Computability-Based Analysis of Market Predictability

Trabajo de Fin de Máster
Máster en Metodos Formales en Ingeniería Informática

Universidad Complutense de Madrid
2021-2022
Autor: Carlos Ignacio Isasa Martín
Tutor: Ismael Rodríguez Laguna

Convocatoria: Septiembre 2022
Nota: 9.8/10

Acknowledgements

First and foremost, I'd like to thank my parents for always being there. Their support has carried me through the years and I'm sure it will be there for life. I'd also like to thank Lucas, Gregory, Koichi, Álvaro and Kilian, for lending me an ear at 5 in the morning when I'm stressed out of this world, even though they don't understand my ramblings. To Tati, for letting me take over the dinning room to sleep, eat and work under the AC. To Ismael, for his invaluable help writing this thesis. And finally to Paula, for her always present understanding and support.

Abstract

Financial markets are not often seen as computational systems but, what if we could treat them as one? What would their computational power be? Could we predict if prices are going to stabilize through other, more powerful, computational analysis? In this work we provide two different formal models of markets as computational systems and prove that both are Turing complete, strongly indicating that predicting price stabilization is not possible.

Keywords

Economics, Financial Markets, Computability, Turing Completeness.

Resumen

Los mercados financieros no suelen analizarse como sistemas computacionales, pero ¿qué pasaría si pudieramos tratarlos como uno? ¿Cuál sería su potencia computacional? ¿Podríamos predecir si los precios de un mercado se estabilizarán usando análisis computacionales más potentes? En este trabajo describiremos dos modelos formales de mercados como sistemas computacionales y demostraremos que ambos son Turing completos, lo cuál nos da una intuición fuerte de que predecir la estabilización en el precio no es posible.

Palabras Clave

Economía, Mercados Financieros, Computabilidad, Turing Completitud.

Contents

CHAPTER 1

Introduction

Stock markets (markets from now on) are one of the most important institutions of every capitalist system. Investing in them is often seem as a way of making money, being by expecting returns in the long run or purely speculating on market trends. But what would happen if we treat them as a way to find the "correct" price of a stock? Let us suppose that we design a closed system where traders are allowed to buy and sell stock until they are satisfied with its price. Could we predict what would that price be knowing before hand the initial state of the market and every trader's strategy?

In order to do that we are going to construct two distinct formal models of markets as computational systems. These markets will have a initial state, which has a different set of parameters for each model, and a set of traders, each one with their own strategy. The markets will then iterate, as traders buy and sell stock according to their own strategies until every trader thinks that the price is right for the stock and stops buying and selling. The differences between these two models lie in the way that each of them assumes how markets work. For the first one we used a fine grain approach to market behaviour, building it in a way that models precisely how the price of the stock changes by using transactional offers such that the last accepted one marks the price of the stock. However, the trade off for this is having traders that have little expressiveness and their strategies depend only on the most significant digit of the amount of the last offer. For the second model we used broader strokes to model the market, having the price vary according to the supply and

demand law and traders trying to buy or sell volumes of the stock to model supply and demand. The advantage lies in traders having a much more expressive language, which we can use to model more realistic behaviours.

We are going to prove that both models are Turing complete, and as such, predicting if the price is going to stabilize and how much is it going to be are problems as hard as the Halting problem, which makes them undecidable. According to Rice's Theorem, as the system is Turing complete, this undecidability extends to every non-trivial semantic property (e.g. will the stock price reach X at some point? or will X price be reached before Y price?). If we assume our models as a realistic modelization of stock markets, they prove a deep unpredictability in markets, as markets will be unpredictable even in the complete absence of uncertainty. We model the market as deterministic, free of foreign influences (e.g. good or bad news) and the strategies of all traders are known, and even under those conditions we cannot predict if the price is going to stabilize.

CHAPTER 2
Preliminaries

2.1 Markets from a Computational Standpoint

There are several branches of formal research about market predictability, as proving whether or not you can predict the equilibrium price or the next price of a stock is directly correlated to making money. A well-known line of research in market predictability is on the weak efficiency hypothesis, which says that future prices are not influenced by past events, only the current one, and its fluctuations behave like a random walk. The main argument supporting this assertion comes from the supposition that if a piece of information regarding the price of the stock is widely available to traders (as past prices are), traders can trade using that information until the price reflects it [1].

On the one hand we have papers such as [2], defending the position that the weak efficiency hypothesis is correct. Samuelson does this by creating a stochastic process that models the price of the stock and then proves that this system behaves like a martingale (i.e. a random walk). As this paper approaches markets modelling them as stochastic systems instead of a computational model, it is not really of interest to us. Still, it provides insight on how economists are approaching this problem.

On the other other hand, [3] rises an interesting counterpoint,

asserting that markets are weakly efficient if and only if P=NP. Maymin does this by virtue of building a market in such a way that the information available to all traders will outgrow the capacity of traders to check for strategies if P≠NP. This paper has a fair share of problems (e.g. assumptions of the capacity of traders to check strategies, assumptions of the speed of a market to balance itself in the construction of a market that solves NP-complete problems in polynomial time, etc.), but it is one of the few papers that treats markets as computational objects.

The debate that surrounds the weak efficiency hypothesis is one of the most important lines of research of economics, but, besides Maymin paper, there seems to be a lack of papers approaching it from a computational standpoint. Thus, we shall move on to papers that deal with market predictability without going into the weak efficient hypothesis.

First, we have [4], perhaps the paper that is the closest to this work, even though it uses the results to argue something unrelated to our work. This paper uses a computational approach in order to prove that the complexity of markets has grown over time. For that, it uses several market models that were developed at different times in history and proves that each of them is characterized by a different type of automaton, each automaton being more complex as time passes. The line of research of this paper belongs in a branch of economics called "evolutionary economics", which argues that superior economic systems will replace inferior ones through history. Evolutionary economics as a field is not of interest for us, but this paper is one of the few that tries to show the equivalence between markets and other computational systems, which is what we are trying to do.

Finally, we have [5], which tries to deal with the prediction of market prices through probabilistic Turing machine. Aspnes et al. model market as stochastic processes, where the price depends on the strategies chosen randomly by traders. This model is translated to probabilistic circuits and then properly classified in a new complexity class, which the authors call *BCPP* or bounded conditional probabilistic polynomial-time. The definition and analysis of this

class is outside of the scope of our work, but it still interesting to see how researchers try to merge a stochastic approach to market theory with a computational approach. When we compare our models, we are basing ours on a deterministic behaviour (as opposed to a probabilistic one), and as such we can base their unpredictability on Turing machines' own unpredictability, which means that we do not need new complexity classes.

2.2 Turing Completeness

As the objective of this work is to prove that markets are Turing complete, let us define properly what Turing complete is:

Definition 2.1. A system is said to be **Turing complete** if it is able to simulate any arbitrary Turing machine.

Proving that a market is Turing complete is not an easy task. For the first of our proofs, we shall use a universal Turing machine:

Definition 2.2. A universal Turing machine (UTM) is a Turing machine capable of simulating an arbitrary Turing machine.

Universal Turing machines are the backbone of the theory that developed the first computers. Even though they are theoretical constructs and not practical to build in real life, these machines are still useful for proving the Turing completeness of other systems. As time progressed, smaller and smaller UTMs were build for this purpose, as researchers as Rogozhin [6] or Kudlek [7] developed them. However, proving that a Turing machine is universal is not an easy task and for that, other systems that were Turing complete were used. We have now come full circle, where we use a Turing complete system to prove that a Turing machine is universal to prove that another system is Turing complete. One of these models is m-tag systems, which we will present now:

Definition 2.3. An m-tag system is a three-tuple $T = (m, A, P)$ where:

- m is a positive integer
- $A = \{a_1, \ldots, a_{n+1}\}$ is a finite alphabet

- P is a map from $\{a_1, \ldots, a_n\}$ to A^*, the set of finite words on alphabet A, and from a_{n+1} to `STOP`.

The words $\alpha_i = P(a_i)$ are called the productions of the tag system T. The letter a_{n+1} is called the halting symbol. Productions are usually displayed as follows:

$$\mathcal{T} = \begin{cases} a_i \mapsto \alpha_i & i \neq n+1 \\ a_{n+1} \mapsto \texttt{STOP} \end{cases}$$

A computation of the m-tag system T on word $\beta \in A^*$ is a sequence of words on A, $\beta = \beta_0\beta_1 \ldots$ such as every β_k is transformed into β_{k+1} by removing the m first letters and appending α_i to it, with a_i being the first letter of β_k. The computation stops in the step K if there are less than m letters on β_k or if the first letter of β_k is a_{n+1} [6].

Theorem 2.4. *The family of 2-tag systems is Turing complete [8].*

Proof. First, let us begin by rewriting how 2-symbol Turing machines work. Usually a step of the computation of a Turing machine in state q_i is performed as follows:

1. The Turing machine reads the symbol under the head.
   - If 0 is read:
     (a) The Turing machine writes the symbol $S_{i,0}$.
     (b) The Turing machine moves to the direction $D_{i,0}$.
     (c) The Turing machine state changes to $Q_{i,0}$.
   - If 1 is read:
     (a) The Turing machine writes the symbol $S_{i,1}$.
     (b) The Turing machine moves in the direction $D_{i,1}$.
     (c) The Turing machine state changes to $Q_{i,1}$.

However, this can be rewritten as:

1. The Turing machine writes the symbol S_i.
2. The Turing machine moves in the direction D_i.

3. The Turing machine reads the symbol under the head.
   - If 0 is read:
     (a) The Turing machine state changes to $Q_{i,0}$.
   - If 1 is read:
     (a) The Turing machine state changes to $Q_{i,1}$.

by doubling the amount of states. This technique allows us to have states that can only write a symbol and can only move in one direction, not depending on what the symbol read is (as the state change is done as soon as the symbol is read).

We can summarize the internal state of a Turing machine after reading a symbol with a triplet (Q, L, R), where Q is the current state q_i, L is the number whose binary representation is given by the symbols on the left of the head (with the least significant bit being the symbol closest to the head) and R is the number whose binary representation is given by the symbols on the right of the head (with the least significant bit being the symbol closest to the head). As Q is taken right after reading the symbol under the head, and after re-defining how 2-symbol Turing machines work it contains all the information regarding what reading that symbol does, it is not necessary to include the symbol in the description.

The transitions of the Turing machine can be extracted from how these triplets evolve. For example, a move to the right (a move to left will cause the same transformations but with L and R inverted) writing the symbol S_i will perform the following transformations on L and R:

$$L \leftarrow 2L + S_i$$
$$R \leftarrow \left\lfloor \frac{R}{2} \right\rfloor$$

while Q will update to $Q_{i,0}$ if R was even or to $Q_{i,1}$ if R was odd. As all the information is contained into this series of triplets, to prove that 2-tag systems are Turing complete we just have to find a way to simulate them.

Given a Turing machine with states $q_1, \ldots, q_r$, we define a tag system with symbols $x_i, A_i, \alpha_i, B_i, \beta_i, C_i, c_i, D_{i0}, d_{i0}, D_{i1}, d_{i1}, S_i, s_i, T_{i0}, t_{i0}, T_{i1}, t_{i1}$ with $i = 1, \ldots, r$, so each state q_i has a set of associated symbols. We will show an example of a given q_i assuming a move to the right (a move to the left can be simulated with the appropriate changes). Given the Turing machine description (Q_i, L, R) we represent it in the 2-tag system with the word $A_i x_i (\alpha_i x_i)^L B_i x_i (\beta_i x_i)^R$, where the superscripts L and R represent repeating L and R times, respectively. We will drop the subscript i from now on for typographic reasons, as every symbol will have it, and simply rewrite the word as:

$$Ax(\alpha x)^L Bx(\beta x)^R$$

Now, if S_i (the symbol that Q_i writes) is 0 then this 2-tag system takes production $A \mapsto Cx$ and if S_i is 1 then this 2-tag system takes production $A \mapsto Cxcx$ (with all the symbols in the 2-tag system having the subscript i as all of those symbols belong to the state Q_i), as we need to take into account if L is going to be transformed into $2L$ or $2L + 1$. Additionally, $\alpha \mapsto cxcx$. After these transformations we end up with the word:

$$Bx(\beta x)^R Cx(cx)^{L'}$$

with L' being either $2L$ or $2L + 1$ depending on S_i. Next, $B \mapsto S, \beta \mapsto s$ and we have:

$$Cx(cx)^{L'} S(s)^R$$

Next we apply productions $C \mapsto D_1 D_0$ and $c \mapsto d_1 d_0$ and these yield:

$$S(s)^R D_1 D_0 (d_1 d_0)^{L'}$$

Next, we apply productions $S \mapsto T_1 T_0$ and $s \mapsto t_1 t_0$. With these rules we end up with two cases:

- If R is odd, the next tape space to the right from the space under the head (which is the one that is going to be under the head next, as we are moving to the right) will be 1. Furthermore, as the first rule removes one instance of the symbol s,

we end up with an even number of instances of the symbol s, and after applying the rules we end up with:

$$D_1 D_0 (d_1 d_0)^{L'} T_1 T_0 (t_1 t_0)^{(R-1)/2}$$

The rules for D_1 and d_1 are $D_1 \mapsto A_1 x_1$, $d_1 \mapsto \alpha_1 x_1$ where A_1, α_1 and x_1 correspond to the letters associated to the state $Q_{i,1}$. This yields the word

$$T_1 T_0 (t_1 t_0)^{(R-1)/2} A_1 x_1 (\alpha_1 x_1)^{L'}.$$

Finally, we have the rules $T_1 \mapsto B_1 x_1$ and $t_1 \mapsto \beta_1 x_1$, where B_1, β_1 and x_1 correspond to the letter associated to the state $Q_{i,1}$. We end up with the word

$$A_1 x_1 (\alpha_1 x_1)^{L'} B_1 x_1 (\beta_1 x_1)^{(R-1)/2}$$

which corresponds to the triple $(Q_{i,1}, 2L + S_i, (R-1)/2)$, the one that is a description of the internal state of the Turing machine after being in the internal state (Q_i, L, R), moving right and reading 1.

