Lecture 14: Turing Machine Variants

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Formal Definition: Turing Machine with Stop

A **Turing machine with stop** is a 7-tuple $(\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{q_{accept}}, \boldsymbol{q_{reject}})$ where

- $\boldsymbol{Q}$: set of states
- $\boldsymbol{\Sigma}$: input alphabet (**not containing** blank symbol ⊔)
- $\boldsymbol{\Gamma}$: tape alphabet (⊔ ∈ $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma} \in \boldsymbol{\Gamma}$ as well as other symbols)
- $\boldsymbol{\delta}: \boldsymbol{Q} \times \boldsymbol{\Gamma} \rightarrow \boldsymbol{Q} \times \boldsymbol{\Gamma} \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}$: transition function

$$q_i \xrightarrow{\mathbf{0} \rightarrow \mathbf{x}, \mathbf{S}} q_j$$

- Tape head is **not forced to move** left or right,
- Can **stay** at current position on tape

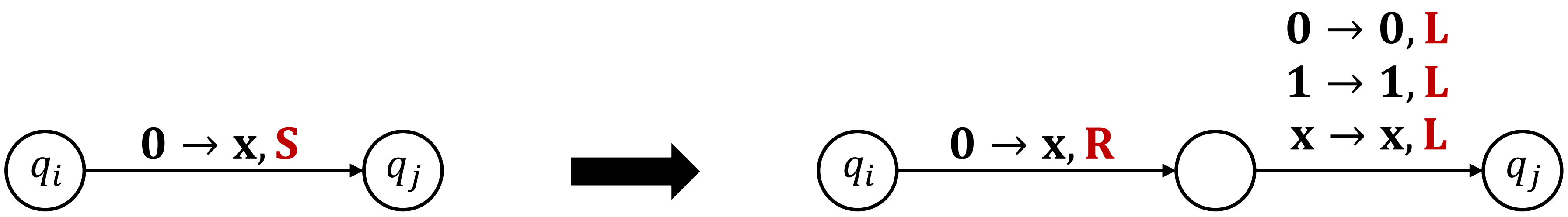
- $\boldsymbol{q_0} \in \boldsymbol{Q}$: single start state
- $\boldsymbol{q_{accept}} \in \boldsymbol{Q}$: single accept state
- $\boldsymbol{q_{reject}} \in \boldsymbol{Q}$: single reject state (with $\boldsymbol{q_{reject}} \neq \boldsymbol{q_{accept}}$)

Equivalence of TM with Stop

The computational power of **TMs** and **TMs with stop** are equivalent:

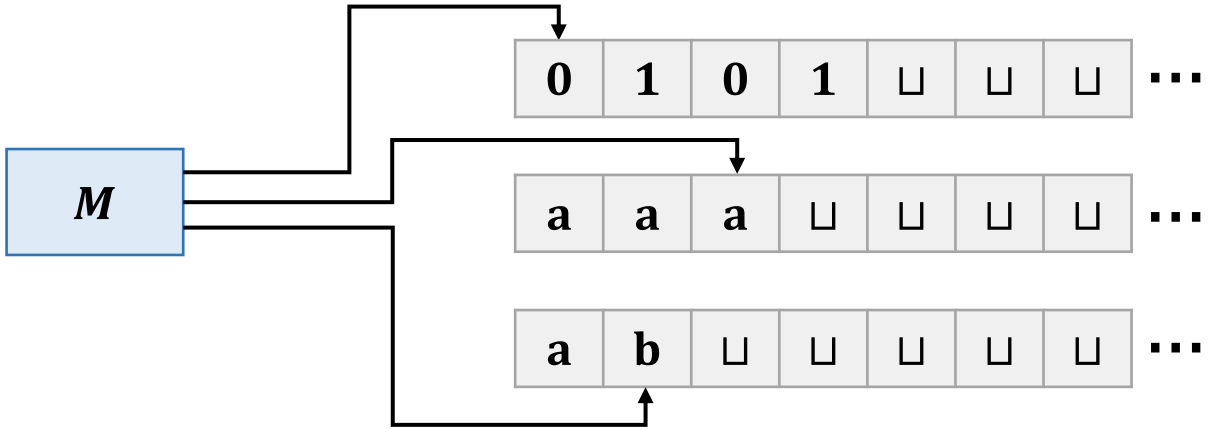
- Every **TM** is can be converted into an **equivalent TM with stop** by not changing anything
  - (TMs are a **special case** of TMs with stop where **S** is never used)

- Every **TM with stop** can be converted into an **equivalent TM** by replacing a stop transition with **two transitions** (move right, then move back)



No matter what we read after moving right, don't change tape then move back

Multitape Turing Machines



Similar to original TM but with:

- Multiple tapes (some finite k tapes)
- Each tape has **its own tape head** (which can also stop)
- Initial configuration:
  - Input w on **tape 1**, all other tapes blank
  - All tape heads at the beginning of respective tapes

Multitape Turing Machines

The **transition function** for a multitape Turing machine $\boldsymbol{M}$ with $\boldsymbol{k}$ tapes is defined as follows:

$$\boldsymbol{\delta}: \boldsymbol{Q} \times \boldsymbol{\Gamma}^{\boldsymbol{k}} \rightarrow \boldsymbol{Q} \times \boldsymbol{\Gamma}^{\boldsymbol{k}} \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^{\boldsymbol{k}}$$

A transition $\boldsymbol{\delta}(\boldsymbol{q_i}, \boldsymbol{a_1}, \dots, \boldsymbol{a_k}) = (\boldsymbol{q_j}, \boldsymbol{b_1}, \dots, \boldsymbol{b_k}, \mathbf{L}, \dots, \mathbf{R})$ means

- If $\boldsymbol{M}$ is in state $\boldsymbol{q_i}$ and tape heads 1 through $\boldsymbol{k}$ points at cells with content $\boldsymbol{a_1}, \dots, \boldsymbol{a_k}$ then
  - $\boldsymbol{M}$ changes to state $\boldsymbol{q_j}$
  - Replaces $\boldsymbol{a_1}, \dots, \boldsymbol{a_k}$ with $\boldsymbol{b_1}, \dots, \boldsymbol{b_k}$
  - Each tape head moves $\mathbf{L}$, $\mathbf{R}$, or $\mathbf{S}$ as specified

Formal Definition: Multitape Turing Machine

A **multitape Turing machine** with $\boldsymbol{k}$ **tapes** is a 7-tuple $(\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{q_{accept}}, \boldsymbol{q_{reject}})$ where

- $\boldsymbol{Q}$: set of states
- $\boldsymbol{\Sigma}$: input alphabet (**not containing** blank symbol ⊔)
- $\boldsymbol{\Gamma}$: tape alphabet (⊔ ∈ $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma}$ ∈ $\boldsymbol{\Gamma}$ as well as other symbols)
- $\boldsymbol{\delta}: \boldsymbol{Q} \times \boldsymbol{\Gamma}^{\boldsymbol{k}} \to \boldsymbol{Q} \times \boldsymbol{\Gamma}^{\boldsymbol{k}} \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^{\boldsymbol{k}}$: transition function
- $\boldsymbol{q_0} \in \boldsymbol{Q}$: single start state
- $\boldsymbol{q_{accept}} \in \boldsymbol{Q}$: single accept state
- $\boldsymbol{q_{reject}} \in \boldsymbol{Q}$: single reject state (with $\boldsymbol{q_{reject}} \neq \boldsymbol{q_{accept}}$)

Equivalence of Multitape Turing Machines

The computational power of **TMs** and **multitape TMs** are equivalent:

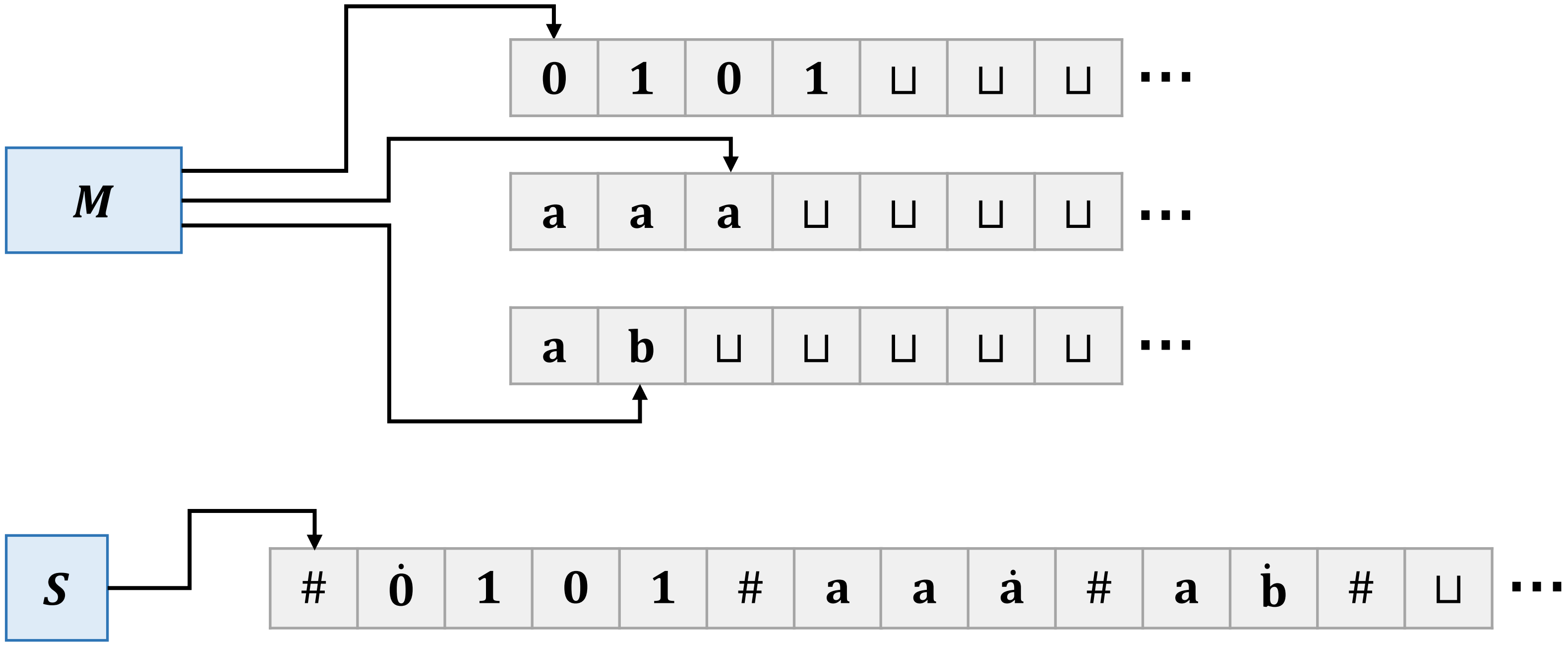
- Every **TM** is can be converted into an **equivalent multitape TM** by not changing anything
  - (TMs are a **special case** of multitape TMs with $\boldsymbol{k = 1}$)

- We will show that every **multitape TM** can be converted into an **equivalent TM**

Multitape TM to Single-tape TM

Idea of converting multitape TM $\boldsymbol{M}$ with $\boldsymbol{k}$ tapes to single-tape TM $\boldsymbol{S}$

- On single tape, use symbol # to split up tape into $\boldsymbol{k}$ tapes
- Use $\dot{}$ on a symbol to represent tape head location in each tape
  - E.g. For tape alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{a}, \mathbf{b}, \sqcup\}$, add tape symbols $\{\dot{\mathbf{0}}, \dot{\mathbf{1}}, \dot{\mathbf{a}}, \dot{\mathbf{b}}, \dot{\sqcup}\}$



Multitape TM to Single-tape TM

Computation on input $\boldsymbol{w = w_1 w_2 \dots w_n}$

- Prepare the tape of $\boldsymbol{S}$ to represent all $\boldsymbol{k}$ tapes of $\boldsymbol{M}$

$$\# \; \dot{w}_1 \; w_2 \dots w_n \; \# \; \dot{\sqcup} \; \# \; \dot{\sqcup} \; \# \dots \#$$

- Simulating a transition of $\boldsymbol{M}$
  - **Determine symbols** under “virtual tape heads” by scanning left to right
  - **Execute transition** via second scan (update cells and positions)
  - If a tape needs more room, shift tape content to the right to **add a** ⊔

Turing Machine Equivalence Corollary

A language L is **Turing-recognizable**

if and only if

some **single-tape Turing machine** recognizes it

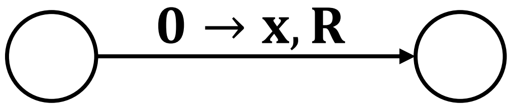
if and only if

some **multitape Turing machine** recognizes it

Deterministic Turing Machines

All the TM variants so far have been **deterministic TMs**:

- In a **state** and reading **tape symbol**, there is **exactly one transition** (write symbol, move tape head, and go to next state)
- For an input string, there is **one branch of computation**
- $\boldsymbol{\delta}: \boldsymbol{Q} \times \boldsymbol{\Gamma} \rightarrow \boldsymbol{Q} \times \boldsymbol{\Gamma} \times \{\mathbf{L}, \mathbf{R}\}$ in single-tape TM

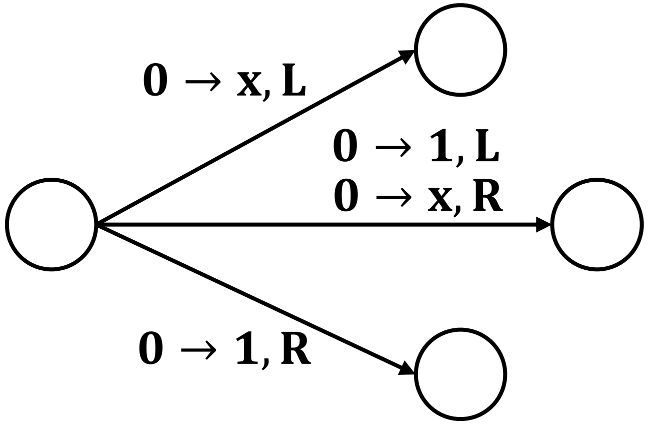


Nondeterministic Turing Machines (NTM)

A **nondeterminisic TM** (**NTM**) is defined just like the single-tape TM, but the transition function is defined as:

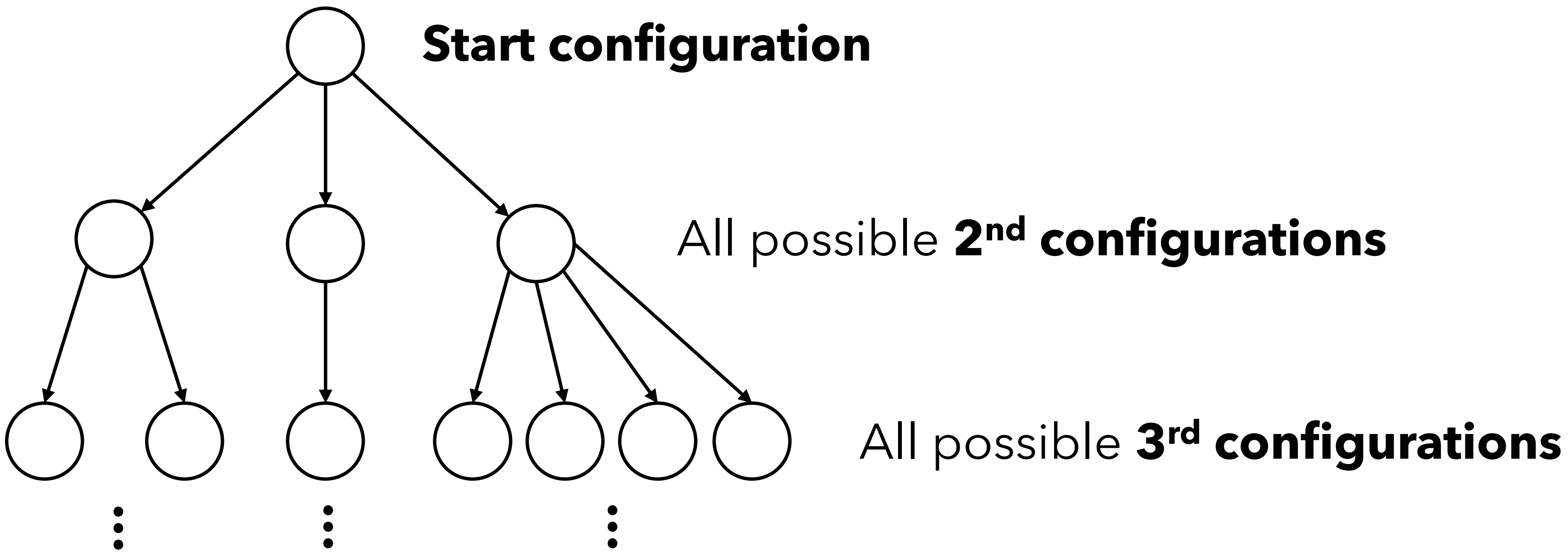
$$\boldsymbol{\delta}: \boldsymbol{Q} \times \boldsymbol{\Gamma} \rightarrow \boldsymbol{\mathcal{P}}(\boldsymbol{Q} \times \boldsymbol{\Gamma} \times \{\mathbf{L}, \mathbf{R}\})$$

- In a **state** and reading **tape symbol**, there can be **multiple transitions**
- Can go to different states, write different symbols, move tape head in different directions



A nondeterministic TM **accepts** if **any branch of computation** accepts

Computation on NTMs



Note: Branches may end in **accept**, **reject**, or branches can be infinite length since TM can enter **infinite loop**

Equivalence of Nondeterministic Turing Machines

The computational power of **deterministic TMs** and **nondeterministic TMs** with stop are equivalent:

- For each **deterministic TM**, there exists a **nondeterministic TM** which recognizes the same language
  - A deterministic TM is just a **special case** of an NTM

- For each **nondeterministic TM**, there exists a **deterministic TM** which recognizes the same language
  - We will show how to **simulate an NTM** on a deterministic TM

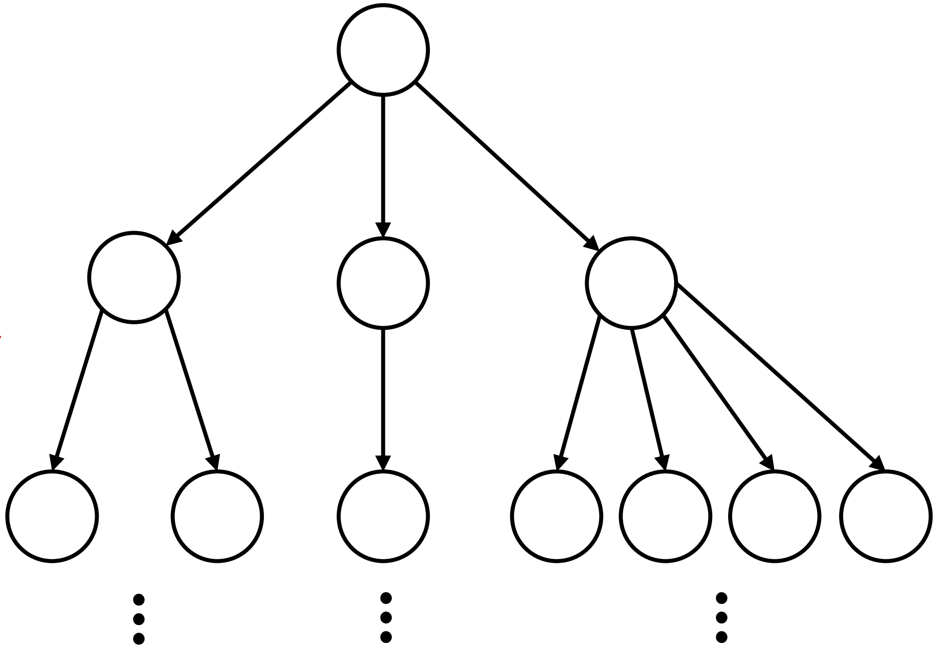
NTM to TM

For each **nondeterministic TM**, there exists a **deterministic TM** which recognizes the same language

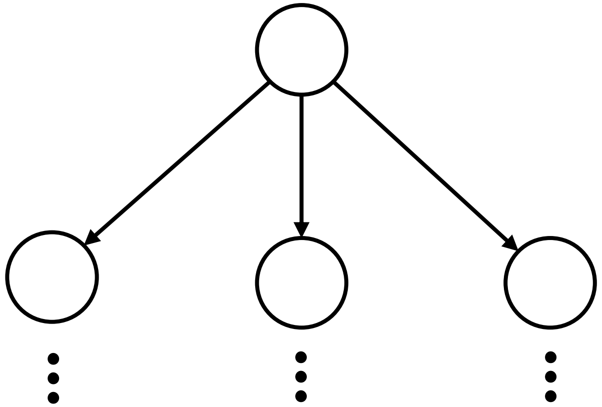
Proof: Simulate a **nondeterministic TM** N with a (deterministic) **multitape TM** D

Idea: D executes the computation of **every branch** in the computation tree of N in a **breadth first search** manner, until an accept state is encountered

We **cannot execute in DFS manner**. A branch which infinitely loops is infinitely long, and we would get stuck in execution.

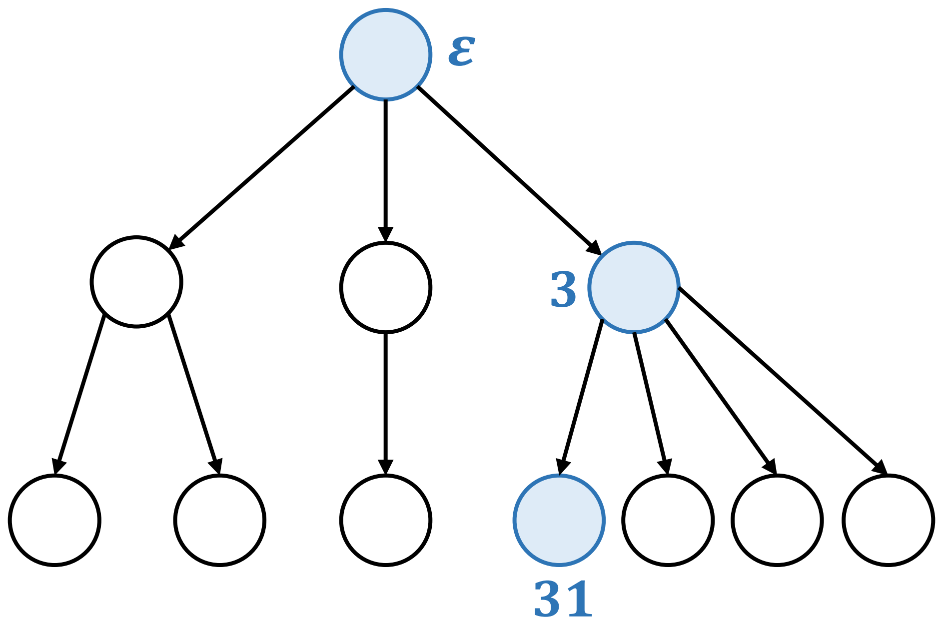


NTM to TM



- During execution, we need a way to **address nodes** in the computation tree
- Every node can have at most $\boldsymbol{b}$ children, where $\boldsymbol{b}$ is the number of **possible options** in transition function
  - Each state has finite possible options since $\boldsymbol{Q}$, $\boldsymbol{\Gamma}$, and **directions** are finite
- Assign symbols $\boldsymbol{\Gamma_b} = \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{b}\}$ to all possible options
  - E.g: $\mathbf{1}$ = (go $\boldsymbol{q_i}$, write ⊔, move **R**) , $\mathbf{2}$ = (go $\boldsymbol{q_i}$, write ⊔, move $\mathbf{L}$), **3** = (go $\boldsymbol{q_i}$, write 0, move $\boldsymbol{L}$), … for all possibilities

NTM to TM

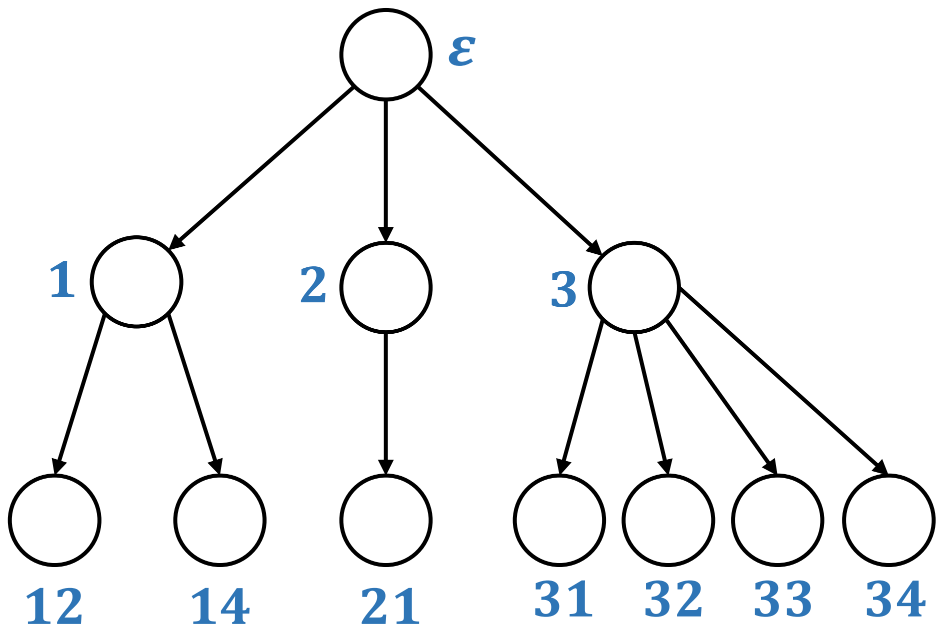


Each node has a string address over $\boldsymbol{\Gamma_b}$ which are the choices made at each step

- Root has address $\boldsymbol{\varepsilon}$
- Address $\mathbf{31}$ corresponds to taking $\mathbf{3}^{rd}$ option at root, followed by $\mathbf{1}^{st}$ option

Note: Some **addresses may not exist** (are invalid) since not all transition options are available in a state

NTM to TM



BFS order of nodes can be **enumerated using addresses**:

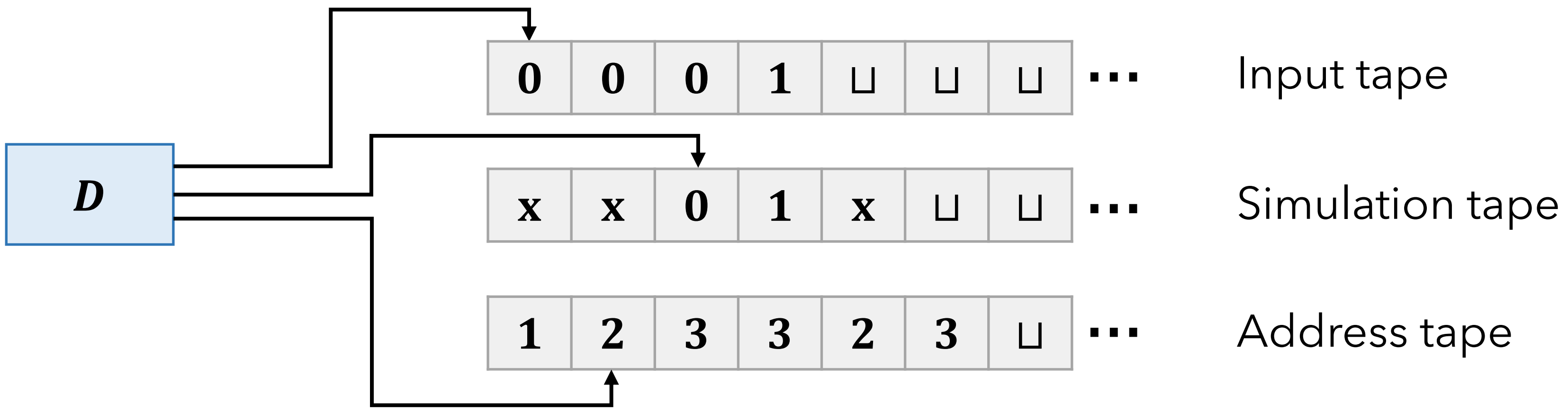
- E.g. Suppose a maximum of **4** possible transition options in TM
- BFS order: $\boldsymbol{\varepsilon}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{11}, \mathbf{12}, \mathbf{13}, \mathbf{14}, \mathbf{21}, \mathbf{22}, \dots$

In the above example, some addresses are invalid
(non-existent sequence of configurations in TM)

NTM to TM

The multitape TM $\boldsymbol{D}$ we use to simulate NTM $\boldsymbol{N}$ uses **three tapes**:

- **Tape 1**: contains input string (never changes)
- **Tape 2**: used to simulate $\boldsymbol{N}$'s tape on its current branch of nondeterministic computation
- **Tape 3**: Keeps track of location (address) in N's computation tree

