Lecture #6: Introduction to Turing Machines
Completion of Design Example

About This Document

Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. The second video and lecture slides for Lecture #6 considered the problem of designing a Turing machine that decides the language

$$L = \{\omega \in \Sigma^\star \mid \omega \text{ has the same number of } \mathtt{a}\text{'s as } \mathtt{b}\text{'s}\}$$

This documents completes that example — by giving the details needed to get various things started, and then presenting exercises (which can usually be completed by modifying material that has been provided shortly before the exercise) that can be used to complete a proof that the Turing machine M, that has been presented, really *does* decide L.

This is not your grandparent's "Introduction to Computation" course and you will *not* be asked to solve problems on tests or assignments by providing the material that follows — at least, not with this length. However, it *will* be necessary to solve simpler Turing machine design problems later on, in the course, when proving that languages are undecidable (or unrecognizable) — so some experience with Turing machine design may be helpful.

High-Level Description

The lecture material included the algorithm shown in Figure 1 on page 2 and a proof that this algorithm always halts, when executed on an input string $\omega \in \Sigma^\star$ — accepting ω if $\omega \in L$ and rejecting ω if $\omega \notin L$.

On input $\omega \in \Sigma^\star$:

1. Consider all symbols in ω to be ***unmarked***.
2. `while` (there is at least one unmarked symbol) `{`
3. `if` (the leftmost unmarked symbol is `a`) `{`
4. Mark this "leftmost unmarked symbol".
5. `if` (there is also an unmarked copy of `b`) `{`
6. Mark the leftmost copy of `b`
   `} else {`
7. ***reject*** ω
   `}`
   `} else {`
8. Mark the leftmost symbol (which must be `b`)
9. `if` (there is also an unmarked copy of `a`) `{`
10. Mark the leftmost copy of `a`
    `} else {`
11. ***reject*** ω
    `}`
    `}`
    `}`
12. ***accept*** ω

Figure 1: High-Level Algorithm to Decide Membership in L

Implementation-Level Description

Tape Alphabet

As noted in the lecture material, this algorithm can be implemented using a tape alphabet that includes two additional tape symbols, along with the symbols in the input alphabet, Σ, and the "blank" symbol, $\sqcup$:

- The symbol `X` is used to mark the leftmost cell of the tape: The symbol is written at the beginning of the first sweep right on the tape during the computation (if the input string is not accepted right away) and is never replaced at this cell. This symbol is not written on any other cell of the tape.
- The symbol `x` is used to "mark" the symbols on the input string after the first one (so that

is used on the cells to the rightmost cell of the tape).

Thus the tape alphabet, for the Turing machine being designed, is the alphabet

$$\Gamma = \Sigma \cup \{\sqcup, \mathtt{X}, \mathtt{x}\} = \{\mathtt{a}, \mathtt{b}, \mathtt{X}, \mathtt{x}, \sqcup\}/ \tag{1}$$

A More Detailed Algorithm

An "implementation-level" algorithm is shown (as pseudocode) in Figure 2 on page 4 and Figure 3 on page 5.

The high-level algorithm in Figure 1 and the implementation-level algorithm in Figures 2 and 3 are related as follows.

1. There is nothing to do, to implement the step at line 1 in the algorithm in Figure 1 — this is, effectively, just stating that none of the symbols on the tape have been replaced with copies of $\mathtt{X}$ or $\mathtt{x}$ when the computation begins.

2. It is necessary to restructure the code, when moving from the high-level algorithm to the interpretation-level description, because something different happens the first time that a pair of symbols is matched than what happens later on ($\mathtt{X}$ is used to mark a symbol instead of $\mathtt{x}$). The test at line 2 of the algorithm in Figure 1 is whether an unmarked symbol, either $\mathtt{a}$ or $\mathtt{b}$, is visible on the tape. This corresponds to the following steps and tests in the implementation-level description:

   - The test, "Is $\sqcup$ visible", at line 1 is effectively checking whether there are no unmarked symbols because there are no non-blank symbols at all — so that the loop test (at line 2 in the high-level algorithm) fails if the test at line 1 of the implementation-level description is passed.

