这日闲来无事，逛网游络。忽到一处，有《缠论》一书映入眼帘，被其“市场哲学的数学原理”的醒目副标触动视欲，于是信手翻来，倒也随处可见“深入浅出”，直至日薄西山方知爱不释手。

如此好书岂能让它停于理论！

看官，您可曾听说过：10 多年前的亿安科技案，名动一时。亿安科技是中国首只股价突破百元的股票，也是首只跌幅达到百元的股票。它编织了许多暴富的神话，但也带来了更多的噩梦。和您讲这个故事，是要告诉您：《缠论》就是亿安科技操盘手李彪的杰作。

我的座右铭是：到最风险的领域去做最稳健的事业。

因此，我想把《缠论》理论付诸实施，把缠学理论转化为普通老百姓都可以引用的技术指标和自动交易软件。

如果您有足够的耐心，您将看到我把《缠论》一直转换到自动交易程序出来为止。如果说做市商的行当是金融寡头们的摇钱树，那么稳健的 EA 将是普通老百姓的聚宝盆，这就是我的理想！

撰稿人：

陆锡林 2011 年 9 月 17 日于浙江金华。

《缠论》的前言

关于“缠中说禅”股市技术理论

——市场哲学的数学原理

一、为什么叫“缠中说禅”？

1、以股市为基础。缠者，价格重叠区间也，买卖双方阵地战之区域也；禅者，破解之道也。以阵地战为中心，比较前后两段之力度大小，大者，留之，小者，去之。

2、以现实存在为基础。缠者，人性之纠结，贪嗔疾慢疑也；禅者，觉悟、超脱者也。以禅破缠，上善若水，尤如空筒，随波而走，方入空门。

3、缠中说禅的哲学路线安排。由股市之解决之道，至论语之入世之道，至佛学之大至深大圆满境界。以静坐、心经、佛号，引入大超脱之路。然“理则顿悟，乘悟并销；事须渐除，依次第进”，有缘者得之，无缘者失之，而

得并未得，失并未失，一颗明珠，总有粉碎虚空，照破山河之日。

二、“缠中说禅”股市技术理论成立的前提

两个前提：价格充分有效和市场里的非完全绝对趋同交易。

三、“缠中说禅”股市技术简解

1、以走势中枢为中间点的力度比较，尤如拔河，力大者，持有原仓位，力小者，反向操作。

2、把走势全部同级别分解，关注新的走势之形成，以前一走势段为中间点与再前一走势段比大小，大者，留之，小者，去之。

3、进行多重赋格性的同级别分解操作，尤如行船、尤如开车，以不同档位适应不同情况，则可一路欣赏风景矣。

4、其至高，则眼中有股，心中无股，当下于五浊纷缠之股市得大自在，亦于五浊纷缠之现世得大自在，即为“缠中说禅”。

四、《缠论》的本质

分为两个部分：

1.形态学。走势中枢、走势类型、笔、线段之类的东西。

2.动力学。任何涉及背驰的、走势中枢、走势的能量结构之类的东西。

两者的结合。

注：截止目前尚没有任何涉及成交量的分析或者说明，或许这也正是体现了价格包容一切市场信息的原则。

五、学习《缠论》的线路图

分型-笔-线段-走势中枢-走势

趋势-背驰-区间套-转折及其力度

第一课　分型

从本课开始，作者试图将《缠论》的理论应用到外汇市场上去。本课程所涉及到的外汇交易平台是 MT4 平台，所涉编程语言是 MetaQuotes Language 4 (MQL4)。

