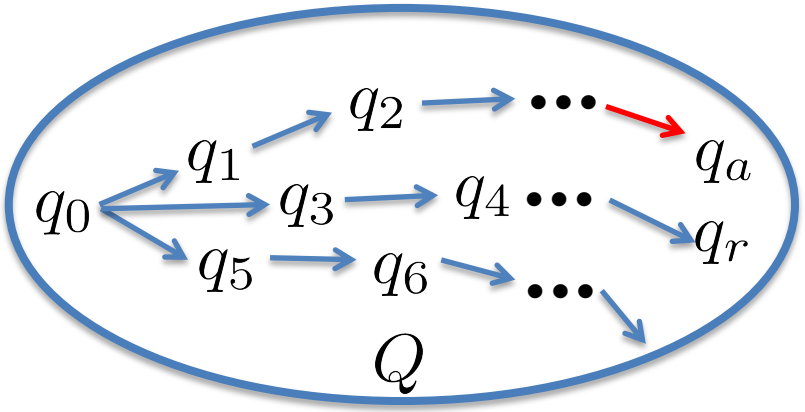
Accepted/recognized strings

Let $\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be a TM

A string $s = \sigma_1 \cdots \sigma_k (\sigma_i \in \Sigma)$ is accepted/recognized by $\mathbf{M}$ if there exist configurations $C_1, \ldots, C_k$ s.t.

$C_1 = q_0 s$ (start),
C_1 derives C_2,...,C_{k-1} derives C_k
$C_k = u q_a v$, where $uv = s$ (accept)

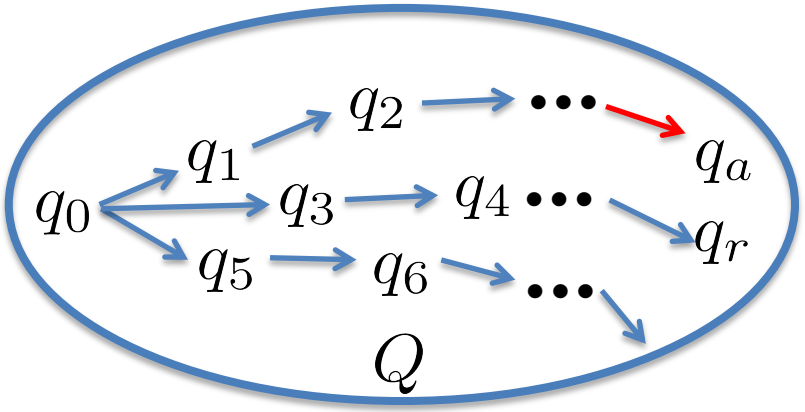


Accepted/recognized languages

Let $\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be a TM

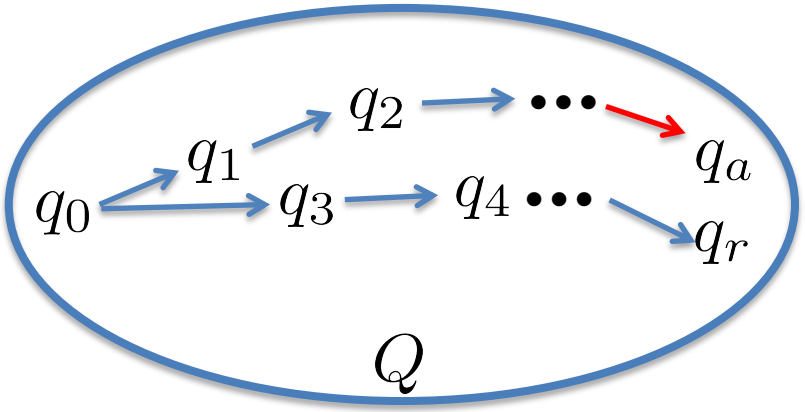
The language accepted/recognized by $\mathbf{M}$ is:

$$L_a(\mathbf{M}) = \{s \in \Sigma^* \mid s \text{ recognized by } \mathbf{M}\}$$



Decider

Defn. Let $\mathbf{D} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be a TM.
If $\mathbf{D}$ halts on all input strings $s \in \Sigma^*$, then call $\mathbf{D}$ a *decider*.

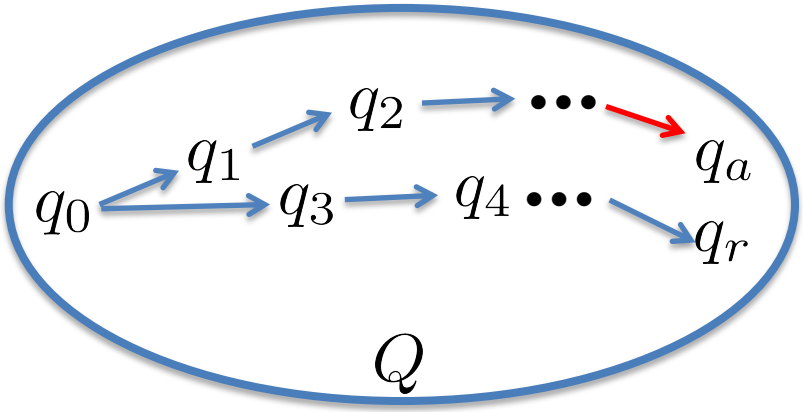


Decider

Defn. Let $\mathbf{D} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be a TM.
If $\mathbf{D}$ halts on all input strings $s \in \Sigma^*$, then call $\mathbf{D}$ a *decider*.