     On the other hand, the loop test (in the high level algorithm) must pass the first time it is checked, and the loop body must be executed at least once, if the test at line 1 in the implementation-level description fails.

     Other executions of the test at line 2 in the high level algorithm are carried out using the steps at line 18 and 19 in the implementation-level description, along with the tests at lines 20 and 28. If either the test at 20 or 28 is passed, then the loop test at line 1 in the high-level algorithm is also being passed (and the loop body is executed again). If both of these tests are failed then there are no unmarked symbols left on the tape, so that the loop test at line 1 in the high-level algorithm is being failed as well.

3. The test at line 3 (checking whether the leftmost unmarked symbol is $\mathtt{a}$) is implemented using the tests at lines 3 (corresponding to the first execution of this test) and at line 20

On input $\omega \in \Sigma^\star$ {

1. `if` ($\sqcup$ is visible) {
2. Write $\sqcup$, moving right, and `accept`

   }
3. `if` (`a` is visible) {
4. Write `X`, moving right.
5. Move right, without changing any symbols, until either `b` or $\sqcup$ is visible.
6. `if` (`b` is visible) {
7. Write `x`, moving left.
8. Moving left, without changing any symbols, until `X` is visible.
9. Write `X`, moving right.

   } `else` {
10. Write $\sqcup$, moving right, and `reject`.

   }

   } `else` {
11. Write `X`, moving right.
12. Move right, without changing any symbols, until either `a` or $\sqcup$ is visible.
13. `if` (`a` is visible) {
14. Write `x`, moving left.
15. Moving left, without changing any symbols, until `X` is visible.
16. Write `X`, moving right.

   } `else` {
17. Write $\sqcup$, moving right, and `reject`.

   }

   }

Figure 2: Implementation-Level Description of Turing Machine Deciding L — Beginning

(corresponding to all subsequent executions of this test) in the implementation-level description of the Turing machine — because the tape head has been moved to the leftmost unmarked symbol on the tape, when these tests in the implementation-level description are reached.

4. The step at line 4 in the high-level algorithm is implemented using the step at line 4 in

```
18.  while (true) {
19.    Move right, without changing any symbols, until either a, b, or ⊔ is
       visible.
20.    if (a is visible) {
21.      Write x, moving right.
22.      Move right, without changing any symbols, until either b or ⊔ is
         visible.
23.      if (b is visible) {
24.        Write x, moving left.
25.        Move left, without changing any symbols, until X is visible.
26.        Write X, moving right.
         } else {
27.        Write ⊔, moving right, and reject.
         }
28.    } else if (b is visible) {
29.      Write x, moving right.
30.      Move right, without changing any symbols, until either a or ⊔ is
         visible.
31.      if (a is visible) {
32.        Write x, moving left.
33.        Move left, without changing any symbols, until X is visible.
34.        Write X, moving right.
         } else {
35.        Write ⊔, moving right, and reject.
         }
       } else {
36.      Write ⊔, moving right, and accept.
       }
     }
```

Figure 3: Implementation-Level Description of a Turing Machine Deciding L — End

the implementation-level description, for the first execution of the step in the high-level algorithm, and using the step at line 21 in the implementation-level description, for all subsequent executions of this step in the high-level algorithm.

5. The test at line 5 in the high-level algorithm is implemented by the steps at lines 5 and 6 in

the implementation-level description, for the first execution of this test, and by the steps at lines 22 and 23 in the implementation-level description, for all subsequent executions of this test: The executions of the steps at line 5 and 22 move the tape head right, until either the symbol being looked for is found, or $\sqcup$ is reached (indicating that the desired symbol is not on the tape at all). The tests at lines 6 and 23 then check whether the desired symbol has been found.

6. The step at line 6, in the high-level algorithm, is executed when an unmarked copy of `b` — matching a previously marked copy of `a` — has been found. This is implemented by the step at line 7 in the implementation-level description, for the first execution of the step in the high-level algorithm, and by the step at line 24 in the implementation-level description, for all subsequent executions of the step in the high-level algorithm.

