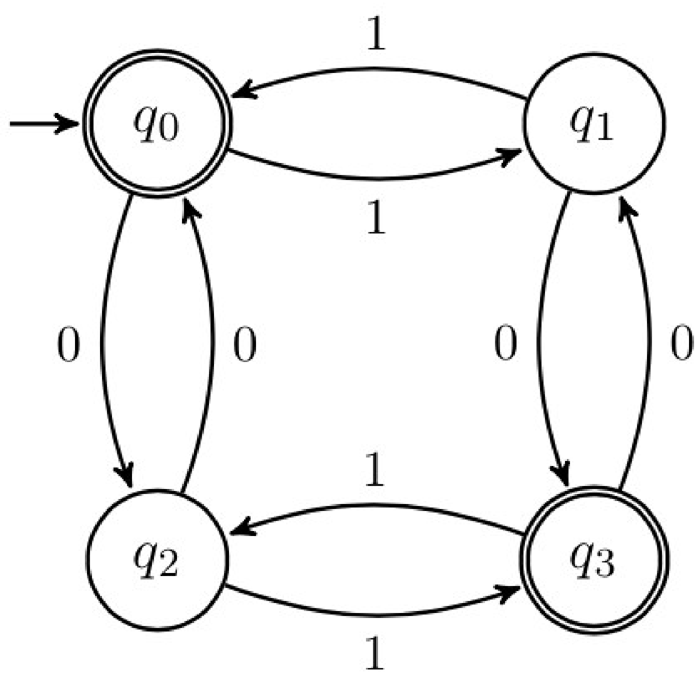
CSE 311: Foundations of Computing

Lecture 21: Directed Graphs & Finite State Machines



Last Class: Relations & Composition

Let A and B be sets,
A **binary relation from** A **to** B is a subset of $A \times B$

Let A be a set,
A **binary relation on** A is a subset of $A \times A$

The composition of relation R and S, $S \circ R$ is the relation defined by:

$S \circ R = \{\, (a, c) \mid \exists\, b$ such that $(a,b) \in R$ and $(b,c) \in S \,\}$

Last Class: Powers of a Relation

R^0 is identity for $\circ$

$$R^0 = \{(a,a) \mid a \in A\}$$ "the equality relation on A"

$a=b$

$(a,c) \in R \circ R^0$ iff $\exists b$ $(a,b) \in R^0$ and $(b,c) \in R$

iff $(a,a) \in R^0$ and $(a,c) \in R$

iff $(a,c) \in R$.

$$R^{n+1} = R^n \circ R \text{ for } n \geq 0$$

Last Class: Directed Graphs

$\cancel{\subset A \times B}$

$\subset A \times A$

$E \subset V \times V$.

