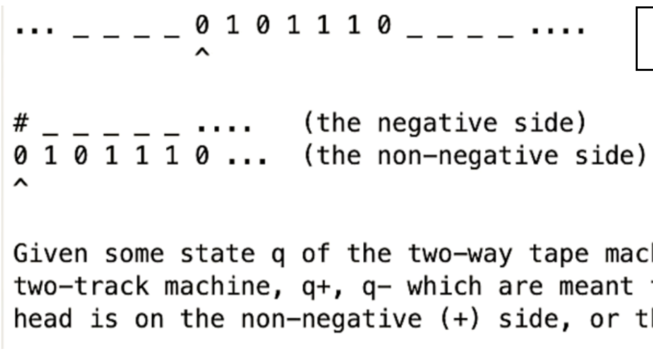
Lecture 13 interactive notes from Michael Albert, 2024 (annotated)

Simulating a two-way tape machine using a two-track machine

The lines below represent a two-way tape TM followed by a two-track machine that simulates the first TM.



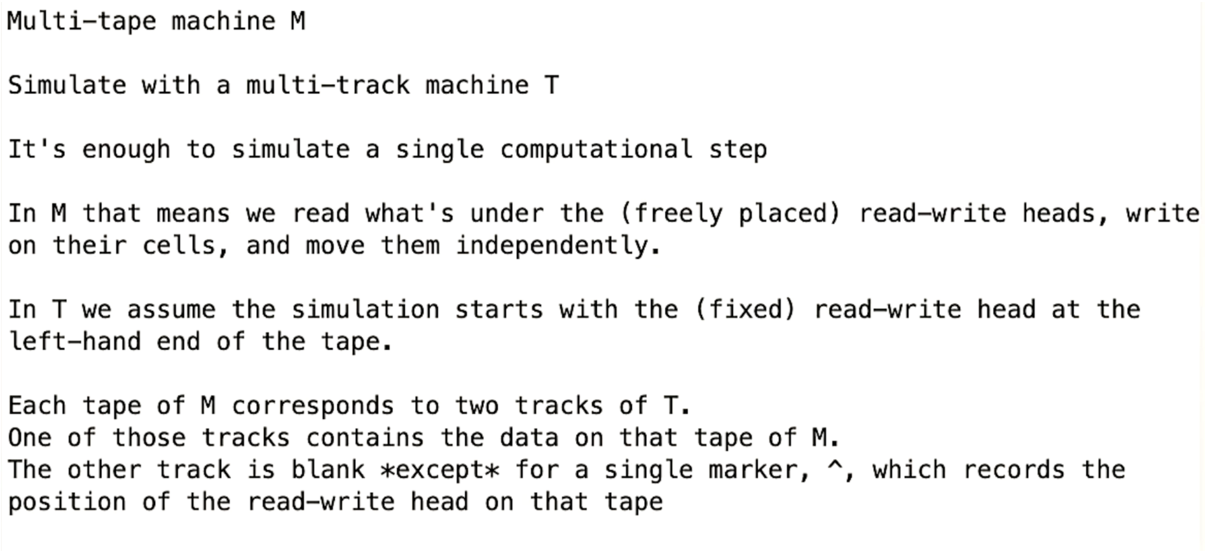
```
... _ _ _ _ 0 1 0 1 1 1 0 _ _ _ _ ....
            ^

# _ _ _ _ _ ....   (the negative side)
0 1 0 1 1 1 0 ...  (the non-negative side)
^

Given some state q of the two-way tape machine, we define two states for the
two-track machine, q+, q- which are meant to be interpreted as 'the read-write
head is on the non-negative (+) side, or the negative side (-).
```

