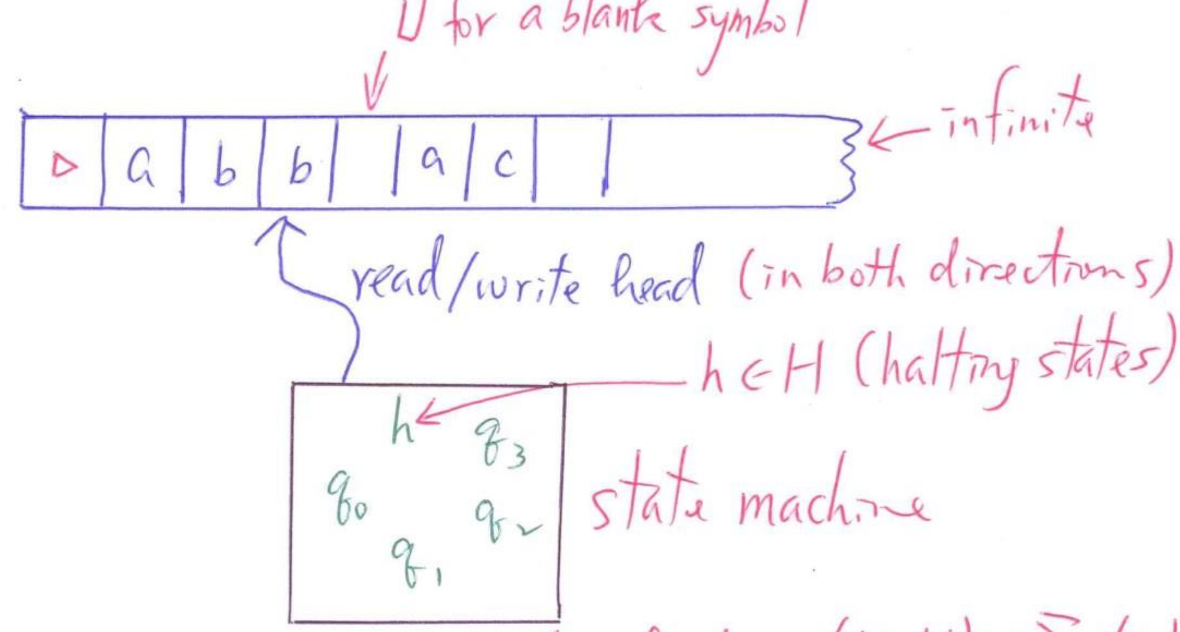
Turing Machines

Kun-Mao Chao



transition function $(K-H) \times \Sigma$ to $K \times (\Sigma \cup \{\leftarrow, \rightarrow\})$

$M = (K, \Sigma, \delta, s, H)$

initial state (s); halting states (H)

K: a finite set of states;

Σ: alphabet (tape symbols: a, b, c, ▷, ⊔)

$\rightarrow$ no
$\leftarrow$ no

✡ for all $q \in K-H$, if $\delta(q, \triangleright) = (p, b)$, then $b = \rightarrow$

never erased; leftmost barrier

✡ for all $q \in K-H$ and $a \in \Sigma$, if $\delta(q, a) = (p, b)$, then $b \neq \triangleright$.

M never writes a ▷.

Once the machine reaches a halting state, it stops.

Two distinguished halting states:
- y "yes" (accept)
- n "no" (reject)

☆ $L = \{a^n b^n c^n : n \geq 0\}$ Kun-Mao Chao

Notation

$\overset{\circlearrowleft d}{R}$: The machine moves its head right one square, then if that square contains a d, it moves its head one square further to the right. (Continue this way if a d is found in the right square.)

$R \xrightarrow{a} dR$: The machine moves its head right one square, then if that square contains an a, it writes a d and moves its head one square further to the right.

$L_{\sqcup}$: The machine finds the first blank square to the left of the currently scanned square.

