# CS 4510 : Automata and Complexity
# State Control & Nondeterminism

Subrahmanyam Kalyanasundaram

February 11, 2010

I want to talk in more detail about two great questions that were asked in class today.

## The Role of State Control in Simulations

I was asked this question as we were understanding the simulation of multi-tape TM's using single tape TM's. For the sake of completeness, let me recall how the simulation goes. The single tape machine $S$, divides the tape into $k$ pieces using delimiters (the # symbols) and special symbols to denote the position of the head. To simulate one transition of the multi-tape machine $M$, $S$ needs to make two passes over the tape. In the first pass, it goes through the whole tape and determines the positions of the $k$ heads, and the contents of these locations, "remembering" them in the process. Then the head comes back, does another pass on the tape, and modifies the tape in order to simulate the changes in the multi-tape machine $M$.

The question was about the state control of $S$. The state control – a word denoting the states and the transition between states – of $S$ has to encode all that I mentioned above. It has to encode the states in such a way as to move the tape head from left to right once, look for the marked symbols (which are indicators of the head positions in the multi-tape machine), remember these symbols and come back. All this is achieved by having an appropriate set of states $Q$ and transition $\delta$ – recall the shifting example in class, where we saw how we could shift a string in the tape using appropriate state control. The state control should also simulate the state control of $M$ and then it should initiate the second pass over the tape making the necessary changes. The number of states of $S$ is definitely much more than the number of states of $M$ – the state control is much more complicated than in $M$. It has to encode lot more information, and transitions in order to make the machine $S$ work. The point is that, for any given multi-tape TM $M$, there exists a single tape TM $S$, having **a much more complicated, but finite set of states**. I believe the point that needs emphasis is that the set of states of $S$ is finite. This is a key step in showing the existence of the TM $S$ – the book does not stress this point well enough.

## The Concept of Nondeterminism

Onto the second question now. We were discussing the nondeterministic algorithm that tests if a given integer $w$ is composite or not. The algorithm guesses two integers, and checks if the product

is the given integer. If we get a nontrivial factorization, we are done. If we are given an $n$ bit integer as an input, we could limit the guesses to two $n$ bit integers, it is pointless to guess longer integers. The only step requiring any real computation is the product calculation and checking if the product is the same as the input $w$. This seems much shorter and easier than the deterministic algorithm of checking all possible factors – from 2 through $\sqrt{w}$. There are many possible guesses which would not work – many rejecting computation paths. But if there is a nontrivial factorization, then there is a set of guesses that yield this factorization of $w$, and would be accepted. And this is all that we need, just one accepting computation path. If the given number was not composite, then there would be no nontrivial factorization, and all possible guesses would be rejected.

Then we had the question, or should I say a complaint, someone asked how long would the nondeterministic computation *really* take. I think this is a very normal question. The question stems from the way we understand nondeterminism. The mistake that we tend to make is we try to equate nondeterministic machines with the real world computers. We think, how would a real computer do this, and it is hard to find parallels. This is because a nondeterministic machine is quite different – it is a computation model, and one should **not** think of these as real computers. Nondeterminism is an idea, a concept that is used to understand the power of computation (we will see in chapters 7 and 8 details on how we use nondeterministic classes) and as a reference to compare deterministic models. So one should think of nondeterministic machines as a separate concept and should not try to think how a real computer would perform nondeterminism. NTM's should be thought of as machines with several possible computation paths, and they accept if there is at least one accepting path, reject only if no path is accepting. Maybe it is best to think of them as special "out-of-the-world" machines, which can do computation on several paths simultaneously.