A Linear Time Algorithm for Pricing European Sequential Barrier Options

Peng Gao **Ron van der Meyden**

School of Computer Science and Engineering, UNSW
and Formal Methods Program, National ICT Australia
{gaop, meyden}@cse.unsw.edu.au

Abstract

Financial derivatives are contracts concerning rights and obligations to engage in future transactions on some underlying financial instrument. A major concern in financial markets is to compute an expected value of such contracts as a basis for trading decisions. The Cox, Ross and Rubinstein (CRR) binomial tree model is a popular discrete approach to such computations, which requires time quadratic in the number of discrete temporal steps to contract termination. Lyuu has shown that barrier options can be valued with respect to the CRR model in linear time, using a combinatorial method. The paper develops a generalization of Lyuu's result, showing that a class of more complex options comprised of a sequence of barriers can be valued in linear time.

1 Introduction

Financial derivatives are contracts concerning rights and obligations to buy or sell certain underlying financial assets during some specified period. Option contracts are one simple kind of financial derivative. For example, a 3-month crude oil option contract states an agreed price for a crude oil transaction 3 months in the future of the date of issue. Such contracts can be traded, and the value of such a contract at a time prior to the expiry date depends on the current crude oil market price and expectations concerning price movements.

Although the history of financial derivatives can be dated back to 16^{th} century, a mathematical pricing methodology was not created until the early 1970s (Black & Scholes 1973), drawing on the theory of Brownian motion from physics. This "Black-Scholes" pricing theory enabled widespread adoption of derivative instruments, which have developed into a fundamental part of modern finance and microeconomics.

The Black-Scholes theory uses stochastic differential equations, so is difficult to compute. A discrete approximation method called the CRR Binomial Tree (or, more accurately, "Binomial Lattice") method was proposed by Cox, Ross and Rubinstein (Cox, Ross & Rubinstein 1979). This lattice method is extremely popular in option pricing as this model can handle the investor's nondeterministic choices during the life cycle and can be easily implemented in computer programs. A binomial lattice of depth n consists of a tree-like lattice structure of n^2 nodes, and the valuation of a contract on such a lattice is by a dynamic programming technique that uses a backward induction from the leaf nodes to the root to compute the expected payoff of the contract. Thus, the time and space complexity of this computation is $O(n^2)$.

Certain special classes of derivative contracts can be priced more efficiently with respect to the binomial tree model. *Barrier option* contracts are a generalization of basic options contracts that state that an option to buy or sell at a given price becomes valid provided the market price of the underlying asset reaches a particular "barrier" value. In *European* options the transaction must occur on the date of expiry of the option. (In American options it can occur on any date up to expiry.) Lyuu (Lyuu 1998) gives a linear time algorithm based on CRR binomial lattices for the valuation of European barrier options. Lyuu's algorithm uses a combinatorial "lattice path-counting" technique. In this paper, we show that Lyuu's technique can be generalized to yield a linear time valuation algorithm for a larger class of derivative contracts: *European sequential barrier options*, (Pfeffer 2001), which state that the option becomes valid after the market price has traversed a given sequence of barrier values.

For the proof of this result, we develop a notation for binomial lattice path properties that draws on ideas from *process logic*, a type of modal logic developed in computer science. We show that the problem of counting the number of paths satisfying certain properties expressed in this notation can be solved symbolically by means of logical transformations of the path counting expressions. We also give some experimental results to compare the performance of the resulting pricing algorithm with the standard backward induction method.

The structure of the paper is as follows. Section 2 gives some background on derivates and the binomial lattice method. In Section 3 we formally represent binomial lattices and options contracts using probabilistic automata. We define sequential barrier options in this model in Section 4. Section 5 defines our path notation and gives the symbolic path counting transformation rules. Section 6 gives the pricing algorithm based on these rules, and discusses its complexity. Correctness proofs for the pricing rules are given in Section 7. Section 8 gives some empirical performance results of an implementation of the method, compared to the backward induction method. We discuss possible future work in Section 9.

2 Background

In this section, we give a basic classification of option contracts and related value and position issues.

Copyright copyright 2007, Australian Computer Society, Inc. This paper appeared at Computing: The Australian Theory Symposium(CATS2007), Ballarat, Australia. Conferences in Research and Practice in Information Technology(CRPIT), Vol. 65. Joachim Gudmundsson and Barry Jay, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

2.1 Classification

In the over-the-counter derivative market, investors can agree on arbitrarily customized option contracts for their own specific risk and reward needs, but some basic parameters can be used to classify well standardized contract structures.

Call and Put Options A *call option* gives the holder the right to buy the underlying asset by a certain date for a certain price, while a *put option* gives the holder the right to sell the underlying asset by a certain date for a certain price. The price in the contract is known as the *exercise price* or *strike price*, the date in the contract is known as the *expiration date* or *maturity.*

European and American style *American options* can be exercised at any time up to the expiration date. *European options* can only be exercised on the expiration date itself. Most of the options that are traded on exchanges are American style. In the exchange-traded equity options market, one contract is usually an agreement to buy or sell 100 shares.

Option Positions There are two sides to every contract. On one side is the investor who has taken the long position (i.e., has bought the option). On the other side is the investor who has taken a short position (i.e., has sold or *written* the option). The writer of an option receives cash when the agreement comes into being, but has potential liabilities later. The writer's profit or loss is the reverse of that for the purchaser of the option. There are four types of option positions:

1. A long position in a call option.
2. A long position in a put option.
3. A short position in a call option.
4. A short position in a put option.

