5-5-2012

Computable Linear Orders and Turing Reductions

Whitney P. Turner
patton@math.uconn.edu, patton@math.uconn.edu

Recommended Citation
https://opencommons.uconn.edu/gs_theses/246
Computable Linear Orders and Turing Reductions

Whitney Patton Turner

B.A. Mathematics, Albion College, Albion, Michigan, 2009

A Thesis
Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science at the University of Connecticut

2012
I would like to thank my advisor Reed Solomon. I would not have been able to complete this without his direction and support. For the past two years, Reed has been an outstanding mentor and teacher. He has always had time to help and work with me through the many revisions and corrections. I could not have asked for a more exceptional advisor. I would also like to thank Johanna Franklin and Henry Towsner for being on my advisory committee.

I would like to thank my past and current office-mates for being my friends and making life entertaining in Storrs. Thank you to the Math Department’s administrative assistants Tammy Prentice and Monique Roy for always knowing the answer to any question or problem that came up, and to David Gross and Reed Solomon for their advice and helping me with the many student or administrative related issues.

I would like to thank my parents, Rita and Lee Patton, for everything they have done to help me throughout my life. I love you very much Mom and Dad! I would also like to thank my siblings: Courtney, Bryan, Bree, Chelsea, Dack, and Bree, and in laws, the Turners. I love you all! Finally, I would like to thank my wonderful husband, Dustin Turner for being an all around awesome guy. Without you, I would never have even gone into math!
# TABLE OF CONTENTS

**Introduction : Introduction** ................................................. 1
  0.1. Computability Theory ................................................. 1
  0.2. Linear Orders ......................................................... 3

**Ch. 1 : Completeness** .................................................... 10
  1.1. Dense ............................................................... 10
  1.2. Discrete ........................................................... 13
  1.3. Finite Block ....................................................... 17

**Ch. 2 : Discrete** .......................................................... 21
  2.1. Construction I ..................................................... 21
  2.2. Construction II .................................................... 32

**Ch. 3 : Dense** ............................................................. 40
  3.1. Construction I ..................................................... 40
  3.2. Construction II .................................................... 51

**Bibliography** .............................................................. 24
0.1 Computability Theory

The main focus of this thesis is to measure the complexity of a variety of relations on computable linear orders. To do this measurement, we will use two reducibilities. A set $A \subseteq \mathbb{N}$ is Turing reducible to a set $B$ if $A = \phi^B_e$ meaning that there is an oracle machine that computes the characteristic function of $A$ using oracle $B$. We denote this as $A \leq_T B$. During our proofs, this basically means that we can ask our set $B$ questions, specifically whether or not a number is in $B$. $A$ is many-one reducible or $m$-reducible to $B$ if there is a computable function $f$ such that $x \in A$ if and only if $f(x) \in B$. We denote this as $A \leq_m B$. Recall that if $A \leq_m B$, then $A \leq_T B$, but not conversely. We will give a concrete example of when the converse fails in this thesis.

Through this thesis, we will use standard notation from computability theory as found in Robert I. Soare’s *Recursively Enumerable Sets and Degrees* or Hartley Rogers, Jr.’s *Theory of Recursive Functions and Effective Computability*. We use $\phi_0, \phi_1, \phi_2,...$ to denote the standard list of partial computable functions.
and $W_0, W_1, W_2,...$ to denote their domains. Recall that the sets $W_e$ are called computable enumerable or c.e. sets. We utilize several familiar index sets: $K$, $Fin$, and $Inf$, formally defined as:

(i) $K = \{ x | \phi_x(x) \text{ converges} \}$

(ii) $Fin = \{ x | W_x \text{ is finite} \}$

(iii) $Inf = \{ x | W_x \text{ is infinite} \}$

These sets live in a hierarchy which is defined by quantifier complexity. We define the $\Sigma^0_n$ and $\Pi^0_n$ sets in the following way.

**Definition:** Let $A$ be a set.

(i) $A$ is in $\Sigma^0_0$ or $\Pi^0_0$ if and only if $A$ is computable.

(ii) For $n \geq 1$, $A$ is in $\Sigma^0_n$ if there is a computable $R(x, y_1, y_2, ..., y_n)$ such that

$$x \in A \text{ if and only if } (\exists y_1)(\forall y_2)(\exists y_3)...(Qy_n)R(x, y_1, y_2, ..., y_n).$$

(iii) For $n \geq 1$, $A$ is in $\Pi^0_n$ if there is a computable $R(x, y_1, y_2, ..., y_n)$ such that

$$x \in A \text{ if and only if } (\forall y_1)(\exists y_2)(\forall y_3)...(Qy_n)R(x, y_1, y_2, ..., y_n).$$

By fully writing out their definitions, we have that $K \in \Sigma^0_1$, $Fin \in \Sigma^0_2$, and $Inf \in \Pi^0_2$. $A \in \Sigma^0_n(\Pi^0_n)$ is $\Sigma^0_n(\Pi^0_n)$-complete if for any arbitrary set $B \in \Sigma^0_n(\Pi^0_n)$, we have $B \leq^m A$. We will use the facts that $K$ is $\Sigma^0_1$-complete, $Fin$ is $\Sigma^0_2$-complete, and $Inf$ is $\Pi^0_2$-complete.
0.2 Linear Orders

A linear order is a pair \((\mathcal{D}, \leq_{\mathcal{D}})\) where \(\mathcal{D}\) is a set and \(\leq_{\mathcal{D}}\) is a binary relation on \(\mathcal{D}\) which is reflexive, transitive, and anti-symmetric. For this thesis, we will work with countable (and often computable) linear orders and will typically assume that \(\mathcal{D} = \mathbb{N}\). Our standard notation for a linear order will be \(\mathcal{L} = (\mathbb{N}, \leq_{\mathcal{L}})\). We say that \(\mathcal{L} = (\mathbb{N}, \leq_{\mathcal{L}})\) is computable if and only if the binary relation \(\leq_{\mathcal{L}}\) is computable.

We will use several classical notions from the theory of linear orders. Specifically, we need the following definitions.

- A linear order \(\mathcal{L}\) is **discrete** if every element \(a \in \mathcal{L}\) has an immediate successor and an immediate predecessor unless \(a\) is the least or greatest element. If \(a\) is the least element of \(\mathcal{L}\), we require \(a\) to have an immediate successor and if \(a\) is the greatest element of \(\mathcal{L}\), then we require \(a\) to have an immediate predecessor. Note that every finite linear order is discrete. An interval \((a, b) \subset \mathcal{L}\) is discrete if the ordering given by \(\leq_{\mathcal{L}}\) restricted to \((a, b)\) is discrete.

- A linear order \(\mathcal{L}\) is **dense** if \(\mathcal{L}\) is isomorphic to the usual order on \(\mathbb{Q}\). (Recall that we assume our orderings are countable.) As above, we say an interval \((a, b)\) in \(\mathcal{L}\) is dense if the order given by \(\leq_{\mathcal{L}}\) restricted to \((a, b)\) is dense.
Let \( L \) be a linear order and let \( a \in L \). We say \( b \) is in the same block as \( a \) if the interval \([a, b]\) (if \( a \leq_L b \)) or \([b, a]\) (if \( b \leq_L a \)) is finite. The block of \( a \) is the set of elements \( b \) such that \( b \) is in the same block as \( a \).

Let \( L \) be a computable linear order. The following are ordering relations we will examine:

(i) \( \text{FinBl}_L = \{c \mid c \text{ is in a finite block in } L\} \)

(ii) \( \text{Den}_L = \{(b, c) \mid (b, c) \text{ is dense in } L\} \)

(iii) \( \text{Dis}_L = \{(b, c) \mid (b, c) \text{ is discrete in } L\} \)

We can calculate the complexity of each of these relations as follows. \( \text{Den}_L \in \Pi^0_2 \) because \( (b, c) \in \text{Den}_L \) if and only if

\[
\begin{align*}
b &<_L c \land \forall x, y \in (b, c) \ [x <_L y \rightarrow \exists z \ (x <_L z <_L y)] \\
&\land \forall x \in (b, c)[\exists z(b <_L z <_L x) \land \exists u(x <_L u <_L c)].
\end{align*}
\]

To analyze the complexity of \( \text{FinBl}_L \) and \( \text{Den}_L \), we first consider the immediate predecessor relation \( \text{Pred}_L(x, y) \) and the immediate successor relation \( \text{Succ}_L(x, y) \). \( \text{Pred}_L(x, y) \) holds if and only if

\[
x <_L y \land \neg \exists z(x <_L z <_L y)
\]

and hence is \( \Pi^0_1 \). \( \text{Succ}_L(x, y) \) holds if and only if

\[
y <_L x \land \neg \exists z(y <_L z <_L x)
\]
and hence is also $\Pi^0_1$. We can now show that $\text{Dis}_L \in \Pi^0_3$ because $\langle b, c \rangle \in \text{Dis}_L$ if and only if

$$b <_L c \land \forall x \in (b, c) \exists u, v(Pred_L(u, x) \land Succ_L(v, x))$$

Finally, to analyze $\text{FinBl}_L$, we also need the complexity of the limit from below relation, $\text{LimBelow}_L(x)$ and the limit from above relation, $\text{LimAbove}_L(x)$. $\text{LimBelow}_L(x)$ holds if and only if

$$\forall y (y <_L x \rightarrow \exists z(y <_L z <_L x))$$

and hence is $\Pi^0_2$. $\text{LimAbove}_L(x)$ holds if and only if

$$\forall y (x <_L y \rightarrow \exists z(x <_L z <_L y))$$

and hence is also $\Pi^0_2$. Note the following subtlety of these definitions. If $L$ has a least element $a$, then $\text{LimBelow}_L(a)$ holds since we do not require that there is a $y <_L x$ in the definition of $\text{LimBelow}_L(x)$. Similarly, if $L$ has a greatest element $a$, then $\text{LimAbove}_L(a)$ holds. This aspect of these definitions will make the definition of $\text{FinBl}_L(x)$ more compact.

Now, we have that $\text{FinBl}_L(x) \in \Sigma^0_3$ because $\text{FinBl}_L(x)$ holds if and only if

$$\exists y (y = \langle x_1, ..., x_n \rangle \land \exists i \leq n(x_i = x) \land$$

$$\forall i \leq n (\text{Succ}_L(x_{i+1}, x_i)) \land \text{LimBelow}_L(x_1) \land \text{LimAbove}_L(x_n))$$

In Chapter 1, we will show that each of these relations is complete for some computable $L$. 
Theorem: There are computable linear orders $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ such that $\text{Den}_{\mathcal{L}_1}$ is $\Pi^0_2$-complete, $\text{Dis}_{\mathcal{L}_2}$ is $\Pi^0_3$-complete, and $\text{FinBl}_{\mathcal{L}_3}$ is $\Sigma^0_3$-complete.

