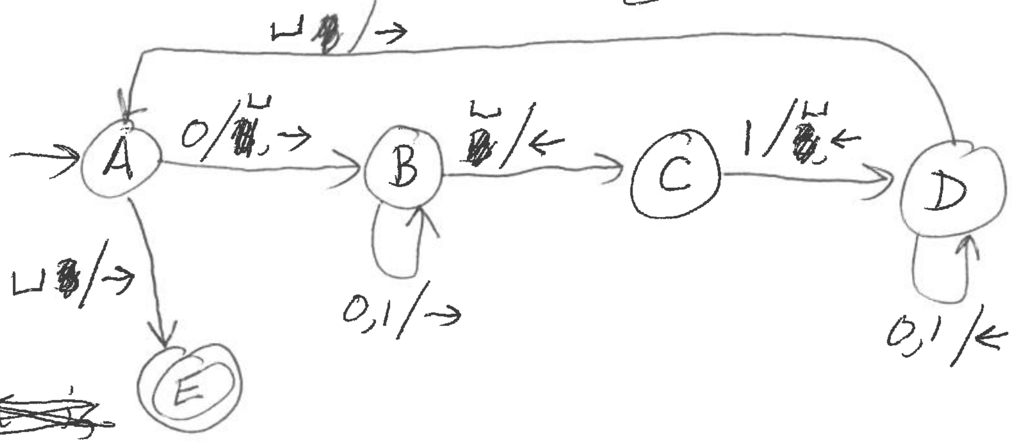
CSCE 355
4/14/2025

TM computations (lang recognition, functions) ①
Church-Turing thesis

TM computation example: $\{0^n 1^n : n \geq 0\}$




Blank: ⊔

~~Input is~~

Input is ~~110~~ 0011

Computation on input 0011:

A0011 ⊢ B011 ⊢ 0B11 ⊢ 01B1 ⊢ 011B⊔

⊢ 01C1 ⊢ 0D1⊔ ⊢ D01 ⊢ D⊔01

optional

⊢ A01 ⊢ ⊔B1 ⊢ 1B⊔ ⊢ C1 ~~A⊔~~

optional

⊢ D⊔ ⊢ A⊔ ⊢ E⊔ (end/halt) accept!

no transition defined

(because E is an accept state)

Computation on 001:

A001 ⊢ ⊔B01 ⊢ ⊔0B1 ⊢ ⊔01B⊔ ⊢ ⊔0C1⊔

⊢ ⊔D0⊔⊔ ⊢ D⊔0⊔⊔ ⊢ ⊔A0⊔ ⊢ ⊔⊔B⊔ ⊢ ⊔C⊔

halt & reject

Given any TM $M$ and input $w$ over $M$'s input alphabet, this defines a unique computation (deterministic): every ID has at most one successor ID.

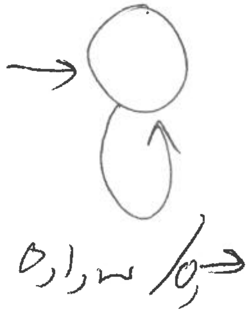
$$\underbrace{q_0 w}_{ID_0} \vdash ID_1 \vdash ID_2 \vdash \cdots \vdash ID_k \vdash$$

Def: We say that $M$ halts on input $w$ if there is some $n \geq 0$ such that $ID_n$ has no successor. Then $M$ halts in $n$ steps.

~~M accepts~~

If $M$ halts in $n$ steps on $w$, then let $q$ be the state in $ID_n$ (the last ID). $M$ accepts $w$ iff $q$ is an accepting state, and rejects $w$ otherwise.

If there is no such $n$, that is $ID_{k+1}$ exists for all $k \geq 0$, then this is an infinite computation and we say that $M$ loops on input $w$.




loops on any input

tape alphabet is $\{0,1,␣\}$

**Def:** Let $M$ be a TM with input alphabet $\Sigma'$. The language recognized by $M$ (write $L(M)$) is

$$L(M) := \{w \in \Sigma'^* : M \text{ accepts } w\}.$$

So: $M$ accepts all strings in $L(M)$ and no others. (If $w \notin L(M)$, then $M$ either rejects $w$ or loops on $w$.)

**Def:** A TM $M$ is total, or a decider if $M$ halts on all inputs. In this case we say that $M$ decides $L(M)$.

In the prev example, the TM is a decider and so decides $\{0^n 1^n : n \geq 0\}$.

**Def:** Let $L$ be a language. $L$ is Turing-recognizable (T-rec) if $L = L(M)$ for some TM $M$.

$L$ is decidable if $L = L(M)$ for some decider $M$.

Church–Turing thesis: Turing machines capture our intuitive notion of computation (algorithm).