2.2 Contract Payoff and Profit

It is often useful to characterize European option positions in terms of the terminal value or payoff to the investor at maturity. The initial cost of the option is then not included in the calculation. If K is the strike price S_T is the final price of the underlying asset, the payoff from a long position in a European call option is

$$max(S_T - K, 0)$$

And the payoff to the holder of a short position in the European call option is:

$$-max(S_T - K, 0)$$

Similarly, the payoff to the holder of a long position in a European put option is

$$max(K - S_T, 0)$$

and the payoff from a short position in a European put option is

$$-max(K - S_T, 0)$$

Also notice that, the sum of the payoff from the holder of a long position of an option and the payoff from the holder of a short position in the same contract is always zero. This is called *put-call parity.*

2.3 Barrier Options

A barrier option is a path-dependent exotic option whose payoff is dependent on whether the underlying asset price breaks certain barrier(s) during (part of) the life time of the contract or not. Options with a single barrier can be classified into four categories:

1. up-and-in
2. up-and-out
3. down-and-in
4. down-and-out

Consider a portfolio which consists of one up-and-in barrier option and one up-and-out option with the identical underlying asset, barrier price, exercise price and expiration date. Then in any circumstance, there will be exactly one barrier option in the money and thus the portfolio replicates a vanilla option, without the barrier condition. This is called *in-out parity.*

The Black-Scholes pricing methodology assumes that the underlying asset price follows a geometric Brownian motion. With respect to this assumption, a closed form formula for a European style down-and-in call exists, due to (Merton 1994):

$$Se^{-q\tau}(H/S)^{2\lambda}N(x) - Xe^{-r\tau}(H/S)^{2\lambda-2}N(x - \rho\sqrt{\tau})$$

where

$$x = \frac{ln(H^2/(SX)) + (r - q + \rho^2/2)\tau}{\rho\sqrt{\tau}}$$

$$\lambda = \frac{r - q + \rho^2/2}{\rho^2}$$

and S is the current price of the underlying asset, H is the barrier with $S \geq H$, q is the stock's dividend yield, τ is the time to maturity and X is the exercise price. The value of a European up-and-in put is

$$Xe^{-r\tau}(H/S)^{2\lambda-2}N(-x+\rho\sqrt{\tau}-Se^{-q\tau}(H/S)^{2\lambda}N(-x))$$

for $S \leq H$. The down-and-out call and up-and-out call can be priced via the in-out parity.

Many derivative contracts do not admit closed form solutions, and a common technique for their valuation is based on a discrete approximation to the continuous dynamics known as a *binomial tree* or, more accurately, a *binomial lattice.* This is a grid-like structure of the form depicted in Figure 1. Here the horizontal dimension represents time and the vertical dimension represents the stock price. At each time step, the stock price either increases, with probability P_u, by a factor u, or decreases, with probability $P_d = 1 - P_u$, by a factor d. It is assumed that $ud = 1$, so that an upwards move followed by a downwards move reaches the same price level as a downwards move followed by an upwards move, creating the lattice structure.

Although a closed form formula for European-style barrier options exists, algorithms based on the lattice model are still important, since they provide a method for pricing more complicated options such as the American-style counterparts, for which analytical solutions are not known.

3 Definitions and Semantics

In this section we present an automaton theoretic framework within which we may define the class of

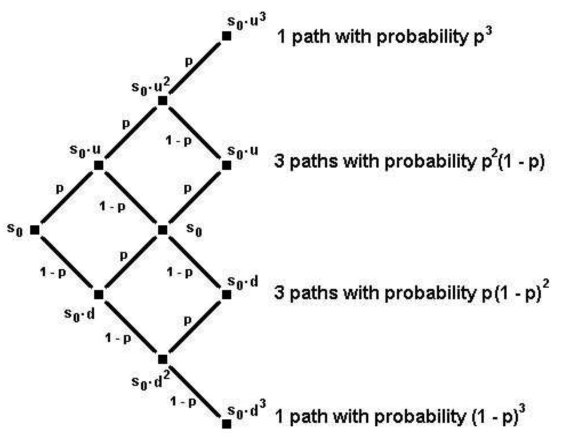
$s_0 \cdot u^3$
1 path with probability p^3
$s_0 \cdot u^2$
$s_0 \cdot u$
3 paths with probability $p^2(1 - p)$
s_0
$s_0 \cdot d$
3 paths with probability $p(1 - p)^2$
$s_0 \cdot d^2$
$s_0 \cdot d^3$
1 path with probability $(1 - p)^3$
p
1 - p

Figure 1: A binomial lattice

contracts we study in this paper, as well as more general contracts. We use a timed probabilistic transition system to model the market behavior of the underlying asset (the binomial lattice) and a standard finite state automaton to model the derivative contract. These two systems form a two-level hierarchy structure. The contract automaton is like a monitor: it reads in the output from the underlying market automaton and makes a state transition according to the definition of the contract. Thus the payoff of the contract is determined by both the state of the price automaton (price level and time) and the contract automaton.

We model the market in the underlying asset using binomial trees, captured formally using a probabilistic transition system. Formally, a *market automaton* is a tuple $\mathcal{M} = \langle M, m_0, \delta_m, \mathit{price} \rangle$, where

- $M = \mathbf{Z} \times \mathbf{N}$ is the set of states of the market,
- $m_0 = (0, 0)$ is the initial state,
- $\delta_m : M \times M \to [0, 1]$ is a transition probability function, such that for some probability $p \in (0, 1)$,

$$\delta_m((l,t),(l',t')) = \begin{cases} p & \text{if } (l',t') = (l+1,t+1) \\ 1-p & \text{if } (l',t') = (l-1,t+1) \\ 0 & \text{otherwise} \end{cases}$$

- $\mathit{price} : \mathbf{Z} \to \mathbf{R}$ is a *price function* which maps an integer market level to a real value market price.

The first component l of state $(l, t) \in M$ represents a price level within the binomial tree; the second component t represents a time. The actual price of the asset associated to a market level l is the value $\mathit{price}(l)$ specified by the price function. Generally, we take, $\mathit{price}(n) = Su^n$ where

$$u = e^{\sigma\sqrt{\delta t}},$$

(where σ is the volatility of the asset and δt is the time increment) which is a popular way to match the stock price volatility (Hull 2002).

The probability p is determined by a risk-neutral analysis. The basic idea for risk-neutral analysis is the assumption that in a risk-neutral world, all individuals are indifferent to risk (Hull 2002). Thus investors require no compensation for risk, and the expected return on all securities is the risk-free interest rate. In the binomial tree, after one time step δt, the expectation of the asset price S is $pSu + (1-p)Sd$, where $d = 1/u$. By risk-neutrality, this should be equal to the value obtained had the initial value S been invested at the risk-free interest rate r, viz. $Se^{r\delta t}$. We can now deduce that

$$p = \frac{e^{r \cdot \delta t} - d}{u - d}.$$

In fact, this is correct not just in a risk-neutral world but in the real world as well (Hull 2002).