   Now, in order to be ready for the next execution of the loop test, it is necessary to move the tape head back to the left — and it is sufficient to move it to the cell immediately to the *right* of the leftmost cell on the tape — which is the leftmost cell that might store an unmarked symbol. This is carried out using the steps at line 8 and 9 in the implementation-level description, for the first execution of the loop body in the high-level algorithm, and using the steps at line 25 and 26 in the implementation-level description, for all subsequent executions of the loop body in the high-level algorithm.

7. The step at line 7, in the high-level algorithm, is executed when a copy of `a` has been found and marked, but it was not possible to find an unmarked copy of `b` to match it. This is implemented by the step at line 10 in the implementation-level description, for the first execution of the loop-body in the high-level algorithm, and by the step at line 27 in the implementation-level description, for all subsequent executions of the loop body in the high-level algorithm.

8. The step at line 8 in the high-level algorithm is implemented using the step at line 11 in the implementation-level description, for the first execution of the step in the high-level algorithm, and using the step at line 29 in the implementation-level description, for all subsequent executions of this step in the high-level algorithm.

9. The test at line 9 in the high-level algorithm is implemented by the steps at lines 12 and 13 in the implementation-level description, for the first execution of this test, and by the steps at lines 30 and 31 in the implementation-level description, for all subsequent executions of this test: The executions of the steps at lines 12 and 30 move the tape head right, until either the symbol being looked for is bound, or $\sqcup$ is reached (indicating that the desired symbol is not on the tape at all). The tests at lines 13 and 31 then check whether the desired symbol has been found.

10. The step at line 10, in the high-level algorithm, is executed when an unmarked copy of `a` — matching a previously marked copy of `b` — has been found. This is implemented by

the step at line 14 in the implementation-level description, for the first execution of the step in the high-level algorithm, and by the step at line 32 in the implementation-level description, for all subsequent executions of the step in the high-level algorithm.

In order to be ready for the next execution of the loop test, it is necessary to move the tape head back to the left — and it is sufficient to move it to the cell immediately the the *right* of the leftmost cell on the tape — which is the leftmost cell that might store an unmarked symbol. This is carried out using the steps at lines 15 and 16 in the implementation-level description, for the first execution of the loop body in the high-level algorithm, and by the steps at lines 33 and 34 in the implementation-level description, for all subsequent executions of the loop body in the high-level algorithm.

11. The step at line 11 in the high-level algorithm is executed when a copy of `b` has been found and marked, but it was not possible to find an unmarked copy of `a` to match it. This is implemented by the step at line 17 in the implementation-level description, for the first execution of the loop-body in the high-level algorithm, and by the step at line 35 in the implementation-level description, for all subsequent executions of the loop body in the high-level algorithm.

12. Finally, the step at 12 in the high-level algorithm is executed when it has been determine that that there are no unmarked symbols left at all, so that the input should be accepted. This is implemented by the step at line 2 of the implementation-level description (used when the input string is the empty string), as well as the step at line 36 in the implementation-level algorithm (which is reached whenever the input is a non-empty string that belongs to L).

Correctness of the implementation-level description can be established by arguing that it is an accurate implementation of the high-level algorithm — which simply adds details about how the use of `X` to mark the first symbol in a non-empty input string, while `x` is used to mark all other symbols in the input string, along with details about how the tape head must be moved as steps are carried out.

Formal Description

The implementation-level description can now be used to obtain a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

where

- $Q = \{q_0, q_{\texttt{a},1}, q_{\texttt{b},1}, q_1, q_{\texttt{a},2}, q_{\texttt{b},2}, q_2, q_3, q_{\text{accept}}, q_{\text{reject}}\}$,