- If R is even, the next tape space to the right from the space under the head (which is the one that is going to be under the head next, as we are moving to the right) will be 0. Furthermore, as the first rule removes one instance of the symbol s, we end up with an odd number of the symbol s and after we apply the last rule for the symbol s, we delete the symbol D_1. We end up with:

$$D_0 (d_1 d_0)^{L'} T_1 T_0 (t_1 t_0)^{R/2}$$

The rules for D_0 and d_0 are $D_0 \mapsto A_0 x_0$, $d_0 \mapsto \alpha_0 x_0$ where A_0, α_0 and x_0 correspond to the letters associated to the state $Q_{i,0}$. As T_1 gets deleted by the last production rule for the symbol d_0, this yields the word

$$T_0 (t_1 t_0)^{R/2} A_0 x_0 (\alpha_0 x_0)^{L'}.$$

Finally, we have the rules $T_0 \mapsto B_0 x_0$ and $t_0 \mapsto \beta_0 x_0$, where B_0, β_0 and x_0 correspond to the letter associated to the state $Q_{i,0}$. We end up with the word

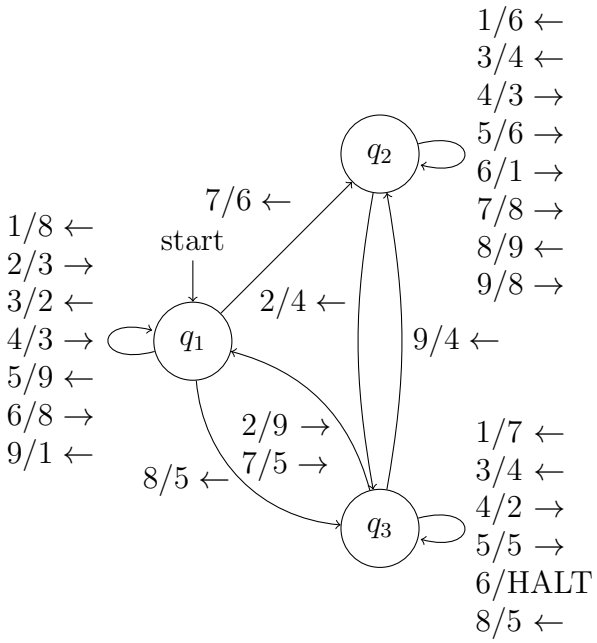
$$A_0 x_0 (\alpha_0 x_0)^{L'} B_0 x_0 (\beta_0 x_0)^{R/2}$$

which corresponds to the triple $(Q_{i,0}, 2L+S_i, R/2)$, the one that is a description of the internal state of the Turing machine after being in the internal state (Q_i, L, R), moving right and reading 0.

If the state Q_i calls for a move to left on the tape of the Turing machine we repeat the same process, but changing the rules for A, α and B, β. With this construction we have proven that 2-tag systems are able to simulate an arbitrary Turing machine and therefore are Turing complete [8]. □

The relevance of 2-tag systems comes from the fact that simulating them on a UTM is relatively easy and, because of that, they are the most common system used in developing small UTMs. One of them, UTM(3,9), is the UTM that we will use for the proof of Turing completeness of the first model:

Definition 2.5. The Universal Turing Machine with 3 states and 9 symbols (1 is the blank symbol) **UTM(3,9)** has the following structure [7]:



Theorem 2.6. *UTM(3,9) is a universal Turing machine*

Proof. Let $T = (2, A, P)$ be a 2-tag system with $A = \{a_1, \ldots, a_n, a_{n+1}\}$ where P maps each a_i to $\alpha_i = a_{i1} \ldots a_{im_i}$ (so there are m_i symbols in the production rule for a_i). We start by assigning a number N_i to each a_i, defined as:

$$N_1 = 1, \quad N_{k+1} = N_k + 2m_k + 2 \quad (k \in \{1, \ldots, n\})$$

For each production α_i and each symbol a_i we will have a code in the Turing machine tape language that represents them (P_i and A_i, respectively). For the productions α_i with $i \in \{1, \ldots, n\}$ we have:

$$P_i = 252225^{N_{im_i}}225^{N_{im_i-1}} \ldots 225^{N_{i2}}225^{N_{i1}}$$

where $5^{N_{i1}}$ means the symbol 5 repeated N_{i1} times. For α_{n+1} we have $P_{n+1} = 62$. Finally, for the symbols a_i we have $A_i = 5^{N_i}$.

Given a word $\beta = a_r a_s \ldots a_w$ in the language of the 2-tag machine, the initial configuration of the tape of the Turing machine will be:

$$Q_L P S Q_R$$

where $P = P_{n+1}P_n \ldots P_1 25$ (with the symbols 2 and 5 being a marker for the separation of P and S), $S = A_r 7 A_s 7 \ldots 7 A_w 7$ and Q_R and Q_L are the infinite amounts of blank symbols to left and right of the tape. The head of the Turing machine will start over the first symbol of S.

They way the Turing machine maps the productions of the 2-tag system T is as follows. Given β_1 and β_2 such that $\beta_1 \mapsto \beta_2$, the following transformation will happen on the Turing machine tape:

$$Q_L P R_1 S_1 Q_R \Rightarrow Q_L P R_2 S_2 Q_R$$

where S_1 and S_2 simulate β_1 and β_2 and R_1 and R_2 are remains of previous operations (R_1 does not occupy any tape space in the first step). The head position will start and end at the beginning of each S_n and the initial state will be the same as the end state, q_1.

Now that we have defined the initial configuration we can go into the inner workings of this simulation:

1. First, the machine finds in S_1 the code P_r corresponding to the production of A_r and after that it deletes the codes for A_r and A_s. Let S_1 be S_1' after this transformation

2. Second, the machine writes the code of P_r between S_1' and Q_R as $A_{r1}, A_{r2}, \ldots, A_{rm_r}$, which fully turns S_1' into S_2

3. Finally, the machine restores its tape, restoring the tape spaces that were filled with substitute symbols back to the original symbols.

The TM rules used for the first stage of the modelling are:

$$\begin{array}{ccccc} q_1 & 5 & 9 & L & q_1 \\ q_1 & 9 & 1 & R & q_1 \\ q_1 & 1 & 9 & L & q_1 \\ q_1 & 2 & 3 & R & q_1 \\ q_1 & 3 & 2 & L & q_1 \\ q_1 & 7 & 6 & L & q_2 \end{array}$$

Given the initial state of the machine and this set of rules, the Turing machine will delete A_r while looking for P_r. This happens because, given the construction of N_r, it will be equal to the amount of 2's between A_r and P_r, so what the machine does is: it deletes a 5 from A_r (changing it to a 9), go to the left until it finds a 2, delete it (changing it to a 3) and go back to A_r to delete a 5 again. This process will stop when the machine reaches the marker 7 that separates A_r from A_s, but at that moment, every tape space that originally contained a 2 between the header and P_r will contain a 3 and every tape space that originally contained a 5 between the header and P_r will contain a 1, and we will have the following tape:

$$\begin{aligned} &Q_L P_{n+1} P_n \ldots P_r P'_{r-1} \ldots P'_1 31 A'_r 6 A_s 7 \ldots 7 A_w 7 Q_R, \quad \text{with} \\ &P'_i = 313331^{N_{im_i}} 31^{N_{im_i-1}} \ldots 331^{N_{i2}} 331^{N_{i1}}, \\ &A'_i = 1^{N_i} \end{aligned}$$

while the head will be on the last symbol of A'_r and the Turing machine will be in the state q_2. Now for the second stage we start

by going left until we find the first tape space containing 5, which will be the last symbol of P_r. We do this because of the following rules:

$$\begin{array}{cccc} q_2 & 1 & 6 & L & q_2 \\ q_2 & 3 & 4 & L & q_2 \\ q_2 & 8 & 9 & L & q_2 \\ q_2 & 5 & 1 & R & q_2 \end{array}$$

After that, we proceed to move to the right until we reach Q_R with the following rules:

$$\begin{array}{cccc} q_2 & 5 & 1 & R & q_2 \\ q_2 & 7 & 8 & R & q_2 \\ q_2 & 4 & 3 & R & q_2 \\ q_2 & 6 & 1 & R & q_2 \\ q_2 & 1 & 6 & L & q_2 \end{array}$$

So we now have the following tape state:

$$Q_L P_{n+1} P_n \dots P_{r+1} P_r^{-1} P'_{r-1} \dots P'_1 31 A'_r 1 A'_s 8 \dots 8 A'_w 86 Q_R, \quad \text{with}$$
$$P_r^{-1} = 252225^{N_{rm_r}} 225^{N_{rm_{r-1}}} \dots 225^{N_{r2}} 225^{N_{r1}-1} 1$$

and the head is over the last symbol of A'_w. What the machine does now is: it goes back to the left, looking again for the first tape space containing the symbol 5, which will be in P_r^{-1}, and then moves right to find Q_r. After N_{r1} iterations we end up with:

$$Q_L P_{n+1} P_n \dots P_{r+1} P_r^{-N_{r1}} P'_{r-1} \dots P'_1 31 A'_r 1 A'_s 8 \dots 8 A'_w 81^{N_{r1}-1} 6 Q_R, \quad \text{with}$$
$$P_r^{-N_{r1}} = 252225^{N_{rm_r}} 225^{N_{rm_{r-1}}} \dots 225^{N_{r2}} 221^{N_{r1}}$$

Now, after we move to the left to reach for $P^{-N_{r1}}$ we will find 22 written on the tape. The Turing machine will now write the symbol 9 on the first space of Q_R using the following rules:

$$\begin{array}{cccc} q_2 & 2 & 4 & L & q_3 \\ q_3 & 2 & 9 & R & q_1 \\ q_1 & 4 & 3 & R & q_1 \\ q_1 & 6 & 8 & R & q_1 \\ q_1 & 9 & 1 & R & q_1 \\ q_1 & 1 & 9 & L & q_1 \end{array}$$

and the tape will look as follows:

$Q_L P_{n+1} P_n \dots P_{r+1} P_r^{*-N_{r1}} P''_{r-1} \dots P''_1 38 A''_r 8 A''_s 1 \dots 1 A''_w 1 A''_{r1} 9 Q_R$, with
$P_r^{*-N_{r1}} = 252225^{N_{rm_r}} 225^{N_{rm_{r-1}}} \dots 225^{N_{r2}} 938^{N_{r1}} 9$,
$P''_i = 383338^{N_{im_i}} 338^{N_{im_i-1}} \dots 338^{N_{i2}} 338^{N_{i1}}$,
$A''_i = 8^{N_i}$

After this, the Turing machine will move to the left, restoring the part S of the tape, until it reaches the symbol 9 in $P^{*-N_{r1}}$. It will do this using the following rules:

$$\begin{array}{cccc} q_1 & 8 & 5 & L & q_3 \\ q_3 & 8 & 5 & L & q_3 \\ q_3 & 1 & 7 & L & q_3 \\ q_3 & 3 & 4 & L & q_3 \end{array}$$

The tape now looks like this:

$Q_L P_{n+1} P_n \dots P_r^{**} P'''_{r-1} \dots P'_1 4555 \dots 5557 A_t \dots 7 A_w 7 A_{r1} 9 Q_R$, with
$P'''_i = 454445^{N_{im_i}} 445^{N_{im_i-1}} \dots 445^{N_{i2}} 445^{N_{i1}}$,
$P_r^{**} = 252225^{N_{rm_r}} 225^{N_{rm_{r-1}}} \dots 225^{N_{r2}} 945^{N_{r1}}$

with the head being over the symbol 9 on P^{**}. We use the rule

$$\begin{array}{cccc} q_3 & 9 & 4 & L & q_2 \end{array}$$

and now we keep going for every sequence of 5 belonging to P_r, adding A_{rn} in each n-th iteration. Finally, the tape reaches the sequence 52, with that 2 being the first symbol of P_r and that 5 being the last symbol of P_{r+1}. That shows that every symbol of α_i has been written and the machine now enters into the third stage, where it restores the tape in P. The rules that show how this is done are the following:

$$\begin{array}{cccc} q_2 & 2 & 4 & L & q_3 \\ q_3 & 5 & 5 & R & q_3 \\ q_3 & 4 & 2 & R & q_3 \end{array}$$

Finally we arrive at the symbol 7 before A_t and the simulation begins again with the next symbol on β:

$$q_3 \quad 7 \quad 1 \quad R \quad q_1$$

After this, the tape will look as follows:

$$Q_L P 25111 \ldots 111 A_t 7 \ldots 7 A_w 9 Q_R$$

which correctly maps the next step of the 2-tag system. The list of 1 between P and A will not matter as the machine in the state q_1 will replace them with 9 while moving to the left without stopping until the first occurrence of the symbol 2. In the same vein, the occurrence of the symbol 9 at the of A will not matter, as q_2 treats the symbol in the same way as it treats the symbol 7.

Finally, we have to address what happens if the symbol a_{n+1} is read. As the Turing machine in the state q_2 reaches $P_{n+1} = 62$ we have the following rules:

$$\begin{array}{llll} q_2 & 2 & 4 & L & q_3 \\ q_3 & 6 & \texttt{HALT} & & \end{array}$$

and the Turing machine will halt, simulating correctly the behaviour of the original 2-tag system [7]. □

CHAPTER 3
First Proof of the Turing Completeness of Markets

3.1 Markets as Formal Objects

3.1.1 Formal Definitions

From now on, given the tuple T, $T(\tau)$ refers to the τ component of T.

Definition 3.1. Given a set of **traders** ($\mathcal{T}$), a **transactional offer** O is defined as a tuple $(T, a, \texttt{type})$ with $T \in \mathcal{T}, a \in \mathbb{Q}^+$(**the amount**) and $\texttt{Type} \in \{\texttt{BUY}, \texttt{SELL}\}$. If $\texttt{type} = \texttt{BUY}$ we call it **buy offer**, while if $\texttt{type} = \texttt{SELL}$ we call it **sell offer**.

We say that a transactional offer O_1 is **larger** than another offer O_2 if $O_1(a) > O_2(a)$. Analogously, we say that it is **smaller** if $O_1(a) < O_2(a)$.

Our definition of market will be based on the idea of transactional offers. Traders place offers, both to sell and to buy the stock. The price of the stock will be the amount of the last transactional offer fulfilled, like in stock markets. This characteristics let us model a market from the ground up, using the building blocks of real markets. Traders will base their decision on both the amount of transactional offers of each type that were fulfilled in some time period and the most significant digit of the amount of the last fulfilled offer,

which is unrealistic but possible under real stock markets rules. Besides that, we use work shifts (shifts from now on) to group traders that trade at the same time of the day, allowing us to have more flexibility on how we define these traders, while also mimicking real world economics, where traders from different countries trade at different times. Finally, we will also employ a memory, which gives us the amount of already fulfilled transactions that the traders will take into account.

Definition 3.2. A **market** with n shifts and memory k, $n, k \in \mathbb{N}$, $\mathcal{M}_{n,k}$ is defined as a 6-tuple $(\mathcal{T}, \mathcal{O}, \Delta, \texttt{next}, C, S)$ where:

- $\mathcal{T}$ is a **set of traders**. Each trader T is defined as a 4-tuple (S_B, S_S, P, σ). where $S_B, S_S \subseteq \{0, \ldots, k\}$ dictate when does the trader put or fulfill an offer, $P : \{0, \ldots, 9\} \to \{1, \ldots, 9\} \cup \texttt{HALT}$ dictates the amount of the offers they put up, depending on the amount of the last fulfilled offer and $\sigma \in \{1, \ldots, n\}$ shows they shift that they belong in.

- $\mathcal{O}$ is a **set of transactional offers**.

- Δ contains the information belonging to the last k **transactions** made and is a pair (γ, α), where $\gamma \in \{\texttt{BUY}, \texttt{SELL}\}^k$ shows the type of the last k fulfilled offers and $\alpha \in \{0, \ldots, 9\}$ shows the most significant digit of the amount of the last fulfilled offer.

- `next` dictates which kind of offer will be put in which shift, and is defined as a n-tuple of mappings $\nu_1 \ldots \nu_n$ such that: $\nu_i : \{0, \ldots, 9\} \to \{\texttt{BUY}, \texttt{SELL}\}$.

- C shows how shifts change. It is defined as a n-tuple of mappings $c_1, \ldots, c_n$ such that $c_i : \{0, \ldots, 9\} \to \{1, \ldots, n\}$.