A *run* of a market automaton $\mathcal{M}$ is a sequence $\rho = s_0 s_1 \ldots s_n$ of states of $\mathcal{M}$ such that for all $k = 0, \ldots, n-1$ we have $\delta(s_k, s_{k+1}) > 0$, i.e., the probability of a transition from s_k to s_{k+1} is nonzero. For $n \geq 0$, we write $\mathtt{Runs}_n(\mathcal{M})$ for the set of runs of the market automaton of length $n + 1$.

An *initialised run* is a run $s_0 \ldots s_n$ with s_0 equal to the initial state q_0 of the automaton. If ρ is a finite sequence, define $\mathit{fin}(\rho)$ to be its last element and $\mathit{init}(\rho)$ to be its first element. A state $s \in M$ is *reachable* if there exists an initialised run ρ with $\mathit{fin}(\rho) = s$. For a time $T \in \mathbf{N}$, we write $\mathit{reach}_T(\mathcal{M})$ for the set of reachable states of $\mathcal{M}$ with the second component equal to T. Clearly, not all states of the market automaton are reachable. For example, $\mathit{reach}_1(\mathcal{M}) = \{(1, 1), (-1, 1)\}$. Thus the state $(0, 1)$ is not reachable. More generally,

$$\mathit{reach}_T(\mathcal{M}) = \{(l, T) \mid -T \leq l \leq T,\ T + l \text{ even}\}.$$

We have defined the state space to be larger than the set of reachable states in order to facilitate later constructions in which we consider runs, commencing at states other than initial state, that lie outside the binomial tree.

For each $n \geq 1$, let $\mathtt{IRuns}_n(\mathcal{M})$ be the set of initialised runs of $\mathcal{M}$ of length n. The function $P : \mathtt{IRuns}_n(\mathcal{M}) \to [0, 1]$ defined by

$$P(s_0 s_1 \ldots s_{n-1}) = \Pi_{k=0}^{n-2}\ \delta(s_k, s_{k+1})$$

gives a probability distribution on $\mathtt{IRuns}_n(\mathcal{M})$.

We model financial contracts using deterministic automata that take states of a market automaton as input.[1] Formally, a *contract automaton* is a tuple $\mathcal{C} = \langle Q, q_0, \delta_c, F, \mathit{pay} \rangle$, where

- Q is a set of contract states,
- $q_0 \in Q$ is the *initial* contract state,
- $\delta_c : Q \times M \to Q$ is a transition function,
- $F \subseteq Q$ is the set of final states of the automaton,
- $\mathit{pay} : F \times M \to \mathbf{R}$ is a *payoff function.*

Let $\rho = s_0 s_1 \ldots s_n$ be a run of a market automaton $\mathcal{M}$. The *run of the contract automaton* $\mathcal{C}$ on input ρ is the sequence of pairs $\mathcal{C}(\rho) = (s_0, q_0), (s_1, q_1), \ldots (s_n, q_n)$ defined by $q_{k+1} = \delta_c(q_k, s_{k+1})$ for $k = 0 \ldots n-1$. If q_n is a final state of $\mathcal{C}$, but q_k is not final for $k < n$, then we say that the run is *terminated.* The *payoff* on a terminated run ρ, denoted $\mathit{pay}(\rho)$, is defined to be the payoff in the final state, i.e., $\mathit{pay}(q_n, s_n)$.

[1]More generally, one could also allow inputs corresponding to decisions of the parties to the contract. We omit these because we are primarily interested in this paper in European options, where they are not needed.

Example 1

A standard European "down-and-in" option is an exotic option which gives an ordinary European option iff the price barrier (which should be lower than the initial asset price) was touched by the asset price from above in the life circle of the option. Consider a European "down and in" call barrier option with a termination time T, strike price X, and barrier $B < X$ corresponding to a level l_B in the binomial tree (i.e., l_B is the greatest level l such that $\mathit{price}(l) \leq B$). Such a contract can be represented by a contract automaton with

- states $Q = \{\texttt{out}, \texttt{in}, \texttt{exp\_out}, \texttt{exp\_in}\}$,
- initial state out,
- final states $F = \{\texttt{exp\_out}, \texttt{exp\_in}\}$,
- transition function defined by

$$\delta_c(\texttt{out}, (l,t)) = \begin{cases} \texttt{out} & \text{if } l > l_B \text{ and } t < T \\ \texttt{in} & \text{if } l \leq l_B \text{ and } t < T \\ \texttt{exp\_out} & \text{if } l > l_B \text{ and } t \geq T \\ \texttt{exp\_in} & \text{if } l \leq l_B \text{ and } t \geq T \end{cases}$$

and

$$\delta_c(\texttt{in}, (l,t)) = \begin{cases} \texttt{in} & \text{if } t < T \\ \texttt{exp\_in} & \text{if } t \geq T \end{cases}$$

and $\delta_c(\texttt{exp\_out}, (l,t)) = \texttt{exp\_out}$ and $\delta_c(\texttt{exp\_in}, (l,t)) = \texttt{exp\_in}$.

- payoff function defined for $q \in F$ by:

$$\mathit{pay}(q, (l,t)) = \begin{cases} \mathit{price}(l) - X & \text{if } q = \texttt{exp\_in} \text{ and } \mathit{price}(l) \geq X \\ 0 & \text{otherwise} \end{cases}$$

where X is the exercise price of the option.

Intuitively, the states of the contract could be currently in-the-money/out-of-the-money but before the expiry date in, out, or expired in-the-money/out-of-the-money (i.e., exp_in, exp_out). Initially, the option is out-of-the-money, and the only way it can change the state to being in-the-money is by reaching the barrier. The option is expired when $t \geq T$, after which it cannot change state. If the option expired in-the-money, we have the choice to exercise the option if the market price is higher than the strike price. Otherwise, the option is worthless.