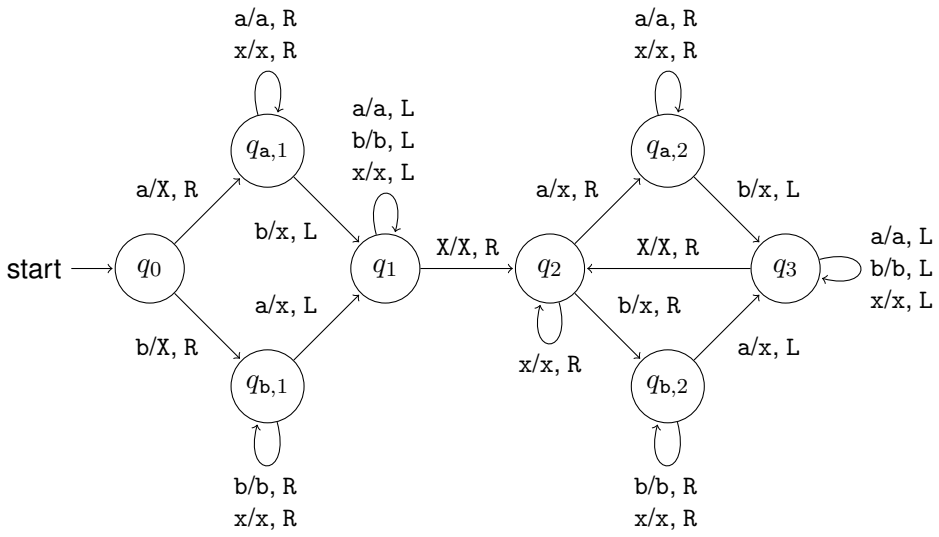

Figure 4: A Turing Machine that Decides L

- $\Sigma = \{\mathtt{a}, \mathtt{b}\}$,
- $\Gamma = \{\mathtt{a}, \mathtt{b}, \mathtt{x}, \mathtt{X}, \sqcup\}$, and
- an *incomplete* transition diagram is as shown in Figure 4 on page 8.

Missing transitions are as follows.

- Missing transitions for $\sqcup$ out of q_0 and q_2 go to q_{accept}, moving tight. That is,

$$\delta(q, \sqcup) = (q_{\text{accept}}, \sqcup, \mathtt{R}) \quad \text{for } q \in \{q_0, q_2\}.$$

- Missing transition for $\sqcup$ out of $q_{\mathtt{a},1}$, $q_{\mathtt{b},1}$, $q_{\mathtt{a},2}$ and $q_{\mathtt{b},2}$ go to q_{reject}, moving right. That is,

$$\delta(q, \sqcup) = (q_{\text{reject}}, \sqcup, \mathtt{R}) \quad \text{for } q \in \{q_{\mathtt{a},1}, q_{\mathtt{b},1}, q_{\mathtt{a},2}, q_{\mathtt{b},2}\}.$$

- M never re-enters state q_0 after the first time it leaves it (when seeing either a symbol in Σ or $\sqcup$), so transitions for $\mathtt{x}$ and $\mathtt{X}$ out of q_0 can never be used. They can, therefore, be set in any way one wants. To keep things consistent and simple, let us set

$$\delta(q_0, \sigma) = (q_{\text{reject}}, \sigma, \mathtt{R}) \quad \text{for } \sigma \in \mathtt{x}, \mathtt{X}\}.$$

- The tape head for M must be pointing to a non-blank symbol during a sweep to the left, whenever it is in either state q_1 or q_3, so transitions for $\sqcup$ out of these states can never be used. They can, therefore, be set in any way one wants to, as well. To keep things consistent and simple, let us set

$$\delta(q, \sqcup) = (q_{\text{reject}}, \sqcup, \mathtt{R}) \quad \text{for } q \in \{q_1, q_3\}.$$

- Similarly, because these states are only used when the tape head is located to the *right* of the leftmost cell of the tape, transitions for $\mathtt{X}$ out of $q_{\mathtt{a},1}$, $q_{\mathtt{b},1}$, q_2, $q_{\mathtt{a},2}$ or $q_{\mathtt{b},2}$ can also never be used. They can also be set in any way one wants. To keep things, and consistent, and simple, let us set

$$\delta(q, \mathtt{X}) = (q_{\text{reject}}, \mathtt{X}, \mathtt{R}) \quad \text{for } q \in \{q_{\mathtt{a},1}, q_{\mathtt{a},2}, q_2, q_{\mathtt{a},2}, q_{\mathtt{b},2}\}.$$

It can be argued that this (formally described) Turing machine is consistent with the implementation-level description, as follows.

1. The test (whether $\sqcup$ is visible) at line 1 of the implementation-level description — as the computation begins — is implemented using transitions out the start state, q_0: The transition for $\sqcup$ out of q_0 is used if the test would be passed, and a transition for either $\mathtt{a}$, $\mathtt{b}$, $\mathtt{x}$ or $\mathtt{X}$ out of q_0 is used if the test would be failed.