In Chapters 2 and 3, we will consider the complexity of $\text{Den}_{\mathcal{L}}$, $\text{Dis}_{\mathcal{L}}$, and $\text{Block}_{\mathcal{L}}$ when we fix the element $b$ to be the least element in $\mathcal{L}$. Specifically, if $\mathcal{L}$ is a computable linear order and $b \in \mathcal{L}$, then we define

(i) $\text{Den}_\mathcal{L}(b) = \{c \mid (b, c) \text{ is dense in } \mathcal{L}\}$

(ii) $\text{Dis}_\mathcal{L}(b) = \{c \mid (b, c) \text{ is discrete in } \mathcal{L}\}$

(iii) $\text{Block}_\mathcal{L}(b) = \{c \mid c \text{ is in the same block as } b\}$

We show that we can code both $0'$ and $0''$ into these relations by a Turing reduction when $b$ is the least element of $\mathcal{L}$.

Theorem: There are computable linear orders $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ with $b$ denoting the least element in each order such that $0'' \leq_T \text{Den}_{\mathcal{L}_1}(b)$, $0'' \leq_T \text{Dis}_{\mathcal{L}_2}(b)$, and $0'' \leq_T \text{Block}_{\mathcal{L}_3}(b)$.

Before starting the main results of this thesis, we present a theorem originally due to Carl Jockusch which has not appeared in print. This theorem shows that for the $\text{Block}_\mathcal{L}(b)$ relation, we cannot improve our Turing reduction in the
previous theorem to an $m$-reduction.

**Theorem**: If $\mathcal{L}$ is a computable linear order and $B$ is a block in $\mathcal{L}$ with least element $b \in B$, then $K \not\leq_m B$.

**Proof**: Let $\mathcal{L}$ be a computable linear order and let $B$ be a block in $\mathcal{L}$ with least element $b \in B$. We want to show that $K \not\leq_m B$.

By way of contradiction, assume that $K \leq_m B$ and fix a computable function $f$ such that $n \in K$ if and only if $f(n) \in B$. Without loss of generality, we assume that for all $n$, $b <_\mathcal{L} f(n)$.

We define two partial computable functions $g(x, y)$ and $h(x, y)$ using the $s$-$m$-$n$ theorem:

$$\phi_{g(x,y)}(u) = \begin{cases} 
\uparrow & \text{if } f(x) <_\mathcal{L} f(y) \\
0 & \text{if } f(x) \geq_\mathcal{L} f(y)
\end{cases}$$

$$\phi_{h(x,y)}(u) = \begin{cases} 
0 & \text{if } f(x) <_\mathcal{L} f(y) \\
\uparrow & \text{if } f(x) \geq_\mathcal{L} f(y)
\end{cases}$$

We want to apply the following theorem which is an adjustment to the regular
recursion theorem found in Hartley Rogers, Jr.’s *Theory of Recursive Functions and Effective Computability*.

**Double Recursive Theorem:** For any recursive functions $g$ and $h$, there exist $m$ and $n$ such that $\phi_m = \phi_{g(m,n)}$ and $\phi_n = \phi_{h(m,n)}$.

By this theorem, fix some $n$ and $m$ such that $\phi_n = \phi_{g(m,n)}$ and $\phi_m = \phi_{h(m,n)}$. Using our function $f$, the relationship between point placement in our linear order can be broken down into three possibilities. We are using the placement of points in $\mathcal{L}$ and in particular, whether the points are included in the block $B$, to create a contradiction.

(i) $f(n) = f(m)$

If $f(n) = f(m)$, then we know that

$$\phi_n(n) = \phi_{g(n,m)}(n) \downarrow = 0 \Rightarrow \phi_n(n) \downarrow = 0$$

So, $n \in K$ which implies that $f(n) \in B$. On the other hand, we also know that

$$\phi_m(m) = \phi_{h(n,m)}(m) \uparrow \Rightarrow \phi_m(m) \uparrow.$$

So, $m \notin K$ which implies that $f(m) \notin B$. Thus, we have a contradiction since $f(n) = f(m)$, but $f(n)$ is in the block and $f(m)$ is not in the block.
(ii) \( f(n) <_L f(m) \)

If \( f(n) <_L f(m) \), then we know that

\[
\phi_n(n) = \phi_{g(n,m)}(n) \uparrow \Rightarrow \phi_n(n) \uparrow
\]

So, \( n \notin K \) which implies that \( f(n) \notin B \). On the other hand, we also know that

\[
\phi_m(m) = \phi_{g(n,m)}(m) \downarrow = 0 \Rightarrow \phi_m(m) = 0.
\]

So, \( m \in K \) which implies that \( f(m) \in B \). Thus, we have a contradiction since \( b <_L f(n) <_L f(m) \), but \( f(n) \) is not in the block and \( f(m) \) is in the block.

(iii) \( f(m) <_L f(n) \) If \( f(m) <_L f(n) \), then we know that

\[
\phi_n(n) = \phi_{g(n,m)}(n) \downarrow = 0 \Rightarrow \phi_n(n) = 0
\]

So, \( n \in K \) which implies that \( f(n) \in B \). On the other hand, we also know that

\[
\phi_m(m) = \phi_{g(n,m)}(m) \uparrow \Rightarrow \phi_m(m) \uparrow.
\]

So, \( m \notin K \) which implies that \( f(m) \notin B \). Thus, we have a contradiction since \( b <_L f(m) <_L f(n) \), but \( f(n) \) is in the block and \( f(m) \) is not in the block.

Thus, we know that the relation between the computable function \( f \) and our linear order derives a contradiction in each case. Therefore, \( K \not\leq_m B \).
In this chapter, we prove the completeness results stated in the Introduction. Recall that if $\mathcal{L}$ is a computable linear order, then $\text{Den}_\mathcal{L} \in \Pi^0_2$, $\text{Dis}_\mathcal{L} \in \Pi^0_3$, and $\text{FinBl}_\mathcal{L} \in \Sigma^0_3$. We show that in each case, we can construct a computable $\mathcal{L}$ for which the relation is complete at the given level of the arithmetic hierarchy.

1.1 Dense

**Theorem:** There is a computable linear order $\mathcal{L}$ for which

$\text{Den}_\mathcal{L} = \{ (b, c) \mid (b, c) \text{ is dense in } \mathcal{L} \}$ is $\Pi^0_2$-complete.

**Proof:** Since $\text{Inf} = \{ e \mid W_e \text{ is infinite} \}$ is $\Pi^0_2$-complete, it suffices to build a computable linear order $\mathcal{L}$ such that $\text{Inf} \leq_m \text{Den}_\mathcal{L}$. To accomplish this reduction, we use pairs of witness points $b_n$ and $c_n$, and we make the interval $(b_n, c_n)$ dense if and only if $W_n$ is infinite. The requirements are the following:

$$R_n : n \in \text{Inf} \text{ if and only if } (b_n, c_n) \text{ is dense in } \mathcal{L}.$$ 

**Construction:**

Stage 0: Set down the set of even numbers in their usual order and label the
numbers in pairs as \( b_n \) and \( c_n \). Each pair \( b_i \) and \( c_i \) is associated to the domain \( W_i \).

\[
b_0 < L c_0 < L b_1 < L c_1 < L ... < L b_n < L c_n < L ... 
\]

**Stage s+1:** At stage \( s+1 \), we examine each \( W_n \), for \( n \leq s \), to see if it receives a new element at stage \( s + 1 \). Since the requirements for each \( W_n \) are applied to each separate interval, we can treat each such requirement individually. Note that this means that there is no injury in this construction.

**Case I:** Assume a new element enters \( W_n \). We need to make progress towards making the interval \((b_n, c_n)\) dense. To accomplish this, suppose the interval \((b_n, c_n)\) currently contains \( m \) many points and appears as

\[
b_n < L z_m < L z_{m-1} < L ... < L z_1 < L c_n
\]

Let \( y_n^1, ... , y_n^{m+1} \) be the \( m + 1 \) least unused odd numbers. Place the odd numbers into the interval \((b_n, c_n)\) between each current pair of successor points as follows:

\[
b_n < L y_n^{m+1} < L z_m < L y_n^m < L z_{m-1} < L ... < L y_n^2 < L z_1 < L y_n^1 < L c_n
\]

In later constructions, we will describe this process of adding a new point between each pair of current successors in \([b_n, c_n]\) as **partially densifying the interval** \((b_n, c_n)\).

**Case II:** Assume no new elements are enumerated into \( W_n \). We leave the interval \((b_n, c_n)\) as it is and do not add points towards densifying the interval.
Verification:

(i) \( n \in \text{Inf} \) if and only if \((b_n, c_n)\) is dense

We know that \( n \in \text{Inf} \) if and only if \( W_n \) is an infinite domain. This is true if and only if we enumerate infinitely many points into \( W_n \). When a point enumerates into \( W_n \), we place points into \((b_n, c_n)\). This process, when done infinitely often, creates a dense interval. Thus, if there are infinitely many points in \( W_n \), then \((b_n, c_n)\) is dense.

We know that \( n \notin \text{Inf} \) if and only if \( W_n \) is finite. This is true if and only if only a finite number of points are enumerated in \( W_n \) which implies that only finitely many points were placed into \((b_n, c_n)\). Thus, if \( n \notin \text{Inf} \), then \((b_n, c_n)\) is finite and specifically, is not dense.

(ii) Effective Construction

The construction is effective because there are only a finite number of things done at each stage for each \( W_n \). The points we place into each \([b_n, c_n]\) are all odd and thus, there will never be a lack of available numbers as there are only finitely many odds used at each stage. Also, since there are indices \( W_n \) for which \( W_n \) is infinite, the construction will use all of the odd numbers. Hence the domain of \( L \) is \( \mathbb{N} \), which is computable. This implies that to compare \( i \) and \( j \) in order, we need just run the construction until both elements appear and compare where they land in \( L \).
1.2 Discrete

**Theorem:** There is a computable linear order $L$ such that

$$\text{Dis}_L = \{(b, c) \mid [b, c] \text{ is discrete in } L\}$$

is $\Pi^0_3$-complete.

**Proof:** Let $X$ be a $\Pi^0_3$-complete set. We need to build a computable linear order $L$ such that $X \leq_m \text{Dis}_L$. Since $\text{Fin} = \{e \mid W_e \text{ is finite }\}$ is $\Sigma^0_2$-complete, we can fix a computable function $f(x, n)$ such that

$$n \in X \text{ if and only if } \forall x(W_{f(x,n)} \text{ is finite })$$

To accomplish this, we will use pairs of witness points $b_n$ and $c_n$ to meet the following requirements.

$$R_n : \forall x(W_{f(x,n)} \text{ finite}) \text{ if and only if } (b_n, c_n) \text{ is discrete in } L.$$  

**Construction:**

*Stage 0:* Effectively partition the even numbers into infinitely many infinite sets $X, P_0, P_1, P_2, \ldots$ We will use these sets of numbers to put down a basic structure for our computable order $L$ that will be filled in at later stages.