The language accepted/recognized by a decider $\mathbf{D}$ is:

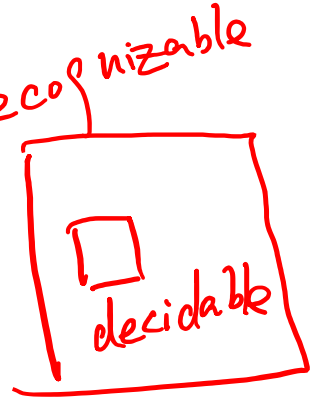
$$L_a(\mathbf{D}) = \{s \in \Sigma^* \mid s \text{ recognized by } \mathbf{D}\}$$



Turing-recognizable/decidable

Defn. A language $L \subseteq \Sigma^*$ is *Turing-recognizable* if there exists a TM $\mathbf{M}$ s.t. $L_a(\mathbf{M}) = L$.

(also called *recursively enumerable*)



Defn. A language $L \subseteq \Sigma^*$ is *Turing-decidable* if there exists a decider $\mathbf{D}$ s.t. $L_a(\mathbf{D}) = L$.

(also called *recursive*)

Turing-recognizable/decidable

Defn. A language $L \subseteq \Sigma^*$ is *Turing-recognizable* if there exists a TM $\mathbf{M}$ s.t. $L_a(\mathbf{M}) = L$.

(also called *recursively enumerable*)

Defn. A language $L \subseteq \Sigma^*$ is *Turing-decidable* if there exists a decider $\mathbf{D}$ s.t. $L_a(\mathbf{D}) = L$.

(also called *recursive*)

Note: a decidable L is also recognizable, but not vice versa

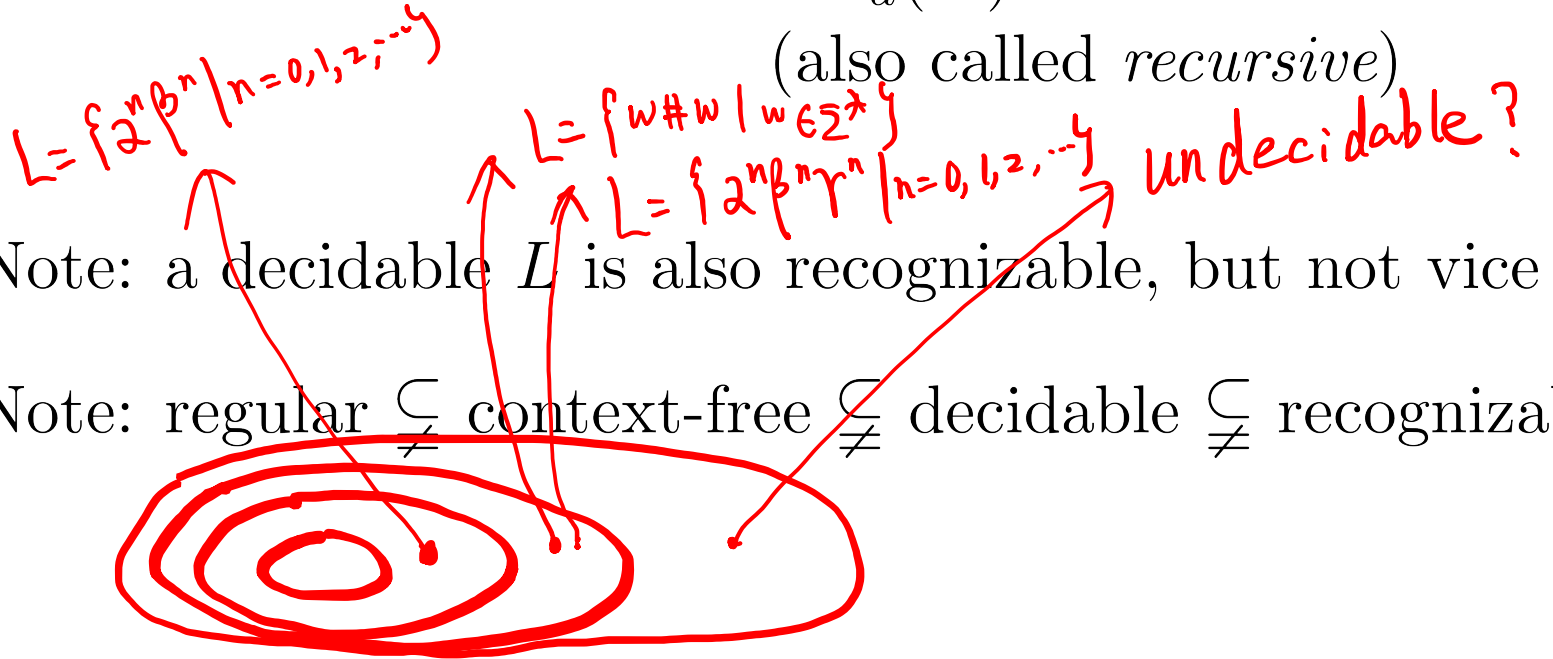
Turing-recognizable/decidable

Defn. A language $L \subseteq \Sigma^*$ is *Turing-recognizable* if there exists a TM $\mathbf{M}$ s.t. $L_a(\mathbf{M}) = L$.

