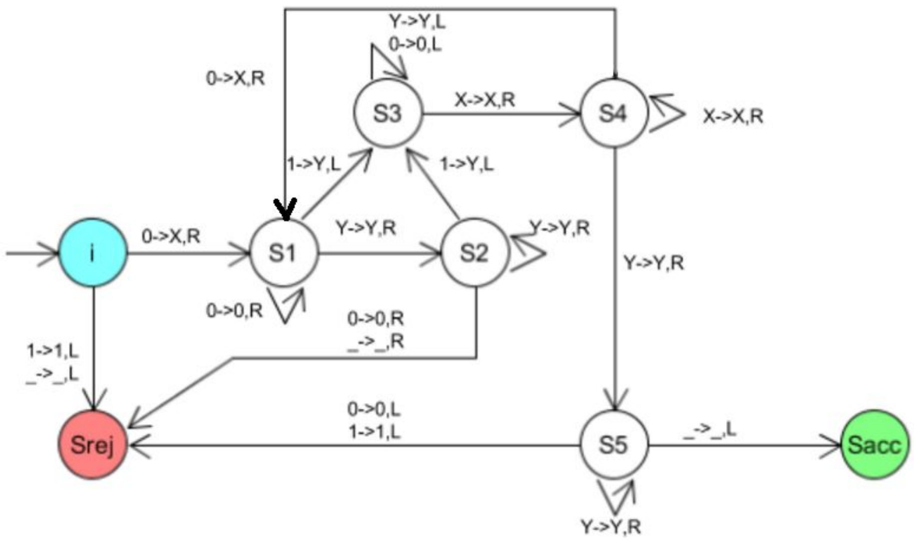
Before we can write down the set of states S or the transition mapping t, let us draw a **transition diagram** which is the Turing machine equivalent of drawing a finite state acceptor.



- $i \rightarrow S_{rej}$ represents step 1 of the algorithm.
- $i \rightarrow S_1$ and $S_4 \rightarrow S_1$ represent step 2 of the algorithm.
  ($i \rightarrow S_1$ at the first pass through the string, $S_4 \rightarrow S_1$ at subsequent passes)
- $S_1 \rightarrow S_1$, $S_1 \rightarrow S_2$ and $S_2 \rightarrow S_2$ represent the first part of step 3.
- $S_2 \rightarrow S_{rej}$ represents the second part of step 3.
- $S_1 \rightarrow S_3$ represents step 4.
- $S_3 \rightarrow S_3$ and $S_3 \rightarrow S_4$ represent the first sentence in step 5.
- $S_4 \rightarrow S_4$ and $S_4 \rightarrow S_5$, $S_5 \rightarrow S_5$ represent the second sentence in step 5.
- $S_5 \rightarrow S_{rej}$ represents the first half of the third sentence in step 5.
- $S_5 \rightarrow S_{acc}$ represents the second half of the third sentence in step 5.
- $S_4 \rightarrow S_1$ represents the step 6.

We have accounted for all pieces of our algorithm, therefore we have written down a Turing machine when A = {0, 1}, Ã = {0, 1, X, Y, _ }

S = {i, S_{acc}, S_{rej}, S_1, S_2, S_3, S_4, S_5}

- i = Initial State
- $S_{acc} \in S$ = Accept State
- $S_{rej} \in S$ = Reject State

We just have to write the **transition mapping** $t : S \times \tilde{A} \rightarrow S \times \tilde{A} \times \{L, R\}$

1. $t(i, 0) = (S_1, X, R)$ If in initial state and read 0, write X & move right to state 1
2. $t(i, 1) = (S_{rej}, 1, L)$
3. $t(i, \_) = (S_{rej}, \_, L)$

These are the only 3 transitions possible out of state i, but $t : S \times \tilde{A} \rightarrow S \times \tilde{A} \times \{L, R\}$ so technically, to write down the full transition mapping we must assign triplets in $S \times \tilde{A} \times \{L, R\}$ even to inputs from $\tilde{A}$ that cannot occur in i.

4. $t(i, X) = (S_{rej}, X, L)$
5. $t(i, Y) = (S_{rej}, Y, L)$

We assign S_{rej} some element of $\tilde{A}$ and one of the allowable tapehead distinctions.

Technically, the Turing machine halts when it enters either an accepting or rejecting state, so in practice we can define

$$\check{S} = \{i, S_1, S_2, S_3, S_4, S_5\} = S \setminus \{S_{acc}, S_{rej}\} \text{ (Set of resulting states)}$$

and $t : \check{S} \times \tilde{A} \rightarrow S \times \tilde{A} \times \{L, R\}$, so we avoid writing down the transitions from S_{acc} and S_{rej}.
We only have states S_1, S_2, S_3, S_4 and S_5 left.

6. $t(S_1, 0) = (S_1, 0, R)$
7. $t(S_1, Y) = (S_2, Y, R)$ On the diagram
8. $t(S_1, Y) = (S_3, Y, L)$

9. $t(S_1, X) = (S_{rej}, X, R)$ Not on the diagram - cannot occur, so added for completeness
10. $t(S_1, \_) = (S_{rej}, \_, R)$

11. $t(S_2, Y) = (S_2, Y, R)$
12. $t(S_2, 1) = (S_3, Y, L)$ On the diagram - can occur
13. $t(S_2, 0) = (S_{rej}, 0, R)$
14. $t(S_2, \_) = (S_{rej}, \_, R)$

15. $t(S_2, X) = (S_{rej}, X, R)$ Not on the diagram - cannot occur, so added for completeness

16. $t(S_3, Y) = (S_3, Y, L)$
17. $t(S_3, 0) = (S_3, 0, L)$ On the diagram - can occur
18. $t(S_3, X) = (S_4, X, R)$

19. $t(S_3, \_) = (S_{rej}, \_, R)$ Not on the diagram - cannot occur, so added for completeness
20. $t(S_3, 1) = (S_{rej}, 1, R)$

21. $t(S_4, X) = (S_4, X, R)$
22. $t(S_4, Y) = (S_5, Y, R)$ On the diagram - can occur
23. $t(S_4, 0) = (S_1, X, R)$

24. $t(S_4, 1) = (S_{rej}, 1, R)$ Not on the diagram - cannot occur, so added for completeness
25. $t(S_4, \_) = (S_{rej}, \_, R)$

26. $t(S_5, Y) = (S_5, Y, R)$
27. $t(S_5, \_) = (S_{acc}, \_, L)$ On the diagram - can occur
28. $t(S_5, 0) = (S_{rej}, 0, L)$
29. $t(S_5, 1) = (S_{rej}, 1, L)$

30. $t(S_5, X) = (S_{rej}, X, L)$ Not on the diagram - cannot occur, so added for completeness

Moral of the Story:
The transition mapping is a very inefficient way of specifying a Turing machine as a lot of transitions cannot occur unlike what we saw for a finite state acceptor, where the input alphabet was exactly the alphabet of the language.
Here $A \subset \tilde{A}$. Therefore we will specify a Turing machine via either an algorithm or the transition diagram **only**.