LESSON 70

Turing Machines and Computability

80 minutes (2 periods) | Computer Systems and Networks | Outcome 1.13 | Outcome 1.14 | Outcome 1.11 | Outcome 2.4 | Outcome 1.6 | Outcome 1.9 | Outcome 2.15

WHAT THIS LESSON DOES

In this lesson, you'll explore the fascinating world of Turing machines and computability. Learn what a Turing machine is, understand its historical importance, and discover the limits of what computers can solve through clear, step-by-step guidance.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.13	identify important computing developments that have taken place in the last 100 years and consider emerging trends that could shape future computing technologies
1.14	explain when and what machine learning and AI algorithms might be used in certain contexts
1.11	discuss the complex relationship between computing technologies and society including issues of ethics
2.4	illustrate examples of abstract models
1.6	explain the operation of a variety of algorithms
1.9	use modelling and simulation in relevant situations
2.15	explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components and communication protocols

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the fundamental concept and components of abstract computational models.
- Recognise the historical importance and impact of theoretical computing on modern technology.
- Grasp the boundaries of computational problem-solving and undecidable issues.
- Explore the notion of machine intelligence and its evaluation through conceptual tests.
- Connect theoretical computation principles to practical applications in computing and networking.

BEFORE THE LESSON

- This is a theory and pen-and-paper lesson, so no coding environment is needed. Have blank paper or grid paper ready for the practical task in step 4.
- Work through the even-number-of-1s Turing machine yourself first so you can model the state table and tape moves confidently, since many teachers meet this only briefly in their own training.

Turing Machines and Computability

HOW THE LESSON RUNS

- Introduction
- Historical Context
- What is a Turing Machine?
- Practical Task: Turing Machine
- Universal Turing Machines
- Limits of Computability
- Relevance to Networking and Computing
- Turing Test
- Wrap Up

TEACHING MOVES

- 2. Historical Context. Anchor the date: 1936, before any electronic computer existed. Stress that the Turing machine was a thought experiment answering a maths question, and that Bletchley Park and the Bombe came later. Keep it to the point that theory preceded hardware.
- 3. What is a Turing Machine? Draw the four parts on the board as you name them: infinite tape, read/write head, current state, transition rules. Emphasise it is not a physical machine. Run one transition live so they see read, write, move, change state as a single cycle.
- 4. Practical Task: Turing Machine. Have students draw the tape and a two-column state table before writing any rules. Circulate and ask each pair to trace their machine on input '101' out loud. Catch the common error of forgetting the blank cell halt condition.
- 5. Universal Turing Machines. Frame the UTM as the idea behind a stored-program computer: one fixed machine that reads a description of another machine as data. Link it to how their laptop runs any program without being rebuilt.
- 6. Limits of Computability. State clearly that undecidable does not mean hard, it means impossible for any algorithm ever. Give the halting problem in plain words: no program can decide for all programs whether they stop. Avoid attempting the full proof at this level.
- 7. Relevance to Networking and Computing. Connect back to state and transition: TCP/IP moves between connection states the way the machine moves between states. Keep this as an analogy, not a claim that TCP is a Turing machine.
- 8. Turing Test. Run a two-minute discussion: does passing the imitation game prove a machine thinks, or only that it imitates well? Draw out the distinction between behaviour and understanding rather than settling it.
- 9. Wrap Up. Ask students to state in one sentence each what a Turing machine is, what the halting problem shows, and what the Turing test measures. Use their answers to spot who has conflated the three.

WHAT TO WATCH FOR

- Students think undecidable means the problem is very difficult or would take a very long time. Contrast a slow but solvable problem with the halting problem. Undecidable means no correct algorithm can exist at all, not that we have not found a fast enough one yet.
- Students believe passing the Turing test proves a machine is genuinely intelligent or conscious. Point out the test measures indistinguishable behaviour in conversation only. Ask whether convincing imitation is the same as understanding, and note this is a live debate, not a settled fact.
- Students treat the Turing machine as a real early computer Turing built. Return to 1936 and the phrase thought experiment. It is a mathematical model to define computability, not hardware. The Bombe was a separate wartime device.

DIFFERENTIATION

- Emerging: For the paper task, give a shorter input such as '11' and a partly filled state table so the student completes the transitions rather than starting from nothing. Let them trace a machine you have already drawn before asking them to design one.
- Developing: Ask them to test their machine on a second input, such as '1011', and explain why it accepts or rejects, so they reason about the rules rather than following them once.