(also called *recursively enumerable*)

Defn. A language $L \subseteq \Sigma^*$ is *Turing-decidable* if there exists a decider $\mathbf{D}$ s.t. $L_a(\mathbf{D}) = L$.

(also called *recursive*)

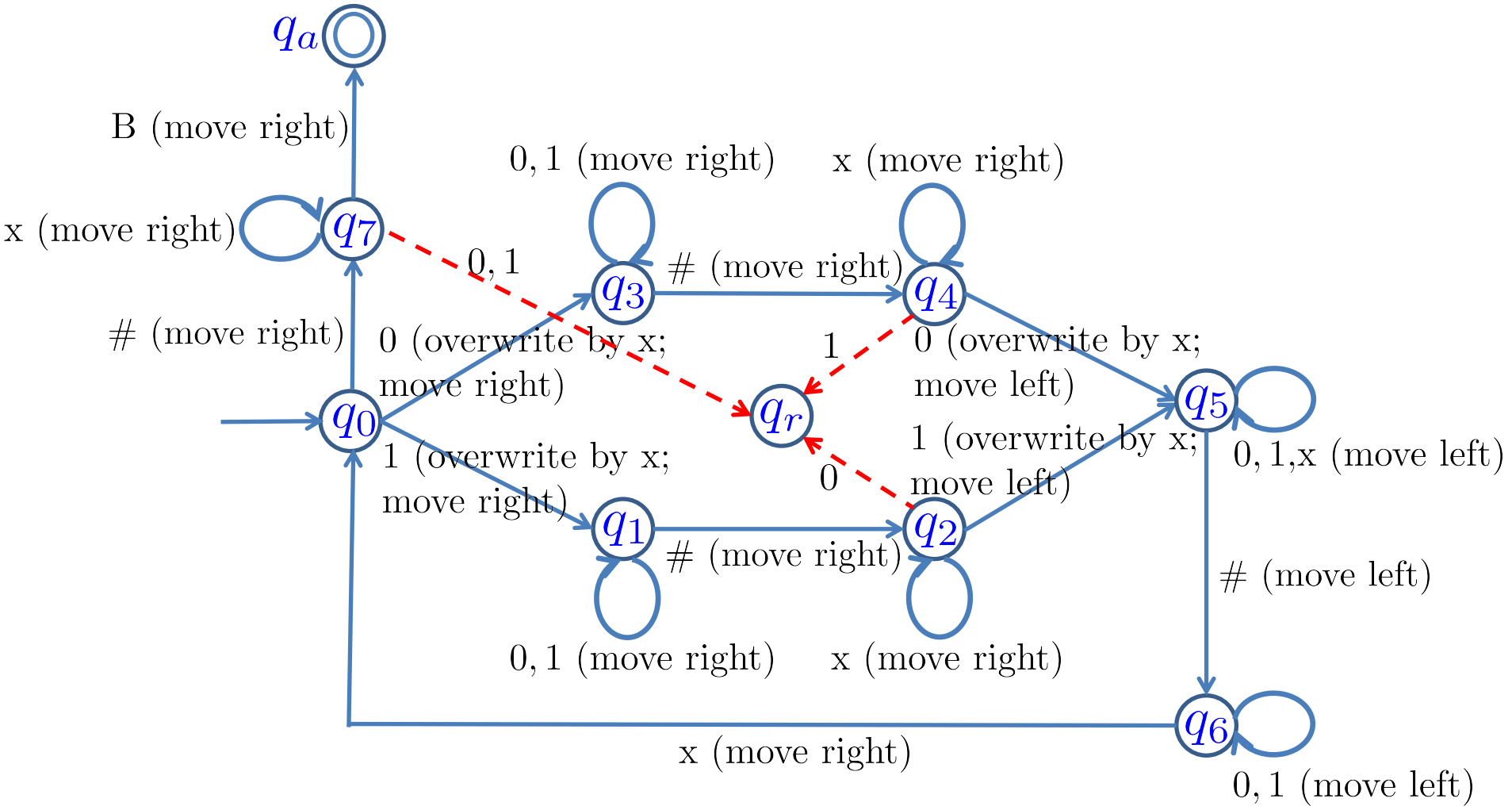


Note: a decidable L is also recognizable, but not vice versa

Note: regular $\subsetneq$ context-free $\subsetneq$ decidable $\subsetneq$ recognizable

Example

Turing-decidable language: $L = \{w\#w \mid w \in \{0,1\}^*\}$



Example

Turing-decidable language: $L = \{w\#w \mid w \in \{0,1\}^*\}$

L is also Turing-recognizable

Example

Turing-decidable language: $L = \{w\#w \mid w \in \{0,1\}^*\}$
L is also Turing-recognizable

Turing-recognizable language (but not decidable):
we will see one when studying "undecidability"

Variants of Turing machine

Since 1936 the original TM, there were many variants proposed in hope of more generality.

However, all have been proved to have just the same expression power as the original TM model
(i.e. recognizing the same class of languages)

Variants of Turing machine

Consider a (deterministic) Turing machine (TM) $\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\}$: transition function

i.e. tape head is allowed to stop

Does this new feature add more expression power?

Variants of Turing machine

Consider a (deterministic) Turing machine (TM)
$\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\}$: transition function

i.e. tape head is allowed to stop

Does this new feature add more expression power?