G = (V, E) V – vertices

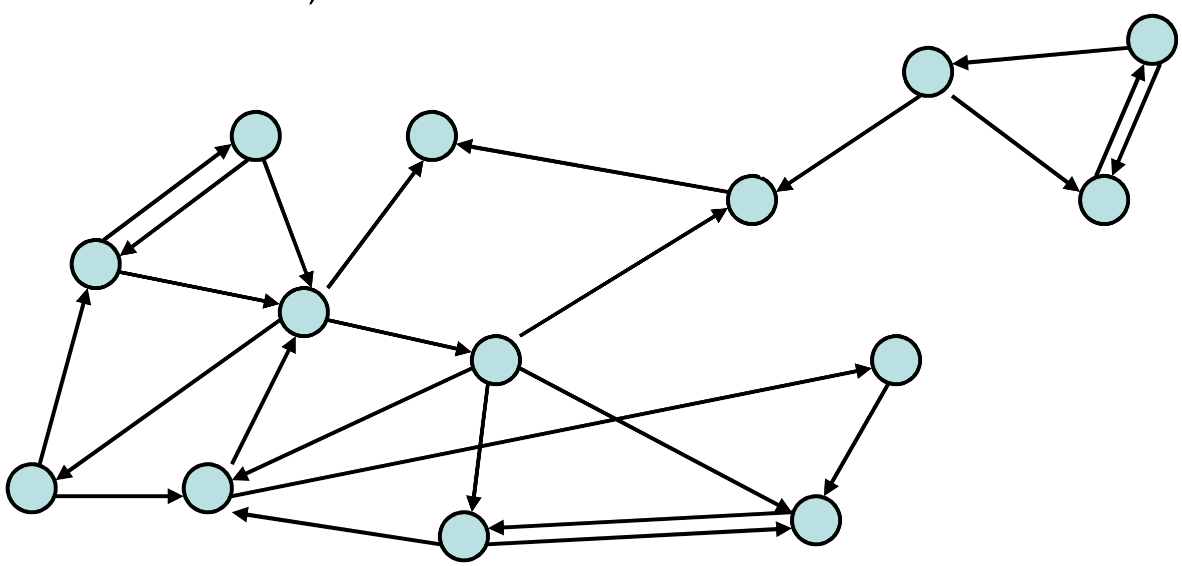
E – edges, ordered pairs of vertices

Path: $v_0, v_1, \ldots, v_k$ with each (v_i, v_{i+1}) in E

Simple Path: none of $\mathbf{v_0}, \ldots, \mathbf{v_k}$ repeated

Cycle: $\mathbf{v_0} = \mathbf{v_k}$

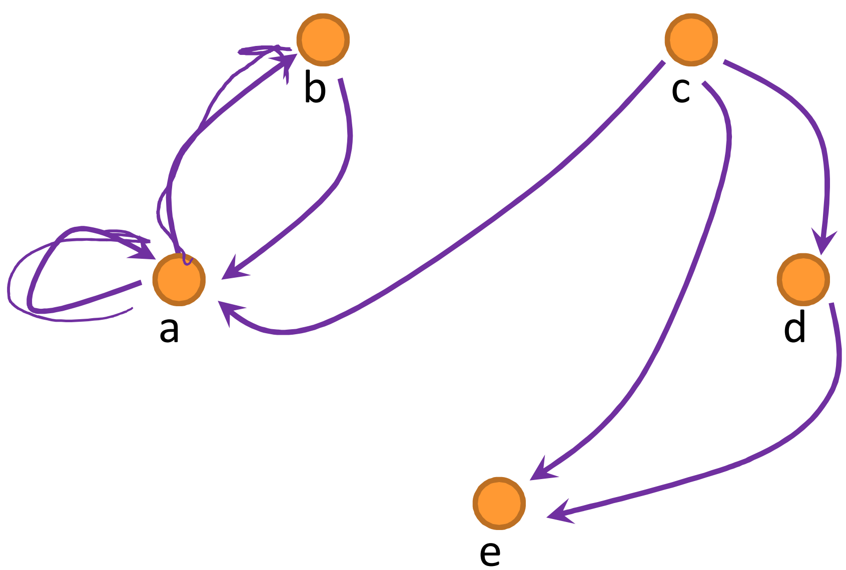
Simple Cycle: $\mathbf{v_0} = \mathbf{v_k}$, none of $\mathbf{v_1}, \ldots, \mathbf{v_k}$ repeated



Last Class: Representation of Relations

Directed Graph Representation (Digraph)

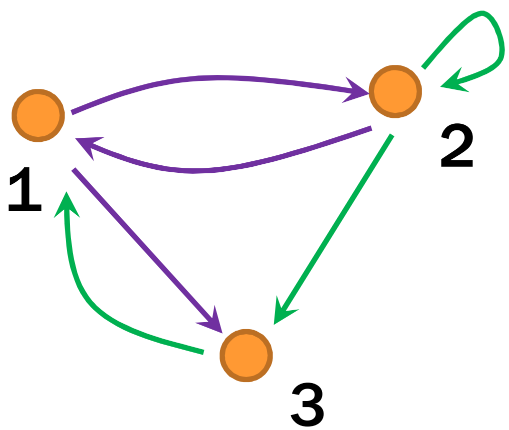
{(a, b), (a, a), (b, a), (c, a), (c, d), (c, e) (d, e) }

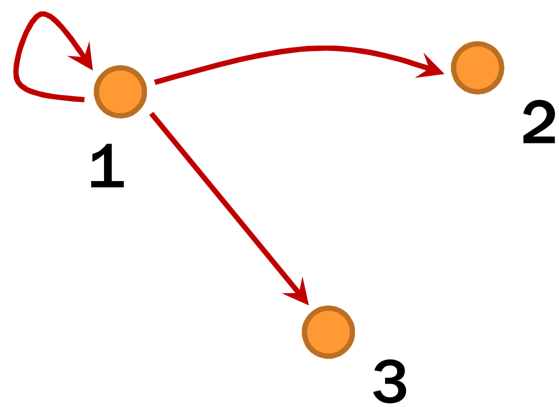


Last Class: Relational Composition using Digraphs

If $S = \{(2,2),(2,3),(3,1)\}$ and $R = \{(1,2),(2,1),(1,3)\}$

Compute $S \circ R$





Paths in Relations and Graphs

$(v_0, v_1), \ldots, (v_{k-1}, v_k)$

$v_0, v_1, \ldots, v_k$.

Def: The **length** of a path in a graph is the number of edges in it (counting repetitions if edge used > once).

Let $\boldsymbol{R}$ be a relation on a set $\boldsymbol{A}$. There is a path of length $\boldsymbol{n}$ from **a** to **b** in the digraph for $\boldsymbol{R}$ if and only if (**a**,**b**) $\in \boldsymbol{R}^{n}$

$R^2 = R \circ R$. $(a,c) \in R^2$ iff $\exists b$ $(a,b) \in R \wedge (b,c) \in R$.

iff $\exists b$ a, b, c is a path.

R^n $(v_0, v_n) \in R^n$ iff $\exists v_1, \ldots, v_{n-1}$ s.t. $(v_i, v_{i+1}) \in R$.

iff $\exists v_1, \ldots, v_{n-1}$ $v_0, v_1, \ldots, v_n$ is a path

Connectivity In Graphs

Def: Two vertices in a graph are **connected** iff there is a path between them.

Let $\boldsymbol{R}$ be a relation on a set $\boldsymbol{A}$. The **connectivity** relation $\boldsymbol{R}^*$ consists of the pairs (a,b) such that there is a path from a to b in $\boldsymbol{R}$.

$$\boldsymbol{R}^* = \bigcup_{k=0}^{\infty} \boldsymbol{R}^k$$

Note: Rosen text uses the wrong definition of this quantity. What the text defines (ignoring k=0) is usually called R^+

How Properties of Relations show up in Graphs

binary

Let R be a relation on A.

R is **reflexive** iff $(a,a) \in R$ for every $a \in A$

R is **symmetric** iff $(a,b) \in R$ implies $(b, a) \in R$

R is **antisymmetric** iff $(a,b) \in R$ and $a \neq b$ implies $(b,a) \notin R$

R is **transitive** iff $(a,b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$

How Properties of Relations show up in Graphs

Let R be a relation on A.

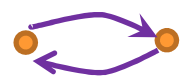
RST = "equivalence relation" =
RAT = "partial order" $\leq$, $<$

