

Table of Contents

AUTOMATA-2008

A simple cellular automata model for FX market forecasting	1
<i>Michael Duane Archer</i>	
1 Introduction	2
2 The Trend Machine — basic assumptions	2
3 Work and efforts to 1999 — TTM v1.x	8
4 Significant issues	12
5 Current efforts — TTM v2.x	13
6 Price-time frames or increments	15
7 TTM v2.x — The trade-off	15
8 Explorations	15
9 Discussion	16

A simple cellular automata model for FX market forecasting

Michael Duane Archer

Molokai Technology Partners, USA

Email: Duane@FXpraxis.com

Home page: www.fxpraxis.com

Abstract. Since 1991 I have been to apply cellular automata (CA) to the problem of time series analysis generally and to forecasting Foreign Exchange (FOREX) prices, specifically [1]. Based on a number of back-tests I have run, I now believe the answer to the question, “Is the market a Computer?” to be ‘Yes!’ This paper provides a very general discussion of my basic research using CA concepts for market price forecasting using what I call Trend Machine (TTM); a simplistic but perhaps effective conversion of 1D CAs to market forecasting. Specific algorithms and parameters are proprietary at this time. The paper is limited to making the conceptual connection between CA and markets and to an historical narrative of my efforts. A Simple Cellular Automata Model for FX Market Forecasting is written for a wide audience of CA theoreticians and market practitioners. I am neither a computer scientist nor a mathematician; I am a trader, since 1973, and researcher with passable programming skills in Visual Basic, Smalltalk (Dolphin), Prolog (Visual) and my current platform of choice, C#.net. I have my own programming team; some of whom have been with me since the mid-1980s. I developed the successful Jonathan’s Wave expert system-neural network hybrid in the mid-1980s for commodity futures. The Trend Machine grew out of my conclusion AI methods — expert systems, neural networks, genetic algorithms — applied to the markets were fundamentally curve-fitting methods and functionally no different than conventional approaches such as stochastics, moving averages and oscillators. My interest in CA is purely practical — to find a consistent, risk-tolerable method of forecasting currency prices.

1 Introduction

My ‘seed’ idea for Trend Machines (TTM) came from reading, *Three Scientists and Their Gods* while doing consulting for the Winnipeg Grain Exchange in 1991 [2]. The book included an interview with and narrative about Ed Fredkin who brought CA out of the shadows in the 1970s. I had coincidentally been working on a method for describing market prices in binary notation, ‘Binary Charting’ for a program Bar Chart Stat Pro which analyzes bar chart sequences (‘1’ = ‘Up’ and ‘0’ = Down as the basic syntax.) Like peanut butter and chocolate in the old Reese’s commercial, the two ideas happily collided! The Eureka Moment came late one evening while my family was vacationing in Pensacola, Florida shortly after completion of the Winnipeg job. I woke my daughter, Brandy. We rushed to the local convenience store, purchased several packages of small wooden matches (much to the consternation of the store clerk) and experimented with a variety of a game described in *Further Mathematical Diversions* [3].

Some of the early TTM research, coding and back-testing was done with my Jonathan’s Wave programming team of Richard Rowe (St. John’s, Newfoundland, Canada), Antonio Riga (Toronto, Ontario, Canada) and James Bickford (Boulder, Colorado, USA). I lost contact with both Mr. Rowe and Mr. Riga in the mid-1990s; Mr. Bickford passed away in 2007. Curiously it was Mr. Rowe who very early on anticipated a problem I was not to acknowledge until much later — that my ‘perfect world’ TTM would run aground because of the Traveling Salesman P-NP Problem. My attempts to solve this issue cost me considerable time and effort but I did discover some interesting thoughts on P-NP which are beyond the scope of this paper.

I believe (after 17 years) I am very close to a significant breakthrough in algorithmic trading — but only the markets can, and will, tell the tale.

2 The Trend Machine — basic assumptions

Market forecasting methods have not been successful over the long term. Why? Techniques may be profitable for limited periods of time and/or specific environments, such as trending markets or trading markets. Success would be measured by profitability in diverse market conditions, for significant periods of time without dramatic ‘draw downs’ or deterioration of the capital base.