No: one can always convert it back to the original TM by replacing S by first L then R;
i.e. the new TM can be *simulated* by the orignial TM

Variants of Turing machine

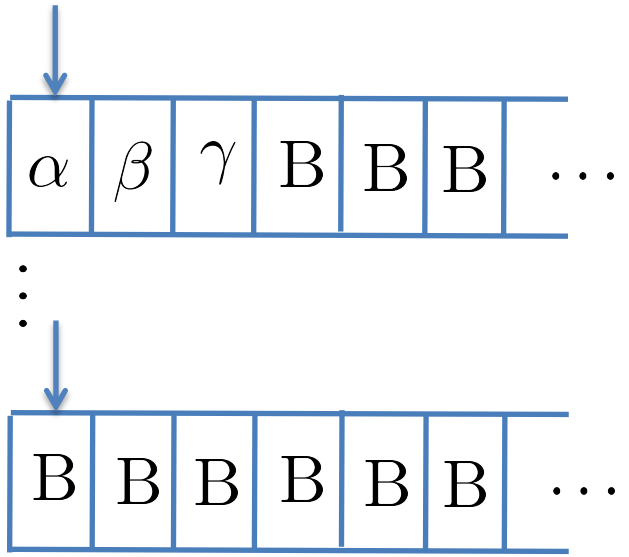
Consider a (deterministic) Turing machine (TM) $\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where

$\delta(q_i, r_1, \cdots, r_k) = (q_j, r_1', \cdots, r_k', \underbrace{L, R \cdots, L, L}_{k})$

$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\mathrm{L}, \mathrm{R}\}^k$: transition function

$(\underbrace{\Gamma \times \cdots \times \Gamma}_{k})$

i.e. there are $k \geq 1$ tapes (simultaneous read/write/move on k tapes)



Does this new feature add more expression power?

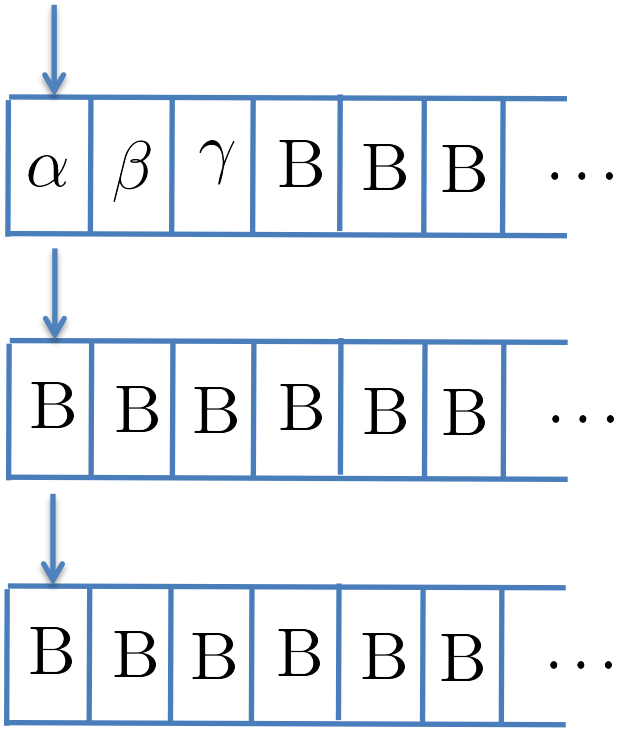
Variants of Turing machine

Fact: Every multi-tape TM has an equivalent single-tape TM.

Variants of Turing machine

Fact: Every multi-tape TM has an equivalent single-tape TM.

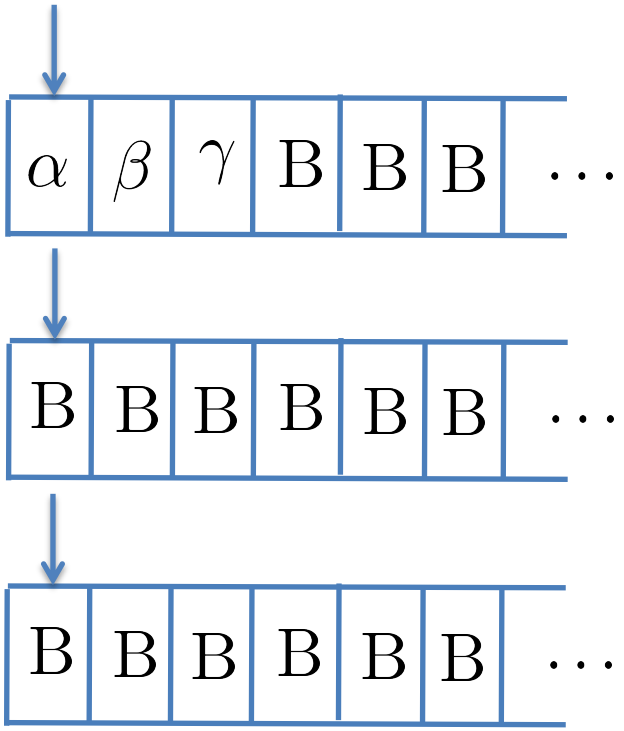
Consider a 3-tape TM **M**.



Variants of Turing machine

Fact: Every multi-tape TM has an equivalent single-tape TM.

Consider a 3-tape TM **M**.

