**TURING MACHINES** **Math118, O. Knill**

ABSTRACT. This is an excursion into a class of dynamical systems called Turing machines. They are remarkable because any computation can be done by Turing machines. Because Turing machines can be realized as subshifts and subshifts are abundant in dynamical systems theory, most dynamical systems like the Henon map would be capable to do any possible computation.

TURING MACHINES. A Turing machine is a dynamical system $(Y, T)$ defined as follows. Define $Y = X \times S = \{0,1\}^Z \times S$, where $S$ is a finite set of states. The set $S$ contains an element 0, which is called the halting state. The set $\{(\ldots,0,\ldots)\} \times S$ is called the empty tape. The set $X$ is the space of $0,1$ sequences for which only finitely many 1 are called data. The Turing machine is defined by three maps from finite sets to finite sets.

| | |
|---|---|
| $f : \{0,1\} \times S \to \{0,1\}$ | defines the new letter |
| $g : \{0,1\} \times S \to S$ | defines the new state |
| $h : \{0,1\} \times S \to \{-1,0,1\}$ | decides whether to move the tape to left, right or stay |

one can define now a continuous map on the compact metric space $Y$ by

$$T(x,s) = (\sigma^{h(x_0,s)}(\cdots, x_{-2}, x_{-1}, f(x_0,s), x_1, x_2 \cdots), g(x_0,s)) \; .$$

This dynamical system is called a Turing machine. Note that this is not a CA, since the map does not commute with the shift. But alread John von Neumann noticed that one can find for every Turing machine a CA, which simulates the Turing machine. Note that the set $Y$ is not compact but it is a subset of a compact set.

HALTING STATE The description of a Turing machine is given by a finite amount of information, because the three involved functions map finite sets into finite sets. The set $X \times \{0\}$ is called the halte set. One step of a Turing machine can be described as follows: the Turing machine with tape $x$ and state $s$ moves the tape $h(x,s)$ steps goes into the state $s$ and then writes the entry $f(x,s)$ at the position 0.

CHURCH THESES. Turing showed, that every computation which can be done by known computations can be done by Turing machines. The question of what actually can be computed is probably beyond the scope of mathematics. There is a widely accepted statement called the Church thesis (1934) which tells that everything which can be computed can be computed with a partial recursive function. Such functions can be computed by Turing machines. Everything we know to compute can be computed with partial recursive functions.

TURING MACHINES AS DATA. The set of pairs $(T, x)$ where $T$ is a Turing machine and $x \in X$ is an input data, is countable. We can encode therefore the set of such pairs into data $X$. Let $TM \subset X$ be the set of all the so obtained pairs $(T, x)$. Denote by $H$ the subset of $TM$, which consists of halting Turing machines.

DECIDABLE SETS IN TM. A subset $Z$ of $TM$ is called **decidable**, if there exists a Turing machine, which tells after finitely many steps, whether a given $x \in TM$ is in $Z$ or not.

THE HALTE PROBLEM IS NOT DECIDABLE.

THEOREM (Turing) The subset $H \subset TM$ of all halting Turing machines is not decidable.

PROOF. Assume the halting problem is decidable. Then there exists a Turing machine HALT which returns from the input $(T, x)$ the output HALT(T, x) = true, if $T$ halts with the input $x$ and otherwise returns HALT(T, x) = false. Turing constructs a Turing machine DIAGONAL, which has as an input an input $x$ and does the following

```
1) Read[x]
2) Define Stop=Halt[(x,x)];
3) While Stop==True repeat Stop:=True.
4) Print[Stop]
```

Now, either (DIAGONAL,DIAGONAL) is in the set $H$ or it is not.

(i) Assume first DIAGONAL is in $H$. Then the variable $Stop$ was $True$, which means that the program DIAGONAL runs for ever. So, Halt[(DIAGONAL,DIAGONAL)]=False, and DIAGONAL is not in $H$.

