1 Undecidability; the Church-Turing Thesis

The Church-Turing thesis: A Turing machine that halts on all inputs is the precise, formal notion corresponding to the intuitive notion of an algorithm.

Note that algorithms existed before Turing machines were invented; for example, Euclid's algorithm to compute the greatest common divisor of two positive integers, and algorithms to multiply two integers.

The Church-Turing thesis cannot be proved because it relates a formal concept (Turing machines) to a vaguely defined informal one. An algorithm is defined as a sequence of instructions that can be unambiguously carried out by a human to obtain some kind of a result.

However, this thesis can be supported in various ways.

1. No one has yet found a natural example of an algorithm that could not be simulated by a Turing machine.
2. Also, the fact that all reasonable extensions to Turing machines do not increase their power, is a justification of the Church-Turing thesis.

Using the Church-Turing thesis, if one can show that a problem cannot be solved on a Turing machine, then it is reasonable to conclude that it cannot be solved by any computer or by any human.

This brings up the question of the relative strengths of Turing machines and humans.

- Of course people can do a lot of things that do not make sense for a Turing machine, such as enjoying a musical composition, or playing a game for fun.
- Computers are very good at some games, including chess.
- They have had a harder time with others, such as "go."

- They can do some mathematical proofs faster than humans, but are much worse in general at mathematical discovery.
- Computers currently lack insight and creativity in mathematics; could these be programmed in?
- Things that are easy for people but hard for computers: Speaking and understanding speech, vision, motion, and commonsense reasoning. These things are so easy for us that we do them to relax. We talk to others, and go on walks and look at the world, for enjoyment. We even sometimes deliberately say nonsensical things to make a joke, and understand what we are doing with no trouble.

  See CACM September 2015, Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence.

Computers and automation are taking over more and more jobs from humans. Where will this lead? What about 3-d printing, for example? These are social questions that need to be considered.

In principle, how much of what humans do, could a computer do? Is it possible that even if this were possible, it would not be economically feasible?

Consider also Turing's test for when a computer is able to simulate a human.

A related question is whether there is some other kind of a computer that might be more powerful than Turing machines. Other kinds of computers would be

- analog computers,
- computers utilizing randomness,
- quantum computers,
- DNA computers,
- optical computers,

and so forth. So far no other reasonable computing device has been found that is more powerful than a Turing machine in terms of what can be computed, given unlimited time and resources.

1.1 Universal Turing Machines

A *universal Turing machine* M takes as input a Turing machine T and an input x to T and simulates (interprets) T on input x. Thus, M halts on input $(T,\ x)$ iff T halts on input x, and sometimes it is also convenient to say that the output of M on input (T, x) is the same as the output of T on x.

To design a universal Turing machine it is necessary to

1. encode Turing machines as strings of symbols, and also to
2. encode inputs as strings of symbols.

This is not as simple as it may seem, because M has a fixed alphabet but T can have an arbitrary, very large alphabet, and x may also be written in a large alphabet. We use $encode(T)$ to represent an encoding of T in a fixed alphabet, and similarly $encode(x)$ for strings. The details of this encoding, and of M, will be given later.

- Universal Turing machines do exist, and are not difficult to describe.
- This is amazing in a way; it means that there is a single computer that can do anything any other computer can do.
- It's like having a tool that can be a screwdriver or a hammer or a saw or a knife or anything else.
- If this were not true for computers, we might need to use one computer for addition, another for subtraction, and so on – what a bother that would be!
- Modern stored program computers that can execute an arbitrary program, are essentially universal Turing machines.
- There has been some controversy about whether the concept of a universal Turing machine helped in the development of the modern stored program computer.

There has been an interest in seeing how simple a universal Turing machine can be.

- Hopcroft and Ullman designed one with 40 states and an alphabet of a few symbols.
- Minsky designed 6 state 6 symbol universal Turing machine and
- a 4 symbol 7 state universal Turing machine. Some of the power of these machines comes from the encoding used.
- There is controversy about a claimed 2 symbol 3 state universal Turing machine.

Universal Turing machines are also helpful for showing the existence of unsolvable problems; perhaps the encoding is more relevant for that than the universal Turing machine itself.