CAFÉ:
A Complex Adaptive Financial Environment

Ron Even
Courant Institute of Mathematical Sciences
251 Mercer St.
New York, NY 10012

Bud Mishra
Courant Institute of Mathematical Sciences
251 Mercer St.
New York, NY 10012

November 21, 1995

Abstract

This paper describes the Complex Adaptive Financial Environment (CAFÉ), a simulator for complex adaptive systems implemented in Java. CAFÉ's object oriented design makes it suitable for many types of simulation. We give an example of a market simulation where food is traded for gold and explore the effects of adding several kinds of speculators to the system. This paper describes the software structure and design of CAFÉ, building upon the object-oriented and distributed features of the Java programming language. Although the primary application for this system is in the computational finance area, we envision a much more general usage.

1 Introduction

Over the last couple of decades, there has been a merging of computational and financial fields, prompted by the emergence of program trading, sophisticated market-modeling, derivative markets, and more recently, E-commerce. Perhaps less well-known is the fact that new computational paradigms have been discovered that rely primarily on an efficient-market-like model among computing agents. Examples of these algorithms abound: traffic-flow control in a B-ISDN ATM switch network, load balancing in a parallel computing environment, software intelligent agents maintaining coherency of a data-base, etc. In view of these developments, it has become important to develop a wide variety of tools to study and experiment with realistic models of financial markets, to understand the nature of the computational models embedded in the distributed structure of the market (e.g., the Walrasian *tatonnement* resulting in market equilibrium) and finally, to use the resulting tools in creating effective distributed algorithms rapidly.

To this end, we have designed a new tool: Complex Adaptive Financial Environment (CAFÉ), a simulator for complex adaptive systems implemented in Java. Currently, CAFÉ's object-oriented design makes it suitable for many types of simulation; however, one can easily augment this system to create a full-fledged virtual economy or to adapt it to particular computational needs. Note that here, by "complex system" we mean a system with many agents interacting, possibly in complex ways. Some examples of complex systems are economies, biological systems, and chemical reactions. Thus, the possible usage of CAFÉ can easily go beyond the description given here.

2 Previous Work and Comparison with CAFÉ

There are several possible approaches to studying complex systems. Among them are historical analysis, human simulation, and analytical modeling. Each of these methods has its drawbacks.

- Historical analysis limits us to studying only those situations for which we can find historical data and is also subject to over-fitting.

- Human simulation is limited in the complexity we can hope to achieve and by the duration for which we can run it.
- Analytical modeling often requires drastic simplifying assumptions which greatly limit its applicability. For instance, it is common in mathematical finance to assume that in a multi-agent system all agents share a common utility function and are perfectly rational. This assumption is rather unrealistic and gives rise to several paradoxical situations.

CAFÉ addresses these problems by providing a great deal of flexibility in modeling a multi-agent system. CAFÉ shares its philosophy with many existing research efforts from three related areas: computational economy, simulation, and the study of bounded rationality.

- *Computational Economy*: Computational Economy addresses the problem of allocating distributed resources to competing users. The *Spawn* [6] system supports concurrent applications in a heterogeneous distributed system. It is organized as a market economy where the commodities are slices of CPU time on idle workstations, the buyers are applications needing CPU time, and the sellers are idle workstations with CPU time to spare. Each processor with idle cycles conducts an auction where applications compete for CPU time. Several experiments on *Spawn* have exhibited promising results regarding the effectiveness of the computational market.

  Wellman's *Walras* [5] system and the related work of Doyle (RECON [2] system) have found applications in distributed configuration design, wide-area networks with intelligent agents, and distributed multi-commodity flow problems. These economies work with producer and consumer agents, and supply and demand based on a power utility function. It has been shown that these economies reach equilibrium under fairly general assumptions. However, there are some thorny problems regarding the optimality of the system in equilibrium as well as situations where many of the assumptions may fail.

- *Market Models*: Steiglitz et. al. [4] describe a system to simulate a simple market with two commodities which they called gold and food. The economy is made up of producers who can each either farm (produce food) or mine (produce gold), each at a randomly assigned skill level. Two types of speculators exist in the market: *trend-based*, those that trade based on the second derivative of price and *adaptive*, those that trade by a method called adaptive expectations. Among the findings from the simulations were that speculators aided in speeding the convergence of price to its equilibrium point, trade volume was less volatile with speculators, and farmers and miners produced more in the presence of speculators, i.e., the overall efficiency was enhanced by the price stability.

  CAFÉ borrows many basic concepts from the preceding model. However, CAFÉ has a very rich set of speculators and as a result, exhibits much better performance in convergence to equilibrium, reduction in volatility and avoidance of volatility clustering. Furthermore, CAFÉ allows the economy to be modified much more easily.