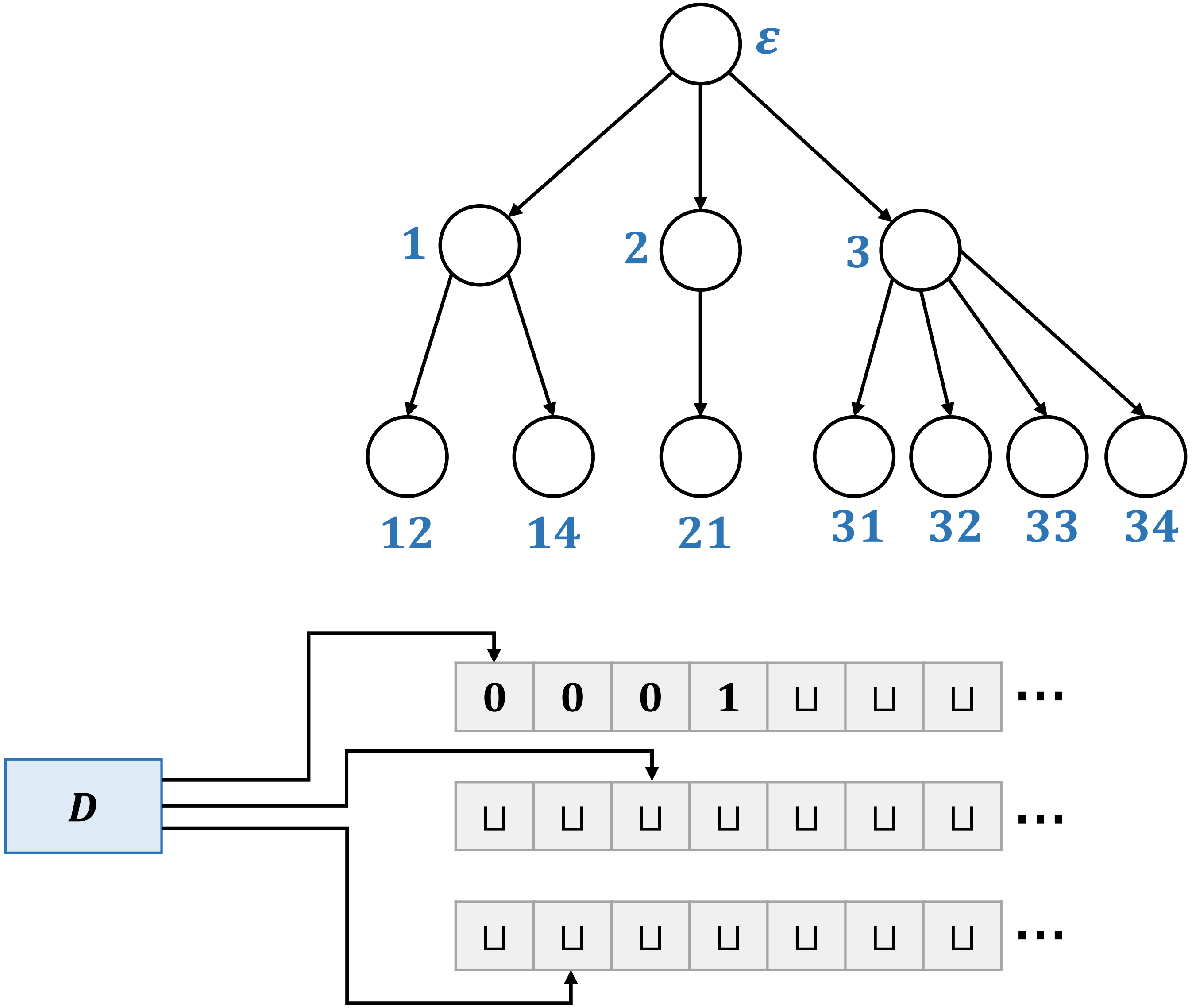


NTM to TM

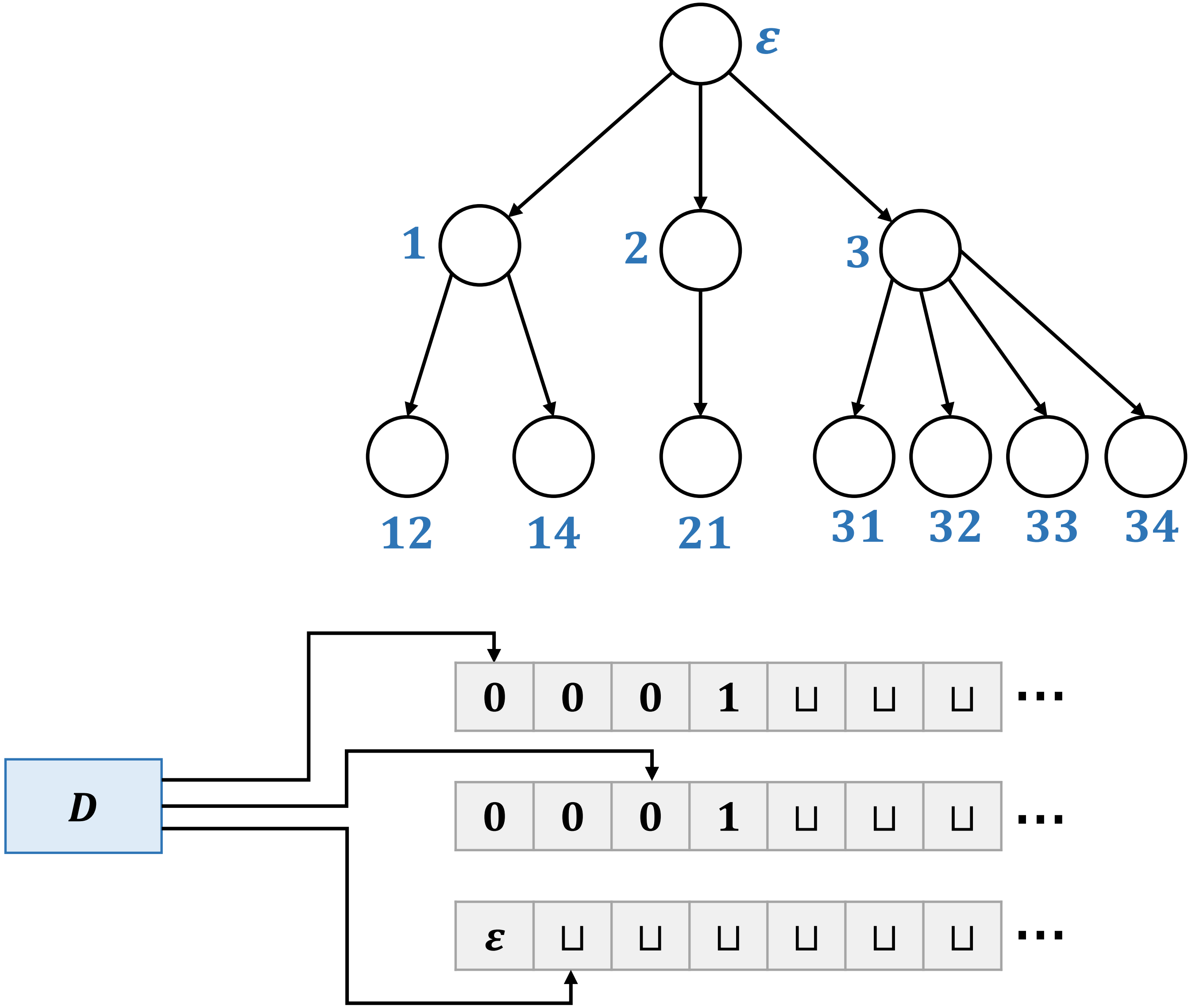
Simulating NTM $\boldsymbol{N}$ on $\boldsymbol{D}$:

1. Initially, **tape 1** contains input $\boldsymbol{w}$, **tape 2** and **tape 3** are empty
2. Copy input on **tape 1** to **tape 2** and set **tape 3** to $\boldsymbol{\varepsilon}$
3. Simulate the **computation branch** described by tape 3 address (start to finish)
   - If computation path has **invalid choice** in transition function, go to **step 4**
   - If computation path leads to **reject**, go to **step 4**
   - If computation path leads to **accept**, then halt and **output accept**
4. Replace address on tape 3 with **next address in BFS ordering**. Simulate next branch of $\boldsymbol{N}$'s computation by going to **step 2**