第一课，讲解缠论分型指标：BoundTheoryCandle.mq4

一、MQL4 中的分型指标

⑴系统函数

在 MQL4 中，系统技术指标函数 iFractals()可以很方便地构建由比尔威廉分型指标：

```
#property indicator_chart_window
#property    indicator_buffers 2
#property indicator_color1 Red
#property indicator_color2 Blue
//+------------------------------------------------------------------+
double UppBuffer[],LowBuffer[];
//+------------------------------------------------------------------+
int init() {
     IndicatorBuffers(2);
     SetIndexBuffer(0,UppBuffer);
     SetIndexBuffer(1,LowBuffer);
     SetIndexStyle(0,DRAW_ARROW);
     SetIndexStyle(1,DRAW_ARROW);
     SetIndexArrow(0,SYMBOL_ARROWDOWN);
     SetIndexArrow(1,SYMBOL_ARROWUP);
   return(0);
}
//+------------------------------------------------------------------+
int start() {
   int limit;
   int counted_bars = IndicatorCounted();
   if(counted_bars>0) counted_bars--;
   limit = Bars-counted_bars;
   for(int i=0; i<limit; i++) {
      UppBuffer[i] = iFractals(Symbol(),0,MODE_UPPER,i);
      LowBuffer[i] = iFractals(Symbol(),0,MODE_LOWER,i);
   }
   return(0);
}
//+------------------------------------------------------------------+
```

指标走势图如图 1 所示。

图 1

⑵运算代码

利用系统函数构建技术指标虽然简洁，但只能知其然，不能知其所以然。因此我们有必要了解比尔威廉分型指标的运算代码：

```
//+------------------------------------------------------------------+
//|                                                     Fractals.mq4 |
//|                      Copyright ? 2005, MetaQuotes Software Corp. |
//|                                       http://www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "Copyright ? 2005, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_color1 Red
#property indicator_color2 Blue
//---- buffers
double ExtUpFractalsBuffer[];
double ExtDownFractalsBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

```
//+------------------------------------------------------------------+
int init() {
//---- indicator buffers mapping
   SetIndexBuffer(0,ExtUpFractalsBuffer);
   SetIndexBuffer(1,ExtDownFractalsBuffer);
//---- drawing settings
   SetIndexStyle(0,DRAW_ARROW);
   SetIndexArrow(0,119);
   SetIndexStyle(1,DRAW_ARROW);
   SetIndexArrow(1,119);
//----
   SetIndexEmptyValue(0,0.0);
   SetIndexEmptyValue(1,0.0);
//---- name for DataWindow
   SetIndexLabel(0,"Fractal Up");
   SetIndexLabel(1,"Fractal Down");
//---- initialization done
   return(0);
}
//+------------------------------------------------------------------+
//| Custor indicator deinitialization function                       |
//+------------------------------------------------------------------+
int deinit() {
   return(0);
}
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int start() {
   bool bFound;
   double dCurrent;
   int i, nCountedBars=IndicatorCounted();
//---- last counted bar will be recounted
   if(nCountedBars<=2) i=Bars-nCountedBars-3;
   if(nCountedBars>2) {
      nCountedBars--;
      i = Bars-nCountedBars-1;
   }
//----Up and Down Fractals
   while(i>=2) {
      //----Fractals up
      bFound=false;
      dCurrent=High[i];
      if(dCurrent>High[i+1] && dCurrent>High[i+2] && dCurrent>High[i-1] && dCurrent>High[i-2]) {
```

```
      bFound=true;
      ExtUpFractalsBuffer[i]=dCurrent;
   }
   //----6 bars Fractal
   if(!bFound && (Bars-i-1)>=3) {
      if(dCurrent==High[i+1] && dCurrent>High[i+2] && dCurrent>High[i+3] && dCurrent>High[i-1] && dC
         bFound=true;
         ExtUpFractalsBuffer[i]=dCurrent;
      }
   }
   //----7 bars Fractal
   if(!bFound && (Bars-i-1)>=4) {
      if(dCurrent>=High[i+1] && dCurrent==High[i+2] && dCurrent>High[i+3] && dCurrent>High[i+4] &&
         bFound=true;
         ExtUpFractalsBuffer[i]=dCurrent;
      }
   }
   //----8 bars Fractal
   if(!bFound && (Bars-i-1)>=5) {
      if(dCurrent>=High[i+1] && dCurrent==High[i+2] && dCurrent==High[i+3] && dCurrent>High[i+4] &&
         bFound=true;
         ExtUpFractalsBuffer[i]=dCurrent;
      }
   }
   //----9 bars Fractal
   if(!bFound && (Bars-i-1)>=6) {
      if(dCurrent>=High[i+1] && dCurrent==High[i+2] && dCurrent>=High[i+3] && dCurrent==High[i+4] &
         bFound=true;
         ExtUpFractalsBuffer[i]=dCurrent;
      }
   }
   //----Fractals down
   bFound=false;
   dCurrent=Low[i];
   if(dCurrent<Low[i+1] && dCurrent<Low[i+2] && dCurrent<Low[i-1] && dCurrent<Low[i-2]) {
      bFound=true;
      ExtDownFractalsBuffer[i]=dCurrent;
   }
   //----6 bars Fractal
   if(!bFound && (Bars-i-1)>=3) {
      if(dCurrent==Low[i+1] && dCurrent<Low[i+2] && dCurrent<Low[i+3] && dCurrent<Low[i-1] && dCu
         bFound=true;
         ExtDownFractalsBuffer[i]=dCurrent;
      }
```

