Georg Cantor's Cardinality Results

Size of a set

- Size of a finite set can be determined by counting.
  - A robust way of comparing the sizes of two large sets is by pairing elements in them.

- For comparing the sizes of two infinite sets, we need to resort to pairing elements in them.

Cardinality (size) of a set

- Two sets are defined to have the same size (or cardinality) *if and only if* they can be placed in one-to-one correspondence with each other.

- A set is *countable* if it can be put in one-to-one correspondence with the set of natural numbers N.

Example

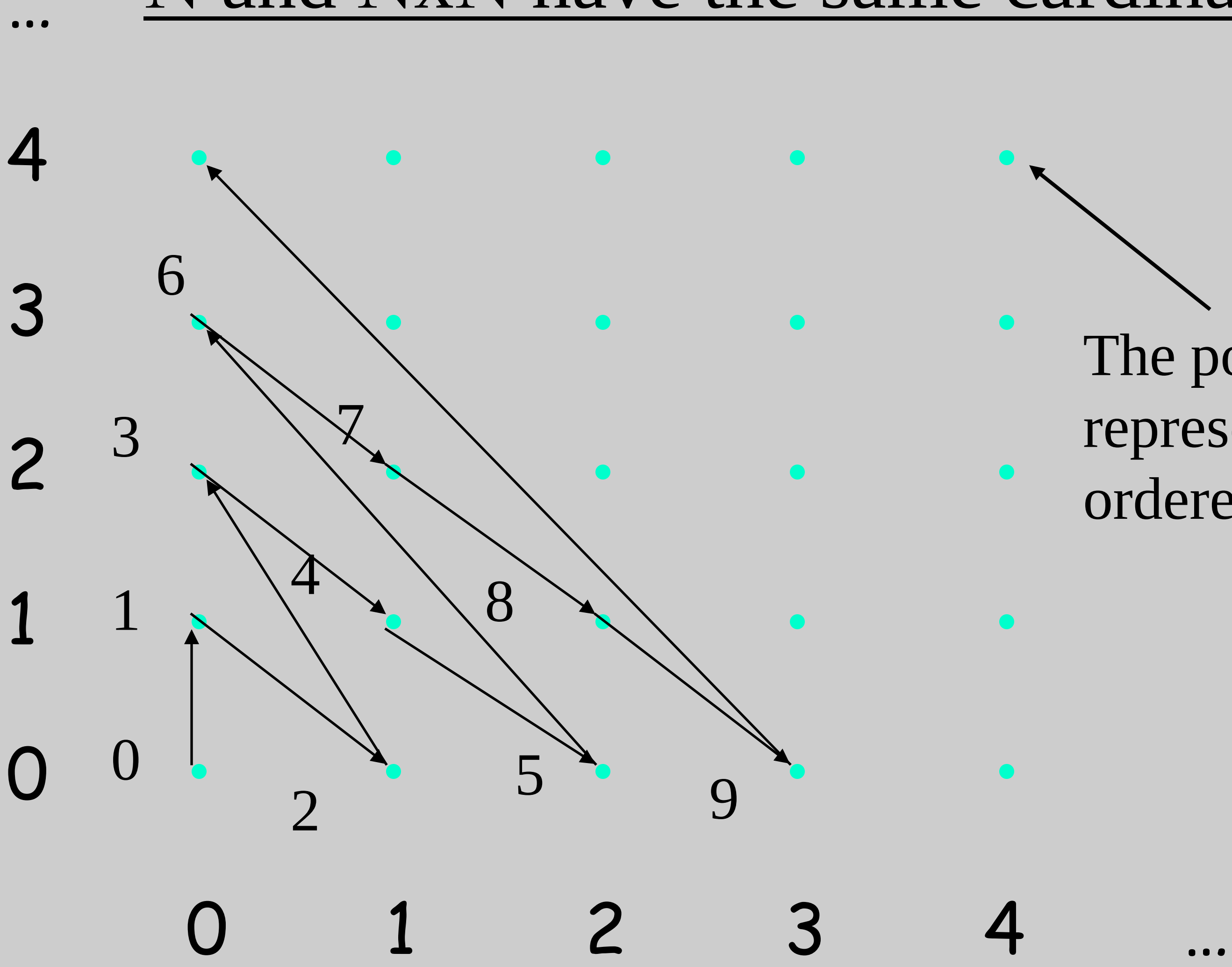
- Set of even numbers is countable.
  - 0, 1, 2, 3, 4, …
  - 0, 2, 4, 6, 8, …

- Formally, define f:N->E as f(n) = 2*n pairs each natural number with a unique even number such that it is one-to-one and onto.