Two primary factors — directional movement and volatility — can reasonably describe the environment of any market. These primary market environments (MEs) plus two secondary environments — Thickness

and Rhythm and a tertiary environment, Shape, describe every market [4].

Directional Movement (DM) is the net change in prices from a given price-time unit, x , to the next price-time unit, $x + 1$. Volatility is the aggregate price change over that same period given a minimum fluctuation value. DM might be measured in a 90° arc from the horizontal chart axis representing zero to the vertical chart axis representing a real number of '4' or '10' or defining whatever precision is needed. V may be measured as a percentage of 100% and divided in kind as DM. '1-1' would represent a market with low directional movement and low volatility. '10-10' would represent a market with high directional movement and high volatility. ME effectively smoothes prices and has a wide variety of applications.

Most market forecasting techniques are linear. Proportionality is assumed between two or more factors. A change in factor 'X' causes a proportionate and constant change in factor 'Y'. This is an assumption about markets implied in all pattern recognition techniques.

But pattern recognition suffers some major theoretical flaws. Patterns (or input), no matter how sophisticated, identified as similar may lead to widely divergent output. If 'X' then 'Y' is not true of market data. Traders may recognize this from the 'sensitivity to initial conditions' of chaos theory. Nor is the converse true (If 'Y' then 'X'). Even if 'X' then 'Y' is true, the converse is a logical fallacy — modus ponens.

Internalizing the pattern search does not help. Expert systems, neural networks and even genetic algorithms suffer the same pattern recognition flaw as do moving averages, relative strength and stochastic tools. At best they are non-linear tools seeking linear information.

The author built the first expert system for market forecasting in the 1980s, Jonathan's Wave. In AI Expert magazine [5] Harvey Newquist described it as a hybrid expert system neural network. In futures, it generated a 48% annual return with a zero expectation of a 50% drawdown and negatively correlated with other trading systems at the time [6]. I wrote about Jonathan's Wave twice for Futures magazine [7].

At an AI conference in 1991 I suggested these difficulties to the attendees and elicited a very negative response [8].

Complexity theory studies processes resistant to linear analysis. Economies, ecologies, physical and life sciences are all areas with processes falling under the complex category. Information theory, game theory, encryption and coding and even corporate decision-making may be added. These processes seem to possess an internal organization not describable, much less predictable with linear methods of logic, mathematics or statistics. The markets appear also to be complex processes.

Some non-linear processes, such as the markets, generate time-series data as a by-product.

- Complex processes share several features; different types of complexity are defined by dominant features. In addition to resistance to linear methods and tools the most important are:
- Complex processes cannot be broken down and analyzed piece-by-piece. Nor can they be 'reverse engineered.' Study of the effect does not lead to knowledge of the cause. This is the finis for pattern recognition.
- Complex systems are sensitive to initial conditions. A very small change in initial conditions may lead to enormous changes later in the process. This is the defining characteristic of closed-end processes associated with chaos theory.
- Complex processes often manifest abrupt changes from one kind of behavior ('continuity') to another kind of behavior ('discontinuity'). Catastrophe theory studies such open-ended systems. Why does a sideways market 'suddenly' break out into a trending market?
- Recursive/Fractal: The behavior of complex systems is often played out at many different levels, all with great structural similarities. That is, each level is a macrocosm of the level below it and a microcosm of the level above. Another way to state this is that the processes are dimensionally recursive.
- Very simple system elements and rules create surprisingly diverse and complex behavior.
- Complex processes self-organize information. They appear to adapt by generating new, internalized rules. This self-organization can be likened to a computer program calling a sub-routine or a function with a new value. Complex systems behave as computational entities!

Consider the analogous characteristics of price markets: stocks, commodities, currencies, including the performance of investment fiduciaries:

- Linear methods have failed. The markets cannot be analyzed piece by piece. Nor does the study of the behavior (prices or information patterns) yield information about causes.
- The markets, as open-ended processes, are sensitive to minute input changes — the buy order that turns the market.
- Trading ranges suddenly erupt into trends, and visa versa.
- Two simple elements or rules create the complex tapestry of the markets: 'buy' and 'sell.'
- Hourly, daily, weekly and monthly bar charts of a stock or commodity exhibit many structural similarities. Without labeling it would be impossible to decide the time frame of a chart.