To define this basic structure, first place the numbers in $X$ in $L$ in their usual order and label them as follows:

$$b_0 <_L c_0 <_L b_1 <_L c_1 <_L b_2 <_L c_2 <_L \ldots$$

For each $n$, we will place the numbers in $P_n$ into the interval $(b_n, c_n)$ in their usual order so our ordering $L$ at stage 0 looks like:
Label the points in each \( P_n \) in groups of three as follows:

\[
u^0_n <_L d^0_n <_L v^0_n <_L u^1_n <_L d^1_n <_L v^1_n <_L ...
\]

Therefore, our order \( L \) at stage 0 looks like:

\[
b_0 <_L u^0_0 <_L d^0_0 <_L v^0_0 <_L u^1_0 <_L d^1_0 <_L v^1_0 <_L c_0 <_L b_1 <_L u^0_1 <_L d^0_1 <_L v^0_1 <_L c_1 <_L b_2 <_L ...
\]

Stage \( s+1 \): At stage \( s+1 \), we let each requirement \( R_n \) for \( n \leq s \) act in turn. Since \( R_n \) will work only in the interval \( (b_n, c_n) \), we can treat each requirement individually and there is no injury in this construction.

Action for \( R_n \): For each \( x \leq s \), we check if \( W_{f(x,n)} \) has received a new element.

Case I: Assume \( W_{f(x,n)} \) has received a new element. We need to make progress towards making the interval \( (b_n, c_n) \) not discrete. Let \( y \) and \( z \) be the least unused odd numbers. We add \( y \) and \( z \) to \( L \) as the immediate predecessor and successor of \( d_n^x \) as follows:

\[
b_n <_L ... <_L u^x_n <_L finite <_L y <_L d^x_n <_L finite <_L v^x_n <_L z <_L finite <_L v^x_n <_L ... <_L c_n
\]

Notice that if we add a new predecessor and successor for \( d^x_n \) infinitely often, then \( d^x_n \) becomes a limit point and \( (b_n, c_n) \) is not discrete. However, if we add
only finitely many such points, the interval \((u^*_n, v^*_n)\) will be finite.

Case II: Assume \(W_{f(x,n)}\) had not received a new element. We leave the interval \((b_n, c_n)\) looking discrete, so we do not add any new points and we move on to \(x + 1\).

Verification:

(i) \(n \in X\) if and only if \((b_n, c_n)\) is discrete

First, suppose \(n \notin X\). In this case, we can fix an \(x\) such that \(W_{f(x,n)}\) is infinite. Since \(W_{f(x,n)}\) receives a new element at infinitely many stages, we add a new successor and predecessor to \(d^x_n\) infinitely often. Therefore, \(d^x_n\) is a limit point and has neither an immediate predecessor nor an immediate successor in \(\mathcal{L}\). Since \(d^x_n \in (b_n, c_n)\), the interval \((b_n, c_n)\) is not discrete in \(\mathcal{L}\).

On the other hand, suppose \(n \in X\). In this case, each set \(W_{f(x,n)}\) is finite and hence each interval \((u^x_n, v^x_n)\) is finite. Thus the interval \((b_n, c_n)\) in \(\mathcal{L}\) looks like

\[
\begin{align*}
  u^0_n <_\mathcal{L} & \text{ finite } <_\mathcal{L} v^0_n <_\mathcal{L} u^1_n <_\mathcal{L} \text{ finite } <_\mathcal{L} v^1_n <_\mathcal{L} \ldots \\
\end{align*}
\]

Since \((b_n, c_n)\) has order type \(\mathbb{N}\), it is discrete.
(ii) Effective Construction

The construction is effective because there are only a finite number of things done at each stage for each $W_n$. The points we place into each $(b_n, c_n)$ are all odd and thus, there will never be a lack of available numbers as there are only finitely many odds used at each stage. Also, since there are numbers $n \notin X$, and hence sets $W_{f(x,n)}$ which are infinite, the construction will use all of the odd numbers. Hence the domain of $L$ is $\mathbb{N}$, which is computable. This implies that to compare $i$ and $j$ in order, we need just run the construction until both elements appear and compare where they land in $L$. 
1.3 Finite Block

**Theorem:** There is a computable linear order \( \mathcal{L} \) such that 
\[ \text{FinBl}_{\mathcal{L}} = \{ c \mid c \text{ is in a finite block in } \mathcal{L} \} \text{ is } \Sigma^0_3\text{-complete.} \]

**Proof:** Let \( X \) be a \( \Sigma^0_3 \)-complete set. We need to build a computable linear order \( \mathcal{L} \) such that \( X \leq_m \text{FinBl}_{\mathcal{L}} \). Since \( \text{Inf} \) is \( \Pi^0_2 \)-complete, we can fix a computable function \( f(n, x) \) such that
\[ n \in X \text{ if and only if } \exists x (W_{f(x,n)} \text{ is infinite}). \]

To build \( \mathcal{L} \), we will use witness points \( c_n \) and meet the requirements:

\[ R_n : \exists x (W_{f(x,n)} \text{ is infinite}) \text{ if and only if } c_n \in \text{FinBl}_{\mathcal{L}} \]

**Construction:**

**Stage 0:** Effectively partition the even numbers into infinitely many infinite sets \( X, P_0, P_1, P_2, ... \). We will use these sets of numbers to put down a basic structure for our computable order \( \mathcal{L} \) that will be filled in at later stages.

To define this basic structure, first place the numbers in \( X \) in \( \mathcal{L} \) in their usual order and label them as follows:

\[ c_0 <_\mathcal{L} c_1 <_\mathcal{L} c_2 <_\mathcal{L} ... \]

For each \( n \), we will place the numbers in \( P_n \) around \( c_n \) and order them in order type \( \mathbb{Z} \) with labels as follows:
Stage s+1: At stage s+1, we let each requirement $R_n$ for $n \leq s$ act in turn. Since $R_n$ will act within the part of $L$ defined by the $P_n$ points, we can treat each requirement individually and there is no injury in this construction.

Action for $R_n$: For each $x \leq s$, we check if $W_{f(x,n)}$ has received a new element.

Case I: Assume $W_{f(x,n)}$ does receive a new element. We need to make progress towards making $c_n$ a member of a finite block. So, let $z_1$ and $z_2$ be the two least unused odd numbers. Place $z_1$ into $L$ as the immediate predecessor of $d_n^{x-1}$ (or $c_n$ if $x = 0$) and $z_2$ into $L$ as the immediate successor of $d_n^{x-1}$ (or $c_n$ if $x = 0$). The order looks like:

$$
\ldots <_L b_1 <_L b_0 <_L c_0 <_L d_0^0 <_L d_1^0 <_L \ldots <_L b_1 <_L b_0 <_L c_1 <_L d_0^0 <_L d_1^1 <_L \ldots
$$

Notice that if $b_n^{x-1}$ and $d_n^{x-1}$ receive new predecessors and successors infinitely often, then they become limit points from below and above respectively, and the block containing $c_n$ cannot extend beyond $[b_n^{x-1}, d_n^{x-1}]$.

Case II: Assume $W_{f(x,n)}$ did not receive a new element. We do nothing in this case and do not add any new points to $L$. Proceed to $x + 1$.  

18
Verification:

(i) \( n \in X \) if and only if \( c_n \) is in a finite block

First, suppose \( n \in X \). We can fix \( x \) such that \( W_{f(x,n)} \) is infinite. Assume we have fixed the least such \( x \). Since \( b_{n}^{x-1} \) receives infinitely many new predecessors, it is a limit point from below. Similarly, \( d_{n}^{x-1} \) is a limit point from above. (If \( x = 0 \), then \( c_n \) is a limit point from below and above, and hence is in a block of size 1. We continue assuming \( x \neq 0 \)). Thus, our order around \( c_n \) looks like

\[ \ldots <_{L} b_{n}^{x-1} <_{L} \text{finite} <_{L} b_{n}^{0} <_{L} \text{finite} <_{L} c_n <_{L} \text{finite} \]
\[ <_{L} d_{n}^{0} <_{L} \text{finite} <_{L} d_{n}^{x-1} <_{L} \ldots \]

The interval \([b_{n}^{x-1}, d_{n}^{x-1}]\) is finite and constitutes the block containing \( c_n \). Therefore, \( c_n \) is a finite block.

On the other hand, suppose \( n \notin X \). In this case, \( W_{f(x,n)} \) is finite for all \( x \) and hence each interval of the form \([b_{n}^{x}, b_{n}^{x-1}]\), \([b_{n}^{0}, c_n]\), \([c_n, d_{n}^{0}]\), and \([d_{n}^{x-1}, d_{n}^{x}]\) is finite. Therefore, the block containing \( c_n \) has order type \( \mathbb{Z} \) and is infinite.

(ii) Effective Construction

The construction is effective because there are only a finite number of
things done at each stage. The points we place into each set of even numbers around $c_n$ are all odd and thus, there will never be a lack of available numbers as there are only finitely many odds used at each stage. Also, since there are numbers $n \in X$ and hence infinite sets, the construction will use all of the odd numbers. Hence the domain of $L$ is $\mathbb{N}$, which is computable. This implies that to compare $i$ and $j$ in order, we need just run the construction until both elements appear and compare where they land in $L$. 
Chapter 2
Discrete

The main goal of this chapter is to construct a computable linear order $\mathcal{L}$ with a least element $b$ such that

$$0'' \leq_T \text{Dis}_{\mathcal{L}}(b) = \{c \mid (b, c) \text{ is discrete in } \mathcal{L}\}$$

and

$$0'' \leq_T \text{Block}_{\mathcal{L}}(b) = \{c \mid [b, c] \text{ is finite in } \mathcal{L}\}$$

We first give a simpler construction coding $0'$ instead of $0''$ and then we show how to modify this construction to code $0''$.

2.1 Construction I

Recall: We define an interval as discrete if every element has a successor and predecessor except if the interval has a least or greatest element. If the interval has a least element, the least element will not have a predecessor and if the interval has a greatest element, then the greatest element will not have a successor. In particular, finite intervals are discrete and we will utilize that part of the definition in this proof.

Theorem: There is a computable linear order $\mathcal{L}$ with least element $b$ such that

$$0' \leq_T \text{Dis}_{\mathcal{L}}(b) = \{c \mid (b, c) \text{ is discrete in } \mathcal{L}\}.$$
Proof: In order to prove this theorem, we want to build a computable linear order \( L \) around a least element \( b \) such that the interval \((b, x_n)\) is discrete if and only if \( n \in K \). We have the following requirements:

\[
R_n : n \in K \text{ if and only if } x_n \in Dis_L(b)
\]

with ordering \( R_0 < R_1 < R_2 < \ldots \)

The basic strategy for a single requirement \( R_0 \) is to put down a pair of points \( l_0 \) and \( x_0 \) such that

\[
b <_L l_0 <_L x_0.
\]

Our goal is to do one of two things in the interval \((l_0, x_0)\) depending on whether \( 0 \in K \) or not. If \( 0 \notin K \), then we want to make the open interval \((l_0, x_0)\) isomorphic to \( \omega^* \). This action makes \( l_0 \) into a limit point from above and hence, makes \((b, x_0)\) not discrete because \( l_0 \) has no successor. If \( 0 \in K \), then we want to make \([l_0, x_0]\) finite which makes \((l_0, x_0)\) discrete. In the context of a single requirement, this also makes \((b, x_0)\) finite and thus, discrete.