R is **reflexive** iff $(a,a) \in R$ for every $a \in A$

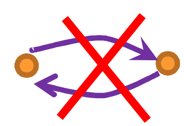
at every node

R is **symmetric** iff $(a,b) \in R$ implies $(b, a) \in R$

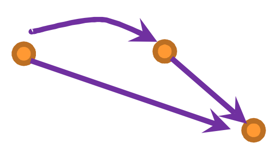
or



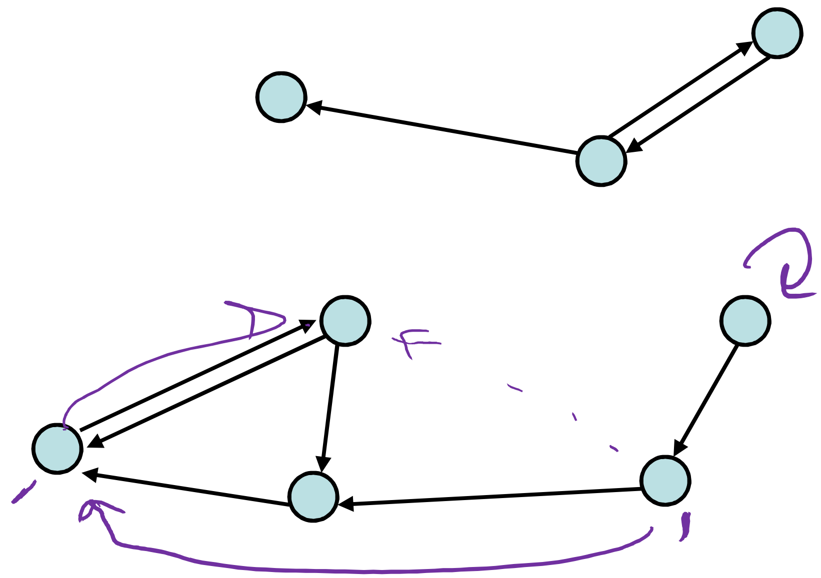
R is **antisymmetric** iff $(a,b) \in R$ and $a \neq b$ implies $(b,a) \notin R$



R is **transitive** iff $(a,b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$



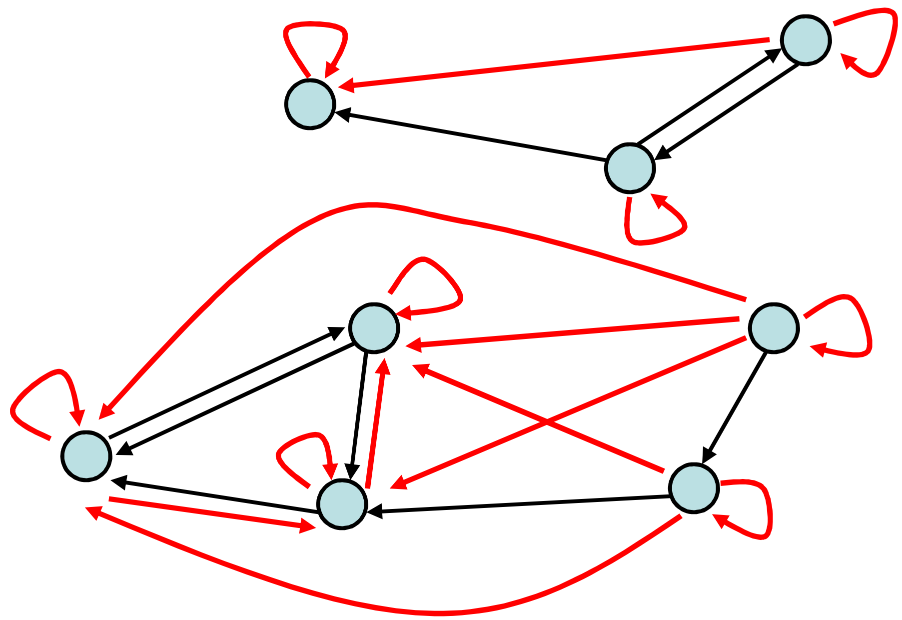
Transitive-Reflexive Closure



Add the **minimum possible** number of edges to make the relation transitive and reflexive.

The **transitive-reflexive closure** of a relation $\boldsymbol{R}$ is the connectivity relation $\boldsymbol{R}^*$

Transitive-Reflexive Closure



Relation with the **minimum possible** number of **extra edges** to make the relation both transitive and reflexive.

The **transitive-reflexive closure** of a relation $\boldsymbol{R}$ is the connectivity relation $\boldsymbol{R}^*$

n-ary Relations

Let $\boldsymbol{A_1, A_2, \ldots, A_n}$ be sets. An $\boldsymbol{n}$**-ary** relation on these sets is a subset of $\boldsymbol{A_1 \times A_2 \times \cdots \times A_n}$.

Relational Databases

$\text{Strings} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{R}.$

STUDENT

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Von Neuman	481080220	555	3.78
Russell	238082388	022	3.85
Einstein	238001920	022	2.11
Newton	1727017	333	3.61
Karp	348882811	022	3.98
Bernoulli	2921938	022	3.21

Relational Databases

STUDENT

Student_Name	ID_Number	Office	GPA	Course
Knuth	328012098	022	4.00	CSE311
Knuth	328012098	022	4.00	CSE351
Von Neuman	481080220	555	3.78	CSE311
Russell	238082388	022	3.85	CSE312
Russell	238082388	022	3.85	CSE344
Russell	238082388	022	3.85	CSE351
Newton	1727017	333	3.61	CSE312
Karp	348882811	022	3.98	CSE311
Karp	348882811	022	3.98	CSE312
Karp	348882811	022	3.98	CSE344
Karp	348882811	022	3.98	CSE351
Bernoulli	2921938	022	3.21	CSE351

What's not so nice?

Relational Databases

STUDENT

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Von Neuman	481080220	555	3.78
Russell	238082388	022	3.85
Einstein	238001920	022	2.11
Newton	1727017	333	3.61
Karp	348882811	022	3.98
Bernoulli	2921938	022	3.21

TAKES

ID_Number	Course
328012098	CSE311
328012098	CSE351
481080220	CSE311
238082388	CSE312
238082388	CSE344
238082388	CSE351
1727017	CSE312
348882811	CSE311
348882811	CSE312
348882811	CSE344
348882811	CSE351
2921938	CSE351

Better

Database Operations: Projection

Find all offices: $\Pi_{\mathbf{Office}}(\mathrm{STUDENT})$

Office
022
555
333

Find offices and GPAs: $\Pi_{\mathbf{Office,GPA}}(\mathrm{STUDENT})$

Office	GPA
022	4.00
555	3.78
022	3.85
022	2.11
333	3.61
022	3.98
022	3.21

Database Operations: Selection

Find students with GPA > 3.9 : $\boldsymbol{\sigma}_{\mathbf{GPA>3.9}}$(STUDENT)

