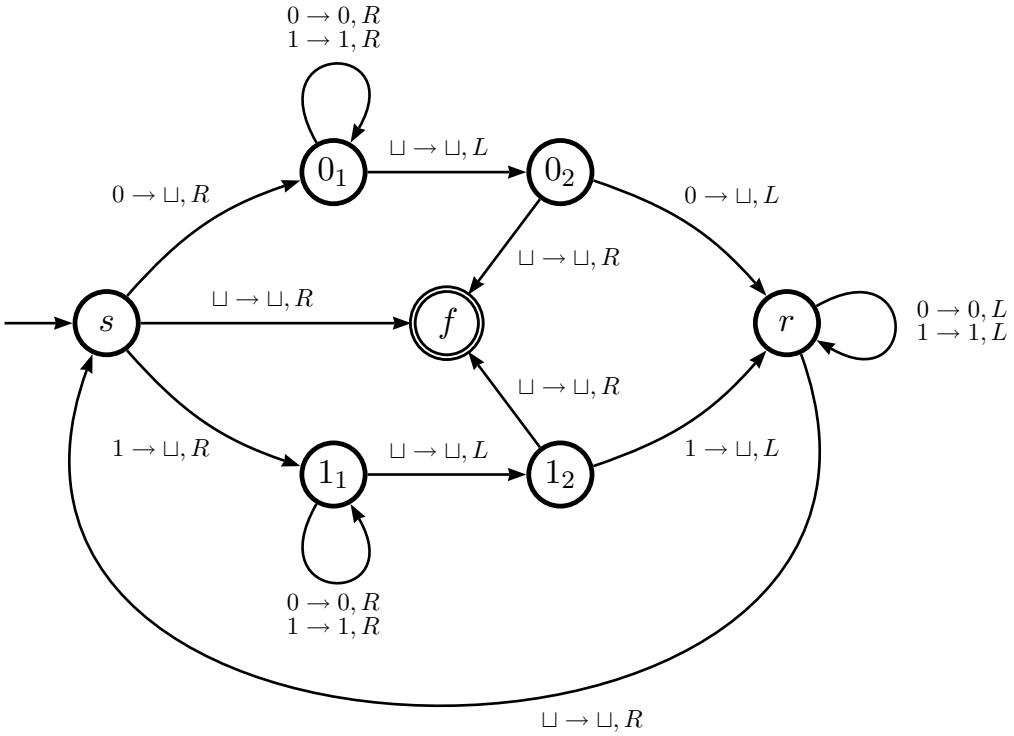
1. **(Designing a single-tape deterministic TM)**

   The basic idea is to shuttle back and forth between the two ends of the input string, trimming it down by one symbol on each end every time we find a match. In English, our TM does the following.

   1 Read the symbol a under the tape head.
   2 If $a = \sqcup$, the whole string has been matched, so ACCEPT.
   3 Otherwise, write $\sqcup$ and move right.
   4 Repeatedly move right until the tape head sees a $\sqcup$.
   5 Move left.
   6 Read the symbol b under the tape head.
   7 If $b = \sqcup$, the whole string has been matched, so ACCEPT.
   8 Otherwise, if $b \neq a$, we have a mismatch, so REJECT.
   9 Otherwise, we have a match, so trim: write $\sqcup$ and move left.
   10 Repeatedly move left until the tape head sees a $\sqcup$.
   11 Move right and repeat from start.

   Note that even palindromes (including ε) will be accepted by the first ACCEPT (Line 2), while odd ones will be accepted by the second ACCEPT (Line 7). Our formal state diagram is structured as follows (REJECT transitions, for simplicity, are not drawn): the start state s implements Lines 1-3, 0_1 and 1_1 implement Lines 4-5, for the two cases where the symbol last consumed was a 0 or a 1, respectively, 0_2 and 1_2 implement Lines 6-9, and r implements the loop-back, Lines 10-11.



