Hamilton Institute

Maynooth University
National University of Ireland Maynooth

Thermodynamics of a multistranded one-dimension Scaffolded DNA Computer

Ahmed Shalaby
1st year PhD
Computer Science Department

Supervisor: Damien Woods

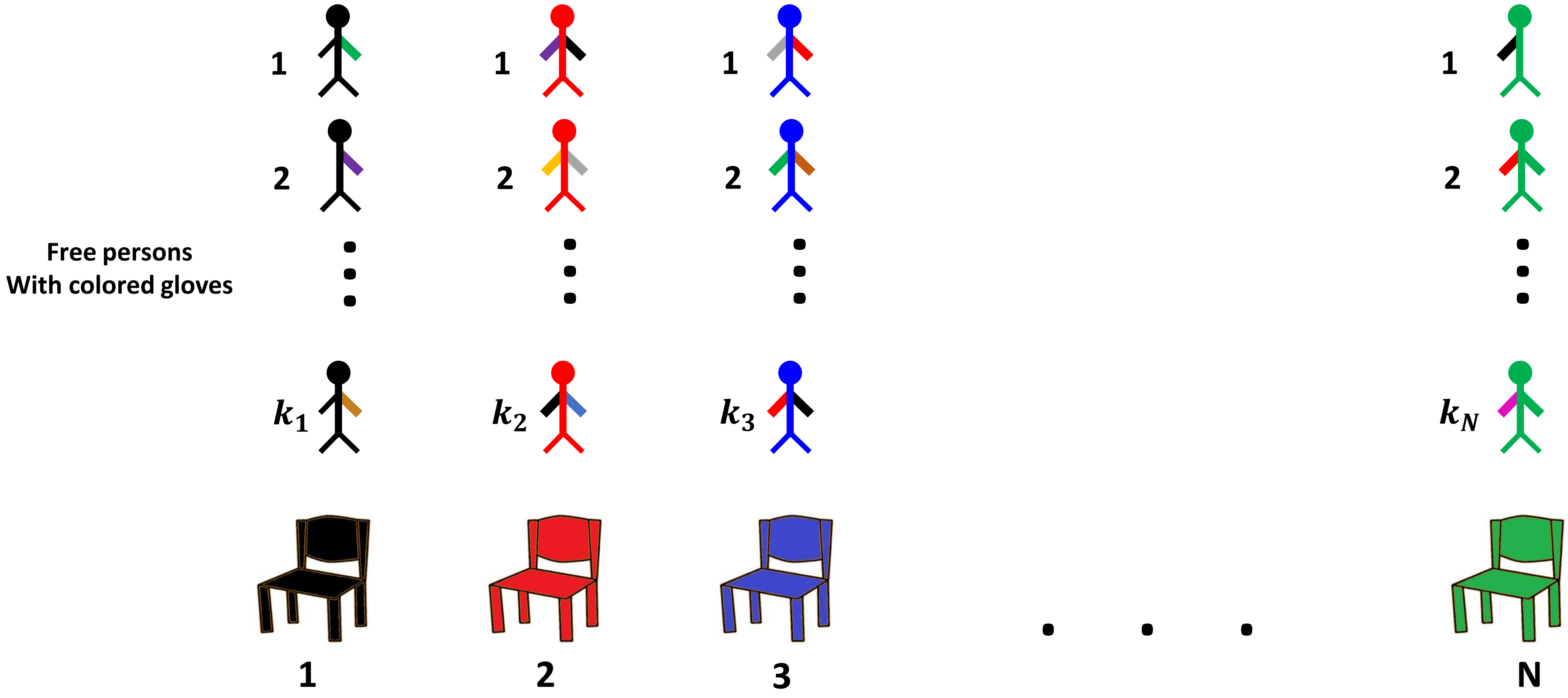
Fondúireacht Eolaíochta Éireann sfi Dá bhfuil romhainn

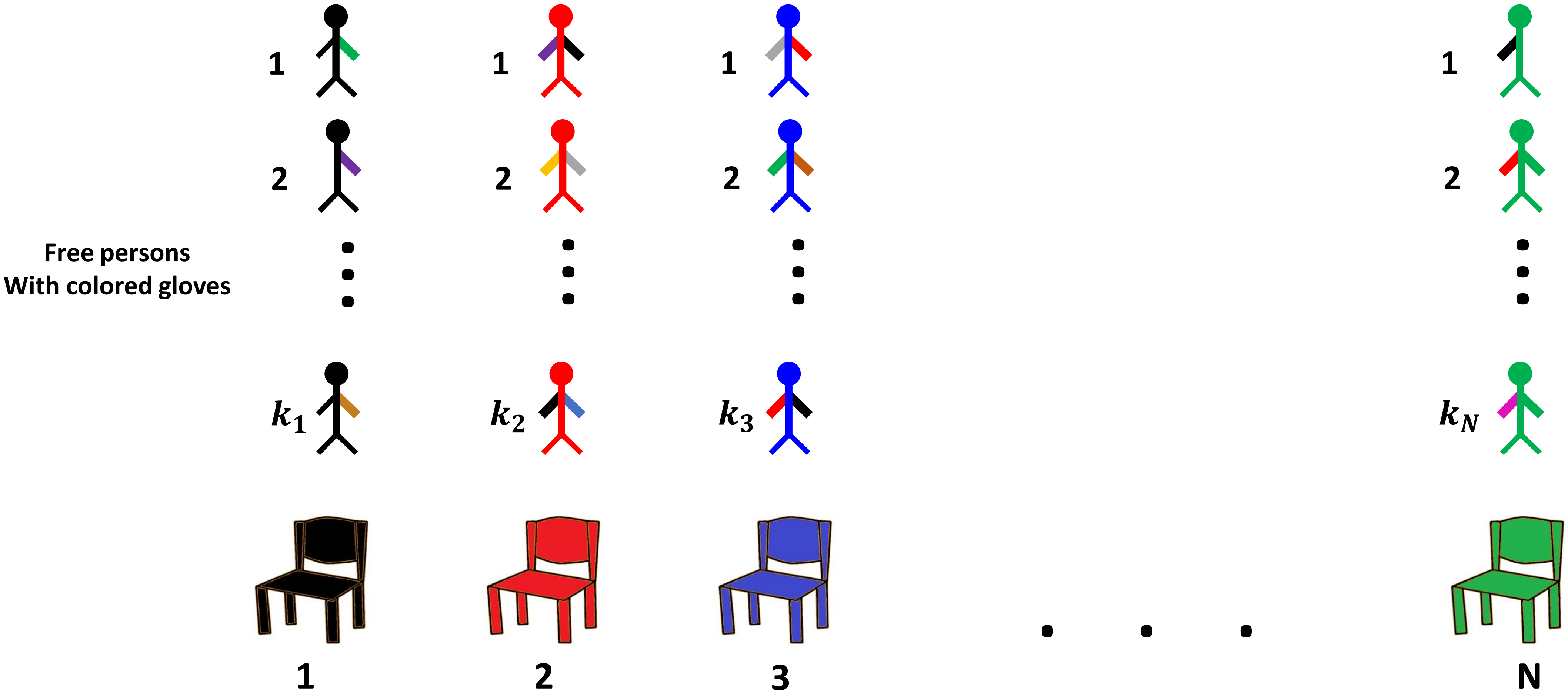
erc

Agenda

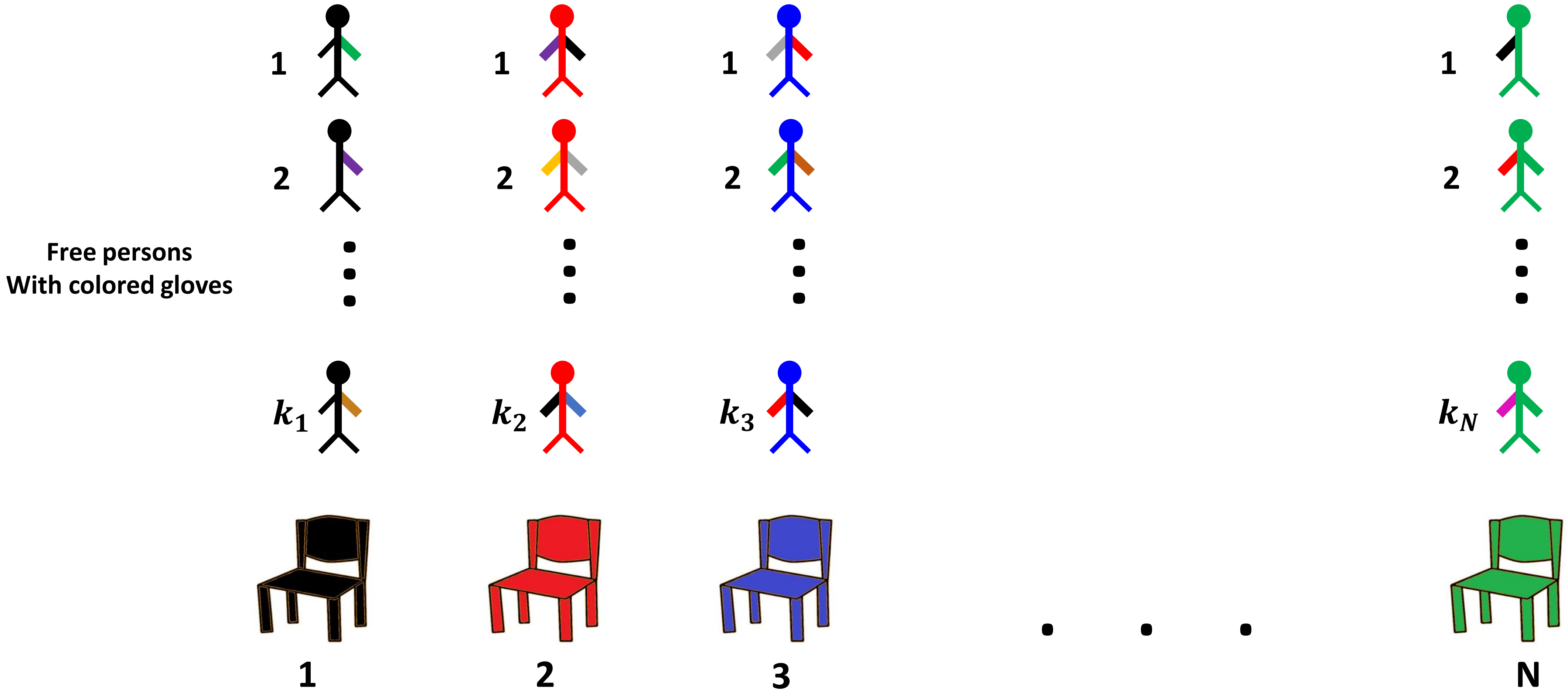
- What is 1D Scaffolded DNA Computer?
- What is the thermodynamic aspects we are interested in?
  - Minimum free energy.
  - Partition function.
- How to compute Partition function efficiently?
- SDC kinetic simulation and experimental results.


1
2
3
. . .
N


Free persons
With colored gloves
1
2
k_1
k_2
k_3
k_N
1
2
3
N



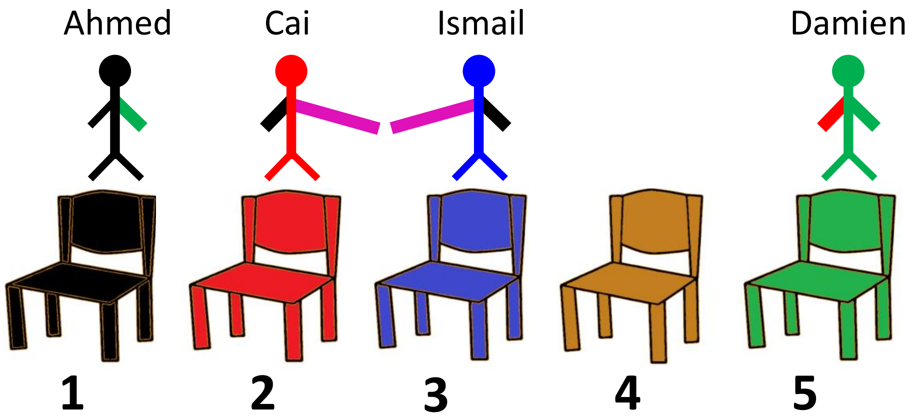
- How many different configurations we will have ?



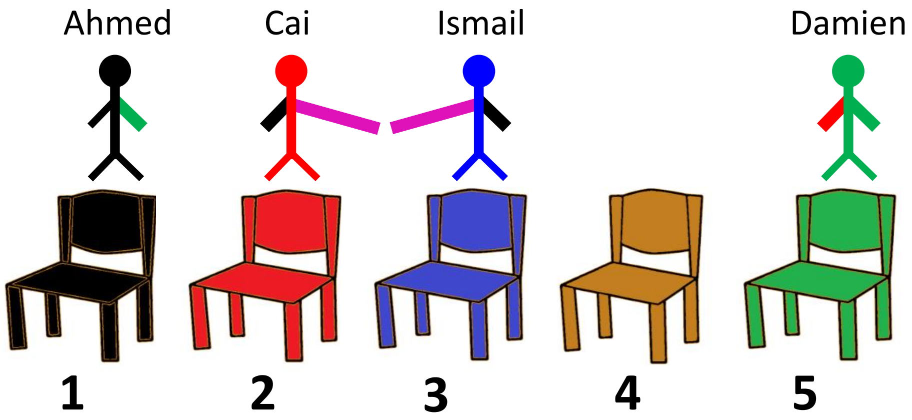
- How many different configurations we will have ?

(Exponential in the # of chairs)

X

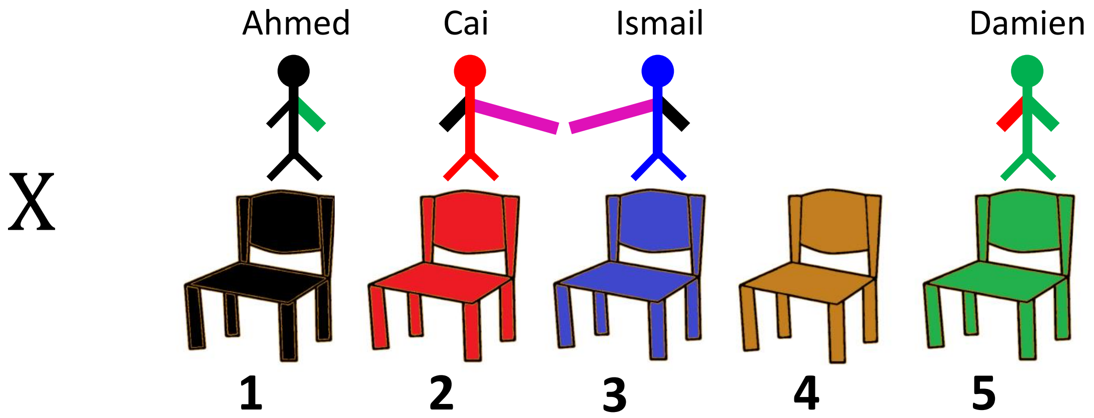


X



$$E(X) = \text{sit}(\text{Ahmed}) + \text{sit}(\text{Cai}) + \text{sit}(\text{Ismail}) + \text{sit}(\text{Damien}) + 4 * \text{sit\_convincing\_cost} + \text{handshake}(\text{Cai}, \text{Ismail}).$$

\+ \+ \+ \+ - \+



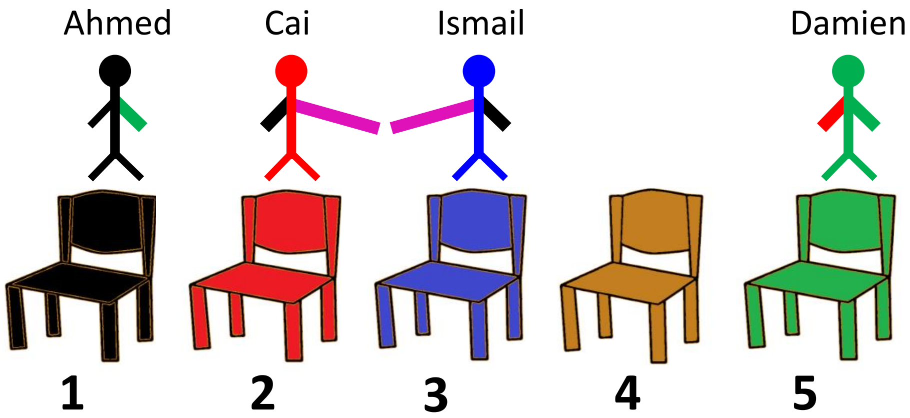
$$E(X) = \text{sit(Ahmed)} + \text{sit(Cai)} + \text{sit(Ismail)} + \text{sit(Damien)} + 4 * \text{sit\_convincing\_cost} + \text{handshake(Cai, Ismail)}.$$