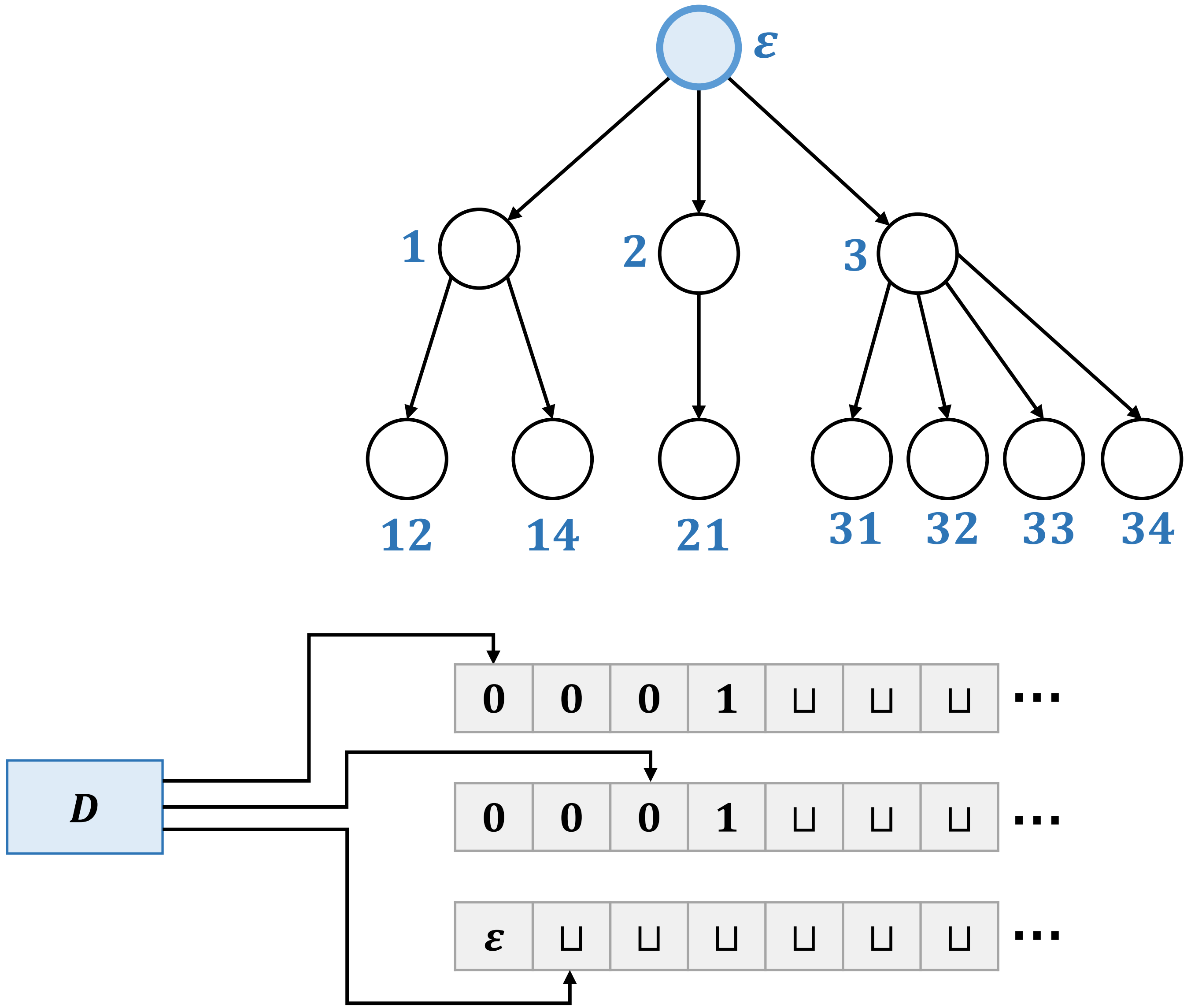
NTM to TM



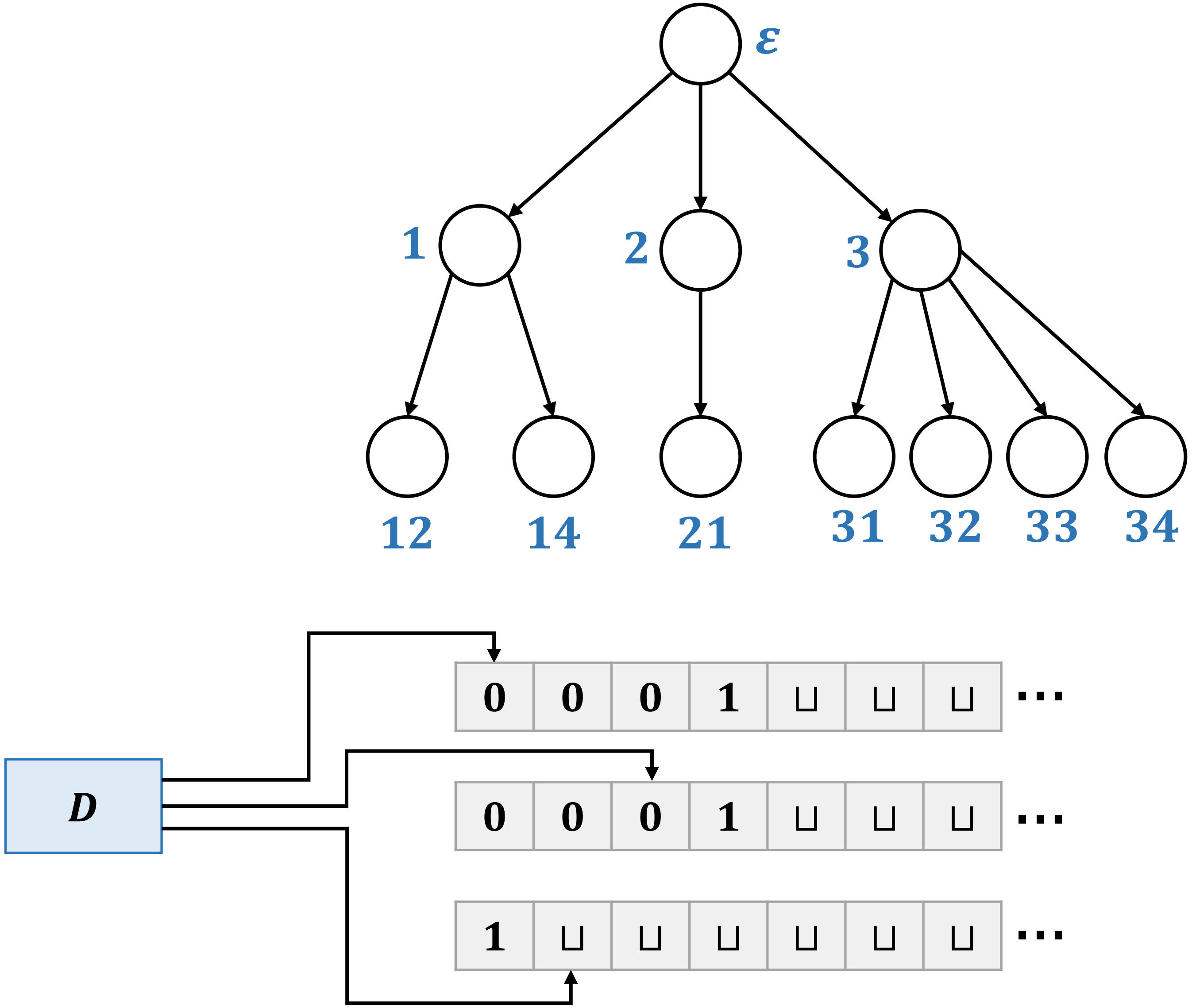
NTM to TM



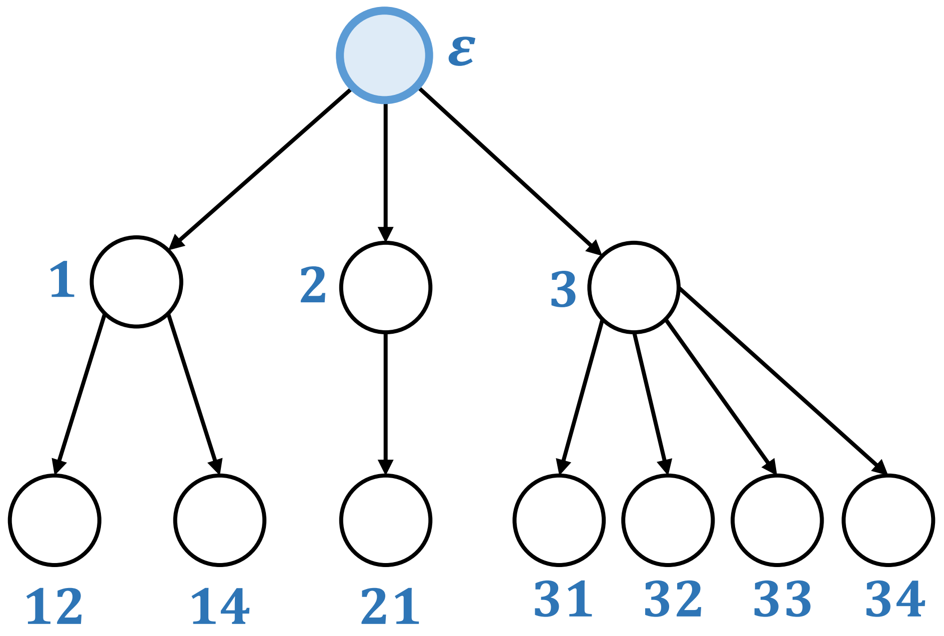
NTM to TM

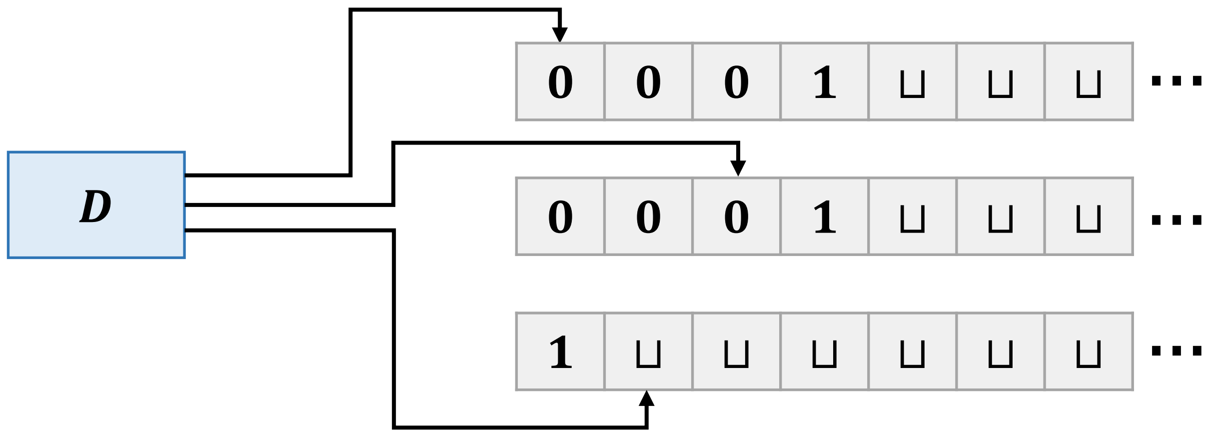


NTM to TM

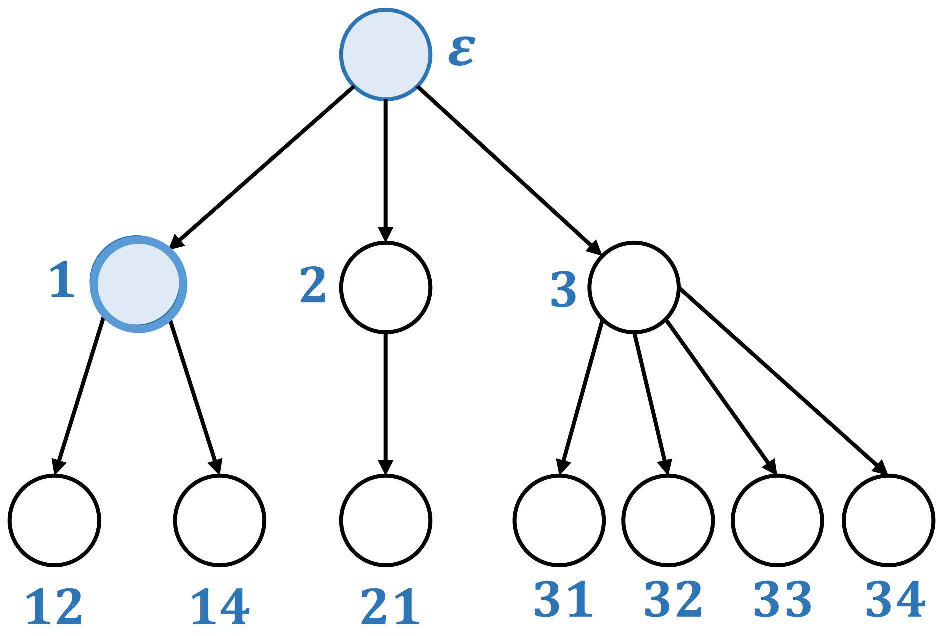


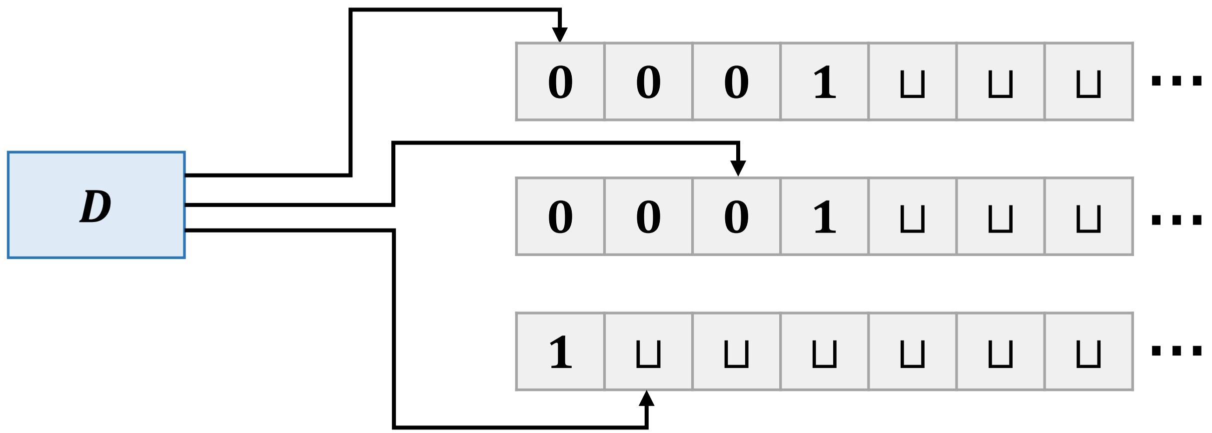
NTM to TM



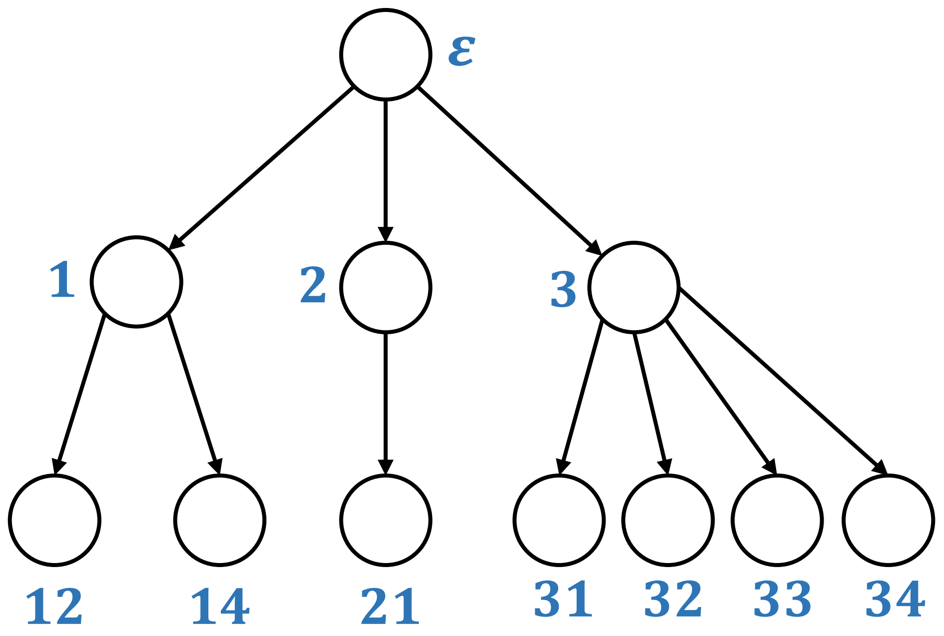


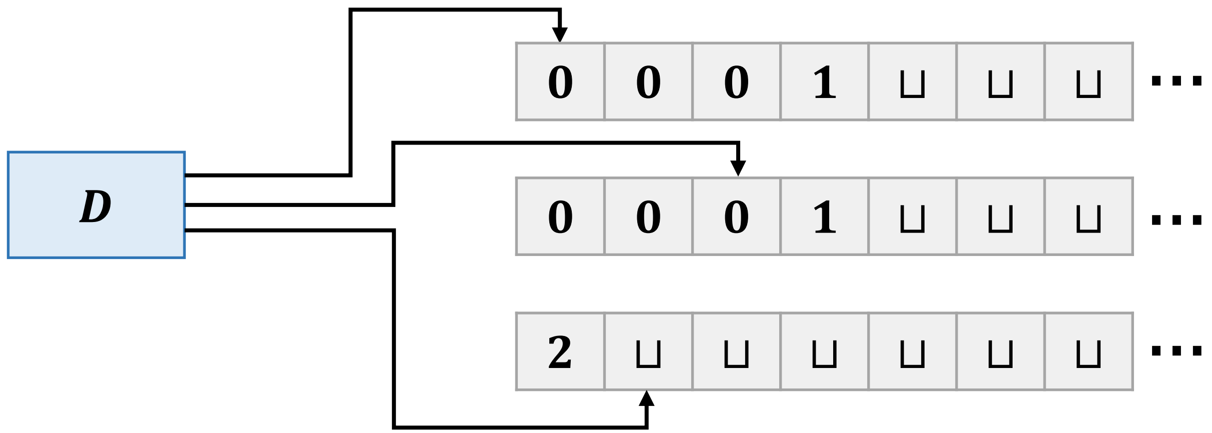
NTM to TM



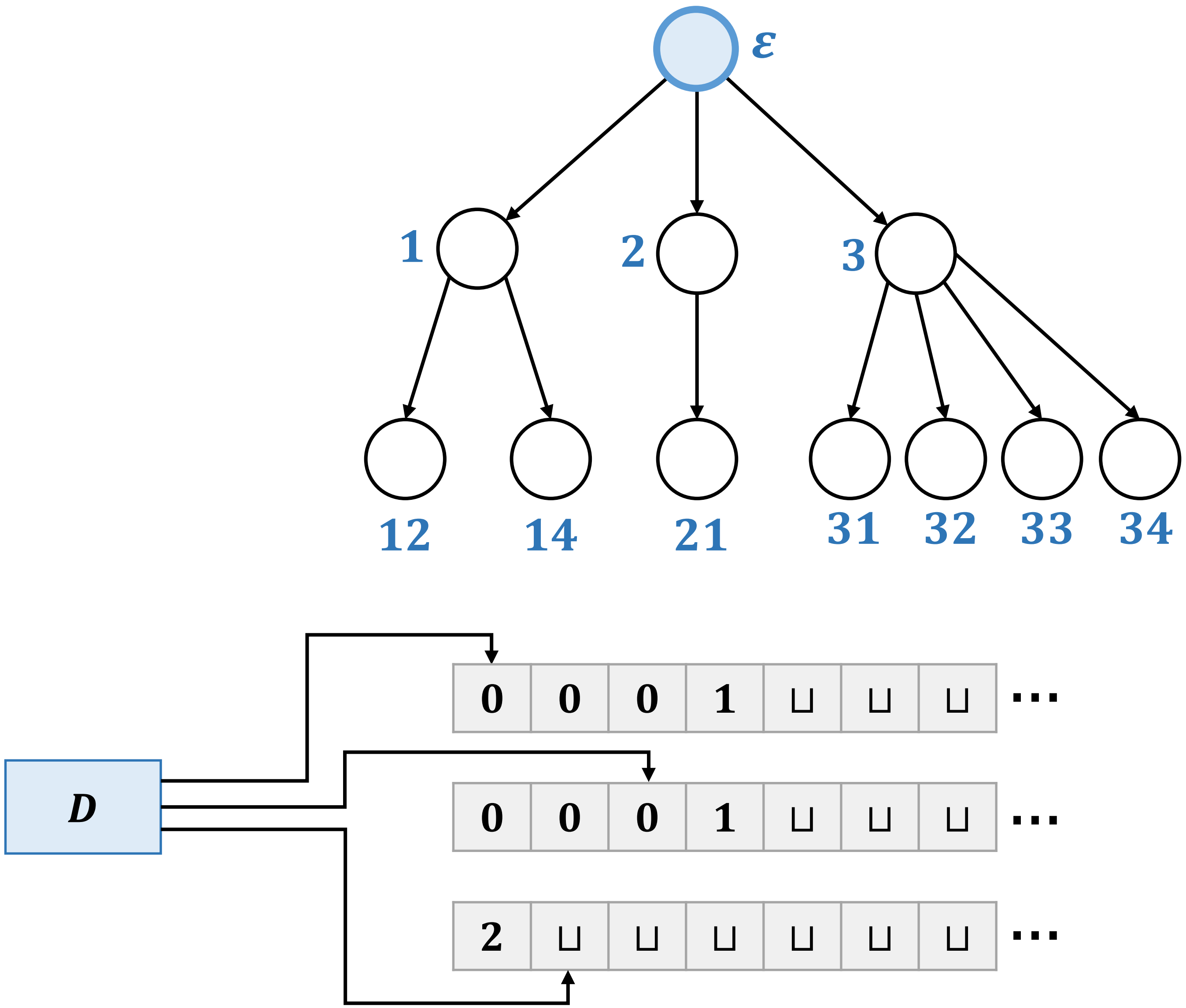


NTM to TM

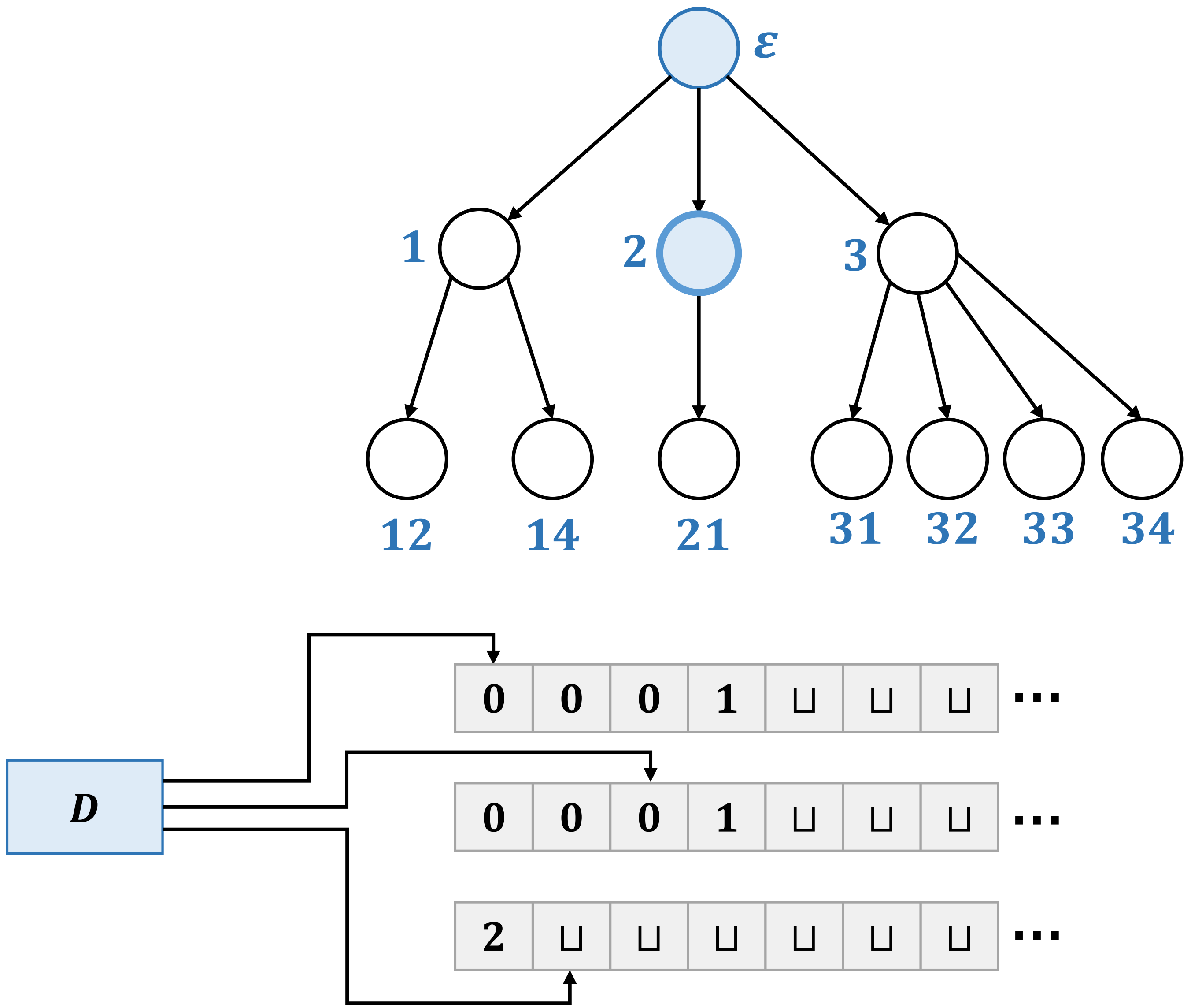




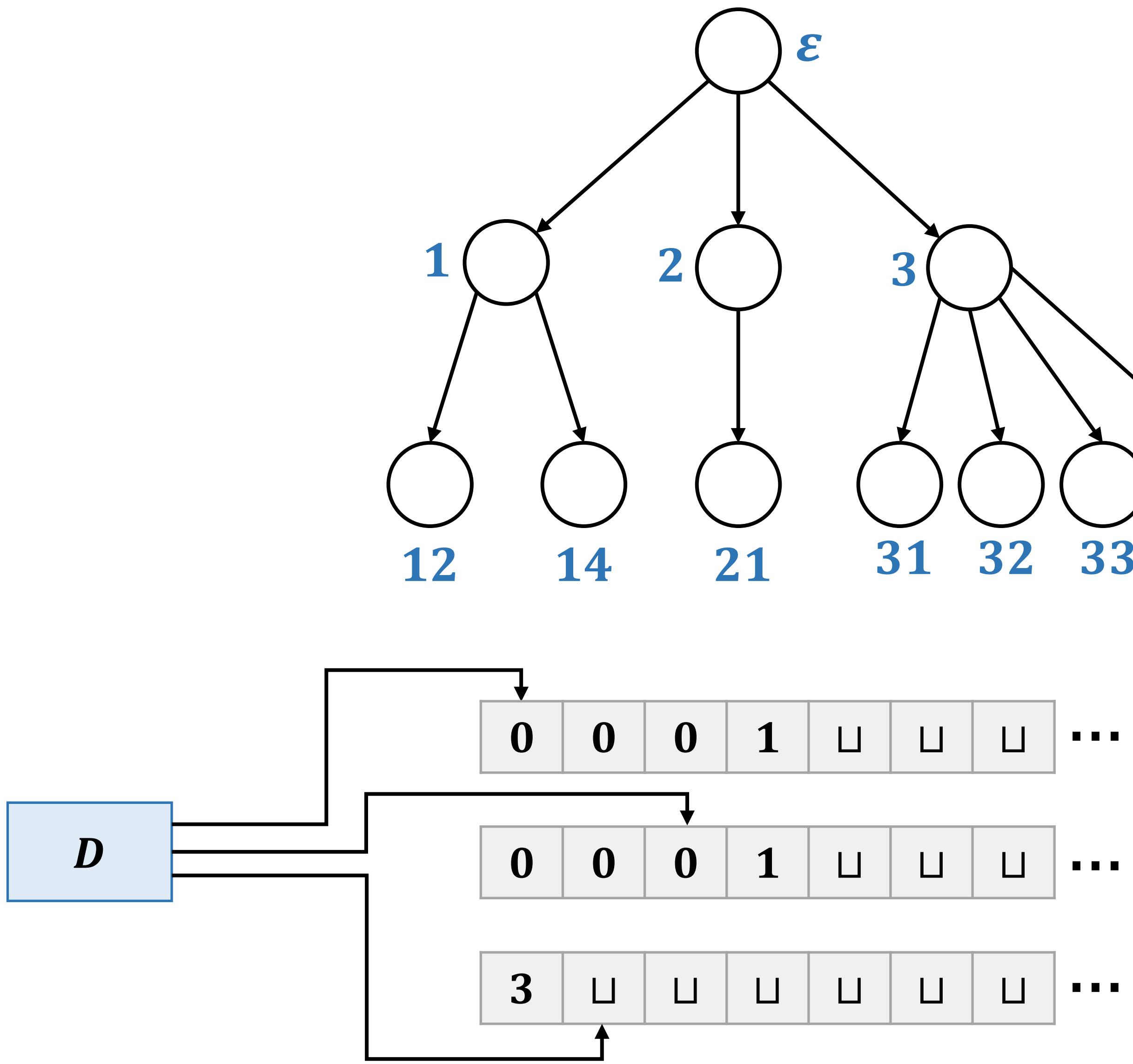