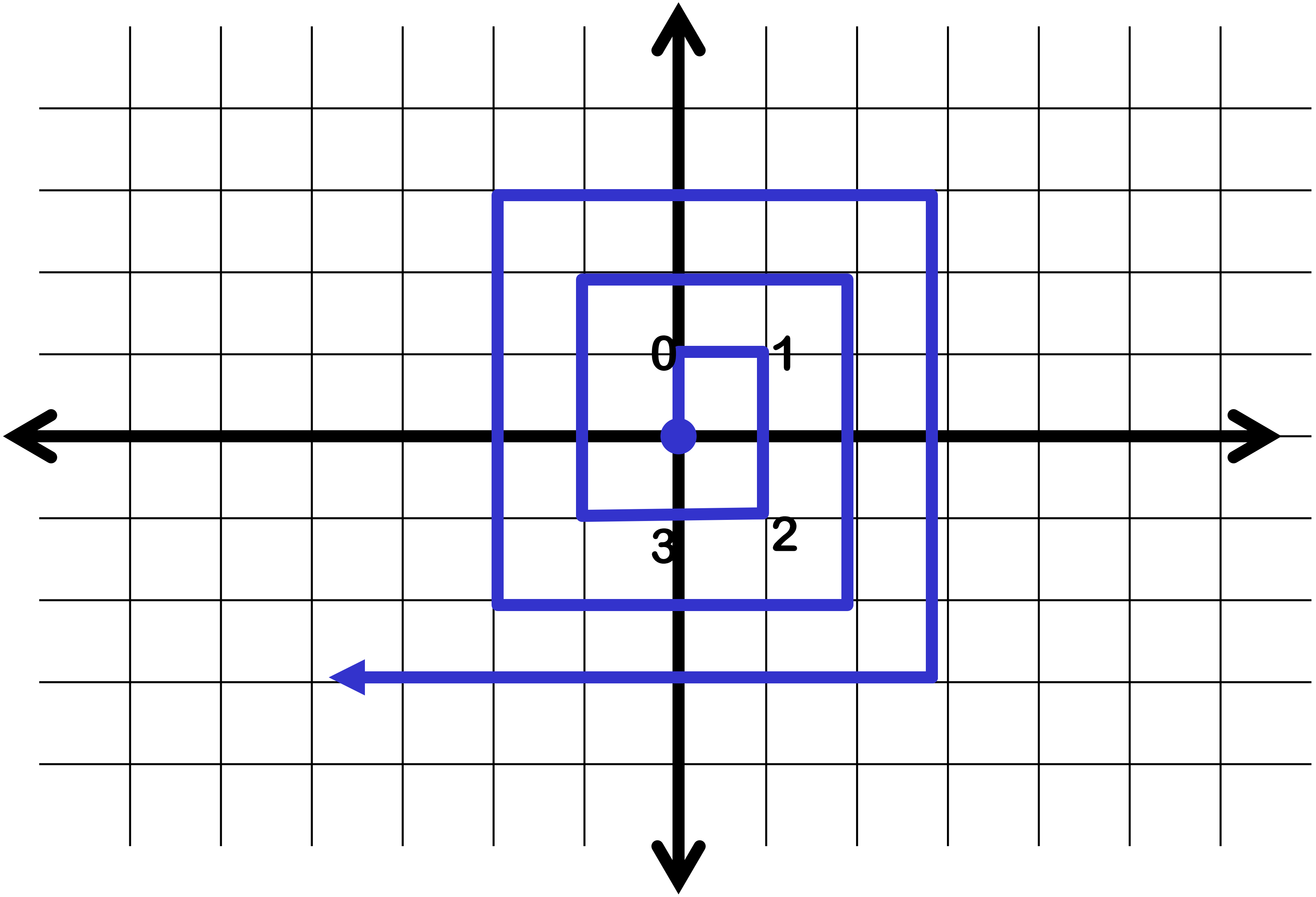
Examples of countable sets

- Set of integers.
- Set of strings over English alphabet.
- Set of pairs of natural numbers N x N.
- Set of rational numbers.
  - Note that rationals are *dense*, that is, between any pair of distinct rationals, there is a rational.

N and NxN have the same cardinality



The point (x,y) represents the ordered pair (x,y)



The point at x,y represents x/y

Examples of Uncountable Sets

- Set of reals between 0 and 1
- Set of characteristic functions

  `N -> {0,1}`

- Set of subsets of N = Poweset(N).
- Set of reals.
- Set of functions `N -> N.`

Fundamental Issues in Computability

- Given that the set of functions over N is uncountable and the set of algorithms (strings of ASCII characters) are countable, *which functions over N are computable*, that is, can be mechanized?
- How do we *characterize* the set of computable functions?
  - TURING MACHINES
- How do we show that it is *impossible* to compute a certain function?

