COMPUTATION: DAY 1

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Wednesday, January 23, 2026.

1. What is computation

1.1 If this course were to be summed up in a single idea, it would be the Turing Machine. The Turing Machine is Alan Turing's abstract device in the paper *On Computable Numbers, with an Application to the Entscheidungsproblem* (1936). By the time Turing began his work, Kurt Gödel (1931) had shown that some statements in arithmetic are true but unprovable. Turing was exploring what could be determined mechanically about such statements.

1.2 Since the time of Euclid, mathematicians assumed that all reasoning in a formal system preserved truth: starting from axioms and proceeding forward by inference rules. In the classical axiomatic view, all statements derivable from axioms were considered true (the system is sound[1]) and any true statement could be inferred from the axioms (the system is complete). Gödel showed that this need not be the case.

1.3 Turing took up this problem and showed that the boundary between what is provable and what is not provable is itself undecidable: there is no machine that can always determine provability for every statement. A proof can be represented as a process that runs and eventually halts on his machine; if the machine halts, it demonstrates both the statement itself and its provability. There is no analogous finite process that can decide, for all statements, whether a proof exists.

1.4 Colloquially, a Turing machine may run indefinitely without halting, so it is impossible to determine in general whether it will ever halt. This has consequences for computer programs: no general method exists to prove the correctness of arbitrary programs. Correctness can only be guaranteed for restricted classes of programs or for limited properties that do not encompass all possible computations.

[1]Actually, Gödel's proof concerns consistency: it guarantees that no contradiction can be proved (i.e., no statement ϕ exists such that both ϕ and $\neg\phi$ are provable).

2. What is a Finite Automata

2.1 A Turing machine is a device that computes by deterministic logical rules. According to the Church–Turing thesis, it can emulate any system that computes by such step-by-step rules. The machine consists of a memory tape and a computational element called a finite automaton. In this class, we will first examine the computational abilities of the finite automaton without using the memory tape.

2.2 A finite state automaton (FSA) is a computational model consisting of a finite set of states, of which the machine is in exactly one at any given time. The machine reads a string of symbols from a finite alphabet, one symbol at a time. On reading a symbol, the machine transitions to a new state according to a function of the current state and the input symbol. By designating a fixed start state and a set of final states, the FSA can serve as a decision machine: it starts in the start state, processes the input string symbol by symbol, and at the end accepts or rejects the string depending on whether the resulting state is among the final states.

2.3 At this point, we introduce mathematical notation to formalize the finite state automaton described in the preceding paragraph, and to prepare the student for the mathematical approach that will be central to this course.

Definition 2.1 (Finite State Automaton)**.** A finite state automaton (FSA) is a 5-tuple

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

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function, which specifies the next state for each current state and input symbol,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accepting (or final) states.

2.4 In the example of Figure 1, the following conventions are used. The start state q_0 is indicated by an arrow pointing to it. Final states, such as q_1, are indicated by a double circle. Transitions, such as $\delta(q_0, a) = q_1$, are depicted by arrows between the states, labeled with the input symbol that triggers the transition.

Definition 2.2 (Language of a Finite State Automaton)**.** Given a finite state automaton $M = (Q, \Sigma, \delta, q_0, F)$, the *language recognized by* M is the set $\mathcal{L}(M) \subseteq \Sigma^*$

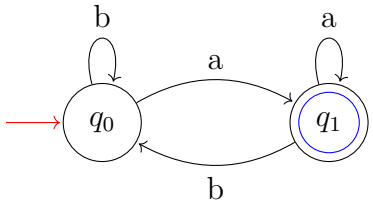

FIGURE 1. A finite state automaton accepting all strings ending with the symbol a.

such that,

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \text{the machine } M \text{ ends in a state of } F \text{ after reading } w\}.$$

Note that Σ^* denotes the set of all finite strings over the alphabet Σ, including the empty string ε.

Exercise: What is the language of the FSA in Figure 1.

