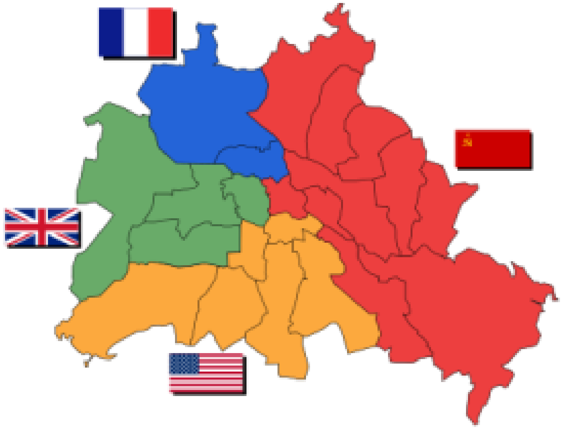
STCS Vigyan Vidushi 2025

Cake Cutting

Rohit Vaish

Fair Division

Fair Division



Fair Division

Fair Division

Fair Division

Fair Division

Divisible

Fair Division

Divisible

Indivisible

Fair Division

Divisible




Indivisible

Fair Division

Divisible

Indivisible

Fair Division

Divisible

Indivisible

Fair Division

Divisible

Indivisible

Fair Division

Divisible

Indivisible

Cake Cutting

How to fairly divide a cake

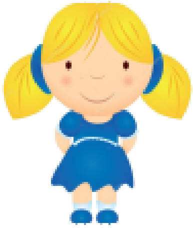
Cake Cutting

How to fairly divide a cake

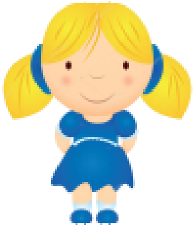
among agents with differing preferences?

Why is this problem interesting?

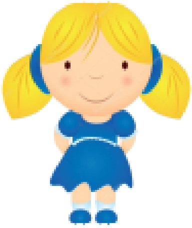
Why is this problem interesting?



Why is this problem interesting?



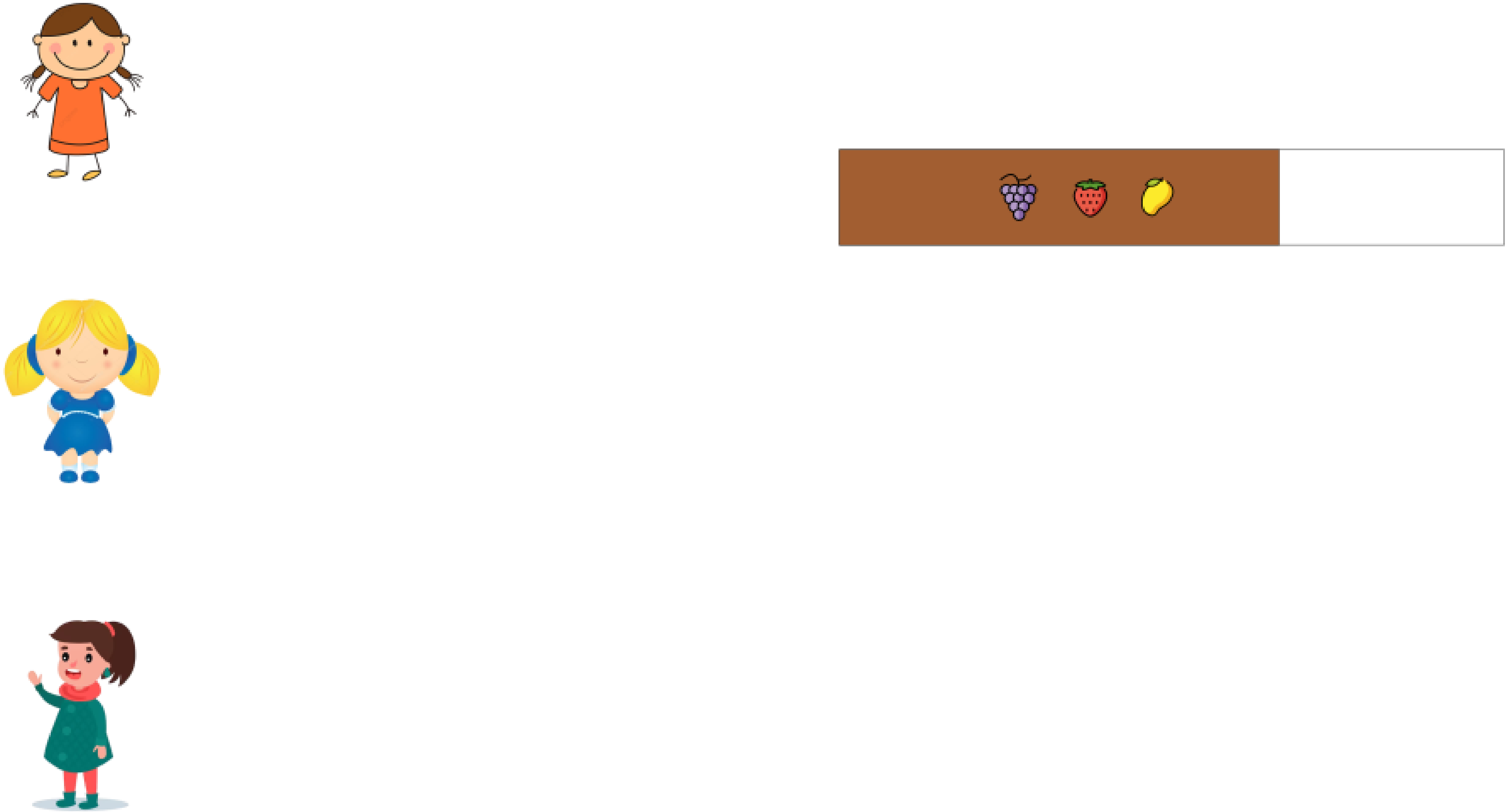
Why is this problem interesting?