A two-way TM has a tape that is infinite in both directions

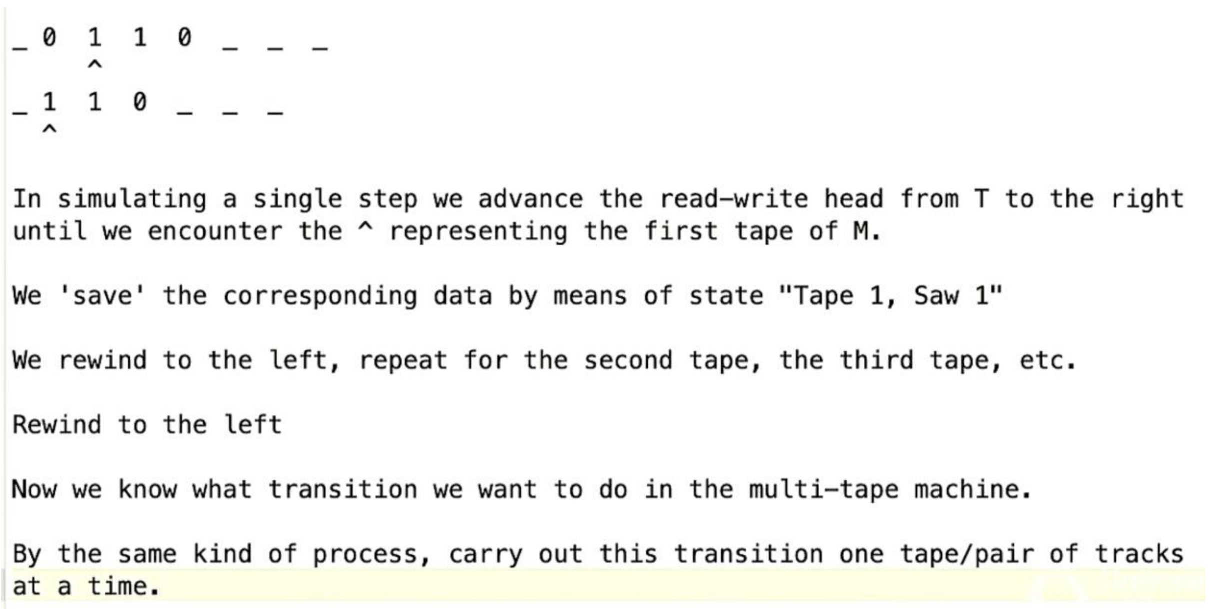
The top track represents the negative (left) tape. The bottom represents the non-negative (right) side. The middle position appears on one tape, with # on the other

Refer to Notes 11 for details about the transitions used on the two-track machine to simulate the transitions on the two-way tape machine.

Simulating a multi-tape machine using a multi-track machine

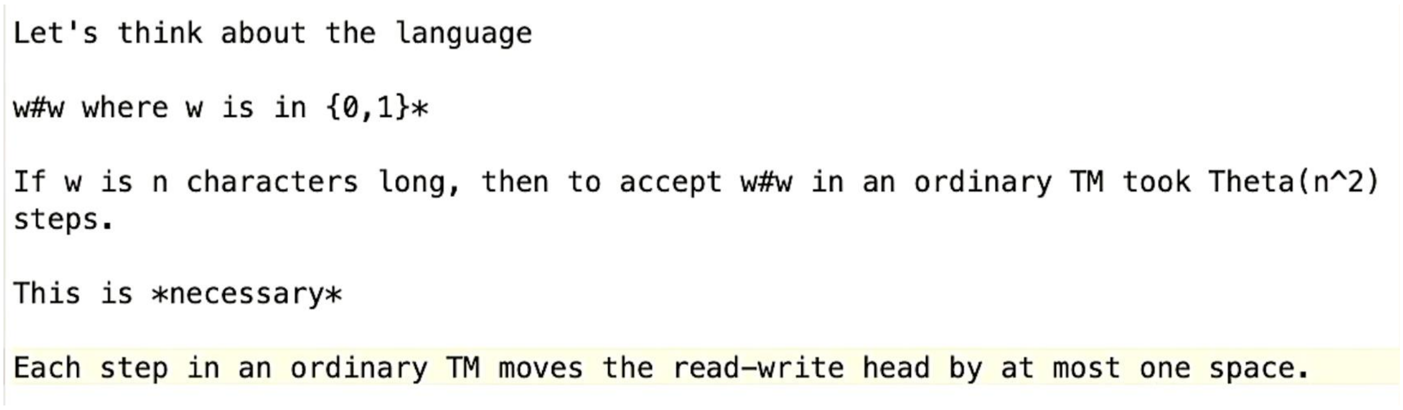


```
Multi-tape machine M

Simulate with a multi-track machine T

It's enough to simulate a single computational step

In M that means we read what's under the (freely placed) read-write heads, write
on their cells, and move them independently.

In T we assume the simulation starts with the (fixed) read-write head at the
left-hand end of the tape.

Each tape of M corresponds to two tracks of T.
One of those tracks contains the data on that tape of M.
The other track is blank *except* for a single marker, ^, which records the
position of the read-write head on that tape
```



```
_ 0  1  1  0  _  _  _
     ^
_ 1  1  0  _  _  _
  ^

In simulating a single step we advance the read-write head from T to the right
until we encounter the ^ representing the first tape of M.

We 'save' the corresponding data by means of state "Tape 1, Saw 1"

We rewind to the left, repeat for the second tape, the third tape, etc.

Rewind to the left

Now we know what transition we want to do in the multi-tape machine.

By the same kind of process, carry out this transition one tape/pair of tracks
at a time.
```