3. THINK-A-DOT

3.1 The Think-A-Dot was created by Joseph Weisbecker (patent US566881A,1966) and marketed by E.S.R. Inc, as a *Computer Action game*. A plastic box has eight circular windows showing yellow or blue. A marble enters from one of three holes in the top, travels past three of the windows flipping the color and exiting from one of two holes in the bottom. The marble is deflected left or right, the direction according to the current color of the dot, and the color flips.

3.2 The Think-A-Dot FSA has $|Q| = 2^8$ states, according to the pattern of colors: eight windows each one of two colors. The alphabet Σ is left, center or right, according to the hole in which the marble enters. The transition function is form the color pattern at before the marble enters to the color pattern when the marble leaves. The Think-A-Dot, tilted left or right sets the color pattern. Choose one as the start state. Choose any color pattern as a final state. The language of the Think-A-Dot is then all sequences form,

$$\mathcal{L}(\text{Think-A-Dot}) = \{\,\text{left}, \text{center}, \text{right}\,\}^*$$

that leads from the starting pattern to the desired ending pattern.

3.3 The instruction manual[2] to Think-a-Dot already notes something of interest,

[2]Think a Dot flyer in box.

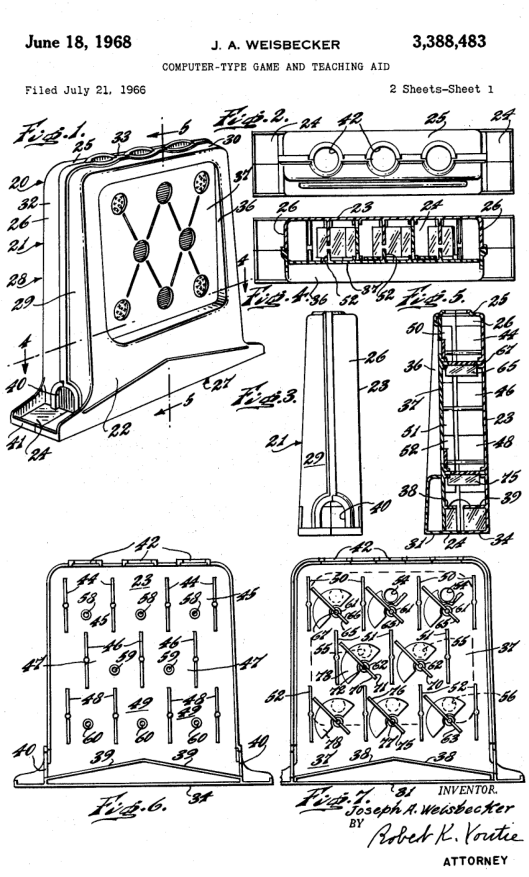

FIGURE 2. Think-A-Dot US566881A filed 1966

> *Think-A-Dot can also count in a manner similar to large computers. Remember the color of the top-right, middle-right, and bottom-right dots. Now drop the marble into the right-hand hole eight times. The eighth time all three dots will return to their original colors! This shows that Think-A-Dot can count and remember that you dropped the marble through the same home eight times in a row.*

The game is to find a sequence from $\mathcal{L}$(Think-A-Dot) where the ending configuration is all blue (for instance). For the analysis I wish to thank Jaap Scherphuis for his webpage. [3]

[3]Jaap Scherphuis `https://www.jaapsch.net/puzzles/thinkadot.htm`

Lemma 3.1. The colors of the first row of the Think-A-Dot are unchanged when a marble is dropped an even number of times in a given hole. The colors in the second row the Think-A-Dot are unchanged when a marble is dropped four times in a given hole. The colors in the third row the Think-A-Dot are unchanged when a marble is dropped eight times in a given hole.

Lemma 3.2. A state transition caused by a marble can be undone by dropping seven marbles into the same hole.

Theorem 3.1. The FSA of Think-A-Dot is a strongly connected directed graph.

Exercise: Develop a strategy for playing Think-A-Dot. For instance, from either of the a starting points to all blue or all yellow.