\+ \+ \+ \+ - \+

$$E(X) = \sum_{p \in X} \text{sit(p)} + l * \text{sit\_convincing\_cost} + \sum_{p_i, p_{i+1} \in X} \text{handshake}(p_i, p_{i+1}).$$

configuration X of size l

X



$$E(X) = \text{sit(Ahmed)} + \text{sit(Cai)} + \text{sit(Ismail)} + \text{sit(Damien)} + 4 * \text{sit\_convincing\_cost} + \text{handshake(Cai, Ismail)}.$$

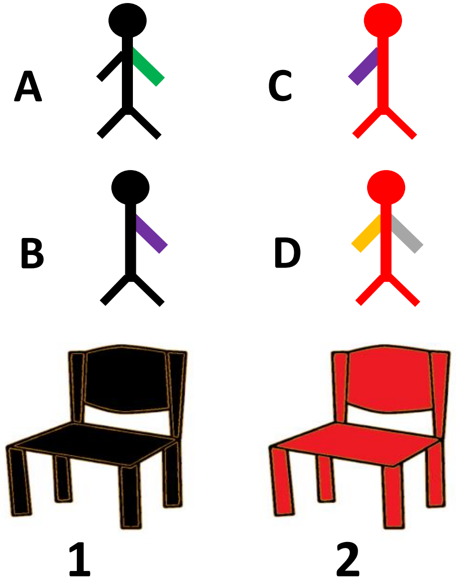
\+ \+ \+ \+ - \+

$$E(X) = \sum_{p \in X} \text{sit(p)} + l * \text{sit\_convincing\_cost} + \sum_{p_i, p_{i+1} \in X} \text{handshake}(p_i, p_{i+1}).$$

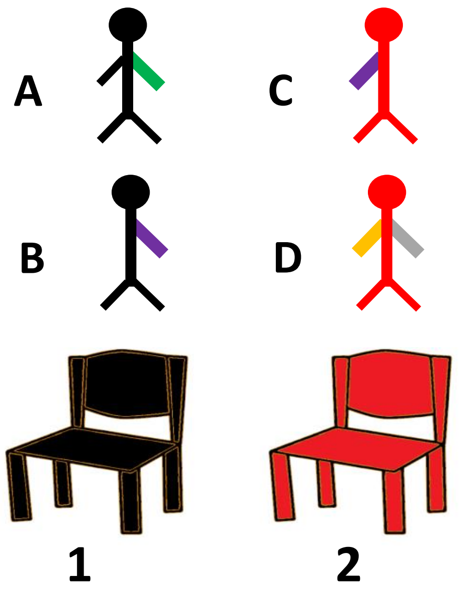
configuration X of size l

We further assume the following:

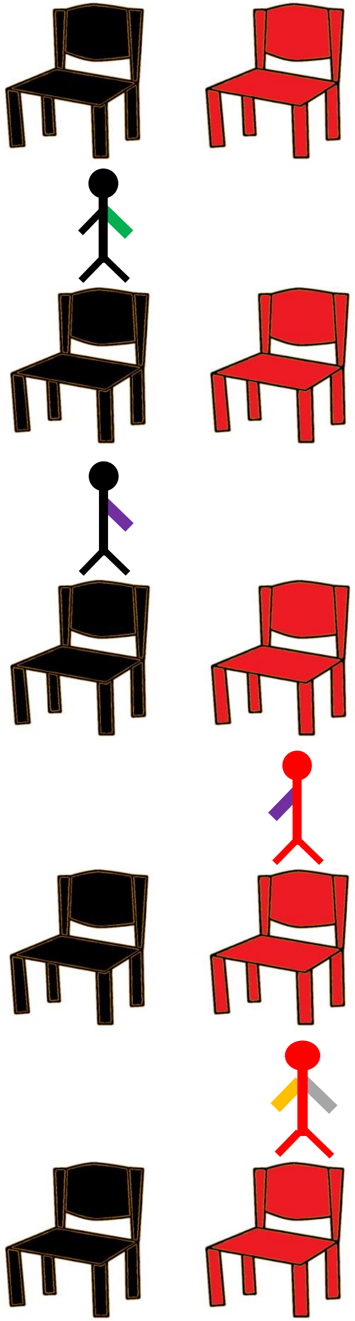
- $|\text{sit}(p)| > |\text{sit\_convincing\_cost}|$. (you always gain by convincing a person to sit)

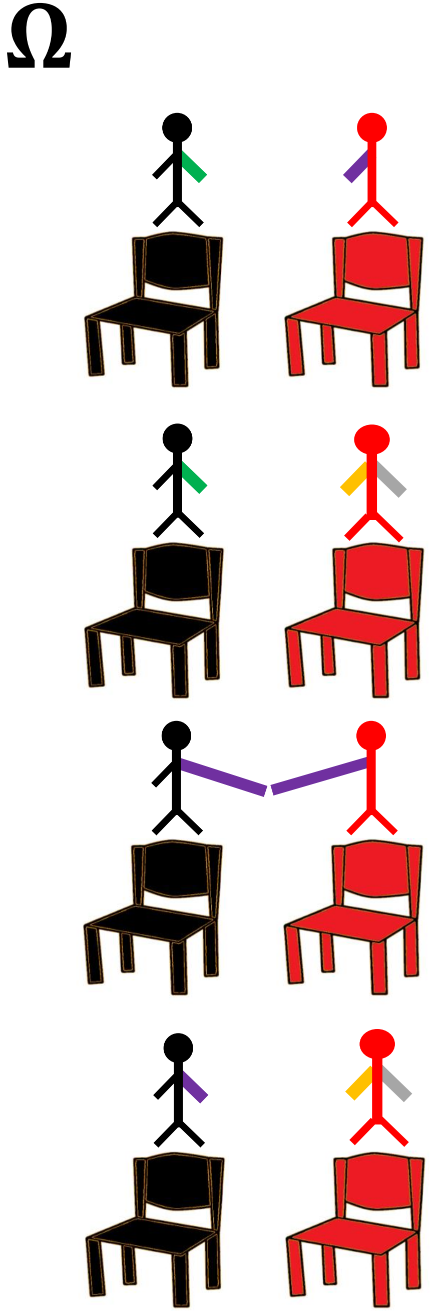


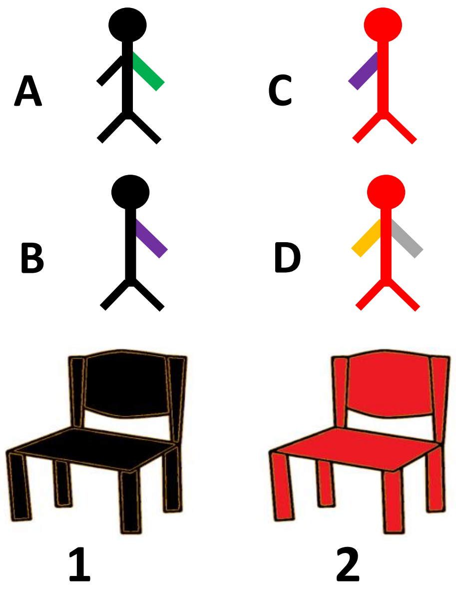
- sit(A) = sit(B) = 3
- sit(C) = sit(D) = 3
- handshake(B, C) = 1
- sit_convincing_cost = -1



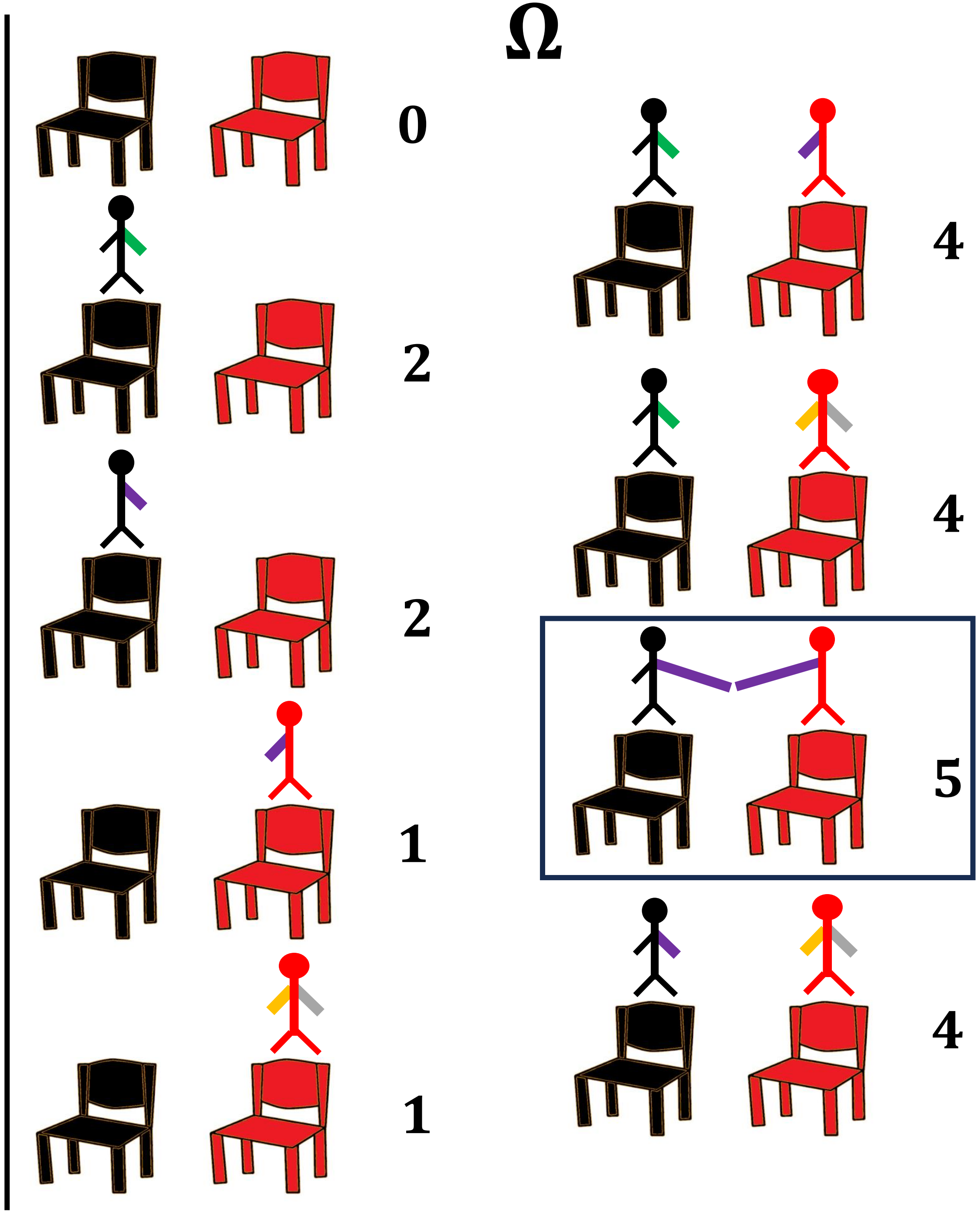
- sit(A) = sit(B) = 3
- sit(C) = sit(D) = 3
- handshake(B, C) = 1
- sit_convincing_cost = -1

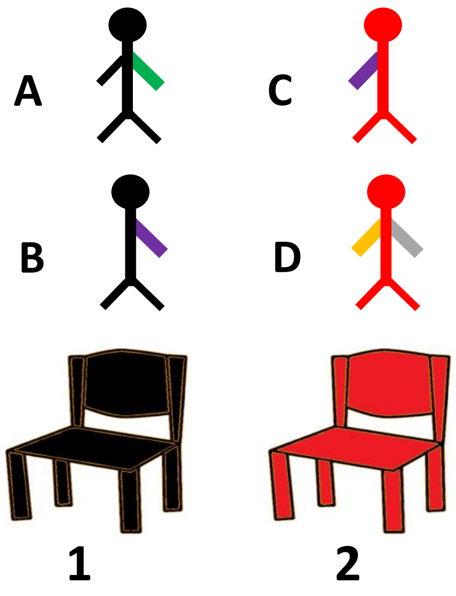




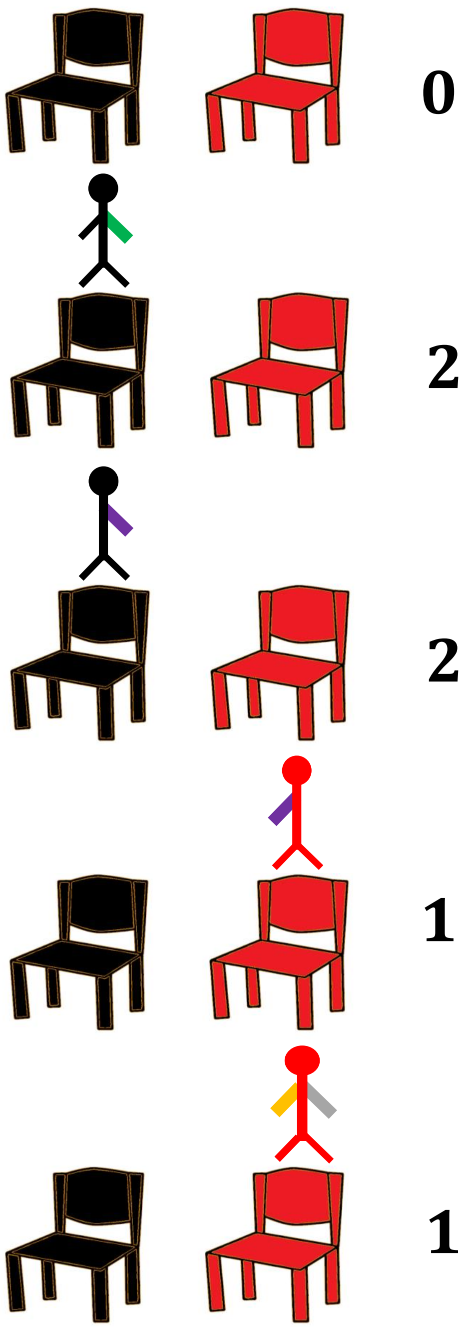


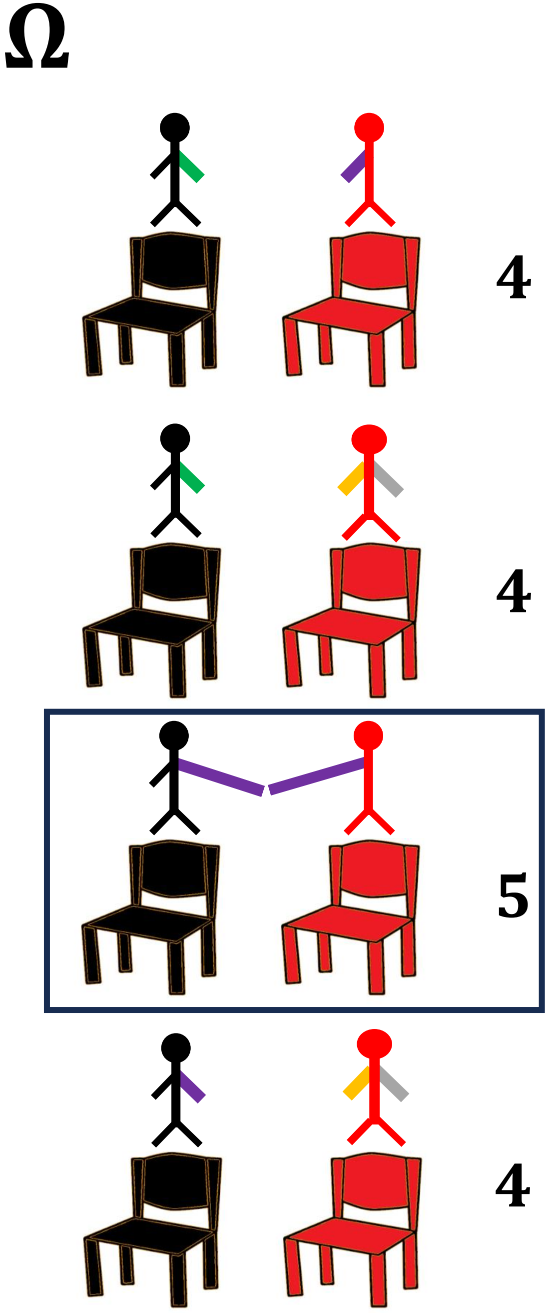
- $\text{sit}(A) = \text{sit}(B) = 3$
- $\text{sit}(C) = \text{sit}(D) = 3$
- handshake(B, C) = 1
- sit_convincing_cost = -1