To accomplish this goal, at each stage \( s \), we check whether \( 0 \in K_s \). If not, then we add a new least point in the interval \((l_0, x_0)\).

\[
b <_L l_0 <_L \text{ new point } <_L z_k <_L \ldots <_L z_1 <_L z_0 <_L x_0
\]

In this case, we regard \( R_0 \) as a building state requirement and in the general
construction, we will be taking the $B$ outcome (for building).

On the other hand, if $0 \in K_s$, then we want to stop building our $\omega^*$-chain and restrain the interval $[l_0, x_0]$ from ever growing again. We regard $R_0$ as a restraining state requirement. In the general construction, we will be taking the $R$ outcome (for restraining).

To handle a second requirement $R_1$, we need a second pair of witness points $l_1 <_L x_1$. The placement of these points depends on the action of $R_0$. As long as $R_0$ is in the building state, we are working under the assumption that $[l_0, x_0]$ will not be discrete in the limit and therefore we can put any points we want into the interval $(b, l_0)$. Thus, we place the points $l_1$ and $x_1$ as follows:

$$b <_L l_1 <_L x_1 <_L l_0 <_L x_0$$

The requirement $R_1$ now works exactly as $R_0$ did. As long as $1 \notin K_s$, $R_1$ continues to add points to $(l_1, x_1)$ towards making this interval isomorphic to $\omega^*$. If $1 \in K_s$, then $R_1$ restrains $[l_1, x_1]$ by not allowing any additional points to enter this interval.

However, consider what happens if $R_0$ changes to the restraining state. In this case, $R_0$ freezes the finite size of $[l_0, x_0]$ and wants to also make sure that $(b, x_0)$ is finite. Therefore, $R_1$ needs to stop adding points in its current interval $(l_1, x_1)$.
since these points are added into the interval \([b, l_0]\).

In this situation, \(R_1\) adds new witness points \(l_1^*\) and \(x_1^*\) and places them such that
\[
 b < \mathcal{L} l_1 < \mathcal{L} \text{ Finite} < \mathcal{L} x_1 < \mathcal{L} l_0 < \mathcal{L} \text{ Finite} < \mathcal{L} x_0 < \mathcal{L} l_1^* < \mathcal{L} x_1^*.
\]

\(R_1\) can now proceed as before using the interval \((l_1^*, x_1^*)\). Notice that if \(0 \in K\) and \(1 \in K\), then \(R_0\) makes \([l_0, x_0]\) finite and makes \([b, l_0]\) finite (by forcing \(R_1\) to stop using witnesses \(l_0\) and \(x_0\)). \(R_1\) also makes \([l_1^*, x_1^*]\) finite. Thus, \((b, x_0)\) and \((b, x_1^*)\) are both finite (and hence discrete), winning \(R_0\) and \(R_1\).

Notice that with two requirements, we need to know the outcome at \(R_0\) in order to know which interval in \(\mathcal{L}\) codes information about whether \(1 \in K\). To use \(\{c \mid (b, c)\ \text{is discrete in} \ \mathcal{L}\}\) to compute \(K\), we proceed as follows. First, we need to ask if \((b, x_0)\) is discrete. If the interval is not discrete, then we know that \(0 \notin K\) and that the witness pair for \(R_1\) is \((l_1, x_1)\). So, we ask if \((b, x_1)\) is discrete. If so, then \(1 \notin K\) and if not, then \(1 \in K\).

On the other hand, if \((b, x_0)\) is discrete, then we know that \(0 \in K\). So, at some finite point in the construction, we switched our witness pair for \(R_1\) to \((l_1^*, x_1^*)\). Therefore, to determine if \(1 \in K\), we need to ask if \((b, x_1^*)\) is discrete. If it is discrete, then \(1 \in K\) and if it is not discrete \(1 \notin K\).
The witness $x_2$ is set down based upon the restrictions of the higher priority requirements $R_0$ and $R_1$.

- If $0 \in K$ and $1 \in K$, then $x_2$ is set down such that $b <_L x_0 <_L x_1 <_L x_2$.
- If $0 \in K$ and $1 \notin K$, then $x_2$ is set down such that $b <_L x_0 <_L x_2 <_L x_1$.
- If $0 \notin K$ and $1 \in K$, then $x_2$ is set down such that $b <_L x_1 <_L x_2 <_L x_0$.
- If $0 \notin K$ and $1 \notin K$, then $x_2$ is set down such that $b <_L x_2 <_L x_1 <_L x_0$.

The rest of the witnesses are set down based upon the higher priority requirements.

Notice that, as described above for $R_0$ and $R_1$, in order to determine which interval in $L$ codes information about whether $2 \in K$, we need to know the outcomes for $R_0$ and $R_1$. The answer to the question of whether $(b, x_0)$ is discrete tells us which witness pair for $R_1$ codes the information about whether $1 \in K$. Once we know which witness pair codes this information, we can ask a discreteness question to determine which witness pair for $R_2$ codes information about whether $2 \in K$. In general, to determine which witness pair codes information about whether $n \in K$, we will have to use discreteness questions to determine the correct witness pairs for $0, 1, ..., n - 1$. This process illustrates why our reduction is a Turing reduction as opposed to an $m$-reduction.
We will be setting up a tree of strategies $T = \{R, B\}^{<\omega}$ such that $R <_L B$. We want to indicate that the order determined by the tree will be represented by $L$ as opposed to $\mathcal{L}$ which refers to the actual linear order. The basic universal strategy is to stay in a build state until $n$ is enumerated into $K$. During this time, we will be building an $\omega^\ast$-chain between the witness pair $x_n$ and $l_n$. When $n$ is enumerated into $K$, we want to switch to the restrain strategy which will not allow any new points to be introduced in the interval $[l_n, x_n]$.

It remains to describe where a strategy $\alpha \in T$ places its witness points $l_\alpha$ and $x_\alpha$ when it is first eligible to act. To describe this placement, we treat the pair $l_\alpha$ and $x_\alpha$ as a single entity $w_\alpha$ and write

$$w_\alpha <_L w_\beta$$

as an abbreviation for $l_\alpha <_L x_\alpha <_L l_\beta <_L x_\beta$. Our method of adding points (as described below) will ensure that the intervals $(l_\alpha, x_\alpha)$ and $(l_\beta, x_\beta)$ are always disjoint. For distinct strategies $\alpha$ and $\beta$, we place $w_\alpha <_L w_\beta$ if and only if either

- $\beta <_L \alpha$ ($\beta$ is to the left of $\alpha$ in the tree of strategies)
- or $\alpha \subseteq \beta$ and $\beta(|\alpha|) = R$ ($\beta$ extends $\alpha \ast R$)
- or $\beta \subseteq \alpha$ and $\alpha(|\beta|) = B$ ($\alpha$ extends $\beta \ast B$).
Local Action for $\alpha$ for $R_e$:

(i) When $R_\alpha$ is first eligible to act, place a new pair of witnesses $l_\alpha <_\mathcal{L} x_\alpha$ in $\mathcal{L}$ as described above.

(ii) If $e \notin K_s$, then add a new least point into the interval $(l_\alpha, x_\alpha)$ and take outcome $\alpha * B$.

(iii) If $e \in K_s$, do not add any points to $(l_\alpha, x_\alpha)$ and take outcome $\alpha * R$.

Note two properties of the placement of points in our linear order $\mathcal{L}$. First, only $\alpha$ is allowed to put points in the interval $(l_\alpha, x_\alpha)$. This protects our interval against other witnesses encroaching on its territory. Second, when $l_\alpha$ and $x_\alpha$ are placed, the interval contains no $l_\beta$ and $x_\beta$ points. This serves the same purpose as the previous restriction in that it preserves the previous intervals.

Construction

Stage 0: We begin with the empty set. So, we need to set down point $b$.

Stage $s+1$: Follow the path down the tree of strategy to level $s$ as directed by the action of the strategies eligible to act. When we reach level $s + 1$, end the stage.
Verification:

(i) True Path

The true path in our tree of strategies is the leftmost path visited infinitely often. Notice that if \( \alpha \) is on the true path, then either \( \alpha \) always takes outcome \( \alpha \star B \) or at some stage \( s \), \( \alpha \) switches to outcome \( \alpha \star R \) and always takes \( \alpha \star R \) at all future stages. Therefore, as the construction proceeds, the paths taken only move left and a node \( \alpha \) at level \( n \) is on the true path if and only if \( \alpha \) is eventually on the path at every stage past some stage \( s \).

(ii) Lemma 2.1.1: Let \( \alpha \) be on the true path. If \( \alpha \star R \) is on the true path, then \( (b,x_\alpha) \) is finite and hence discrete.

Proof: Fix \( t \) such that for all \( s \geq t \), \( \alpha \star R \) is on the path at stage \( s \). To prove this lemma, we need to consider the ways that a point could possibly enter the interval \( (b,x_\alpha) \) after stage \( t \). There are two possibilities:

(a) \( \alpha \) places points into \( [l_\alpha,x_\alpha] \) after stage \( t \).

If we have taken the outcome \( R \) at stage \( t \), then we know that \( n \) entered \( K \) by stage \( t \). By the construction, there is no possibility for points to enter \( [l_\alpha,x_\alpha] \) because we cannot return to the building outcome.

(b) Another strategy \( \beta \) places points into \( (b,x_\alpha) \). In this case, we must have \( w_\beta <_L w_\alpha \). Consider the ways in which this could happen.

- If \( \beta \subseteq \alpha \), then \( w_\beta <_L w_\alpha \) means \( \alpha(|\beta|) = R \) and hence \( \beta \star R \subseteq \alpha \).

Therefore, \( \beta \star R \) is on the true path and is on the current path at all
stages \( s \geq t \). By the action at \( \beta, \beta \) does not add any points to \([l_{\beta}, x_{\beta}]\).

- If \( \alpha \subseteq \beta \), then \( w_{\beta} \prec_{\mathcal{L}} w_{\alpha} \) means \( \beta(\lvert \alpha \rvert) = B \) and hence \( \alpha * B \subseteq \beta \).

However, \( \alpha * B \) is never on the path after state \( t \), so \( \beta \) is never eligible to act after stage \( t \). Therefore, \( \beta \) cannot add new points to \([l_{\beta}, x_{\beta}]\).

- If \( \alpha \) and \( \beta \) are incomparable, then \( w_{\beta} \prec_{\mathcal{L}} w_{\alpha} \) means \( \alpha \prec_{\mathcal{L}} \beta \). Since our path only moves left and \( \alpha \) is on the true path, \( \beta \) is never eligible to act after stage \( t \) and never adds any new points to \( \mathcal{L} \) after stage \( t \).