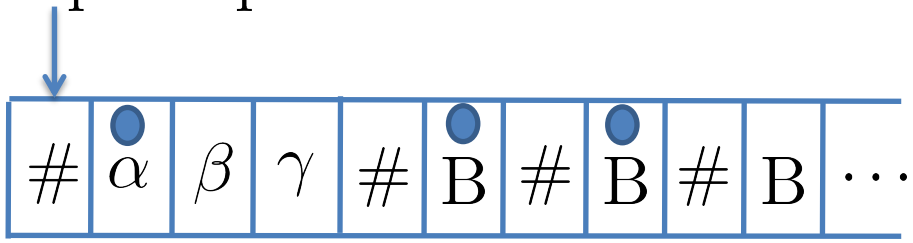


Construct a 1-tape TM **S** to simulate **M**.

The tape of **S** is initialized as follows:

#: separate contents in different tapes

dot: virtual heads to track multiple tape heads

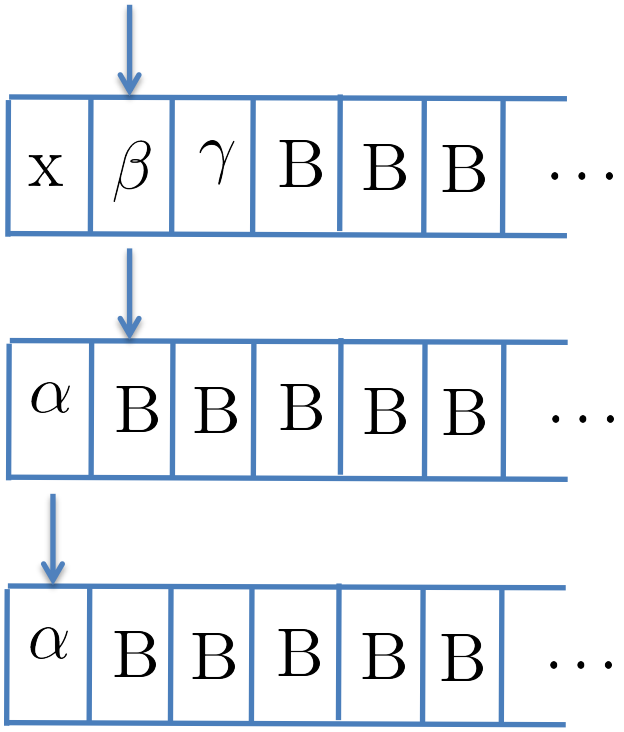


Variants of Turing machine

Fact: Every multi-tape TM has an equivalent single-tape TM.

Consider a 3-tape TM **M**
and one transition:

$$\delta(q_0, \alpha, \mathrm{B}, \mathrm{B}) = (q_1, \mathrm{x}, \alpha, \alpha, \mathrm{R}, \mathrm{R}, \mathrm{L})$$

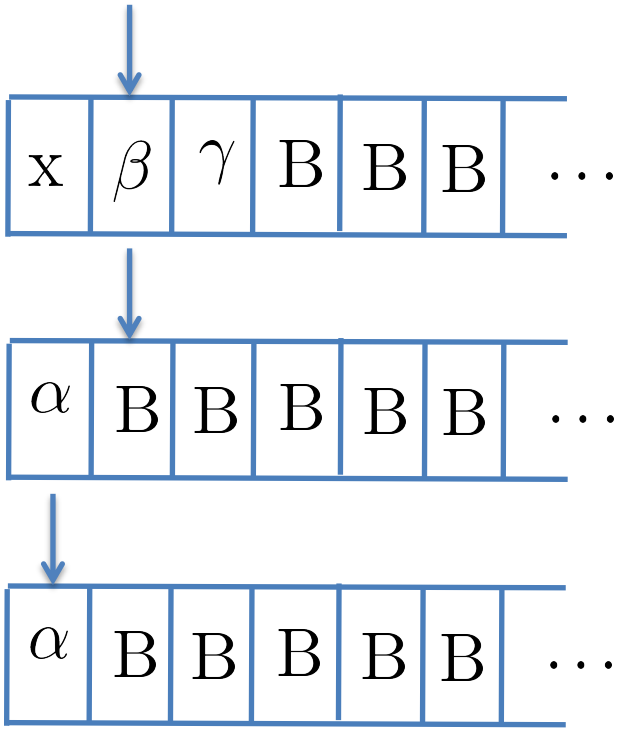


Variants of Turing machine

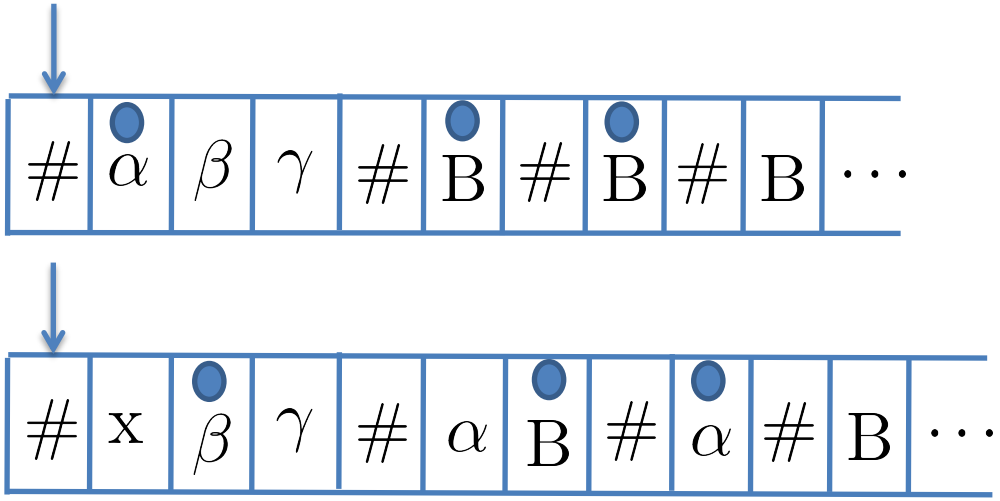
Fact: Every multi-tape TM has an equivalent single-tape TM.