- I believe self-organization is the Major Market Force.

Discounting, the smoothing of spreads and the gradually lessening of the effectiveness of new market indicators are examples of the market utilizing self-organization as an adaptive ‘immune system.’ Discounting is a minor example of the market self-organizing input (buy and sell orders) into prices (output).

Is it possible to forecast the behavior of complex processes, especially those that generate time-series data as a by-product, especially stocks, currencies (FX) and commodities?

Since 1991 I have explored the investment markets, particularly currencies and commodities, as a complex process. Despite some initial excitement, chaos theory didn’t catch. Although markets do appear to be recursive, that function does not seem sufficient to generate forecasts of any real accuracy. I will discuss inter-dimensional recursiveness later in this paper. Markets are open-ended and chaos applies primarily to closed-end processes. Catastrophe is a much better try, but catastrophe does a better job describing than predicting.

Markets self-organize buy and sell orders into prices. It does this by the mechanism of an internal algorithm determining how much UP a buy order creates and how much DOWN a sell order creates. The ongoing flow of behavior (prices) is non-linear. This behavior is impossible to predict without some knowledge of the underlying algorithm for the market in question.

The markets may not be predictable using computers, per se, but may be very predictable as computers — as computational entities. Pattern recognition tools and linear methods study the market-computer’s screen dump. What needs to be studied is the program or algorithm generating the prices I see on the market-computer screen. I have been looking at the markets ‘inside-out’ (or ‘outside-in’) just as pre-Copernican astronomers took the Earth as the center of the solar system. The Market in the Machine!

The market is the pattern, and it is continually unfolding across the frontier of ‘now.’

The data by-product of any market encrypts useful information about the underlying process.

Many processes generate data as a by-product. The market is a complex process and prices are the primary by-product. Assumption: the by-product carries information about the underlying process. Please keep this in mind: prices are the by-product of an underlying process.

Algorithmic Forecasting (the term as first used by this author in 1992) attempts to duplicate the algorithm that creates or *describes* at least a part of the self-organizing behavior of a data producing complex process. Once this algorithm is found, the process can be modeled (on a regular computer!) and ran through the barrier of ‘now’ into the future.

Today, generally, the terms ‘algorithmic forecasting’ and ‘algorithmic trading’ reference any automated trading system or scheme in all market classes — securities, commodities and FOREX.

Viewing the market as a computer or computational entity solves two great mysteries of technical analysis:

1. How is information about prices transmitted from the past to the future?
2. Do past prices influence future behavior?

The markets behave as an algorithm with a feedback loop. Prices at T[ime]1 (input) are at least partially organized by a market’s specific algorithm. This yields new prices at T2 (output) that in turn becomes input for the next iteration of the algorithm, leading to prices (output) at T3. Buy and sell orders become the algorithmic parameters.

In sixteen years of research I’ve been able to draw two conclusions: a) the underlying algorithm is a major factor in prices, but not the only factor, b) the degree to which it is a factor ebbs and flows resulting in our conclusion that ‘The markets may be ‘busted’ from time to time for relatively short periods of time.’

Are yet-to-be-entered orders Acts of God, or can they be predicted?

Clearly I cannot predict new orders perfectly, in advance. But the market-as-computer tells us the algorithm of a market creates a sieve, or template, through which all orders must pass. In programming, a function may return a different value each time it is called, but the value is delimited by the parameters of the function.

It may be possible to find ‘basins of attraction’ or ‘areas of criticality’ — areas in price and time to which new orders are pulled or attracted. Limits in commodities are an artificially created basin of attraction. This is an assumption of TTM v2.x, please see the sections below.

Can the markets, or any other data-producing complex process be modeled for algorithmic forecasting to ‘tease out’ self-organizing, algorithmic behavior?

The model I developed uses cellular automata (automaton, singular) as the basis of a market-as-computer model. A typical cellular automa-

ton may be visualized as a grid of squares, or cells. Imagine an infinite chessboard, or chart grid.