In all cases, we see that no strategy \( \beta \neq \alpha \) can add new points to \((b, x_{\alpha})\) after stage \( t \).

(iii) **Lemma 2.1.2**: Let \( \alpha \) be on the true path. If \( \alpha * B \) is on the true path, then \((b, x_{\alpha})\) is not discrete.

**Proof**: If \( \alpha * B \) is on the true path, then it is eligible to act infinitely often and it adds points to make \((l_{\alpha}, x_{\alpha})\) isomorphic to \( \omega^* \). Therefore, \( l_{\alpha} \in (b, x_{\alpha}) \) and \( l_{\alpha} \) has no immediate successor. Hence, \((b, x_{\alpha})\) is not discrete.

(iv) **Lemma 2.1.3**: Let \( \alpha \) be the \( R_c \) strategy on the true path. Then, the following are equivalent:

- \((b, x_{\alpha})\) is discrete.
- \((b, x_{\alpha})\) is finite.
- \( e \in K \)
- \( \alpha * R \) is on the true path.
Proof: If $e \in K$, then by our local action for $\alpha$, $\alpha * R$ is on the true path. By Lemma 2.1.1, $(b, x_\alpha)$ is discrete and finite. If $e \notin K$, then by our local action for $\alpha$, $\alpha * B$ is on the true path. By Lemma 2.1.2, $(b, x_\alpha)$ is not discrete and hence infinite.

(v) **Lemma 2.1.4**: $0' \leq_T \text{Dis}_L(b) = \{ c \mid (b, c) \text{ is discrete in } L \}$

Proof: We define a function $f : \mathbb{N} \to T$ by setting $f(e) = \alpha$ if $\alpha$ is the $R_e$ strategy on the true path. Notice that $f$ is computable from $\text{Dis}_L(b)$ since $f(0)$ is the unique $R_0$ strategy and by Lemma 2.1.3,

$$f(e + 1) = \begin{cases} f(e) * B & \text{if } (b, x_{f(e)}) \text{ is not discrete} \\ f(e) * R & \text{if } (b, x_{f(e)}) \text{ is discrete} \end{cases}$$

We can compute $0'$ from $f$ using Lemma 2.1.3 since

$$n \in K \text{ if and only if } (b, x_{f(n)}) \text{ is discrete}$$

and thus $n \in K$ if and only if $f(n + 1) = f(n) * R$. Therefore, we have $K \leq_T f$ and $f \leq_T \text{Dis}_L(b)$, so $K \leq_T \text{Dis}_L(b)$.

(vi) **Effective Construction**

The construction is effective because there are only a finite number of things done at each stage for each requirement $R_n$. Also, since there are $n \notin K$, we will build at least one infinite $\omega^*$-chain, the construction will use all of the natural numbers. Hence the domain of $L$ is $\mathbb{N}$, which is
computable. This implies that to compare $i$ and $j$ in order, we need just run the construction until both elements appear and compare where they land in $L$.

**Corollary:** $0' \leq_T \text{Block}_L(b) = \{ c \mid [b, c] \text{ is finite in } L \}$

**Proof:** By Lemma 2.1.3, if $\alpha$ is an $R_e$ strategy on the true path, then $(b, x_\alpha)$ is discrete if and only if $(b, x_\alpha)$ is finite. Therefore, we could equivalently define the function $f$ in Lemma 2.1.4 by

$$f(e + 1) = \begin{cases} 
  f(e) * B & \text{if } (b, x_{f(e)}) \text{ is infinite} \\
  f(e) * R & \text{if } (b, x_{f(e)}) \text{ is finite}
\end{cases}$$

Thus, $f \leq_T \text{Block}_L(b) = \{ c \mid [b, c] \text{ is finite in } L \}$ and hence $K \leq_T \text{Block}_L(b) = \{ c \mid [b, c] \text{ is finite in } L \}$. 

31
2.2 Construction II

**Theorem:** There is a computable linear order \( \mathcal{L} \) with least element \( b \) such that
\[
0'' \leq_T \text{Dis}_\mathcal{L}(b) = \{c \mid (b,c) \text{ is discrete in } \mathcal{L}\}.
\]

**Proof:** In order to prove this theorem, we want to build a computable linear order \( \mathcal{L} \) around a least element \( b \) such that the interval \((b,x_n)\) is not discrete if and only if \( n \in \text{Inf} \) where \( \text{Inf} = \{e \mid W_e \text{ is infinite}\} \). We have the following requirements:

\[
R_n : n \in \text{Inf} \text{ if and only if } (b,x_n) \notin \text{Dis}_\mathcal{L}(b)
\]

with ordering \( R_0 < R_1 < R_2 < ... \)

The basic strategy for a single requirement \( R_0 \) is similar to the strategy in the \( 0' \) construction. We want to put down a pair of points \( l_0 \) and \( x_0 \) such that
\[
b <_\mathcal{L} l_0 <_\mathcal{L} x_0.
\]

Our goal is to do one of two things in the interval \([l_0,x_0]\) depending on whether \( 0 \in \text{Inf} \) or not. If \( 0 \in \text{Inf} \), then we want to make the open interval \((l_0,x_0)\) isomorphic to \( \omega' \). This action makes \( l_0 \) into a limit point from above and hence, makes \((b,x_0)\) not discrete because \( l_0 \) has no successor. If \( 0 \notin \text{Inf} \), then we want to make \([l_0,x_0]\) finite which makes \([l_0,x_0]\) discrete. In the context of a single requirement, this also makes \((b,x_0)\) finite and thus, discrete.
To accomplish this goal, at each stage $s$, we check whether $W_0$ had received a new point. If so, then we add a new least point in the interval $(l_0, x_0)$.

\[ b <_{L} l_0 <_{L} \text{new point} <_{L} z_k <_{L} ... <_{L} z_1 <_{L} z_0 <_{L} x_0 \]

In this case, we regard $R_0$ as a building state requirement and in the general construction, we will be taking the $B$ outcome (for building).

On the other hand, if $W_0$ has not received a new point, then we want to stop (at least temporarily) building our $\omega^*$-chain and restrain the interval $[l_0, x_0]$ from growing. We regard $R_0$ as a restraining state requirement. In the general construction, we will be taking the $R$ outcome (for restraining).

We will be setting up a tree of strategies $T = \{B, R\}^{<\omega}$ such that $B <_{L} R$. Notice that we have switched from $R <_{L} B$ in the $0'$ construction to $B <_{L} R$ in the $0''$ construction. In the $0''$ construction, it is possible to take both the $B$ and $R$ outcomes infinitely often. For example, we would do this if there are infinitely many stages at which $W_n$ gets a new element and infinitely many stages at which $W_n$ does not get a new element. In this case, the true outcome is the $B$ outcome since $W_n$ is infinite. In order to have the true path be the leftmost path visited infinitely often, we need $B <_{L} R$ for the $0''$ construction.

The basic universal strategy is to stay in a restrain state until $W_n$ adds a new
point. In a restrained stage, we will not allow any new points to be introduced in the interval \([l_n, x_n]\). When \(W_n\) grows, we want to switch to the build strategy and add a single point towards building a copy of \(\omega^*\) in \((l_n, x_n)\). If we take the outcome \(B\) infinitely often, then \((l_n, x_n)\) will grow to a copy of \(\omega^*\).

It remains to describe where a strategy \(\alpha \in T\) places its witness points \(l_\alpha\) and \(x_\alpha\) when it is first eligible to act. To describe this placement, we treat the pair \(l_\alpha\) and \(x_\alpha\) as a single entity \(w_\alpha\) and write

\[w_\alpha <_L w_\beta\]

as an abbreviation for \(l_\alpha <_L x_\alpha <_L l_\beta <_L x_\beta\). Our method of adding points (as described below) will ensure that the intervals \((l_\alpha, x_\alpha)\) and \((l_\beta, x_\beta)\) are always disjoint. For distinct strategies \(\alpha\) and \(\beta\), we place \(w_\alpha <_L w_\beta\) if and only if either

- \(\alpha <_L \beta\) (\(\alpha\) is to the left of \(\beta\) in the tree of strategies)
- or \(\alpha \subseteq \beta\) and \(\beta(|\alpha|) = R\) (\(\beta\) extends \(\alpha * R\))
- or \(\beta \subseteq \alpha\) and \(\alpha(|\beta|) = B\) (\(\alpha\) extends \(\beta * B\)).

**Local Action for \(\alpha\) for \(R_\xi\):**

(i) When \(R_\alpha\) is first eligible to act, place a new pair of witnesses \(l_\alpha <_L x_\alpha\) in \(L\).

Let \(s\) be the last stage at which \(\alpha\) was eligible to act (with \(s = 0\) if this is the first time \(\alpha\) is eligible to act).
(ii) If $W_{e, \alpha} \neq W_{e, \bar{\alpha}}$, then add a new least point into the interval $(l_\alpha, x_\alpha)$ and take outcome $\alpha \ast B$.

(iii) If $W_{e, \alpha} = W_{e, \bar{\alpha}}$, do not add any points to $(l_\alpha, x_\alpha)$ and take outcome $\alpha \ast R$.

Note two properties of the placement of points in our linear order $\mathcal{L}$. First, only $\alpha$ is allowed to put points in the interval $(l_\alpha, x_\alpha)$. This protects our interval against other witnesses encroaching on its territory. Second, when $l_\alpha$ and $x_\alpha$ are placed, the interval contains no $l_\beta$ and $x_\beta$ points. This serves the same purpose as the previous restriction in that it preserves the previous intervals.

Construction

Stage 0: We begin with the empty set. So, we need to set down point $b$.

Stage $s+1$: Follow the path down the tree of strategy to level $s$ as directed by the action of the strategies eligible to act. When we reach level $s + 1$, end the stage.

Verification:

(i) True Path

First let $s$ be an $\alpha$-stage if $\alpha$ is eligible to act at stage $s$. The true path in our tree of strategies is the leftmost path visited infinitely often. Assume
\(\alpha\) is on the true path. If there are infinitely many \(\alpha\)-stages when we take outcome \(\alpha \ast B\), then \(\alpha \ast B\) is on the true path. Otherwise, there exists a stage \(t\) such that for all \(\alpha\)-stages after \(t\), we take \(\alpha \ast R\) and \(\alpha \ast R\) is on the true path. Note that if \(\alpha\) is on the true path, then there exists only finitely many stages \(s\) when the true path is to the left of \(\alpha\).

(ii) \textbf{Lemma 2.2.1}: Let \(\alpha\) be on the true path. If \(\alpha \ast R\) is on the true path, then \((b, x_\alpha)\) is finite and hence discrete.

\textit{Proof:} Fix a stage \(t\) such that for all \(s \geq t\), the path is not to the left of \(\alpha \ast R\). To prove this lemma, we need to consider the ways that a point could enter the interval \((b, x_\alpha)\) after stage \(t\).