4. Dr. Nim

4.1 The Dr. Nim toy was patented by John Thomas "Jack" Godfrey[4] in 1968[5], and marketed by the E.S.R. Corporation of Montclair N.J. a round 1966.[6] The company describes Dr. Nim as *a "binary digital computer" specially designed to play the game of Dr. Nim.*[7] The game board a plastic table a little bit similar to a Pinball machine as it has channels to guide marbles and 6 plastic pieces moved by the traveling marble. A plastic piece when tilted left direct marbles to a pusher that will release a following marble, allow Dr. Nim to take multiple marbles in a turn. When the piece is struck by a marble on the left, it is forced to tilt right and Dr. Nim's turn is over.

4.2 There are three wing shaped pieces interlocking ways to cycle through 4 configurations. With each wing labelled a, b, c, capitalized if the wing is flipped up the states are,

$$Q = \{\, abc, Abc, aBc, abC \,\}$$

with the FSA show in Figure 4. The alphabet is $\Sigma = \{\, x \,\}$. The marked final state is that corresponding to a winning strategy.

4.3 The standard setup is to start with 15 marbles and the player and Dr. Nim take turns launching one to three marbles down the game table. The player taking the

[4] `https://www.legacy.com/us/obituaries/lsj/name/john-godfrey-obituary?id=23927195`

[5] Patent 3390471A, 1968

[6] *E.S.R., Inc was founded several years ago by three scientists and engineers who wanted to provide toys and educational devices that would have those qualities that lead children and adults to fuller and more successful lives — and yet bring enjoyment to all.*

[7] *How to play Dr. Nim*, `https://www.cs.miami.edu/home/burt/learning/csc427.242/docs/491_Dr-Nim-Manual5b15d.pdf`, 1966.

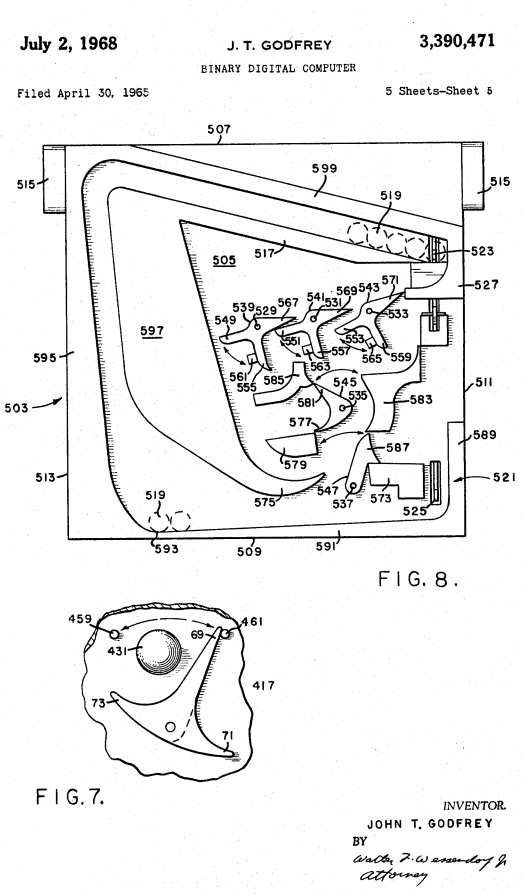

FIGURE 3. Dr. Nim US3390471, 1968

last marble losses. The strategy for winning can be diagramed with a list of marbles painted white or black. Whether it be Dr. Nim or his human opponent, if they begin their turn with a black marble perfect game-play will end with a lose for that player. However, in the case of a white marble, that player playing correctly will go on to win. See Figure 5.

Exercise: Show that if a player is given the FSA in a accepting state, that player will loose the game. If however, a player is given the FSA in a non-accepting state, that player will go on to win the game.