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Karp	348882811	022	3.98

Retrieve the name and GPA for students with GPA > 3.9:
$\boldsymbol{\Pi}_{\mathbf{Student\_Name,GPA}}(\boldsymbol{\sigma}_{\mathbf{GPA>3.9}}$(STUDENT))

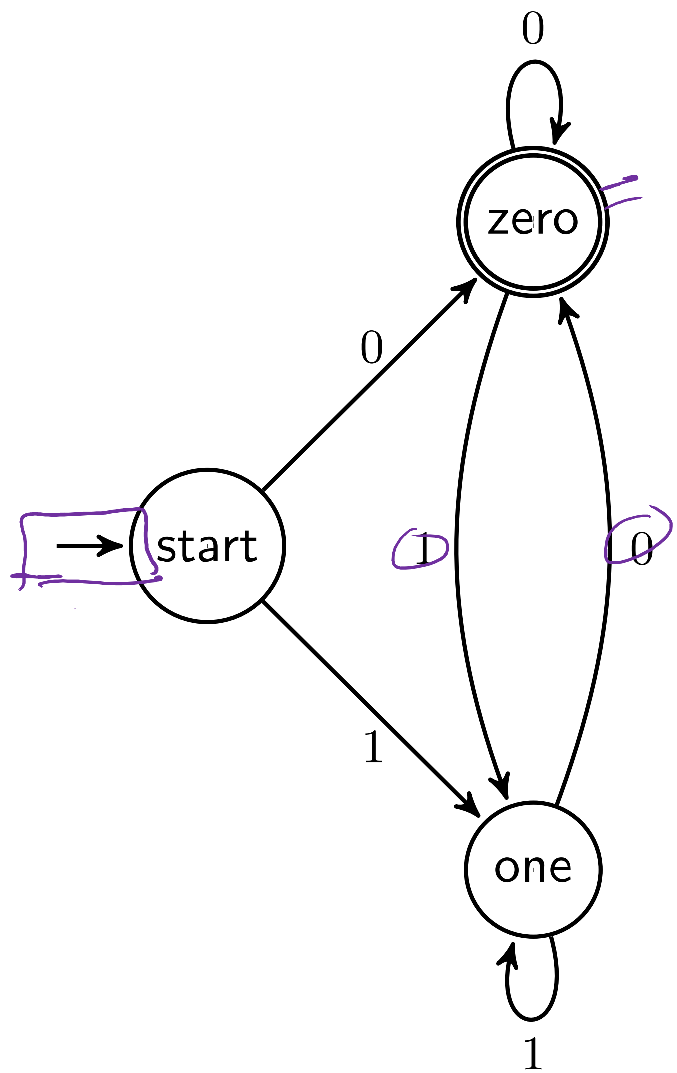
Student_Name	GPA
Knuth	4.00
Karp	3.98

Database Operations: Natural Join

Student ⋈ Takes

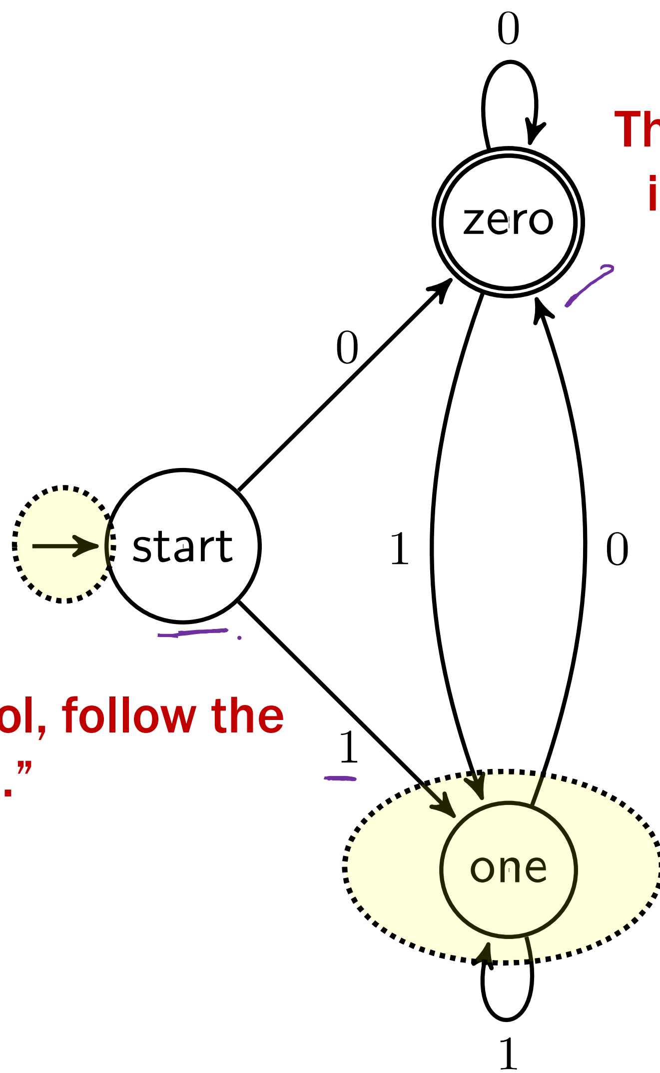
Student_Name	ID_Number	Office	GPA	Course
Knuth	328012098	022	4.00	CSE311
Knuth	328012098	022	4.00	CSE351
Von Neuman	481080220	555	3.78	CSE311
Russell	238082388	022	3.85	CSE312
Russell	238082388	022	3.85	CSE344
Russell	238082388	022	3.85	CSE351
Newton	1727017	333	3.61	CSE312
Karp	348882811	022	3.98	CSE311
Karp	348882811	022	3.98	CSE312
Karp	348882811	022	3.98	CSE344
Karp	348882811	022	3.98	CSE351
Bernoulli	2921938	022	3.21	CSE351

Selecting strings using labeled graphs as “machines”



Finite State Machines

"Start here"