(a) \(\alpha\) places points in \([l_\alpha, x_\alpha]\) after stage \(t\).

If \(\alpha\) places a point in \([l_\alpha, x_\alpha]\), then it takes outcome \(\alpha \ast B\). Since \(\alpha \ast B <_L \alpha \ast R\) and the path is never left of \(\alpha \ast R\) after stage \(t\), \(\alpha\) cannot place any points in \([l_\alpha, x_\alpha]\) after stage \(t\).

(b) Another strategy \(\beta\) places points in \([b, x_\alpha]\). In this case, we must have \(w_\beta <_L w_\alpha\). Consider the ways this could happen.

(i) If \(\beta \subseteq \alpha\), then \(w_\beta <_L w_\alpha\) means \(a(|\beta|) = R\) and hence \(\beta \ast R \subseteq \alpha\). If \(\beta\) adds points to \(L\), it takes outcome \(\beta \ast B\) which is left of \(\alpha \ast R\). Therefore, \(\beta\) adds no more points after stage \(t\).

(ii) If \(\alpha \subseteq \beta\), then \(w_\beta <_L w_\alpha\) means \(\beta(|\alpha|) = B\) and hence \(\alpha \ast B \subseteq \beta\). Since \(\alpha\) takes outcome \(R\) at all \(\alpha\)-stages after \(t\), \(\beta\) is never eligible to act after
stage $t$ and hence does not add any points after stage $t$.

(iii) If $\alpha$ and $\beta$ are incomparable, then $w_\beta <_L w_\alpha$ means $\beta <_L \alpha$. However, the path is never to the left of $\alpha * R$ after stage $t$ and therefore, never to the left of $\alpha$ after stage $t$. Hence, $\beta$ cannot add points after stage $t$.

In all cases, we see that no strategy $\beta \neq \alpha$ can add new points to $(b, x_\alpha)$ after stage $t$.

(iii) **Lemma 2.2.2:** Let $\alpha$ be on the true path. If $\alpha * B$ is on the true path, then $(b, x_\alpha)$ is not discrete.

**Proof:** If $\alpha * B$ is on the true path, then it is eligible to act infinitely often and it adds points to make $(l_\alpha, x_\alpha)$ isomorphic to $\omega_*$. Therefore, $l_\alpha \in (b, x_\alpha)$ and $l_\alpha$ has no immediate successor. Hence, $(b, x_\alpha)$ is not discrete.

(iv) **Lemma 2.2.3:** Let $\alpha$ be the $R_c$ strategy on the true path. Then the following are equivalent:

- $(b, x_\alpha)$ is discrete.
- $(b, x_\alpha)$ is finite.
- $e \notin Inf$
- $\alpha * R$ is on the true path.

**Proof:** If $e \notin Inf$, then by our local action for $\alpha$, $\alpha * R$ is on the true path. By Lemma 2.2.1, $(b, x_\alpha)$ is discrete and finite. If $e \in Inf$, then by our local
action for $\alpha, \alpha \ast B$ is on the true path. By Lemma 2.2.2, $(b, x, \alpha)$ is not discrete and hence infinite.

(v) **Lemma 2.2.4:** $0'' \leq_T \text{Dis}_L(b) = \{c \mid (b, c) \text{ is discrete in } L\}$

*Proof:* We define a function $f : \mathbb{N} \rightarrow T$ by setting $f(e) = \alpha$ if $\alpha$ is the $R_\varepsilon$ strategy on the true path. Notice that $f$ is computable from $\text{Dis}_L(b)$ since $f(0)$ is the unique $R_0$ strategy and by Lemma 2.2.3,

$$f(e + 1) = \begin{cases} f(e) \ast B & \text{if } (b, x, f(e)) \text{ is not discrete} \\ f(e) \ast R & \text{if } (b, x, f(e)) \text{ is discrete} \end{cases}$$

We can compute $0''$ from $f$ using Lemma 2.2.3 since

$$n \in \text{Inf} \text{ if and only if } (b, x, f(n)) \text{ is not discrete}$$

and hence $n \in \text{Inf} \text{ if and only if } f(n + 1) = f(n) \ast B$. Therefore, we have $\text{Inf} \leq_T f$ and $f \leq_T \text{Dis}_L(b)$, so $\text{Inf} \leq_T \text{Dis}_L(b)$.

(vi) **Effective Construction**

The construction is effective because there are only a finite number of things done at each stage for each requirement $R_n$. Also, since there are $n \in \text{Inf}$, we will build at least one infinite $\omega^*$-chain, the construction will use all of the natural numbers. Hence the domain of $L$ is $\mathbb{N}$, which is computable. This implies that to compare $i$ and $j$ in order, we need just run the construction until both elements appear and compare where they land in $L$. 38
Corollary: \( 0'' \leq_T Block_L(b) \)

Proof: By Lemma 2.2.3, if \( \alpha \) is an \( R_e \) strategy on the true path, then \((b, x_\alpha)\) is discrete if and only if \((b, x_\alpha)\) is finite. Therefore, we could equivalently define the function \( f \) in Lemma 3.2.4 by

\[
f(e + 1) = \begin{cases} 
  f(e) \ast B & \text{if } (b, x_{f(e)}) \text{ is infinite} \\
  f(e) \ast R & \text{if } (b, x_{f(e)}) \text{ is finite} 
\end{cases}
\]

Thus, \( f \leq_T Block_L(b) \) and hence \( 0'' \leq_T Block_L(b) \).
Chapter 3
Dense

In this chapter, we construct a computable linear order \( \mathcal{L} \) with a least element \( b \) such that

\[
0'' \leq_T \text{Den}_\mathcal{L}(b) = \{ c \mid (b, c) \text{ is dense in } \mathcal{L} \}.
\]

We first give a simpler construction coding \( 0' \) instead of \( 0'' \).

3.1 Construction I

Recall: We define an interval as dense if it is isomorphic to \( \mathbb{Q} \).

Theorem: There is a computable linear order \( \mathcal{L} \) with least element \( b \) such that

\[
0' \leq_T \text{Den}_\mathcal{L}(b) = \{ c \mid (b, c) \text{ is dense in } \mathcal{L} \}.
\]

Proof: In order to prove this theorem, we want to build a computable linear order \( \mathcal{L} \) around a least element \( b \) such that the interval \((b, x_n)\) is dense if and only if \( n \notin K \). We have the following requirements:

\[
R_n : n \notin K \text{ if and only if } (b, x_n) \in \text{Den}_\mathcal{L}(b)
\]
with ordering \( R_0 < R_1 < R_2 < \ldots \)

The basic strategy for a single requirement \( R_0 \) is to put down a point \( x_0 \) such that

\[
b <_L x_0.\]

Our goal is to do one of two things in the interval \((b, x_0)\) depending on whether \( 0 \in K \) or not. If \( 0 \notin K \), then we want to make the open interval \((b, x_0)\) isomorphic to \( \mathbb{Q} \). This action makes \((b, x_0)\) dense. If \( 0 \in K \), then we want to make \((b, x_0)\) not dense.

To accomplish this goal, at each stage \( s \), we check whether \( 0 \in K_s \). If not, then we add new points between each point in the interval \((b, x_0)\).

\[
b <_L \text{new} <_L w_k <_L \text{new} <_L \ldots <_L \text{new} <_L w_1 <_L \text{new} <_L w_0 <_L \text{new} <_L x_0
\]

In this case, we regard \( R_0 \) as a building state requirement and in the general construction, we will be taking the \( B \) outcome (for building). Since we will repeat this process many times, we introduce the following terminology. Let \((u, v)\) be a finite interval in \( \mathcal{L} \) at stage \( s \). To partially densify \((u, v)\) means to add a new least element and a new greatest element to this open interval and to add one new point between each pair of points in \((u, v)\) which are currently successors. Notice that if a fixed interval \((u, v)\) is partially densified infinitely
often, then \((u,v)\) has order type \(\mathbb{Q}\).

On the other hand, if \(0 \in K_s\), then we want to stop building our \(\mathbb{Q}\) and restrain the interval \((b,x_0)\) from becoming dense. To do this, we want to add two points \(b < z_0 < y_0 < x_0\) as immediate predecessors of \(x_0\) and not allow any points to enter interval \((z_0,x_0)\). If we maintain this restraint, then \(z_0\) will be an immediate predecessor of \(y_0\) and hence \((b,x_0)\) will not be dense. We regard \(R_0\) as a restraining state requirement. In the general construction, we will be taking the \(R\) outcome (for restraining).

To handle a second requirement \(R_1\), we need a witness point \(x_1\). The placement of this points depends on the action of \(R_0\). As long as \(R_0\) is in the building state, we are working under the assumption that \((b,x_0)\) will be dense in the limit and therefore, we want to protect the interval \((b,x_0)\). Thus, we place the point \(x_1\) as follows:

\[ b <_L x_0 <_L x_1 \]

The requirement \(R_1\) now works exactly as \(R_0\) did. As long as \(1 \notin K_s\), \(R_1\) continues to partially densify \((b,x_1)\), making this interval isomorphic to \(\mathbb{Q}\). Notice that if \(0 \notin K\) and \(1 \notin K\), then the action of \(R_1\) towards making \((b,x_1)\) isomorphic to \(\mathbb{Q}\) does not injure the action of \(R_0\) towards making \((b,x_0)\) isomorphic to \(\mathbb{Q}\). If \(1 \in K_s\), then \(R_1\) restrains \((b,x_1)\) by not allowing this interval to become dense by
placing two points $x_0 < \mathcal{L} z_1 < \mathcal{L} y_1 < \mathcal{L} x_1$ as immediate predecessors of $x_1$ and not allowing any points to enter between $[z_1, x_1]$. Since we have $x_0 < \mathcal{L} z_1 < \mathcal{L} y_1$, the requirement $R_0$ can make $(b, x_0)$ dense while the requirement $R_1$ can make $(b, x_1)$ not dense by making $z_1$ and immediate predecessor of $y_1$.

However, consider what happens if $R_0$ changes to the restraining state. In this case, $R_0$ adds two points $b < z_0 < y_0 < x_0$ as immediate predecessors of $x_0$ and does not allow any points to enter between $(z_0, y_0)$ which will make sure that $(b, x_0)$ is not dense. Therefore, $R_1$ needs to stop partially densifying its current interval $(b, x_1)$ since this action adds points in the interval $(z_0, y_0)$.

In this situation, $R_1$ adds a new witness point (or chooses one in the interval $(b, z_0)$) $x^*_1$ and places it such that

$$b < \mathcal{L} x^*_1 < \mathcal{L} z_0 < \mathcal{L} y_0 < \mathcal{L} x_0 < \mathcal{L} \text{ finite} < \mathcal{L} x_1.$$  

$R_1$ can now proceed as before using the interval $(b, x^*_1)$. Notice that if $0 \in K$ and $1 \in K$, then $R_0$ makes $(b, x_0)$ not dense with the witnesses $z_0 < \mathcal{L} y_0$ and $R_1$ makes $(b, x_1)$ not dense with witnesses $z^*_1 < \mathcal{L} y^*_1$. Thus, $(b, x_0)$ and $(b, x^*_1)$ are both not dense, winning $R_0$ and $R_1$.