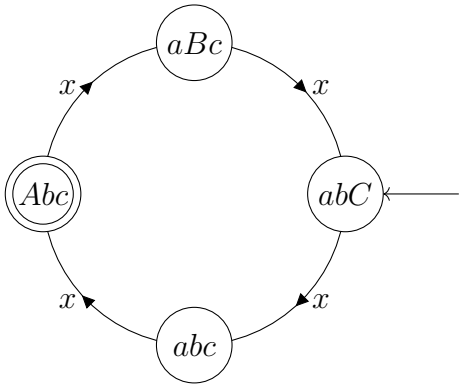

FIGURE 4. Dr. Nim as a finite automata

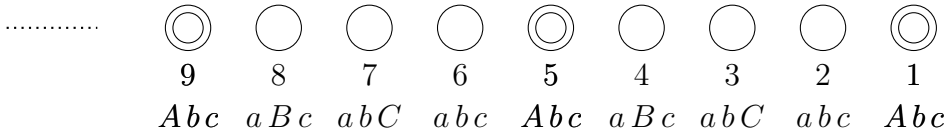

FIGURE 5. Dr. Nim strategy

5. Kleene algebra

5.1 Let $\mathcal{K}$ be some sort of collection, for instance, elements of strings over Σ, called Σ^* (the notational similarity to the Kleene star is not accidental). The defined operations on this collection are *concatenation*, *union* and *Kleene star*,

5.2 Concatenation: If $K, K' \in \mathcal{K}$ then $K \cdot K' \in \mathcal{K}$, and the operation is associative. In the interpretation in Σ^*, it is the set resulting as,

$$K \cdot K' = \{\, k \cdot k' \,|\, \forall k \in K, \forall k' \in K' \,\},$$

the string that begins with all the letters in k, in order, immediately followed by all the letters in k', in order. In an analogy to computation, this is creating a computation from the in-order sequencing of sub-computations.

5.3 Union: If $K, K' \in \mathcal{K}$ then $K \cup K' \in \mathcal{K}$, and the operation is associative, commutative and idempotent. In the interpretation in Σ^*, it is the set resulting as,

$$K \cup K' = \{\, k \,|\, \forall k \in K \cup K' \,\},$$

any and all strings that are in K or in K'. In an analogy to computation, this is creating a computation from a choice of two computations, such as any branching construct.

5.4 Kleene star: If $K \in \mathcal{K}$ then $K^* \in \mathcal{K}$. In the interpretation in Σ^*, it is defined as,

$$K^0 = \{\varepsilon\}, \qquad K^{n+1} = K \cdot K^n, \qquad K^* = \bigcup_{n \geq 0} K^n.$$

any and all strings that can be formed by taking a finite set of strings from K and concatenating them. In the analogy of computation, this is iteration, or a looping construct. The definition is remarkable because the union is infinite.

5.5 Unit for Concatenation: We also require the existence of a unique unit object for concatenation. In the interpretation of Σ^*, this is the singleton set containing only the empty string ε for which,

$$\{\varepsilon\} \cdot K = K \cdot \{\varepsilon\} = K.$$

5.6 Unit for Union: We also require the existence of a unique unit object for union. In the interpretation of Σ^*, this is the empty set $\emptyset$ for which,

$$\emptyset \cup K = K \cup \emptyset = K.$$

Note that,

$$\emptyset^* = \{\varepsilon\}.$$

5.7 Distributive law: For all $K, K', K'' \in \mathcal{K}$,

$$\begin{aligned} K \cdot (K' \cup K'') &= (K \cdot K') \cup (K \cdot K'') \\ (K' \cup K'') \cdot K &= (K' \cdot K) \cup (K'' \cdot K) \end{aligned}$$

5.8 Regular languages: A *language* is a subset of Σ^*. A class of languages in a collection of languages often defined by some property holding for every language in the collection. For example, a *regular language*[8] are a language whose membership is decided by a finite state automata, equivalently, that are described by a Regular Expression.

5.9 Classes of languages that are Kleene algebras are of interest because they model the computational notions of composition, choice and iteration. The class of Regular languages is a Kleene algebra.

5.10 Kleene algebras do not define intersection. To note is that the distributive law does not work for intersection. Instead,

$$K_1 \cdot (K_2 \cap K_3) \subseteq (K_1 \cdot K_2) \cap (K_1 \cdot K_3)$$

An example is $K_1 = \{\varepsilon, a\}, K_2 = \{a\}, K_3 = \{aa\}$.

[8] *Representation of Events in Nerve Nets and Finite Automata*, Stephen Kleene. RAND Corporation RM-704, 1951.