Math 260A — Mathematical Logic — Scribe Notes
UCSD — Winter Quarter 2012
Instructor: Sam Buss

Notes by: Angela Hicks
April 2nd and 4th

1 Notes on Turing's paper

We spent most of today's lecture reading Turing's paper, "On Computable Numbers, with an application to the Entscheidungsproblem" written in 1936, (where 'Entscheidungsproblem' translates to 'the decision problem.') We made a few isolated comments throughout the class:

- A universal Turing machine take as an input the description of a Turing machine M and gives as an output the output of M.
- The modern approach is to study Turing machines to simulate a modern style computer (which is not subject to finite memory) which in turn allows us to accept that we study all computable functions.
- Could you compute more with something besides a Turing Machine? Turing machines can simulate randomness and to a certain extent quantum computing for example.
- Are there instructions a Turing Machine cannot carry out? 'Draw a Picture. If it is attractive, go to state q_1, otherwise go to state q_2.' or 'Try a bunch of examples. From them, recognize a theorem and prove it.' are both instructions that a Turing machine would have trouble with, but they are not really algorithms.
- Changes to the Turing machine, instead of a single tape and head, include one way infinite tape or multiple tapes.

We also noted the Church-Turing Thesis: Any algorithmically computable process can be carried out by a Turing machine. (The converse is obvious.)

2 Turing machines

Definition 1. An *alphabet* is a finite set of *symbols*.

Definition 2 (Turing Machine)**.** A Turing machine is specified by:

$$(Q, q_0, \Sigma, \Gamma, \delta)$$

- Q is a finite set of states with distinguished states q_H (halt), q_Y (accept), q_N (reject), and q_P (pause).
- $q_0 \in Q$, the start state
- Σ the input alphabet (with b, the blank state, not in Σ.)
- Γ the working alphabet ($\Sigma \cup \{b\} \subset \Gamma$)
- δ is the transition function.

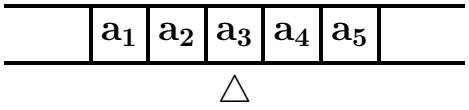
  $\delta : (Q \backslash \{q_H, q_Y, q_N, q_P\}) \times \Gamma \to \Gamma \times \{L, R, N\} \times Q$, where the first output is what is written to the square, the second output gives the direction of movement, and the third the state.

Definition 3 (Configuration)**.** The configuration includes the state and tape contents and the tape head position. It consists of:

1. State q
2. Right word contents of the tape, starting under the tape head.
3. Left word contents of the tape, starting left of the tape head.

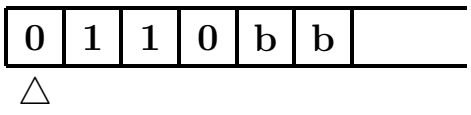
Example 1. For the below tape head, it might be represented as: q, a_3 a_4 a_5, a_2 a_1.

Using this convention, the "Next" function is then easily definable.

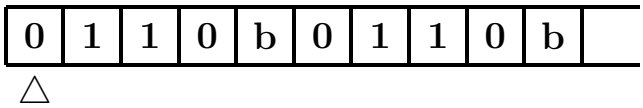


As an example, δ would be of the form $\delta(q, a_3) = \langle b, R \backslash L \backslash N, q' \rangle$.

Next, we'll give an example of a program that copies. For example, we'd like to start with the imput



and get the output:



To do that, we would use the following program:

q_0	0	$0'$	R	q_1
q_0	1	$1'$	R	q_3
q_1	0	0	R	q_1
q_1	1	1	R	q_1
q_1	b	b	R	q_2
q_3	0	0	R	q_1
q_3	1	1	R	q_1
q_3	b	b	R	q_4
q_2	0	0	R	q_2
q_2	1	1	R	q_2
q_2	b	0	L	q_5
q_4	0	0	R	q_4
q_4	1	1	R	q_4
q_4	b	1	L	q_5
q_5	0	0	L	q_5
q_5	1	1	L	q_5
q_5	b	b	L	q_5
q_5	$0'$	0	R	q_0
q_5	$1'$	1	R	q_0
q_0	b	b	N	q_H

Note that we use q_1 and q_2 as states to copy a 0, while we use q_3 and q_4 to copy a 1. The third through fifth lines, for example, represent working to copy a 0, scanning to find the first blank space. For this program $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 0', 1', b\}$.

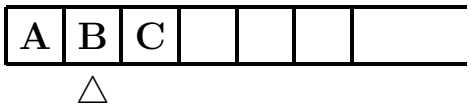
Claim 1. *WLOG,* $\Gamma = \Sigma \cup \{b\}$*. In other words, we can compute the same things if we restrict* Γ*.*

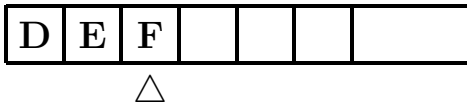
Note that if we even have just $0 \in \Sigma$, using 0 and b, we can represent 2^k words in standardized k length spaces. To do this, we may need some sort of stretch function. A stretch function can be made progressively, with for example the successive tape states:

- 0 1 1 0 b
- 0 b 1 1 0 b
- 0 b 1 b 1 0 b
- 0 b 1 b 1 b 0 b

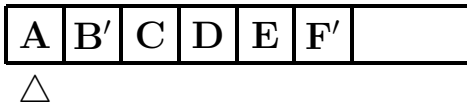
We also discussed $O(n) = cn$ time, where c is a constant that can be made smaller by making Γ bigger.

There are several generalizations of a standard Turing machine. One is where the Turing machine has multiple tape heads, as below:





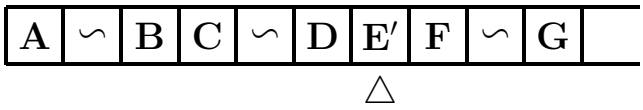
This can be simulated with a single tape (using bread crumbs to mark the position of both tape heads, as below:



Another modification is to allow two dimensional tape:

A	**B**	**D**	**G**	
C	**E′**	**H**		
F	**I**			
J				

This can be simulated using a single tape, where we use $\backsim$ for a carriage return.



It is less obvious how to simulate moving downward in a two dimensional tape this way, but it can be done by noticing that if a space is n past a carriage return, the space directly below it will occur $n + 1$ past the next carriage return.

We ended today with a restatement of the Modern Church-Turing Thesis: A Turing machine can simulate an (idealized) modern computer (where by idealized, we mean that it has no limits on memory.)