Notice that with two requirements, we need to know the outcome of $R_0$ in order to know which interval in $\mathcal{L}$ codes information about whether $1 \in K$. To use
{c | (b, c) is dense} to compute K, we proceed as follows. First, we need to ask if
(b, x_0) is dense. If the interval is dense, then we know that 0 \not \in K and that the
witness for R_1 is x_1. So, we ask if (b, x_1) is dense. If so, then 1 \not \in K and if not,
then 1 \in K.

On the other hand, if (b, x_0) is not dense, then we know that 0 \in K. So, at some
finite point in the construction, we switched our witness for R_1 to \*x_1. Therefore,
to determine if 1 \in K, we need to ask if (b, \*x_1) is dense. If it is dense, then 1 \not \in K
and if it is not dense, 1 \in K.

The witness x_2 is set down based upon the restrictions of the higher priority
requirements R_0 and R_1.

- If 0 \not \in K and 1 \not \in K, then x_2 is set down such that b <_{\mathcal{L}} x_0 <_{\mathcal{L}} x_1 <_{\mathcal{L}} x_2.

- If 0 \not \in K and 1 \in K, then x_2 is set down such that b <_{\mathcal{L}} x_0 <_{\mathcal{L}} x_2 <_{\mathcal{L}} x_1.

- If 0 \in K and 1 \not \in K, then x_2 is set down such that b <_{\mathcal{L}} x_1 <_{\mathcal{L}} x_2 <_{\mathcal{L}} x_0.

- If 0 \in K and 1 \in K, then x_2 is set down such that b <_{\mathcal{L}} x_2 <_{\mathcal{L}} x_1 <_{\mathcal{L}} x_0.

The rest of the witnesses are set down based upon the higher priority require-
ments.

Notice that, as described above for R_0 and R_1, in order to determine which
interval in \mathcal{L} codes information about whether 2 \in K, we need to know the out-
comes for R_0 and R_1. The answer to the question of whether (b, x_0) is dense tells
us which witness for \( R_1 \) codes the information about whether \( 1 \in K \). Once we know which witness codes this information, we can ask a denseness question to determine which witness for \( R_2 \) codes information about whether \( 2 \in K \). In general, to determine which witness codes information about whether \( n \in K \), we will have to use denseness questions to determine the correct witness for \( 0, 1, ..., n - 1 \). This process illustrates why our reduction is a Turing reduction as opposed to an \( m \)-reduction.

We will be setting up a tree of strategies \( T = \{R, B\}^{<\omega} \) such that \( R <_L B \). The basic universal strategy is to stay in a build state until \( n \) is enumerated into \( K \). During this time, we will be building \( Q \) between the witness \( x_n \) and \( b \). When \( n \) is enumerated into \( K \), we want to switch to the restrain strategy which will protect the interval \((b, x_n)\) from being dense by inserting \( z_n < y_n < x_n \) as immediate predecessors of \( x_n \) and restraining any new elements from entering \((z_n, y_n)\).

It remains to describe where \( \alpha \) places its witness point \( x_\alpha \) when it is first eligible to act. To describe this placement, we treat the point \( x_\alpha \) as an entity \( w_\alpha \) (to keep a similar notation as in the Discrete Construction Proofs) and write

\[
w_\alpha <_L w_\beta
\]

as a notation for \( x_\alpha <_L x_\beta \). For distinct strategies \( \alpha \) and \( \beta \), we place \( w_\alpha <_L w_\beta \) if and only if either
• \( \alpha <_L \beta \) (\( \alpha \) is to the left of \( \beta \) in the tree of strategies)

• or \( \alpha \subseteq \beta \) and \( \beta(|\alpha|) = B \) (\( \beta \) extends \( \alpha \cdot B \))

• or \( \beta \subseteq \alpha \) and \( \alpha(|\beta|) = R \) (\( \alpha \) extends \( \beta \cdot R \)).

Local Action for \( \alpha \) for \( R_e \):

(i) When \( R_e \) is first eligible to act, place a new witness \( x_\alpha \) in \( \mathcal{L} \) or choose an existing point that satisfies the ordering conditions above.

(ii) If \( e \notin K_s \), then partially densify the interval \((b, x_\alpha)\) and take outcome \( \alpha \cdot B \).

(iii) If \( s \) is the least stage such that \( e \in K_s \), add \( z_\alpha \) and \( y_\alpha \) as immediate predecessors of \( x_\alpha \) and take outcome \( \alpha \cdot R \). If \( z_\alpha \) and \( y_\alpha \) have already been added, just take outcome \( \alpha \cdot R \).

Note a property of the placement of points in our linear order \( \mathcal{L} \). Only \( \alpha \) is allowed to place \( z_\alpha \) and \( y_\alpha \) into \((b, x_\alpha)\). This protects our interval against other witnesses encroaching on its territory and making it dense.

Construction

Stage 0: We begin with the empty set. So, we need to set down point \( b \).

Stage \( s+1 \): Follow the path down the tree of strategies to level \( s \) as directed by the action of the strategies eligible to act. When we reach level \( s + 1 \), end the
stage.

**Verification:**

(i) **True Path**

The true path in our tree of strategies is the leftmost path visited infinitely often. Notice that if $\alpha$ is on the true path, then either $\alpha$ always takes outcome $\alpha \ast B$ or at some stage $s$, $\alpha$ switches to outcome $\alpha \ast R$ and always takes $\alpha \ast R$ at all future stages. Therefore, as the construction proceeds, the paths taken only move left and a node $\alpha$ at level $n$ is on the true path if and only if $\alpha$ is eventually on the path at every stage past some state $s$.

(ii) **Lemma 3.1.1**: Let $\alpha$ be on the true path. If $\alpha \ast R$ is on the true path, then $(b, x_\alpha)$ is not dense.

*Proof*: Fix $t$ such that $\alpha \ast R$ is first on the true path at stage $t$ and hence is on the path at stage $s$ for all $s \geq t$. At stage $t$, $\alpha$ places the points $y_\alpha$ and $z_\alpha$ such that $z_\alpha \prec L y_\alpha \prec L x_\alpha$ and $(z_\alpha, x_\alpha) = \{y_\alpha\}$. To show that $(b, x_\alpha)$ is not dense, it suffices to show that no strategy can add points to $[z_\alpha, x_\alpha]$ after stage $t$. There are two possibilities:

(a) $\alpha$ places points into $[z_\alpha, x_\alpha]$ after stage $t$.

If we have taken the outcome $R$ at stage $t$, then we know that $n$ entered $K$ by stage $t$. By the construction, there is no possibility for points to enter $[z_\alpha, x_\alpha]$ because we cannot return to the building outcome.
(b) Another strategy $\beta$ places points into $[z_\alpha, x_\alpha]$. In this case, we must have $w_\alpha <_L w_\beta$. Consider the ways in which this could happen.

- If $\beta \subseteq \alpha$, then $w_\alpha <_L w_\beta$ means $\alpha(|\beta|) = R$ and hence $\beta * R \subseteq \alpha$.

  Therefore, $\beta * R$ is on the true path and is on the current path at all stages $s \geq t$. By the action at $\beta$, $\beta$ does not add any new points to $\mathcal{L}$.

- If $\alpha \subseteq \beta$, then $w_\alpha <_L w_\beta$ means $\beta(|\alpha|) = B$ and hence $\alpha * B \subseteq \beta$.

  However, $\alpha * B$ is never on the path after stage $t$, so $\beta$ is never eligible to act after stage $t$. Therefore, $\beta$ cannot add new points to $\mathcal{L}$.

- If $\alpha$ and $\beta$ are incomparable, then $w_\alpha <_L w_\beta$ means $\alpha <_L \beta$. Since our path only moves left and $\alpha$ is on the true path, $\beta$ is never eligible to act after stage $t$ and never adds any new points to $\mathcal{L}$ after stage $t$.

In all cases, we see that no strategy $\beta \neq \alpha$ can add new points to $[z_\alpha, x_\alpha]$ after stage $t$.

(iii) **Lemma 3.1.2**: Let $\alpha$ be on the true path. If $\alpha * B$ is on the true path, then $(b, x_\alpha)$ is dense.

**Proof**: If $\alpha * B$ is on the true path, then it is eligible to act infinitely often and it adds points to make $(b, x_\alpha)$ isomorphic to $\mathbb{Q}$. Hence, $(b, x_\alpha)$ is dense.
Lemma 3.1.3: Let $\alpha$ be an $R_e$ strategy on the true path. Then, $(b,x_\alpha)$ is dense if and only if $e \notin K$ if and only if $\alpha \ast B$ is on the true path.

Proof: If $e \notin K$, then by our local action for $\alpha$, $\alpha \ast B$ is on the true path. By the Lemma 3.1.2, $(b,x_\alpha)$ is dense. If $e \in K$, then $\alpha \ast R$ is on the true path and by Lemma 3.1.1, $(b,x_\alpha)$ is not dense.

Lemma 3.1.4: $0' \leq_T Den_L(b) = \{c \mid (b,c) \text{ is dense in } L\}$

Proof: We define a function $f : \mathbb{N} \rightarrow T$ by setting $f(e) = \alpha$ if $\alpha$ is the $R_e$ strategy on the true path. Notice that $f$ is computable from $Den_L(b)$ since $f(0)$ is the unique $R_0$ strategy and by Lemma 3.1.3,

$$f(e+1) = \begin{cases} f(e) \ast B & \text{if } (b,x_{f(e)}) \text{ is dense} \\ f(e) \ast R & \text{if } (b,x_{f(e)}) \text{ is not dense} \end{cases}$$

We can compute $0'$ from $f$ using Lemma 3.1.3 since

$n \in K$ if and only if $(b,x_{f(n)})$ is not dense.

and hence $n \in K$ if and only if $f(n+1) = f(n) \ast R$. Therefore, we have $K \leq_T f$ and $f \leq_T Dis$, so $K \leq_T Dis$.