■

We say that a contract *expires at time* $T \in \boldsymbol{N}$ if all runs of length $T+1$ are terminated. In what follows, we confine ourselves to contracts that expire at some fixed time T. A sufficient condition for this is that the transition function satisfy that $\delta_c(q,(l,t))$ is final iff $t \geq T$, for all $q \in Q \setminus F$ and $(l,t) \in M$.

For contracts expiring at a fixed time T, it is convenient to use a slightly different form of contract automata, defined as tuples $\mathcal{C} = \langle Q, q_0, \delta_c, \mathit{pay} \rangle$, in which we omit the set of final states, and take *pay* to be a function from Q to $\mathbf{R}$. Each such automaton generates a contract automaton $\mathcal{C}(T) = \langle Q', q'_0, \delta'_c, F', \mathit{pay}' \rangle$ in the above form, defined by $Q' = Q \times \{0,1\}$, $q'_0 = (q_0, 0)$, $F' = Q \times \{1\}$ and

$$\delta'_c((q,x),(l,t)) = \begin{cases} (\delta_c(q,(l,t)), 0) & \text{if } x = 0 \text{ and } t < T \\ (\delta_c(q,(l,t)), 1) & \text{if } x = 0 \text{ and } t \geq T \\ (q,1) & \text{if } x = 1 \end{cases}$$

Intutively, in this form, the payoff function gives the payoff to be received at a given contract state if the contract *were* to expire in that state.

Example 2

European barrier options as defined in Example 1 can be represented more succinctly with

- states $Q = \{\texttt{out}, \texttt{in}\}$
- initial state out,
- transition function defined by

$$\delta_c(q,(l,t)) = \begin{cases} \texttt{out} & \text{if } q = \texttt{out} \text{ and } l \geq l_B, \\ \texttt{in} & \text{otherwise} \end{cases}$$

- payoff function defined for $q \in Q$ by:

$$\mathit{pay}(q,(l,t)) = \begin{cases} \mathit{price}(l) - X & \text{if } q = \texttt{in} \text{ and } \mathit{price}(l) \geq X, \\ 0 & \text{otherwise} \end{cases}$$

■

For contracts $\mathcal{C}$ expiring at time T, we can define the *expected payoff* in a market $\mathcal{M}$, to be the expected value of the payoff in terminated runs, i.e.,

$$E(\mathcal{M}, \mathcal{C}) = \Sigma_{\rho \in \texttt{IRuns}_T(\mathcal{M})} \ \mathit{pay}(\rho) \cdot P(\rho) \qquad (1)$$

where the probabilities $p(\rho)$ on runs ρ arise from the distribution on $\texttt{IRuns}_T(\mathcal{M})$ described above. The *price* of the contract is the value which, invested at time 0 at the risk-free rate of interest r, gives a payoff equivalent to the expected payoff of the contract. For contracts expiring at time T, this can be expressed as $D(T) \cdot E(\mathcal{M}, \mathcal{C})$, where $D(T) = e^{-r \cdot T \cdot \delta t}$ is the discounting factor.

4 Sequential Barrier Options

Now we introduce the multiple barrier options, which are an extension of the single barrier options described above.

Example 3

Suppose we have a sequence of barriers B_0, B_1, B_2... B_m, given as levels in the binomial lattice. Consider a call option which comes into existence if and only if the stock price level hits B_0 and then B_1 and so on, eventually reaching B_m. The rest of the definition remains as in Example 2. We call this a *multiple-barrier hit-and-in call option.*

- states $Q = \{\texttt{out}, \texttt{out}_0, \texttt{out}_1, \cdots, \texttt{out}_{\texttt{m}-1}, \texttt{in}\}$
- initial state out,
- transition function defined by

$$\delta_c(q,(l,t)) = \begin{cases} \texttt{out} & \text{if } q = \texttt{out} \text{ and } l \neq B_0 \\ \texttt{out}_0 & \text{if } q = \texttt{out} \text{ and } l = B_0 \\ \texttt{out}_0 & \text{if } q = \texttt{out}_0 \text{ and } l \neq B_1 \\ \texttt{out}_1 & \text{if } q = \texttt{out}_0 \text{ and } l = B_1 \\ \cdots & \\ \texttt{out}_\texttt{m} - 1 & \text{if } q = \texttt{out}_\texttt{m} - 1 \text{ and } l \neq B_m \\ \texttt{in} & \text{if } q = \texttt{out}_\texttt{m} - 1 \text{ and } l = B_m \end{cases}$$

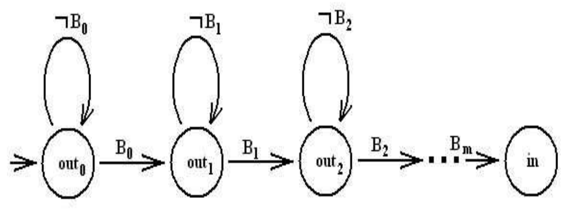

Figure 2: A multiple barrier contract automaton

Figure 2 gives an illustration of this automaton. ■

The multiple barrier option, which consists of arbitrary number of barriers, generalize the sequential barrier options of Pfeffer (Pfeffer 2001), which consists of only two barriers. Sequential barrier options have recently become popular in the Japanese over-the-counter equity and foreign exchange derivative markets.

5 Partial Symbolic Pricing Rules

To develop the efficient pricing calculation, we work with the expected payoff function. We first note that all runs ending at a given level in the binomial tree have the same probability. For, consider a run ρ with $fin(\rho) = (l, T)$. Let l_u be the number of "up" moves in this run, and l_d the number of "down" moves. By definition, the probability of the run is $P(\rho) = p^{l_u}(1-p)^{l_d}$. Then we have $l_u + l_d = T$ and $l_u - l_d = l$. Thus, $l_u = (T+l)/2$ and $l_d = (T-l)/2$, so the probability of the run depends only on T and l. (Recall that $T + l$ must even for (l, T) to be reachable, hence $T - l$ is even also.) We write this probability as $P(l, T)$.