- *Bounded Rationality*: Historically, agents in economic models are assumed to exhibit perfect rationality, i.e. their utility functions take into account all available data and make the best decision possible based on that data. In contrast, real world agents can cope only with bounded rationality, since the agents have only limited computational resources and often need to react to available information quickly (on-line). Recently, Brian Arthur has introduced a new model of agents with bounded rationality [1], based on some simple and intuitive heuristics. In essence, the main heuristic consists of building several simple models based on pattern matching, using the "best" of these hypotheses to make decisions, and constantly modifying hypotheses based on feedback.

  He shows the effectivity of these methods by considering a particular example (the so-called "Arthur's Santa Fe Bar Problem"). The example centers around a bar in Santa Fe which is "fun" if it is occupied by fewer than 60 people, but not if it is occupied by more than 60. The total population comtemplating attending the bar is 100. Each person wants to attend the bar if fewer than 60 people will be there but wants to stay home if more than 60 people will be there. Under a completely rational agent model, the

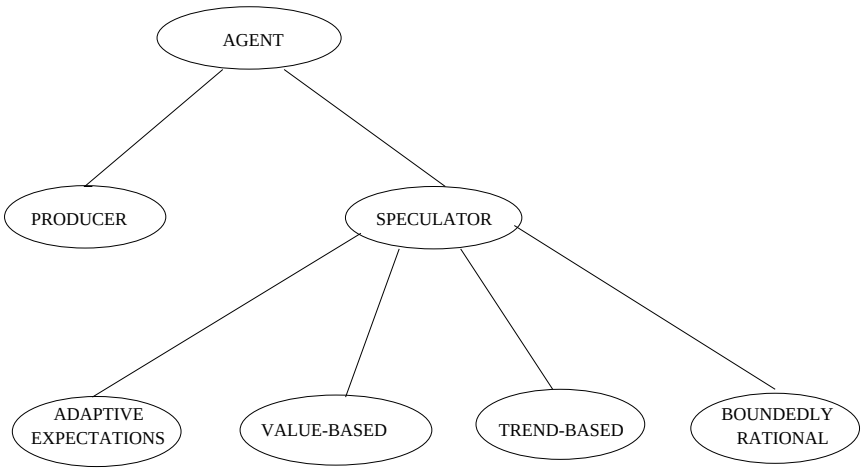

Figure 1: Agent Hierarchy in CAFÉ

agents fail to reach an equilibrium point as the very basic assumption of the Walrasian model does not hold here. Arthur introduces a model of bounded rationality whereby each person maintains a list of competing hypotheses and bases his decision on the hypothesis that appears to be working the best so far. This model displays rapid convergence to the equilibrium point as well as only small deviations from the optimal value. Although, this example may appear somewhat contrived, it captures many of the similar phenomena in communication networks and distributed computing systems.

The agents (mostly the speculators) in CAFÉ use similar bounded rationality to interact on-line and to obtain quick convergence to the equilibrium.

3 Structure of CAFÉ

CAFÉ is a simulator along the lines of Steiglitz's [4], but with an object-oriented approach which facilitates extensibility and allows simulations of different complex adaptive systems. Figure 1 shows the different types of classes derived from the Agent class.

Auction: The current auction works in rounds. In each round, orders are solicited from all the agents in the market. A market clearing price is established based on these orders. Then, as many orders as possible are closed at this price.

Agents: Agents are defined as objects that provide a function (method) that generates an order, a function that consumes, and a function that produces. This allows both Producers and Speculators to be derived from the Agent class.

Producers: Producers are derived from agents. They each have different randomly assigned skill levels in mining and farming. They all use the same utility function to generate their bids.

Speculators: There are four kinds of speculators in CAFÉ: Those that trade based on trends in prices, those that trade by adaptive expectations, those that trade based on fundamentals, and those that exhibit bounded rationality. At present, all speculators sell all their food when they decide to sell, and buy as much as possible when they decide to buy.

Trend-Based Speculators: Each trend-based speculator looks for trends that are at least c rounds[1] long, and places an order to buy or sell depending on whether the trend is up or down.

[1] A constant depending only on the speculator.

```
abstract public class Agent {
   public Holdings Networth;
   public Agent(){
      Networth=new Holdings();
   }
   abstract public Order GetBid(double price);
   public void Consume() {
      Networth.FoodDecr();
   }
   abstract public void Produce(double price);
   public Bar[] Report() {
      return null;
   }
}
```

Figure 2: Java code that defines the class Agent

Adaptive Expectation Speculators: Each adaptive expectation speculator keeps track of a predicted price. Each round, it updates its predicted price by taking a convex combination of its former predicted prices and the actual price. Based on this value, speculator either sells or buys, depending on whether the asset appears to be overpriced or underpriced.