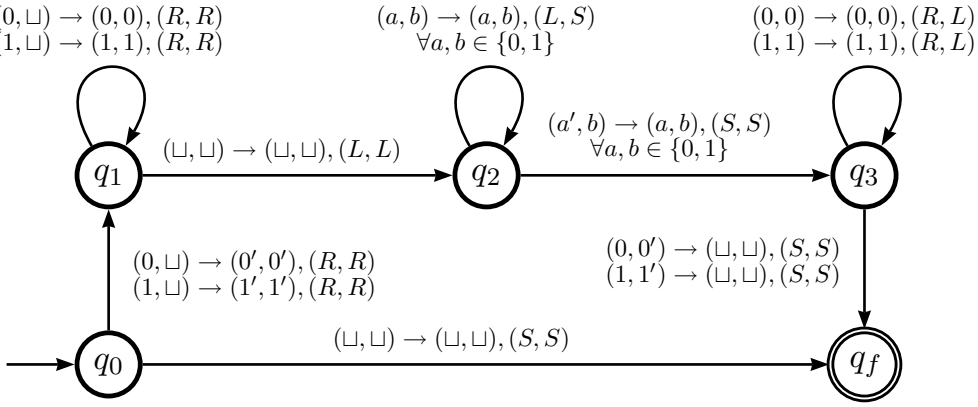
2. **(Designing a two-tape deterministic TM)**

   The basic idea here is to copy the input string onto the second tape, then have the two tape heads

run in opposite directions, matching a symbol at a time as they go. In English, the TM does the following.

1 If the input string is ε, ACCEPT.
2 Mark the first symbol on Tape 1.
3 Copy it to Tape 2 and move right on both tapes.
4 Repeatedly copy symbols from Tape 1 to Tape 2, moving right, until a $\sqcup$ is seen on Tape 1.
5 Move left once on both tapes.
6 Repeatedly move left on Tape 1, until the marked symbol is seen, staying put on Tape 2.
7 If the current symbols are equal (upto marking), move right on Tape 1 and left on Tape 2.
8 Otherwise, REJECT.
9 Repeat 7-8 until the marked symbol is seen on Tape 2.
10 If these last two symbols are equal (upto marking), ACCEPT, otherwise, REJECT.

The formal state diagram is as follows. The start state q_0 implements Lines 1-3, q_1 implements Lines 4-5, q_2 implements Line 6, and q_3 implements Line 7-10.

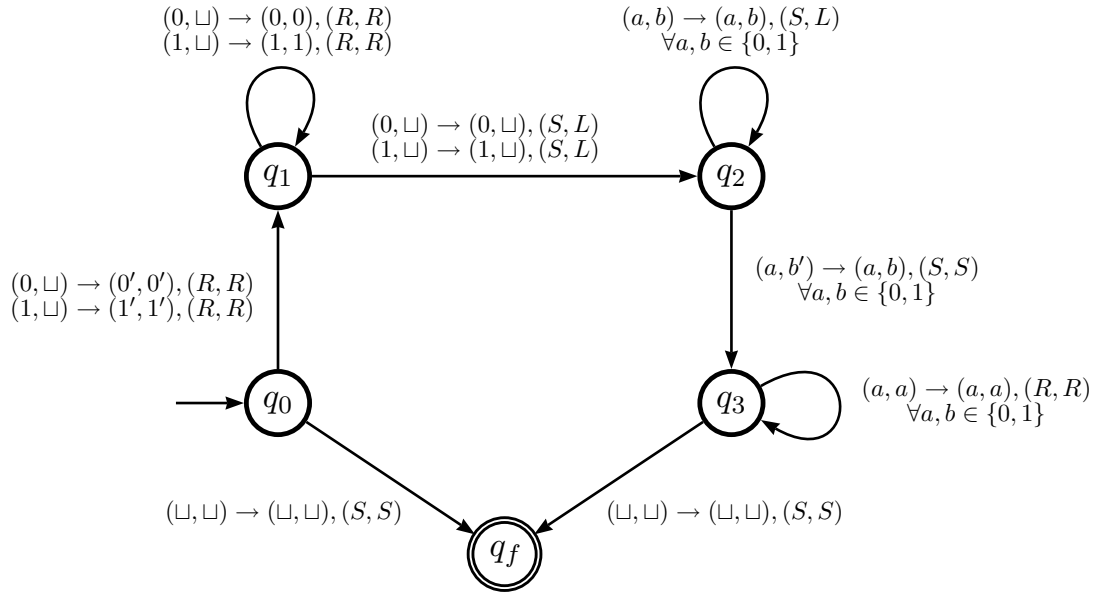


3. **(Designing a two-tape nondeterministic TM)**

Intuitively, we of course want to use nondeterminism to guess the halfway point, copying this guessed "half" to the second tape. We can then attempt to match the two "halves" symbol-by-symbol. In English, our TM works as follows.