TMs can ~~do~~ simulate any reasonable machine architecture. Need

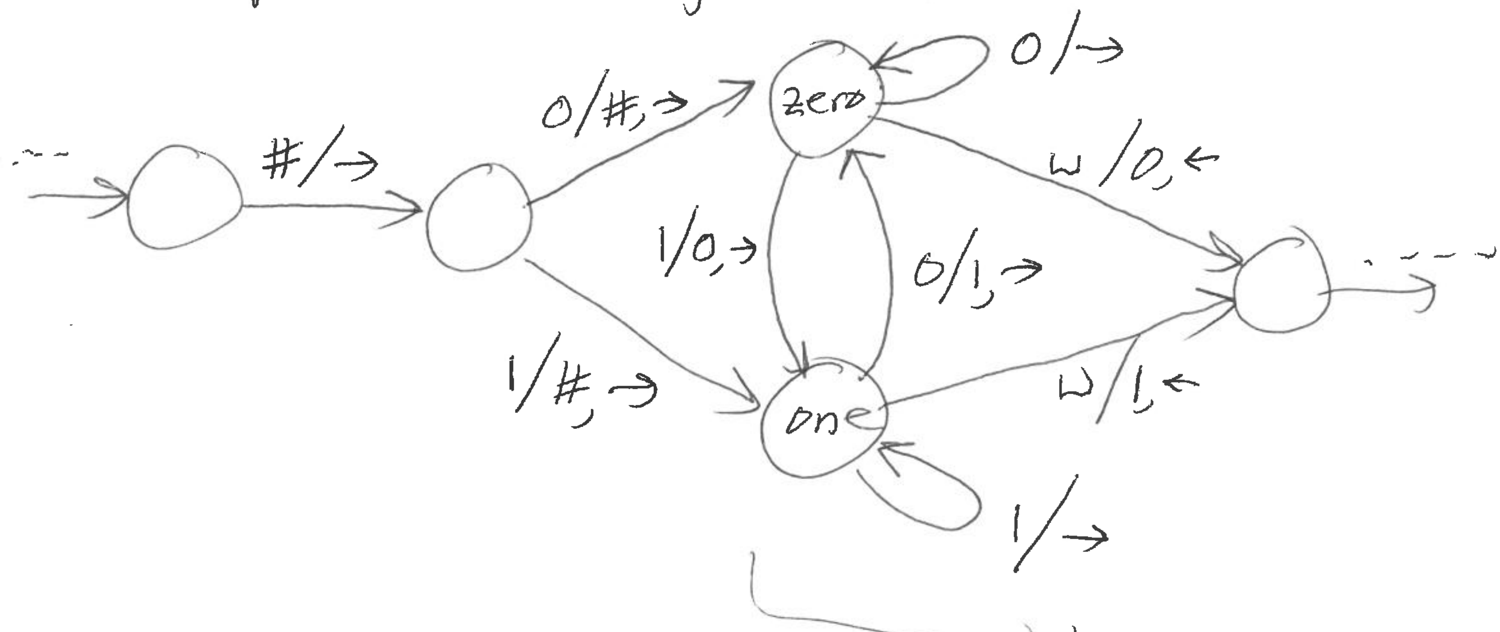
- memory movement
- copy
- arith ops $+, \cdot, -, /$
- comparisons / logical ops

Memory movement

~~$w \in \Sigma$~~

$\#w \rightsquigarrow \#\#w$ $\qquad w \in \{0,1\}$

bump $w$ one cell rightwards:

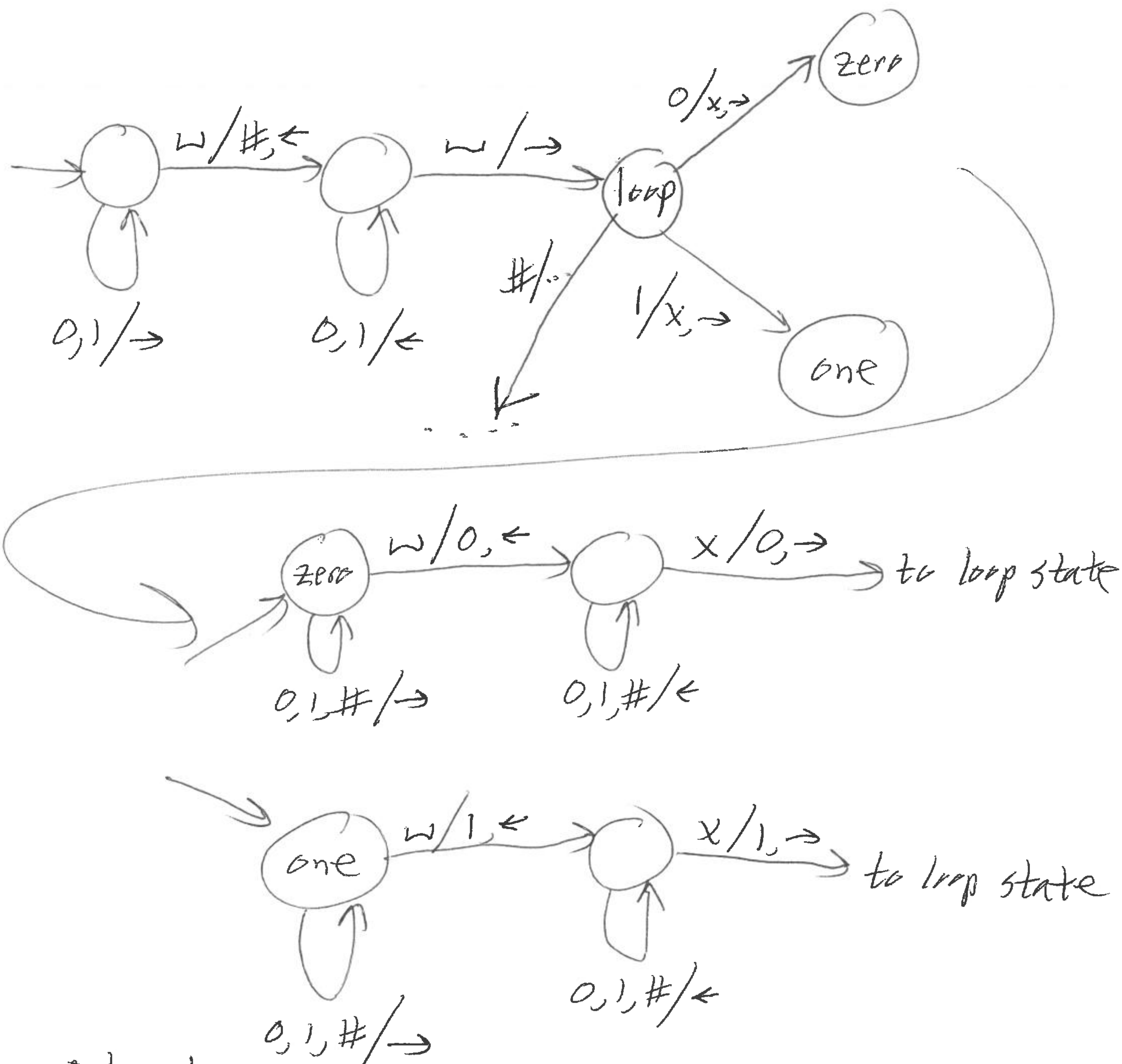



state ~~sa~~ reflects last symbol erased

Ex: Bump w one cell leftwards (similar)

Copying:

$w \rightsquigarrow w\#w \quad (w \in \{0,1\}^*)$




0 1

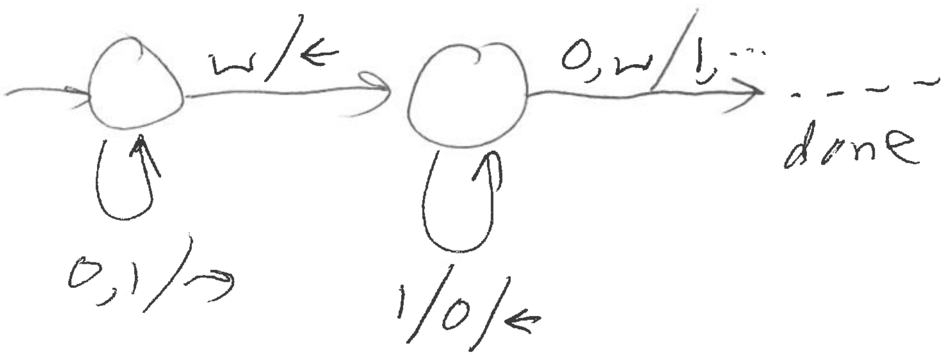
0 # 0 1

## Increment/decrement

$w$ in binary $\rightsquigarrow$ $w+1$ in binary (increment)

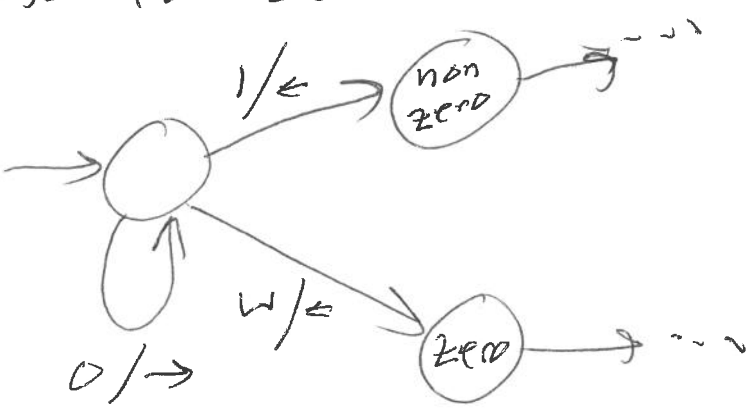
Ex: $w = 1011 \longrightarrow 1100$

Increment




decrement is similar, but don't decrement 0 (zero)

test for zero




Addition (informal)

$w, x \in \{0,1\}^*$

$w\#x \rightsquigarrow y$ ($y = w+x$ numerically) add:

```
while w ≠ 0
    dec w
    inc x
```

subtraction:

```
while w ≠ 0 and x ≠ 0
    dec w
    dec x
```

truncated subtraction (don't go < 0)