- $S \in \{1, \ldots, n\}$ denotes the shift that we are currently in.

With the following properties:

- Two traders in the same shift cannot act at the same time:

$$\begin{aligned} T_1, T_2 \in \mathcal{T} \wedge T_1(\sigma) = T_2(\sigma) \wedge \\ (\neg(T_1(S_B) \cap T_2(S_B) = \emptyset) \vee \\ \neg(T_1(S_S) \cap T_2(S_S) = \emptyset)) \\ \implies T_1 = T_2 \end{aligned}$$

- Some trader must act at every instance:

$$\forall m \in \{1, \ldots, n\} \bigcup_{T \in \mathcal{T}: T(\sigma) = m} T(S_B) = \{0, 1, 2, 3, 4\}$$
$$\forall m \in \{1, \ldots, n\} \bigcup_{T \in \mathcal{T}: T(\sigma) = m} T(S_S) = \{0, 1, 2, 3, 4\}$$

- The buy offer with the largest amount cannot have an amount larger than the amount of the sell offer with the smallest amount:

$$\begin{aligned} \mathcal{B} := \max_a \{O \in \mathcal{O}, O(\texttt{type}) = \texttt{BUY}\} \wedge \\ \mathcal{S} := \min_a \{O \in \mathcal{O}, O(\texttt{type}) = \texttt{SELL}\} \\ \implies \mathcal{B} < \mathcal{S} \end{aligned}$$

Traders belonging to $\mathcal{T}$ have a set S_B, their strategy to buy, which dictates how many of the last k transactions must be a buy offer fulfilled for them to put or fulfill a buy offer, and a set S_S, their strategy to sell, which dictates how many of the last k transactions must be a sell offer fulfilled for them to put or fulfill a sell offer. Additionally, they have a mapping P from integers 0 to 9 to naturals 1 to 9, which maps the most significant digit of the amount of the latest transaction ($\Delta(\alpha)$) to an integer that represents the most significant digit of the amount of the next transactional offer they put. Finally they have a digit between 1 and n that shows the shift assigned to them. Let us see an example:

$$
\begin{aligned}
T_1 = (&\{2,3\},\\
&\{1,4\},\\
&\{0 \mapsto 1,\\
&\ 1 \mapsto 2,\\
&\ 2 \mapsto 3,\\
&\ 3 \mapsto 4,\\
&\ 4 \mapsto 5,\\
&\ 5 \mapsto 6,\\
&\ 6 \mapsto 7,\\
&\ 7 \mapsto 8,\\
&\ 8 \mapsto 9,\\
&\ 9 \mapsto 9\},\\
&\ 1)
\end{aligned}
$$

This trader, T_1, puts and fulfills buy offers only when the last 2 or 3 out of k transactions were buy offers fulfilled. They put and fulfill sell offers only when the last 1 or 4 out of k transactions were sell offers fulfilled. When they add an offer, its amount will have its most significant digit equal to the corresponding one in the mapping, depending on the most significant digit of the amount of the last transactional offer fulfilled. Finally, T_1 is assigned to shift one.

The pair Δ contains the type of the last four transactional offers fulfilled and the most significant digit of the amount of the latest. For example, given $k = 4$, $\Delta = ((\texttt{BUY}, \texttt{SELL}, \texttt{SELL}, \texttt{BUY}), 6)$ shows that there were two sell offers and two buy offers fulfilled and that the most significant digit of the amount of the last transactional offer fulfilled is 6. Finally the function `next` gives us the type of the next offer that will be put, depending on the most significant digit of the amount of the last transaction, $\Delta(\alpha)$.

Given a market, it will progress in the following way:

- If $\texttt{next}(\nu_S(\Delta(\alpha))) = \texttt{BUY}$ (the next offer to be placed is a buy offer):

1. Let β be the number of occurrences of `BUY` in $\Delta(\gamma)$. The only trader T_1 that satisfies that $\beta \in T_1(S_B) \wedge T_1(\sigma) = S$ will put (adding it to $\mathcal{O}$) a buy offer with an amount larger than every buy offer in $\mathcal{O}$ and whose most significant digit is $T_1(P(\Delta(\alpha)))$ if $T_1(P(\Delta(\alpha))) \neq \texttt{HALT}$. If $T_1(P(\Delta(\alpha))) = \texttt{HALT}$, then the traders will agree that the price is right (or that no one is interested in the product anymore if $\alpha = 0$) and the market will stop.
2. Let ρ be the number of occurrences of `SELL` in $\Delta(\gamma)$. The only trader T_2 that satisfies that $\rho \in T_2(S_S) \wedge T_2(\sigma) = S$ will fulfill (removing it form $\mathcal{O}$) the smallest sell offer. If the trader has to fulfill its own offer, that offer is cancelled instead.
3. The shift S changes to $c_S(\Delta(\alpha))$.
4. $\Delta(\alpha)$ gets updated to the most significant digit of a, the amount of the offer fulfilled or cancelled by T_2. In the case that no sell offer exists then $\Delta(\alpha)$ gets updated to 0.
5. $\Delta(\gamma)$ will change from $(\gamma_0, \gamma_1, \ldots, \gamma_n)$ to $(\gamma_1, \ldots, \gamma_n, \texttt{SELL})$.
6. We begin again by seeing how α has changed.

- If $\texttt{next}(\nu_S(\Delta(\alpha)) = \texttt{SELL}$ (the next offer to be placed is a sell offer):

1. Let ρ be the number of occurrences of `SELL` in $\Delta(\gamma)$. The only trader T_1 that satisfies that $\rho \in T_1(S_S) \wedge T_1(\sigma) = S$ will put (adding it to $\mathcal{O}$) a sell offer with an amount smaller than every sell offer in $\mathcal{O}$ and whose most significant digit is $T_1(P(\Delta(\alpha)))$ if $T_1(P(\Delta(\alpha))) \neq \texttt{HALT}$. If $T_1(P(\Delta(\alpha))) = \texttt{HALT}$, then the traders will agree that the price is right (or that no one is interested in the product anymore if $\alpha = 0$) and the market will stop.
2. Let β be the number of occurrences of `BUY` in $\Delta(\gamma)$. The trader T_2 that satisfies that $\beta \in T_2(S_S) \wedge T_2(\sigma) = S$ will fulfill (removing it form $\mathcal{O}$) the largest buy offer. If the trader has to fulfill its own offer, the offer is cancelled instead.
3. The shift S changes to $c_S(\Delta(\alpha))$.

4. $\Delta(\alpha)$ gets updated to the most significant digit of a, the amount of the offer fulfilled or cancelled by T_2. In the case that no buy offer exists then $\Delta(\alpha)$ gets updated to 0.
5. $\Delta(\gamma)$ will change from $(\gamma_0, \gamma_1, \ldots, \gamma_n)$ to $(\gamma_1, \ldots, \gamma_n, \texttt{BUY})$.
6. We begin again by seeing how α and S has changed.

3.1.2 Example

Let $\mathcal{M}_{2,4}$ be a market with two shifts and a memory of four defined as follows: $\mathcal{T}$ contains two traders:

$$
\begin{aligned}
T_1 = (&\{0,1,2,3,4\}, \\
&\{0,1,2,3,4\}, \\
&\{0 \mapsto \texttt{HALT}, \\
&\ 1 \mapsto 2, \\
&\ 2 \mapsto 3, \\
&\ 3 \mapsto 4, \\
&\ 4 \mapsto 5, \\
&\ 5 \mapsto 6, \\
&\ 6 \mapsto 7, \\
&\ 7 \mapsto 8, \\
&\ 8 \mapsto 9, \\
&\ 9 \mapsto 1, \}, \\
&\ 1)
\end{aligned}
\qquad
\begin{aligned}
T_2 = (&\{0,1,2,3,4\}, \\
&\{0,1,2,3,4\}, \\
&\{0 \mapsto \texttt{HALT}, \\
&\ 1 \mapsto 9, \\
&\ 2 \mapsto 8, \\
&\ 3 \mapsto 7, \\
&\ 4 \mapsto 6, \\
&\ 5 \mapsto 5, \\
&\ 6 \mapsto 4, \\
&\ 7 \mapsto 3, \\
&\ 8 \mapsto 2, \\
&\ 9 \mapsto 1\}, \\
&\ 2)
\end{aligned}
$$

$\mathcal{O}$ contains 2 offers, one sell offer and one buy offer:

$$
\begin{aligned}
O_1 &= (T_1, 9000, \texttt{BUY}) \\
O_2 &= (T_2, 80000, \texttt{SELL})
\end{aligned}
$$

The 4 last transactions and the amount of the last are:

$$\Delta = ((\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY}), 4)$$

The `next` tuple contains the following two mappings:

$$\begin{aligned}
\nu_1 :0 &\mapsto \texttt{BUY} \\
1 &\mapsto \texttt{BUY} \\
2 &\mapsto \texttt{SELL} \\
3 &\mapsto \texttt{SELL} \\
4 &\mapsto \texttt{BUY} \\
5 &\mapsto \texttt{SELL} \\
6 &\mapsto \texttt{SELL} \\
7 &\mapsto \texttt{BUY} \\
8 &\mapsto \texttt{SELL} \\
9 &\mapsto \texttt{SELL}
\end{aligned}
\qquad
\begin{aligned}
\nu_2 :0 &\mapsto \texttt{SELL} \\
1 &\mapsto \texttt{BUY} \\
2 &\mapsto \texttt{SELL} \\
3 &\mapsto \texttt{BUY} \\
4 &\mapsto \texttt{SELL} \\
5 &\mapsto \texttt{BUY} \\
6 &\mapsto \texttt{SELL} \\
7 &\mapsto \texttt{BUY} \\
8 &\mapsto \texttt{BUY} \\
9 &\mapsto \texttt{BUY}
\end{aligned}$$

Finally C has two functions, $c_1 : \{0, \ldots, 9\} \mapsto 2$ and $c_2 : \{0, \ldots, 9\} \mapsto 1$, and the initial shift S is 1. Let us see how the market will progress:

1. As $S = 1$ and $\texttt{next}(\nu_1(4) = \texttt{BUY}$, then T_1 will put a buy offer with an amount with most significant digit equal to $T_1(P(4)) = 5$ and larger than the rest of buy offers, so $O_3 = (T_1, 50000, \texttt{BUY})$. The set of offers now looks like:

$$\begin{aligned}
O_1 &= (T_1, 9000, \texttt{BUY}) \\
O_2 &= (T_2, 80000, \texttt{SELL}) \\
O_3 &= (T_1, 50000, \texttt{BUY})
\end{aligned}$$

2. Now T_1 will fulfill the smallest sell offer and the set of offers is now:

$$\begin{aligned}
O_1 &= (T_1, 9000, \texttt{BUY}) \\
O_3 &= (T_1, 50000, \texttt{BUY})
\end{aligned}$$

3. The shift S gets updated to $C(c_1(4)) = 2$.

4. $\Delta(\alpha)$ changes to 8.

5. $\Delta(\gamma)$ stays the same as it was already, $(\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY})$.

6. As $S = 2$ and $\texttt{next}(\nu_2(8) = \texttt{BUY}$, then T_2 will put a buy offer with an amount with most significant digit equal to $T_1(P(8)) = 2$ and larger than the rest of buy offers, so $O_4 = (T_2, 2000, \texttt{BUY})$. The set of offers now looks like:

$$O_1 = (T_1, 9000, \texttt{BUY})$$
$$O_3 = (T_1, 50000, \texttt{BUY})$$
$$O_4 = (T_2, 200000, \texttt{BUY})$$

7. Now T_1 has to fulfill the smallest sell offer but there is none so they do nothing.

8. The shift S gets updated to $C(c_2(8)) = 1$.

9. $\Delta(\alpha)$ changes to 0 as no offer was fulfilled.

10. $\Delta(\gamma)$ stays the same as it was already, $(\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY})$.

11. As $S = 1$ and $\texttt{next}(\nu_1(0)) = \texttt{HALT}$ the traders have agreed that they do not want to get rid of the product and thus they stop trading.

3.1.3 Why Does This Construction Model a Market?

Transactional offers are the backbone of markets. It is the way traders interact with the product, by putting offers and waiting for someone to agree on them. Using transactional offers we assure that we are approaching markets from the lowest level possible instead of trying to model price evolutions. This way, everything that can happen in a real market can be simulated by our model, with no corner cases or generalizations about price evolution.

Traders are modelled to have a strategy, buying and selling depending on the last k transactions (4 in the last example), as a way to model realistic behaviours. Traders having a shift assigned models real world limitations on time, as traders from different corners of the world do not work at the same time.

The aspects that are the most unrealistic are the dependence for everything on the most significant digit of the amount of the last transaction and traders halting their transactions all together. However, as these aspects (among many others from the model) are just restrictions, if we can prove that this model is Turing complete then any generalization will also be Turing complete, as this implementation will be a specific instance of the generalization.

3.2 The Modified Universal Turing Machine with 3 states and 9 symbols

For our proof of Turing completeness, we will use a modified version of UTM(3,9)

Definition 3.3. The Turing machine UTM$(3, 9)^*$ is defined as UTM(3,9) with the blank symbol, 1, being expanded to two symbols, 0 and 1. All transition rules that write a blank symbol will write 1 and the transitions

$$\begin{array}{cccccc} q_1 & 0 & 8 & L & q_1 \\ q_2 & 0 & 6 & L & q_2 \\ q_3 & 0 & 7 & L & q_3 \end{array}$$

will be added to the machine.

The symbol 0 will be used only at the ends of the tape (from both sides). Given the tape

$$Q_L T Q_R$$

where Q_L and Q_R are the infinite parts consisting only of blank symbols and T is the relevant part of the tape, 0 will be used as the blank symbol in Q_L and Q_R and 1 will be used as the blank symbol in T.

3.3 Proving Turing Completeness

In order to prove that a market is able to simulate this Turing Machine we have to show that it simulates correctly several things.

First, the tape and the notion of adjacency between spaces. Second, the head and its movements. Third, the controller. And finally, the states of the machine. We will do this by defining a market with three shifts and a memory of four, $\mathcal{M}_{3,4}$. Each shift will have four traders with the same mapping P (totalling to 12 traders), modelling the controler of the UTM.

3.3.1 Tape

The spaces of the tape that are not under the head will be represented by $\mathcal{O}$ (the set of offers) at the step of $\mathcal{M}_{3,4}$ before checking α (the most significant digit of the last fulfilled offer) and S (the current shift). Those spaces and its symbols will be represented as follows:

- Spaces to the left of the head are given by buy offers, the larger the amount, the closer to the head, with the largest buy offer being the space adjacent to the head from the right. The most significant digit of the amount will tell us the symbol of the tape space.

- Spaces to the right are given as sell offers, following the same logic with a little change. The smallest sell offer will be the closest to the head from the right, and every sell offer from that position to the right will be larger than the previous one.

Finally, the reason for expanding the blank symbol, as explained before in section 3.2 becomes clear. We cannot have transactions with an amount equal to 0, so we choose to use 1 to represent the empty symbol in the middle of the tape. That way, we can construct a transactional offer with an amount that has 1 as its most significant digit. We can still find the symbol 0 on the tape, but that means that all the symbols in one direction from that point are 0. We understand this as no more buy offers (if that direction is the left) or sell offers (if it is the right). One could think that only using 1 as the symbol for empty space would work just as good as having 0 and 1, but as the tape initializes, it has infinite empty spaces to the left and to the right, which means that, if only 1 was used, we

would have infinite transactional offers and that is not realistic at all.