- sit(A) = sit(B) = 3
- sit(C) = sit(D) = 3
- handshake(B, C) = 1
- sit_convincing_cost = -1





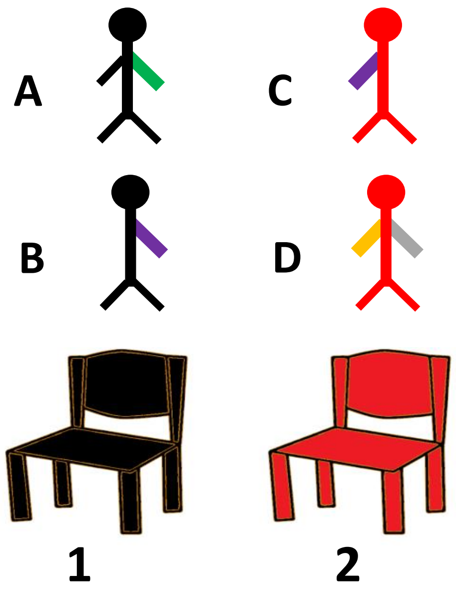
The Maximum Energy M

$$M := \max_{X \in \Omega} \{E(X)\}$$

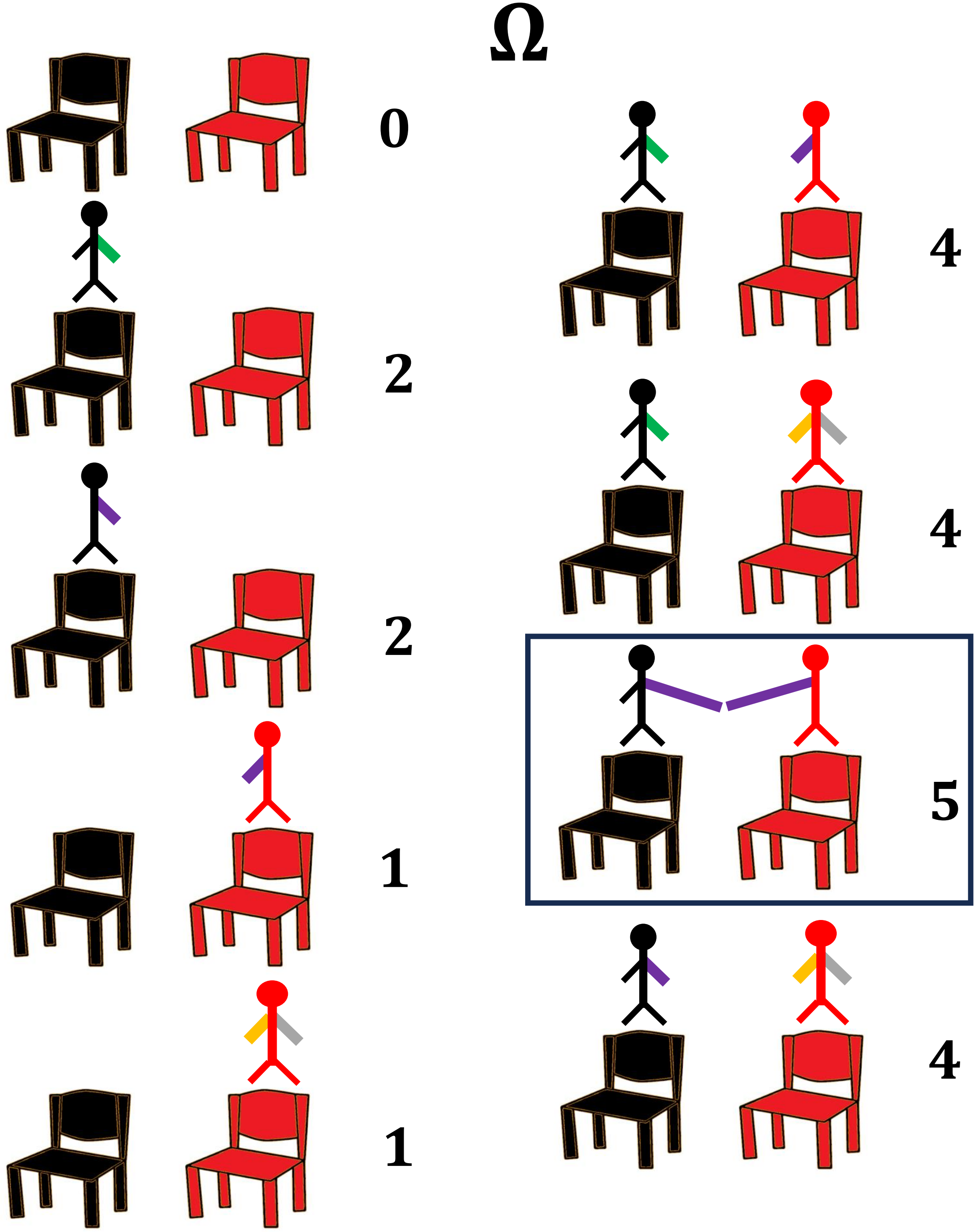
The Partition function Q

$$Q := \sum_{X \in \Omega} e^{\frac{E(X)}{c}}$$

Where c is some specific constant



- sit(A) = sit(B) = 3
- sit(C) = sit(D) = 3
- handshake(B, C) = 1
- sit_convincing_cost = -1



The Maximum Energy M

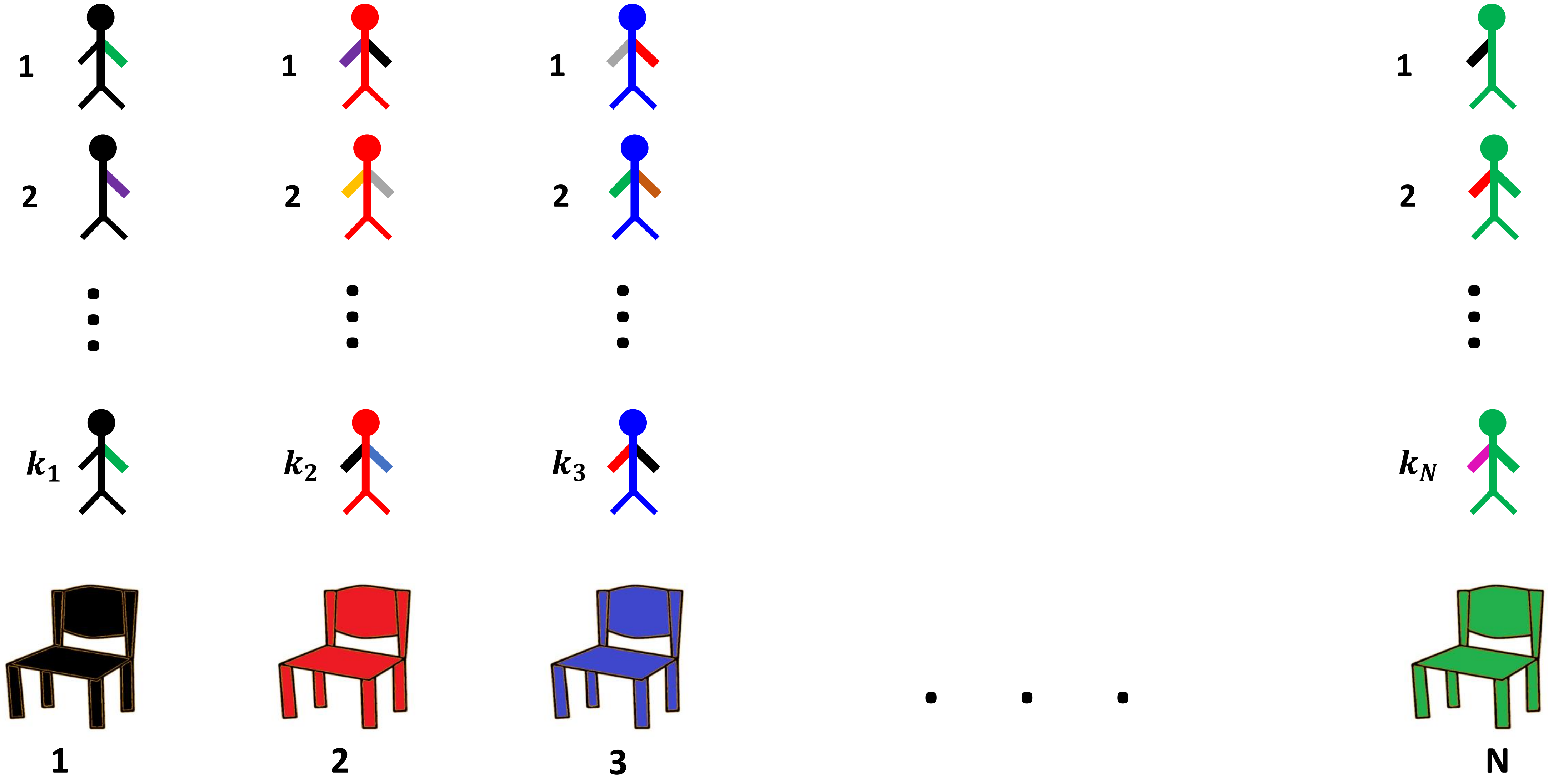
$$M := \max_{X\in\Omega}\{E(X)\}$$

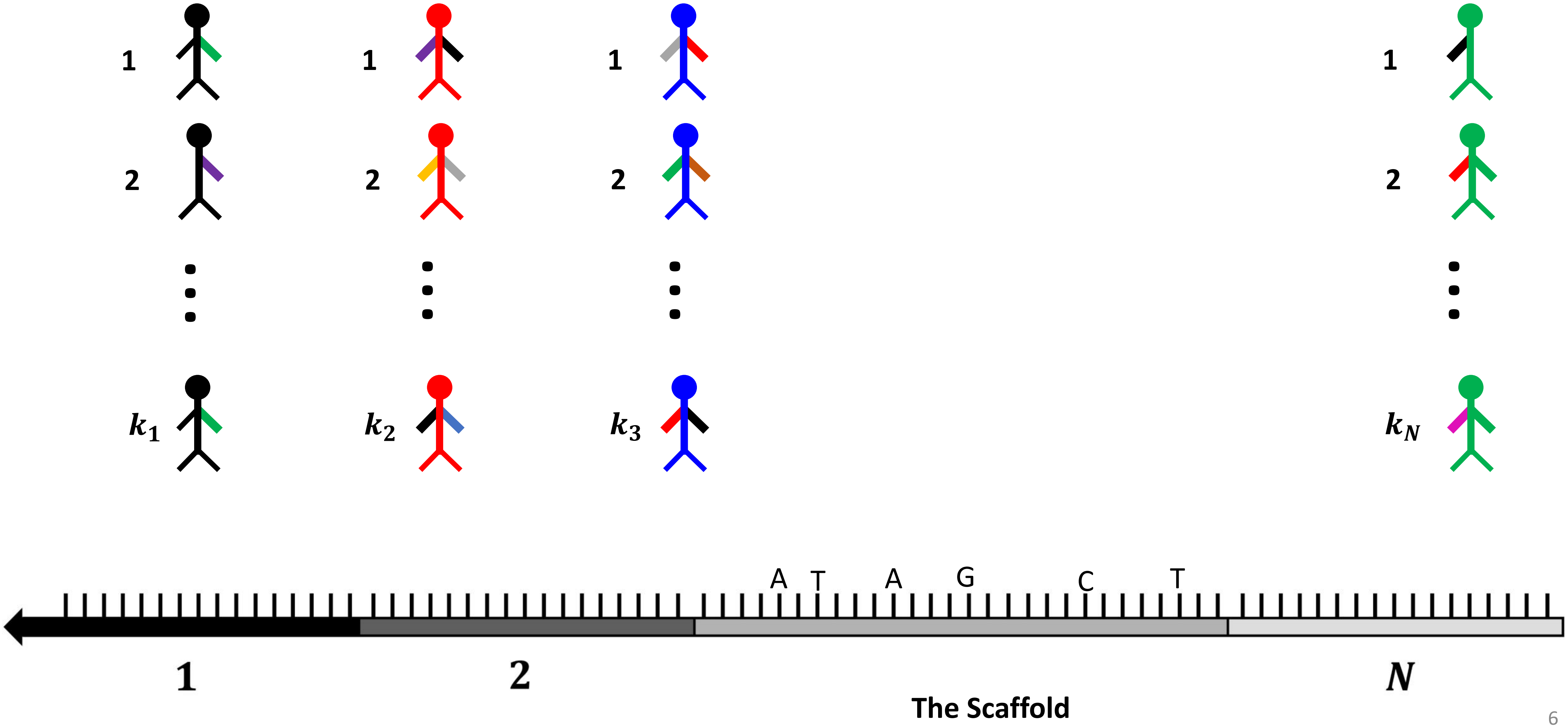
The Partition function Q

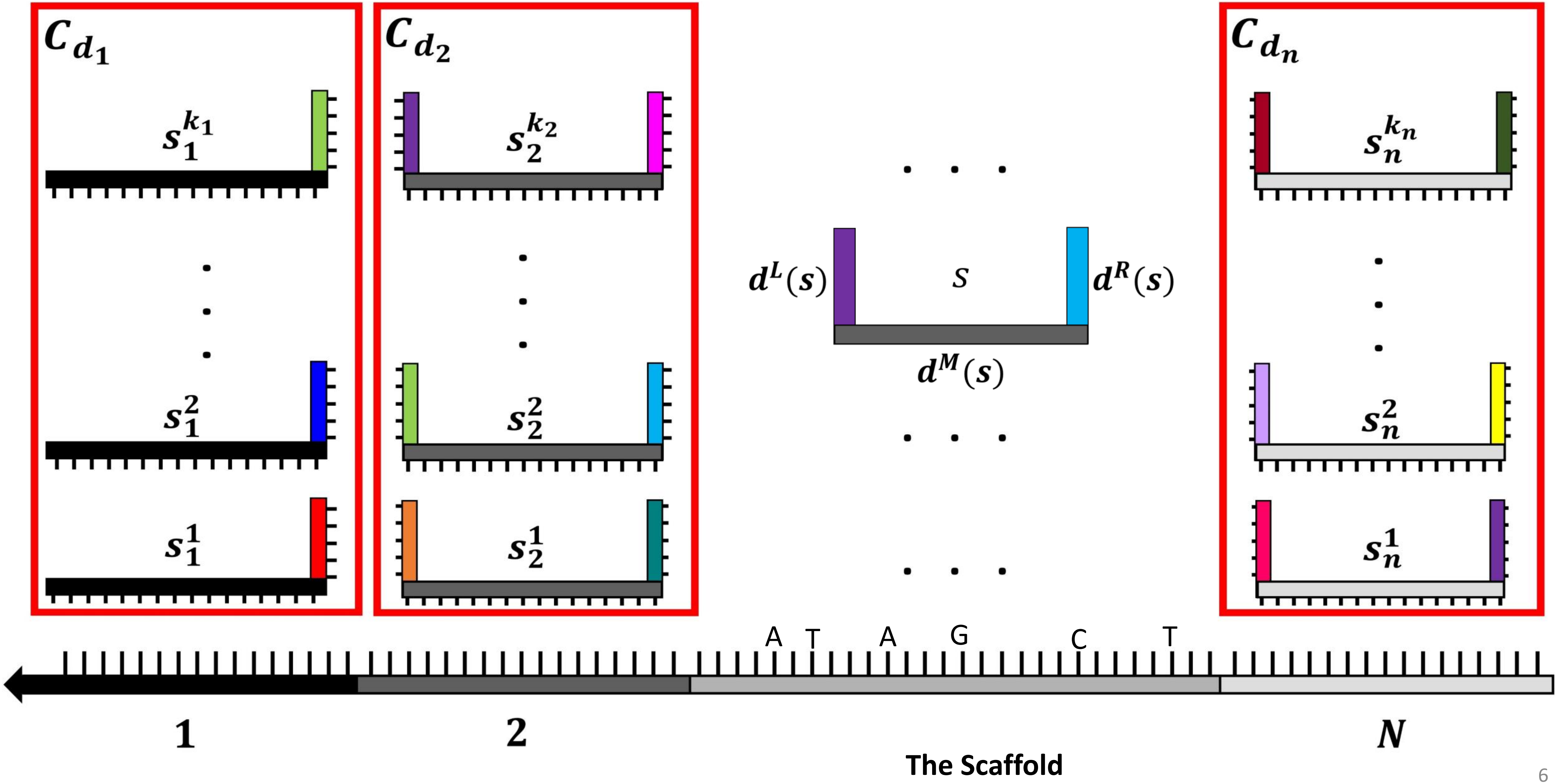
$$Q := \sum_{X\in\Omega} e^{\frac{E(X)}{c}}$$

Where c is some specific constant

Why are we interested in this ?