Effective Construction

The construction is effective because there are only a finite number of things done at each stage for each requirement $R_n$. Also, since there are $n \notin K$, we will build at least one infinite $Q$, the construction will
use all of the natural numbers. Hence the domain of $L$ is $\mathbb{N}$, which is computable. This implies that to compare $i$ and $j$ in order, we need just run the construction until both elements appear and compare where they land in $L$. 
3.2 Construction II

**Theorem:** There is a computable linear order $\mathcal{L}$ with least element $b$ such that

$$0'' \preceq_T Den_{\mathcal{L}}(b) = \{ c \mid (b, c) \text{ is dense in } \mathcal{L} \}.$$ 

**Proof:** In order to prove this theorem, we want to build a computable linear order $\mathcal{L}$ around a least element $b$ such that the interval $(b, x_n)$ is dense if and only if $n \in \text{Inf}$ where $\text{Inf} = \{ e \mid W_e \text{ is infinite} \}$. We have the following requirements:

$$R_n : n \in \text{Inf} \text{ if and only if } (b, x_n) \in \text{Den}_{\mathcal{L}}$$

with ordering $R_0 < R_1 < R_2 < ...$

The basic strategy for a single requirement $R_0$ is similar to the strategy in the $0'$ construction. We want to put down an $x_0$ such that

$$b <_{\mathcal{L}} x_0.$$ 

Our goal is to do one of two things in the interval $(b, x_0)$ depending on whether $0 \in \text{Inf}$ or not. If $0 \in \text{Inf}$, then we want to make the open interval $(b, x_0)$ isomorphic to $\mathbb{Q}$. This action makes $(b, x_0)$ dense. If $0 \notin \text{Inf}$, then we want to make $(b, x_0)$ not dense.

To accomplish this goal, at each stage $s$, we check whether $W_0$ adds a new point. If so, then we partially densify $(b, x_0)$ by adding new points in the interval $(b, x_0)$. 

51
In this case, we regard $R_0$ as a building state requirement and in the general construction, we will be taking the $B$ outcome (for building).

On the other hand, if $W_0$ does not add a new point, then we want to stop building (at least temporarily) our copy of $Q$ and restrain the interval $(b, x_0)$ from densifying. To do this, we want to add two points $b < z_0 < y_0 < x_0$ as immediate predecessors of $x_0$ and not allow any points to enter between $[z_0, x_0]$. We regard $R_0$ as a restraining state requirement. In the general construction, we will be taking the $R$ outcome (for restraining). However, if we see $W_0$ get a new element at a later stage, we initialize $z_0$ and $y_0$ in the sense that we forget that these points had any special significance and we regard the parameters $y_0$ and $z_0$ as undefined. When we partially densify $(b, x_0)$, we treat the points formally labeled by $y_0$ and $z_0$ as any other points in $(b, x_0)$ and add a new point between them.

We will be setting up a tree of strategies $T = \{B, R\}^\omega$ such that $B <_L R$. The basic universal strategy is to stay in a restrain state until $W_n$ adds a new point. In the restrained stage, we will not allow any new points to be introduced in the interval $[z_n, x_n]$. When $W_n$ grows, we want to switch to the build strategy. In this strategy, we will forget about any $y_n$ and $z_n$ designation and continue to build a copy of $Q$ between $b$ and $x_n$. 

52
It remains to describe where $\alpha$ places its witness point $x_{\alpha}$ when it is first eligible to act. To describe this placement, we treat $x_{\alpha}$ as an entity $w_{\alpha}$ (to keep with previous notation) and write

$$w_{\alpha} <_L w_{\beta}$$

as notation for $x_{\alpha} <_L x_{\beta}$. For distinct strategies $\alpha$ and $\beta$, we place $w_{\alpha} <_L w_{\beta}$ if and only if either

- $\beta <_L \alpha$ ($\beta$ is to the left of $\alpha$ in the tree of strategies)
- or $\alpha \subseteq \beta$ and $\beta(|\alpha|) = B$ ($\beta$ extends $\alpha * B$)
- or $\beta \subseteq \alpha$ and $\alpha(|\beta|) = R$ ($\alpha$ extends $\beta * R$).

**Local Action for $\alpha$ for $R_{e}$:**

(i) When $R_{e}$ is first eligible to act, place a new witness $x_{\alpha}$ in $L$ (or choose a point $x_{\alpha}$ in $L$ satisfying the order conditions above).

Let $\hat{s}$ be the last stage at which $\alpha$ was eligible to act (with $\hat{s} = 0$ if this is the first time $\alpha$ is eligible to act).

(ii) If $W_{e,\hat{s}} \neq W_{e,\hat{s},\alpha}$, then initialize $y_{\alpha}$ and $z_{\alpha}$ (if they are defined), partially densify $(b, x_{\alpha})$, and take outcome $\alpha * B$.

(iii) If $W_{e,\hat{s}} = W_{e,\hat{s},\alpha}$, add points $z_{\alpha}$ and $y_{\alpha}$ as immediate predecessors of $x_{\alpha}$ (unless they are already defined) and take outcome $\alpha * R$. 

53
Note a property of the placement of points in our linear order \( L \): only \( \alpha \) is allowed to place \( z_\alpha \) and \( y_\alpha \) into the interval \((b, x_\alpha)\). This protects our interval against other witnesses encroaching on its territory.

**Construction**

**Stage 0:** We begin with the empty set. So, we need to set down point \( b \).

**Stage \( s+1 \):** Follow the path down the tree of strategy to level \( s \) as directed by the action of the strategies eligible to act. When we reach level \( s + 1 \), end the stage.

**Verification:**

(i) **True Path**

First let \( s \) be an \( \alpha \)-stage if \( \alpha \) is eligible to act at stage \( s \). The true path in our tree of strategies is the leftmost path visited infinitely often. If \( \alpha \) is on the true path, then either there are infinitely many \( \alpha \)-stages when we take \( \alpha \ast B \) and \( \alpha \ast B \) is on the true path, or there exists a stage \( t \) such that for all \( \alpha \)-stages after \( t \), we take \( \alpha \ast R \) and \( \alpha \ast R \) is on the true path. Note that if \( \alpha \) is on the true path, then there exists only finitely many stages \( s \) when the true path is to the left of \( \alpha \).
Lemma 3.2.1: Let $\alpha$ be on the true path. If $\alpha \ast R$ is on the true path, then $(b, x_\alpha)$ is not dense.

Proof: Fix the least stage $t$ such that $\alpha \ast R$ is on the path at stage $t$ and the path is never to the left of $\alpha \ast R$ after $t$. At stage $t$, $\alpha$ defines $y_\alpha$ and $z_\alpha$ and places them so that $z_\alpha \prec_L y_\alpha \prec_L x_\alpha$ and $(z_\alpha, x_\alpha) = \{y_\alpha\}$. Since $\alpha$ never takes outcome $B$ after stage $t$, these witnesses $y_\alpha$ and $z_\alpha$ are never initialized by $\alpha$. Therefore they remain defined forever. To prove that $(b, x_\alpha)$ is not dense, it suffices to show that no strategy can add points to $[z_\alpha, x_\alpha]$ after stage $t$. There are two possibilities.

(a) $\alpha$ places points in $(z_\alpha, x_\alpha)$ after stage $t$.

If $\alpha$ places a point in $[z_\alpha, x_\alpha]$, then it takes outcome $\alpha \ast B$. Since $\alpha \ast B \prec_L \alpha \ast R$ and the path is never left of $\alpha \ast R$ after stage $t$, $\alpha$ cannot place any points in $[z_\alpha, x_\alpha]$ after stage $t$.

(b) Another strategy $\beta$ places points in $[z_\alpha, x_\alpha]$. In this case, we must have $w_\alpha \prec_L w_\beta$. Consider the ways this could happen.

(i) If $\beta \subseteq \alpha$, then $w_\alpha \prec_L w_\beta$ means $\alpha(|\beta|) = R$ and hence $\beta \ast R \subseteq \alpha$. If $\beta$ adds points to $L$, it takes outcome $\beta \ast B$ which is left of $\alpha \ast R$. Therefore, $\beta$ adds no more points after stage $t$.

(ii) If $\alpha \subseteq \beta$, then $w_\alpha \prec_L w_\beta$ means $\beta(|\alpha|) = B$ and hence $\alpha \ast B \subseteq \beta$. Since $\alpha$ takes outcome $R$ at all $\alpha$-stages after $t$, $\beta$ is never eligible to act after stage $t$ and hence does not add any points after stage $t$. 

55
(iii) If \( \alpha \) and \( \beta \) are incomparable, then \( w_\alpha <_L w_\beta \) means \( \beta <_L \alpha \). However, the path is never to the left of \( \alpha * R \) after stage \( t \) and therefore, never to the left of \( \alpha \) after stage \( t \). Hence, \( \beta \) cannot add points after stage \( t \).

In all cases, we see that no strategy \( \beta \neq \alpha \) can add new points to \([z_\alpha, x_\alpha]\) after stage \( t \).

(iii) **Lemma 3.2.2**: Let \( \alpha \) be on the true path. If \( \alpha * B \) is on the true path, then \((b, x_\alpha)\) is dense.

*Proof*: If \( \alpha * B \) is on the true path, then it is eligible to act infinitely often and it adds points to make \((b, x_\alpha)\) isomorphic to \( Q \). Hence, \((b, x_\alpha)\) is dense.

(iv) **Lemma 3.2.3**: Let \( \alpha \) be an \( R_e \) strategy on the true path. Then, \((b, x_\alpha)\) is not dense if and only if \( e \in \text{Inf} \) if and only if \( \alpha * R \) is on the true path.

*Proof*: If \( e \notin \text{Inf} \), then by our local action for \( \alpha \), \( \alpha * R \) is on the true path. By the Lemma 3.2.1, \((b, x_\alpha)\) is not dense. If \( e \in \text{Inf} \), then \( \alpha * B \) is on the true path and by Lemma 3.2.2, \((b, x_\alpha)\) is dense.

(v) **Lemma 3.2.4**: \( 0'' \leq_T \text{Den}_{L}(b) = \{ c \mid (b, c) \text{ is dense in } L \} \)

*Proof*: We define a function \( f : \mathbb{N} \to T \) by setting \( f(e) = \alpha \) if \( \alpha \) is the \( R_e \) strategy on the true path. Notice that \( f \) is computable from \( \text{Den}_{L}(b) \) since \( f(0) \) is the unique \( R_0 \) strategy and by Lemma 3.2.3,

\[
f(e + 1) = \begin{cases} 
  f(e) * B & \text{if } (b, x_{f(e)}) \text{ is dense} \\
  f(e) * R & \text{if } (b, x_{f(e)}) \text{ is not dense}
\end{cases}
\]
We can compute $0''$ from $f$ using Lemma 3.2.3 since
\[ n \in \text{Inf} \text{ if and only if } (b, x_{f(n)}) \text{ is dense}. \]

and hence $n \in \text{Inf}$ if and only if $f(n + 1) = f(n) \ast B$. Therefore, we have $\text{Inf} \leq_T f$ and $f \leq_T \text{Den}_L(b)$, so $\text{Inf} \leq_T \text{Den}_L(b)$.

(vi) Effective Construction

The construction is effective because there are only a finite number of things done at each stage for each requirement $R_n$. Also, since there are $n \in \text{Inf}$, we will build at least one infinite $Q$, the construction will use all of the natural numbers. Hence the domain of $L$ is $\mathbb{N}$, which is computable. This implies that to compare $i$ and $j$ in order, we need just run the construction until both elements appear and compare where they land in $L$. 

57