```
      }
      //----7 bars Fractal
      if(!bFound && (Bars-i-1)>=4) {
         if(dCurrent<=Low[i+1] && dCurrent==Low[i+2] && dCurrent<Low[i+3] && dCurrent<Low[i+4] && d
            bFound=true;
            ExtDownFractalsBuffer[i]=dCurrent;
         }
      }
      //----8 bars Fractal
      if(!bFound && (Bars-i-1)>=5) {
         if(dCurrent<=Low[i+1] && dCurrent==Low[i+2] && dCurrent==Low[i+3] && dCurrent<Low[i+4] && 
            bFound=true;
            ExtDownFractalsBuffer[i]=dCurrent;
         }
      }
      //----9 bars Fractal
      if(!bFound && (Bars-i-1)>=6) {
         if(dCurrent<=Low[i+1] && dCurrent==Low[i+2] && dCurrent<=Low[i+3] && dCurrent==Low[i+4] &&
            bFound=true;
            ExtDownFractalsBuffer[i]=dCurrent;
         }
      }
      i--;
   }
//----
   return(0);
}
//+------------------------------------------------------------------+
```

利用运算代码所构建的技术指标，其走势图与系统函数构建的是一致的：

图 2

二、对运算代码的解剖

现在我们来讨论图 2 中 A 点指标是怎么画上去的。计算威廉分型指标至少应有 5 柱 K 线，中间一柱的最高价为最高时画 Up 线，即图中红点；中间一柱的最低价为最低时画 Dn 线，即图中兰点。因此在图 3 中，汇价没有走到 B 点时，交易图上是没有 A 点指标的。

图 3

只有当汇价运行至 B 点（即新的一柱 K 线出现时），A 点才满足 5 柱运算法

则，于是画面上才会在 A 点添加上指标红点，这就是所谓的未来函数效应，即以后续 K 线来画此前的指标线。

更甚者，在威廉分型指标的运算代码中，我们可以看到，它分别对 5 柱、6 柱、7 柱、8 柱和 9 柱 K 线进行判断后再作出指标线应当画在哪柱 K 线上的，以致于它有可能产生指标线移动的现象。