The first cellular automata gedanken experiments of the famous John Von Neumann. He developed many details of CA after a suggestion of Stanislaw Ulam. My first exposure to CA was in 1971 when I attended a lecture by Professor Ulam at the University of Colorado in Boulder. (Chess Tradition in Man's Culture, hosted by Professor Eugene Salome; lecture by Professor Ulam — Chess and Mathematics.)

A two dimensional cellular automata (2D-CA), the 'infinite chessboard', is composed of five elements:

1. The cell universe
2. The individual cell(s)
3. The cell state
4. The cell neighborhood
5. The cell rules or algorithm.

The Cell Universe is a continuous arrangement of cells; the infinite chessboard.

A Cell is an individual unit, usually square or rectangular (but occasionally polygon tiled) within the cell universe; 'King 5' or 'Queen Bishop 15,000' on the infinite chessboard.

The Cell State refers to the condition of a cell. Usually a cell has only two states — ON or OFF (colored, uncolored). It is possible for cells to have multiple or even 'fuzzy' states.

The Cell Universe State may refer to either the current state of all the cells in a CA, or a collection of all prior universe states (generations) and the current state (generation) in some models.

The Cell Neighborhood is a selected group of cells surrounding a given cell. Cell neighborhoods may be either local or remote. Cells in a local neighborhood are physically adjacent to the cell. Remote cells are not adjacent. Two typical local neighborhoods are the Von Neumann neighborhood (side adjacent cells) and the Langton neighborhood (side adjacent and diagonal adjacent cells). Neighborhoods may also be state local or state remote.

Cell Rules or Algorithms are sets of instructions telling a given cell what state to assume in the next generation in response to the state condition of its neighbors in the current generation. Cell rule sets are usually simple, but may be large in number.

Neighborhood 'state conditions' may be very simple or enormously complex. Meta-level CA's may be used to define state conditions. I have spent more time exploring state conditions than any other factor in my lengthy and exhaustive research. In a typical CA only the state conditions of the neighbors in the current generation are used. But it is

possible for cell rules (and neighborhoods) to rely on several previous generations (i.e., state remote). In one experiment I used a CA algorithm to determine which neighborhoods to use a Trend Machine.

A CA is the sequential iteration of the cell rules resulting in changes to each successive generation of the cell universe.

As they progress from generation to generation CAs evolve into one of four types of behavior:

1. Death — all cells die and go to OFF
2. Stasis — a finite loop of cell universe states repeats. It may take several thousand iterations of the cell rules for a CA to display Stasis.
3. Random — cell behavior changes and fluctuates without rhyme or reason. Random CAs almost always evolve into Death or Stasis.
4. Life — the CA generates new and interesting behaviors. Cells are born and die; groups prosper and falter. Complexity increases and the CA exhibit self-organization.

It seems impossible to predict the type of CA from the cell rules without actually running the CA. Two extremely similar rule sets may lead to life and death. This, in fact, is the quietus for TTM v1.x, as below.

CAs are examples of computational entities. Cells states act as both input (data) and (program). Cellular automata are being used today to study many complex processes; especially those for which self-organization is a primary characteristic. The emerging science of A-life grew out of early CA experiments. CAs of Type 4 mimic the complete topology of life: birth, death, reproduction, adaptation and mobility.

Who is the market participant who has not had occasion to exclaim that the market seems often to have '*a life of its own?*'

3 Work and efforts to 1999 — TTM v1.x

A Trend Machine (TTM) uses a mapping calculus to convert a CA into a market model for the purpose of algorithmic forecasting. By modeling market price data as a CA it is easy to search for the algorithms that self-organize buy and sell orders into market prices.

The TTM calculus converts each element of a CA into a Trend Machine component:

CA	TREND MACHINE
cell state	price up / price down
cell neighborhood	price history
cell universe	the market
cell rules	algorithm engine/ca-base
previous state	previous price
current state	current price
next state	future price

In a simple 2D Trend Machine (TTM) each iteration of the CA moves forward in time (from left to right) and the cell universe is the collection of cell universe states. A basic 2D calculus would integrate rules forcing iterations into a single vertical column causing the CA to mimic a bar chart showing the range of high to low prices over a specified period of time. Another involves mapping formations such as the gliders found in Life to price chart formations.