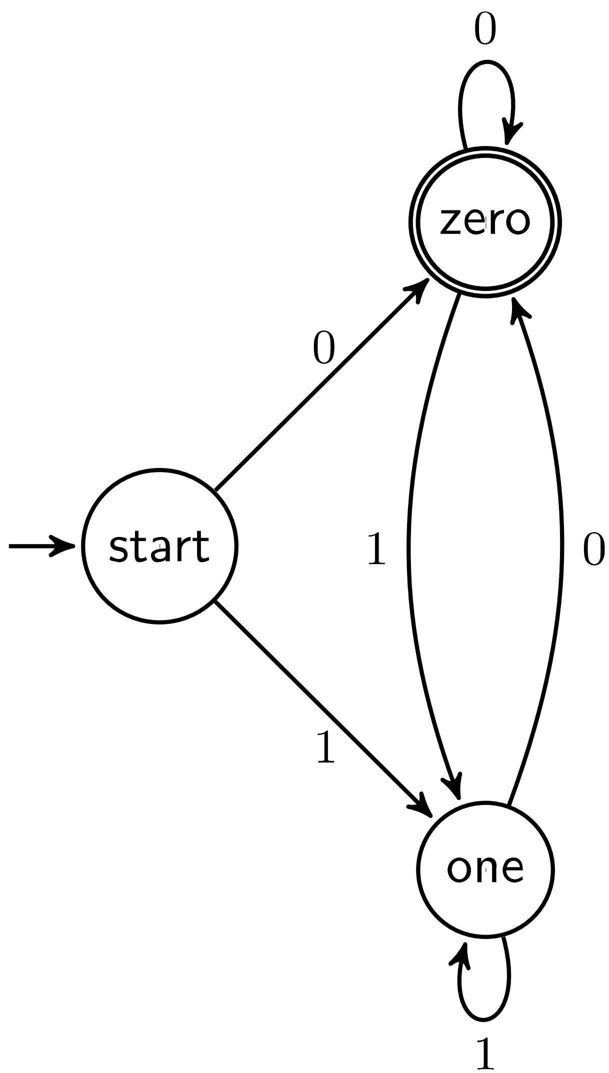


The "double circle" means "the input is good if it ends here"

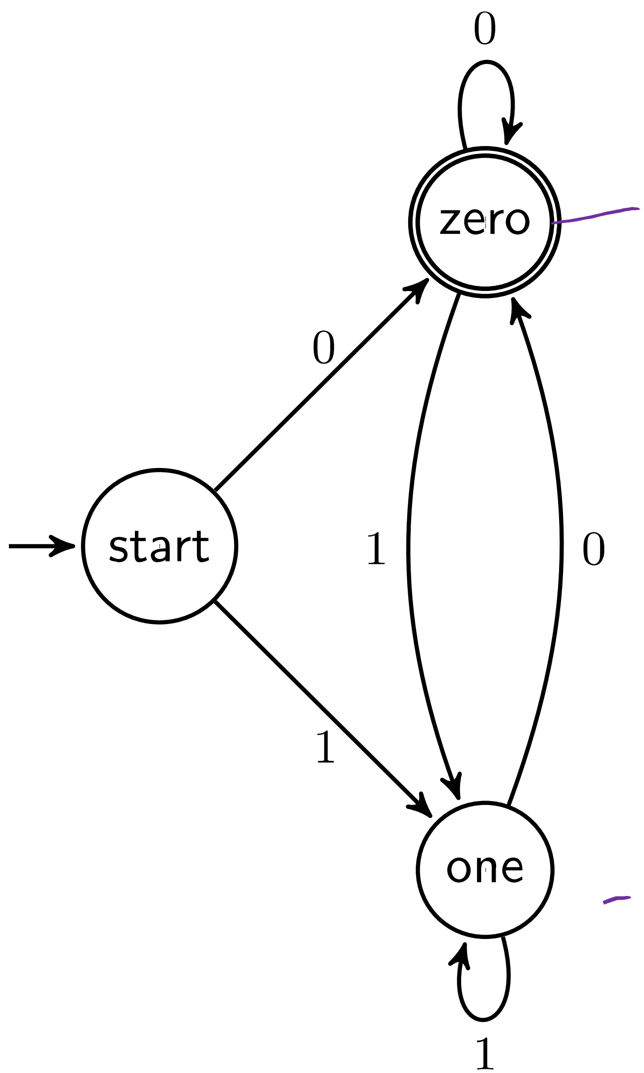
"If I get this symbol, follow the arrow..."

The circles are called "states"
We're only in a single state at any point in time...

Which strings does this machine say are OK?



Which strings does this machine say are OK?

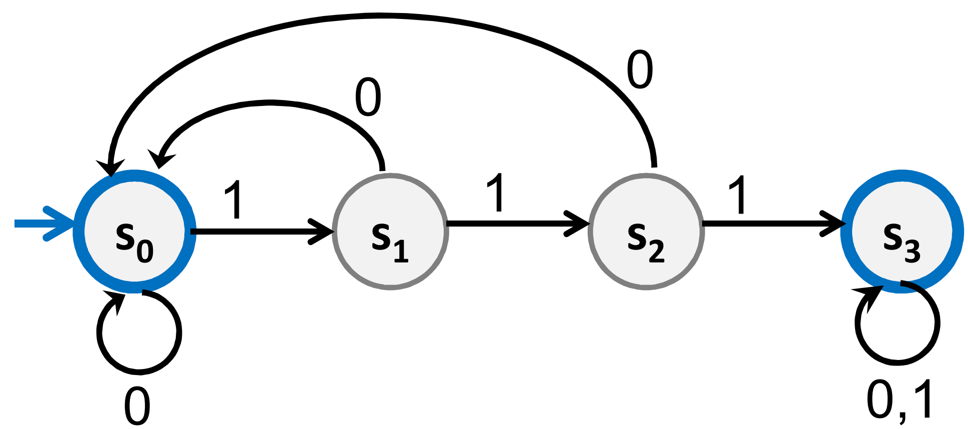


The set of all binary strings that end in 0

Finite State Machines

- States
- Transitions on input symbols
- Start state and final states
- The "language recognized" by the machine is the set of strings that reach a final state from the start