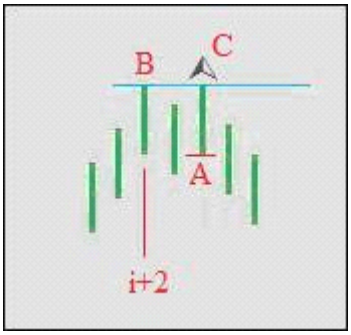

图 4

在图 4 中，当汇价运行至图中 A 点时，B 点满足 5 柱运算法则，且应在上添加指标图点，之后若 K 线形成如图形态时，指标图点应在 C 点的位置，由于有6 柱、7 柱、8 柱和 9 柱 K 线判断代码的存在，B 点的指标图点不见了，于是就形成了指标图点从 B 点移动到 C 点的现象。这在自动交易中就更加有害了。

因此，比尔威廉分型指标是一个有着严重未来函数效应的技术指标。

三、缠论中的分型

在缠学中，分型是指 K 线的形态关系，它的基本内容包括：

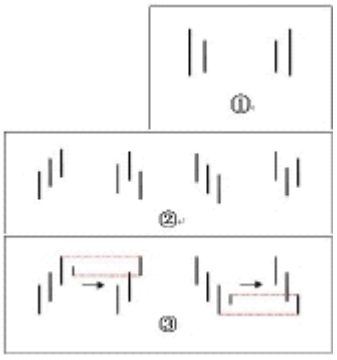

图 5

⑴基本概念

K线包含关系：指一K线的高低点全在另一K线的范围里(如图5中①)。

非包含关系的三相邻K线完全分类：分为四类——上升K线、顶分型、下降K线、底分型。(如图5中②)

K线包含关系的处理：在向上时，把两K线的最高点当高点，而两K线低点中的较高者当成低点，这样就把两K线合并成一新的K线；反之，当向下时，把

两K线的最低点当低点，而两K线高点中的较低者当成高点，这样就把两K线合并成一新的K线。(如图5中③)

⑵概念要点

K线合并方向：假设，第n根K线满足第n根与第n+1根的包含关系，而第n根与第n-1根不是包含关系，那么，如果第n根K线的高点大于第n-1根K线的高点，则称第n-1、n、n+1根K线是向上的；如果第n根K线的低点小于第n-1根K线的低点，则称第n-1、n、n+1根K线是向下的。

K 线包含关系的顺序原则：先用第 1、2 根 K 线的包含关系确认新的 K 线，然后用新的 K 线去和第 3 根比，如果有包含关系，继续用包含关系的法则结合成新的 K 线；如果没有，就按正常 K 线去处理。

⑶重点与难点

缠学分形的重点与难点在于 K 线包含关系的处理，网友们最典型理解是：

●比较第 i 根和第 i+1 根的包含关系，看第 i-1 根是上涨还是下跌，上涨就上包含，下跌就下包含，包含只有结合律没有传递律，不能说 K1 包含 K2，K2 包含 K3.,就说 K1 包含 K3，那是不对的。包含关系必须依次来处理。

●K 线包含关系的处理方向：可以自左向右顺序处理包含，我们简称左处理；也允许自右向左顺序处理包含，我们简称右处理。

●K 线包含关系的典型处理图示，如图 6 所示。

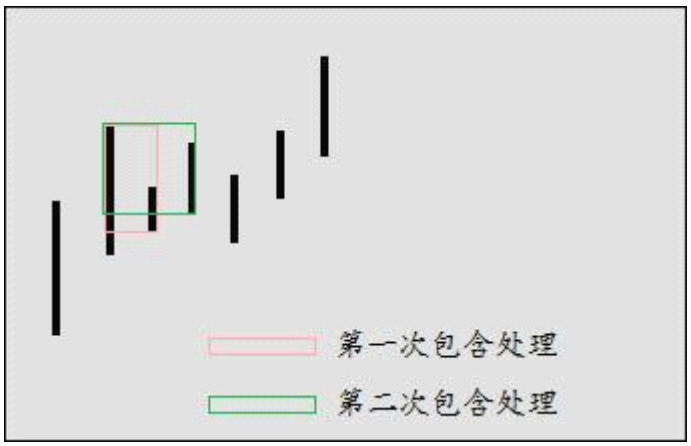

图 6

四、缠论分型指标

显而易见，任何指标的右处理都将导致未来函数效应的产生，因此我们在开发缠论分型指标中采用的是左处理规则。本人认为：采用左处理方案可以最大限度地降低缠论分型指标的未来函数效应，这也是我们研究缠论分型指标的动机所在。

作为编程技巧的交流，我们向读者提供二款缠论分型指标的代码范例（这些

代码范例读者只需复制下来即可成为完整的指标代码文件）。

⑴自左向右统一包含的缠论分型指标

●代码档案