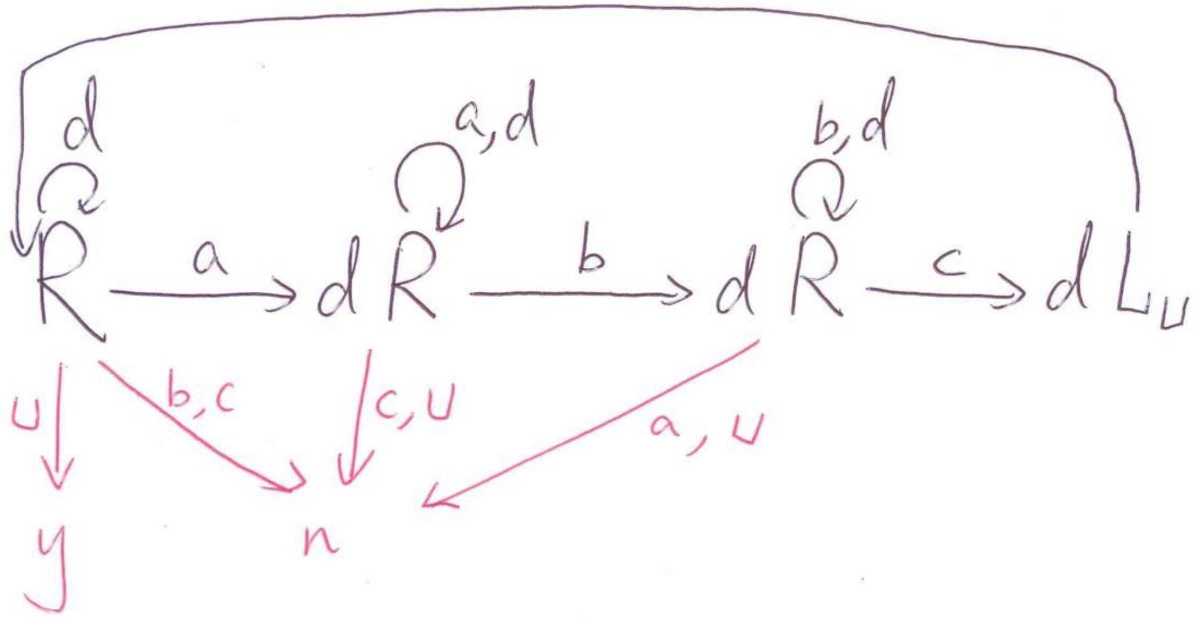
$(>L \circlearrowleft \overline{\sqcup})$

↑

keep moving leftward until a blank is found or a leftend ($\triangleright$) is found.

Kun-Mao Chan

$L = \{a^n b^n c^n : n \geq 0\}$



blank symbols

▷	a	a	b	b	c	c	⊔ ...

d a b b c c ⊔

d a d b c c ⊔

d a d b d c ⊔

$\longleftarrow L_{\sqcup}$

d d d b d c ⊔

d d d d d c ⊔

d d d d d d ⊔

$\longleftarrow L_{\sqcup}$

$\xrightarrow{R^{\neq d}} \xrightarrow{\sqcup}$ y accept.

Kun-Mao Chao

Ex.

$\triangleright$	a	a	b	c	c	$\sqcup$	$\sqcup$	...

d a b c c $\sqcup$

d a d c c $\sqcup$

d a d d c $\sqcup$

$\longleftarrow L_{\sqcup}$

d d d d c $\sqcup$

n reject.

Ex.

$\triangleright$	a	b	$\sqcup$	$\sqcup$	...

d b $\sqcup$

d d $\sqcup$

n reject

Ex.

$\triangleright$	b	$\sqcup$	$\sqcup$	...

n reject

The Halting Problem

Kun-Mao Chao

Suppose that halts(P, X)

always determines whether the program P would halt

on input X. (It returns "yes" if it does halt; otherwise "no".)

```
diagonal(X)
a:  if halts(X, X) then goto a   /* loop forever */  never halt
                    else halt
```

Does diagonal(diagonal) halt?

⇒ if halts(diagonal, diagonal) then loop forever
else halt

We show that the halting problem is undecidable by the following argument.

If halts(diagonal, diagonal) returns "yes", that means diagonal(diagonal) halts. But then diagonal(diagonal) will loop forever.

If halts(diagonal, diagonal) returns "no", that means diagonal(diagonal) does not halt. But then diagonal(diagonal) will halt.

A contradiction. The program halts(P, X) does not exist.