Value-Based Speculators: The value-based speculator has a built in price that it considers the "actual" value. If the going price for the asset fluctuates outside of some margin about that price, the speculator buys or sells.

Bounded Rationality in CAFÉ: The last type of speculator is called a pool speculator, because it maintains a pool of predictor functions à la Arthur's boundedly rational agents. There is a large bag of functions from which each speculator chooses a fixed-size subset. Each speculator keeps track of how well each predictor in his "bag" is doing, using the best so far to decide whether or not to buy or sell in the upcoming auction. When a predictor becomes particularly bad, the speculator discards it and chooses a new one from the pool of contenders.

4 Java and CAFÉ

We chose to implement CAFÉ in Java because it provides security and World Wide Web access and facilitates object oriented design. In the near future, CAFÉ will be made available on the World Wide Web, and anyone with access to the Web will be able to add agents to the system. Java provides facilities for dynamically linking code across the Web without compromising the security of the hardware or software of the system running CAFÉ. Figure 2 gives a code fragment from CAFÉ which exemplifies the use of some of Java's object oriented features. The fact that the class is defined as abstract means that it is impossible to create an instance of the class—the class is defined for the purpose of deriving new classes (subclasses) from it. Inside the class definition are the variables and methods (functions) associated with the class. Abstract methods are not provided but must be provided by any non-abstract subclasses of Agent. Other methods are provided by the Agent class but may be overridden by any subclass. For example, the Report method returns an array of Bar objects. The default behavior, is to return a null pointer, but the Pool subclass, for example, overrides this method to return some useful information. The Agent class declares all the variables and methods that every type of agent must provide.

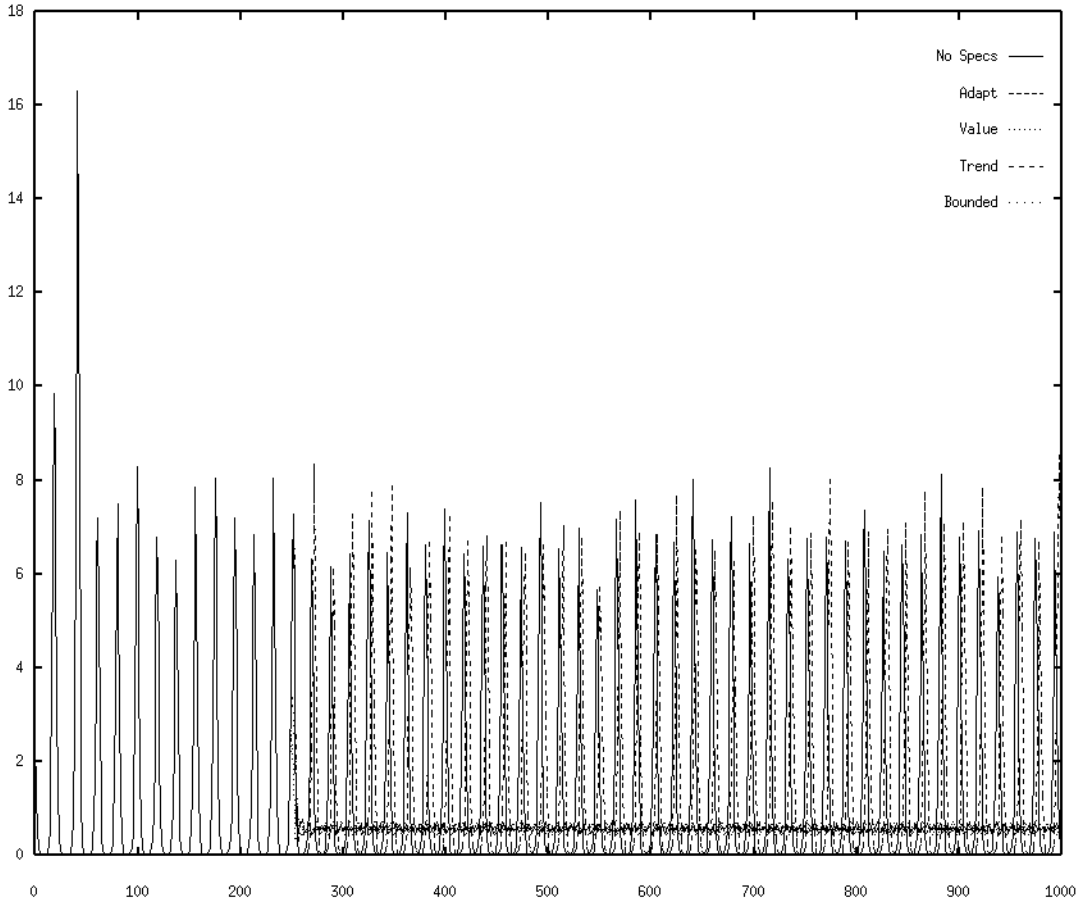

Figure 3: Price vs. Time with no speculators and with each type of speculator. Speculators were introduced at time 250.