TTM and CAs may be constructed in one, two, three or n-dimensions.

The Algorithm Engine (AE) is roughly equivalent to the inference engine in an expert system. The algorithm engine consists of three components:

1. The basic structural design for building algorithms
2. The variables that can be altered within the AE
3. The seed or initial cell universe state

Re-writing refers to the method used to convert market prices (or other time-series data) to TTM states and back again into market prices. Re-writing may be, and often is, inter-dimensional. Two dimensional CAs may be 'rewritten' to generate binary, one-dimensional CAs. CAs may also be cross-dimensional, as data output from 'sound' may be re-written to a 'visual' dimension. In this sense, algorithms are dimensionally recursive!

In a one dimensional TTM (1D-TTM) there is no grid. Cell states are simply '1' or '0'. In the basic model a '1' state would mean prices UP over the previous state and '0' would mean prices DOWN from the previous state. There is only one cell in each iteration of the cell universe. A sequence of cell states becomes a binary string. This effectively converts the market data 'by-product' into CA ready-to-use material!

I have worked primarily with 1D-CAs, but 2D, 3D and n-dimensional models (especially cross-dimensionality) have some fascinating possibilities.

It is theoretically possible to describe a higher dimensional TTM or CA in a one-dimensional scheme. Encoding the information in the

binary string does this. (It has been previously demonstrated that a one-dimensional CA is a Turing Machine and all CAs of dimensionality $1 + n$ can be described in 1 dimension, see *The Emperor's New Mind* by Roger Penrose.)

Indeed, since the limitations of generating uniqueness from a binary string are definitional-limited by the length of the seed, the most useful algorithms for *The Trend Machine* come from 2D-CAs which have been re-written to 1D, although this researcher has found a method to generate unique 1D CAs. It is, unfortunately, probably impossible to predict the output of these algorithms without actually running them. Unique binary strings are also more plentiful at 2D and 3D levels.

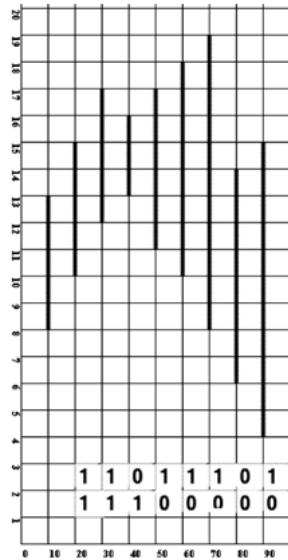


Fig. 1. Basic ordinal diagram. Each '1' and '0' may represent either an ordinal value (Up/Down) or a cardinal value (Up/Down with fixed, specific numeric value to each unit), in some re-writing schemes.

The binary string output of a 1D-TTM may be re-written (converted to/interpreted as market data) in many different ways. For example, see Fig. 1:

- Last (Close Only)
- Point and Figure
- Bar-Range / Horizontal Cluster

- Bar Range / Vertical Cluster
- Candlestick
- Pretzel
- Mercury¹

Current versions of TTM use an ordinal 'stacked semaphore' configuration of High, Low and Last prices:

```

HIGH  10011101010110110100110101000
CLOSE 10010000111101110110101010100
LOW   01101111010101011110100010101

```

Specific cell-states, neighborhood rules, functions (parameters) and transition rules are currently proprietary.

There are many different possible rewrite schemes — even for binary. While 'High, Low, Close' here means 'from the previous unit' CAs may and do take rules from other than the nearest (or, dominant) neighbor. Such distant neighbor seeding can significantly increase the number of possible unique algorithms. The string representations of '1' and '0' may also be either ordinal or cardinal values, simply 'UP' or 'UP 1.5 Units.' Converting to cardinal would probably increase effectiveness but cannot be done until the Caching Problem (see below) is solved.

Using my Market Environments methodology [8] has allowed my work to remain in the ordinal domain.

I have already mentioned higher dimensions can be encoded in binary strings. It is also possible to re-write directly between dimensional models.

