = 0;  
            foreach (double value in values)  
                sum += value;  
            return sum / values.Count;  
        }  
  
        private void DrawTable(string title, string value, Color bgColor, Color textColor)  
        {  
            Chart.DrawText(title, title + "\n" + value, Bars.LastBar.OpenTime, Bars.LastBar.Close, textColor);  
        }  
  
        public enum StopLossTakeProfitType  
        {  
            ATR,  
            ATRSurgeAverage  
        }  
    }  
}