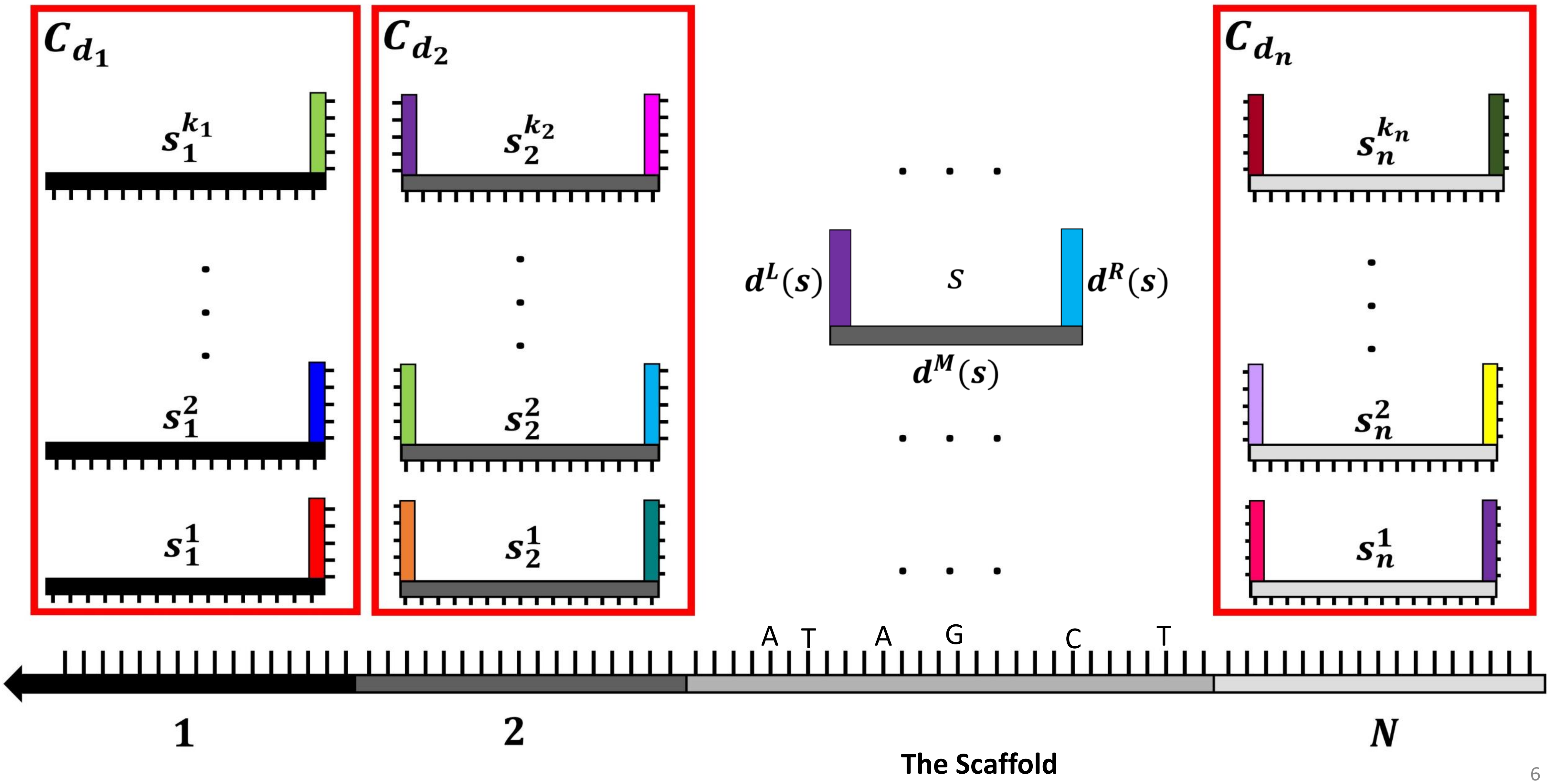

1
2
k_1
1
2
k_2
1
2
k_3
1
2
k_N
1
2
3
N


1
2
k_1
1
2
k_2
1
2
k_3
1
2
k_N
A T A G C T
1
2
N
The Scaffold



C_{d_1}
$s_1^{k_1}$
s_1^2
s_1^1
C_{d_2}
$s_2^{k_2}$
s_2^2
s_2^1
$d^L(s)$
s
$d^R(s)$
$d^M(s)$
C_{d_n}
$s_n^{k_n}$
s_n^2
s_n^1
A T A G C T
1
2
N
The Scaffold

This is 1D Scaffolded DNA Computer



1D Scaffolded DNA Computer Energy model and thermodynamic features

For any configuration X of size l :

DNA $$\Delta G^{\mathcal{S}}(X) = \sum_{s \in X} \Delta G(d^M(s)) + l \cdot \Delta G^{\text{assoc}} + \sum_{s_i, s_{i+1} \in X} \Delta G(d^R(s_i), d^L(s_{i+1})).$$

Chair $$\mathrm{E}(X) = \sum_{p \in X} \mathrm{sit(p)} + l * sitcost + \sum_{p_i, p_{i+1} \in X} \mathrm{handshake}(p_i, p_{i+1}).$$

$$\mathrm{MFE}^{\mathcal{S}} = \min_{X \in \Omega^{\mathcal{S}}} \{\Delta G^{\mathcal{S}}(X)\}$$

$$Q^{\mathcal{S}} = \sum_{X \in \Omega^{\mathcal{S}}} e^{-\Delta G^{\mathcal{S}}(X)/kT}$$

NP-Hard in base level case

T is the temperature in (kelvin) and k is Boltzmann constant.

- Intuitively, the MFE is the energy of the most favored configuration(s) of the system.
- The partition function is a Boltzmann-weighted sum of the energies of every configuration of the system and typically used as a normalization factor to calculate the probability of any configuration X at equilibrium: $p(X) = (e^{-\Delta G^{\mathcal{S}}(X)/kT})/Q^{\mathcal{S}}$.

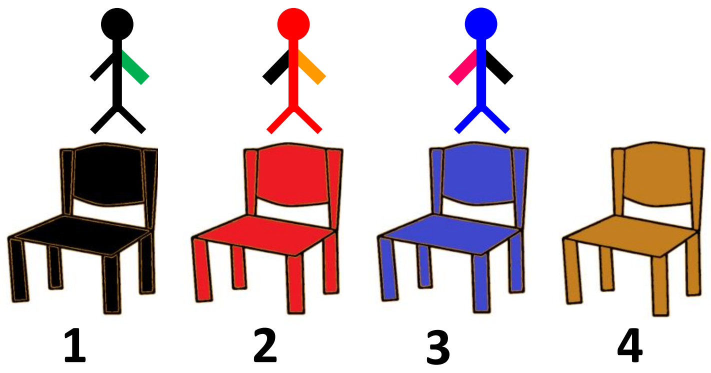
What is interesting in Scaffolded DNA Computer (SDC) ?

What is interesting in Scaffolded DNA Computer (SDC) ?

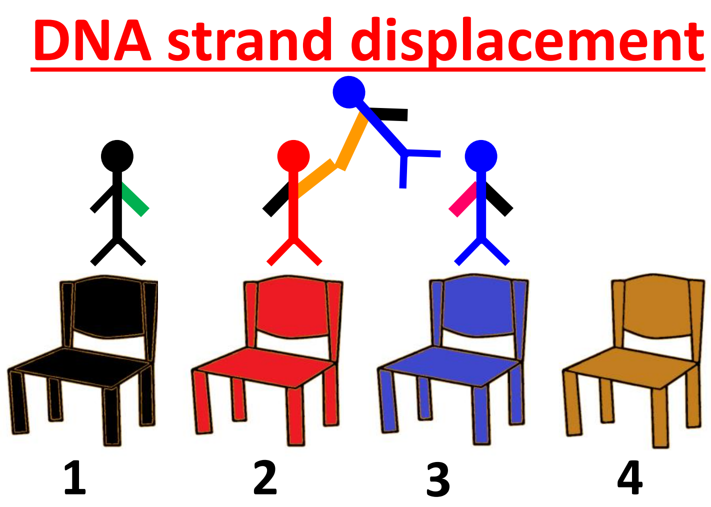
<u>**DNA strand displacement**</u>

What is interesting in Scaffolded DNA Computer (SDC) ?

DNA strand displacement

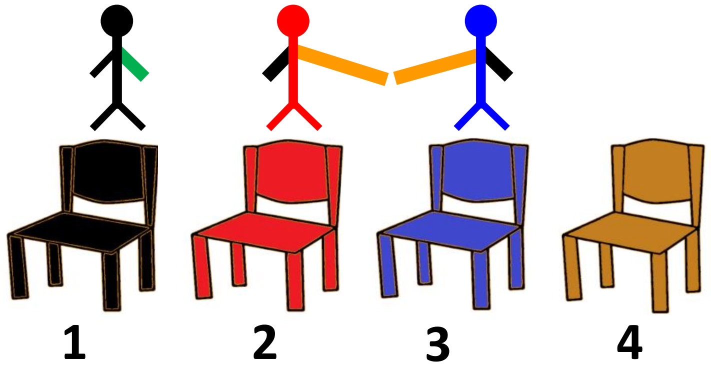


What is interesting in Scaffolded DNA Computer (SDC) ?



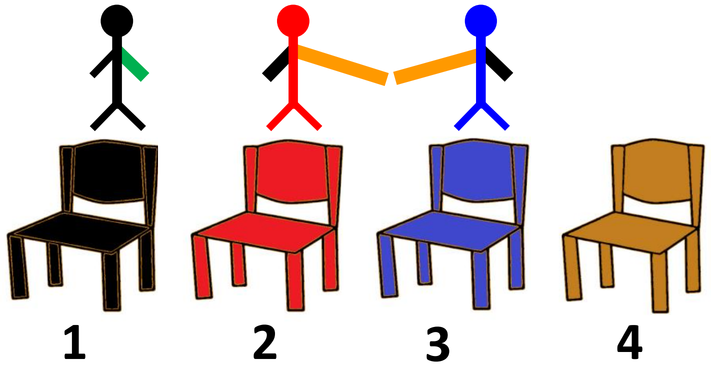
What is interesting in Scaffolded DNA Computer (SDC) ?

DNA strand displacement

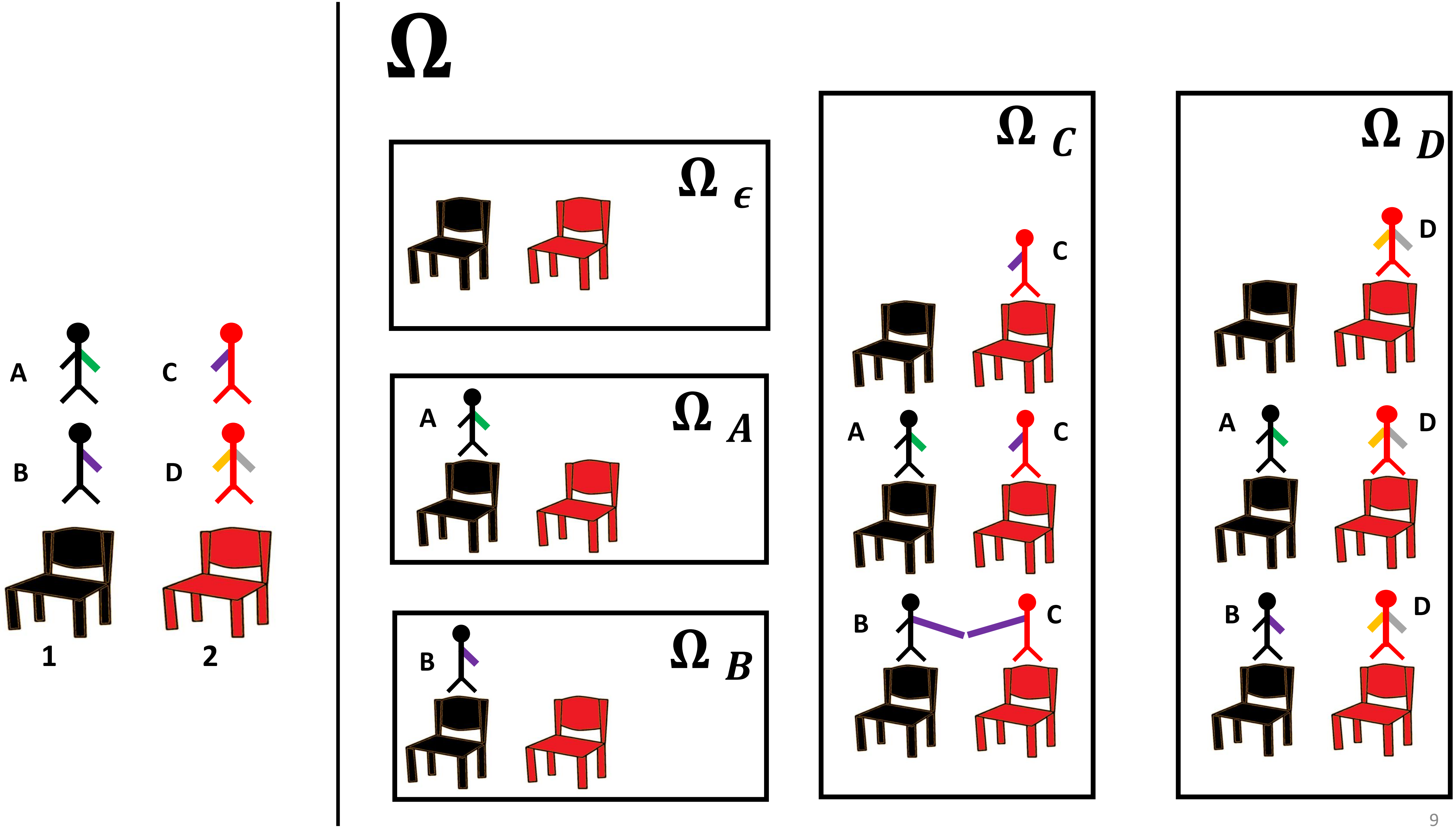


What is interesting in Scaffolded DNA Computer (SDC) ?

