The Church-Turing Thesis and the First Incompleteness Theorem

Alex Creiner

October 7, 2021

1 The Church-Turing Theses

Last time we defined computability in terms of the abstract model of a Turing machine. I think we mostly agreed that this model, being derived from and deeply related to human pen and paper computation, was a natural one.

Claim 1.1. *Any mechanical human pen and paper computation can be represented within the Turing model.*

This claim is not mathematical, and can't be, because what we are doing is fundamentally conjoined with physical reality, that thing we hate. *We are observing a pattern of phenomena in physical reality and attempting to abstract from it. There can never be a fully abstract confirmation that we did such a thing correctly.*

Nonetheless, if we can manage to temporarily overlook our obsession with mathematical rigor, we can probably agree without much argument that the claim is true. A bit of history.

Serious work toward formalizing the notion of computability began in the early 1930s after Hilbert and Ackermann posed their famous Entscheidungsproblem (which translates roughly as "decision problem"), which conjectured that the mathematician's own job of proving the truth or falsehood of a claim could be accomplished via an "effective procedure", and this of course required a formalization of what that meant exactly. Church and his student Kleene defined a set of λ-definable functions, and proposed that a function should taken as computable by an effective procedure if it were λ-definable. Gódel was not at all satisfied with this proposal, believing that the notion should be defined axiomatically. He went on to define his own class of recursive functions, but was hesitant to claim that this class encompassed *all* of the functions which should be considered computable. Soon after, these two classes of functions were shown to be the equivalent, at the same time arriving at the conclusion that the Entscheidungsproblem was false. At almost the same time, Turing released his famous 1936 paper in which he defined what are now known as Turing machines, and sketching a proof that this notion of computability was equivalent to both Church's and Gódel's. By this point, they were all becoming increasingly convinced that any further proposed definitions of computability would be equivalent to those already existing, and subsequently that these were *all* valid ways to define the notion. The conclusion they reached is now known as the **Chuch-Turing Thesis**, which can be stated as follows:

Theorem 1.1 (The Church-Turing Thesis)**.** *Any sufficiently powerful model of computation is equivalent to the Turing machine model. Thus Turing machines are as powerful as any model of computation can be. Because of this, if one can describe an algorithm for solving a problem in such a way as to be reasonably rigorous under some model of computation (i.e. by an unambiguous procedure), then there exists a Turing machine which accomplishes the same task.*

If a model of computation is equivalent to the Turing machine model in the class of problems which it solves, then we say that the model is **Turing complete**. We will discuss the many consequences of this conclusion in the next section.

That's not really all there is to it though. As we discussed in the previous talk, the idea of computation transcends pen and paper. Consider for example a calculator. You punch in an expression, hit a button, and get a result. Obviously computation occurs here, but it isn't human pen and paper computation, nor is it the abstract operation of a Turing machine. Some physical process occurs and completes, involving

directed flows of electricity, switches flipping on and off, a sequence of causes and effects. This is an example of physical computation, and just as the Turing model is a simplified abstraction of human computation, human computation itself, us sitting down and temporarily becoming a simpler machine for the sake of carrying out a process of deduction, is creating an *isomorphism* between our own pen and paper actions and the electrical operation of the calculator.

When you follow instructions to bake a cake, you are carrying out computation. When a cat falls from a tree and flips itself upright in order to fall on it's legs, biological computation is occurring rapidly. Computation goes on all around us all of the time, and can all be related to all other forms of computation through this cybernetic concept of isomorphism, and through Turing's generalization of mind, pen and paper to the control, the executive, and the store.