From CS466: Formal Specification of Languages

- Generator
  - Regular Expressions
- Recognizer
  - Finite State Automata

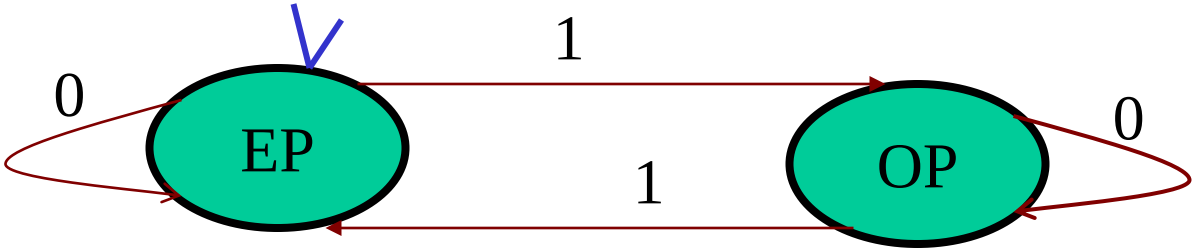
- *Compiler Connection*: FSA is a notation for describing a family of language recognition algorithms that can be specified using regular expressions.

Finite State Automata

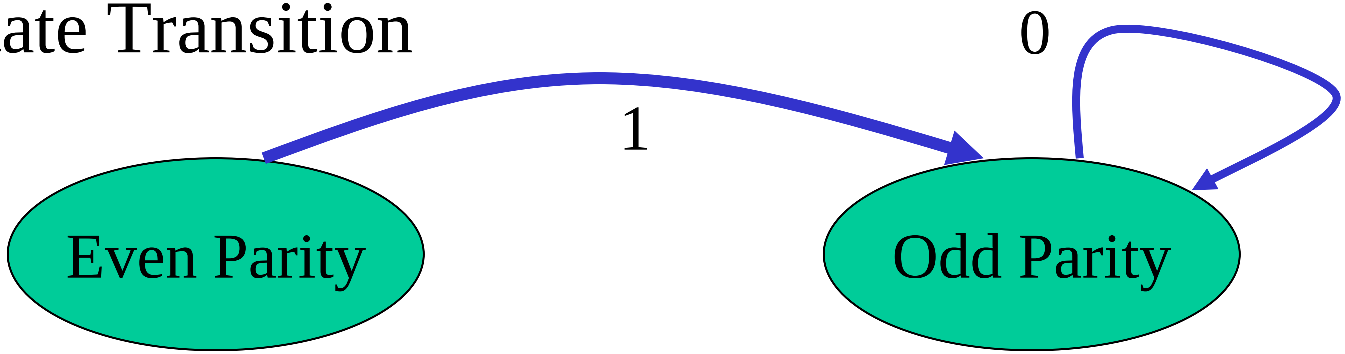
Even Parity Recognizer

Problem: Recognize the language of bit strings that contain an even number of 1s.

Strategy: Read the string from left to right remembering whether even or odd number of 1s were scanned so far (no need to count the number of ones).



- State
  - Indicates the status of the machine after consuming some portion of the input.
  - Summarizes the history of the computation that is relevant to the future course of action.
- Initial / Start State
- Final / Accepting state
- State Transition



Deterministic Finite State Automaton (DFA)