Thus, we may write the expected payoff $E(\mathcal{M}, \mathcal{C}(T))$ of the contract as

$$\sum_{\substack{l \in [-T,T] \\ T+l \text{ even}}} \sum_{q \in Q} pay(q, (l, T)) \cdot N(q, l, T) \cdot P(l, T) \quad (2)$$

where $N(q, l, T)$ is given by

$$|\{\rho \in \mathtt{IRuns}_T(\mathcal{M}) \mid fin(\mathcal{C}(T)(\rho)) = (q, (l, T))\}|.$$

Since the number of levels l in reachable states of the market automaton at time T is $T + 1$, it will follow that the payoff can be calculated in time linear in T, provided that we can establish that the numbers $N(q, l, T)$ can be calculated in constant time.

In order to establish this, we develop a formal calculus within which we can determine the numbers $N(q, l, T)$, based on a logical language for expressing properties of runs. The language is similar to the path expressions of process logic (Pratt 1979).

We define the syntax of the expression language as follows. There are two types of expressions: state expressions and path expressions. A state expression has the form $\mathtt{L} = k$ where $k \in \mathbf{N}$. A path expression has one of the forms $\mathtt{first}(\phi)$, $\mathtt{last}(\phi)$, $\phi \rhd \psi$ (read 'ϕ up to first ψ'), or $\alpha;\beta$ (read 'α chop β'), where ϕ, ψ are state expressions, and α, β are path expressions. For brevity, we write $\phi_0 \rhd \phi_1 \rhd \ldots \rhd \phi_m$ for $(\phi_0 \rhd \phi_1); (\phi_1 \rhd \phi_2); \ldots; (\phi_{m-1} \rhd \phi_m)$.

The semantics of these expressions is defined as follows. In each case, we define a semantic relation $o \models e$ between a semantic object o and an expression e, that intuitively holds when the expression e is true of the object o. For state expressions, the semantic objects are states (l, T) of the contract automaton, and we define the relation $(l, T) \models \mathtt{L} = k$ to hold when $l = k$. For path expressions, we take the semantic objects to be runs $\rho = s_0 \ldots s_n$ of the market automaton, and define

- $\rho \models \mathtt{first}(\phi)$ if $init(\rho) \models \phi$,
- $\rho \models \mathtt{last}(\phi)$ if $fin(\rho) \models \phi$,
- $\rho \models \alpha\ ;\ \beta$ if there exists $k \leq n$ such that $s_0 \ldots s_k \models \alpha$ and $s_k \ldots s_n \models \beta$,
- $\rho \models \phi \rhd \psi$ if $s_0 \models \phi$ and $s_n \models \psi$ and $s_k \not\models \psi$ for $0 < k < n$.

Example 4

Given a contract automaton with multiple strike-and-in barriers $B_0, B_1, \ldots, B_n$, as in Figure 2, together with the initial price level l_0 of the market automaton, we can construct a formula expressing that a run of the automaton ends in the final state *in* with the final market price at level l: $(L = l_0) \rhd (L = B_0) \rhd (L = B_1) \rhd \ldots \rhd (L = B_n); \mathtt{last}(L = l)$. ■

For path expressions α, define $N_T(\alpha)$ to be the number of runs ρ of $\mathcal{M}$ with $|\rho| = T + 1$ such that $\rho \models \alpha$. In order to use the representation of the number $N(q, l, T)$ in the form $N_T(\alpha)$ to calculate its value, we use the following properties of the functions N_T.

1. Reflection Rule:

$$N_T((\mathtt{L} = a \rhd \mathtt{L} = b); \alpha) = N_T((\mathtt{L} = 2b - a \rhd \mathtt{L} = b); \alpha)$$

2. Reduction Rule 1:

$$N_T((\mathtt{L} = a \rhd \mathtt{L} = b); \mathtt{last}(\mathtt{L} = x)) = N_T(\mathtt{first}(\mathtt{L} = a); \mathtt{last}(\mathtt{L} = x))$$

where $a \geq b \geq x$ or $a \leq b \leq x$.

3. Reduction Rule 2:

$$N_T((\mathtt{L} = a \rhd \mathtt{L} = b \rhd \mathtt{L} = c); \alpha) = N_T((\mathtt{L} = a \rhd \mathtt{L} = c); \alpha)$$

where $a \geq b \geq c$ or $a \leq b \leq c$.

4. Counting rule:

$$N_T(\mathtt{first}(\mathtt{L} = a); \mathtt{last}(\mathtt{L} = b)) = \binom{T}{\frac{T-(b-a)}{2}}$$

We give the proof of soundness of these rules in Section 7. The reflection rule captures in a pleasant logical form the reflection principle from combinatorial path counting (Mohanty 1980), which has previously been used by Lyuu (Lyuu 1998) to obtain a linear-time algorithm for (single) barrier option valuation.

6 Algorithms and Time Complexity

6.1 Simplification of the Path Formula

Consider a path expression α_0 of the form

$$(\mathtt{L} = l_0) \triangleright (\mathtt{L} = l_1) \triangleright (\mathtt{L} = l_2) \triangleright \cdots \triangleright (\mathtt{L} = l_m); \mathtt{last}(L = l).$$

We show that there exists a sequence $\alpha_1, \ldots, \alpha_m$ of path expressions such that $N_T(\alpha_i) = N_T(\alpha_{i+1})$ for all $i = 0 \ldots m-1$ and such that $N_T(\alpha_m)$ can be evaluated explicitly using the counting rule. More specifically, α_i has the form $(\mathtt{L} = a_i) \triangleright (\mathtt{L} = l_{i+1}) \triangleright \cdots \triangleright (\mathtt{L} = l_m); \mathtt{last}(L = l)$ for $i < m$, and α_m has the form $(\mathtt{L} = a_m); \mathtt{last}(L = l)$, for constants $a_1 \ldots a_m$, defined inductively as follows. For convenience, we let $a_0 = l_0$.