3.3.2 Head, Rewriting and Movement

The symbol under the head is given by $\Delta(\alpha)$. Rewriting is done through the P mappings of the trader. If reading symbol s makes the Turing machine write the symbol s', then, for the trader T which puts an offer at that time $P(s) = s'$.

The movement is given by the set of mappings `next`. If the machine is in the state q_n and is reading symbol s such that the instruction is to move left, then $\texttt{next}(\nu_n(s)) = \texttt{SELL}$, as adding one sell offer smaller than the rest will make the n-th smallest sell offer become the $n+1$-th smallest (as the n-th further right space turns into the $n+1$-th further right). Analogously, if the instruction is to move right, then $\texttt{next}(\nu_n(s)) = \texttt{BUY}$.

3.3.3 States

The state of the Turing machine is modelled by S. Transitions between state are modelled by C, as given the state q_n, if reading the symbol s makes the machine move to the state q_m then $c_n(s) = m$.

3.3.4 Controller

Control instructions in a Turing machine are represented by a table of conditional statements of the form "if the machine is in state q_n, and the last read cell is symbol s, then write the symbol s', transition to q_n (n and m can be equal) and move left/right." We will simulate this with a table for each state that captures the traders function P, $\Delta(\alpha)$, the values of c_n and the values of ν_n.

3.3.5 Formal Definition

To formally construct $\mathcal{M}_{3,4}$ we have to define each of the components that make it. We start by defining $S = 1$, as UTM(3,9)* starts in q_1. We continue formally defining P and σ for every trader,

c_1, c_2, c_3, and ν_1, ν_2, ν_3. For each state we will have four traders (being the four traders that have the same shift) that have the same policy. These are T_{A1}, T_{B1}, T_{C1} and T_{D1} for q_1 (so all of them have $\sigma = 1$), T_{A2}, T_{B2}, T_{C2} and T_{D2} for q_2 (so all of them have $\sigma = 2$) and T_{A3}, T_{B3}, T_{C3} and T_{D3} for q_3 (so all of them have $\sigma = 3$). The table for q_1 is 3.1, for q_2 is 3.2 and for q_3 3.3.

Inst.	A	$T(P(A))$	$C(c_1(A))$	$\texttt{next}(\nu_1(A))$
0/8 Lq_1	0	8	1	SELL
1/8 Lq_1	1	8	1	SELL
2/3 Rq_1	2	3	1	BUY
3/2 Lq_1	3	2	1	SELL
4/3 Rq_1	4	3	1	BUY
5/9 Lq_1	5	9	1	SELL
6/8 Rq_1	6	8	1	BUY
7/6 Lq_2	7	6	2	SELL
8/5 Lq_3	8	5	3	SELL
9/1 Lq_1	9	1	1	SELL

Table 3.1: Translation of UTM$(3, 9)^*$ for $\mathcal{M}_{3,4}$ for q_1

Inst.	A	$T(P(A))$	$C(c_2(A))$	$\texttt{next}(\nu_2(A))$
0/6 Lq_2	0	6	2	SELL
1/6 Lq_2	1	6	2	SELL
2/4 Lq_3	2	4	3	SELL
3/4 Lq_2	3	4	2	SELL
4/3 Rq_2	4	3	2	BUY
5/6 Rq_2	5	6	2	BUY
6/1 Rq_2	6	1	2	BUY
7/8 Rq_2	7	8	2	BUY
8/9 Lq_2	8	9	2	SELL
9/8 Rq_2	9	8	2	BUY

Table 3.2: Translation of UTM$(3, 9)^*$ for $\mathcal{M}_{3,4}$ for q_2

Inst.	A	$T(P(A))$	$C(c_2(A))$	$\texttt{next}(\nu_2(A))$
0/7 Lq_3	0	7	3	SELL
1/7 Lq_3	1	7	3	SELL
2/9 Rq_1	2	9	1	BUY
3/4 Lq_3	3	4	3	SELL
4/2 Rq_3	4	2	3	BUY
5/5 Rq_3	5	5	3	BUY
6/HALT	6	HALT	X	X
7/5 Rq_1	7	5	1	BUY
8/5 Lq_3	8	5	3	SELL
9/4 Lq_2	9	4	2	SELL

Table 3.3: Translation of UTM$(3, 9)^*$ for $\mathcal{M}_{3,4}$ for q_3

Now, we have to define S_B and S_S for all the traders:

- $T_{A1}(S_B) = T_{A2}(S_B) = T_{A3}(S_B) = \{0, 1\}$ and $T_{A1}(S_S) = T_{A2}(S_S) = T_{A3}(S_S) = \{0, 1\}$. These traders are market contrarians who will sell when the market goes up and will buy when the market goes down.

- $T_{B1}(S_B) = T_{B2}(S_B) = T_{B3}(S_B) = \{3, 4\}$ and $T_{B1}(S_S) = T_{B2}(S_S) = T_{B3}(S_S) = \{3, 4\}$. These traders are market followers who will sell when the market goes down and will buy when the market goes up.

- $T_{C1}(S_B) = T_{C2}(S_B) = T_{C3}(S_B) = \{2\}$ and $T_{C1}(S_S) = T_{C2}(S_S) = T_{C3}(S_S) = \emptyset$. These traders will buy when they want and sell when the market wants to buy, holding the product for longer.

- $T_{D1}(S_B) = T_{D2}(S_B) = T_{D3}(S_B) = \emptyset$ and $T_{C1}(S_S) = T_{C2}(S_S) = T_{C3}(S_S) = \{2\}$. These traders will sell when they want and buy when the market wants to sell, holding the product for less time.

Now, the only parts of the construction that are not defined are Δ at the beginning and $\mathcal{O}$ at the beginning. We define Δ at the beginning as:

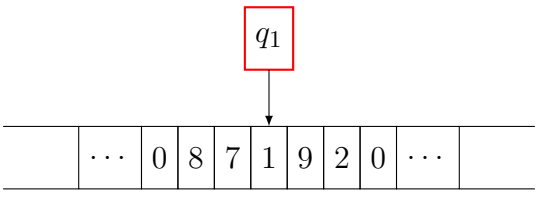
$$\Delta = ((\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY}), a),$$

where a is the symbol under the head of the machine at the initial state. We define the tuple γ this way as a convention, but we can define it however we like, as it its values are unimportant for simulating the Turing Machine, it only adds realism to the market model.

Finally for $\mathcal{O}$ we initialize it containing a number of sell offers equal to the number of spaces to the right of the head until we reach the infinite amount of spaces containing the symbol 0. These sell offers will have amounts that ensure that the smallest one has the most significant digit of its amount equal to the symbol on the first space to the right of the tape, the second smallest has it equal to the symbol on the second space to the right of the tape etc. The set of offers $\mathcal{O}$ will also contain a number of buy offers equal to the number of spaces to the left of the head until we reach the infinite amount of spaces containing the symbol 0. These buy offers will ensure that the largest buy offer has the most significant digit of its amount equal to the first space to the left of the tape, the second largest buy offer has the most significant digit of its amount equal to the second space to the left of the tape, etc. As a convention, we will assign sequentially traders to the offers, from left to right. However, which trader has what starting offer is not really relevant.

3.3.6 Example

Suppose we have the following starting tape for UTM(3,9)*:

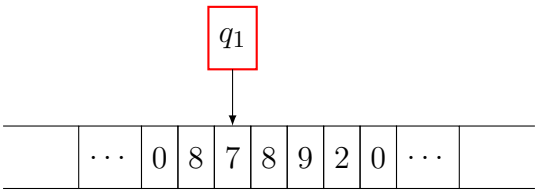


This gives us two aspects of our market: the set of offers, $\mathcal{O}$, and $\Delta(\alpha)$. For the latter, as the symbol under the header is 1, $\Delta(\alpha) = 1$.

For the former we have:

$$(T_{A1}, 80, \texttt{BUY})$$
$$(T_{A2}, 700, \texttt{BUY})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

After the first step of the Turing Machine we have:



Let us see the first step of the market:

1. As $\texttt{next}(\nu_1(1)) = \texttt{SELL}$, the number of occurrences of SELL in $\Delta(\gamma)$ is 0 and $0 \in T_{A1}(S_S)$, T_{A1} puts a sell offer with an amount with most significant digit equal to $T_{A1}(P(1)) = 8$. The set of offers now looks like this:

$$(T_{A1}, 80, \texttt{BUY})$$
$$(T_{A2}, 700, \texttt{BUY})$$
$$(T_{A1}, 8000, \texttt{SELL})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

2. As the number of occurrences of BUY in $\Delta(\gamma)$ is 0, and $0 \in T_{A1}(S_B)$ then T_{A1} fulfills the largest buy offer. The set of offers now looks like this:

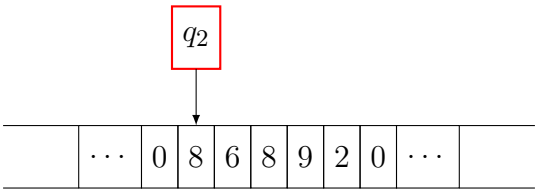
$$(T_{A1}, 80, \texttt{BUY})$$
$$(T_{A1}, 8000, \texttt{SELL})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

3. The shift stays at 1 as $c_1(1) = 1$.

4. $\Delta(\alpha)$ changes to 7, the most significant digit of the amount of the last transaction.

5. $\Delta(\gamma)$ stays $(\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY})$.

As we see, the simulation has stayed as expected to simulate the TM through the first step. The set of offers still models the tape and $\Delta(\alpha)$ is equal to the symbol under the head. Let us see what happens on the second step of the Turing Machine:



And the market evolution:

1. As $\texttt{next}(\nu_1(7)) = \texttt{SELL}$, the number of occurrences of SELL in $\Delta(\gamma)$ is 0 and $0 \in T_{A1}(S_S)$, T_{A1} puts a sell offer with an amount with most significant digit equal to $T_{A1}(P(7)) = 6$. The set of offers now looks like this:

$$(T_{A1}, 80, \texttt{BUY})$$
$$(T_{A1}, 600, \texttt{SELL})$$
$$(T_{A1}, 8000, \texttt{SELL})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

2. As the number of occurrences of BUY in $\Delta(\gamma)$ is 0, and $0 \in T_{A1}(S_B)$ then T_{A1} fulfills the largest buy offer. The set of offers now looks like this:

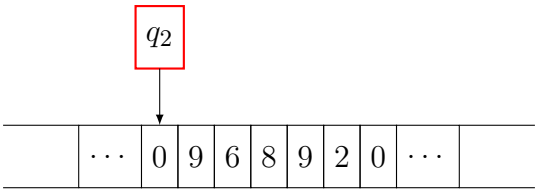
$$(T_{A2}, 600, \texttt{SELL})$$
$$(T_{A1}, 8000, \texttt{SELL})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

3. The shift changes to 2 as $c_1(7) = 2$.

4. $\Delta(\alpha)$ changes to 8, the most significant digit of the amount of the last transaction.

5. $\Delta(\gamma)$ stays $(\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY})$.

After this step, the tape and the symbol under the head keep being correct. This step also shows that the shift S models correctly the state of the machine. Let us see the next step of the Turing Machine:



And the next step of the market simulation:

1. As $\texttt{next}(\nu_2(8)) = \texttt{SELL}$, the number of occurrences of SELL in $\Delta(\gamma)$ is 0 and $0 \in T_{A2}(S_S)$, T_{A2} puts a sell offer with an amount with most significant digit equal to $T_{A2}(P(8)) = 9$. The set of offers now looks like this:

$$(T_{A2}, 90, \texttt{SELL})$$
$$(T_{A1}, 600, \texttt{SELL})$$
$$(T_{A1}, 8000, \texttt{SELL})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

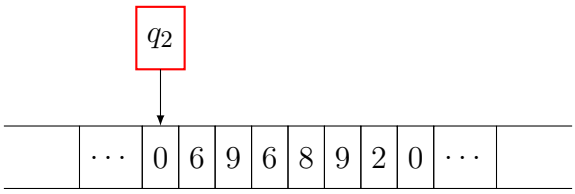
2. As there are no buy offers left, no transaction is done.

3. The shift stays as 2 as $c_2(8) = 2$.

4. $\Delta(\alpha)$ changes to 0, as no transaction was done.

5. $\Delta(\gamma)$ stays $(\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY})$.

With this step, we are able to see that the simulation handles having 0 as a symbol on the tape. It still correctly models both the tape, the head and the state. Let us do a final step of the Turing Machine:



1. As $\texttt{next}(\nu_2(0)) = \texttt{SELL}$, the number of occurrences of SELL in $\Delta(\gamma)$ is 0 and $0 \in T_{A2}(S_S)$, T_{A2} puts a sell offer with an amount with most significant digit equal to $T_{A2}(P(0)) = 6$. The set of offers now looks like this:

$$(T_{A2}, 60, \texttt{SELL})$$
$$(T_{A2}, 90, \texttt{SELL})$$
$$(T_{A1}, 600, \texttt{SELL})$$
$$(T_{A1}, 8000, \texttt{SELL})$$
$$(T_{A3}, 9000, \texttt{SELL})$$
$$(T_{A3}, 20000, \texttt{SELL})$$

2. As there are no buy offers left, no transaction is done.

3. The shift stays as 2 as $c_2(0) = 2$.

4. $\Delta(\alpha)$ changes to 0, as no transaction was done.

5. $\Delta(\gamma)$ stays $(\texttt{BUY}, \texttt{BUY}, \texttt{BUY}, \texttt{BUY})$.

As we see, both the Turing machine and the market have entered in an infinite loop, where the machine reads 0, writes 6 and moves to the left (just to find another 0) and the market has the most significant digit of the last transaction as 0, so a trader adds a sell offer with the most significant digit of its amount equal to 6 and tries to fulfill a buy offer when no buy offers are set (keeping the most significant digit of the last transaction to 0).

CHAPTER 4
Second Proof of the Turing Completeness of Markets

For this second proof we will rewrite the definition of market, getting rid of transactional offers. This definition aims to model a more realistic behaviour from the traders at the cost of taking a more high level view of stock markets, dropping the fine grain approach that the first proof had.

4.1 Markets as Formal Objects

4.1.1 Formal Definitions

This new definition of the market will take into account the amount of the stock that is bought and the price will vary in function of it according to supply and demand laws. Traders will make decisions depending on the last k market capitalization (i.e. the total value of the company shares, from now on market caps) of the stock and the last k volumes of stock that traders wanted to sell (e.g. if we have two traders T_1 and T_2 and T_1 tries to buy x amounts of stock while T_2 tries to sell y, we will take y into account, instead of the total amount of shares that was sold which is $\min(x, y)$), which is a more realistic approach than looking at the most significant digit of the price of the last transactional offer.

Definition 4.1. A **market** with memory $k \in \mathbb{N}$, $\mathcal{M}_k$ is defined as a 3-tuple $(P, V, \mathcal{T})$ where:

- C is the memory of the **market cap** of the stock. It is defined as a k-tuple $C = (c_1, \ldots, c_k)$, where $c_n \in \mathbb{Q}^+$ represent the market cap n iterations ago.