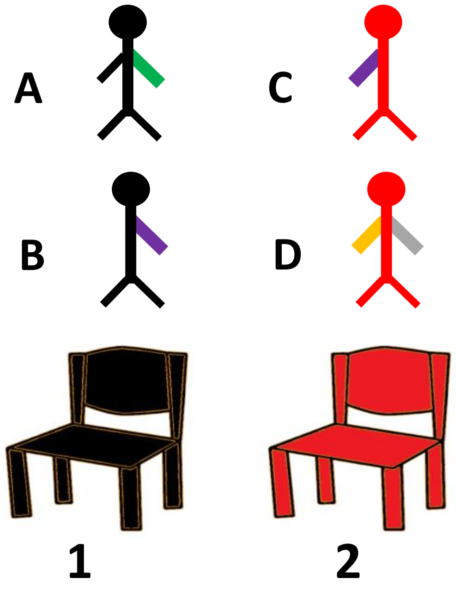
DNA strand displacement

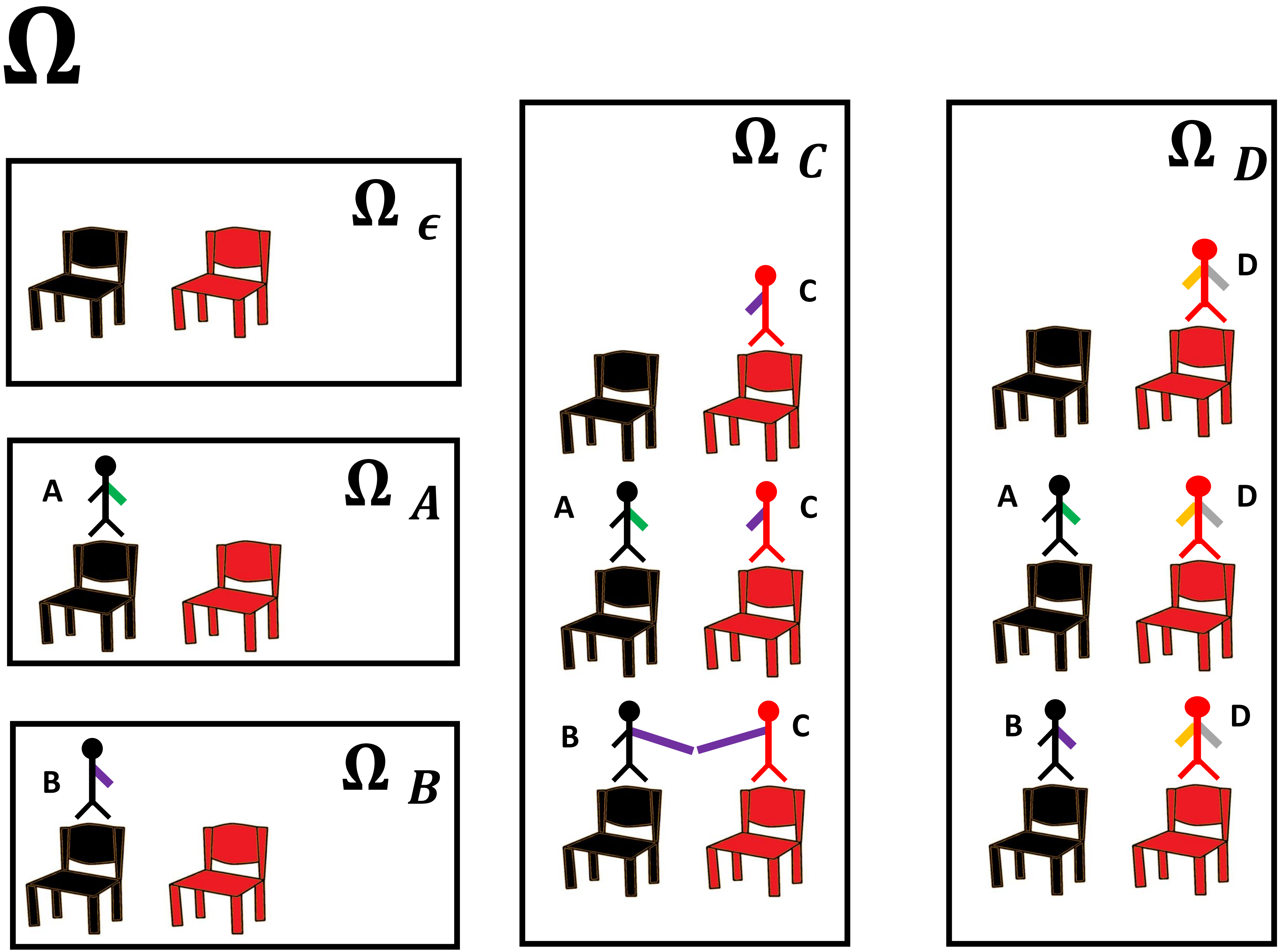


- computationally expressive:
  - simulate finite state machines in 1D.
  - Boolean circuits in 2D.
- amenable to DNA storage applications.
- The SDC has been implemented experimentally in 1D, showcasing a total of 10 programs that solve problems such as
  - Bit-copy
  - Addition of two 4-bit numbers
  - Parity of an 8-bit input (is the number of 1s odd?)
  - Graph Reachability (is there a path in an input graph from a source node *s* to a target node *t*)
- Initial designs and results in 2D include Addition of two 7-bit numbers and a simple Bit-Copy program.

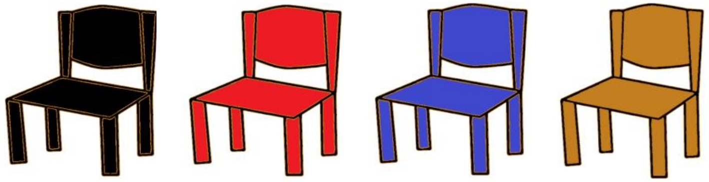

A
C
B
D
1
2
Ω
Ω ϵ
Ω A
Ω B
Ω C
Ω D

Is there recursive way to build these classes from themselves?