Fig. 6/8/1 shows two dynamic systems, each with an input and an output. In the upper one, the left-hand axle *I* is the input; it can be rotated to any position, shown on the dial *u*. It is connected through a spring *S* to a heavy wheel *M*, which is rigidly connected to the output shaft *O*. *O*'s degree of rotation is shown on the dial *v*, which is its output. The wheel dips into a trough with liquid *F* which applies a frictional force to the wheel, proportional to the wheel's velocity. If now, starting from given conditions, the input *u* is taken through some sequence of values, so will the output *v* pass through some determinate sequence of values, the particular sequence depending on *v's* initial value, on *v*'s rate of change at that moment, and on the sequence used for the input at *u*.

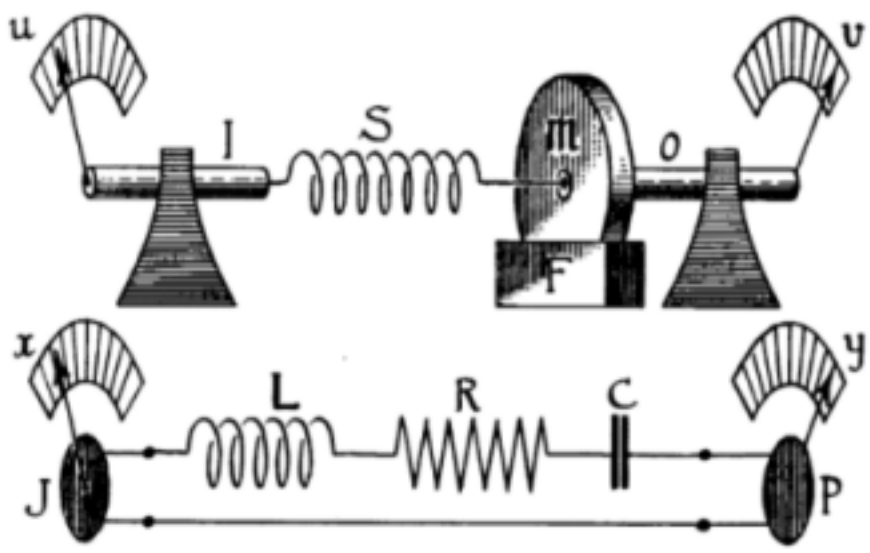


Fig. 6/8/1

The lower system is electrical. Its input is a potentiometer, or other device, *J*, that emits the voltage shown on the scale *x*. In series are an inductance *L*, a resistance *R*, and a capacitance *C*. *P* is a current meter (such as is used in domestic supplies) recording the sum of the currents that have passed through it. The sum is shown on the scale *y*, which is its output.

If now the values of *L, R* and *C* are adjusted to match the stiffness of the spring, inertia of the wheel, and friction at *F* (though not respectively), then the two systems can show a remarkable functional identity. Let them both start from rest. Apply any input-sequence of values at *u*, however long and arbitrary, and get an output sequence at *v*, of equal length: if the same sequence of values is given at *x*, the output at y will be identical, along its whole length with that at *v*. Try another input sequence to *u* and record what appears at v*:* the same input given to *x* will result in