- V is the memory of the **volumes of stock** (measured in currency) that were tried to be sold. It is defined as a k-tuple $V = (v_1, \ldots, v_k)$, where $v_n \in \mathbb{Q}^+$ represent the volume of the stock that was tried to be sold n transactions ago.

- $\mathcal{T}$ is a set of **traders**. Each trader T is defined as a set of pairs of the form $T = \{(S_1, O_1), \ldots, (S_n, O_n)\}$, where each S_i is a logical predicate of the form $\sigma_{i1} \wedge \sigma_{i2} \wedge \cdots \wedge \sigma_{im}$, representing the strategy (i.e. what has to happen for the trader to act). Each σ is a predicate build with comparison operators, numerical expressions built by the operators $+, -, *, /, \texttt{DIV}$ (with `DIV` being the integer division), where the atomic expressions are constants and the elements of C and V. Finally, each O_i is a pair $(\texttt{type}, \nu)$ representing the outcome of the strategy (i.e. what happens if S_i holds), where $\texttt{type} \in \{\texttt{BUY}, \texttt{SELL}\}$ represents if the trader is going to buy or sell and $\nu : \mathbb{Q}^{2k} \to \mathbb{Q}^+$ represents how much as a function of the values of the elements of C and V.

Where we assume that a trader cannot have two outcomes for the same strategy nor strategies that are generalizations of others:

$$\forall T \in \mathcal{T}, \neg\exists (S_i, O_i), (S_j, O_j) \in T : S_i \implies S_j$$

In this model, traders buy and sell according to strategies that are conjunctions of comparisons of the last k market caps and amounts of stock that traders tried to sell. They associate a pair to each of these conjunctions consisting of the type of action that they are taking and the amount of stock that they want to buy or sell. For example, let us assume that a trader T wants to buy twice the amount of stock that traders wanted to sell in the last iteration when the market cap rises through the last 3 iterations (which means that the price of the stock has risen through the last 3 iterations), then T will contain a pair $(c_1 > c_2 \wedge c_2 > c_3 \wedge c_3 > c_4, (\texttt{BUY}, 2v_1))$.

This model advances by checking at each step which strategies hold (i.e. for every $T \in \mathcal{T}$, which S such that $(S, O) \in T$ holds, i.e. $S \equiv \top$), adding the outcomes with the same type among all traders and updating the market cap following a simple supply and demand law. This process continues until no strategy holds, which means that traders do not want to buy or sell anymore and the market becomes stable. We formalize this mechanics as follows:

1. Let $\mathcal{O}$ be the multiset (as several traders may have the same outcome for the same strategy) of actions taken this iteration. It is defined as:

$$\mathcal{O} = \{O | (S, O) \in T, T \in \mathcal{T} : S \equiv \top\}$$

2. Let $\mathcal{O}_B$ and $\mathcal{O}_S$ be two multisets such that:

$$\mathcal{O}_B = \{\nu | (\texttt{BUY}, \nu) \in \mathcal{O}\},$$
$$\mathcal{O}_S = \{\nu | (\texttt{BUY}, \nu) \in \mathcal{O}\}$$

3. Let A_B and A_S be the total amount of stock (in currency) that the traders are trying to buy and sell (respectively):

$$A_B = \sum_{\nu \in \mathcal{O}_B} \nu(c_1, \ldots, c_k, v_1, \ldots, v_k),$$
$$A_S = \min(\sum_{\nu \in \mathcal{O}_S} \nu(c_1, \ldots, c_k, v_1, \ldots, v_k), c_1)$$

   This definition for A_S comes from the fact that it is impossible to try to sell a volume in stock higher than the market capitalization.

4. The market cap updates:

$$c_n := c_{n-1} \quad \forall n \leq k,$$
$$c_1 := c_1 + A_B - A_S$$

5. The volume of stock that traders tried to sell updates:

$$v_n := v_{n-1} \quad \forall n \leq k,$$
$$v_1 := A_S$$

4.1.2 Example

Let us take the following market with memory $k = 3$ and the following parameters:

$$\begin{aligned}
C &= (c_1 = 30, c_2 = 20, c_3 = 25),\\
V &= (v_1 = 5, v_2 = 10, v_3 = 15),\\
\mathcal{T} &= \{T_1 = ((c_1 < c_2), (\texttt{SELL}, c_1/3),\\
&\qquad (c_1 > c_2), (\texttt{BUY}, c_1/3)),\\
&\qquad T_2 = ((c_1 < c_2 \wedge c_2 < c_3), (\texttt{SELL}, c_1/2),\\
&\qquad (c_1 > c_2 \wedge c_2 < c_3), (\texttt{SELL}, c_1/4),\\
&\qquad (c_1 < c_2 \wedge c_2 > c_3), (\texttt{BUY}, c_1/4),\\
&\qquad (c_1 > c_2 \wedge c_2 > c_3), (\texttt{BUY}, c_1/2))\}
\end{aligned}$$

Now, let us see how the market evolves:

1. As $c_1 > c_2$ and $c_2 < c_3$ the outcome for T_1 is $(\texttt{BUY}, c_1/2)$, while the outcome for T_2 is $(\texttt{SELL}, c_1/4)$.

2. The volume of stock that traders are trying to buy is $c_1/2 = 15$, while the volume of stock that traders are trying to sell is $c_1/4 = 7.5$. As such we update the market cap with the updated c_1 being $c_1 = c_1 + 15 - 7.5 = 37.5$:

$$C = (c_1 = 37.5, c_2 = 30, c_3 = 25)$$

3. Now we update V:

$$V = (v_1 = 7.5, v_2 = 5, v_3 = 10)$$

4. Now, since $c_1 > c_2$ and $c_2 > c_3$, both traders will try to buy stock while neither try to sell. Thus, the market cap will keep growing and it will never stabilize.

4.1.3 Why Does This Construction Model a Market?

Traders basing their decisions on market cap is equivalent to them basing their decisions on stock prices, as the stock price is just the

market cap divided by the amount of stock, which is public knowledge in the real world (we can ignore it in our formalization as the market cap provides enough information). This, together with the volume of stock traded, is the basis of technical analysis [9], a very popular way of trading. Evidence of technical analysis providing any benefit is dubious at best and mostly based on traders being irrational [10][11][12], but that does not matter in order to model a realistic market, as we just to model realistic trader behaviour.

However, this model has some problems of its own. Firstly, the evolution of the market cap does not correspond to the actual behaviour of real markets, settling for loosely following them. While it is true that, assuming a fixed amount of stock, the market cap of a stock rises if the total amount of stock that traders are trying to buy is larger than the total amount of stock that traders are trying to sell (and the market cap diminishes if the opposite situation happens), the changes this law incites are more complex than those that our model simulates [13][14].

Secondly, this model does not take into account the amount of stock that each trader possesses. This makes for traders to just sell without ever buying, and, although shorting can explain part of it, there are no limitations that prevent a trader from selling an amount equal to the entire market cap in one iteration and then doing it again in the following one. This, obviously, is a pretty unrealistic situation that is contemplated in our model.

Finally, traders in this model can use the integer division as part of their strategies. Even though traders that follow technical analysis use pretty esoteric concepts in order to make decisions (such as the Fibonacci numbers), using the integer division as a decision making tool is not something that is widely used.

4.2 Proving Turing Completeness

4.2.1 General Ideas

For this model, we are going to prove its Turing completeness by showing that it can simulate any semi-finite 3-symbol $(0, 1, \Sigma$, with Σ being the blank space symbol for the infinite spaces outside the written tape and 0 being the blank space symbol for the spaces that are inside the written part of the tape) Turing machine that never crosses the end line, a known Turing complete system. We will do that by giving a constructive algorithm that creates a market with a set of traders whose strategies are able to simulate the given Turing machine.

The market will start with the last market cap (c_1) having its binary representation sans the most significant bit equal to the initial state of the tape. For example, given a starting tape 101, the last market cap will be 13. The last volume of stock that traders tried to sell (v_1) will be of the form 2^n, where n is equal to the number of tape spaces that are between the first one and the one under the head of the machine (e.g. if the head is over the first tape space then $v_1 = 2^0$, if it is over the second one then $v_1 = 2^1$ etc.). Each time that $c_1 = c_2$ the market will have modeled a transition of the Turing machine, and thus we can check c_1 and v_1 for information about the machine.

To model the states of the Turing machine we are going to use a sequence of market caps, and thus we need to decide the memory of the market. We will associate a code in binary to each state of the machine, using the smallest amount of bits possible (e.g. if the Turing machine has 4 states we will use 2 bits). We will write this code in the market caps associating increases in the market cap to 1 and decreases to 0, so if the state associated with the code 01 is written in the market caps c_3 and c_4, then $c_4 < c_5$ (0) and $c_3 > c_4$ (1). The market will then have a memory of $k = 2(n + 2)$, where n is the amount of bits necessary to assign a code to each state of the Turing machine. Besides the space for the code of the state, we need two extra memory spaces, one to show the machine tape state after

completing a transition and another one that repeats that value in order to show where the code for a state starts (akin to how DNA has a starting codon that shows where the chain starts). Finally we need twice that amount of memory as we need to know what is the code for the state where the transition in the Turing machine began.

Finally the controller of the machine will be represented on the strategies of the traders, which will write the code of the state that the machine arrives in after a transition depending on the original state, the tape state and the tape space under the head (all of this information is either on the market caps or on the volume of stock tried to be sold).

4.2.2 The Algorithm

We shall now give an algorithm that builds the traders required for simulating the Turing machine. We start by letting n be the number of bits required to write the code of a state. Assuming that we just finished simulating a transition of the Turing machine, we have $v_1 = 2^m$ where m denotes the position of the head of the Turing machine as explained before, $c_1 = c_2 = R$ where R represents the current tape state as we explained and $c_3, \ldots, c_{2+n}$ represent the state that we are in. This algorithm will output two traders, T_1 and T_2, who buy and sell (alternating each other) in such a way that the market simulates the behaviour and transitions.

Before delving into the formalisms of the algorithm, we should explain some key insights that will help the reader understand what is going on:

- **Reading a state code:** Every iteration of the market relies on traders reading the state coded as fluctuations of the market cap. Not only which state is coded but where is it coded is important. This comes from the fact that if the current state is coded from c_{3+n} to c_3, the outcomes of the traders' strategies must differ from the outcomes of the strategies that hold if the state is coded from c_{4+n} to c_4 in order to correctly code the next state of the Turing machine. We know where the codification

of the state begins in the market caps by checking where two consecutive market caps are equal, and as such every strategy depends on knowing where do these market caps reside.

- **Writing a state code:** A market needs n iterations to properly codify the state that the Turing machine is in. Each iteration will increase or decrease the market cap by $R/(ni)$, where R codifies the tape state at the moment that the transition happens in the Turing machine, n corresponds to the amount of bits necessary to code every state and i corresponds to the iteration of writing the new state that we are in. We chose $R/(ni)$ in order to guarantee two things: first, that even if the state to be codified requires the market cap to decrease n times, we will not reach a market cap under 0 and second, that the market cap will not be equal to the new tape state in the last iteration before writing the new tape space as a market cap (as we use market caps being equal as a way to identify the beginning of the code of a state).

- **Reading the symbol under the head:** Let c_k and v_k codify, respectively, the tape state and the space under the head as we have explained before. As $c_k \in \mathbb{N}$ when codifying the tape state, we have $c_k = 2^\alpha + \gamma_1 2^{\alpha-1} + \cdots + \gamma_\alpha 2^0$ with $k \in \mathbb{N}$ and $\forall i, \gamma_i \in \{0, 1\}$ meaning that the tape state is $\gamma_\alpha \dots \gamma_1$. We have three cases:

  - **The symbol under the head is Σ:** Then, the head must be α spaces away from the first one, and as such $v_k = 2^\alpha$. As such $c_k \texttt{ DIV } v_k = 1$ and we can check if this is the case in the strategies using the comparison $c_k \texttt{ DIV } v_k \leq 1$.
  - **The symbol under the head is 1:** Then, as the head is less than α spaces away from the first one we have $c_k \texttt{ DIV } v_k > 1$. Additionally, if $v_k = 2^j$ then $\gamma_{\alpha-j} = 1$. It follows then that $c_k \texttt{ DIV } v_k = 2^{\alpha-j} + \gamma_1 2^{\alpha-1-j} + \cdots + \gamma_{\alpha-j-1} 2 + 1$ and $2(c_k \texttt{ DIV } 2v_k) = 2^{\alpha-j} + \gamma_1 2^{\alpha-1-j} + \cdots + \gamma_{\alpha-j-1} 2$. Thus, we can check if the symbol under the head is 1 by adding $(c_k \texttt{ DIV } v_k > 1) \wedge (c_k \texttt{ DIV } v_k > 2(c_k \texttt{ DIV } 2v_k))$ to the strategy.

  - **The symbol under the head is 0:** Following the same logic, if the symbol under the head is 0 and $v_k = 2^j$, then $\gamma_{\alpha-j} = 0$, and $c_k \texttt{ DIV } v_k = 2^{\alpha-j} + \gamma_1 2^{\alpha-1-j} + \cdots + \gamma_{\alpha-j-1}2$. We can then check if the symbol under the head is 0 by adding $(c_k \texttt{ DIV } v_k > 1) \wedge (c_k \texttt{ DIV } v_k \leq 2(c_k \texttt{ DIV } 2v_k))$ to the strategy.

- **Updating the tape:** Let us suppose that we are writing the updated tape state in this iteration and that c_k and v_k give the last tape state and the space under the head, respectively. After increasing and decreasing the market cap n times, the market cap stored in c_1 differs from c_k, so we need to take this into account to write the new tape state. We have four cases:

  - **The tape stays the same:** Then the traders just need to try to buy c_k amount of stock and try to sell c_1 amount of stock. This will result in $c_1 := c_1 + c_k - c_1 = c_k$, which is what we want as the new value of c_1.
  - **The Turing machine rewrites 0 to 1 (resp. a 1 to 0):** Besides using c_k and c_1 in the same way as above, we need to add (resp. subtract) $v_k = 2^j$ from $c_k = 2^\alpha + \gamma_1 2^{\alpha-1} + \cdots + \gamma_\alpha 2^0$, with $k \in \mathbb{N}$ and $\forall i, \gamma_i \in \{0, 1\}$ in order to change $\gamma_{\alpha-j}$ from 0 to 1 (resp. from 1 to 0). The traders will do this by trying to buy (resp. to sell) v_k stock. We end up with $c_1 := c_1 + c_k + v_k - c_1 = c_k + v_k$ (resp. $c_1 := c_1 + c_k - v_k - c_1 = c_k - v_k$).
  - **The Turing machine reads Σ and writes 1:** As we want to update the market cap to $2^{\alpha+1} + 2^\alpha + \gamma_1 2^{\alpha-1} + \cdots + \gamma_\alpha 2^0$ and we have $v_k = 2^\alpha$, besides using c_k and c_1 as the same way that we have used in the other cases, the traders will try to buy $2v_k$ amounts of stock, resulting in $c_1 := c_1 + c_k + 2v_k - c_1 = 2^{\alpha+1} + 2^\alpha + \gamma_1 2^{\alpha-1} + \cdots + \gamma_\alpha 2^0$.
  - **The Turing machine reads Σ and writes 0:** The difference from the previous case is that the the market cap that we want is $2^{\alpha+1} + \gamma_1 2^{\alpha-1} + \cdots + \gamma_\alpha 2^0$. As such, besides using c_k and c_1 as the same way that we have used in the other cases and having the traders try to buy $2v_k$ amounts of stock, the traders will also try to sell v_k

amounts of stock, resulting in $c_1 := c_1 + c_k + 2v_k - c_1 - v_k = 2^{\alpha+1} + \gamma_1 2^{\alpha-1} + \cdots + \gamma_\alpha 2^0$.