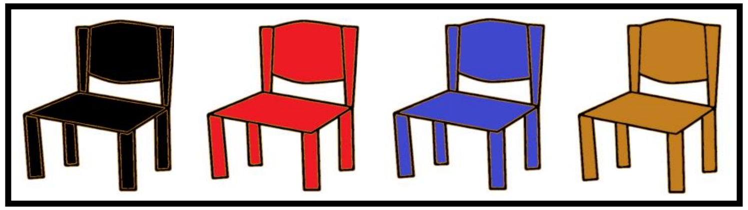




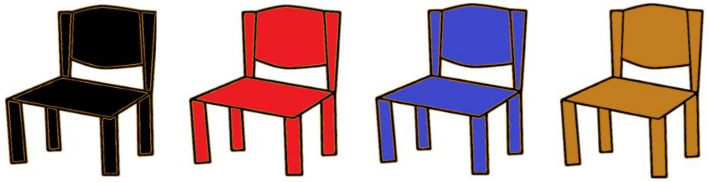
0



0

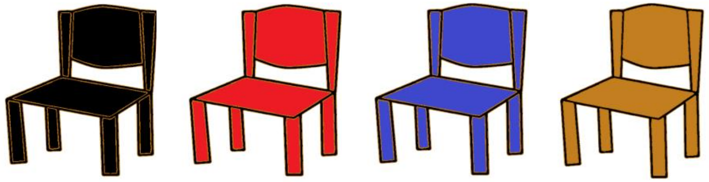


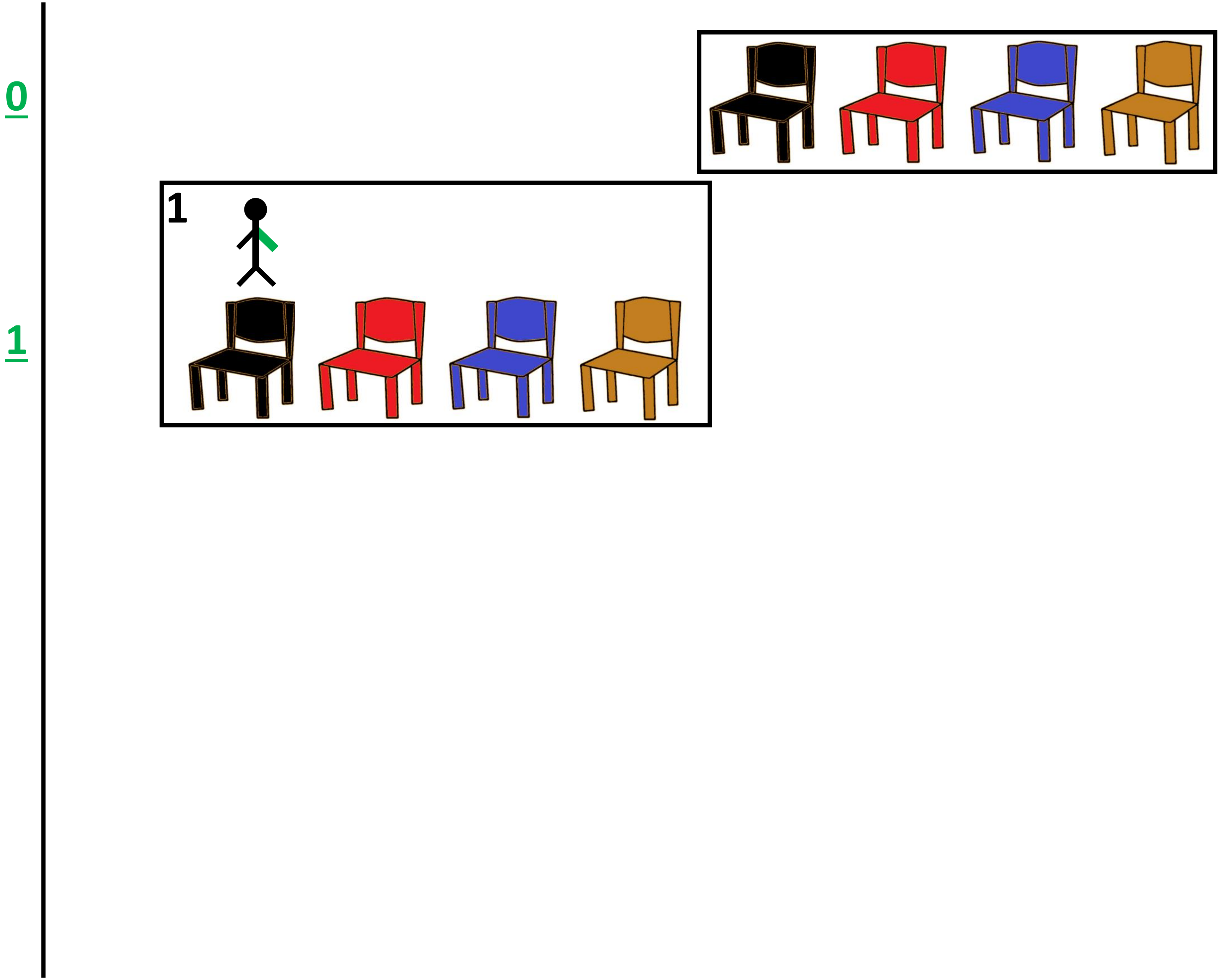
0

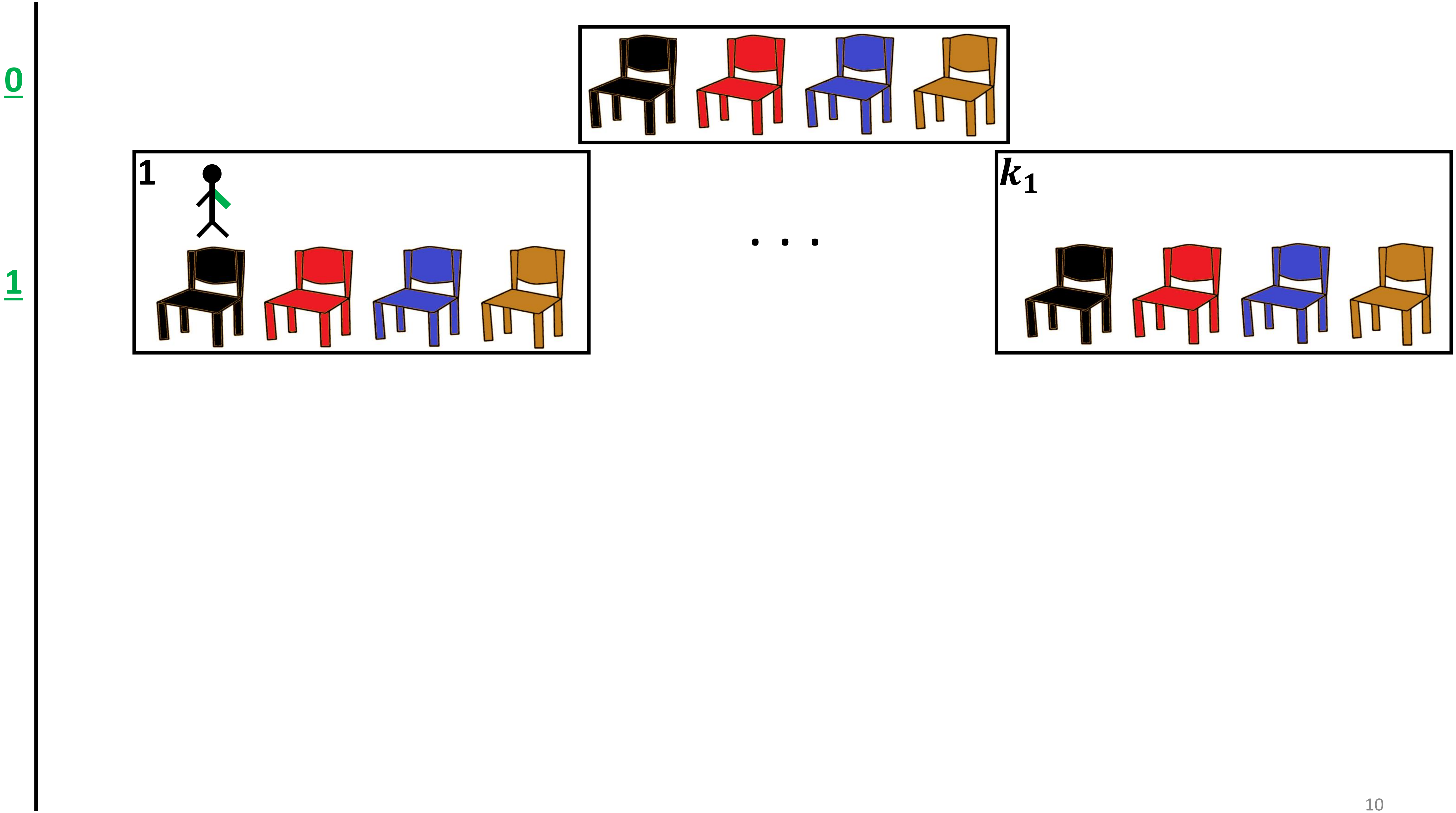


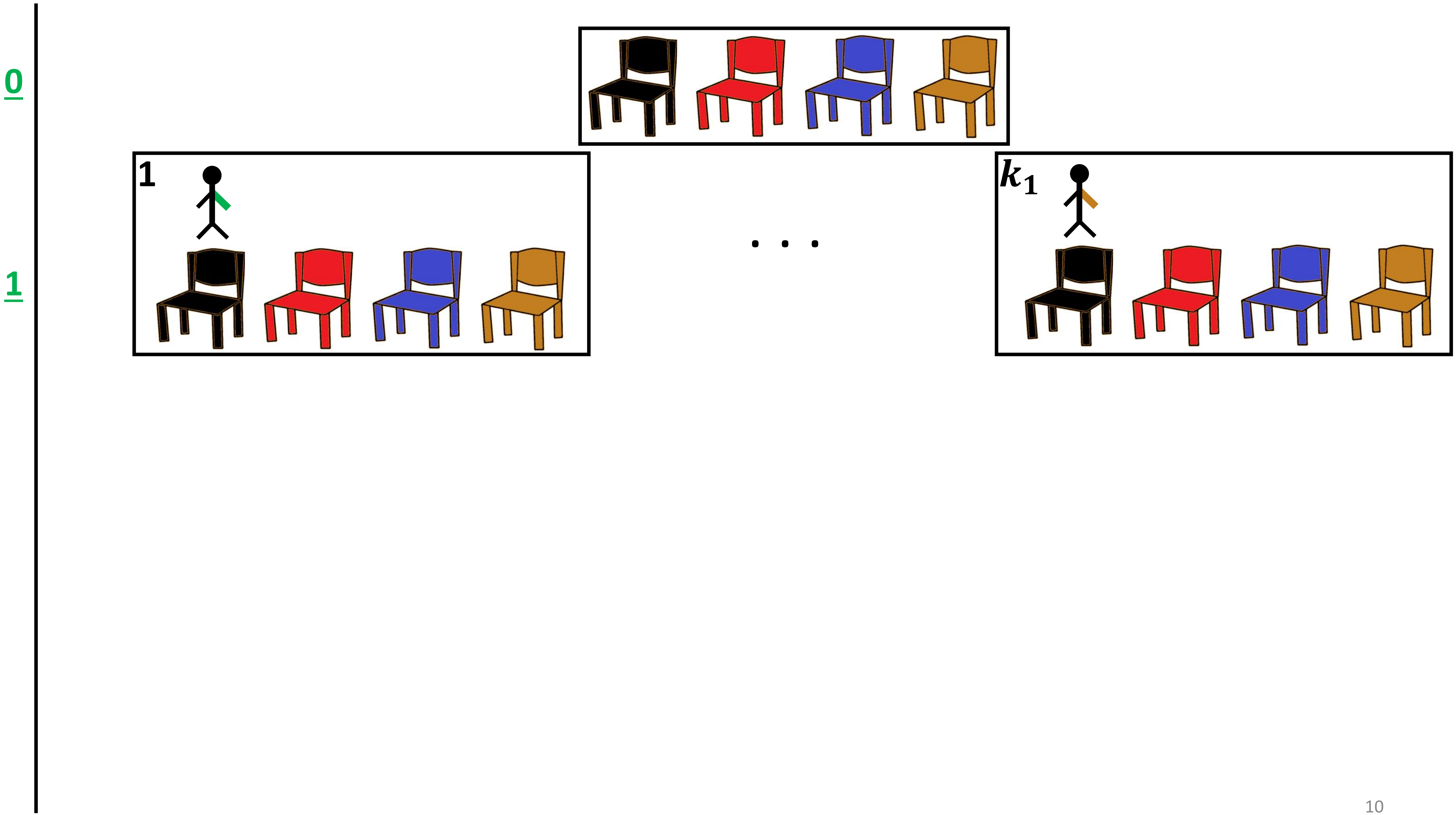
1

1



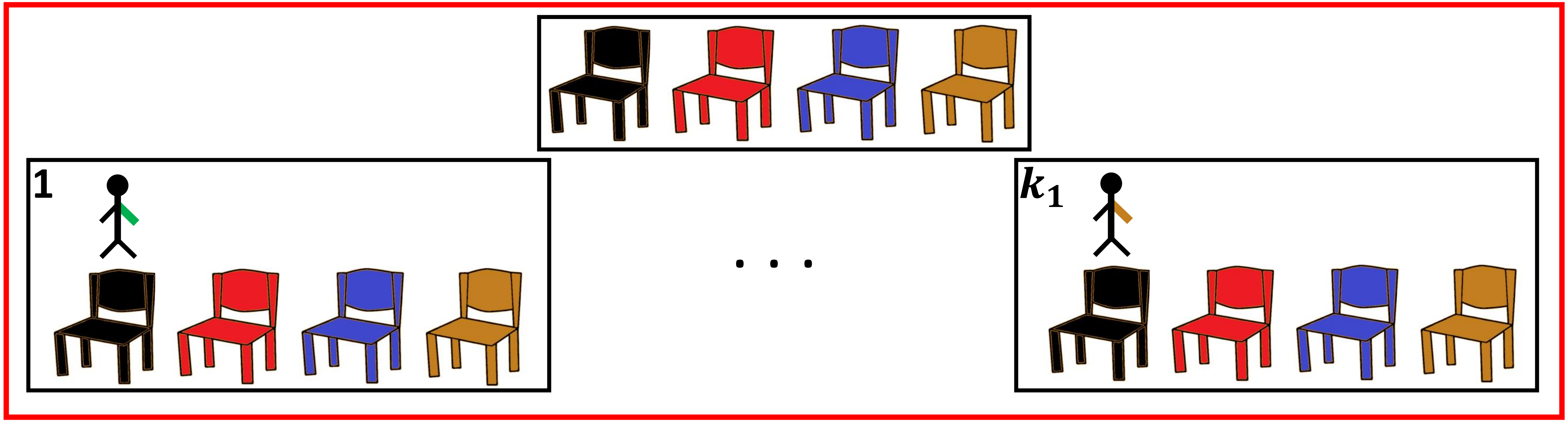

0
1
1


0
1
1
k_1
. . .


0
1
1
k_1
. . .