(ii) Assume now DIAGONAL is not in $H$ Then, the variable $Stop$ becomes $False$, which means that Halt[(DIAGONAL,DIAGONAL)]=true, which implies DIAGONAL is in $H$.

Since the assumption of the existence of a Turing machine HALT leads to a contradiction, a machine DIAGONAL can not exist. This argument of Turing is very similar to Cantors diagonal argument.

UNIVERSAL TURING MACHINE. Turing also showed the existence of a universal Turing machine. This is a machine which can simulate all Turing machines. The universal Turing machine takes a Turing machine with input $(T, x)$ as input and returns as output, the output of the machine $x$. What Turing showed 1936 means translated into the dynamical systems language:

The universal Turing machine can be realized as a dynamical system.

Indeed, there exists a compact set $X$ and a continuous transformation $T$ on $X$, such that for a subset $Z$ of $X$, $(Z, T)$ can do any computation in Mathematics. This tells us also that there are fundamental limitations, what can be said about dynamical systems in general. There are dynamical systems, so that we can not decide for a given set $U$ and a point $x$, whether $T^n(x)$ will every enter $U$ or not. Note that all said here about Turing machines is just rephrasing of what Turing knew 70 years ago already in an other language. This has to be said because there is literature which can give the impression that such statements are a new discovery.

BUSY BEAVER. The **busy beaver problem** is the task to construct a Turing machine which has $n$ states not counting the halting state 0 and satisfies the following: The machine starts on the empty tape and should write as many 1 onto the tape as possible before it stops. For $n = 1, 2, 3, 4$, the optimal solutions are known. For $n = 5$, Heiner Marxen has built a Turing machine in 1989 which produces 4098 marks. Its orbit is has length 11'798'826. You find a Mathematica program which simulates this Turing machine on the website.

REDDY'S THEOREM. A **topological dynamical system** is a pair $(X, T)$, where $X$ is a compact metric space and $T$ is a homeomorphism of $X$. (A homeomorphism is a map which is continuous and invertible and for which the inverse is continuous too). A topological dynamical system is called **expansive**, if there exists $\epsilon > 0$ such that for all $x \neq y \in X$, there exists $n$ such that $d(T^n x, T^n y) \geq \epsilon$. A dynamical system is called **zero dimensional** if $X$ is zero dimensional, that is if there is a basis for $X$ which consists of sets which are both open and closed. (A **basis** is is a set $B$ of subsets such that (i) the empty set is in $B$, arbitrary unions of sets in $B$ are in $B$, the intersection of two sets in $B$ is a union of sets in $B$.)

THEOREM (Reddy) A zero dimensional expansive dynamical system is isomorphic to a subshift.

PROOF (sketch) partition $X$ into $n$ sets $X_i$ which are both closed and open, such that each of the sets has diameter $\leq \epsilon$. An orbit $T^n(x)$ defines a code $y \in A^Z$, where $A$ labels the partition. The expansiveness assures that the encoding is injective.

THE TURING MACHINE AS A SUBSHIFT. We first change the Turing dynamical system to make it expansive. This can be done by a topological trick. The zero-dimensionality is assured already. The abstract theorem of Reddy shows that

COROLLARY. There is a subshift which can simulate the universal Turing machine.

Because a subshift is a subset of the shift and the shift can be realized in a dynamical system with a horse shoe, one obtains

COROLLARY. The map $T(x, y) = (-1.5x^2 - 0.3y, x)$ can simulate any computation.

Proof. An iterate $T^m$ of $T$ contains a horse shoe, on which the dynamics is conjugated to a shift of 2 symbols. The map $T^{mk}$ is on this set conjugated to a shift of $2^k$ symbols.

Again, it is important to state that such corollaries are nothing more than climbing onto the shoulders of Turing and other mathematicians working in topological dynamics. While there is nothing original in such statements, it is amusing. It also illustrates that dynamical systems have relations with the foundations of mathematics or what one sometimes calls the "theory of computation".