1 If the input string is ε, ACCEPT.
2 Otherwise, mark the first symbol on Tape 1, copy it to Tape 2, and move right on both.
3 Repeatedly copy symbols from Tape 1 to Tape 2, moving right on both.
4 Guess when to stop (possibly after 0 repetition).
5 Keep the Tape 1 head on the first uncopied symbol, and move the Tape 2 head left until the marked symbol.
6 If the current symbols are equal (upto marking), move right on both tapes; otherwise, REJECT.
7 Repeat 6 until a $\sqcup$ is read on both tapes, and ACCEPT.

The state diagram is as follows. Our start state q_0 implements Lines 1-2, q_1 implements Lines 3-4 (the transition from q_1 to q_2 is the guess), q_2 implements Line 5, and q_3 implements Lines 6-7.



4. **(Closure properties of decidable languages)**

4.1. The class of decidable languages is closed under union.

Proof. If L_1 and L_2 are decidable languages with deciders M_1 and M_2, respectively, then the following is a two-tape TM for $L_1 \cup L_2$:

"On input w...

1 Copy input string w to Tape 2, so that both tapes now contain w.
2 Simulate M_1 on Tape 1, leaving Tape 2 alone.
3 Simulate M_2 on Tape 2, leaving Tape 1 alone.
4 If either accepts, then ACCEPT; otherwise, REJECT."

Since M_1 and M_2 are deciders, this must also be a decider. Moreover, we know that any two-tape TM has an equivalent single-tape TM. □

4.2. The class of decidable languages is closed under intersection.

Proof. If L_1 and L_2 are decidable languages with deciders M_1 and M_2, respectively, then the following is a two-tape TM for $L_1 \cap L_2$:

"On input w...

1 Copy input string w to Tape 2, so that both tapes now contain w.
2 Simulate M_1 on Tape 1, leaving Tape 2 alone.
3 Simulate M_2 on Tape 2, leaving Tape 1 alone.
4 If both accept, then ACCEPT; otherwise, REJECT."

Since M_1 and M_2 are deciders, this must also be a decider. As before, it has an equivalent single-tape TM. □

4.3. The class of decidable languages is closed under complementation.

Proof. If L is a decidable language with deciders M, then the following is a TM for $\overline{L}$:

"On input w...
1 Simulate M on w.
2 If M rejects, then ACCEPT; otherwise, REJECT."

Since M is a decider, this must also be a decider. □

4.4. The class of decidable languages is closed under concatenation.

Proof. If L_1 and L_2 are decidable languages with deciders M_1 and M_2, respectively, then the following is a three-tape TM for $L_1 \cdot L_2 = \{uv : u \in L_1, v \in L_2\}$:

"On input w...
1 Guess a prefix of w (not necessarily proper), and copy it to Tape 2.
2 Copy the rest of w to Tape 3.
3 Simulate M_1 on Tape 2, leaving the other two tapes alone.
4 Simulate M_2 on Tape 3, leaving the other two tapes alone.
5 If both accept, then ACCEPT; otherwise, REJECT."

Since M_1 and M_2 are deciders, this must also be a decider. It is nondeterministic, but we know that any NDTM has an equivalent DTM. Likewise, its three tapes can be reduced to one. □

4.5. The class of decidable languages is closed under Kleene star.

Proof. If L is a decidable language with deciders M, then the following is a two-tape TM for L^*:

"On input w...
1 If $w = \varepsilon$, ACCEPT.
2 Otherwise, guess a nonempty prefix of w, and copy it to Tape 2.
3 Simulate M on Tape 2.
4 Clean up Tape 2.
5 While the remaining suffix of w is not ε, repeat 2-4 on it.
6 If all simulations of M above accepts, ACCEPT; otherwise, REJECT.

Note that in order to implement the cleanup step (Line 4), M needs to be modified slightly so that it keeps track of the rightmost position on the tape it ever reaches during computation. We then only have to write $\sqcup$'s up to that point to ensure that the tape is completely blank.