0

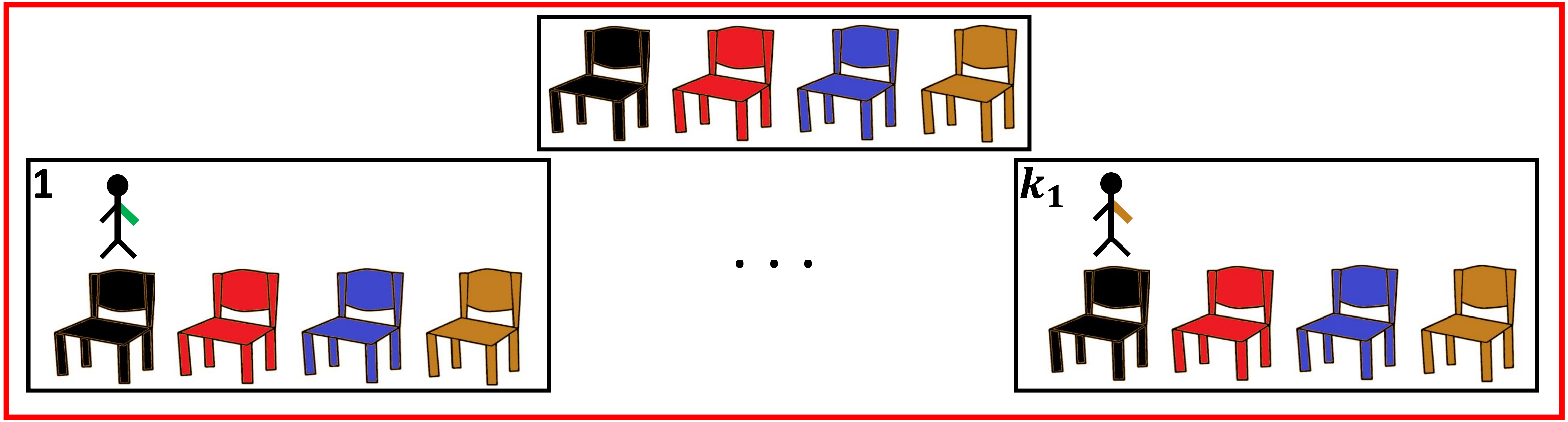
1

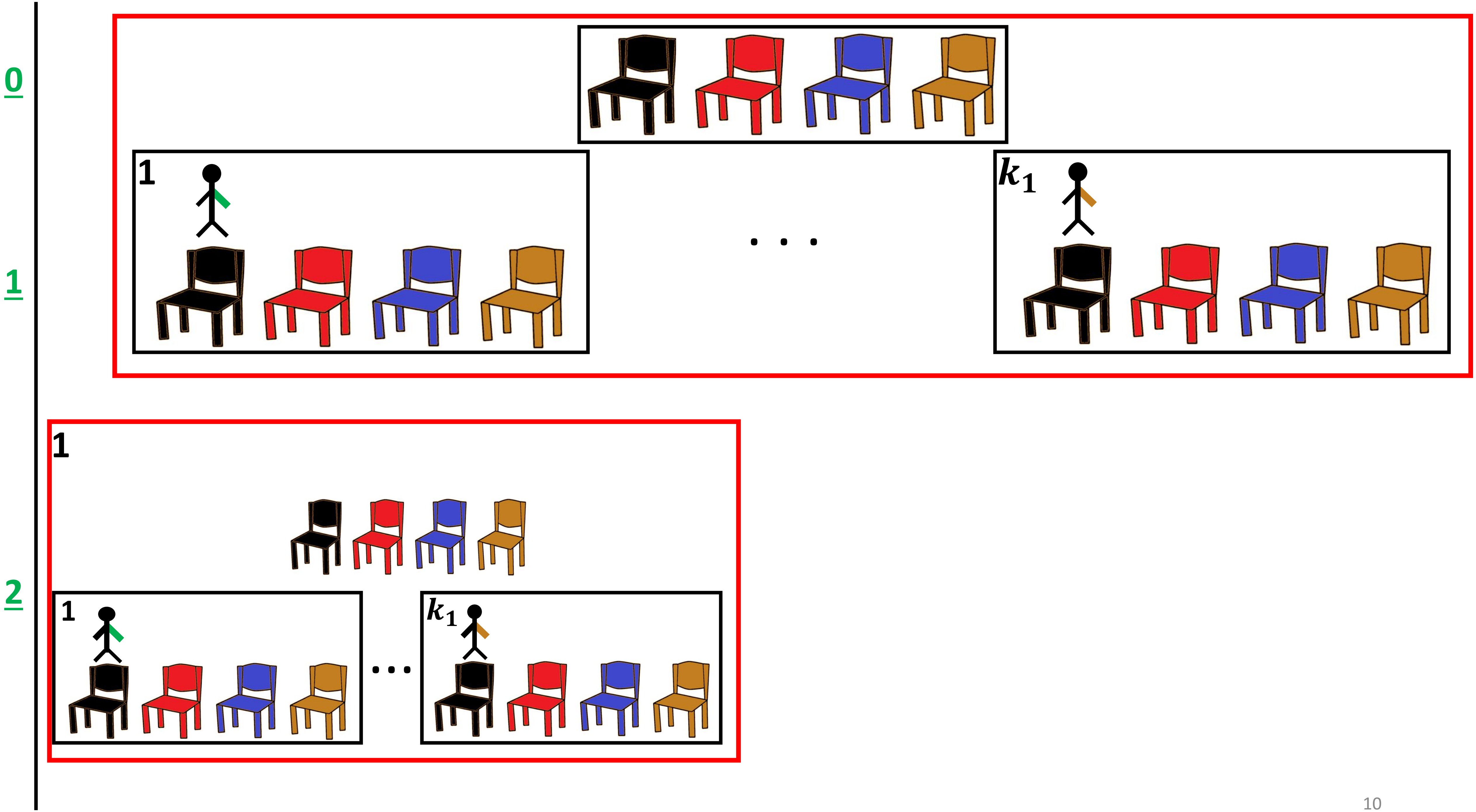


0

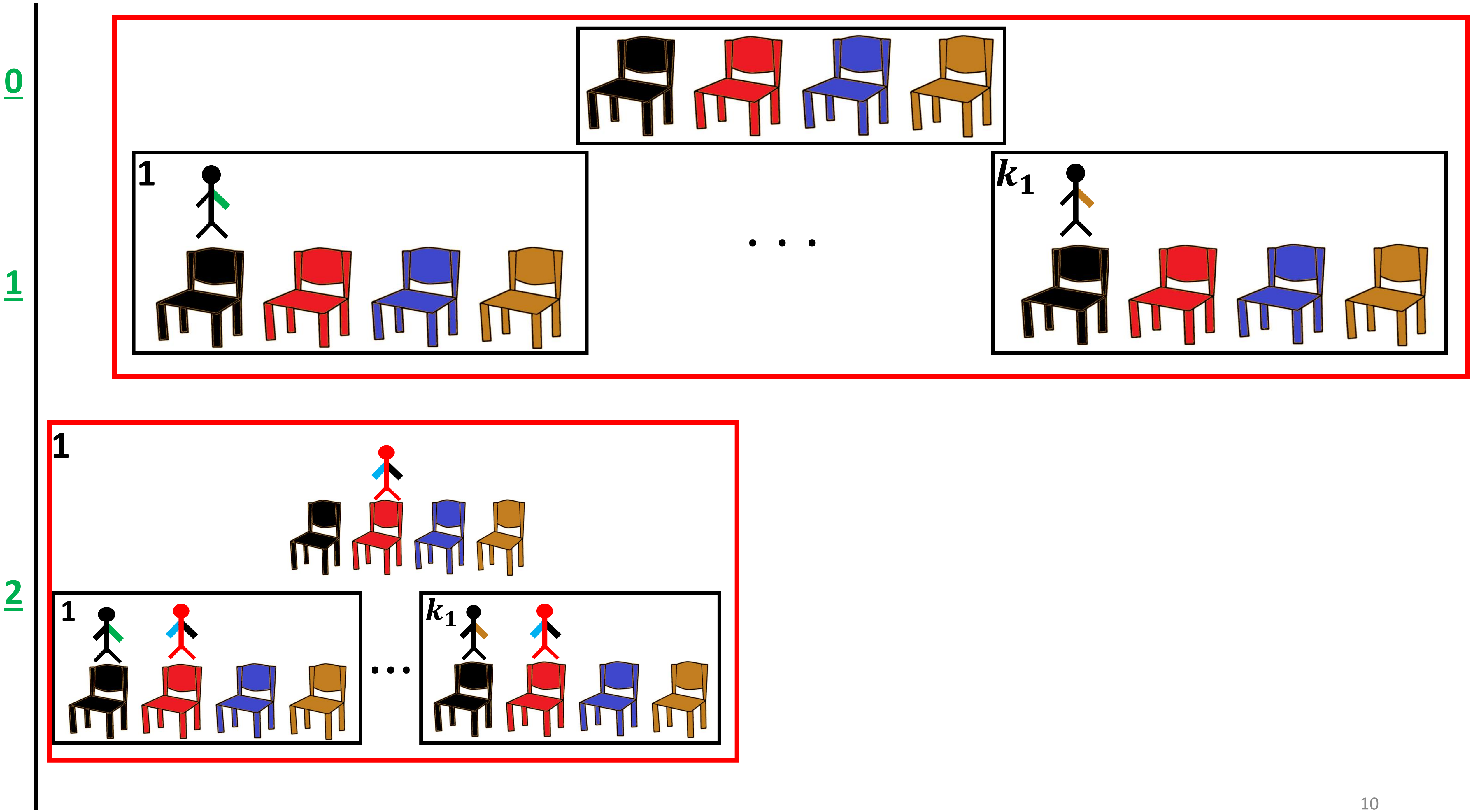
1

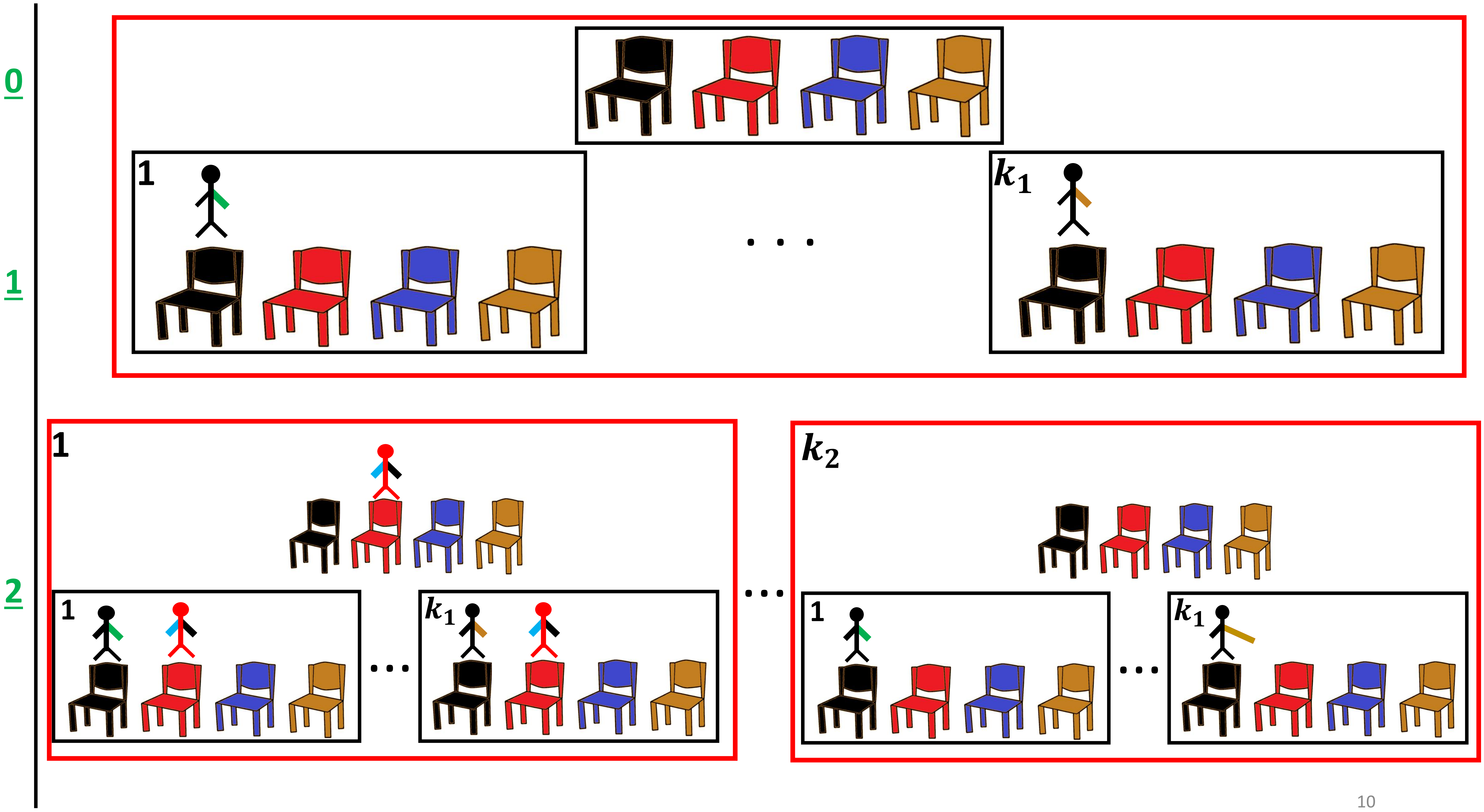
2



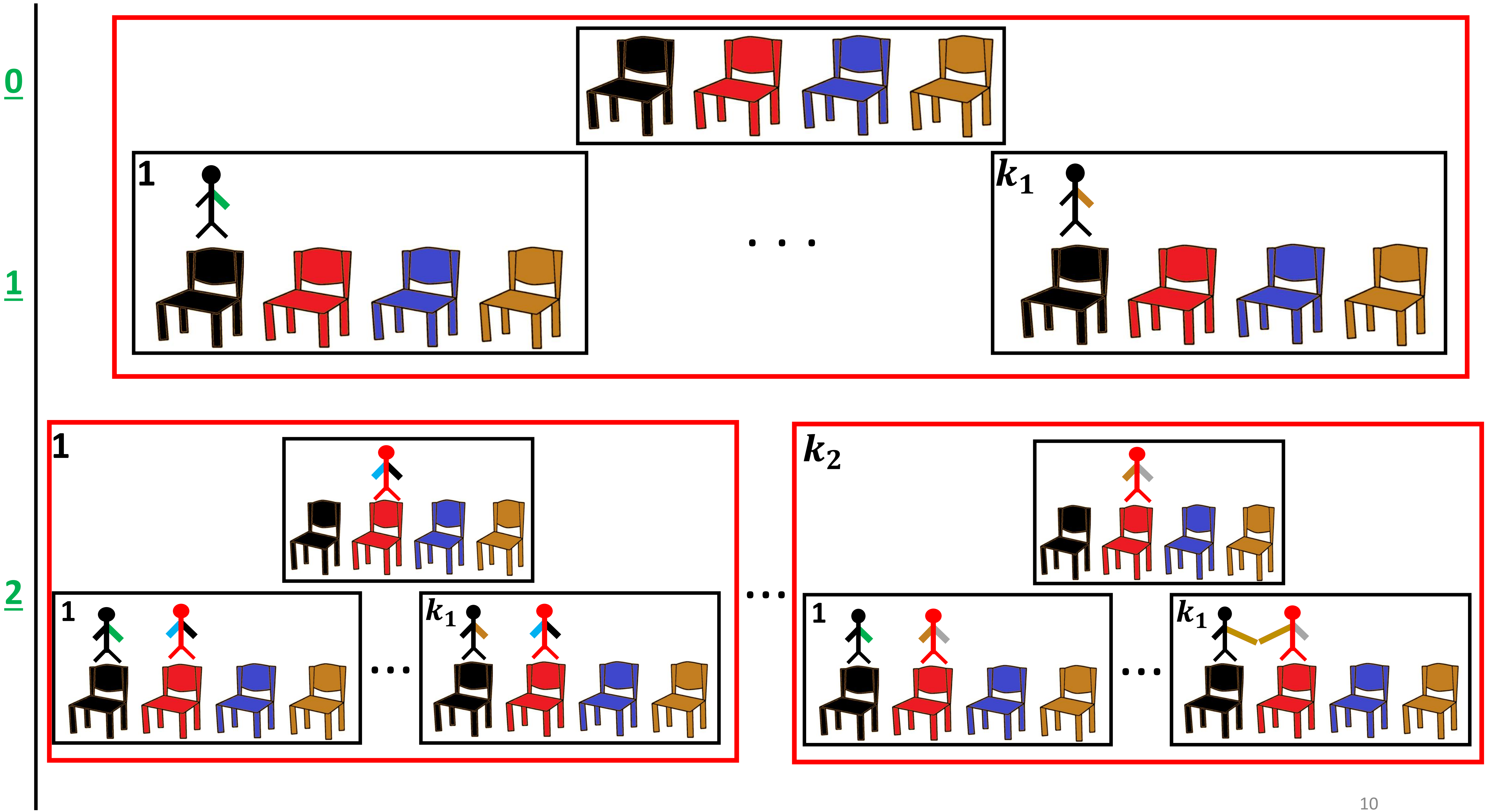


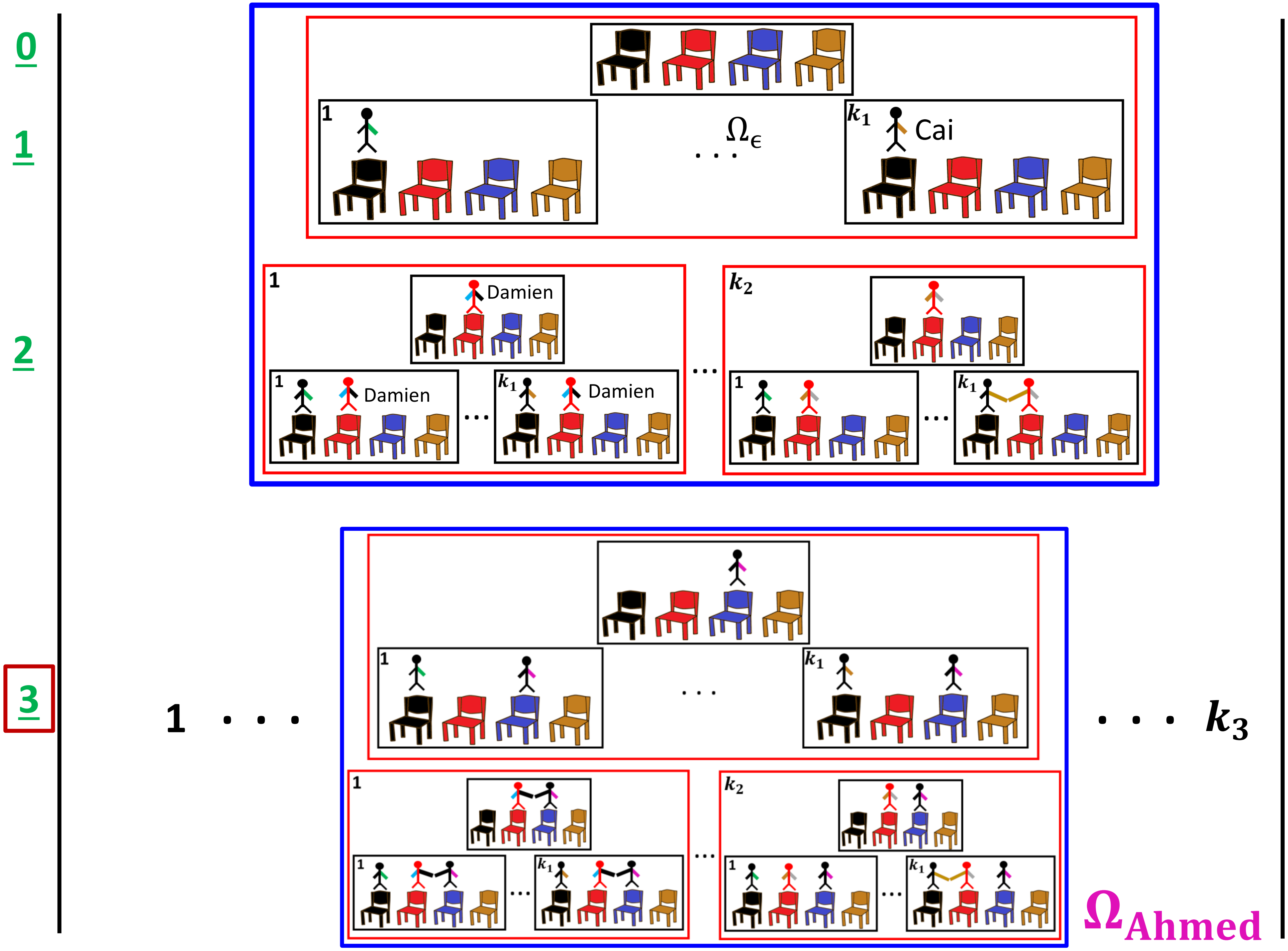
0
1
2
1
k_1
. . .
1
1
k_1
...


0
1
1
k_1
. . .
2
1
1
...
k_1

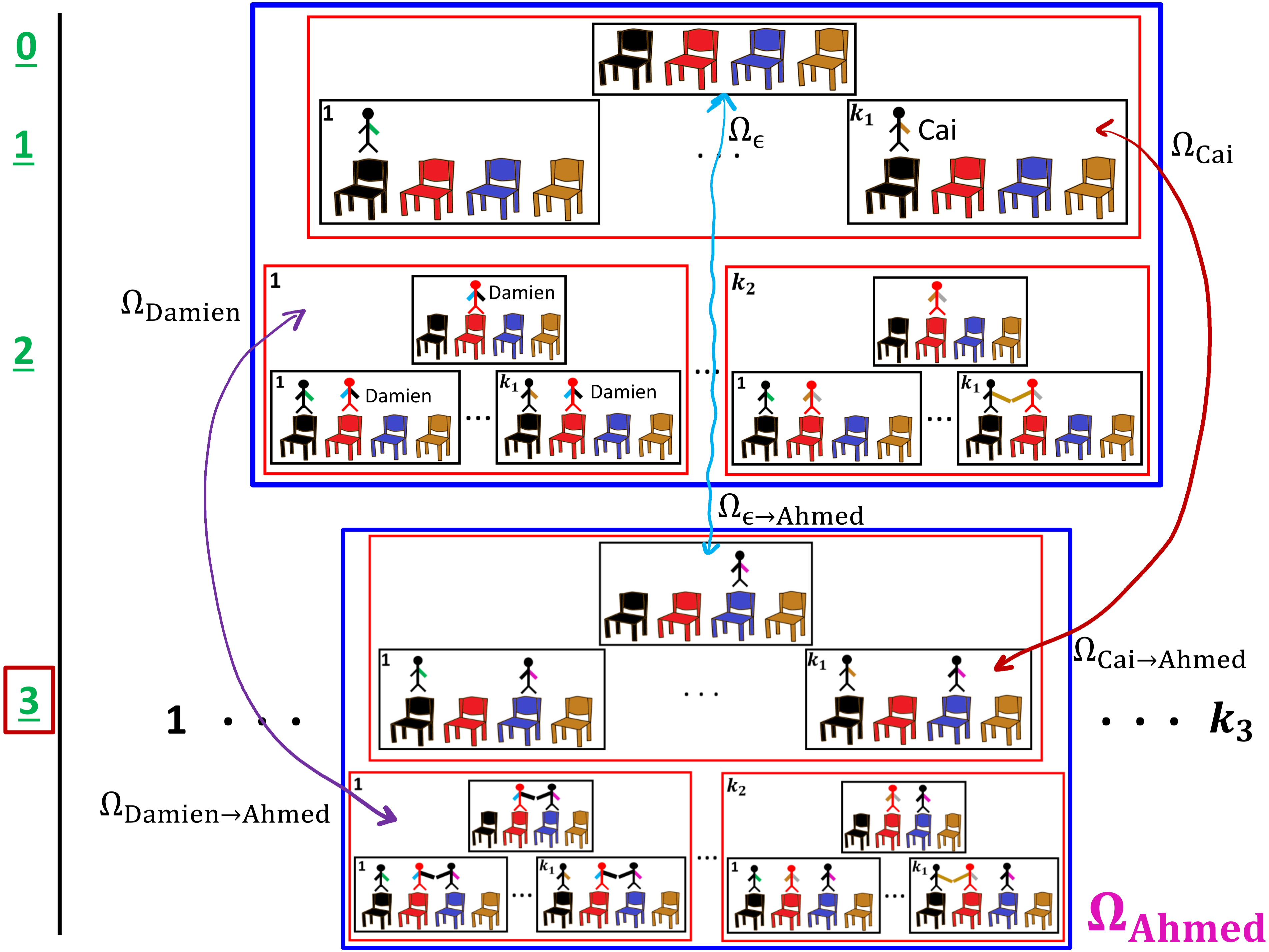


0
1
2
1
k_1
. . .
1
k_2
1
k_1
1
k_1


0
1
2
1
k_1
. . .
1
k_2
1
k_1
1
k_1
...
...
...



0
1
2
3
1
k1
Cai
Ωϵ
Damien
k2
1 . . .
. . . k3
ΩAhmed



Layer = chair

There is a 1-1 correspondence between each **class** of higher layers and a **sub-class** of any class in current layer.

$$\Omega_p \updownarrow \Omega_{p\to p'}$$

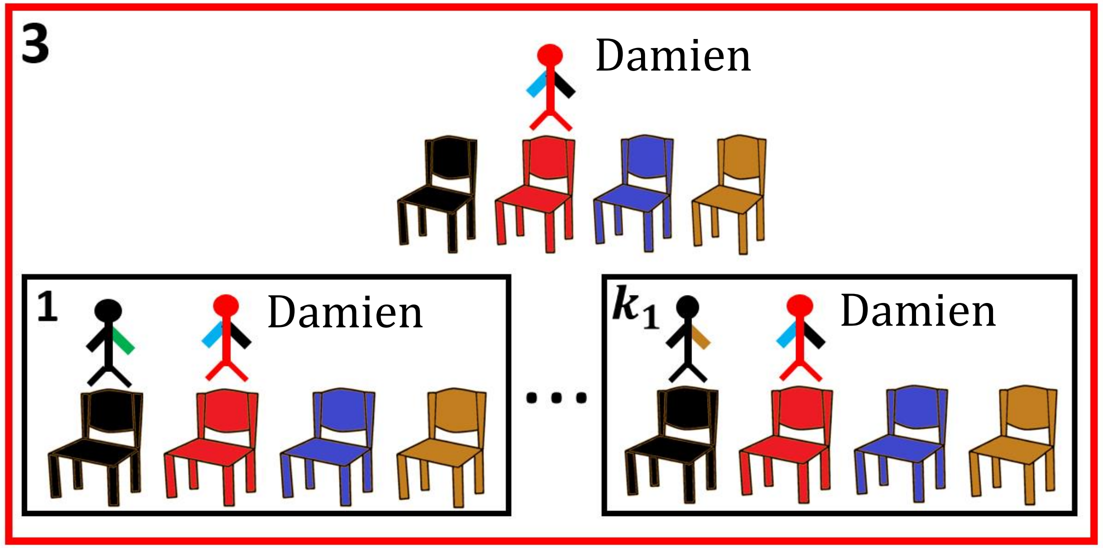
a **sub-class** of the class $\Omega_{p'}$, where p is the nearest neighbour to p'.

Can we use this inductive construction process to propagate information through this hierarchy ?

Can we use this inductive construction process to propagate information through this hierarchy ?

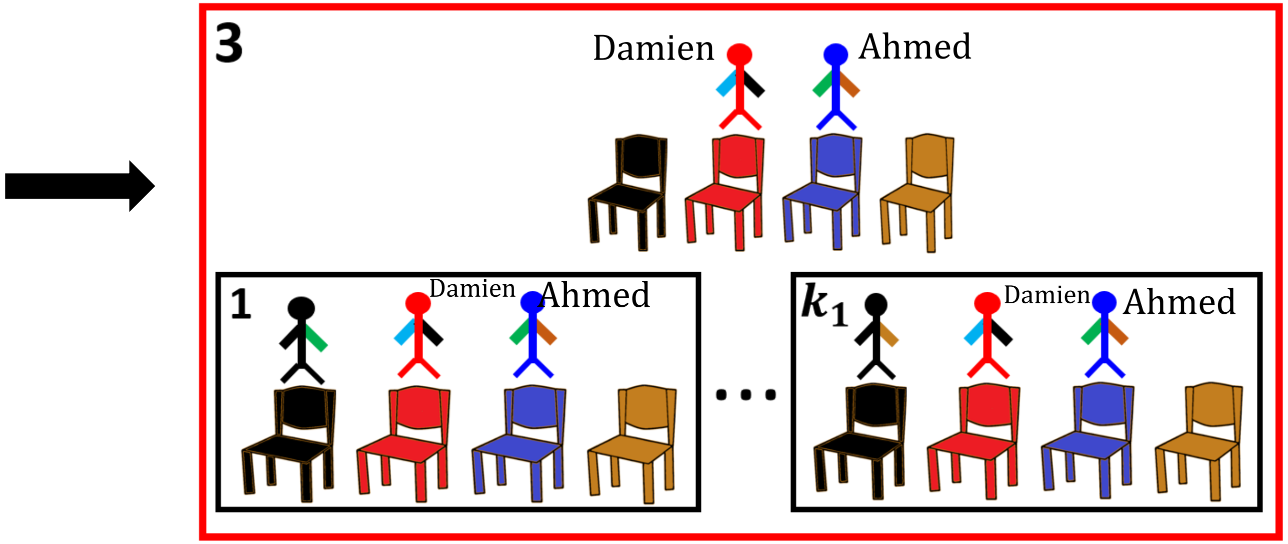
1 - Propagate information from exactly the previous layer (Direct neighbour handshaking possibility).