1. If $\alpha_i = (\mathtt{L} = a_i) \triangleright (\mathtt{L} = l_{i+1}) \triangleright (\mathtt{L} = l_{i+2}) \triangleright \ldots \triangleright (\mathtt{L} = l_m); \mathtt{last}(\mathtt{L} = l)$ contains more than one occurrence of the $\triangleright$ operator, we may eliminate one occurrence of the $\triangleright$ operator as follows. If $a_i \leq l_{i+1} \leq l_{i+2}$ or $a_i \geq l_{i+1} \geq l_{i+2}$, we define $a_{i+1} = a_i$, otherwise $a_{i+1} = 2l_{i+1} - a_i$, which will satisfy either $a_{i+1} \leq l_{i+1} \leq l_{i+2}$ or $a_{i+1} \geq l_{i+1} \geq l_{i+2}$. It then follows using the Reflection Rule and Reduction Rule 1 that $N_T(\alpha_{i+1}) = N_T(\alpha_i)$.
2. For $\alpha_{m-1} = (\mathtt{L} = a_{m-1}) \triangleright (\mathtt{L} = l_m); \mathtt{last}(\mathtt{L} = l)$, we again consider two cases: If $a_{m-1} \leq l_m \leq l$ or $a_{m-1} \geq l_m \geq l$, we define $a_m = a_{m-1}$, otherwise, we define $a_m = 2l_m - a_{m-1}$. In this case, it follows using the Reflection Rule and Reduction Rule 2 that $N_T(\alpha_{m-1}) = N_T(\alpha_m)$.

The counting rule may now be applied to $N_T(\alpha_m)$ to yield a value for $N_T(\alpha_0)$. To capture this as a function of l, we write this value in the form

$$N_T(\alpha_0) = \begin{cases} \binom{T}{f^+(l)} & \text{if } a_{m-1} \leq B_m \leq l \\ & \quad \text{or } a_{m-1} \geq B_m \geq l, \\ \binom{T}{f^-(l)} & \text{otherwise} \end{cases}$$

where $f^+(l) = (T - (a_{m-1} - l))/2$ and $f^-(l) = (T - (2B_m - a_{m-1} - l))/2$.

6.2 Option Pricing

To price the multiple barrier strike-and-in call option, consider the terms in equation (2). The probabilities are given by $P(l,T) = p^{\frac{T+l}{2}} \cdot (1-p)^{\frac{T-l}{2}}$. However, many of the terms are zero, since the payoff $\mathit{pay}(q,(l,T))$ is zero except when $q = \mathtt{in}$ and $\mathit{price}(l) > X$, when it equals $\mathit{price}(l) - X$. Now $N(\mathtt{in}, l, T) = N_T(\alpha_l)$, where α_l is the path expression $(\mathtt{L} = 0) \triangleright (\mathtt{L} = B_0) \triangleright \ldots \triangleright (\mathtt{L} = B_m); \mathtt{last}(L = l))$. Using the procedure of the previous section, we obtain the following pricing formula for the derivative:

$$\begin{aligned} &E(\mathcal{M}, \mathcal{C}(T)) = \\ &\textstyle\sum_{l \in S^+} \ (\mathit{price}(l) - X) \cdot \binom{T}{f^+(l)} \cdot p^{\frac{T+l}{2}} \cdot (1-p)^{\frac{T-l}{2}} \\ &+ \\ &\textstyle\sum_{l \in S^-} \ (\mathit{price}(l) - X) \cdot \binom{T}{f^-(l)} \cdot p^{\frac{T+l}{2}} \cdot (1-p)^{\frac{T-l}{2}} \end{aligned}$$

where $S = \{l \mid [-T,T],\ T + l \text{ even},\ \mathit{price}(l) \geq X\}$ and $S^+ = S \cap C$ and $S^- = S \setminus C$, for $C = \{l \mid a_{m-1} \leq B_m \leq l \text{ or } a_{m-1} \geq B_m \geq l\}$. This sum can be computed with a number of additions and multiplications linear in T.

A European strike-and-out call barrier option is defined by an automaton that has the same states and transitions as the strike-and-in option depicted in Figure 2, but payoff 0 when the automaton is in state $\mathtt{in}$ at time T and payoff $\mathit{price}(l) - X$ at all other states. This contract can be valued in linear time by a very similar approach, based on the equation

$$E(\mathcal{M}, \mathcal{C}(T)) =$$
$$\sum_{l \in S} \ (\mathit{price}(l) - X) \cdot (N_T(0,l) - N(\mathtt{in}, l, T)) \cdot P(l,T)$$

in which the term $(N_T(0,l) - N(\mathtt{in}, l, T))$ counts the number of paths on which a payoff is obtained as the complement of the $N(\mathtt{in}, l, T)$ paths on which a payoff is not obtained. The corresponding put options can also be priced by similar equations.

7 Soundness Proofs

In this section we give the correctness proofs for the rules introduced above.

7.1 Reflection Rule

To prove the Reflection Rule:

$$N_T((\mathtt{L} = a) \triangleright (\mathtt{L} = b); \alpha) = N_T((\mathtt{L} = 2{\cdot}b - a) \triangleright (\mathtt{L} = b); \alpha)$$

we establish that there exists a 1-1 correspondence from the set

$$S_1 = \{\rho \in \mathtt{Runs}_T(\mathcal{M}) \mid \rho \models (\mathtt{L} = a) \triangleright (\mathtt{L} = b); \alpha\}$$

to the set

$$S_2 = \{\rho \in \mathtt{Runs}_T(\mathcal{M}) \mid \rho \models (\mathtt{L} = 2b - a) \triangleright (\mathtt{L} = b); \alpha\}.$$