Example of the performance gain from using an enhanced Turing machine



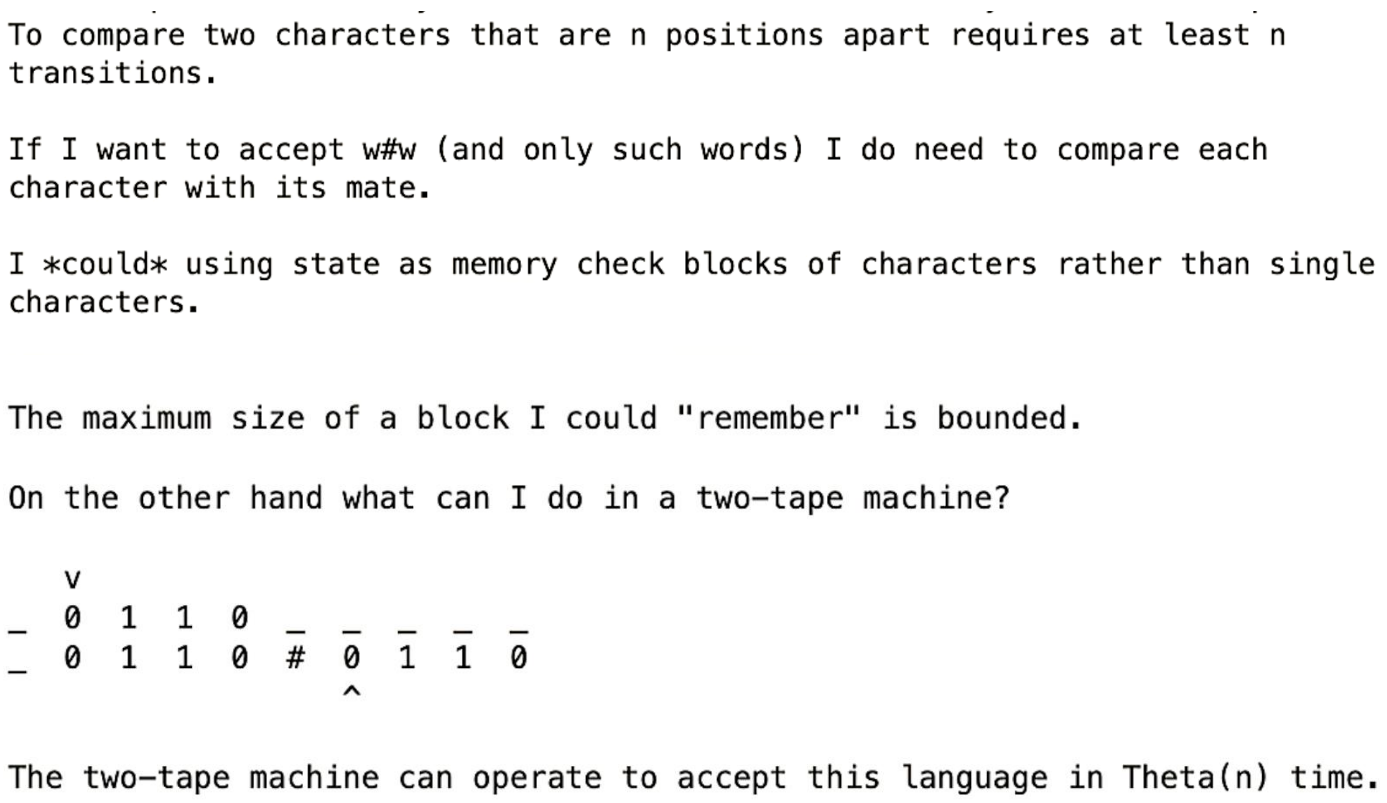
Let's think about the language

w#w where w is in {0,1}*

If w is n characters long, then to accept w#w in an ordinary TM took Theta(n^2) steps.

This is *necessary*

Each step in an ordinary TM moves the read-write head by at most one space.



To compare two characters that are n positions apart requires at least n transitions.

If I want to accept w#w (and only such words) I do need to compare each character with its mate.

I *could* using state as memory check blocks of characters rather than single characters.

The maximum size of a block I could "remember" is bounded.

On the other hand what can I do in a two-tape machine?

```
   v
_  0  1  1  0  _  _  _  _  _
_  0  1  1  0  #  0  1  1  0
                  ^
```

The two-tape machine can operate to accept this language in Theta(n) time.

Moral

More powerful machines don’t let you do things you couldn’t do before, but they can let you do them more quickly.