Ω_{Damien}



Information given: $Q(\Omega_{\boldsymbol{Damien}})$

$\Omega_{\text{Damien}\to\text{Ahmed}}$

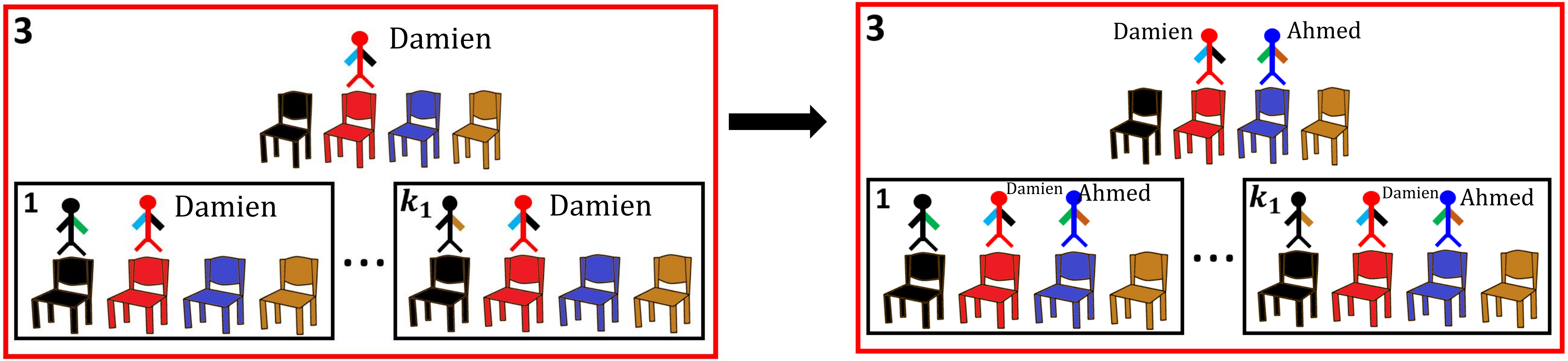


Information needed: $Q(\Omega_{\text{Damien}\to\text{Ahmed}})$

Can we use this inductive construction process to propagate information through this hierarchy ?

1 - Propagate information from exactly the previous layer (Direct neighbour handshaking possibility).

Ω_{Damien} $\Omega_{\text{Damien}\rightarrow\text{Ahmed}}$



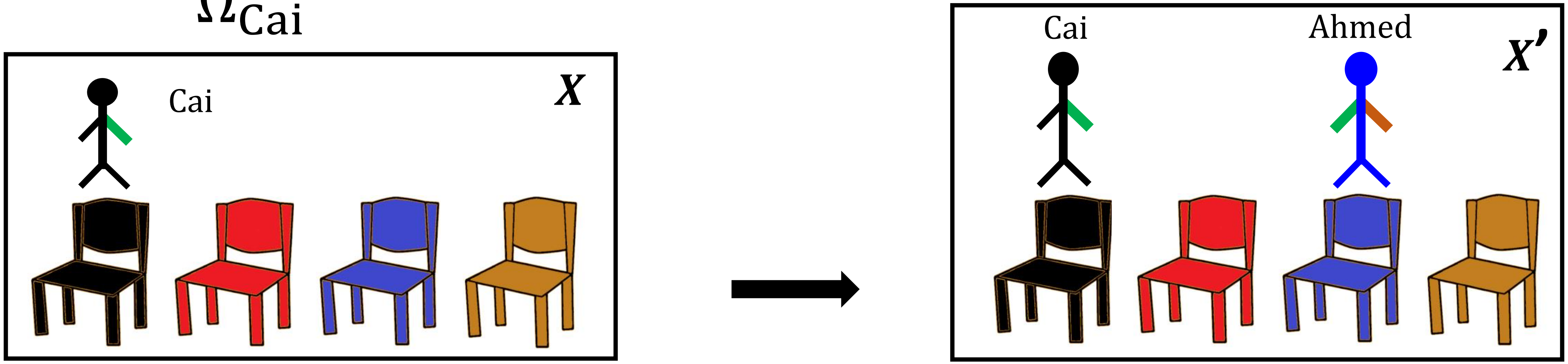
Information given: $Q(\Omega_{\boldsymbol{Damien}})$

Information needed: $Q(\Omega_{\text{Damien}\rightarrow\text{Ahmed}})$

$$Q(\Omega_{\text{Damien}\rightarrow\text{Ahmed}}) = \ldots = e^{\frac{\text{sit(Ahmed)} + \text{Sit\_Cost}}{c}} * e^{\frac{\text{handshake}(\boldsymbol{Damien},\text{Ahmed})}{c}} * Q(\Omega_{\boldsymbol{Damien}})$$

2 - Propagate information from the rest (No handshaking possibility).

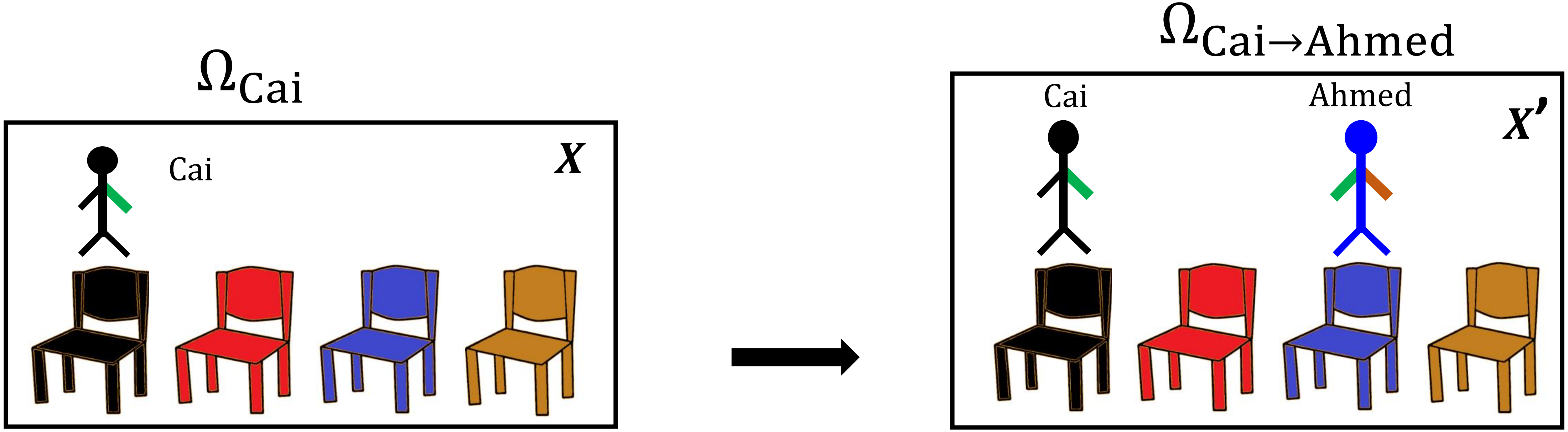
Ω_{Cai} $\Omega_{\text{Cai}\rightarrow\text{Ahmed}}$



Information given: $Q(\Omega_{Cai})$

Information needed: $Q(\Omega_{\text{Cai}\rightarrow\text{Ahmed}})$

2 - Propagate information from the rest (No handshaking possibility).



Information given: $Q(\Omega_{Cai})$

Information needed: $Q(\Omega_{\text{Cai}\rightarrow\text{Ahmed}})$

$$Q(\Omega_{\text{Cai}\rightarrow\text{Ahmed}}) = \ ... \ = e^{\frac{\text{sit(Ahmed)+sit\_cost}}{c}} * Q(\Omega_{\text{Cai}})$$

$$Q(\Omega_p) = e^{\frac{sit(p)+sit\_cost}{c}} * \left[\left(\sum_{\substack{p'\,is \\ direct\ neighbour}} e^{\frac{handshake(p',p)}{c}} * Q(\Omega_{\boldsymbol{p'}}) \right) + 1 + \sum_{\substack{p'\,is\ not \\ direct\ neighbour}} Q(\Omega_{\boldsymbol{p'}}) \right]$$

$$Q = 1 + \sum_{p} Q(\Omega_p)$$

$$Q(\Omega_p) = e^{\frac{sit(p)+sit\_cost}{c}} * \left[\left(\sum_{\substack{p' is \\ direct\ neighbour}} e^{\frac{handshake(p',p)}{c}} * Q(\Omega_{p'}) \right) + 1 + \sum_{\substack{p' is\ not \\ direct\ neighbour}} Q(\Omega_{p'}) \right]$$

▶ **Theorem** There is an $O(k^2N)$ time algorithm for the domain-level partition function for a 1D SDC of length N with $\leq k$ computation strands competing at each scaffold domain.

$$Z^S = 1 + \sum_{s \in T} Z_s^S \quad (6)$$

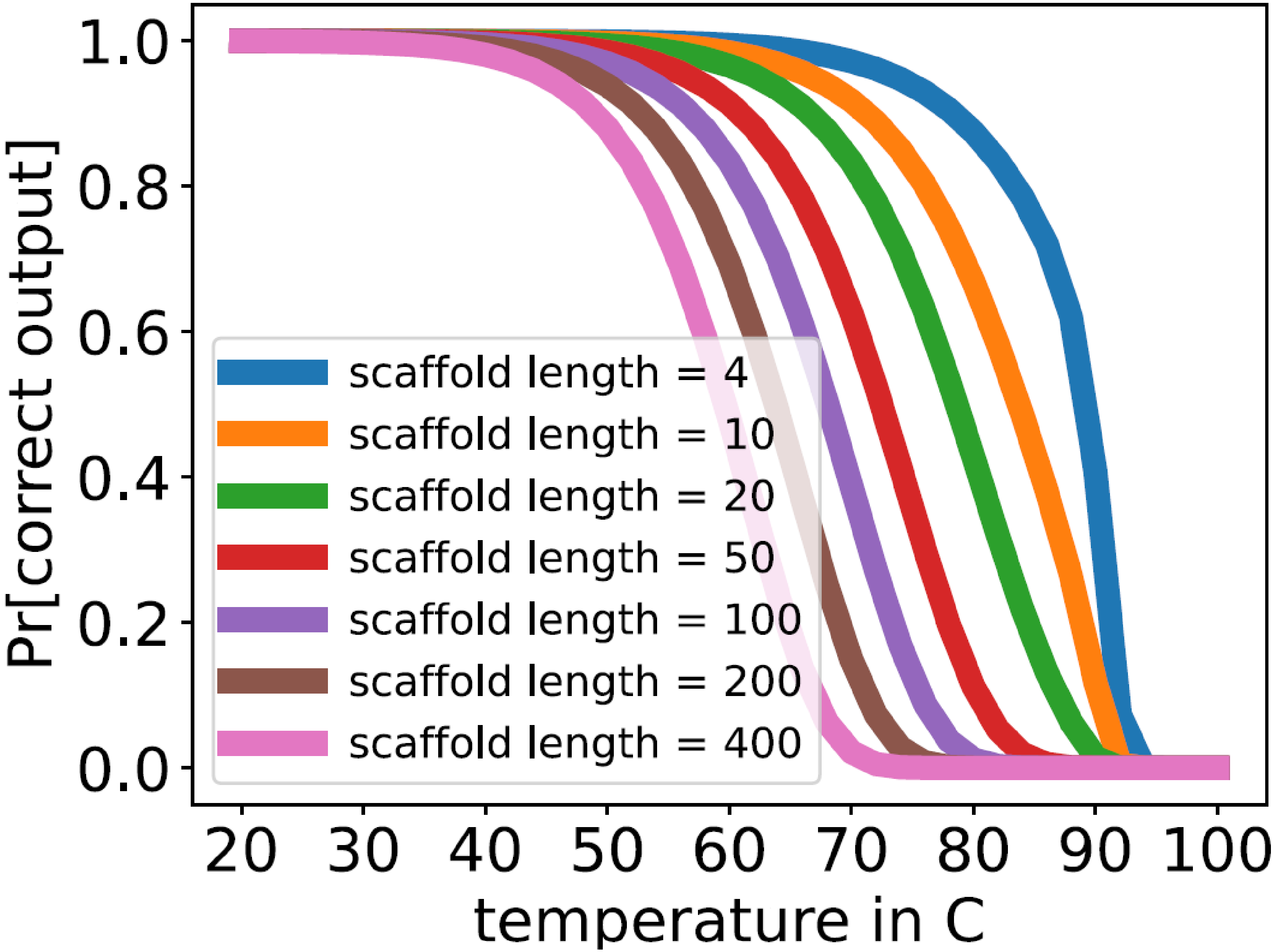
$$Z_s^S = \left(e^{-(\Delta G(d^M(s)) + \Delta G^{\text{assoc}})/kT}\right) \cdot \left[1 + \sum_{s' \in L_s} Z_{(s',s)} * Z_{s'}^S + \sum_{s' \prec s \text{ and } s' \notin L_s} Z_{s'}^S\right] \quad (7)$$

where $Z_{(s',s)} = e^{-\Delta G(d^R(s'), d^L(s))/kT}$.

Algorithm 2 1D SDC partition function algorithm. The proof of Theorem 2 argues that this algorithm returns Z^S as defined in Equation (6). Note that arrays are indexed from 1, and recall that $k_1, \ldots, k_N$ are the counts of competing strands at scaffold domains $d_1, \ldots, d_N$, and we let s_i^j be the j^{th} strand competing at domain d_i.

$Z_{\text{curr}} = [0, 0, \ldots, 0]$ ▷ size $k = \max(k_1, \ldots, k_N)$, current (partial) partition function
$Z_{\text{prev}} = [0, 0, \ldots, 0]$ ▷ size $k = \max(k_1, \ldots, k_N)$, previous (partial) partition function
$Z^S \leftarrow 1$; $\text{sum}_a \leftarrow 0$
for $i \leftarrow 1 \ldots N$ **do**
  $\text{sum}_a \leftarrow \text{sum}_a + \sum_{i \in \{1,\ldots,k\}} Z_{\text{prev}}[i]$ ▷ sum_a: rightmost summation Equation (7)
  $Z_{\text{prev}} \leftarrow Z_{\text{curr}}$
  $Z_{\text{curr}} = [0, 0, \ldots, 0]$
  for $j \leftarrow 1 \ldots k_i$ **do** ▷ each iteration computes Equation (7) for a strand
    $t_1 = \mathrm{e}^{-(\Delta G(d^M(s_i^j)) + \Delta G^{\text{assoc}})/kT}$
    if $i = 1$ **then** ▷ first domain where is no neighbors at all
      $Z_{\text{curr}}[j] = t_1$
    else
      $t_2 \leftarrow 0$
      for $m \leftarrow 1 \ldots k_{i-1}$ **do**
        $t_2 \leftarrow t_2 + \left(\mathrm{e}^{-\left(\Delta G(d^R(s_{i-1}^m), d^L(s_i^j))\right)/kT}\right) \cdot Z_{\text{prev}}[m]$
      end for
      $Z_{\text{curr}}[j] \leftarrow t_1 + t_2 + \text{sum}_a$
    end if
    $Z^S \leftarrow Z^S + Z_{\text{curr}}[j]$ ▷ computing Equation (6)
  end for
end for
return Z^S

▶ **Theorem** There is an $O(k^2 N)$ time algorithm for the domain-level partition function for a 1D SDC of length N with $\leq k$ computation strands competing at each scaffold domain.



Thanks