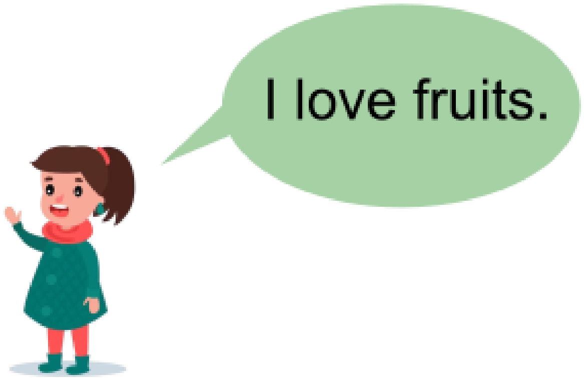
Why is this problem interesting?

Fair

Why is this problem interesting?

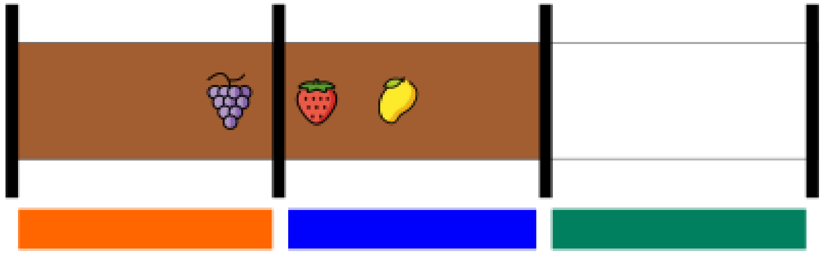


Why is this problem interesting?

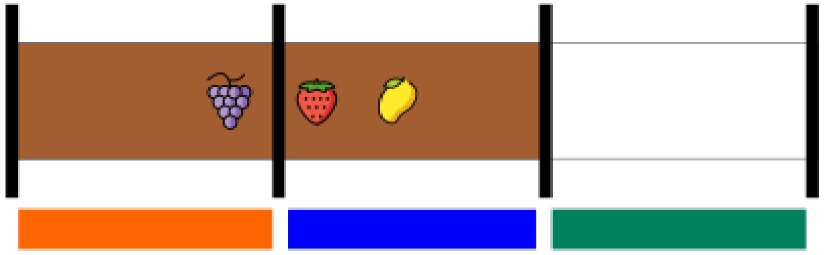


Why is this problem interesting?

Why is this problem interesting?

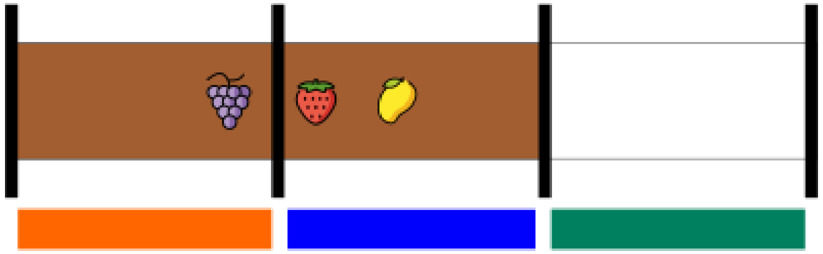


Why is this problem interesting?



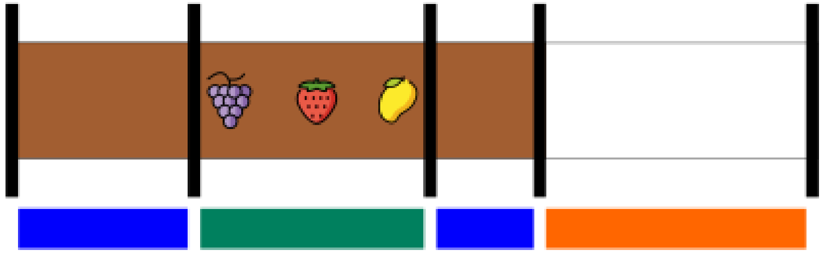
Is this division fair?

Why is this problem interesting?



Is this division fair?

Why is this problem interesting?



A fairer division 

Why is this problem interesting?

Preferences matter!

The Model

The Model

- The resource: Cake [0,1]

The Model

- The resource: Cake [0,1]
- Set of agents {1,2,...,n}

The Model

- The resource: Cake [0,1]
- Set of agents {1,2,...,n}
- *Piece of cake*: Finite union of subintervals of [0,1]

The Model

- The resource: Cake [0,1]
- Set of agents {1,2,...,n}
- *Piece of cake*: Finite union of subintervals of [0,1]

Preferences of Agents

Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

Additivity

for disjoint $X, Y \subseteq [0,1]$,
$$v_i(X \cup Y) = v_i(X) + v_i(Y)$$

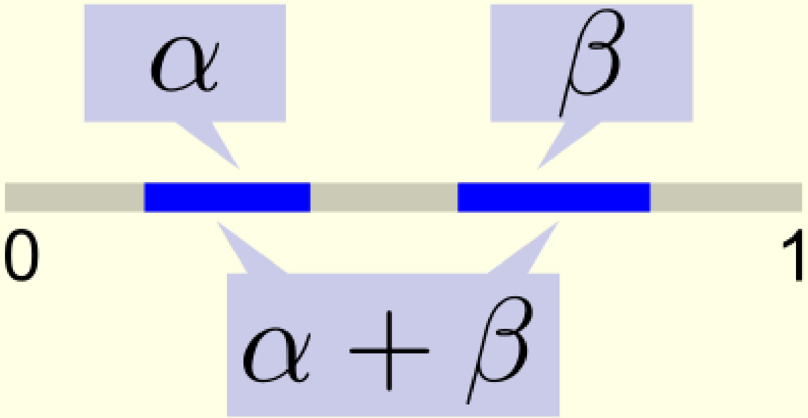
Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

