The idea of universal machines

We’re now ready for one of Turing’s really great ideas: the universal Turing machine.

Up to now, whenever we wanted to solve a problem computationally, we had to create a *different TM for each new problem*.

If you think of TM’s as hardware, this would be like buying a new computer for each new problem, or having to rewire a computer for each new problem that one wants to solve.

People actually had to do this in the early days of computers!
But then someone had a brilliant idea. Instead of rewiring the computer each time for a different problem, one could create a general-purpose computer that would take a program ("software") and input and run the program on that input.

The idea of a stored-program computer can be traced back to Babbage (1837) and Turing (1936). The first electronic stored-program computer was the SSEM, invented by Frederic C. Williams, Tom Kilburn, and Geoff Tootill in 1948 at the University of Manchester.
The universal TM

We can do something similar with a TM.

We can create a *single* one-tape DTM $U$, with input alphabet \{0, 1\} that can simulate every other Turing machine $T$.

What do we mean by “simulate”? It is a little involved because $T$’s input alphabet could be arbitrary (not binary).

We mean that $U$ takes as input both an encoding of $T$ and an encoded version of $T$’s input $x$. The encoding is the one we described before, where we can encode an arbitrary string over an arbitrary alphabet into a binary string.
Finally, $U$ has the property that $T$ accepts $x$ iff $U$ accepts $e(T)e(x)$.

So ultimately we only need one Turing machine, namely $U$. We can simulate every other Turing machine with it!

This idea of a universal TM is very bold!

There is no notion of “universal DFA”: one fixed DFA that can simulate every other DFA.

Similarly, there is no notion of “universal PDA”: one fixed PDA that can simulate every other PDA.
Constructing the universal TM

Here's the idea:

We will construct a 3-tape deterministic TM $U'$. (We know that this, in turn, can be simulated by a 1-tape deterministic TM $U$.)

Tape 1 holds $e(T)e(x)$.

Tape 2 holds an encoded version of $T$'s tape. That is, if $a_1 \cdots a_n$ is the contents of $T$'s tape, then $U'$'s Tape 2 holds $e(a_1) \cdots e(a_n)$. Furthermore if $T$ is scanning $a_i$ then $U'$ is scanning (on Tape 2) the first symbol of $e(a_i)$.

Tape 3 holds the current state of $T$, encoded.
We now do the following, forever, until $T$ reaches $q_{acc}$ or $q_{rej}$.

Initialization:

Copy $e(x)$ to Tape 2; rewind.

Initialize Tape 3 with $e(q_0)$.

Loop: If the current state on Tape 3 is $e(q_{acc})$, then accept and halt. If the current state on Tape 3 is $e(q_{rej})$, then reject and halt.
Otherwise, scan $e(T)$ on Tape 1, looking for a move of $e(\delta)$ that matches the encoding of the symbol on Tape 2 and the state on Tape 3.

When you find it, replace the encoding of the symbol on Tape 2 with the new encoded symbol on Tape 1. This might require shifting of symbols either left or right because the new encoded symbol might take more or fewer symbols than the old one.

Also update the state on Tape 3.

Finally, move left or right on Tape 2 as the move specifies.

Return to the top of the Loop.
This completes our construction of the universal TM $U$.

With $U'$ (and hence $U$) we can simulate any TM $T$ on any input $x$, by feeding $U$ with $e(T)e(x)$.

If $T$ halts and accepts $x$, then $U$ halts and accepts $e(T)e(x)$.

If $T$ halts and rejects $x$, then $U$ halts and rejects $e(T)e(x)$.

If $T$ runs forever on input $x$, then $U$ does the same thing on input $e(T)e(x)$.
Small universal Turing machines

The way we constructed a universal TM is not optimal in terms of criteria like number of states, size of tape alphabet, and efficiency.

There has been some interest in trying to optimize these things.

It is known that there is a universal TM with 15 states and 2 tape symbols, and another one with 2 states and 18 tape symbols.

Furthermore, there is a universal TM with 4 states and 6 tape symbols that only needs 22 different transitions.

This shows that we do not need an immensely complicated construction to achieve universality.