Another point of note is that in guessing each prefix (Line 2), it is crucial that only nonempty prefixes are allowed. This ensures that every incarnation terminates after at most $|w| < \infty$ iterations. Otherwise, an incarnation would be allowed to keep guessing ε-prefixes forever, and this would not then be a decider. To check that this restriction still allows our TM to accept *all* strings in L^*, observe that any string in L^* can in fact be split into nonempty pieces, each of which is in L.

Finally, as before, nondeterminism and the multiple tapes can be done away with to give us an equivalent sigle-tape DTM. □

5. **(Closure properties of recognizable languages)**

5.1. The class of recognizable languages is closed under union.

Proof. If L_1 and L_2 are recognizable languages with recognizers M_1 and M_2, respectively, then the following is a two-tape TM for $L_1 \cup L_2$:

"On input w...

1 Copy input string w to Tape 2, so that both tapes now contain w.
2 Concurrently simulate M_1 on Tape 1 and M_2 on Tape 2.
3 If either accepts, then ACCEPT."

Note that if we simply simulated M_1 on w, then M_2 on w, there is the possibility that $w \in L_2 \setminus L_1$ and M_1 never halts on w, in which case we would never get around to simulating M_2 and would instead loop forever in M_1. Thus, we need to simulate the two machines concurrently, that is, repeatedly running one step of each in turn, during which the other tape is left untouched. Clearly, if either M_1 or M_2 accepts after a finite number of steps, our TM then would as well, in about twice that number of steps. □

5.2. The class of recognizable languages is closed under intersection.

Proof. If L_1 and L_2 are recognizable languages with recognizers M_1 and M_2, respectively, then the following is a two-tape TM for $L_1 \cap L_2$:

"On input w...

1 Copy input string w to Tape 2, so that both tapes now contain w.
2 Simulate M_1 on Tape 1, leaving Tape 2 alone.
3 Simulate M_2 on Tape 2, leaving Tape 1 alone.
4 If both accept, then ACCEPT; otherwise, REJECT."

Here we do not need concurrency because if M_1 never halts, then w should not be accepted anyway, so it is OK for our TM to run forever as well. □

5.3. The class of recognizable languages is closed under concatenation.

Proof. If L_1 and L_2 are recognizable languages with recognizers M_1 and M_2, respectively, then the following is a TM for $L_1 \cdot L_2$:

"On input w...

1 Guess a prefix of w (not necessarily proper), and copy it to Tape 2.
2 Copy the rest of w to Tape 3.
3 Simulate M_1 on Tape 2, leaving the other two tapes alone.
4 Simulate M_2 on Tape 3, leaving the other two tapes alone.
5 If both accept, then ACCEPT; otherwise, REJECT."

As in the previous case, we do not need concurrency, since if $w \in L_1 \cdot L_2$, *some* incarnation will halt and accept after a finite number of steps. □

6. **(Efficient transformation of k-tape to single-tape)**

Claim. *A k-tape TM M can be simulated by a single-tape TM N so that a t-step computation takes at most $O(t^2)$ steps.*

Proof. A key idea that makes your life much easier is to interleave symbols of the k tapes on your single tape. That is, the i^{th} symbol on the j^{th} tape maps to position $(i-1) \cdot k + j$ on the single tape. Assuming, for simplicity, that $m = O(t)$, after t steps on M, each of its tapes can contain at most $O(t)$ non-$\sqcup$ symbols, giving a total of at most $m + kt = O(t)$ non-$\sqcup$ symbols on N's tape.

N then works as follows. First, it moves symbols of the input string to correspond to M's first tape. It then keeps k markers to mark the heads of the k tapes of M, which initially are at positions 1 through k. Now, to simulate a single move by M, N needs to look at the symbol at each of the k heads, and potentially write new values at each one and move them either left or right. It thus requires one sweep through its tape (of length $O(t)$) to read what each of the k heads would have seen, then another sweep to write the new values and move the heads (each head-move takes $k = O(1)$ steps so this is still effectively just one sweep). Therefore, N takes $O(t)$ time to simulate each of M's t moves, resulting in a total of $O(t^2)$ steps. □