Consider a 3-tape TM **M**
and one transition:

$$\delta(q_0, \alpha, \mathrm{B}, \mathrm{B}) = (q_1, \mathrm{x}, \alpha, \alpha, \mathrm{R}, \mathrm{R}, \mathrm{L})$$



Simulate this transition by **S** as follows:



Variants of Turing machine

Consider a nondeterministic Turing machine (TM)
$\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where

$\delta : Q \times \Gamma \to Pwr(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\})$: transition function

i.e. allowing more than one choice of (state,write,move)

Does this new feature add more expression power?

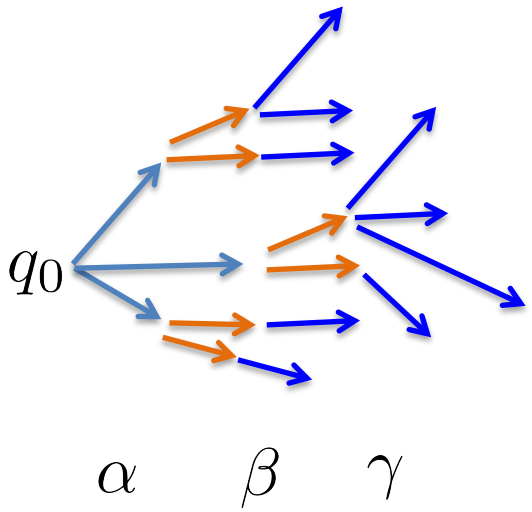
Variants of Turing machine

Fact: Every nondeterministic TM has an equivalent det. TM.

Variants of Turing machine

Fact: Every nondeterministic TM has an equivalent det. TM.

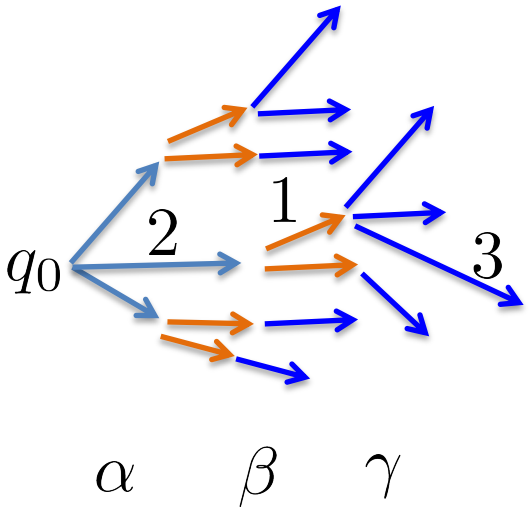
Consider a nondeterministic TM **N** and an input string $\alpha\beta\gamma$



Variants of Turing machine

Fact: Every nondeterministic TM has an equivalent det. TM.