```
//-- BoundTheoryCandle.mq4 --
#property copyright "Copyright 2011, Xilin Lu ( Zhejiang Lanxi )"
#property link      "http://www.WanYiDaForex.com    E-mail: TradingLovers@163.com"
//----
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_color1 Blue
#property indicator_color2 Blue
#property indicator_color3 Red
#property indicator_color4 Red
//----
double Up1[], Up2[], Dn1[], Dn2[];
//+------------------------------------------------------------------+
int init() {
//----
   SetIndexStyle(0,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(0, Up1);
   SetIndexStyle(1,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(1, Up2);
   SetIndexStyle(2,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(2, Dn1);
   SetIndexStyle(3,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(3, Dn2);
//----
   SetIndexBuffer(0,Up1);
   SetIndexBuffer(1,Up2);
   SetIndexBuffer(2,Dn1);
   SetIndexBuffer(3,Dn2);
//----
   return(0);
}
//+------------------------------------------------------------------+
int deinit() {
   return(0);
}
//+------------------------------------------------------------------+
int start() {
   double ThisHigh, ThisLow, NextHigh, NextLow, BeforeHigh, BeforeLow;
//----
   int j, k, i = IndicatorCounted();
```

```
   if( i<0 ) return(-1);
   if( i>0 ) i--;
   i = Bars - i;
//----
   j = i;
   while( j>=0 ) {
      Up1[j] = 0.0;
      Up2[j] = 0.0;
      Dn1[j] = 0.0;
      Dn2[j] = 0.0;
      j--;
   }
//----
   while( i>0 ) {
      //--
      if( Up1[i+1] > 0.0 ) {
         BeforeHigh = Up1[i+1];
         BeforeLow = Up2[i+1];
      } else {
         if( Dn1[i+1] > 0.0 ) {
            BeforeHigh = Dn1[i+1];
            BeforeLow = Dn2[i+1];
         } else {
            BeforeHigh = High[i+1];
            BeforeLow = Low[i+1];
         }
      }
      //--
      ThisHigh = High[i];
      ThisLow = Low[i];
      NextHigh = High[i-1];
      NextLow = Low[i-1];
      j = 0;
      //--发生包含
      while( (ThisHigh>NextHigh && ThisLow<NextLow) || (ThisHigh<NextHigh &&
ThisLow>NextLow) ) {
         if( i<2 ) { break; }
         j++;
         //向上
         if( ThisHigh>BeforeHigh ) {
            ThisHigh = MathMax(ThisHigh, NextHigh);
            ThisLow = MathMax(ThisLow, NextLow);
            NextHigh = High[i-1-j];
            NextLow = Low[i-1-j];
```

```
            } else {
                //向下
                ThisHigh = MathMin(ThisHigh, NextHigh);
                ThisLow = MathMin(ThisLow, NextLow);
                NextHigh = High[i-1-j];
                NextLow = Low[i-1-j];
            }
        }
        //--
        if( j>0 ) {
            //向上
            if( ThisHigh>BeforeHigh ) {
                for( k=j; k>=0; k-- ) {
                    Up1[i-k] = ThisHigh;
                    Up2[i-k] = ThisLow;
                }
            } else {
                //向下
                for( k=j; k>=0; k-- ) {
                    Dn1[i-k] = ThisHigh;
                    Dn2[i-k] = ThisLow;
                }
            }
        } else {
            //向上
            if( ThisHigh>BeforeHigh ) {
                Up1[i] = ThisHigh;
                Up2[i] = ThisLow;
            } else {
                //向下
                Dn1[i] = ThisHigh;
                Dn2[i] = ThisLow;
            }
        }
        //--
        i -= j;
        i--;
    }
//----
    return(0);
}
//+------------------------------------------------------------------+
```

●指标走势图

图中画圈的就是左处理后的 K 线，本指标对 i=0 柱 K 线不作处理，指标走势图如图 7 所示。

图 7

⑵自左向右依次包含的缠论分型指标

●代码档案

```
//-- BoundTheoryCandle.mq4 --
#property copyright "Copyright 2011, Xilin Lu ( Zhejiang Lanxi )"
#property link      "http://www.WanYiDaForex.com    E-mail: TradingLovers@163.com"
//----
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_color1 Blue
#property indicator_color2 Blue
#property indicator_color3 Red
#property indicator_color4 Red
//----
double Up1[], Up2[], Dn1[], Dn2[];
//+------------------------------------------------------------------+
int init() {
//----
   SetIndexStyle(0,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(0, Up1);
   SetIndexStyle(1,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(1, Up2);
   SetIndexStyle(2,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(2, Dn1);
```