- **Moving the head:** Assuming that c_1 contains the current tape state and c_k and v_k codify the last tape state and head position respectively, if the Turing machine moves the head to the right (resp. to the left), traders will try to both buy and sell $2v_k$ (resp. $v_k/2$) volumes of stock, resulting in the market cap staying the same (which links back to having repeating market caps in order to locate the beginning of a new state codification) and v_1 updating to $2v_k$ (resp. $v_k/2$).

- **Halting the machine:** Every strategy will be exhaustive with the significant market caps (i.e. the ones that correspond to the state and the symbol under the head being read). As such, these strategies are disjointed by design. If being in state q and reading symbol σ results in the machine halting, we simply do not add a strategy for that case to the traders. As no trader will act in that situation, the market will halt as well.

We shall give the algorithm for the state with code $0_1 \dots 0_n$, assuming that after reading the symbol 0 on the tape it goes to the state with code $\alpha_1 \dots \alpha_n$, after reading the symbol 1 it goes to the state with code $\beta_1 \dots \beta_n$ and after reading the symbol Σ it goes to the state with code $\gamma_1 \dots \gamma_0$:

1. We start with the strategies and outcomes for reading a 0. Loop for $i := 1 \dots n$

   (a) We define S_{0i}:

   $$\begin{aligned} S_{0i} := \; & (c_{2+n+i} \leq c_{2+n+i+1}) \wedge (c_{2+n+i} \geq c_{2+n+i+1}) \wedge \\ & (c_{2+n+i-1} < c_{2+n+i}) \wedge (c_{2+n+i-2} < c_{2+n+i-1}) \wedge \cdots \wedge \\ & (c_{2+n+i-n} < c_{2+n+i-n+1}) \wedge (c_{2+n+i} \texttt{ DIV } v_{2+n+i} > 1) \wedge \\ & (c_i \texttt{ DIV } v_i \leq 2(c_i \texttt{ DIV } 2v_i)) \end{aligned}$$

   (b) We define O_{0iB} and O_{0iS} depending on α_i:

   - If $\alpha_i = 0$:

   $$\begin{aligned} O_{0iB} &= (\texttt{BUY}, 0), \\ O_{0iS} &= (\texttt{SELL}, c_i/(ni)) \end{aligned}$$

- If $\alpha_i = 1$:

$$O_{0iB} = (\texttt{BUY}, c_i/(ni)),$$
$$O_{0iS} = (\texttt{SELL}, 0)$$

(c) We add the pairs to the traders, alternating each iteration of the loop:

- If i is even:

$$T_1 := T_1 \cup (S_{0i}, O_{0iB}),$$
$$T_2 := T_2 \cup (S_{0i}, O_{0iS})$$

- Else:

$$T_1 := T_1 \cup (S_{0i}, O_{0iS}),$$
$$T_2 := T_2 \cup (S_{0i}, O_{0iB})$$

2. Now we must update the market cap in a way that represents the updated tape. We have two cases:

- If the transition that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 0 does not result in a change of the tape (i.e. it reads a 0 and writes a 0):

(a) We define the strategy and outcome that restores the market cap:

$$\begin{aligned} S_{0c} = \quad & (c_{n+1} \leq c_{n+2}) \wedge (c_{n+1} \geq c_{n+2}) \wedge \\ & (c_{2n+2} < c_{2n+3}) \wedge (c_{2n+1} < c_{2n+2} \wedge \cdots \wedge \\ & (c_{n+3} < c_{n+4}) \wedge (c_{n+1} \ \texttt{DIV} \ v_{n+1} > 1) \wedge \\ & (c_{n+1} \ \texttt{DIV} \ v_{n+1} \leq 2(c_{n+1} \ \texttt{DIV} \ 2v_{n+1})), \\ O_{0cB} = \quad & (\texttt{BUY}, c_{n+1}), \\ O_{0cS} = \quad & (\texttt{SELL}, c_1) \end{aligned}$$

(b) We add the pairs to the traders:

$$T_1 := T_1 \cup (S_{0c}, O_{0cB}),$$
$$T_2 := T_2 \cup (S_{0c}, O_{0cS})$$

- If the transition that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 0 results in a change of the tape (i.e. it reads a 0 and writes a 1):

  (a) We define the strategy and outcome that restores the market cap:

$$\begin{aligned} S_{0c} = \quad & (c_{n+1} \leq c_{n+2}) \wedge (c_{n+1} \geq c_{n+2}) \wedge \\ & (c_{2n+2} < c_{2n+3}) \wedge (c_{2n+1} < c_{2n+2} \wedge \cdots \wedge \\ & (c_{n+3} < c_{n+4}) \wedge (c_{n+1} \texttt{ DIV } v_{n+1} > 1) \wedge \\ & (c_{n+1} \texttt{ DIV } v_{n+1} \leq 2(c_{n+1} \texttt{ DIV } 2v_{n+1})), \\ O_{0cB} = \quad & (\texttt{BUY}, c_{n+1} + v_{n+1}), \\ O_{0cS} = \quad & (\texttt{SELL}, c_1) \end{aligned}$$

  (b) We add the pairs to the traders:

$$\begin{aligned} T_1 &:= T_1 \cup (S_{0c}, O_{0cB}), \\ T_2 &:= T_2 \cup (S_{0c}, O_{0cS}) \end{aligned}$$

3. We define the strategy and outcome that makes the volume of stock that was tried to be sold equal to 2^m where m is the position of the head of the Turing machine. We have two options:

   - If the transitions that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 0 makes the head move to the left:

$$\begin{aligned} S_{0v} = \quad & (c_{n+2} \leq c_{n+3}) \wedge (c_{n+2} \geq c_{n+3}) \wedge \\ & (c_{2n+3} < c_{2n+4}) \wedge (c_{2n+2} < c_{2n+3} \wedge \cdots \wedge \\ & (c_{n+4} < c_{n+5}) \wedge (c_{n+2} \texttt{ DIV } v_{n+2} > 1) \wedge \\ & (c_{n+2} \texttt{ DIV } v_{n+2} \leq 2(c_{n+2} \texttt{ DIV } 2v_{n+2})), \\ O_{0vB} = \quad & (\texttt{BUY}, v_{n+2}/2), \\ O_{0vS} = \quad & (\texttt{SELL}, v_{n+2}/2) \end{aligned}$$

   - If the transitions that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 0 makes the head

move to the right:

$$\begin{aligned} S_{0v} = \quad & (c_{n+2} \leq c_{n+3}) \wedge (c_{n+2} \geq c_{n+3}) \wedge \\ & (c_{2n+3} < c_{2n+4}) \wedge (c_{2n+2} < c_{2n+3} \wedge \cdots \wedge \\ & (c_{n+4} < c_{n+5}) \wedge (c_{n+2}\ \texttt{DIV}\ v_{n+2} > 1) \wedge \\ & (c_{n+2}\ \texttt{DIV}\ v_{n+2} \leq 2(c_{n+2}\ \texttt{DIV}\ 2v_{n+2})), \\ O_{0vB} = \quad & (\texttt{BUY}, 2v_{n+2}), \\ O_{0vS} = \quad & (\texttt{SELL}, 2v_{n+2}) \end{aligned}$$

4. We add the pairs to the traders:

$$\begin{aligned} T_1 &:= T_1 \cup (S_{0v}, O_{0vS}), \\ T_2 &:= T_2 \cup (S_{0v}, O_{0vB}) \end{aligned}$$

5. We continue for the strategies and outcomes for reading 1. Loop for $i := 1 \ldots n$

   (a) We define S_{1i}:

$$\begin{aligned} S_{0i} := \quad & (c_{2+n+i} \leq c_{2+n+i+1}) \wedge (c_{2+n+i} \geq c_{2+n+i+1}) \wedge \\ & (c_{2+n+i-1} < c_{2+n+i}) \wedge (c_{2+n+i-2} < c_{2+n+i-1}) \wedge \cdots \wedge \\ & (c_{2+n+i-n} < c_{2+n+i-n+1}) \wedge (c_i\ \texttt{DIV}\ v_i > 1) \wedge \\ & (c_i\ \texttt{DIV}\ v_i > 2(c_i\ \texttt{DIV}\ 2v_i)) \end{aligned}$$

   (b) We define O_{1iB} and O_{1iS} depending on β_i:

   - If $\beta_i = 0$:

$$\begin{aligned} O_{1iB} &= (\texttt{BUY}, 0), \\ O_{1iS} &= (\texttt{SELL}, c_i/(ni)) \end{aligned}$$

   - If $\beta_i = 1$:

$$\begin{aligned} O_{1iB} &= (\texttt{BUY}, c_i/(ni)), \\ O_{1iS} &= (\texttt{SELL}, 0) \end{aligned}$$

   (c) We add the pairs to the traders, alternating each iteration of the loop:

- If i is even:

$$T_1 := T_1 \cup (S_{1i}, O_{1iB}),$$
$$T_2 := T_2 \cup (S_{1i}, O_{1iS})$$

- Else:

$$T_1 := T_1 \cup (S_{1i}, O_{1iS}),$$
$$T_2 := T_2 \cup (S_{1i}, O_{1iB})$$

6. Now we must update the market cap in a way that represents the updated tape. We have two cases:

   - If the transition that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 1 does not result in a change of the tape (i.e. it reads a 1 and writes a 1):

     (a) We define the strategy and outcome that restores the market cap:

$$\begin{aligned} S_{1c} = \quad & (c_{n+1} \leq c_{n+2}) \wedge (c_{n+1} \geq c_{n+2}) \wedge \\ & (c_{2n+2} < c_{2n+3}) \wedge (c_{2n+1} < c_{2n+2} \wedge \cdots \wedge \\ & (c_{n+3} < c_{n+4}) \wedge (c_{n+1} \texttt{ DIV } v_{n+1} > 1) \wedge \\ & (c_{n+1} \texttt{ DIV } v_{n+1} > 2(c_{n+1} \texttt{ DIV } 2v_{n+1})), \\ O_{1cB} = \quad & (\texttt{BUY}, c_{n+1}), \\ O_{1cS} = \quad & (\texttt{SELL}, c_1) \end{aligned}$$

     (b) We add the pairs to the traders:

$$T_1 := T_1 \cup (S_{1c}, O_{1cB}),$$
$$T_2 := T_2 \cup (S_{1c}, O_{1cS})$$

   - If the transition that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 0 results in a change of the tape (i.e. it reads a 1 and writes a 0):

     (a) We define the strategy and outcome that restores the

market cap:

$$\begin{aligned} S_{1c} = \quad & (c_{n+1} \leq c_{n+2}) \wedge (c_{n+1} \geq c_{n+2}) \wedge \\ & (c_{2n+2} < c_{2n+3}) \wedge (c_{2n+1} < c_{2n+2} \wedge \cdots \wedge \\ & (c_{n+3} < c_{n+4}) \wedge (c_{n+1} \texttt{ DIV } v_{n+1} > 1) \wedge \\ & (c_{n+1} \texttt{ DIV } v_{n+1} > 2(c_{n+1} \texttt{ DIV } 2v_{n+1})), \\ O_{1cB} = \quad & (\texttt{BUY}, c_{n+1}), \\ O_{1cS} = \quad & (\texttt{SELL}, c_1 + v_{n+1}) \end{aligned}$$

(b) We add the pairs to the traders:

$$\begin{aligned} T_1 &:= T_1 \cup (S_{1c}, O_{1cB}), \\ T_2 &:= T_2 \cup (S_{1c}, O_{1cS}) \end{aligned}$$

7. We define the strategy and outcome that makes the volume of stock that was tried to be sold equal to 2^m where m is the position of the head of the Turing machine. We have two options:

   - If the transitions that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 1 makes the head move to the left:

$$\begin{aligned} S_{1v} = \quad & (c_{n+2} \leq c_{n+3}) \wedge (c_{n+2} \geq c_{n+3}) \wedge \\ & (c_{2n+3} < c_{2n+4}) \wedge (c_{2n+2} < c_{2n+3} \wedge \cdots \wedge \\ & (c_{n+4} < c_{n+5}) \wedge (c_{n+2} \texttt{ DIV } v_{n+2} > 1) \wedge \\ & (c_{n+2} \texttt{ DIV } v_{n+2} > 2(c_{n+2} \texttt{ DIV } 2v_{n+2})), \\ O_{1vB} = \quad & (\texttt{BUY}, v_{n+2}/2), \\ O_{1vS} = \quad & (\texttt{SELL}, v_{n+2}/2) \end{aligned}$$

   - If the transitions that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 1 makes the head move to the right:

$$\begin{aligned} S_{1v} = \quad & (c_{n+2} \leq c_{n+3}) \wedge (c_{n+2} \geq c_{n+3}) \wedge \\ & (c_{2n+3} < c_{2n+4}) \wedge (c_{2n+2} < c_{2n+3} \wedge \cdots \wedge \\ & (c_{n+4} < c_{n+5}) \wedge (c_{n+2} \texttt{ DIV } v_{n+2} > 1) \wedge \\ & (c_{n+2} \texttt{ DIV } v_{n+2} > 2(c_{n+2} \texttt{ DIV } 2v_{n+2})), \\ O_{1vB} = \quad & (\texttt{BUY}, 2v_{n+2}), \\ O_{1vS} = \quad & (\texttt{SELL}, 2v_{n+2}) \end{aligned}$$

8. We add the pairs to the traders:

$$T_1 := T_1 \cup (S_{1v}, O_{1vS}),$$
$$T_2 := T_2 \cup (S_{1v}, O_{1vB})$$

9. Finally we build the strategies and outcomes for reading Σ. Loop for $i := 1 \dots n$

   (a) We define $S_{\Sigma i}$:

$$\begin{aligned} S_{\Sigma i} := \quad & (c_{2+n+i} \leq c_{2+n+i+1}) \wedge (c_{2+n+i} \geq c_{2+n+i+1}) \wedge \\ & (c_{2+n+i-1} < c_{2+n+i}) \wedge (c_{2+n+i-2} < c_{2+n+i-1}) \wedge \cdots \wedge \\ & (c_{2+n+i-n} < c_{2+n+i-n+1}) \wedge (c_i \ \texttt{DIV} \ v_i \leq 1) \end{aligned}$$

   (b) We define $O_{\Sigma iB}$ and $O_{\Sigma iS}$ depending on γ_i:

   - If $\gamma_i = 0$:

$$O_{\Sigma iB} = (\texttt{BUY}, 0),$$
$$O_{\Sigma iS} = (\texttt{SELL}, c_i/(ni))$$