5 Lessons from CAFÉ

With the ability to add four (trend based, value based, adaptive expectations based, and boundedly rational) different kinds of speculators into the CAFÉ environment, we have begun generating data, and are observing some encouraging results.

- *Individual Speculators*: With no speculators, price volatility was very high. Figure 3 shows a sample run that was typical of the behavior without speculation and with each of the different types of speculators. Three of the four classes of speculators brought the price near equilibrium very rapidly (in fewer than 10 rounds after their introduction). The trend-based speculators did not seem to be effective in this regard. This was most likely because the price volatility was so high that trends invariably reversed themselves before these speculators could profit from them. The three successful classes of speculators all had almost the same effect on the price. In fact, their graphs are so close as to be almost undistinguishable from each other in Figure 3. Whereas the volatility was over three times the mean price without speculators and with the trend speculators, with any of the other speculators, the volatility dropped to less than 2% of the mean price. The effect of the drop in volatility was quite dramatic. The producers earned considerably under reduced volatility. This was to be expected, since their decisions to farm or mine were more likely to be correct when the price was steady and predictable.

- *Multiple Speculators*: Combining different kinds of speculators did not have a dramatic effect on performance. As long as at least one type of successful speculator was present, the volatility remained low. The trend-based speculators did not have much of an effect when combined with others, since price still fluctuated enough to make them irrelevant. In fact, other speculators performed better in the presence of trend-based speculators, apparently winning their endowments.

- *Bounded Rationality*: In Brian Arthur's work, he observes that while the boundedly rational agents as a group operate at or near some equilibrium point, the individual agents are constantly updating

their predictor functions. We note a similar observation in our simulations (see Figure 3). This makes sense in that if all the agents settle on particular predictor functions, we would expect the price to fall into a pattern, making it easily predictable by one function, hence causing all agents to adopt the same function. This would undermine the equilibrium point. An interesting question raised by this observation is what kind of predictor functions would lead to unsatisfactory results? We chose our pool of predictor functions rather haphazardly. What are the characteristics of the pool that make it suitable for bringing the price to equilibrium? We have some preliminary results in this direction.
We altered the pool speculator class first to use only price history based functions, then to use only functions based on averages of past prices and finally, to use only functions that were not tied to price history (value-based functions). In the first two cases, preliminary results lead us to believe that the overall behavior of the system was not greatly affected. In the last case, however, while the volatility remains low, the mean price of food drops greatly. We still lack an explanation for this phenomenon.

- *Distributivity and the World Wide Web*: We are in the process of providing the ability to add agents running on remote processors. Eventually, CAFÉ will run continuously, allowing remote machines to add agents dynamically from anywhere on the World Wide Web. Since Java provides a facility for dynamic loading of new classes, users will be able to author new classes of agents and add them to the system. The resulting virtual economy will potentially have all the features of a real market. We note parenthetically that the underlying security provided by the Java language makes the system secure, which is an important issue if CAFÉ is to be used for E-commerce.

- *Future Applications*: The topic of using market ideas for network pricing schemes is being actively researched. We are currently adapting CAFÉ to simulate an ATM network. Once completed, this will allow us to explore different pricing strategies at the nodes and compare these pricing strategies to rate-based schemes. The World Wide Web is also stirring up interest in "software robots". These are intelligent software agents that are able to gather and process information for human consumption. Market methodology may provide insight into how to make these robots more efficient.

References

[1] W. Brian Arthur. "Inductive Reasoning and Bounded Rationality," in *Complexity in Economic Theory.*

[2] Jon Doyle. "A Reasoning Economy for Planning and Replanning," in *Technical Papers of the ARPA Planning Institute Workshop*, Tucson, AZ, 1994.

[3] Paul A. Samuelson and William D. Nordhaus. *Economics.* 12th Edition. McGraw-Hill, New York. 1985.

[4] Kenneth Steiglitz, Michael L. Honig, and Leonard M. Cohen. "A Computational Market Model Based on Individual Action," in *Market-Based Control: A Paradigm for Distributed Resource Allocation*, Scott Clearwater (ed.), World Scientific, Hong Kong.

[5] Michael P. Wellman. " A Market-Oriented Programming Environment and Its Application to Distributed Multi-Commodity Flow Problems," *Journal of Artificial Intelligence Research*, **1**:1–23, 1993.

[6] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart and W. Scott Stornetta. "Spawn: A Distributed Computational Economy," *IEEE Transactions on Software Engineering*, **18**(2):103–117, 1992.

[7] Wayne L. Winston. *Operations Research: Applications and Algorithms.* pp. 933–982. Duxbury Press, Boston. 1987.