```
   SetIndexStyle(3,DRAW_HISTOGRAM, 0, 2);
   SetIndexBuffer(3, Dn2);
//----
   SetIndexBuffer(0,Up1);
   SetIndexBuffer(1,Up2);
   SetIndexBuffer(2,Dn1);
   SetIndexBuffer(3,Dn2);
//----
   return(0);
}
//+------------------------------------------------------------------+
int deinit() {
   return(0);
}
//+------------------------------------------------------------------+
int start() {
   double ThisHigh, ThisLow, NextHigh, NextLow, BeforeHigh, BeforeLow;
//----
   int LastDirection, i = IndicatorCounted();
   if( i<0 ) return(-1);
   if( i>0 ) i--;
   i = Bars - i;
//----
   LastDirection = i;
   while( LastDirection>=0 ) {
      Up1[LastDirection] = 0.0;
      Up2[LastDirection] = 0.0;
      Dn1[LastDirection] = 0.0;
      Dn2[LastDirection] = 0.0;
      LastDirection--;
   }
//----
   while( i>0 ) {
      //--
      if( Up1[i+1] > 0.0 ) {
         BeforeHigh = Up1[i+1];
         BeforeLow = Up2[i+1];
         LastDirection = 10;
      } else {
         if( Dn1[i+1] > 0.0 ) {
            BeforeHigh = Dn1[i+1];
            BeforeLow = Dn2[i+1];
            LastDirection = -10;
         } else {
```

```
            BeforeHigh = High[i+1];
            BeforeLow = Low[i+1];
         }
      }
      //--
      ThisHigh = High[i];
      ThisLow = Low[i];
      //--发生包含
      if( (ThisHigh>BeforeHigh && ThisLow<BeforeLow) || (ThisHigh<BeforeHigh &&
ThisLow>BeforeLow) ) {
         //向上
         if( LastDirection==10 ) {
            Up1[i] = MathMax(ThisHigh, BeforeHigh);
            Up2[i] = MathMax(ThisLow, BeforeLow);
            Up1[i+1] = Up1[i];
            Up2[i+1] = Up2[i];
         } else {
            //向下
            Dn1[i] = MathMin(ThisHigh, BeforeHigh);
            Dn2[i] = MathMin(ThisLow, BeforeLow);
            Dn1[i+1] = Dn1[i];
            Dn2[i+1] = Dn2[i];
         }
      //--不包含
      } else {
         //向上
         if( ThisHigh>BeforeHigh ) {
            Up1[i] = ThisHigh;
            Up2[i] = ThisLow;
         } else {
            if( ThisHigh==BeforeHigh ) {
               //向上
               if( ThisLow>BeforeLow ) {
                  Up1[i] = ThisHigh;
                  Up2[i] = ThisLow;
               } else {
                  //向下
                  Dn1[i] = ThisHigh;
                  Dn2[i] = ThisLow;
               }
            } else {
               //向下
               Dn1[i] = ThisHigh;
               Dn2[i] = ThisLow;
```

```
            }
         }
      }
      i--;
   }
//----
   return(0);
}
//+------------------------------------------------------------------+
```

●指标走势图（如图 8 所示）

图 8

“自左向右依次包含的缠论分型指标”取消了“自左向右统一包含的缠论分型指标”中有关一次性将若干个（如果有的话）包含合并处理的代码块，也就是它完全站在本柱的立场上来处理 K 线的包含，实现了真正意义上的“包含关系必须依次来处理”的原则，本人认为它更符合《缠论》原创作者的分型理论。

图中画圈的几处，是提醒读者与图 7 作比较，以进一步加深的指标运算的内在涵义。

以上就是本课的内容。或许有读者要问，缠论分型指标与比尔威廉分型指标的表现形式相距甚远，为什么要将二者并在一起呢？

下节课讲述缠论笔指标将为读者解开迷底。