NTM to TM



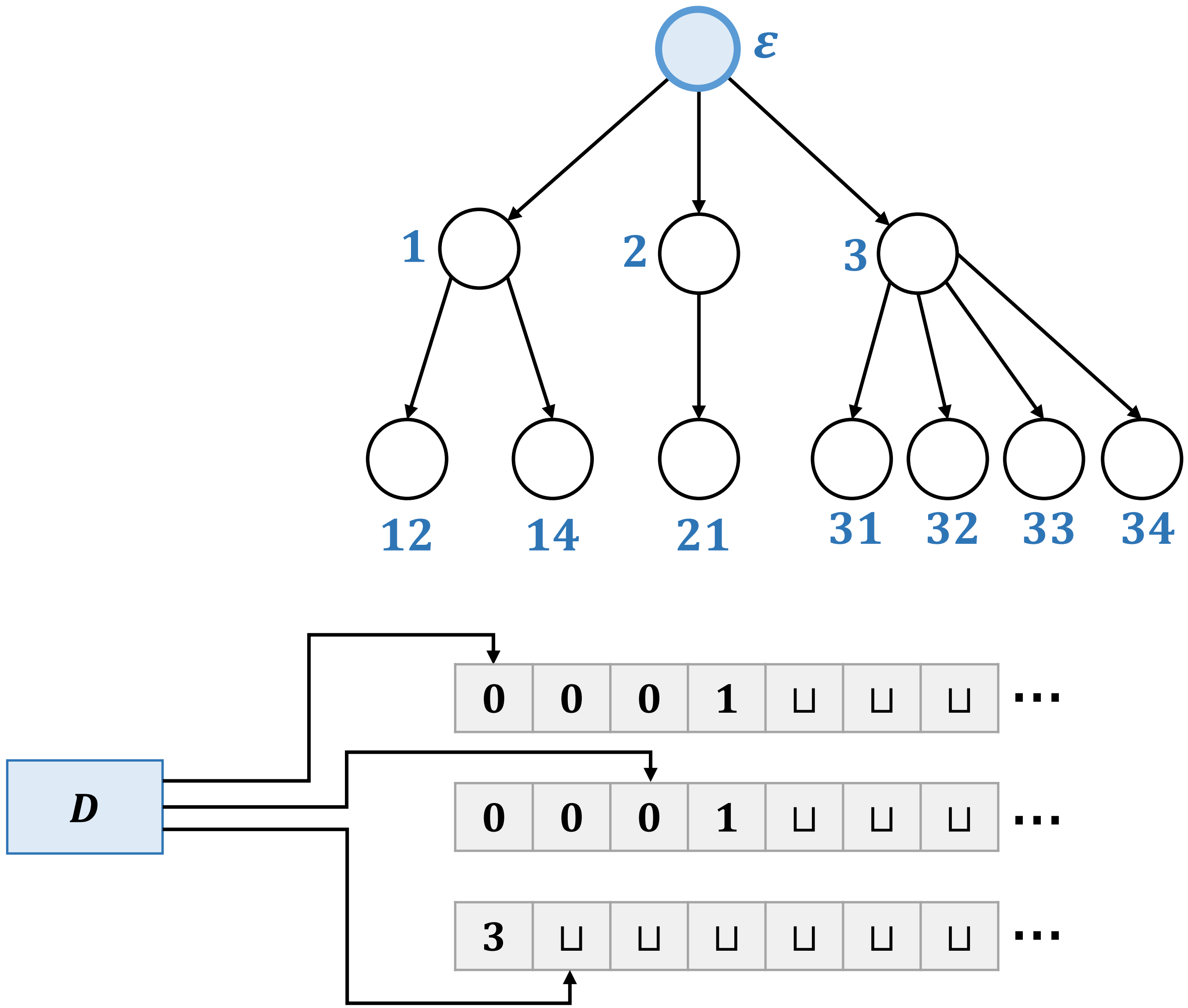
NTM to TM



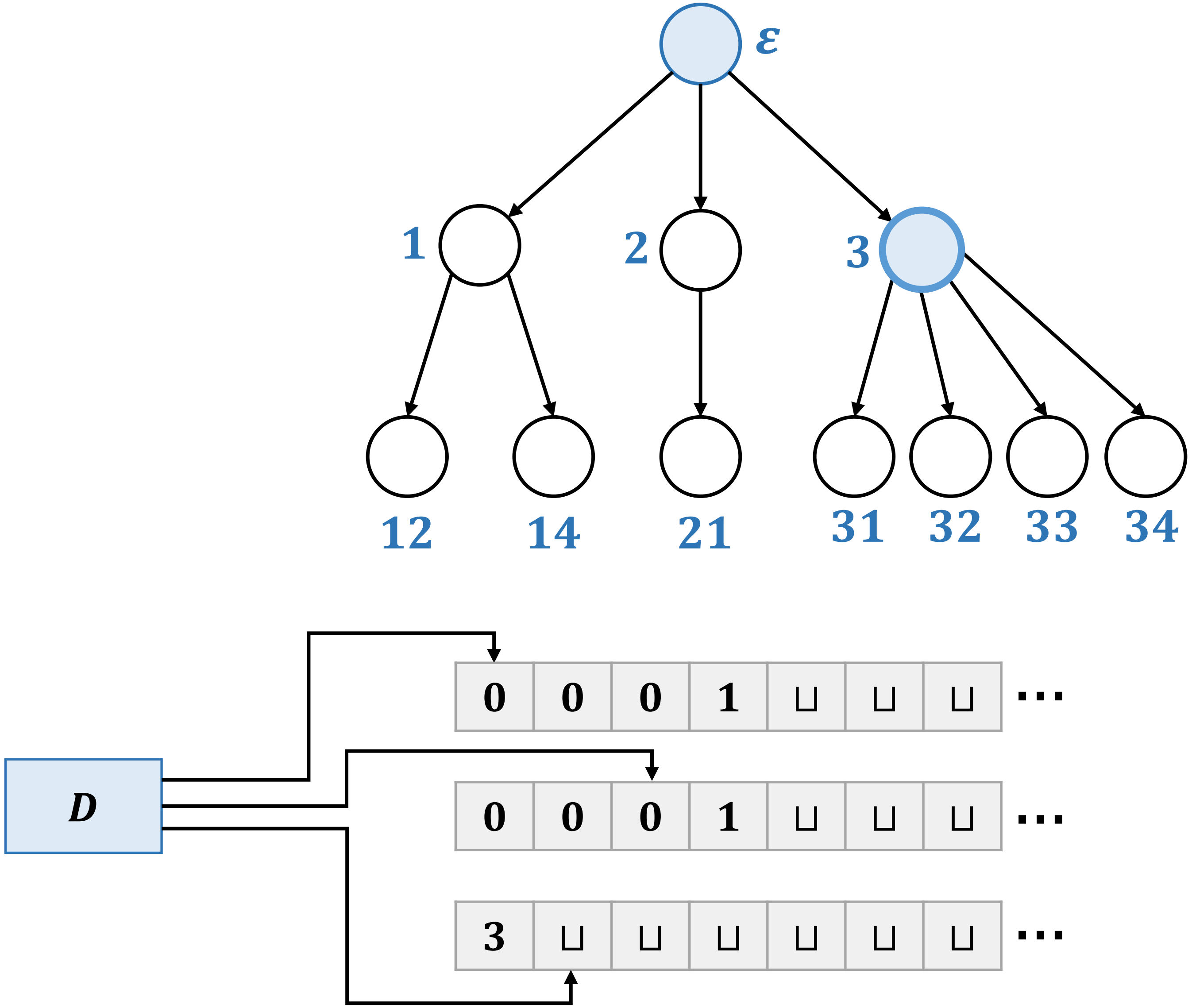
NTM to TM



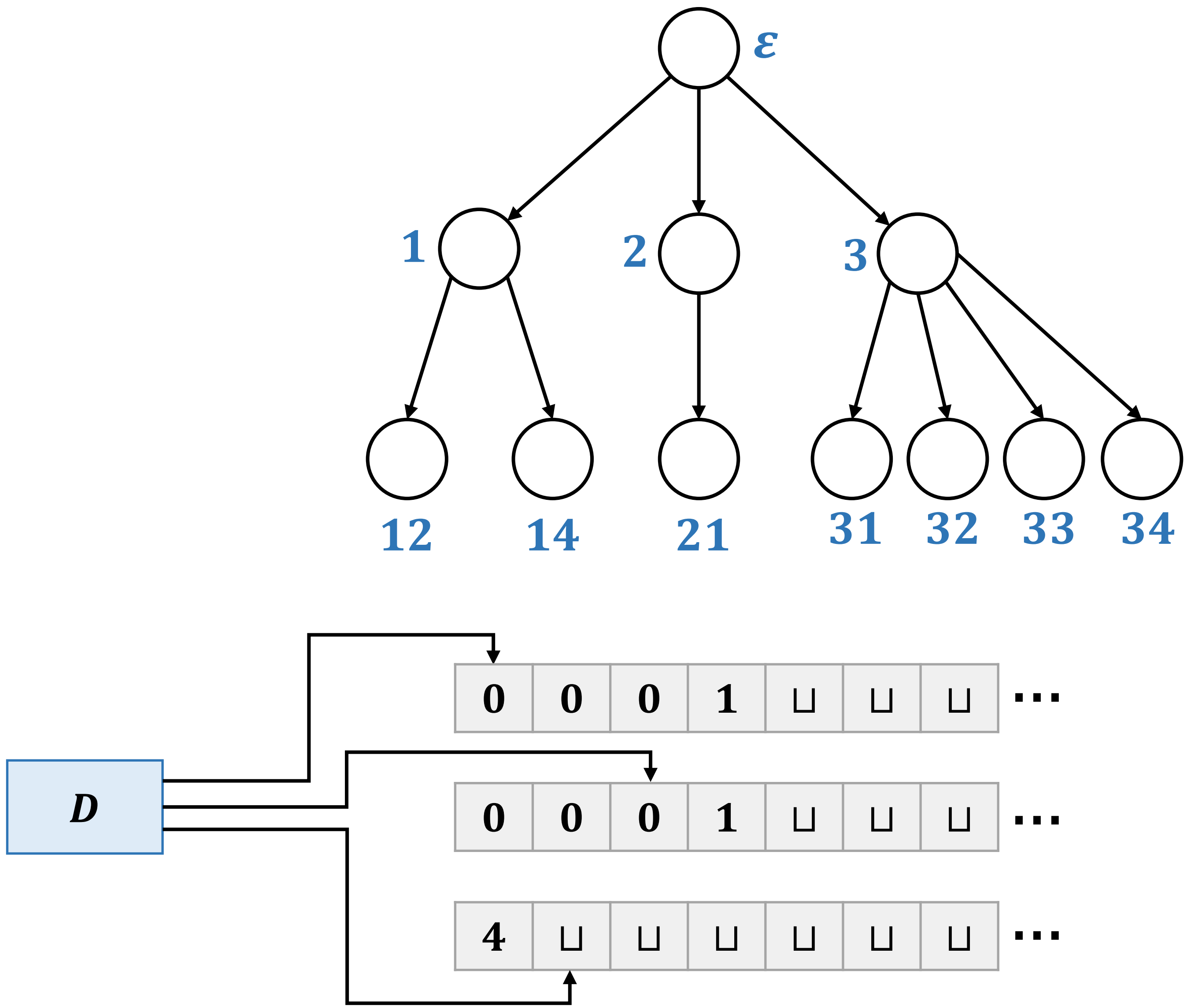
NTM to TM



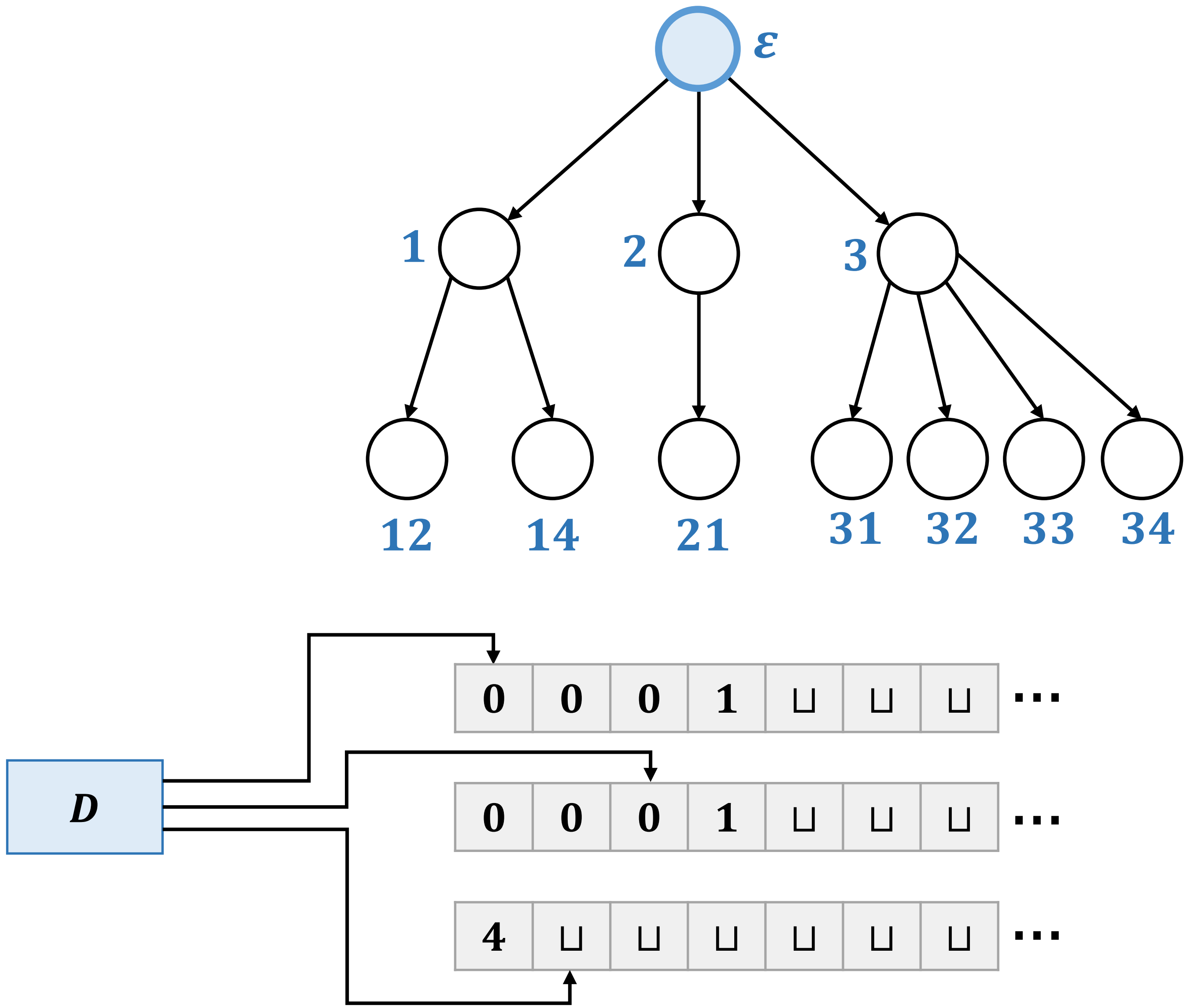
NTM to TM



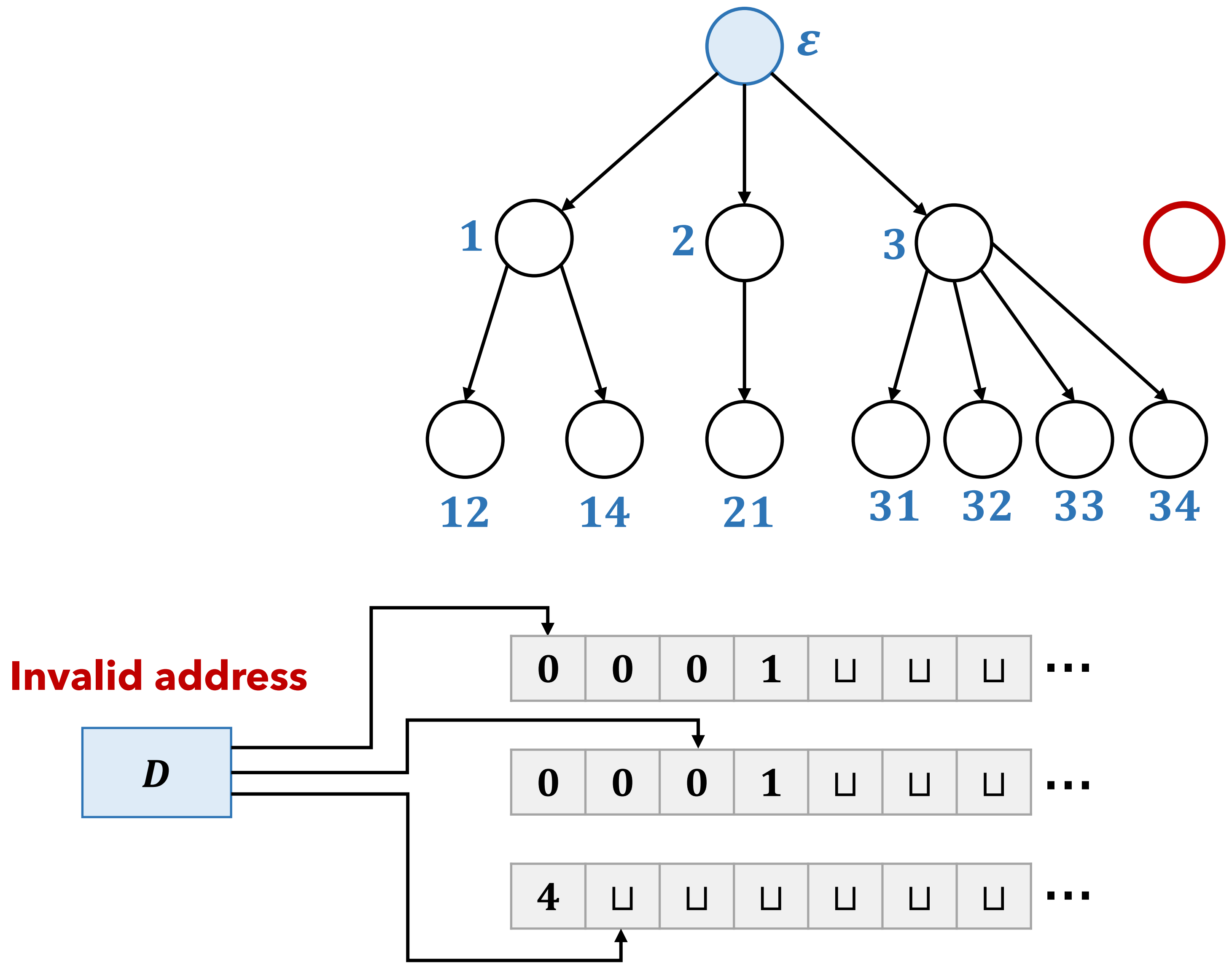
NTM to TM



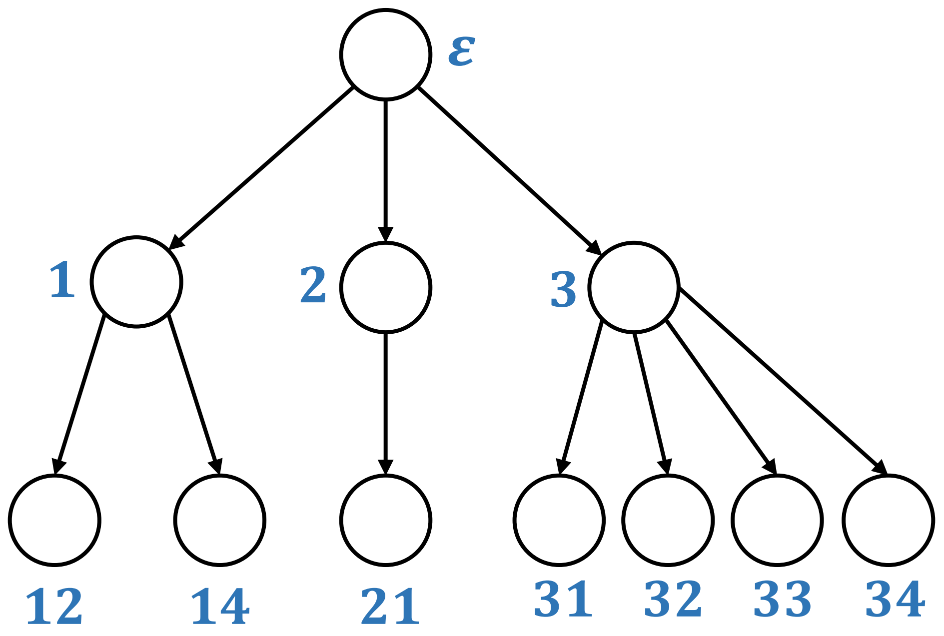
NTM to TM

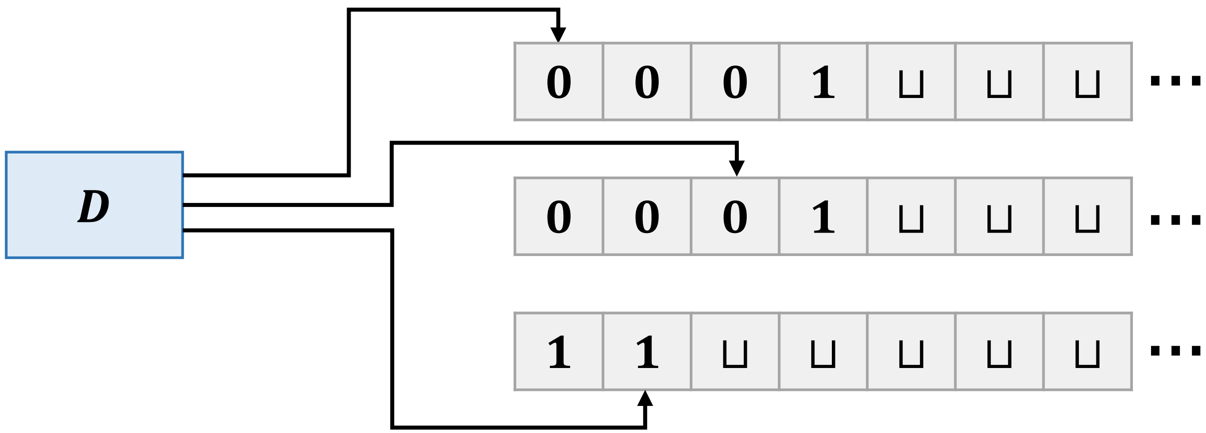


NTM to TM

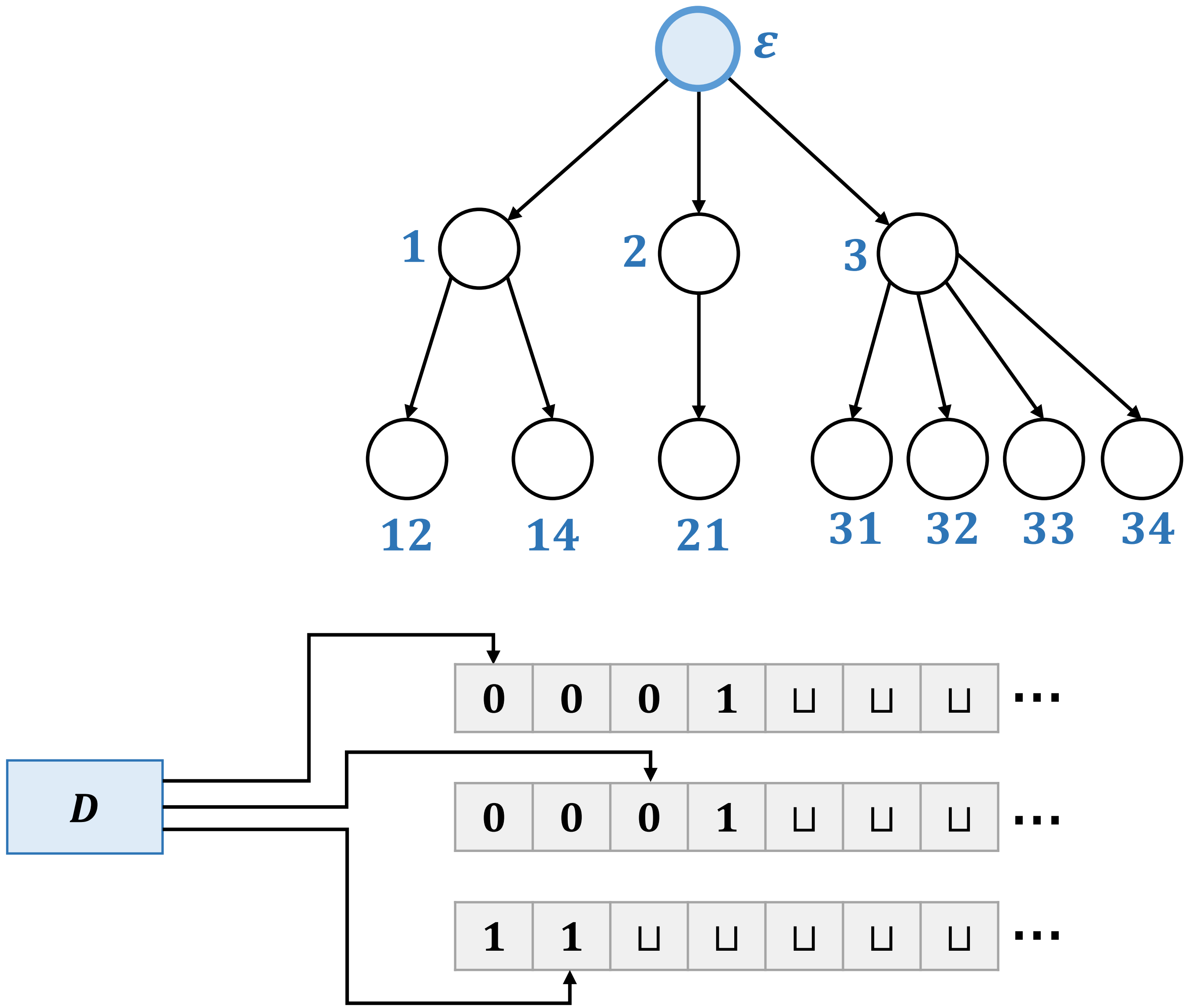


NTM to TM

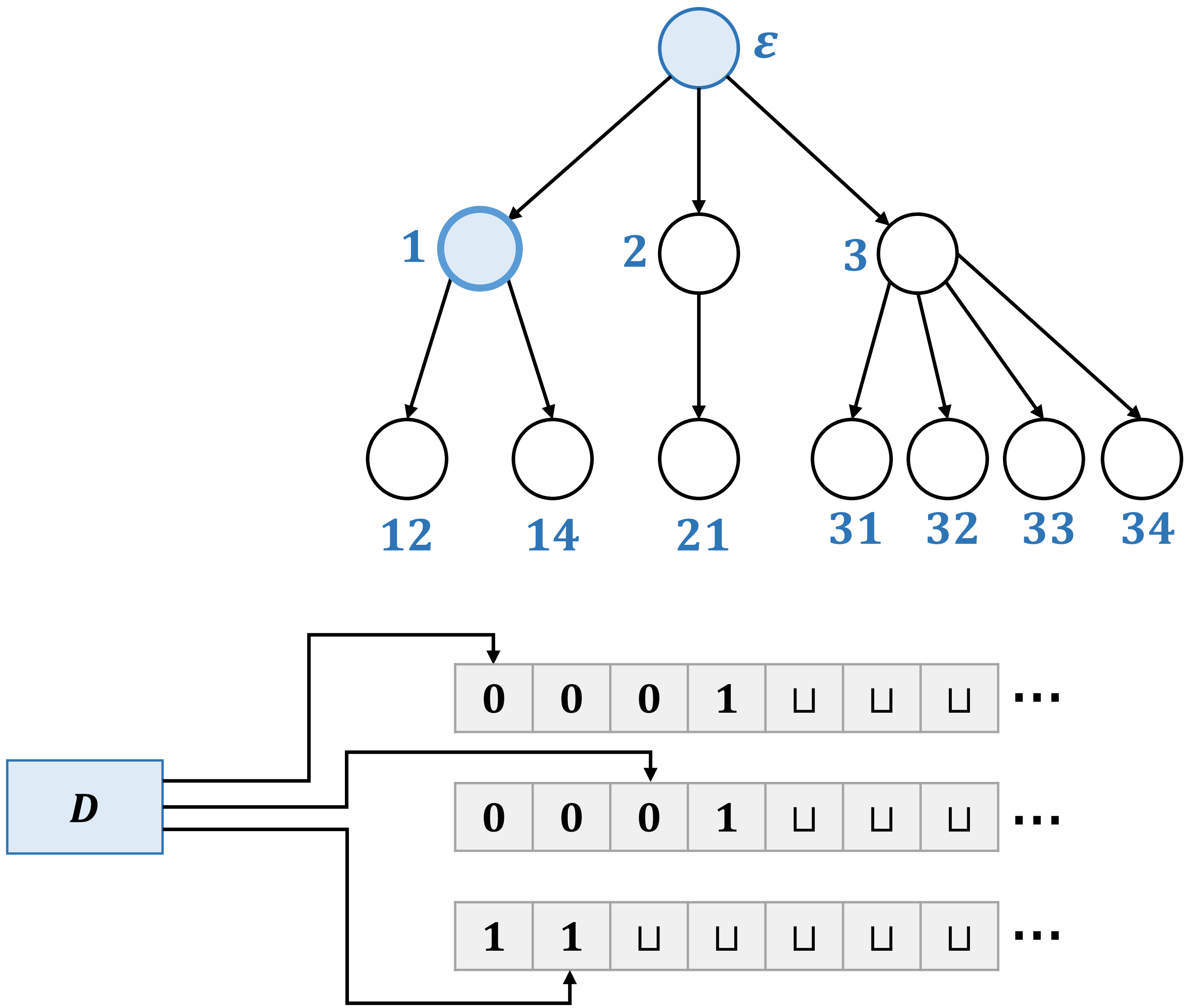




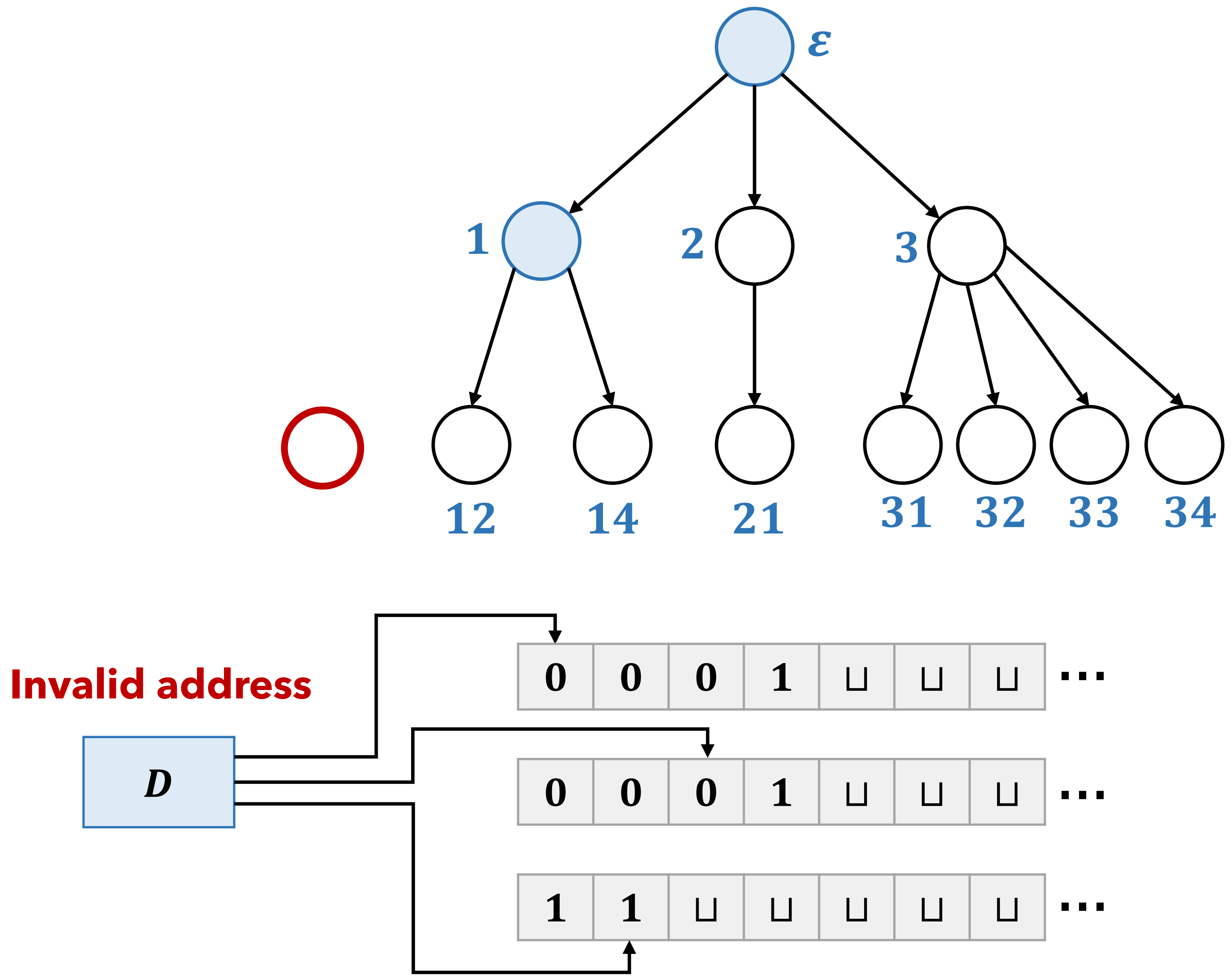
NTM to TM



