# Summarizing Finite State Automata

Here are some points of view on finite state automata that complement each other:

1. In practice, all digital computing machines have finite resources: finite memory and finitely many configurations their (digital) internal parts can be in. But they are, and in future, may be implemented in various ways to improve their efficiency. Yet, if you abstract out all that is common to all such machines, irrespective of their implementation, you get the definition of a deterministic finite state automaton. Therefore, the pumping lemma tells you, for example, that no matter how creative you try to be, you can never build a machine that uses bounded memory and yet can check that the parenthesis in a document are well matched no matter how large that document is. Yet, for certain other problems a bounded memory is enough no matter how large the input is. In practice, it means that for certain problems even a small RAM is enough, for others, you may need to use the hard drive and if the input is too large, even that may not be enough.

2. Although, in practice, all digital machines *can* be modelled by finite state automata to understand some of their limitations, for other reasons, it is useful to model them as auomata with indefinite memory (eg. push-down automata or Turing machines). These alternative models tell you that even though there is a limitation, for some problems the limitation exists only in the memory. In practice, that is useful because it is much easier to upgrade your computer's memory than its CPU. Furthermore, you may be sure that the several GB of memory we now have is more than enough for the inputs you will use in practice. In contrast, there are problems where even an indefinite memory is not enough to "solve" them.

3. It is useful to know if a problem can be solved by a finite state automaton (for all inputs, irrespective of size). Such a problem can make do with bounded memory and the time it takes to stop will be, even in the worst case, at most proportionate to the size of the input.

4. The definition of a finite state automaton is theoretically clean. That helps us to prove the pumping lemma eaily. But many of you have noticed that the theoretical simplicity of the definition of a deterministic finite state automaton comes at a price. It is too restrictive. This is where non-deterministic finite state automata step in.

5. The non-deterministic finite state automata are far more permissive, allowing us to easily build more complicated ones from simpler ones. It attains this permissiveness by sacrificing the theoretical simplicity of the deterministic ones. However, they can all be converted to deterministic ones, so the pumping lemma holds for them too. But now we can design simpler

The definition of a finite state automaton may seem very specific and artificial but it models machines with finite resources (finite memory, finite possible configurations the machine can be in, for example digitial machines). This does not mean that it is useful to model those machines as finite state automata for all purposes, but it can tell us what limitations such machines have in theory.

## The limitations of finite memories

Machines with finitely many possible configurations may be implemented in various way, but finite state automata capture what is common to all of them. If one were to completely ignore how the machines are designed, all one can say is that they have finite states and finite many possible chunks of inputs that can be given at a time. Each input changes the configuration, i.e. the state of the machine. Interacting with the machine may involve inputting many chunks over time and then checking the configuration (or rather a small part of it, to know the ouput).

The pumping lemma is a very simple application of pigeonhole principle that can show us limitations of such machines. Essentially, every finite resource machine is equivalent in power to some finite state automata (even though it may be use a massive number of states), so any limitation of all possible finite state automata is a limitation of finite resource machines.

But finite state automata are by no means the only model. Other common models (push-down automata, Turing machines) update the finite state automata with an inexhaustable supply of memory. Although, theoretically, we do not have infinite memory, these other models are still useful. Most importantly, it tells us that the only impediment to solving the problem when the input size is too large, is the memory and that is significant for it is much easier to upgrade the memory of your computer than to change the CPU (equivalent to the states and the transition function). Also, for many problems in practice one can assume that the memory is indefinite.

## Predictable perfomance.

But finite state automata are not just useful in highlighting limitations. Even in the class of problems where one assumes indefinite memory, they are a very important sub-class because they come with some important guarantees. The memory needed is fixed, and the time it takes to run is proportionate to length of the input. By contrast, in a Turing machine, sometimes one can not even know if it will even halt!

Now while the simple definition of finite state automata is useful to understand its theoretical properties, there are alternatives that make the definition less clean but are more convenient to use. For example, many of you have asked why it has not possibility to just stop and accept or reject without reading the rest of the input when it "knows" it need not do so. The answer is because those machines can easily be converted to the usual deterministic finite state automata. So in practice, you *will* use the more convenient ones, knowing that you can always convert them to the usual ones and therefore you know what their limitations are.

One very convenient model that adds a lot of freedom is the non-deterministic finite state automaton. The point is that non-deterministic finite state automata can be converted to deterministic ones (in a very deterministic way, meaning there is an algorithm to do so). So they can solve exactly the same problems as the limitations of the deterministic ones. But they are