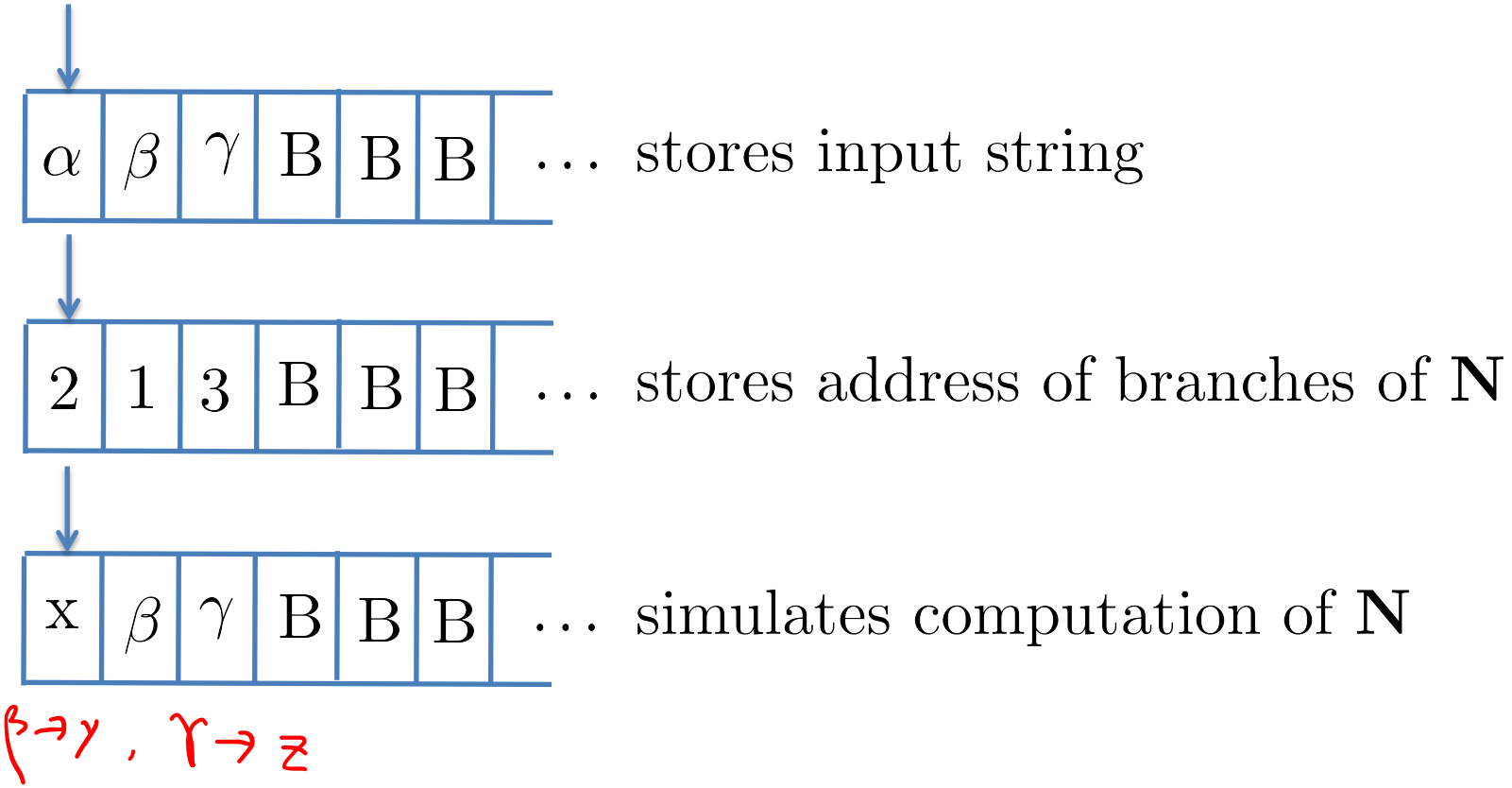
Consider a nondeterministic TM **N** and an input string $\alpha\beta\gamma$



Variants of Turing machine

Fact: Every nondeterministic TM has an equivalent det. TM.

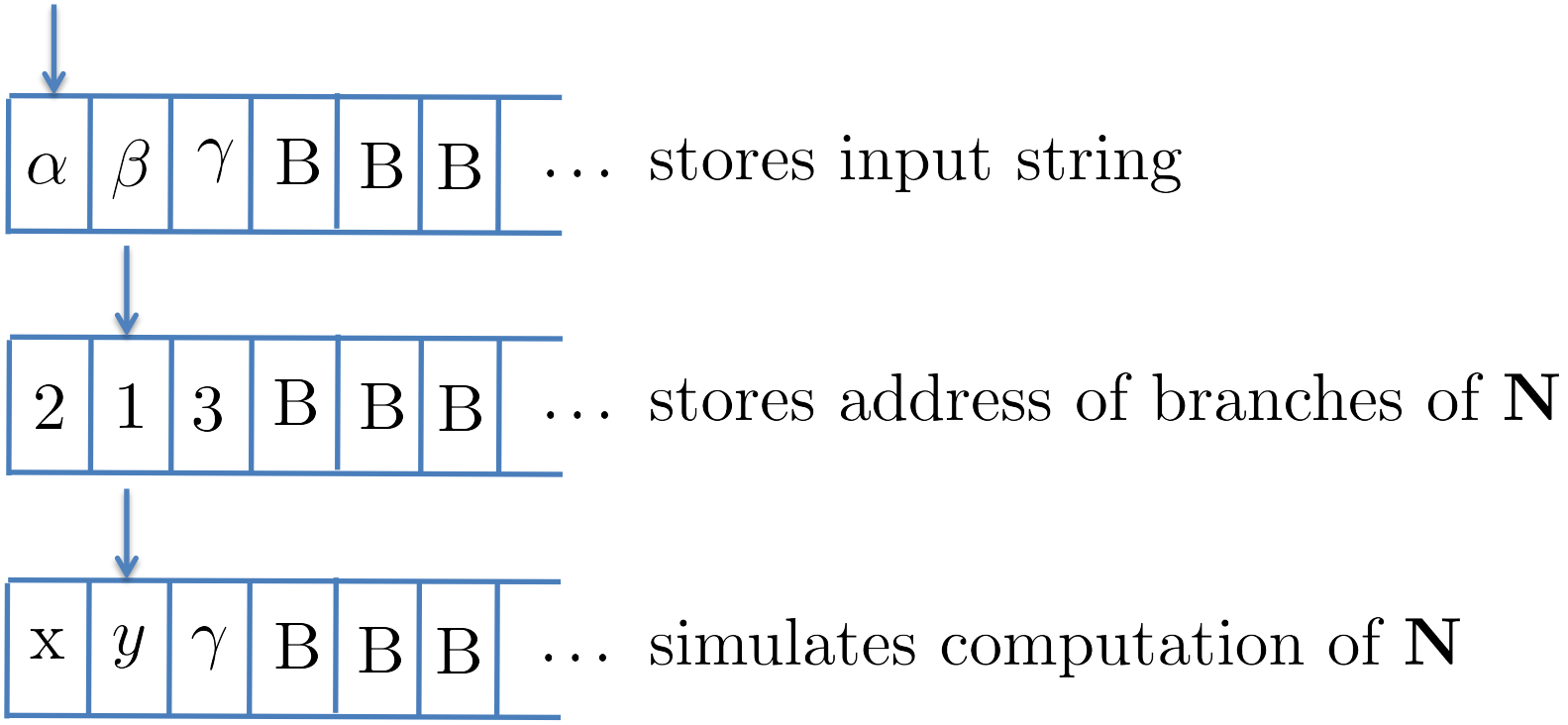
Consider a nondeterministic TM **N** and an input string $\alpha\beta\gamma$
It can be simulated by a deterministic TM **M** using 3 tapes:



Variants of Turing machine

Fact: Every nondeterministic TM has an equivalent det. TM.