2. The step at line 2 of the implementation-level description is also implemented by the transition for $\sqcup$ out of q_0 (in particular, by the *value* in $Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ assigned to $\delta(q_0, \sqcup)$).

3. The test (whether $\mathtt{a}$ is visible) at line 3 of the implementation-level description is also implemented using transitions out of q_0. The transition for $\mathtt{a}$ out of q_0 is used if the test is passed, and (since the possibility that $\sqcup$ is visible has already been eliminated) a transition for either $\mathtt{b}$, $\mathtt{x}$ or $\mathtt{X}$ is used if the test is failed.

4. The step at line 4 of the implementation-level description is also implemented using the transition for $\mathtt{a}$ out of q_0.

5. The step at line 5 of the implementation-level description is implemented using the transitions for $\mathtt{a}$ and $\mathtt{x}$ out of $q_{\mathtt{a},1}$, which keep the Turing machine in state $q_{\mathtt{a}_1}$[1]

6. The test at line 6 of the implementation-level description is implemented using transitions out of $q_{\mathtt{a},1}$: The transition for $\mathtt{b}$ out of $q_{\mathtt{b},1}$ is used if this test is passed, and the transition for $\sqcup$ out of $q_{\mathtt{a},1}$ is used if this test is failed.

[1] The transition for $\mathtt{x}$ out of $q_{\mathtt{a},1}$ is another transition that can never be used. It has been set, as it was, to keep the transitions for $\mathtt{a}$ out of $q_{\mathtt{a},1}$ and $q_{\mathtt{a},2}$ consistent. This is also the case for the transition for $\mathtt{x}$ out of $q_{\mathtt{b},1}$.

7. The step at line 7 of the implementation-level description is also implemented by the transition for $\mathtt{b}$ out of $q_{\mathtt{a},1}$ (in particular, by the *value* in $Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ assigned to $\delta(q_{\mathtt{a},1}, \mathtt{b})$).

8. The step at line 8 of the implementation-level description is implemented by the transitions for $\mathtt{a}$ and for $\mathtt{b}$ out of q_1. (In particular, it is implemented using the *value* in $Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ assigned to $\delta(q_1, \mathtt{a})$ and to $\delta(q_1, \mathtt{b})$).

9. The step at line 9 of the implementation-level description is implemented by the transition for $\mathtt{X}$ out of q_1. (In particular, it is implemented using the *value* in $Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ assigned to $\delta(q_1, \mathtt{X})$).

10. The step at line 10 of the implementation-level description is implemented by the transition for $\sqcup$ out of $q_{\mathtt{a},1}$. (In particular, it is implemented using the *value* in $Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ assigned to $\delta(q_{\mathtt{a},1}, \sqcup)$).

Exercise: Complete the above description by describing how (as many as you need to, of) the steps at lines 11–36 of the implementation-level algorithm are implemented using transitions of the Turing machine M. For the steps with tests, you should be able to identify a set of transitions out of a given state of M whose selection corresponds to the execution of the test. For steps that write symbols onto the tape and move the tape head, you should be able to identify one or more transitions out of a given state, noting how the *values* in $Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ are used to implement the step.

Proving Correctness

Consider, now, the problem of *proving* that the Turing machine M, that has now been designed decides the language L. While some people might find the above information to be convincing, others might see it as too imprecise.

With that noted, one can now proceed by considering the formal description of M to prove useful things about how this Turing machine processes strings.

In particular, one can note that

$$q_0 \vdash \sqcup\, q_{\text{accept}} \qquad \text{(since } \delta(q_0, \sqcup) = (q_{\text{accept}}, \sqcup, \mathtt{R}),)$$