Old State	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_3
s_3	s_3	s_3

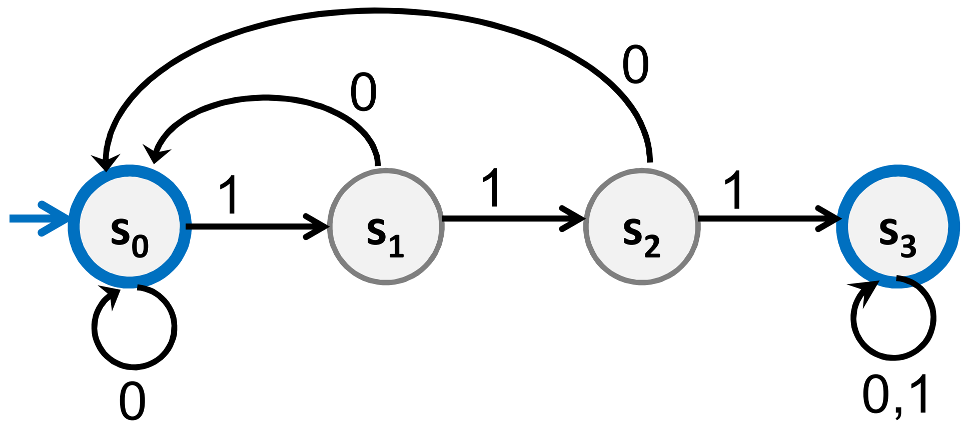


Finite State Machines

- **Each machine designed for strings over some fixed alphabet Σ.**

- **Must have a transition defined from each state for *every* symbol in Σ.**

Old State	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_3
s_3	s_3	s_3

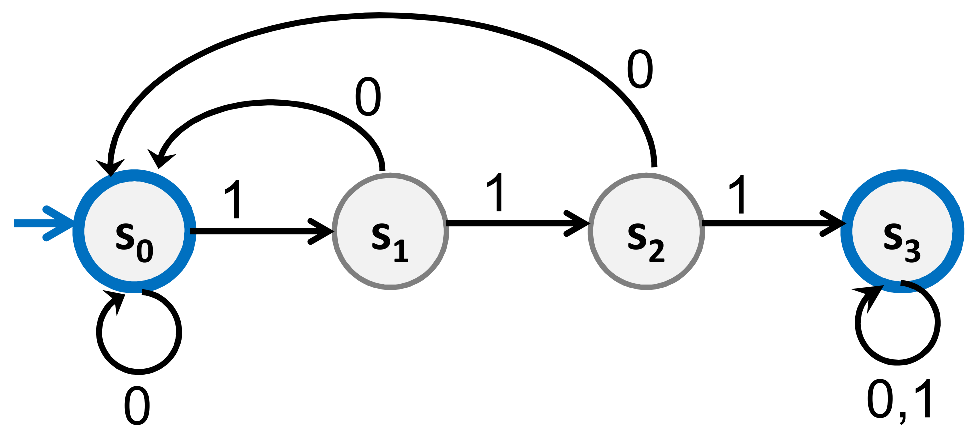


What language does this machine recognize?

s_3 : contains "111"

s_0 : empty or ends w/ zero

Old State	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_3
s_3	s_3	s_3

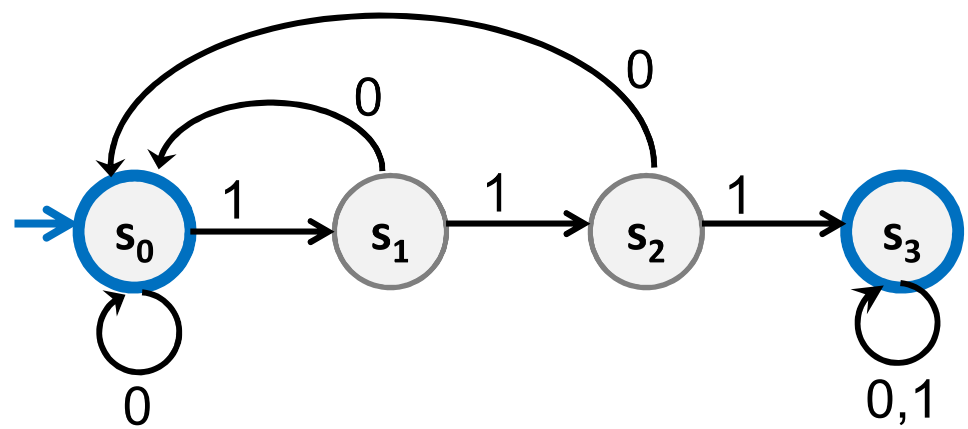


10

What language does this machine recognize?

The set of all binary strings that contain 111 (s_3)
OR don't end in a 1 (s_0)

Old State	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_3
s_3	s_3	s_3



Applications of FSMs (a.k.a. Finite Automata)

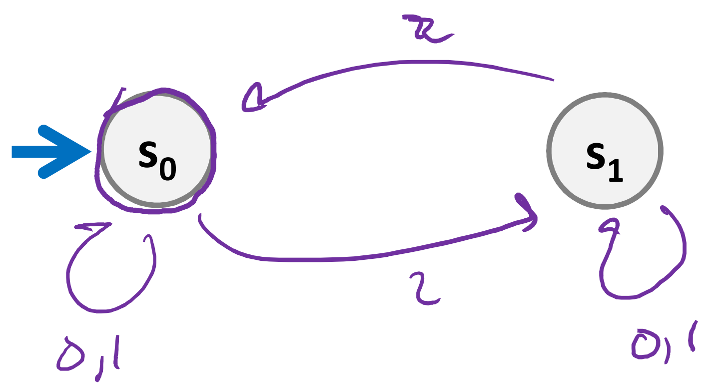
- Implementation of regular expression matching in programs like `grep`
- Control structures for sequential logic in digital circuits
- Algorithms for communication and cache-coherence protocols
  - Each agent runs its own FSM
- Design specifications for reactive systems
  - Components are communicating FSMs