A RAM or Representational Automata Machine is a CA modeled in n-dimensional state-space. A RAM might model all the factors of a 3D-CA (price, volume, open interest) and add, for example, volatility and an inter-market relationship. Like 3D-CAs RAMs allow for such specific factor mapping.

Four steps are involved in making a forecast using a Trend Machine:

1. Convert the market data (prices) to a binary format. Ordinal data conversion smoothes the data and leaves only directional movement and limited volatility information.
2. (a) Use the Algorithm Engine to find or build the algorithm most closely fitting the data.
(b) Search the CA-base for the closest match to the data.
3. Convert the binary CA string to prices using the appropriate format.

¹ Charting techniques developed by FXpraxis.

4. Concatenate the string to generate a forecast.

Forecast 2a is a perfect world. I have not been able to build a robust algorithm engine that would generate an algorithm from a data string output. Given the data can you systematically construct the algorithm (backward construction)? Probably not. Given the algorithm can you predict the output without actually running the algorithm itself (forward construction). Perhaps. All my TTM v1.x models have instead used a CA-base, 2B.

The current TTM CA-base consists of strings from 28 unique 'methods' with multiple permutations, parameter sets and seeds within each model. I am constantly on the hunt for new methods in 1, 2, 3 or n dimensions or in cross dimensional space.

The hunt for algorithms leading to unique strings is fascinating — and a bit addictive. I have at least managed to classify algorithm types and have a general sense of what will definitely not work — lead to uniqueness. I seek a method for determining what definitely will work.

An example of an extremely simple algorithm: Calculate π to x precision. Convert the odd numbers to '0' and the even numbers to '1.' For example: 3.14159265358979323846 \rightarrow 3.01000110001000010111

4 Significant issues

Back-testing is used to find many secondary parameters, such as how long a CA-Base string is needed to generate a forecast — and how long is the forecast meaningful.

The most significant issue has been the failure to find a general purpose Algorithm Engine (AE) that would take a binary string and find appropriate algorithms to generate that string and forecast-concatenations thereof. It appears neither a forward-chaining AE (predicting a string from an algorithm without running it) and a backward-chaining AE (mechanically finding an algorithm to match a string) are possible. But, on the side, I continue to toy with the idea simply because of its enormous attractiveness.

In 1999 a friend and fellow FX trader, Sylvester Torini, suggested a 'CA-Base.' The idea, quite simply — develop a database of binary CA strings. At specific time increments (5-minute, 1-hour, etc) the program would convert market data to the appropriate binary template and search the CA-Base for a perfect match. For example, a string of n_1-n_{500} where in $n_{501}-n_X$ would be reconverted to price format and used as the forecast.

It quickly became apparent a CA-Base of any value would be HUGE. Because of the ‘sensitivity to initial condition premise’ it is only a perfect match that will do.

I saw immediately computer processing speed would not solve the problem. Given Moore’s Law I calculated it would be many years before the Base could be analyzed with even 5-minute price data! I began to investigate the Traveling Salesman, P-NP Problem. Though not identical, the two problems have remarkably similarities.

I spend two years looking for caching algorithms of many varieties; a method for using the free time between increments to pre-sort the CA-Base. Having worked with a group of Go programmers in the UK I spent some effort on Monte Carlo sampling but that floundered quickly for me. By early 2006, I was somewhat stymied and TTM v.1.x seemed to have run its course.

I spent more time on the P-NP Problem and considered the possible solutions types: A complete solution, an approximation solution and a special-case solution. Run out on the complete solution thread, knowing that an approximation solution was a poor fit for a CA model, I was left with the special case solution to work with and ponder.

5 Current efforts — TTM v2.x

It was at lunch with my lead programmer, the late Jim Bickford, in June of 2006 that I devised the strategy currently in place — and which shows substantial promise.

Jim labeled the idea ‘One-Dimensional Catastrophe Topologies’ (ODCT) — accurate, but something of an ontological and semantic stretch. The method is not dissimilar to the Bar Chart Stat Pro program (FXpraxis, 2002) which analyzes sequences of price bars of four types — Inside, Outside, Bear and Bull.