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

Q: Finite set of states

Σ: Finite Alphabet

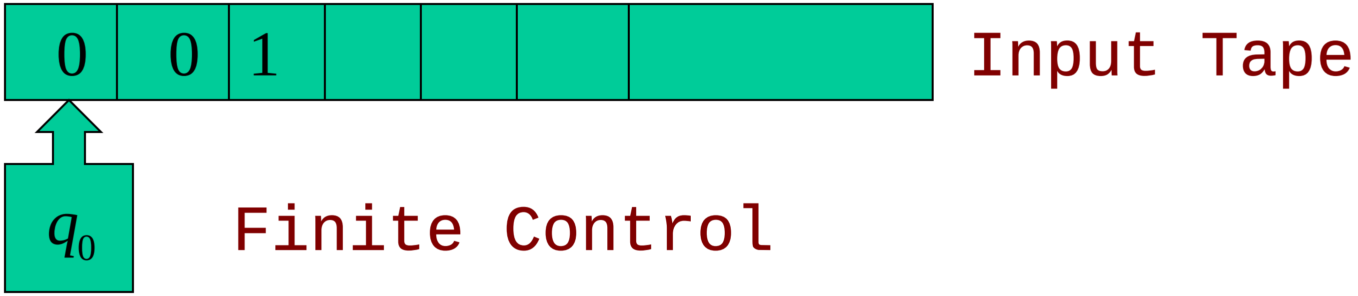
δ: Transition function

total function from QxΣ to Q

q_0: Initial/Start State

F : Set of final/accepting state

Operation of the machine



- Read the current letter of input under the tape head.
- Transit to a new state depending on the current input and the current state, as dictated by the transition function.
- Halt after consuming the entire input.

Associating Language with the DFA

- Machine configuration:

$$[q,\omega] \text{ where } q \in Q, \omega \in \Sigma^*$$

- Yields relation:

$$[q,a\omega] \mapsto^*_{M} [\delta(q,a),\omega]$$

- Language:

$$\{\omega \in \Sigma^* \mid \underbrace{[q_0,\omega] \mapsto^*_{M} [q,\lambda]} \wedge q \in F\}$$

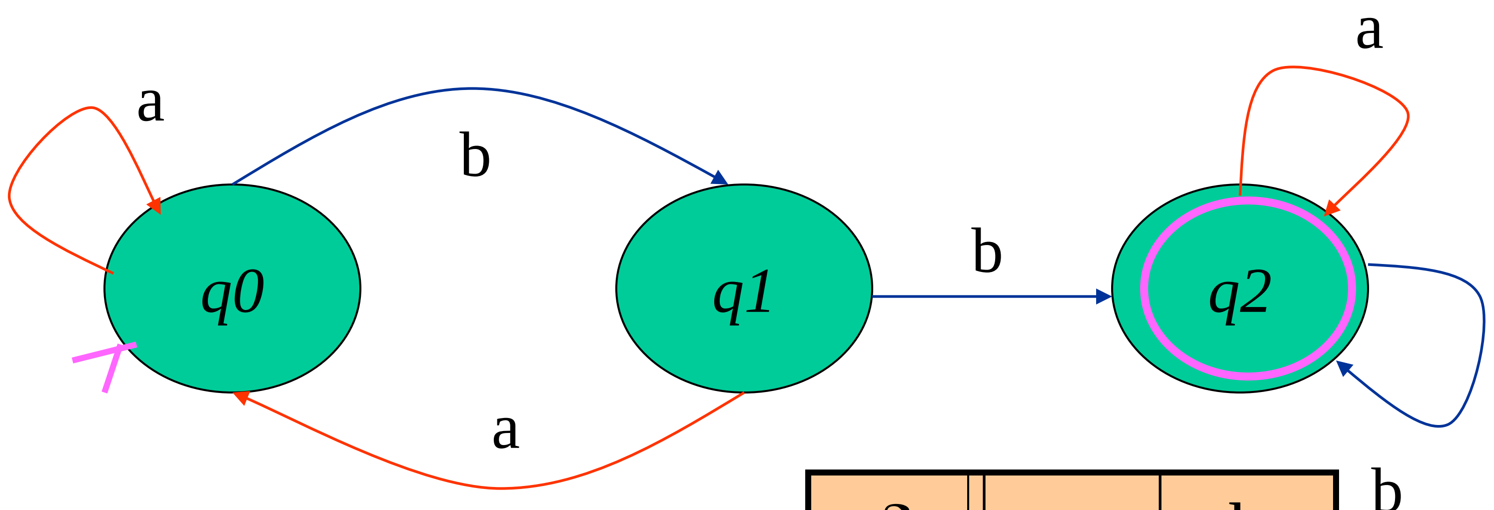
Examples

- Set of strings over $\{a,b\}$ that contain bb

$$(a \cup b)^* bb (a \cup b)^*$$

- Design states by parititioning Σ^*.
  - Strings containing bb — $q2$
  - Strings not containing bb
    - – Strings that end in b — $q1$
    - – Strings that do not end in b — $q0$
- – Initial state: $q0$
- – Final state: $q2$

State Diagram and Table



$$Q = \{q0, q1, q2\}$$

$$\Sigma = \{a, b\}$$

$$F = \{q2\}$$

$$[q0, aab] \mapsto^* [q1, \lambda]$$

δ	a	b
q0	*q0*	*q1*
q1	*q0*	*q2*
q2	*q2*	*q2*

Strings over {*a*,*b*} containing even number of *a*'s and odd number of *b*'s.