Applications of FSMs (a.k.a. Finite Automata)

- Formal verification of systems
  - Is an unsafe state reachable?
- Computer games
  - FSMs provide worlds to explore
- Minimization algorithms for FSMs can be extended to more general models used in
  - Text prediction
  - Speech recognition

Strings over {0, 1, 2}

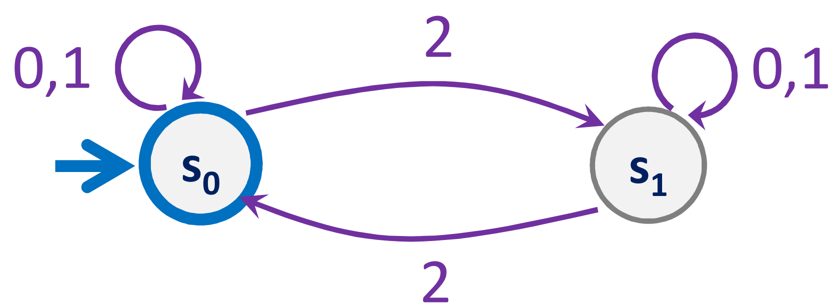
M_1: Strings with an even number of 2's



s_i when $\#2 \equiv i \pmod 2$

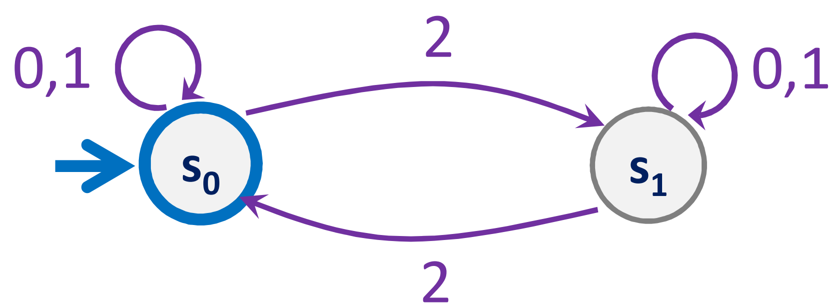
Strings over {0, 1, 2}

M_1: Strings with an even number of 2's

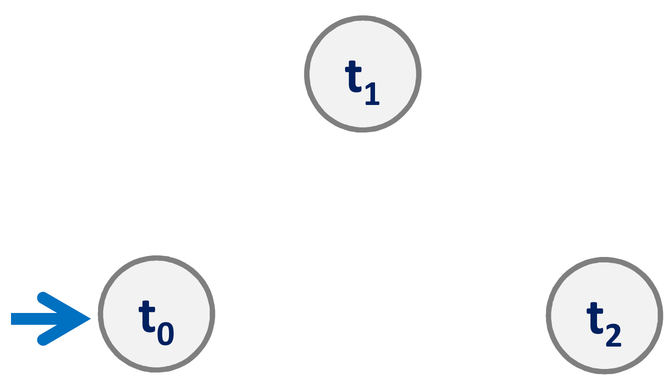


Strings over {0, 1, 2}

M_1: Strings with an even number of 2's