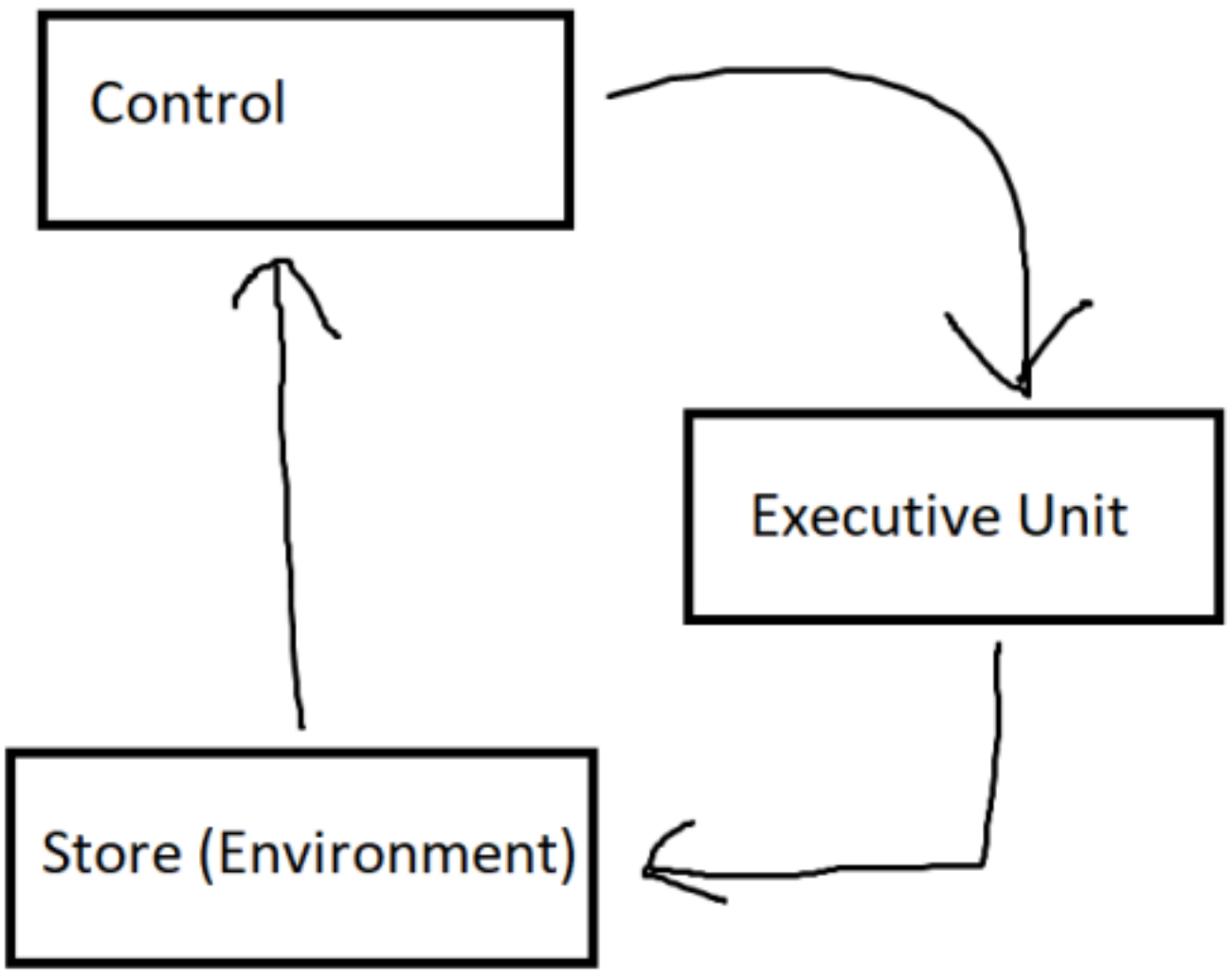
an output at y that copies that at v. Cover the central parts of the mechanism and the two machines are indistinguishable throughout an infinite number of tests applied. Machines can thus show the profoundest similarities in behaviour while being, from other points of view, utterly dissimilar.

Nor is this all. Well known to mathematicians are equations of the type

$$a\frac{d^2z}{dt^2} + b\frac{dz}{dt} + cz = w$$

by which, if a graph is given showing how w varied with time (t), the changes induced in z can be found. Thus w can be regarded as an "input" to the equation and z an "output". If now a, b, and c are given values suitably related to L, R, S, etc., the relation between w and z becomes identical with those between u and v, and between x and y. *All three systems are isomorphic.*

So computation is really a very general thing. And remember, despite Turing basing his machine off of human computation, he *began* by first generalizing the essential elements of human computation to something more general.



In abstracting human computation to the Turing model, we have abstracted the essential elements of human computation which apply to all computation generally. This control system feedback is the essential element.

(1) A controller, that is, a collection of sensory mechanisms for reading the environment along with a decision matrix, reads the environment