Overview

Goals.
- Attempt to understand essential nature of computation by studying properties of a simple machine model.
- Simplest machine that is "as powerful" as conventional computers.

Surprising Fact 1.

Surprising Fact 2.

---

Lecture T1: Turing Machines

Alan Turing (1912-1954)

Overview

Goals.
- Attempt to understand essential nature of computation by studying properties of a simple machine model.
- Simplest machine that is "as powerful" as conventional computers.

Surprising Fact 1.

Surprising Fact 2.

---

Turing Machine Prehistory

1922. Alan Turing receives E. T. Brewster’s book *Natural Wonders*.
  - Process of biological growth.
  - "For, of course, the body is a machine. It is a vastly complex machine, many, many times more complicated than any machine ever built by hands; but still after all a machine."

1926. Alan Turing struggles in school.

1928. Hilbert asked "Is mathematics decidable?"
  - "Did there exist a definitive method which could, in principle, be applied to any assertion, and which guaranteed to produce a correct decision as to whether that assertion was true."


---

Turing Machine Inspiration

Turing sought the most primitive model of a computing device.
- Device should have same basic capabilities as human computer!

Tape.
- Stores input, output, and intermediate results.
- 1 arbitrarily long strip, divided into cells.
- Finite alphabet of symbols.

Tape head.
- Points to one cell of tape.
- Reads a symbol from active cell.
- Overwrites a symbol to active cell.
- Moves left or right one cell at a time.
Fetch, Execute

**States.**
- Finite number of possible machine configurations.
- Determines what the machine does to the active cell and in which way the tape head moves afterwards.

**State transition diagram.**
- Ex. If Turing machine is in state 0 and the input symbol is A then:
  - overwrite the A with X
  - move to state 1
  - move tape head to right

---

**Beginning and Ending**

**Initialization.**
- Set input on some portion of tape.
- Set tape head.
- Set initial state.

**Termination.**
- Stop if enter yes, no, or halt state.
- Infinite loop possible.

---

**Unary to Binary Converter**

---
### Equal Number of 0’s and 1’s

```
...  #  # # 0 0 1 1 1 1 0 # # ...
```

### Dragon Curve

```
...  # 0 0 0 0 0 0 0 0 0 0 0 0 # ...  
...  #  L L R L R L R L R L R L R L R # ...
```

### Power of Two 1’s

```
...  # 1 1 1 1 1 1 1 1 1 # ...  
```

### C Program to Simulate Turing Machine

**Alphabet.**
- 128 ASCII symbols.

**N States.**
- Current state = state.

**Tape.**
- `tape[i]` is `i`th character on tape.
- Index of active cell = head.
- Might run off either end of tape.

```
int N; // number of states
int state; // current state
int head; // position of tape head
char tape[MAX_TAPE]; // contents of tape
```
State type.
- \( \text{states}[i] \in \{ 'L', 'R', 'Y', 'N', 'H' \} \)
- Left, right, yes, no, halt.

State transitions.
- If in state \( s \) and tape head points to character \( c \), then:
  - go to state \( \text{next}[s][c] \)
  - overwrite current cell with \( \text{write}[s][c] \)

```c
state[6] = 'L'
next [6]['a'] = 1
write[6]['a'] = 'b'
```

```
// simulating the Turing machine
done = 0;
while(!done) {
    c = tape[head];                      // read from tape
tape[head] = write[state][c];       // write to tape
    state = next[state][c];             // update state
    switch(states[state]) {             // move head
        case 'R':  head++;                      break;
        case 'L':  head--;                      break;
        case 'Y':  printf("Yes");  done = 1;    break;
        case 'N':  printf("No");   done = 1;    break;
        case 'H':  printf("Halt"); done = 1;    break;
    }
}
```

Universal Turing Machine

Each TM solves one particular problem.
- Ex: is the integer \( x \) prime?
- Analogy: computer algorithm.
- Similar to ancient special-purpose computers.

Goal: "general purpose TM" that can solve many problems.
- Simulate the operations of any special-purpose TM.
- Analogy: computer than can execute any algorithm.
- Similar to von Neumann stored-program computers.

Representation of a Turing Machine

Key idea.
- Store REPRESENTATION of special purpose TM on tape and use as input to the general-purpose TM.
- Analogous to von Neumann architecture.

![Turing Machine Diagram]
Universal Turing Machine

UTM.
- A specific TM that simulates operations of any TM.

How to create.
- Encode 3 ingredients of TM using 3 tapes.
- UTM simulates the TM.
  - read tape 1
  - read tape 3
  - consult tape 2 for what to do
  - write tape 1
  - move head tape 1
  - move head tape 3

Universal Turing Machine: Implications

Existence of UTM has profound implications.
- "Invention" of general-purpose computer.
  - anticipated development of von Neumann stored-program computers
- "Invention" of software.
- Universal framework for studying limitations of computing devices.
- Can simulate any machine (including itself)!

Implicit Physical Principles

Turing machine embodies physical constraints to which all concrete computational processes are subjected.

Fredkin-Toffoli axioms.
- The speed of propagation of information is bounded.
  - no "action at a distance"
  - TM head only move to adjacent cells
- The amount of information which can be encoded in the state of a finite system is bounded.
  - TM stores finitely many symbols per tape cell
- It is possible to construct macroscopic, dissipative physical devices which perform in a recognizable and reliable way the logical functions AND, OR, and FAN-OUT.
  - can fabricate TM out of physical parts, and run it reliably
Summary

Abstract machines are foundation of all modern computers.
- Simple computational models are easier to understand.
- Lead to deeper understanding of computation.

Goal: simplest machine "as powerful" as conventional computers.
- Turing machine = software.
- Universal Turing machine = general purpose computer.

Lecture T1: Extra Slides

Copy

Divisible-By-Three

<table>
<thead>
<tr>
<th>before</th>
<th>...</th>
<th>#</th>
<th>#</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>...</td>
<td>#</td>
<td>#</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>#</td>
<td>#</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>#</td>
</tr>
</tbody>
</table>