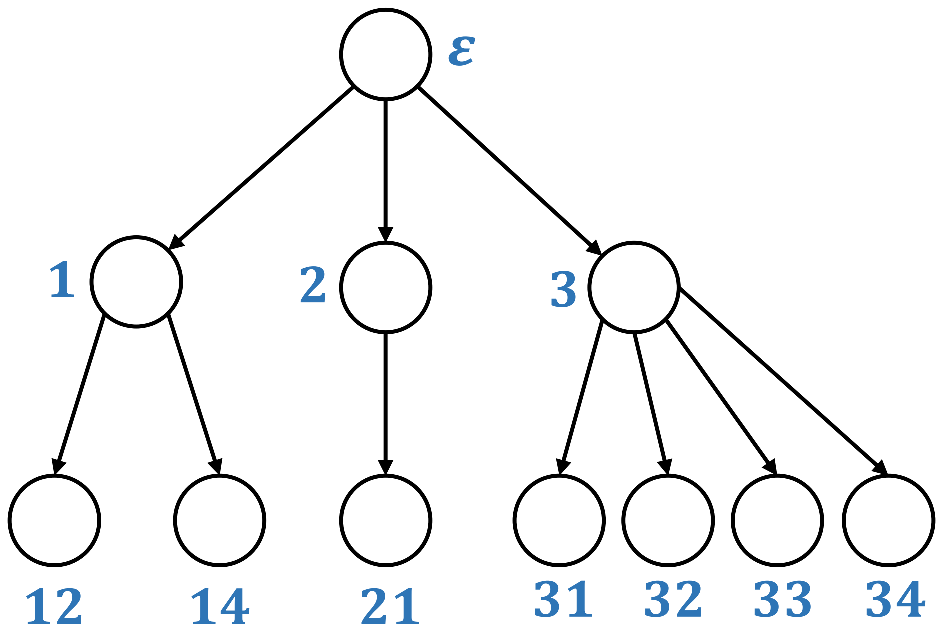
NTM to TM

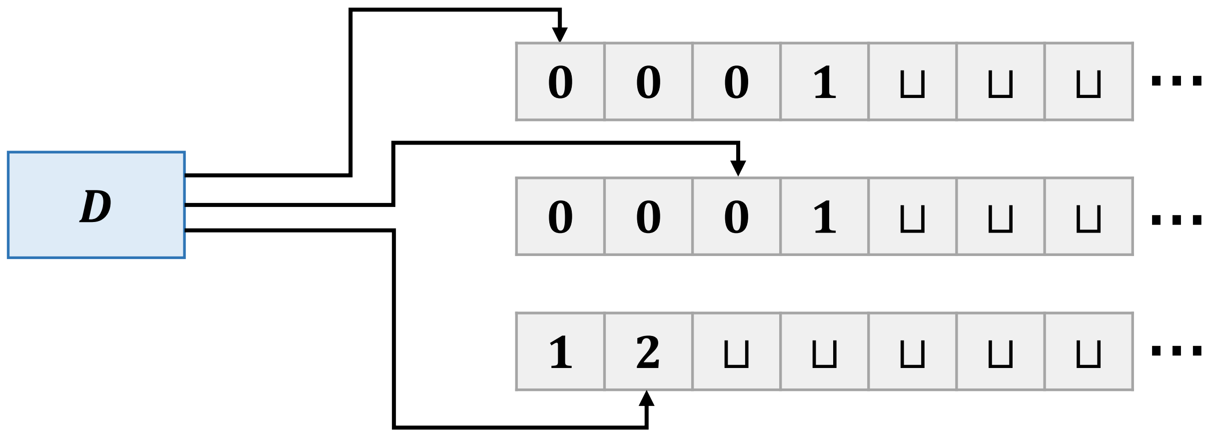


NTM to TM

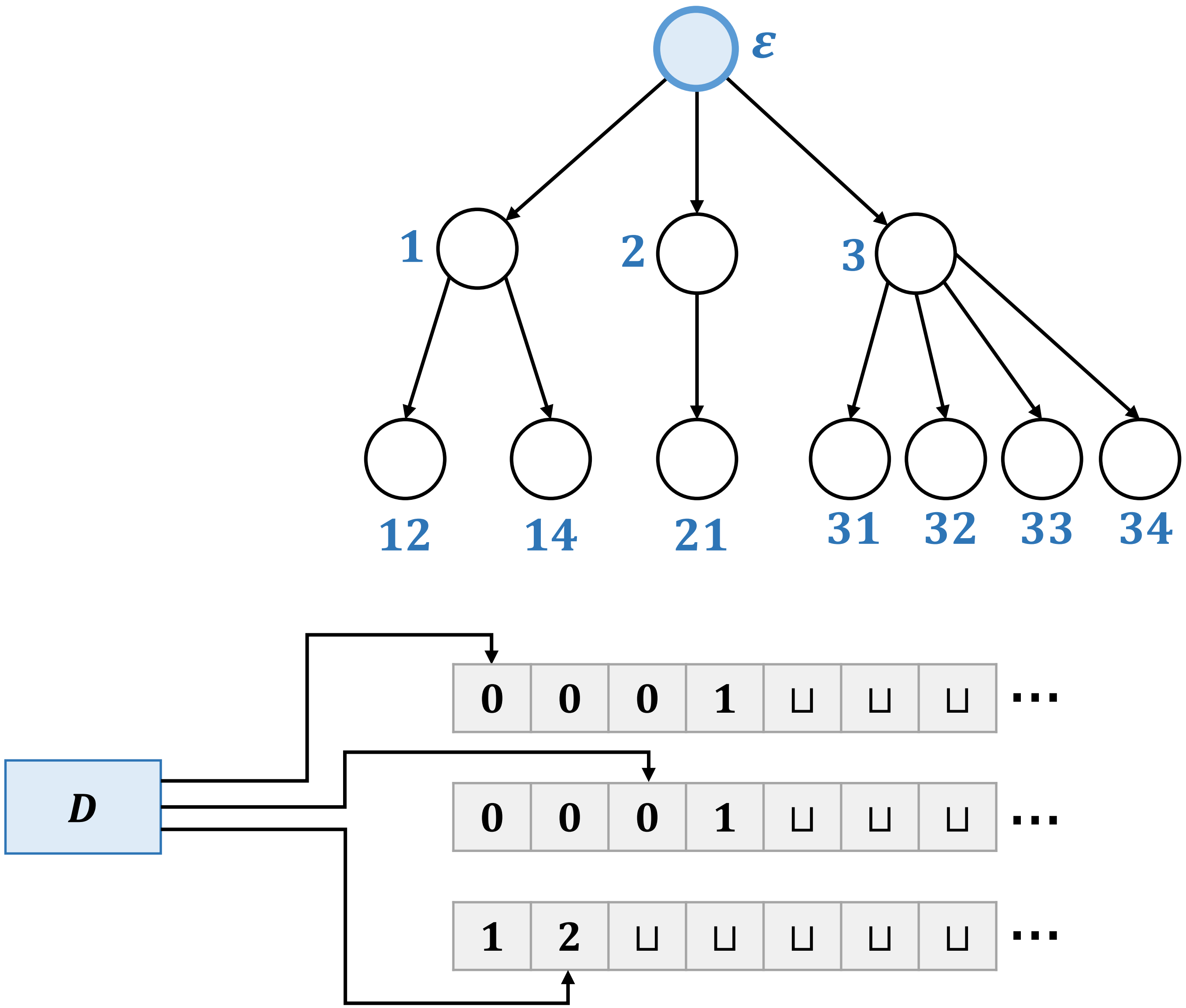


NTM to TM

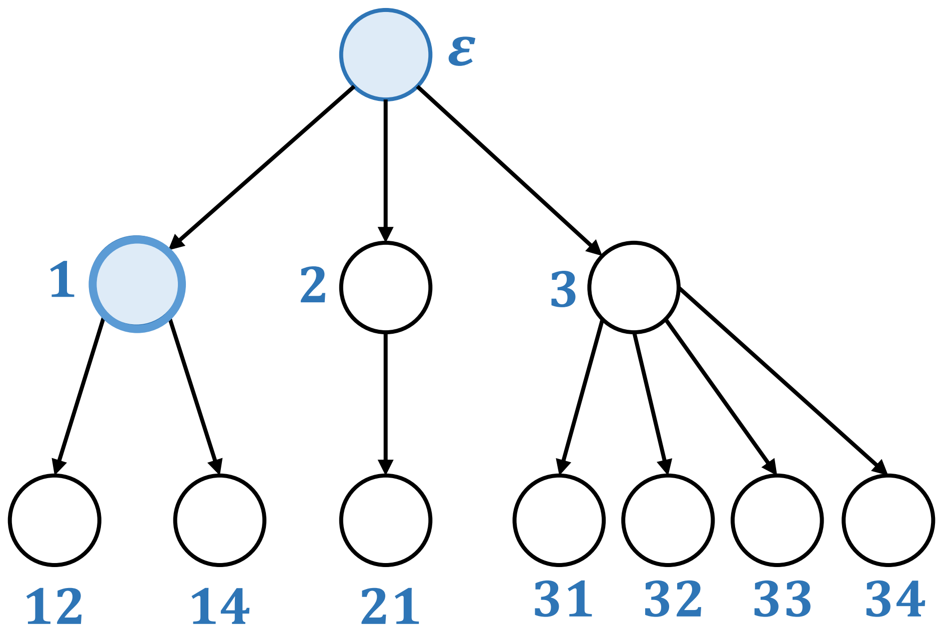


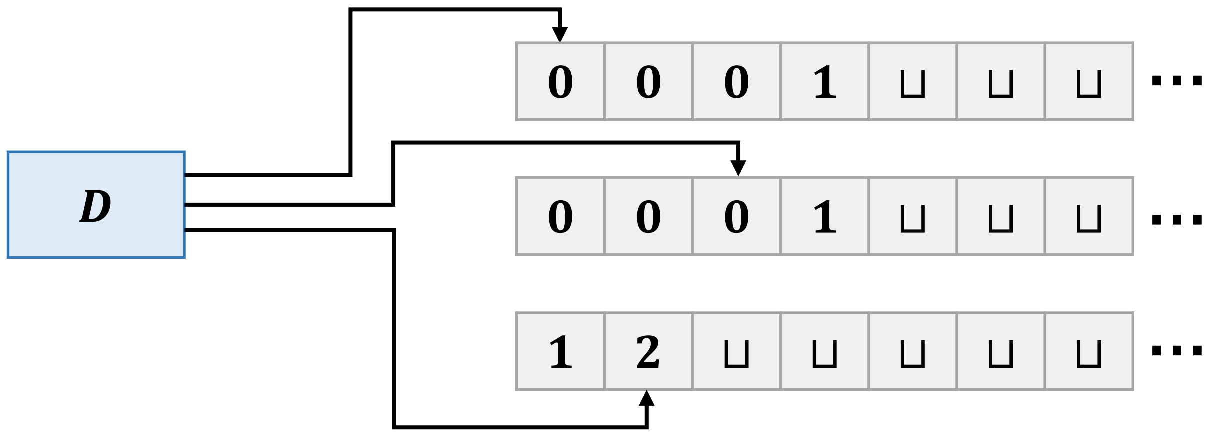


NTM to TM

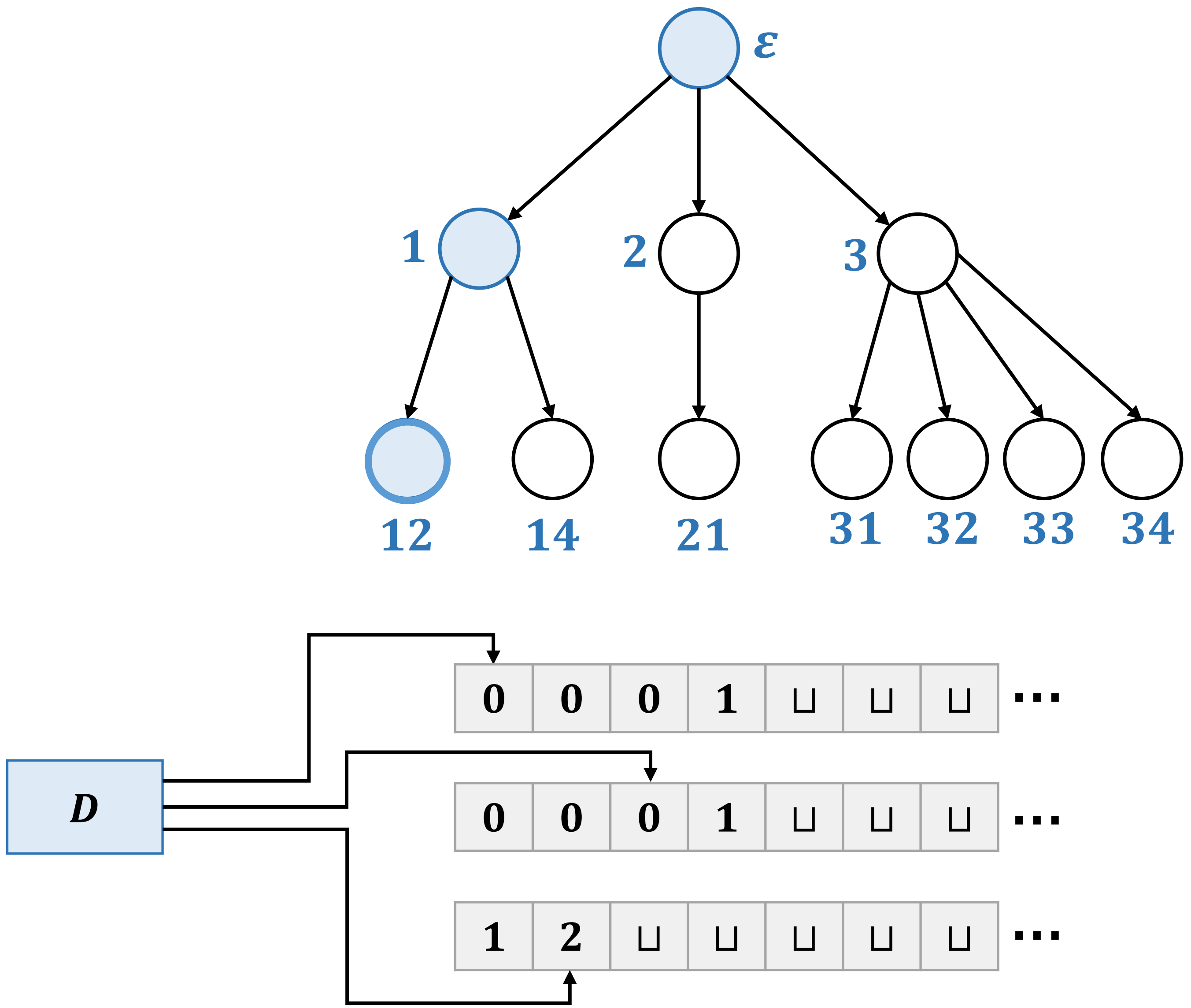


NTM to TM





NTM to TM



Turing Machine Equivalence Corollary

A language L is **Turing-recognizable**

if and only if

some **single-tape Turing machine** recognizes it

if and only if

some **multitape Turing machine** recognizes it

if and only if

some **nondeterministic Turing machine** recognizes it

Nondeterministic Deciders

Recall that:

- A **decider** is a deterministic Turing machine which always halts
- A language is **decidable** if there exists a decider which decides it

A nondeterministic TM is called a **nondeterministic decider** if **all branches of computation halt on all inputs**

Nondeterministic Deciders

Theorem: A language $\boldsymbol{L}$ is **decidable** if and only if a **nondeterministic TM** decides it

Proof:

$\Rightarrow$

- Deterministic TMs are just a **special case** of NTMs
- Therefore, if a deterministic TM decides $\boldsymbol{L}$, then there is an NTM which decides $\boldsymbol{L}$

$\Leftarrow$

- If an NTM decides $\boldsymbol{L}$, then it halts on all branches of computation on all inputs
- When simulating it with a deterministic TM $\boldsymbol{D}$, $\boldsymbol{D}$ will also halt always halt since there are no branches which infinitely loop
- Therefore, if an NTM decides $\boldsymbol{L}$, then there is a deterministic TM which decides $\boldsymbol{L}$