so that M ***accepts*** the empty string, λ. One can also note that

$$\begin{aligned} q_0\,\mathtt{ab} &\vdash \mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{b} && \text{(since } \delta(q_0,\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{X},\mathtt{R})\text{)} \\ &\vdash q_1\,\mathtt{Xx} && \text{(since } \delta(q_{\mathtt{a},1},\mathtt{b}) = (q_1,\mathtt{x},\mathtt{L})\text{)} \\ &\vdash \mathtt{X}\,q_2\,\mathtt{x} && \text{(since } \delta(q_1,\mathtt{X}) = (q_2,\mathtt{X},\mathtt{R})\text{)} \\ &\vdash \mathtt{Xx}\,q_2 && \text{(since } \delta(q_2,\mathtt{x}) = (q_2,\mathtt{x},\mathtt{R})\text{)} \\ &\vdash \mathtt{Xx}\sqcup\,q_{\text{accept}} && \text{(since } \delta(q_2,\sqcup) = (q_{\text{accept}},\sqcup,\mathtt{R})\text{)} \end{aligned}$$

— so that M ***accepts*** the string ab.

Exercise: Use a similar trace of execution to show that M also ***accepts*** the string ba.

One can also note that

$$\begin{aligned} q_0\,\mathtt{a} &\vdash \mathtt{X}\,q_{\mathtt{a},1} && \text{(since } \delta(q_0,\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{X},\mathtt{R})\text{)} \\ &\vdash \mathtt{X}\sqcup\,q_{\text{reject}} && \text{(since } \delta(q_{\mathtt{a},1},\sqcup) = (q_{\text{reject}},\sqcup,\mathtt{R})\text{)} \end{aligned}$$

— so that M ***rejects*** the string a.

Consider, a next, a string $\mathtt{a}^n$ for some positive integer n:

$$q_0\,\mathtt{a}^n \vdash \mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1} \qquad \text{(since } \delta(q_0,\mathtt{a}) = (q_{\mathtt{a}_1},\mathtt{X},\mathtt{R})\text{).}$$

Now, since $\delta(q_{\mathtt{a},1},\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{a},\mathtt{R})$, one can show by induction on i that

$$\mathtt{X}\,q_{a,1}\,\mathtt{a}^{n-1} \vdash^\star \mathtt{Xa}^i\,q_{\mathtt{a},1}\,\mathtt{a}^{n-i-1}$$

for every integer i such that $0 \leq i \leq n-2$. Thus

$$\begin{aligned} q_0\,\mathtt{a}^n &\vdash \mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1} && \text{(since } \delta(q_0,\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{X},\mathtt{R})\text{)} \\ &\vdash^\star \mathtt{Xa}^{n-2}\,q_{\mathtt{a},1}\,\mathtt{a} && \text{(by the above claim, with } i = n-2\text{)} \\ &\vdash \mathtt{Xa}^{n-1}\,q_{\mathtt{a},1} && \text{(since } \delta(q_{\mathtt{a},1},\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{a},\mathtt{R})\text{)} \\ &\vdash \mathtt{Xa}^{n-1}\sqcup\,q_{\text{reject}} && \text{(since } \delta(q_{\mathtt{a},1},\sqcup) = (q_{\text{reject}},\sqcup,\mathtt{R})\text{)} \end{aligned}$$

— so that M also ***rejects*** the string $\mathtt{a}^n$, for every integer n such that $n \geq 2$.

Exercise: Modify the above argument, as needed, to show that M also ***rejects*** the string $\mathtt{b}^n$ for every positive integer n.

It remains only to consider string in $\Sigma^\star$, with length at least three, that include at least one copy of a and at least one copy of b. Let us continue by considering a string $\mathtt{ab}\,\mu$, where μ is a non-empty string in $\Sigma^\star$. M's computation on this string begins as follows.

$$\begin{aligned} q_0\,\mathtt{ab}\mu &\vdash \mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{b}\mu && \text{(since } \delta(q_0,\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{X},\mathtt{R})\text{)} \\ &\vdash q_1\,\mathtt{Xx}\mu && \text{(since } \delta(q_{\mathtt{a},1},\mathtt{b}) = (q_1,\mathtt{x},\mathtt{L})\text{)} \\ &\vdash \mathtt{X}\,q_2\,\mathtt{x}\mu && \text{(since } \delta(q_1,\mathtt{X}) = (q_2,\mathtt{X},\mathtt{R})\text{.)} \end{aligned}$$

Consider, as well, a strong $\mathtt{a}^n\mathtt{b}\mu$, where $n \geq 2$ and $\mu \in \Sigma^\star$.