   - If $\gamma_i = 1$:

$$O_{\Sigma iB} = (\texttt{BUY}, c_i/(ni)),$$
$$O_{\Sigma iS} = (\texttt{SELL}, 0)$$

   (c) We add the pairs to the traders, alternating each iteration of the loop:

   - If i is even:

$$T_1 := T_1 \cup (S_{\Sigma i}, O_{\Sigma iB}),$$
$$T_2 := T_2 \cup (S_{\Sigma i}, O_{\Sigma iS})$$

   - Else:

$$T_1 := T_1 \cup (S_{\Sigma i}, O_{\Sigma iS}),$$
$$T_2 := T_2 \cup (S_{\Sigma i}, O_{\Sigma iB})$$

10. Now we must update the market cap in a way that represents the updated tape. We have two cases:

- If the transition that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol Σ results in writing a 1:
  (a) We define the strategy and outcome that restores the market cap:

$$\begin{aligned} S_{\Sigma c} = \; & (c_{n+1} \leq c_{n+2}) \wedge (c_{n+1} \geq c_{n+2}) \wedge \\ & (c_{2n+2} < c_{2n+3}) \wedge (c_{2n+1} < c_{2n+2}) \wedge \cdots \wedge \\ & (c_{n+3} < c_{n+4}) \wedge (c_{n+1} \texttt{ DIV } v_{n+1} \leq 1), \\ O_{\Sigma cB} = \; & (\texttt{BUY}, c_{n+1} + 2v_{n+1}), \\ O_{\Sigma cS} = \; & (\texttt{SELL}, c_1) \end{aligned}$$

  (b) We add the pairs to the traders:

$$\begin{aligned} T_1 &:= T_1 \cup (S_{1c}, O_{\Sigma cB}), \\ T_2 &:= T_2 \cup (S_{1c}, O_{\Sigma cS}) \end{aligned}$$

- If the transition that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol Σ results in writing a 0:
  (a) We define the strategy and outcome that restores the market cap:

$$\begin{aligned} S_{\Sigma c} = \; & (c_{n+1} \leq c_{n+2}) \wedge (c_{n+1} \geq c_{n+2}) \wedge \\ & (c_{2n+2} < c_{2n+3}) \wedge (c_{2n+1} < c_{2n+2}) \wedge \cdots \wedge \\ & (c_{n+3} < c_{n+4}) \wedge (c_{n+1} \texttt{ DIV } v_{n+1} \leq 1), \\ O_{\Sigma cB} = \; & (\texttt{BUY}, c_{n+1} + 2v_{n+1}), \\ O_{\Sigma cS} = \; & (\texttt{SELL}, c_1 + v_{n+1}) \end{aligned}$$

  (b) We add the pairs to the traders:

$$\begin{aligned} T_1 &:= T_1 \cup (S_{\Sigma c}, O_{\Sigma cB}), \\ T_2 &:= T_2 \cup (S_{\Sigma c}, O_{\Sigma cS}) \end{aligned}$$

11. We define the strategy and outcome that makes the volume of stock that was tried to be sold equal to 2^m where m is the position of the head of the Turing machine. We have two options:

- If the transitions that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 1 makes the head move to the left:

$$\begin{aligned} S_{\Sigma v} = \quad & (c_{n+2} \leq c_{n+3}) \wedge (c_{n+2} \geq c_{n+3}) \wedge \\ & (c_{2n+3} < c_{2n+4}) \wedge (c_{2n+2} < c_{2n+3}) \wedge \cdots \wedge \\ & (c_{n+4} < c_{n+5}) \wedge (c_{n+2} \texttt{ DIV } v_{n+2} \leq 1), \\ O_{\Sigma vB} = \quad & (\texttt{BUY}, v_{n+2}/2), \\ O_{\Sigma vS} = \quad & (\texttt{SELL}, v_{n+2}/2) \end{aligned}$$

- If the transitions that results from being in the state with code $0_1, \ldots, 0_n$ and reading the symbol 1 makes the head move to the right:

$$\begin{aligned} S_{\Sigma v} = \quad & (c_{n+2} \leq c_{n+3}) \wedge (c_{n+2} \geq c_{n+3}) \wedge \\ & (c_{2n+3} < c_{2n+4}) \wedge (c_{2n+2} < c_{2n+3}) \wedge \cdots \wedge \\ & (c_{n+4} < c_{n+5}) \wedge (c_{n+2} \texttt{ DIV } v_{n+2} \leq 1), \\ O_{\Sigma vB} = \quad & (\texttt{BUY}, 2v_{n+2}), \\ O_{\Sigma vS} = \quad & (\texttt{SELL}, 2v_{n+2}) \end{aligned}$$

12. We add the pairs to the traders:

$$\begin{aligned} T_1 &:= T_1 \cup (S_{\Sigma v}, O_{\Sigma vS}), \\ T_2 &:= T_2 \cup (S_{\Sigma v}, O_{\Sigma vB}) \end{aligned}$$

Since the market halts when no trader has a strategy that is activated, if the Turing machine halts when reading the symbol σ in the state with code $\delta_1 \ldots \delta_n$ we ignore that transition when writing the strategies and outcomes for that state. As all strategies are exhaustive, if we do not write a specific strategy for that case, no trader will activate, halting the market. Given the way we have written those strategies, no one will activate in that case, ending the market.

4.2.3 Initial State

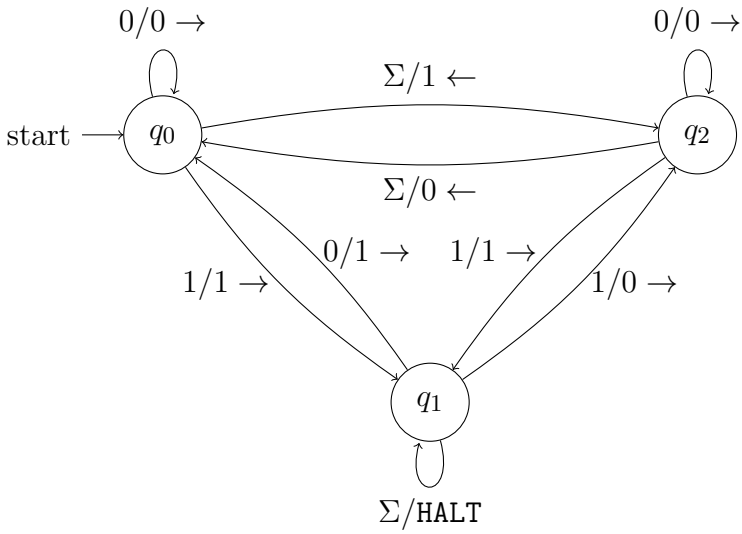
The only things that remain to be sorted out now are the initial values for the market caps and the volumes. We start with initializing $c_1 = c_2 = c_{2+n+1} = c_{2+n+2} = R$, where R has a binary

representation that, without its most significant bit, is equal to the initial state of the tape, sans the blank spaces. For the market caps c_{2+n+1} to c_3 we are going to define $\delta = R/n$ and we are going to define $c_{2+n+2-i} = c_{2+n+3-i} \pm \delta/i$ with $i = 1, \ldots, n$, depending on the code of the initial state of the machine (if the i-th bit is 0 then $c_{2+n+2-i} = c_{2+n+3-i} - \delta/i$ and $c_{2+n+2-i} = c_{2+n+3-i} + \delta/i$ otherwise). Finally, we are going to set $c_{2n+4} = c_{2+n+1}, \ldots, c_{2+n+3} = c_3$.

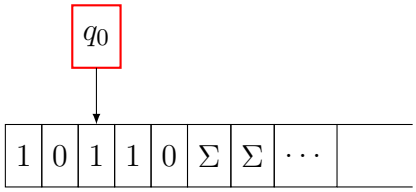
For the volumes, we set $v_1 = v_{2+n+1} = H$ where $H = 2^m$ and m is the distance from the first tape space to the one under the head (if the head is over the first tape space, then $m = 0$). We also set $v_2 = v_{2+n+2} = c_3$ The rest of the volumes are either going to be 0 or δ, as defined previously, depending on the increases and decreases of the market cap.

4.2.4 Example

Let us give an example. Suppose the following Turing machine:



With starting tape:



We need two bits in order to codify all this states so $n = 2$. The codes will be $q_0 = 00$, $q_1 = 01$ and $q_2 = 10$. The memory of the market will be $2(n + 2) = 8$ and R, as defined before, will have a binary representation of 101101, so $R = 45$. As the initial state for the Turing machine is q_0, the initial configuration of C is:

$$\begin{aligned} C = (c_1 &= 45, & c_5 &= 45, \\ c_2 &= 45, & c_6 &= 45, \\ c_3 &= 11.25, & c_7 &= 11.25, \\ c_4 &= 22.5, & c_8 &= 22.5) \end{aligned}$$

For V, as the head is on the third space, $H = 2$. Therefore, the initial state of V is:

$$\begin{aligned} V = (v_1 &= 2^2, & v_5 &= 2^2, \\ v_2 &= 11.25, & v_6 &= 11.25, \\ v_3 &= 11.25, & v_7 &= 11.25, \\ v_4 &= 22.5, & v_8 &= 22.5) \end{aligned}$$

Now we shall define our traders according to the algorithm that we gave previously:

1. When reading 0 on q_0 the Turing machine goes to q_0, does not

rewrite the tape and moves right:

$$
\begin{aligned}
T_1 = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 \leq 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 \leq 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{SELL}, c_6/4)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 \leq 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{BUY}, c_3)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 \leq 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{BUY}, 2v_4))\}, \\
T_2 = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 \leq 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{SELL}, c_5/2)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 \leq 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 \leq 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{SELL}, c_1)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 \leq 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{SELL}, 2v_4))\}
\end{aligned}
$$

2. From this step onward we will use $\cup =$ to denote that the new pairs will be added to the existing ones in T_1 and T_2 to avoid re-writing their whole definition. When reading 1 on q_0 the Turing machine goes to q_1, does not rewrite the tape and

moves right:

$$
\begin{aligned}
T_1 \cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{SELL}, 0)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{BUY}, c_3)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{BUY}, 2v_4))\}, \\
T_2 \cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{SELL}, c_5/2)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{BUY}, c_6/4)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{SELL}, c_1)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{SELL}, 2v_4))\}
\end{aligned}
$$

3. When reading Σ on q_0 the Turing machine goes to q_2, writes 1

and moves left:

$$
\begin{aligned}
T_1\cup = \{(\quad & ((c_5 \le c_6) \wedge (c_5 \ge c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \ \texttt{DIV}\ v_5 \le 1), \\
& (\texttt{BUY}, c_5/2)), \\
(\quad & ((c_6 \le c_7) \wedge (c_6 \ge c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \ \texttt{DIV}\ v_6 \le 1) \\
& (\texttt{SELL}, c_6/4)), \\
(\quad & ((c_3 \le c_4) \wedge (c_3 \ge c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \ \texttt{DIV}\ v_3 \le 1) \\
& (\texttt{BUY}, c_3 + 2v_3)), \\
(\quad & ((c_4 \le c_5) \wedge (c_4 \ge c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \ \texttt{DIV}\ v_4 \le 1) \\
& (\texttt{BUY}, v_4/2))\}, \\
T_2\cup = \{(\quad & ((c_5 \le c_6) \wedge (c_5 \ge c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \ \texttt{DIV}\ v_5 \le 1) \\
& (\texttt{SELL}, 0)), \\
(\quad & ((c_6 \le c_7) \wedge (c_6 \ge c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \ \texttt{DIV}\ v_6 \le 1) \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_3 \le c_4) \wedge (c_3 \ge c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \ \texttt{DIV}\ v_3 \le 1) \\
& (\texttt{SELL}, c_1)), \\
(\quad & ((c_4 \le c_5) \wedge (c_4 \ge c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \ \texttt{DIV}\ v_4 \le 1) \\
& (\texttt{SELL}, v_4/2))\}
\end{aligned}
$$

4. Now we move to q_1. When reading 0 in q_1, the Turing machine

goes to q_0, rewrites the zero to a 1 and moves right:

$$
\begin{aligned}
T_1\cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 > c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 \leq 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 > c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 \leq 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{SELL}, c_6/4)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 > c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 \leq 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{BUY}, c_3 + v_3)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 > c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 \leq 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{BUY}, 2v_4))\}, \\
T_2\cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 > c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 \leq 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{SELL}, c_5/2)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 > c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 \leq 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 > c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 \leq 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{SELL}, c_1)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 > c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 \leq 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{SELL}, 2v_4))\}
\end{aligned}
$$

5. When reading 1 on q_1 the Turing machine goes to q_2, it rewrites

the 1 as 0 and moves right:

$$
\begin{aligned}
T_1\cup = \{(\quad & ((c_5 \le c_6) \wedge (c_5 \ge c_6) \wedge (c_4 < c_5) \wedge (c_3 > c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_6 \le c_7) \wedge (c_6 \ge c_7) \wedge (c_5 < c_6) \wedge (c_4 > c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{SELL}, c_6/4)), \\
(\quad & ((c_3 \le c_4) \wedge (c_3 \ge c_4) \wedge (c_6 < c_7) \wedge (c_5 > c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{BUY}, c_3)), \\
(\quad & ((c_4 \le c_5) \wedge (c_4 \ge c_5) \wedge (c_7 < c_8) \wedge (c_6 > c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{BUY}, 2v_4))\}, \\
T_2\cup = \{(\quad & ((c_5 \le c_6) \wedge (c_5 \ge c_6) \wedge (c_4 < c_5) \wedge (c_3 > c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{SELL}, c_5/2)), \\
(\quad & ((c_6 \le c_7) \wedge (c_6 \ge c_7) \wedge (c_5 < c_6) \wedge (c_4 > c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_3 \le c_4) \wedge (c_3 \ge c_4) \wedge (c_6 < c_7) \wedge (c_5 > c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{SELL}, c_1 + v_3)), \\
(\quad & ((c_4 \le c_5) \wedge (c_4 \ge c_5) \wedge (c_7 < c_8) \wedge (c_6 > c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{SELL}, 2v_4))\}
\end{aligned}
$$

6. After reading Σ on q_1, the Turing machine halts, and as such the traders will not have a strategy for it.

7. Finally, we move to q_2. After reading 0 on q_2, the Turing

machine stays on q_2, does not rewrite anything and moves right:

$$
\begin{aligned}
T_1\cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 > c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 \leq 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{BUY}, c_5/2)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 > c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 \leq 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{SELL}, c_6/4)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 > c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 \leq 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{BUY}, c_3)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 > c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 \leq 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{BUY}, 2v_4))\}, \\
T_2\cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 > c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 \leq 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{SELL}, 0)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 > c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 \leq 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 > c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 \leq 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{SELL}, c_1)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 > c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 \leq 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{SELL}, 2v_4))\}
\end{aligned}
$$

8. When reading 1 on q_2 the Turing machine goes to q_1, does not

rewrite the tape and moves right:

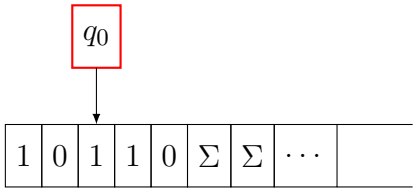
$$
\begin{aligned}
T_1 \cup = \{(\quad & ((c_5 \le c_6) \wedge (c_5 \ge c_6) \wedge (c_4 > c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_6 \le c_7) \wedge (c_6 \ge c_7) \wedge (c_5 > c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{SELL}, 0)), \\
(\quad & ((c_3 \le c_4) \wedge (c_3 \ge c_4) \wedge (c_6 > c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{BUY}, c_3)), \\
(\quad & ((c_4 \le c_5) \wedge (c_4 \ge c_5) \wedge (c_7 > c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{BUY}, 2v_4))\}, \\
T_2 \cup = \{(\quad & ((c_5 \le c_6) \wedge (c_5 \ge c_6) \wedge (c_4 > c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5))), \\
& (\texttt{SELL}, c_5/2)), \\
(\quad & ((c_6 \le c_7) \wedge (c_6 \ge c_7) \wedge (c_5 > c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6))), \\
& (\texttt{BUY}, c_6/4)), \\
(\quad & ((c_3 \le c_4) \wedge (c_3 \ge c_4) \wedge (c_6 > c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))), \\
& (\texttt{SELL}, c_1)), \\
(\quad & ((c_4 \le c_5) \wedge (c_4 \ge c_5) \wedge (c_7 > c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))), \\
& (\texttt{SELL}, 2v_4))\}
\end{aligned}
$$

9. After reading Σ on q_2 the Turing machine goes to q_0, writes 0

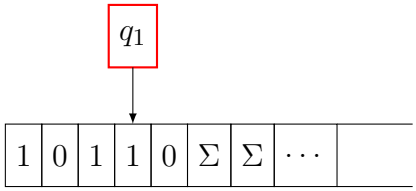
and moves left:

$$
\begin{aligned}
T_1\cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 > c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \ \texttt{DIV} \ v_5 \leq 1), \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 > c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \ \texttt{DIV} \ v_6 \leq 1) \\
& (\texttt{SELL}, c_6/4)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 > c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \ \texttt{DIV} \ v_3 \leq 1) \\
& (\texttt{BUY}, c_3 + 2v_3)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 > c_8) \wedge (c_6 < c_7) \wedge \\
& (c_4 \ \texttt{DIV} \ v_4 \leq 1) \\
& (\texttt{BUY}, v_4/2))\}, \\
T_2\cup = \{(\quad & ((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 > c_5) \wedge (c_3 < c_4) \wedge \\
& (c_5 \ \texttt{DIV} \ v_5 \leq 1) \\
& (\texttt{SELL}, c_5/2)), \\
(\quad & ((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 > c_6) \wedge (c_4 < c_5) \wedge \\
& (c_6 \ \texttt{DIV} \ v_6 \leq 1) \\
& (\texttt{BUY}, 0)), \\
(\quad & ((c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 > c_7) \wedge (c_5 < c_6) \wedge \\
& (c_3 \ \texttt{DIV} \ v_3 \leq 1) \\
& (\texttt{SELL}, c_1 + v_3)), \\
(\quad & ((c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 > c_8) \wedge (c_6 < c_7) \\
& (c_4 \ \texttt{DIV} \ v_4 \leq 1) \\
& (\texttt{SELL}, v_4/2))\}
\end{aligned}
$$

Now that we have defined our initial market, let us simulate two transitions of the Turing machine and its market counterparts. We start with:



As the Turing machine is in the state q_0 and reads 1, it moves to the right, going to the state q_1 and not changing the tape. This results in:



Given the market defined as above, the strategy that holds is $((c_5 \leq c_6) \wedge (c_5 \geq c_6) \wedge (c_4 < c_5) \wedge (c_3 < c_4) \wedge (c_5 \texttt{ DIV } v_5 > 1) \wedge (c_5 \texttt{ DIV } v_5 > 2(c_5 \texttt{ DIV } 2v_5)))$ as once we substitute we end up with:

$$(45 \leq 45) \wedge (45 \geq 45) \wedge (22.5 < 45) \wedge (0 < 22.5) \wedge (11 > 1) \wedge (11 > 10)$$

The outcome for T_1 is $(\texttt{BUY}, 0)$ and for T_2 is $(\texttt{SELL}, c_5/2)$, after substituting $(\texttt{SELL}, 22.5)$. As $A_B = 0$ and $A_S = 22.5$, we update the market cap to:

$$\begin{aligned} C = (c_1 &= 22.5, & c_5 &= 22.5, \\ c_2 &= 45, & c_6 &= 45, \\ c_3 &= 45, & c_7 &= 45, \\ c_4 &= 11.25, & c_8 &= 11.25) \end{aligned}$$

and the volume of stock that traders tried to sell to:

$$\begin{aligned} V = (v_1 &= 22.5, & v_5 &= 22.5, \\ v_2 &= 2^2, & v_6 &= 2^2, \\ v_3 &= 11.25, & v_7 &= 11.25, \\ v_4 &= 11.25, & v_8 &= 11.25) \end{aligned}$$

Now, the next strategy that holds is $((c_6 \leq c_7) \wedge (c_6 \geq c_7) \wedge (c_5 < c_6) \wedge (c_4 < c_5) \wedge (c_6 \texttt{ DIV } v_6 > 1) \wedge (c_6 \texttt{ DIV } v_6 > 2(c_6 \texttt{ DIV } 2v_6)))$. The outcomes of this strategy are $(\texttt{SELL}, 0)$ for T_1 and $(\texttt{BUY}, c_6/4)$ for T_2. As $A_B = 11.25$ and $A_S = 0$, we update the market cap to:

$$\begin{aligned} C = (c_1 &= 33.75, & c_5 &= 11.25, \\ c_2 &= 22.5, & c_6 &= 22.5, \\ c_3 &= 45, & c_7 &= 45, \\ c_4 &= 45, & c_8 &= 45) \end{aligned}$$

and the volume of stock that traders tried to sell to:

$$\begin{aligned} V = (v_1 &= 0, & v_5 &= 11.25, \\ v_2 &= 22.5, & v_6 &= 22.5, \\ v_3 &= 2^2, & v_7 &= 2^2, \\ v_4 &= 11.25, & v_8 &= 11.25) \end{aligned}$$

The next strategy that holds is $(c_3 \leq c_4) \wedge (c_3 \geq c_4) \wedge (c_6 < c_7) \wedge (c_5 < c_6) \wedge (c_3 \texttt{ DIV } v_3 > 1) \wedge (c_3 \texttt{ DIV } v_3 > 2(c_3 \texttt{ DIV } 2v_3))$, which gives the outcomes $(\texttt{BUY}, c_3)$ for T_1 and $(\texttt{SELL}, c_1)$. This results in $A_B = 45$ and $A_S = 11.25$ and the updated market caps are:

$$\begin{aligned} C = (c_1 &= 45, & c_5 &= 45, \\ c_2 &= 33.75, & c_6 &= 11.25, \\ c_3 &= 22.5, & c_7 &= 22.5, \\ c_4 &= 45, & c_8 &= 45) \end{aligned}$$

and the volumes:

$$\begin{aligned} V = (v_1 &= 11.25, & v_5 &= 11.25, \\ v_2 &= 0, & v_6 &= 11.25, \\ v_3 &= 22.5, & v_7 &= 22.5, \\ v_4 &= 2^2, & v_8 &= 2^2) \end{aligned}$$

Finally, the next strategy that holds is $(c_4 \leq c_5) \wedge (c_4 \geq c_5) \wedge (c_7 < c_8) \wedge (c_6 < c_7) \wedge (c_4 \texttt{ DIV } v_4 > 1) \wedge (c_4 \texttt{ DIV } v_4 > 2(c_4 \texttt{ DIV } 2v_4))$ and the outcomes are $(\texttt{BUY}, 2v_4)$ for T_1 and $(\texttt{SELL}, 2v_4)$ for T_2. This results in $A_B = 2^3$ and $A_S = 2^3$ and the updated market caps are:

$$\begin{aligned} C = (c_1 &= 45, & c_5 &= 45, \\ c_2 &= 45, & c_6 &= 45, \\ c_3 &= 33.75, & c_7 &= 11.25, \\ c_4 &= 22.5, & c_8 &= 22.5) \end{aligned}$$

and the volumes:

$$\begin{aligned} V = (v_1 &= 2^3, & v_5 &= 2^2, \\ v_2 &= 11.25, & v_6 &= 11.25, \\ v_3 &= 0, & v_7 &= 11.25, \\ v_4 &= 22.5, & v_8 &= 22.5) \end{aligned}$$

As $c_1 = c_2$ we have simulated one transition of the Turing machine, and we can see that $c_1 = 45$ which corresponds to the tape state as it has not changed, $v_1 = 2^3$ which corresponds to the correct tape space, as it has moved one further right, and $c_4 < c_5$ and $c_3 > c_4$ so the code for the current state is 01, which corresponds to the current state of the Turing machine, q_1.

CHAPTER 5
Conclusions & Future Work

5.1 Comparison of the Models

We have given two market models that are Turing complete. Let us discuss and compare these models in order to understand the differences.

The main strong point of the **first model** is that it goes through the inner workings of markets with a fine comb, using transactional offers to model the price of the stock. Share price, as given in real life financial markets, is the price of the last accepted transaction, which this model perfectly captures. Another strong point that arises from this fine grain approach is that, besides changing shifts depending on the price of the stock, real life markets are a generalization of this model. This means that, if we were able to refine this model in a way that shifts did not depend on the price of the stock, we would have a model that is feasible to replicate in a real life market.

However the model has a huge flaw: everything revolves around the most significant digit of the amount of the last transactional offer. We briefly discussed technical analysis in chapter 4, and how traders who believe in it use esoteric techniques to predict price shifts (e.g. shapes in the graphics), however even these "financial astrologers" would not use the most significant digit of the price as a parameter for their decision making. Furthermore, even if real world traders used the most significant digit of the price to make decisions, shifts would not change depending on it. This flaw makes

believing in this model alone a hard task but, nonetheless, having a fine grain model of a market that is Turing complete is a huge stepping stone.

For the **second model**, its strong point is the realism of its traders. This model was built using much broader strokes than the first one and it generalizes price variation through supply and demand laws. However, traders behave in a much more realistic way, using market capitalization and volumes traded as part of their strategies, which makes the model more believable as something that could arise from real life interactions.

The problem with this model falls mostly on how the market capitalization updates. As we stated before, the price of a share is given by the amount of the last transaction and market capitalization is equal to the price of a share times the amount of shares that the business has. Stating that it varies by adding the amount of shares times its price that traders tried to buy and subtracting the amount of shares times its price that traders tried to sell is not exactly correct, as we should be checking how many shares are being sold and at which price to correctly state the new market capitalization. Even though the general idea behind this transformation is correct (the value of a stock goes up if traders try to buy more of it than the amount traders are trying to sell and vice versa), the evolution of prices is a simplification compared to the more complex systems that real world markets follow.

In conclusion, having two models with distinct strengths and flaws lends more credibility to the idea that markets are Turing complete, even though we have not formally prove it.

5.2 Future Work

Both models share a flaw that we have not discussed yet: share possession is not taken into account. Future work should start by solving this issue, as even though we can ignore it for a bit by taking shorting into account (i.e. borrowing shares to sell them and then re-buying them in the future, betting that the stock price is

going to fall thus earning a benefit), it does not make sense for a trader to short half the value of the company in two consecutive iterations, something that could happen in the second model. An easy solution could be having traders know how many shares does each other trader hold (something that happens in markets based on blockchain, as everyone knows in which wallet is each asset) and, as soon as someone is under certain threshold, a signal is written either on transactional offers or in the market caps (akin to the begin-of-state-code in the second market). This would trigger a sequence of trades that shuffle shares around in a way that every trader has enough, before going back to simulating the Turing machines.

Besides that, other points where the models could be improved are their flaws. An alternative way of changing shifts and traders not depending on the most significant digit of the stock price for the first model, and a finer approach to market cap evolution for the second one would be steps in the right direction.

5.3 Conclusions

Proving that a "real world" object (as opposed to a formal object) is Turing complete is not an easy task. Translating the real world inner workings of these objects to a formal computational model requires precision and ingenuity. Generalize too much and you end up proving the Turing completeness using properties that do not belong in the object. Specify too much and you end up proving the Turing completeness of something that would never be an instance of the real system.

Crafting models such as the ones stated in this work can be done by two methods: start by something that is Turing complete and refine it until it models the behaviour of the real object (bottom-up), or start by something that models the behaviour of the object and work with it until you are able to simulate a Turing machine with it (top-down). Both approaches were used in this work, as the first model was developed by the bottom-up approach and the second one was developed using the top-down approach. Interestingly,

neither of them ended up being better than the other, and each of them has a set of strengths and flaws.

Are markets Turing complete? This question is not completely answered in this work, and even if we keep refining market models, we may be able to nitpick flaws and unrealistic behaviours in every one of them. The models presented in this work might not be perfect but both represent a stepping stone that has never been reached, as far as we know, and, as stated before, proving that both are Turing complete strongly indicates our initial hypothesis.

Bibliography

[1] Nicholas Barberis and Richard Thaler. A survey of behavioral finance. Working Paper 9222, National Bureau of Economic Research, September 2002.

[2] Paul A. Samuelson. *Proof that Properly Anticipated Prices Fluctuate Randomly*, chapter 2, pages 25–38. 2015.

[3] Philip Maymin. Markets are efficient if and only if P = NP, 2010.

[4] Philip Mirowski and Koye Somefun. Markets as evolving computational entities. *Journal of Evolutionary Economics*, 8(4):329–356, 1999.

[5] James Aspnes, David F. Fischer, Michael J. Fischer, Ming-Yang Kao, and Alok Kumar. *Towards Understanding the Predictability of Stock Markets from the Perspective of Computational Complexity*, pages 129–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[6] Yurii Rogozhin. Small universal turing machines. *Theoretical Computer Science*, 168(2):215–240, 1996.

[7] Manfred Kudlek and Yurii Rogozhin. A universal turing machine with 3 states and 9 symbols. In Werner Kuich, Grzegorz Rozenberg, and Arto Salomaa, editors, *Developments in Language Theory*, pages 311–318, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[8] John Cocke and Marvin Minsky. Universality of tag systems with p = 2. *J. ACM*, 11(1):15–20, jan 1964.

[9] C.D. Kirkpatrick and J.A. Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians*. Pearson Education, 2010.

[10] Cheol-Ho Park and Scott H. Irwin. The Profitability of Technical Analysis: A Review. AgMAS Project Research Reports 37487, University of Illinois at Urbana-Champaign, Department of Agricultural and Consumer Economics, October 2004.

[11] Cheol-Ho Park and Scott H. Irwin. What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4):786–826, 2007.

[12] Robert R. Prechter Jr. and Wayne D. Parker. The financial/economic dichotomy in social behavioral dynamics: The socionomic perspective. *Journal of Behavioral Finance*, 8(2):84–108, 2007.

[13] Rui Shen, Zhiqing Meng, Minchao Zheng, and Ricardo López-Ruiz. Research on two-level price-fluctuation supply chain ordering strategy problem. *Discrete Dynamics in Nature and Society*, 2020.

[14] Eivind Brækkan. *Why do Prices Change? An Analysis of Supply and Demand Shifts and Price Impacts in the Farmed Salmon Market*. PhD thesis, 07 2015.