Additivity

for disjoint $X, Y \subseteq [0,1]$,

$$v_i(X \cup Y) = v_i(X) + v_i(Y)$$

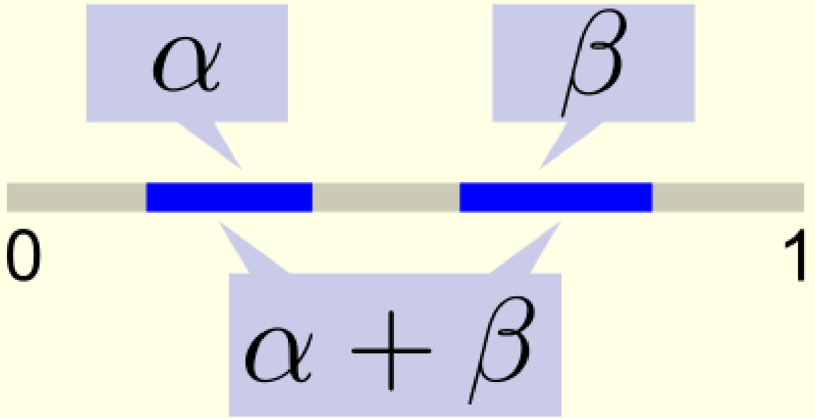


Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

Additivity

for disjoint $X, Y \subseteq [0,1]$,
$$v_i(X \cup Y) = v_i(X) + v_i(Y)$$



Divisibility

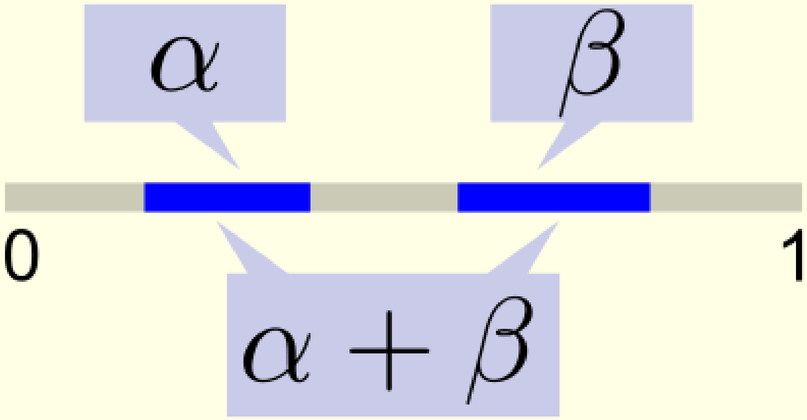
for any $X \subseteq [0,1]$ and any $\lambda \in [0,1]$,
there exists $Y \subseteq X$ s.t. $v_i(Y) = \lambda v_i(X)$

Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

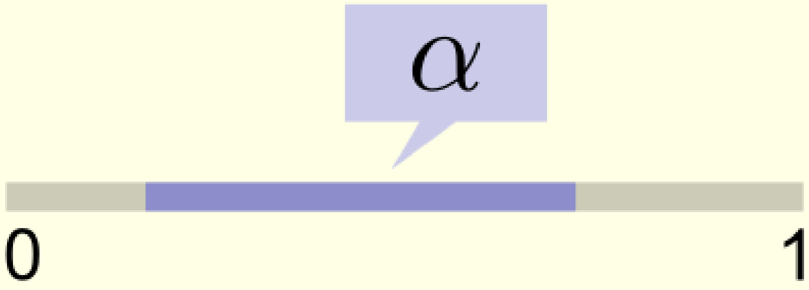
Additivity

for disjoint $X, Y \subseteq [0,1]$,
$v_i(X \cup Y) = v_i(X) + v_i(Y)$



Divisibility

for any $X \subseteq [0,1]$ and any $\lambda \in [0,1]$,
there exists $Y \subseteq X$ s.t. $v_i(Y) = \lambda v_i(X)$

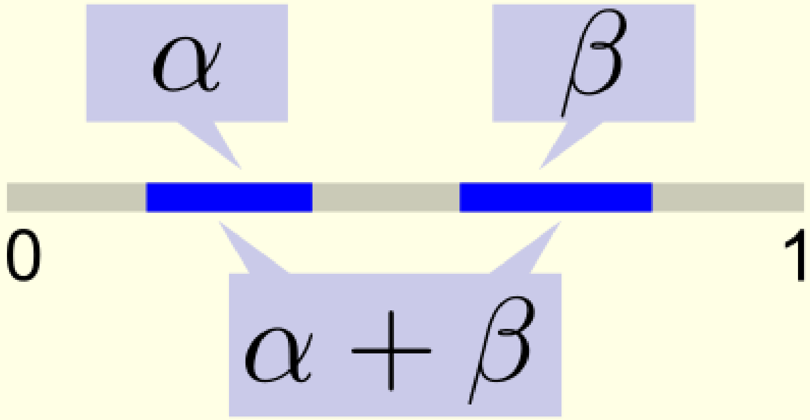


Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

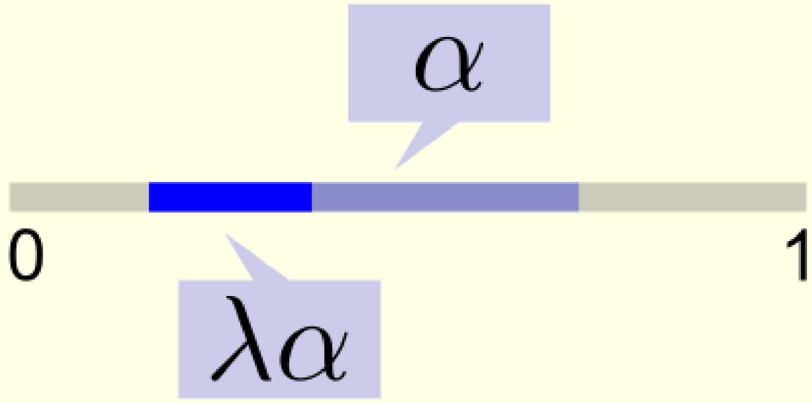
Additivity

for disjoint $X, Y \subseteq [0,1]$,
$v_i(X \cup Y) = v_i(X) + v_i(Y)$



Divisibility

for any $X \subseteq [0,1]$ and any $\lambda \in [0,1]$,
there exists $Y \subseteq X$ s.t. $v_i(Y) = \lambda v_i(X)$