To show this, it is convenient to define $f : \mathtt{Runs}_T(\mathcal{M}) \to \mathtt{Runs}_T(\mathcal{M})$ as follows. Let $\rho = s_0, \ldots, s_T \in \mathtt{Runs}_T(\mathcal{M})$, where $s_i = (l_i, i)$ for $i = 0 \ldots T$. We define $f(\rho)$ to be $(2b - l_0, 0), \ldots, (2b - l_{t-1}, t-1), s_t, \ldots, s_T$ where $t \in [0,T]$ is the least number such that $l_t = b$, if such a t exists, and $f(\rho) = \rho$ otherwise. Since $l = b$ iff $2b - l = b$ and $2b - (2b - l) = l$, it is immediate that $f \circ f$ is the identity on $\mathtt{Runs}_T(\mathcal{M})$. It therefore suffices to show that f maps S_1 into S_2 and f maps S_2 into S_1. We show the former, the latter is similar. Suppose that $\rho = s_0 \ldots s_T \models (\mathtt{L} = a) \triangleright (\mathtt{L} = b); \alpha$, where $s_i = (l_i, i)$. Then there exists a $k \leq T$ such that $s_0 \ldots s_k \models (\mathtt{L} = a) \triangleright (\mathtt{L} = b)$ and $s_k \ldots s_T \models \alpha$. In particular, $s_0 \models \mathtt{L} = a$ and k is the least number such that $s_k \models \mathtt{L} = b$. Thus $f(\rho) = (2b - a, 0), \ldots, s_k, s_{k+1}, \ldots, s_T$, for which $(2b - a, 0), \ldots, s_k \models (\mathtt{L} = 2b - a) \triangleright (\mathtt{L} = b)$ and $s_k, \ldots s_T \models \alpha$. from which it follows that $f(\rho) \in S_2$.

7.2 Reduction Rule 1

To prove the Reduction Rule 1:

$$\begin{aligned} N_T((\mathtt{L} = a \triangleright \mathtt{L} = b); \mathtt{last}(\mathtt{L} = x)) &= \\ N_T(\mathtt{first}(\mathtt{L} = a); \mathtt{last}(\mathtt{L} = x)) \end{aligned}$$

where $a \geq b \geq x$ or $a \leq b \leq x$, it suffices to note that for $\rho = s_0, \ldots s_T \in \mathtt{Runs}_T(\mathcal{M})$, we have $\rho \models (\mathtt{L} = a \triangleright \mathtt{L} = b); \mathtt{last}(\mathtt{L} = x)$ iff $\rho \models N_T(\mathtt{first}(\mathtt{L} = a); \mathtt{last}(\mathtt{L} = x))$. From left to right this is immediate. From right to left, note that if $s_0 \models \mathtt{L} = a$ and $s_T \models (\mathtt{L} = x)$, then since $a \geq b \geq x$ or $a \leq b \leq x$, and the levels increment or decrement by one each step, there must exist k such that $s_k \models \mathtt{L} = b$. Taking the least such k, we obtain that $s_0, \ldots, s_k \models (\mathtt{L} = a \triangleright \mathtt{L} = b)$ and $s_k, \ldots, s_T \models \mathtt{last}(\mathtt{L} = x)$, hence $\rho \models (\mathtt{L} = a \triangleright \mathtt{L} = b); \mathtt{last}(\mathtt{L} = x)$.

7.3 Reduction Rule 2

To prove Reduction Rule 2:

$$N_T((\mathtt{L} = a \triangleright \mathtt{L} = b \triangleright \mathtt{L} = c); \alpha) = N_T((\mathtt{L} = a \triangleright \mathtt{L} = c); \alpha)$$

where $a \geq b \geq c$ or $a \leq b \leq c$, we consider the case where $a \geq b \geq c$, the other case is similar. We show that if $s_0 \ldots s_k$ is a prefix of an element of $\mathtt{Runs}_T(\mathcal{M})$, where $s_i = (l_i, i)$, we have $s_0, \ldots, s_k \models (\mathtt{L} = a \triangleright \mathtt{L} = b); (\mathtt{L} = b \triangleright \mathtt{L} = c)$ iff $s_0, \ldots, s_k \models (\mathtt{L} = a \triangleright \mathtt{L} = c)$. From right to left, this follows from the fact that levels increment or decrement by one at each step. From left to right, suppose that $s_0, \ldots, s_k \models (\mathtt{L} = a \triangleright \mathtt{L} = b); (\mathtt{L} = b \triangleright \mathtt{L} = c)$. Then there exists $m \leq k$ such that $s_0, \ldots, s_m \models (\mathtt{L} = a \triangleright \mathtt{L} = b)$ and $s_m, \ldots, s_k \models (\mathtt{L} = b \triangleright \mathtt{L} = c)$. Thus $l_i > b$ for $i = 0 \ldots m-1$, and $l_i > c$ for $i = m \ldots k-1$. Since $b \geq c$, it follows that $l_i > c$ for $i = 0 \ldots k-1$, hence $s_0, \ldots, s_k \models (\mathtt{L} = a \triangleright \mathtt{L} = c)$.

7.4 Counting Rule

To prove the counting rule, consider a path of $\mathtt{Runs}_T(\mathcal{M})$ from level a to level b. Let u be the number of upward moves along this path, and d the number of downward moves. Then $u + d = T$ and $u - d = b - a$, so there are exactly $d = (T - (b-a))/2$ downwards moves. Since a choice of such a path corresponds to a choice of d times from T for the downward moves to occur, we obtain that there exist $\binom{T}{\frac{T-(b-a)}{2}}$ such paths.

8 Empirical Performance

We implemented both the backward induction pricing method and the combinatorial pricing method discussed above in MATLAB. In order to compare the performance of the two methods, we randomly generate sequential barrier options with 4,5 and 6 barriers. The barrier prices are generated in the following way. For a sequential barrier option with k barriers, we first generate k random numbers between 0 and N where N is the lattice depth. We sort these numbers as $n_1 \leq n_2 \leq \ldots n_k$, then take the barriers to be $S_0 \cdot u^{n_1}$, $S_0 \cdot u^{n_1} \cdot d^{(n_2 - n_1)}$, $S_0 \cdot u^{n_1} \cdot d^{(n_2-n_1)} \cdot u^{(n_3-n_2)} \ldots$, where S_0 is the initial price (a randomly generated real number between 0 and 30), u is the up-rising ratio of the asset and d is u^{-1}. Generally speaking, the ith barrier will be:

$$b_i = S_0 \cdot (u^{2 \cdot \sum_{j=1}^{i-1} (-1)^{j+1} \cdot n_j + (-1)^{i+1} n_i})$$

This guarantees that there is in fact a path that reaches the final state of the option, so the contract makes some sense. The exercise price of these options is randomly set to a real number between 0 and 30, and the option is randomly set to be either a put or a call. The rest of the parameters are given below:

- riskless interest rate : 0.05 per annum
- volatility : 0.02
- life cycle : 30 days