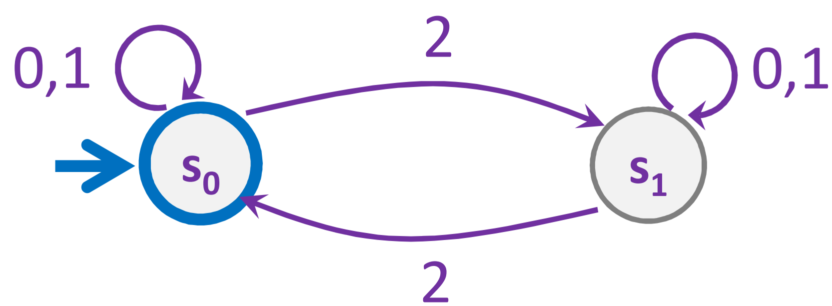


M_2: Strings where the sum of digits mod 3 is 0

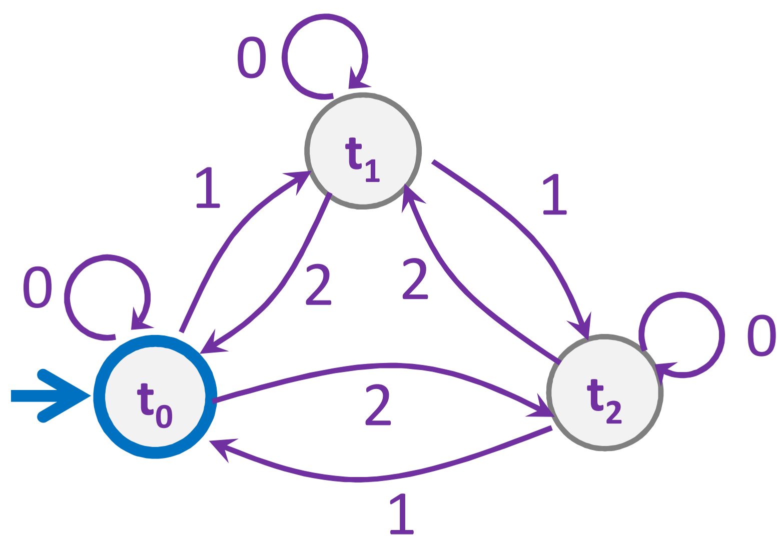


Strings over {0, 1, 2}

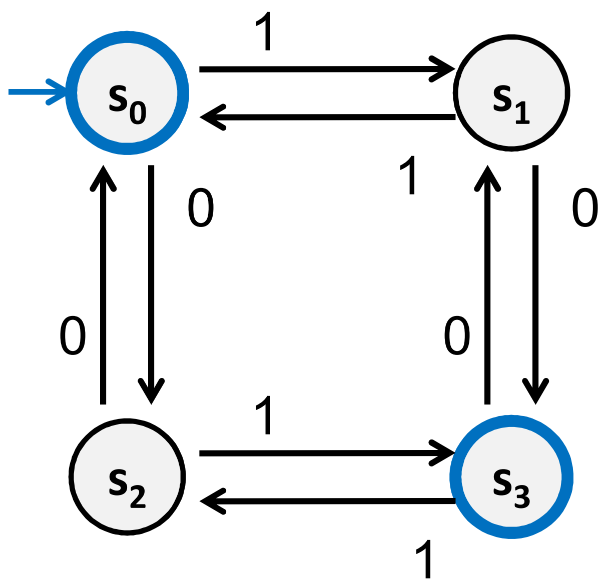
M_1: Strings with an even number of 2's



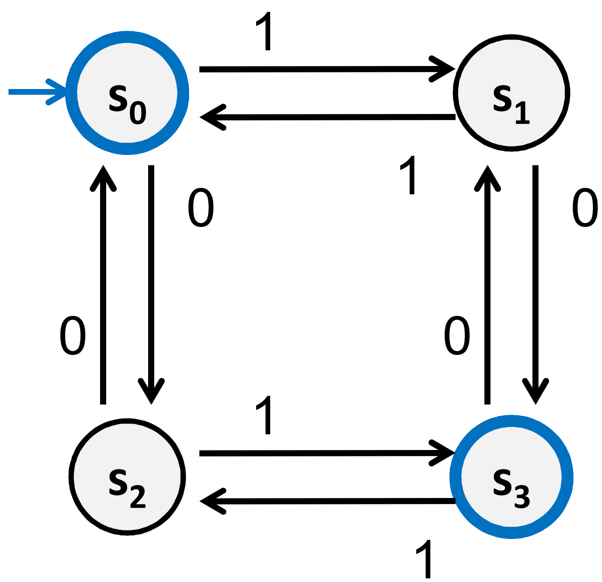
M_2: Strings where the sum of digits mod 3 is 0



What language does this machine recognize?

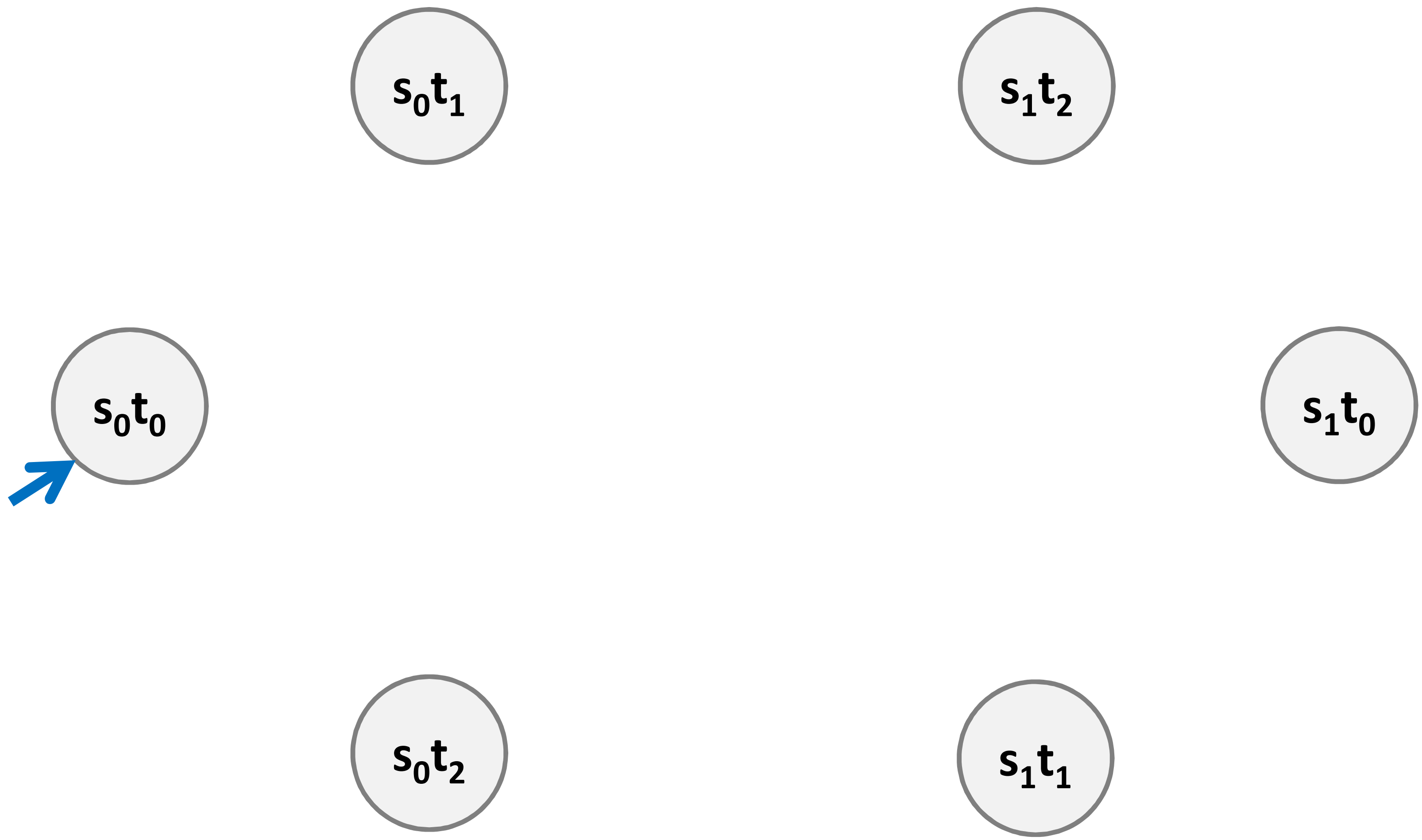


What language does this machine recognize?



The set of all binary strings with # of 1's ≡ # of 0's (mod 2) (both are even or both are odd).

Strings over {0,1,2} w/ even number of 2's and mod 3 sum 0



Strings over {0,1,2} w/ even number of 2's and mod 3 sum 0