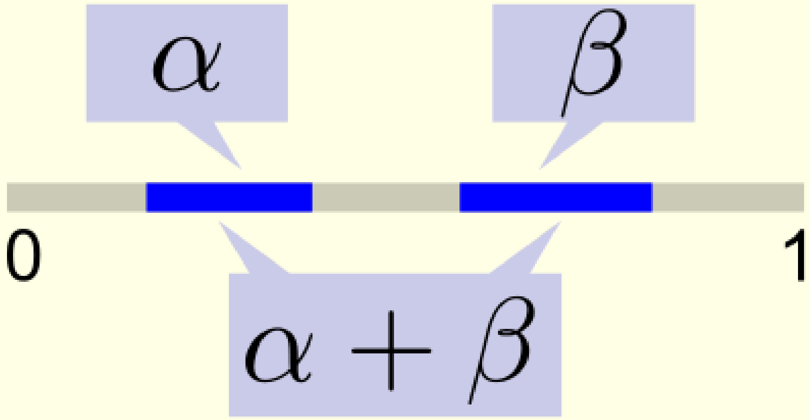


Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

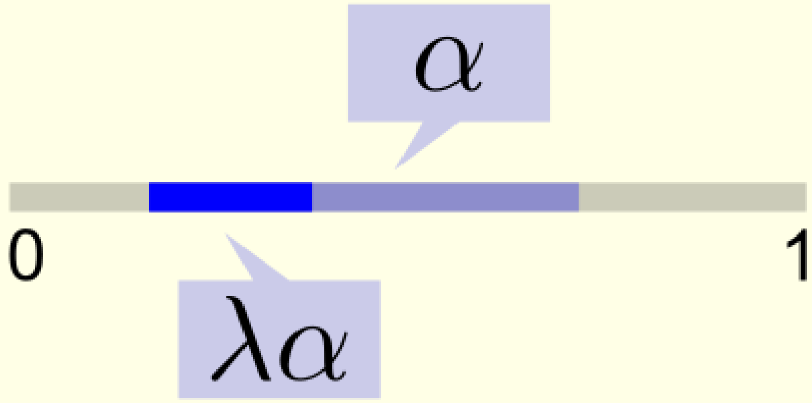
Additivity

for disjoint $X, Y \subseteq [0,1]$,
$v_i(X \cup Y) = v_i(X) + v_i(Y)$



Divisibility

for any $X \subseteq [0,1]$ and any $\lambda \in [0,1]$,
there exists $Y \subseteq X$ s.t. $v_i(Y) = \lambda v_i(X)$

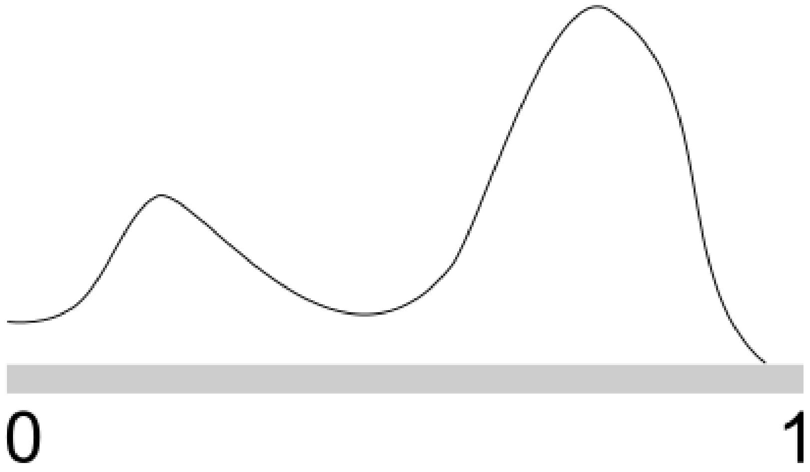


Normalization: for each agent i, $v_i([0,1]) = 1$.

Preferences of Agents

- Valuation function v_i: Assigns a non-negative value to any piece of cake

$$v_i(X) = \int_{x \in X} f_i(x)dx$$

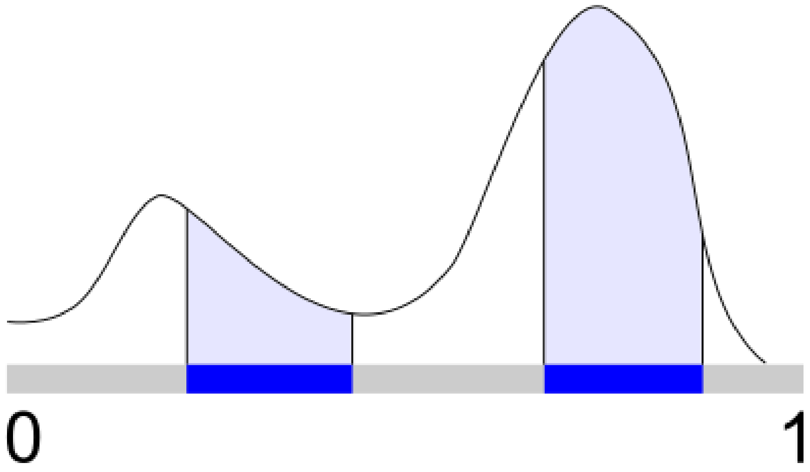


Preferences of Agents

- **Valuation function** v_i: Assigns a non-negative value to any piece of cake

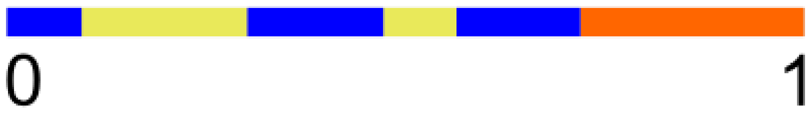
$$v_i(X) = \int_{x \in X} f_i(x) dx$$

value density function



Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.



Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$

Fairness notions

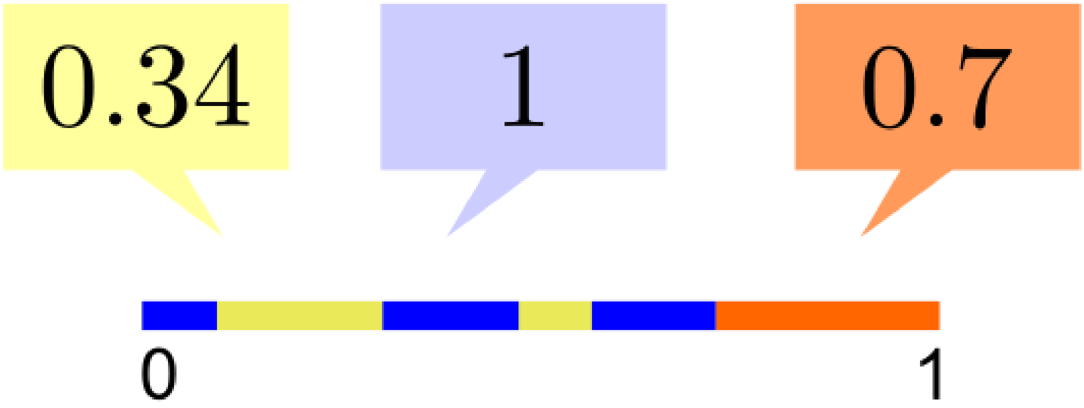
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$



Fairness notions

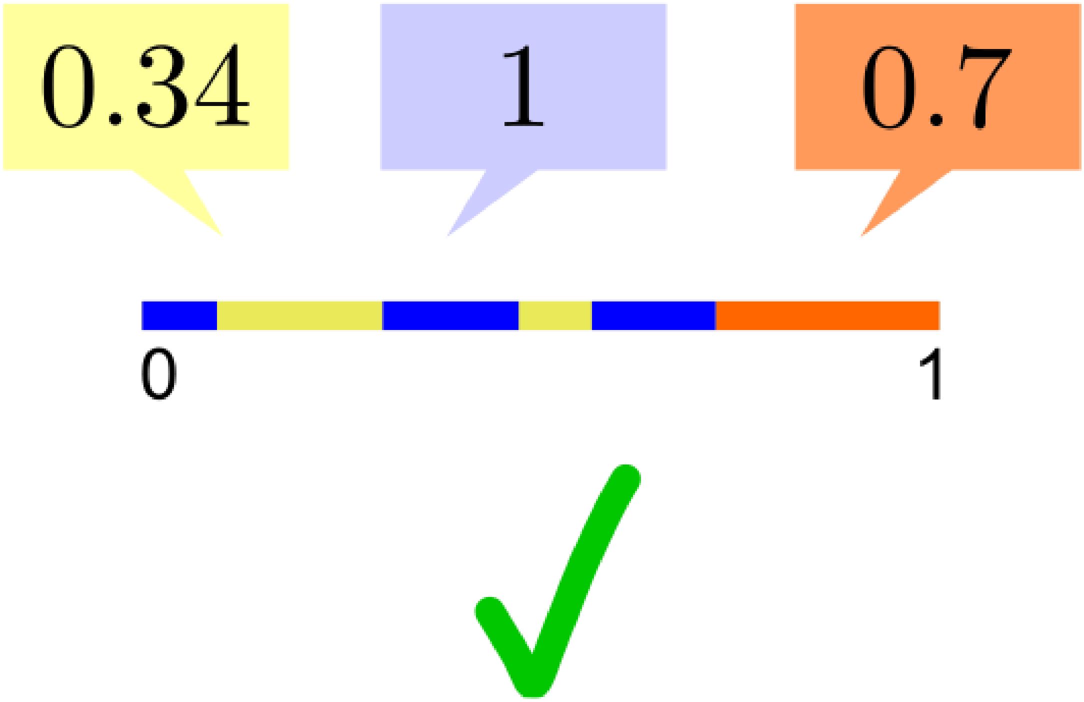
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$



Fairness notions

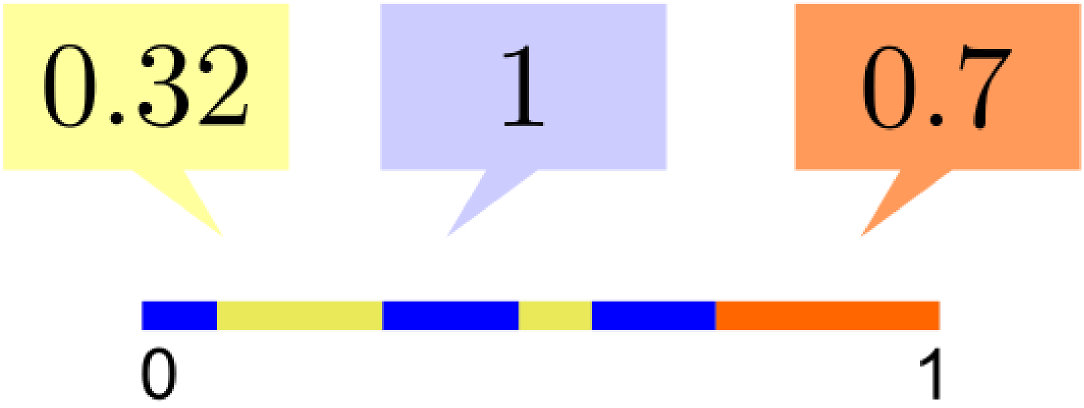
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$



Fairness notions

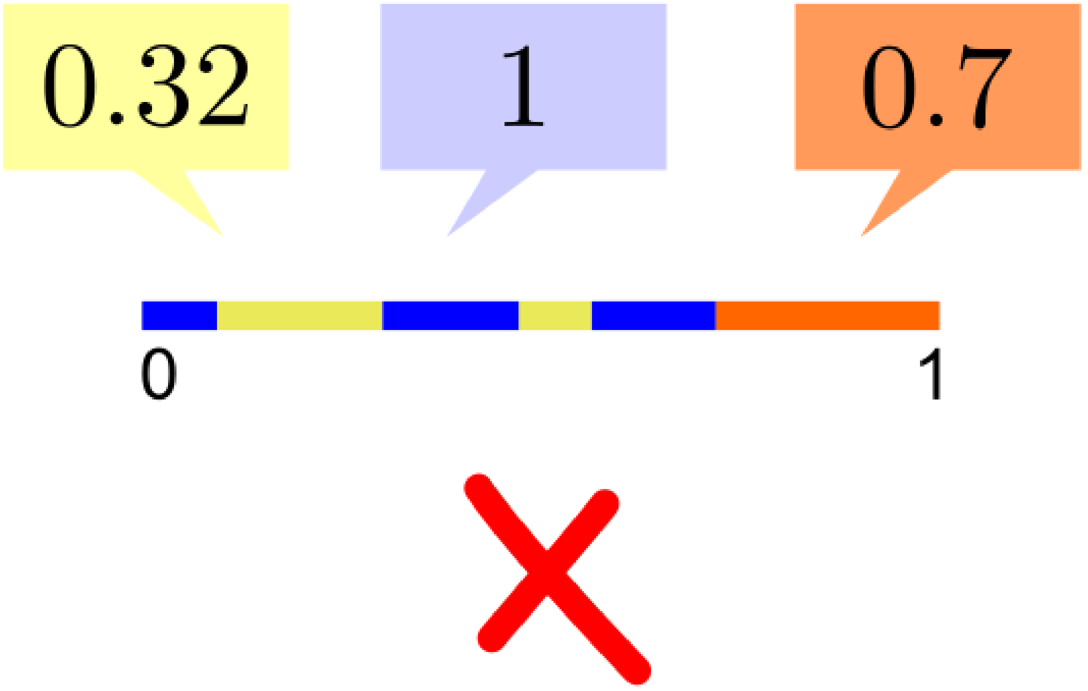
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$



Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Envy-freeness

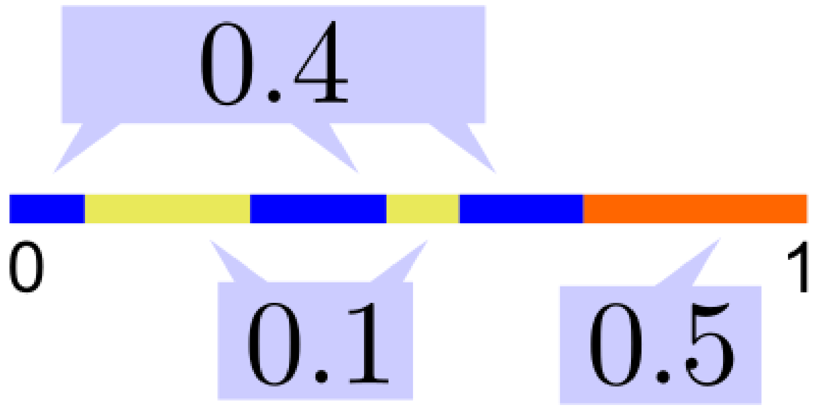
[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.



Envy-freeness

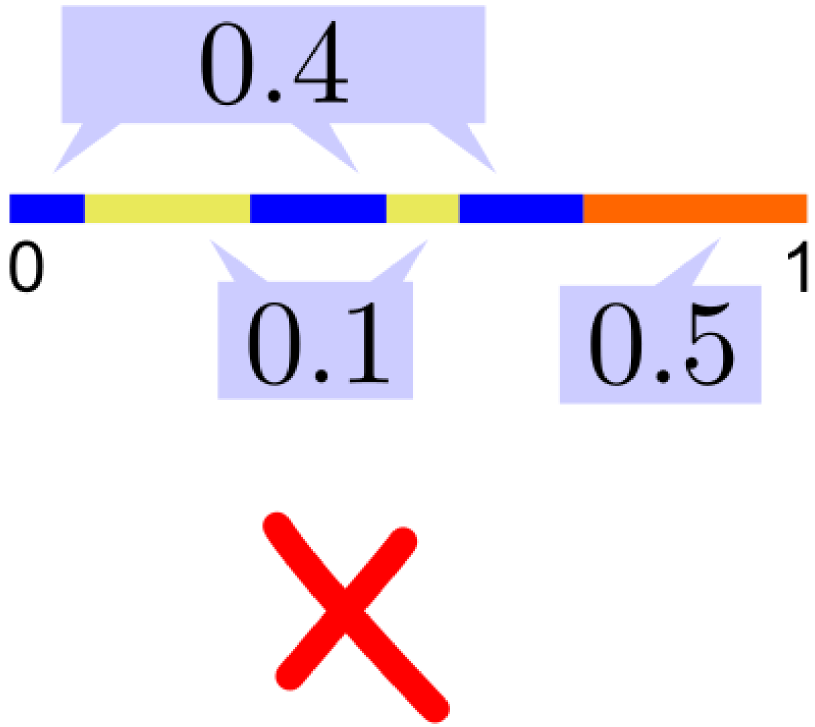
[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.



Envy-freeness

[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$

Envy-freeness

[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality
[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$

Envy-freeness
[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

For two agents (n=2), is one property stronger than the other?

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality
[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$

Envy-freeness
[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

What about three or more agents?

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$

Envy-freeness

[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

EF implies Prop for *any* number of agents

Fairness notions

- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality
[Steinhaus, 1948]

for each agent i,

$$v_i(A_i) \geq \frac{1}{n}$$

Envy-freeness
[Gamow and Stern, 1958; Foley, 1967]

for every pair of agents i,j,

$$v_i(A_i) \geq v_i(A_j)$$

EF implies Prop for *any* number of agents

Prop implies EF for *two* agents (but no more)

Robertson-Webb Query Model

Robertson-Webb Query Model

Types of queries that can be used to access the valuation functions

Robertson-Webb Query Model

Types of queries that can be used to access the valuation functions

$\texttt{eval}_i(x, y)$: returns $v_i([x, y])$

$\texttt{cut}_i(x, \alpha)$: returns y such that $v_i([x, y]) = \alpha$

Robertson-Webb Query Model

Types of queries that can be used to access the valuation functions

$\texttt{eval}_i(x, y)$: returns $v_i([x, y])$

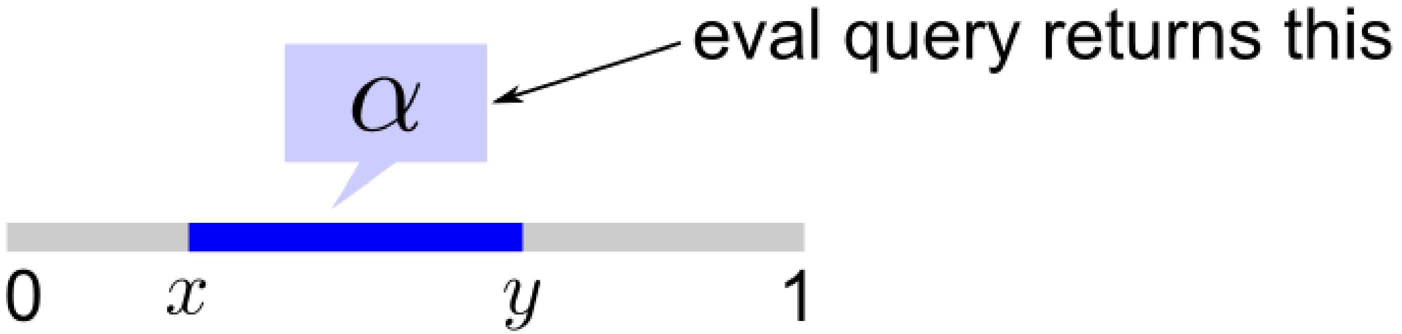
$\texttt{cut}_i(x, \alpha)$: returns y such that $v_i([x, y]) = \alpha$