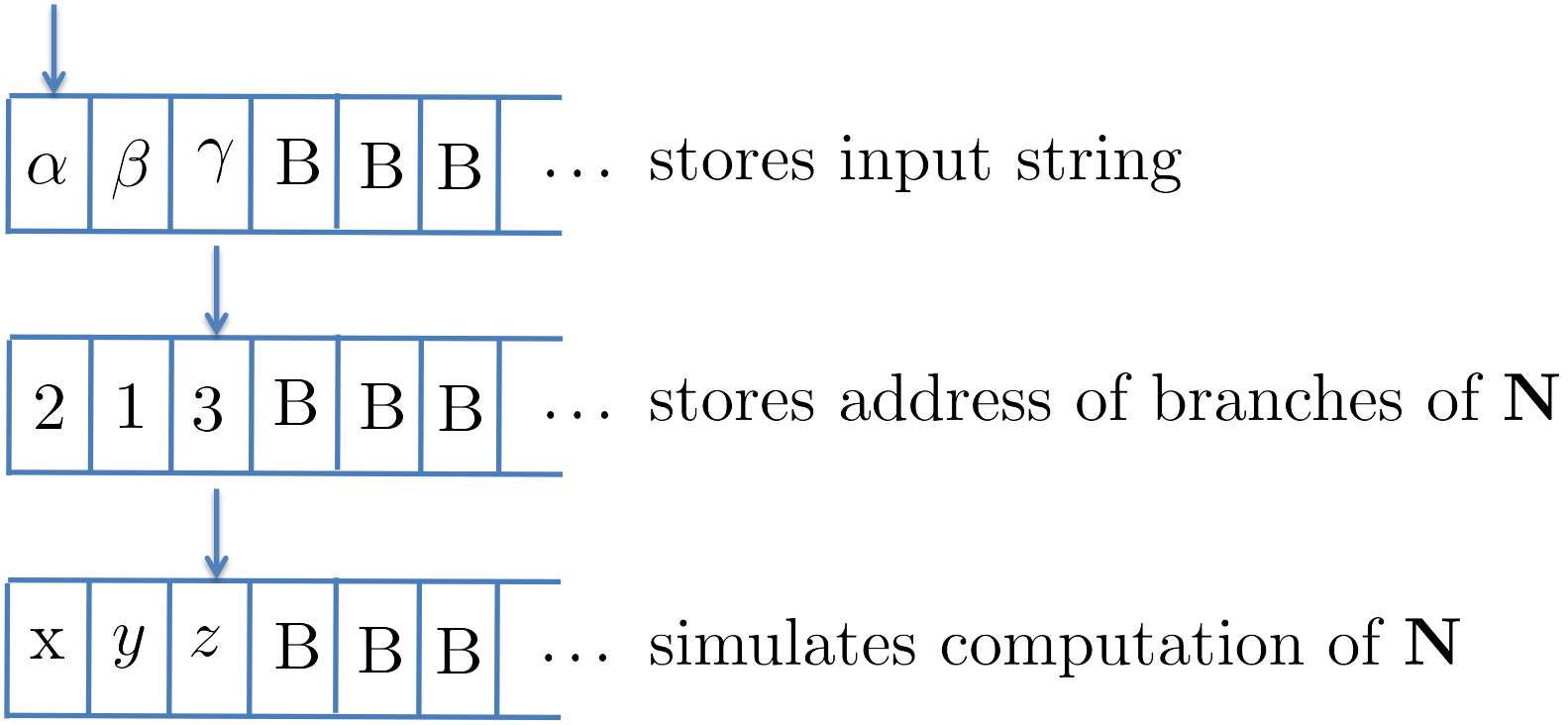
Consider a nondeterministic TM **N** and an input string $\alpha\beta\gamma$
It can be simulated by a deterministic TM **M** using 3 tapes:



Variants of Turing machine

Fact: Every nondeterministic TM has an equivalent det. TM.

Consider a nondeterministic TM **N** and an input string $\alpha\beta\gamma$
It can be simulated by a deterministic TM **M** using 3 tapes:



Recap

Original TM (single-tape, deterministic)

$\Leftrightarrow$ stoppable TM

$\Leftrightarrow$ multi-tape TM

$\Leftrightarrow$ nondeterministic TM

(analogy: different programming languages)

Recap

Original TM (single-tape, deterministic)

$\Leftrightarrow$ stoppable TM

$\Leftrightarrow$ multi-tape TM

$\Leftrightarrow$ nondeterministic TM

(analogy: different programming languages)

All feature: unlimited memory, unrestricted access to tape(s)

All recognize the same class of languages: Turing-recognizable

If all halt on all input strings,
then all decide the same class of languages: Turing-decidable