$$q_0\,\mathtt{a}^n\mathtt{b}\mu \vdash \mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1}\mathtt{b}\mu \qquad \text{(since } \delta(q_0,\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{X},\mathtt{R})\text{)}.$$

Since $\delta(q_{\mathtt{a},1},\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{a},\mathtt{R})$, one can prove by induction on i that

$$\mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1}\mathtt{b}\mu \vdash^\star \mathtt{X}\,\mathtt{a}^i\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1-i}\mathtt{b}\mu$$

for every integer i such that $0 \leq i \leq n-2$. Thus

$$\begin{aligned}
\mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1}\mathtt{b}\mu &\vdash^\star \mathtt{X}\mathtt{a}^{n-2}\,q_{\mathtt{a},1}\,\mathtt{a}\mathtt{b}\mu && \text{(by the above, with } i = n-2\text{)}\\
&\vdash \mathtt{X}\mathtt{a}^{n-1}\,q_{\mathtt{a},1}\,\mathtt{b}\mu && \text{(since } \delta(q_{\mathtt{a},1},\mathtt{a}) = (q_{\mathtt{a},1},\mathtt{a},\mathtt{R})\text{)}\\
&\vdash \mathtt{X}\mathtt{a}^{n-2}\,q_1\,\mathtt{a}\mathtt{x}\mu && \text{(since } \delta(q_{\mathtt{a},1},\mathtt{b}) = (q_1,\mathtt{x},\mathtt{L})\text{)}.
\end{aligned}$$

Since $\delta(q_1,\mathtt{a}) = (q_1,\mathtt{a},\mathtt{L})$, one can prove by induction on i that

$$\mathtt{X}\mathtt{a}^{n-2}\,q_1\,\mathtt{a}\mathtt{x}\mu \vdash^\star \mathtt{X}\mathtt{a}^{n-i-2}\,q_1\,\mathtt{a}^{i+1}\mathtt{x}\mu$$

for every integer i such that $0 \leq i \leq n-2$. Thus

$$\begin{aligned}
\mathtt{X}\mathtt{a}^{n-2}\,q_1\,\mathtt{a}\mathtt{x}\mu &\vdash^\star \mathtt{X}\,q_1\,\mathtt{a}^{n-1}\mathtt{x}\mu && \text{(by the above, with } i = n-2\text{)}\\
&\vdash q_1\,\mathtt{X}\mathtt{a}^{n-1}\mathtt{x}\mu && \text{(since } \delta(q_1,\mathtt{a}) = (q_1,\mathtt{a},\mathtt{L})\text{)}\\
&\vdash \mathtt{X}\,q_2\,\mathtt{a}^{n-1}\mathtt{x}\mu && \text{(since } \delta(q_1,\mathtt{X}) = (q_2,\mathtt{X},\mathtt{R})\text{)}.
\end{aligned}$$

Thus if $n \geq 2$ then

$$\begin{aligned}
q_0\,\mathtt{a}^n\mathtt{b}\mu &\vdash^\star \mathtt{X}\,q_{\mathtt{a},1}\,\mathtt{a}^{n-1}\mathtt{b}\mu && \text{(by the first derivation, above)}\\
&\vdash^\star \mathtt{X}\mathtt{a}^{n-2}\,q_1\,\mathtt{a}\mathtt{x}\mu && \text{(by the second derivation, above)}\\
&\vdash^\star \mathtt{X}\,q_2\,\mathtt{a}^{n-1}\mathtt{x}\mu && \text{(by the third derivation above)}.
\end{aligned}$$

Exercise: Modify the above arguments to show that $q_0\,\mathtt{b}\mathtt{a}\mu \vdash^\star \mathtt{X}\,q_2\,\mathtt{x}\mu$ as well, for any non-empty string $\mu \in \Sigma^\star$, and that $q_0\,\mathtt{b}^n\mathtt{a}\mu \vdash^\star \mathtt{X}\,q_2\,\mathtt{b}^{n-1}\mathtt{x}\mu$ for every integer $n \geq 2$ and for every string $\mu \in \Sigma^\star$.