Instead of using the entire CA-Base I am extracting only strings which represent or lead to

1. A high percentage of ‘0s’ [downtrend],
2. A very high percentage of ‘1s’ [uptrends] ,
3. Nearly equal ‘0s’ and ‘1s’ [trading markets], and
4. some ‘special case’ string ‘formations’ that offer real-time trading opportunities.

Even this represents a rather largish Base, but I am constantly pruning and filtering to make it manageable and applicable to smaller and smaller time frames. In the quest to drop time-frame size, my most current work (February 2008) involves using and searching as a function of

Market Environments (ME) of directional movement (DP) and Volatility (V) instead of prices. This may, in turn, lead to a simplified CA-Base.

It is important to recognize profiting in a market does not require the forecasting of exact prices. Forecasting Directional Movement and Volatility is adequate to the task, even with a precision of 20%-25% against an arc of 90^0 on a price chart.

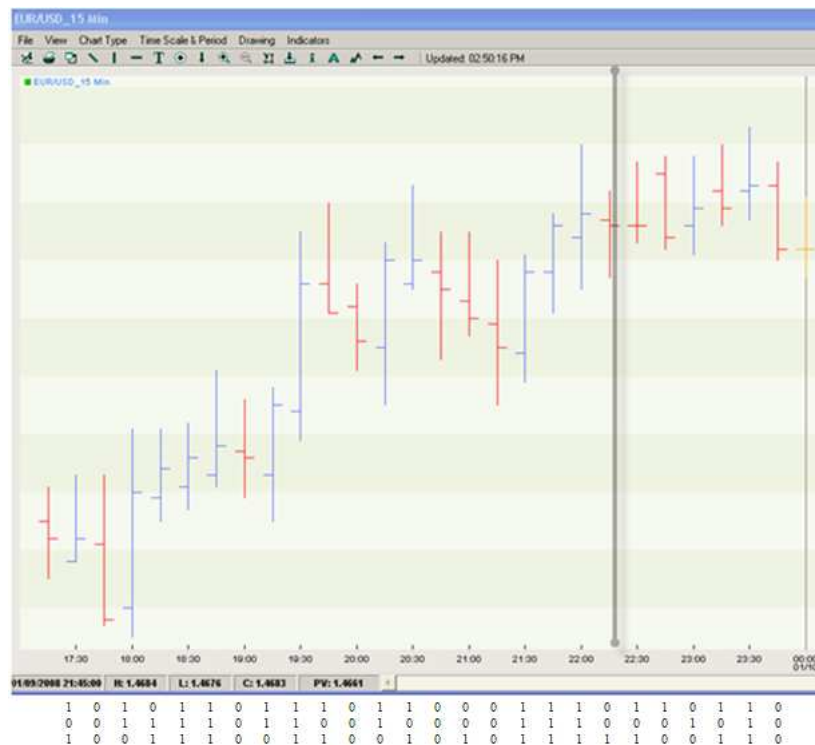


Fig. 2. Comparison of prices for the EUR/USD currency pair (Euro to US Dollar) against a match from the ODCT offline CA-Base. The bars to the left of the vertical line are the ODCT match. The bars to the right of the vertical line are the forecast by concatenating the ODCT match. The program was set to seek only perfect matches of a minimum of 20 pre-forecast units.

The most significant advantage of the current approach — the ODCTs may be searched offline and not real-time while the markets are

running. A still numerically significant but much smaller portion of the CA-Base needs to be searched real-time, see example in Fig. 2.

6 Price-time frames or increments

It became apparent several years ago there was a high correlation between length of time frame and (a) number of forecasts generated and (b) accuracy of forecasts. The shorter the time frame used, the more forecasts are made and the higher they rank statistically. I don't have enough data on time frames of under 15-minutes but it seems intuitive this is an exponential function, or nearly so. One point — it is clear the longer the time frame the more 'noise' there is in the data. The theory of high frequency trading may, indeed, be correct [10]!

Today I am able to use 15-minute data and am confident further work will, using allow increments below 1-minute by using (1) an ODCT CA-Base to analyze offline, (2) ME to smooth the incoming data and (3) a redesigned CA-Base to take full advantage of ODCT and ME.