We tune the lattice depth from 25 to 250. For each lattice depth, we randomly generated 100 sequential options and run both pricing functions in MATLAB 7 on a 2.80 GHz Pentium(R) 4 CPU with 1.0 GB main memory. The average running time for backward induction pricing is compared with the average running time for combinatorial pricing in Figure 3. The curve for backward induction pricing presents a

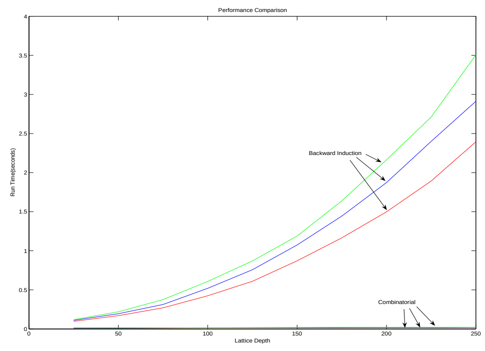

Figure 3: Performance Comparison of both backward induction and combinatorial pricing algorithms

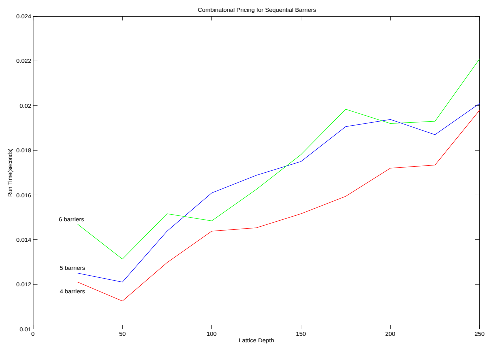

Figure 4: Computation time(in second) of combinatorial pricing for sequential barrier options with 4, 5 and 6 barriers.

clear quadratic profile. The performance of combinatorial pricing is depicted at greater resolution in Figure 4: it is expected to be linear, but the lack of smoothness in the curve appears to be due to a lack of accuracy of MATLAB runtime profiling at this low scale. The computation time always remains below 0.024 second even for large lattice depth.

Convergence It is known that barrier options display a slow sawtooth convergence pattern with respect to CRR binomial tree (Lyuu 2002), due to rounding errors in relating barriers to a price level in the binomial tree. The convergence pattern is smoother with respect to choices of lattice depth that ensure that barrier values fall near price levels in the tree. We run our pricing algorithm with one particular randomly generated sequential barrier option with other parameters stated above and tune the lattice depth from 1 to 250. The convergence of the computed price is depicted in Figure 5. The algorithm demonstrates reasonable precision when the lattice depth is more than 100.

9 Conclusion and Future work

We have generalized Lyuu's work on linear time pricing of European barrier options based on the binomial lattice to a class of European derivatives consisting of a sequence of barriers. In doing so, we have developed a representation for the general contract pricing problem using probabilistic finite state automata, and introduced a path expression language within which we may express useful rules for path counting. Several directions suggest themselves:

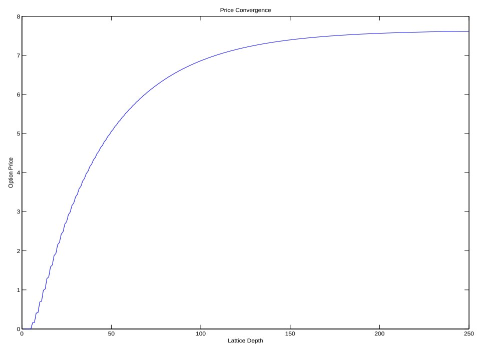

Figure 5: Price convergence for a sequential barrier option with 5 barriers

- There are many more exotic contract types, and it would be interesting to investigate the extent to which the sorts of mixed symbolic and numerical computation techniques used in this paper can be applied to obtain efficient pricing algorithms.
- The algorithmic verification literature has introduced some richer types of automata than we have considered in this paper, such as probabilistic timed automata (Kwiatkowska 2003). It would be interesting to find applications of analysis techniques for such automata to derivative pricing.
- Morgan and McIver (McIver & Morgan 2002) have introduced a quantitative μ calculus to represent a game between two players. The elvaluation of the μ calculus formula gives the player's expectation of the payoff from this game. This kind of turn-based game could be applied to representing and pricing the American-style options since the option price in this case is related to an optimal option trading strategy. The qMμ calculus can specify connectives like maxjunction (which returns the max expectation of the two subformula), and has an operator for recursion, using which the backward induction algorithm can be specified. It would be interesting to find efficient pricing algorithms based on logical transformations of the formulas representing a contract price in this calculus.

We leave exploration of these ideas for future research.

References

Black, F., Scholes, M. (1973), The Pricing of Options and Corporate Liabilities, *in* 'Journal of Political Economy', Vol. 81, No.3, pp. 637-654.

Cox, J., Ross, S. & Rubinstein, M. (1979), Option Pricing: A Simplified Approach, *in* 'Journal of Financial Economics', Vol. 7, pp. 229-264.

Lyuu, Y.D. (1998), Very Fast Algorithm for Barrier Option Pricing and the Ballot Problem, *in* 'The Journal of Derivatives', Vol. 7, No.1, pp. 79-90.

Merton, R.C. (1994), *Continuous-Time Finance*, Blackwell.

Lyuu, Y.D. (2002), *Financial engineering and computation : principles, mathematics, algorithms*, Cambridge University Press.

McIver, A.K. &Morgan C.C. (2002), Results on the Quantitative μ-calculus qMμ*, *in* the 9th Int'l Conference on Logic for Programming and Automated Reasoning.

Pfeffer, D. (2001), Sequential Barrier Options, *in* 'Algo Research Quarterly', Vol. 4, No.3, pp. 65-74.

Hull, J. (2002), *Options, futures and other derivatives*, Prentice Hall.

Mohanty, G. (1980), *Lattice Path Counting and Applications*, Academic Pr.

Pratt, V.R. (1979), Process Logic, *in* 6^th Annual ACM Symposium on Principles of Programming.

Kwiatkowska, M. (2003), Model checking for probability and time: from theory to practice, *in* Proc. Logic and Computer Science.