Now let k be a positive integer and let $\omega \in \Sigma^\star$ such ω includes at least k copies of $\mathtt{a}$ and at least k copies of $\mathtt{b}$. For $1 \leq i \leq k$, let $\mathtt{X}\mu_i$ be the string in $(\Sigma \cup \{\mathtt{X},\mathtt{x}\})^\star$ that is obtained from ω by replacing the leftmost symbol in ω with $\mathtt{X}$ and replacing each of the first i copies and each of the first i copies of $\mathtt{b}$ — except for the leftmost symbol — with $\mathtt{x}$. It follows by above that if ω has length at least three and $k \geq 1$ then

$$q_0\,\omega \vdash^\star \mathtt{X}\,q_2\,\mu_1.$$

Exercise: Prove that if $1 \leq i \leq k-1$, for ω, k, and $\mu_1, \mu_2, \ldots, \mu_k$ as above, then

$$\mathtt{X}\,q_2\,\mu_i \vdash^\star \mathtt{X}\,q_2\,\mu_{i+1}.$$

This can be used by a considering a sequence of cases, and derivations, that resemble those that were used in a proof that $q_0\,\omega \vdash^\star \mathtt{X}\,q_2\,\mu_1$.

The above derivations can be used to prove, by induction on i, that

$$q_0\,\omega \vdash^\star \mathtt{X}\,q_2\,\mu_i$$

for every integer i such that $1 \leq i \leq k$ — so that, in particular, $q_0\,\omega \vdash^\star \mathtt{X}\,q_2\,\mu_k$.

Suppose, next, that $\omega \in \Sigma^\star$ such that ω includes *exactly* k copies of $\mathtt{a}$ and *exactly* k copies of $\mathtt{b}$, for an integer $k \geq 2$. Then $\mu_k = \mathtt{x}^{2k-1}$, so that it follows by the above that

$$q_0\,\omega \vdash^\star \mathtt{X}\,q_2\,\mathtt{x}^{2k-1}.$$

Exercise: Show that $\mathtt{X}\,q_2\,\mathtt{x}^{2k-1} \vdash^\star \mathtt{X}\mathtt{x}^{2k-1} \sqcup\, q_{\text{accept}}$. Then use this, along with results that have been established before this, to show that M ***accepts*** every string $\omega \in \Sigma^\star$ such that $\omega \in L$.

Next suppose that $\omega \in \Sigma^\star$ such that $\omega \notin L$ because ω includes strictly more copies of $\mathtt{a}$ than copies of $\mathtt{b}$. Since the case that ω *only* incudes copies of $\mathtt{a}$ has been considered above suppose, in particular, that there exists a positive integer k such that ω includes exactly k copies of $\mathtt{b}$, and $k+1$ *or more* copies of $\mathtt{a}$. Then there exists a positive integer h and a string $\nu \in \{\mathtt{a},\mathtt{x}\}^\star$ such that

$$\mathtt{X}\mu_k = \mathtt{X}\mathtt{x}^h\mathtt{a}\nu.$$

Exercise: Show that if h and μ are as above then

$$\mathtt{X}\,q_2\,\mathtt{x}^h\mathtt{a}\nu \vdash^\star \mathtt{X}\mathtt{x}^{h+1}\nu \sqcup\, q_{\text{reject}}.$$

Then use this, along result, along with results that have been established before this, to show that M ***rejects*** every string $\omega \in \Sigma^\star$ such that $\omega \notin L$ because ω includes strictly more copies of $\mathtt{a}$ than it does copies of $\mathtt{b}$.

Exercise: Modify the final exercise, as needed, and use it show that M also ***rejects*** every string $\omega \in \Sigma^\star$ such that $\omega \notin L$ because ω includes strictly more copies of $\mathtt{b}$ than it does copies of $\mathtt{a}$.

If you have completed these exercises then — apart from organizing material in a written proof — you will have done all the work needed to prove that M decides L.

Finally, it might be helpful to compare a proof of the correctness of the high-level algorithm, found in the preparatory material for the lecture that discusses Turing machine design, to the proof of correctness of the Turing machine M that has now been discussed too. You might notice that the organizations of the two proofs are the same — and that the second proof is, arguably, what you would get by adding detail to the first.