7 TTM v2.x — The trade-off

In theory, an Algorithm Engine or complete CA-Base search will hit all available trading opportunities. But even a very limited scan of a CA-Base using a variety of caching methods (which will, of course, already filter out many trades) using 1-hour data cannot match the number of trades generated by ODCT with 15-minute data with a full scan.

I continue to search the (still expanding) CA-Base for ODCTs and am in the process of categorizing and further analyzing the latter. My next (and final?) step in the process is to integrate the components into a real-time algorithmic trading program. My programming team is investigating NinjaTrader, www.ninjatrader.com, for this application. Ninja is particularly suitable because its scripting language, NinjaScript is a C# subset.

8 Explorations

When one ponders a problem for 17 years many interesting side roads are discovered; if only one could walk them all!

I continue to dream of a general purpose algorithm engine.

I am exploring a CA-Base populated with binary genetic algorithm strings and using GA as a filter/caching methodology.

I can see more conventional forecasting methods such as expert systems utilizing the TTM methodology.

Other time-series models and problems may benefit from the TTM methodology. I have briefly investigated manufacturing process control as one possibility.

I have spend almost all of my time on a 1D TTM model. The possibilities for 2D, 3D and nD models is staggering. With limited time and money I decided early just ‘not to go there.’ Of particular interest to me is a 3D-TTM using Price, Volume and Open Interest in commodity futures. But many of my algorithms were found by using 2D CAs and collapsing them into 1D. Although the methods are testable it is doubtful they would pass full academic scrutiny.

I have not discussed my research or methods in generating unique binary strings and consider it to be proprietary to my work.

9 Discussion

I continue to believe a method of forecasting market prices with extreme accuracy and very low risk parameters is possible. If it is possible it is only so in a very limited set of market conditions and over very short periods of time. Thus, the quest to operate using very fine price-time increments of 1-minute or less.

If such a method is to be found it will be in the use of non-linear tools and methods. Of those I have concluded cellular automata most closely matches the ontology of markets. 35 years in the markets tell me linear methods are ‘closed’ and have little or no predictive value.

I am currently back-testing TTM v2.41 and anticipate trading at least a prototype model in 2009. I have attempted to attract interest for final research and trading funds from a number of major players in the FX space to no avail. There is a curious déjà vu for me to the reception Jonathan’s Wave received at the same stage in development, circa 1985. A company with a vision finally ‘took a chance’ and did very well because of the success of that program. Curiously, today, the large hedge funds are known as the world’s ultimate risk takers — but in fact, are very conservative, in-the-box thinkers. I consider their quantitative analysis tools, for example, to be deficient and have developing my own methods, based on my Market Environments methodology.

I hope to make TTM v2.x updates available on my website² and am happy to discuss my research (within reason) with others.

² www.fxpraxis.com

References

- [1] Archer, M. ECML '98 Workshop Notes. Application of Machine Learning and Data Mining in Finance. Is the Market a Computer? Technische Universitat Chemnitz, Chemnitz, Germany, April 1998.
- [2] Wright, R. Three Scientists and Their Gods. New York, TimesBooks, 1988.
- [3] Gardner, M. Further Mathematical Diversions. New York, Penguin Books, 1977.
- [4] Archer, M. Back-testing and Market Environments. Currency Codex, www.fxpraxis.com, 2006.
- [5] Newquist, H. Somewhere Over the Rainbow: Using AI to Get to Wall Street. AI Expert, July 1988.
- [6] Baratz, M.. Jonathan's Wave (Review). Boston, Managed Account Reports, 1989.
- [7] Archer, M. An Expert System for Trading Commodities. Conference on Artificial Intelligence. Chicago, University of Chicago, 1989.
- [8] Archer, M. The Artificial Intelligence Approach to Expert Trading Systems. Futures, July 1986. Archer, Michael. How Expert Systems Compare with Toolbox Programs. Futures, May 1987.
- [9] Archer, M. Market Environment Applications. Currency Codex, www.fxpraxis.com, 2004.
- [10] Jorda, O. et al. Modeling High-Frequency FX Data Dynamics. February, 2002.

Index

algorithmic trading, 6

market forecasting, 2

Trend Machine, 2