Robertson-Webb Query Model

Types of queries that can be used to access the valuation functions

$\texttt{eval}_i(x, y)$: returns $v_i([x, y])$

$\texttt{cut}_i(x, \alpha)$: returns y such that $v_i([x, y]) = \alpha$

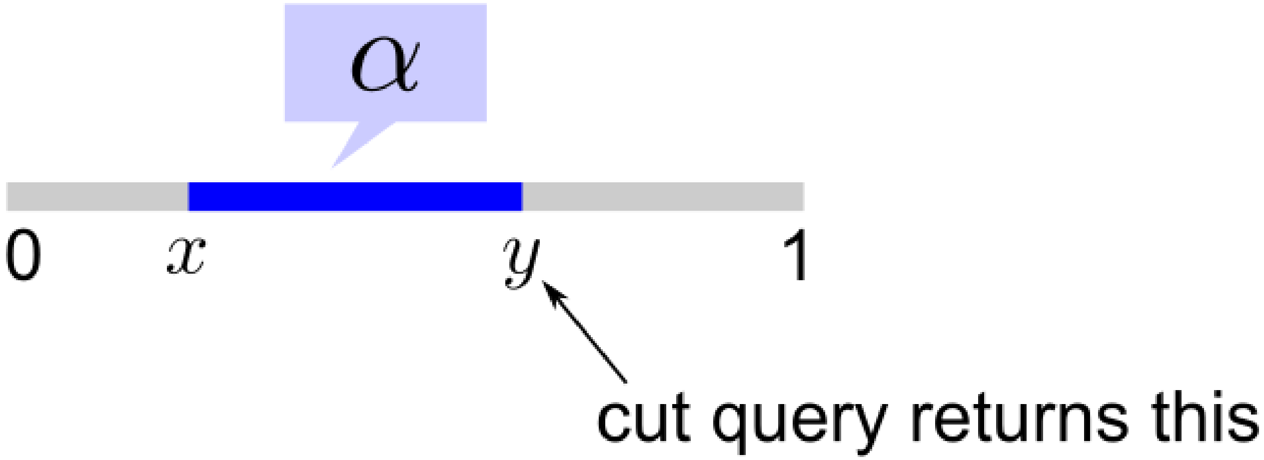


Robertson-Webb Query Model

Types of queries that can be used to access the valuation functions

$\texttt{eval}_i(x, y)$: returns $v_i([x, y])$

$\texttt{cut}_i(x, \alpha)$: returns y such that $v_i([x, y]) = \alpha$



Cake-Cutting Algorithms

Let's start by thinking about proportionality for two agents.

Cut and Choose

Cut and Choose

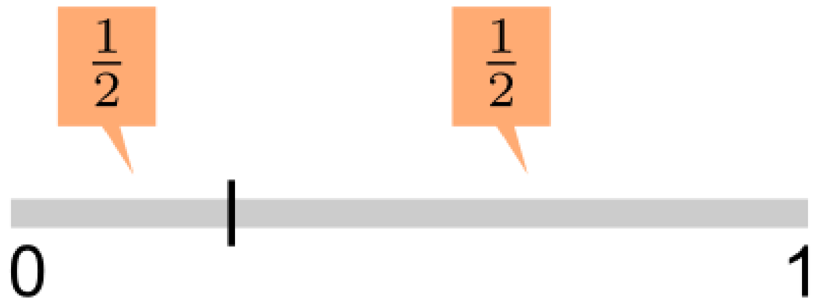
1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

Cut and Choose

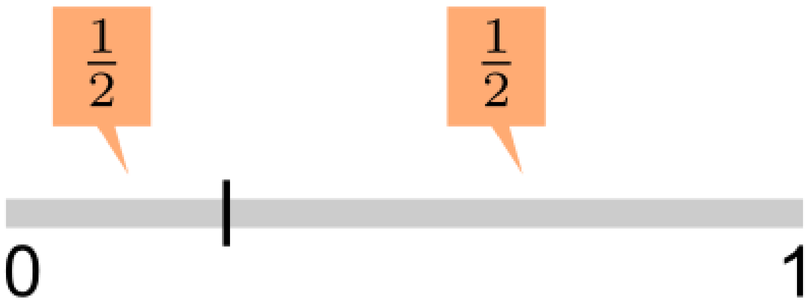
1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).



Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

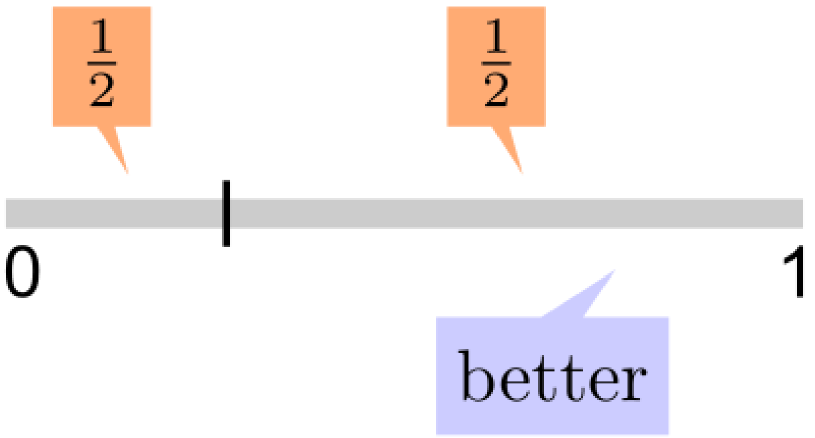
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

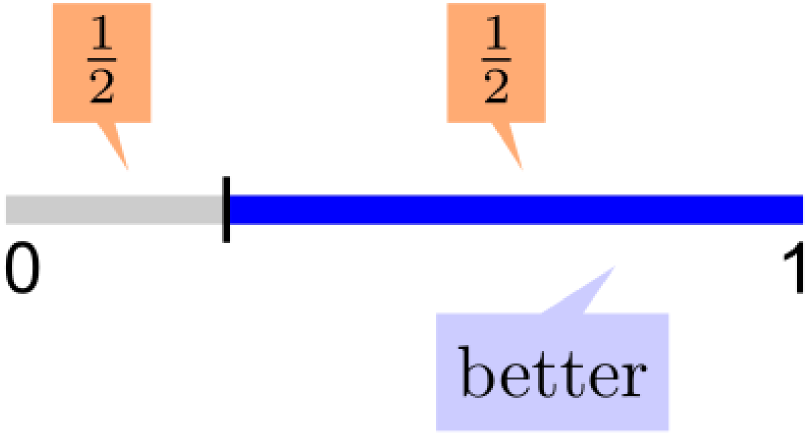
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Cut and Choose

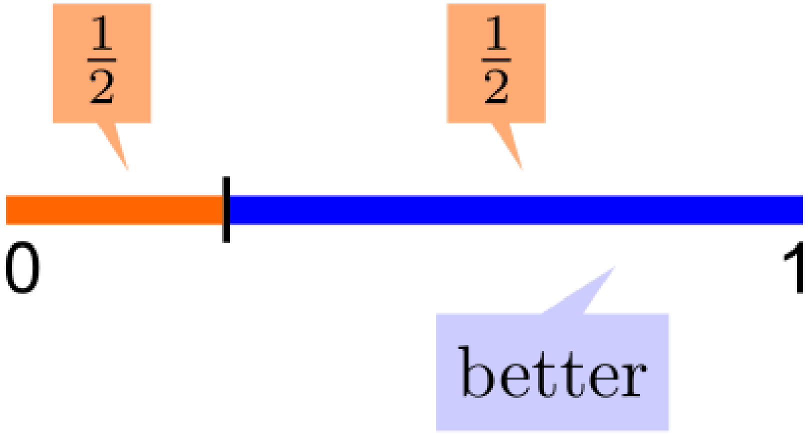
1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



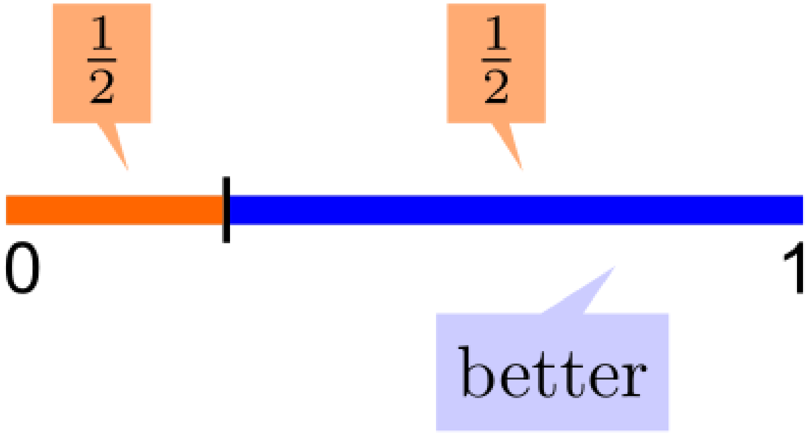
Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.

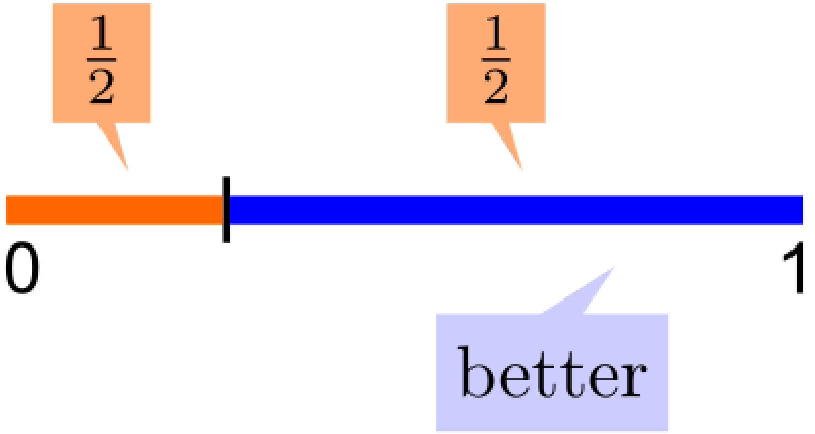


Is the cut-and-choose outcome proportional?

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



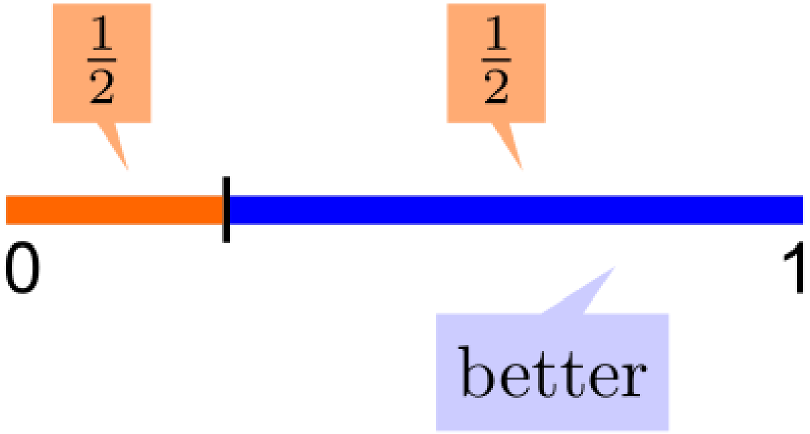
Is the cut-and-choose outcome proportional?

Yes! Agent 2's value is at least 1/2. Agent 1's value is exactly 1/2.

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.

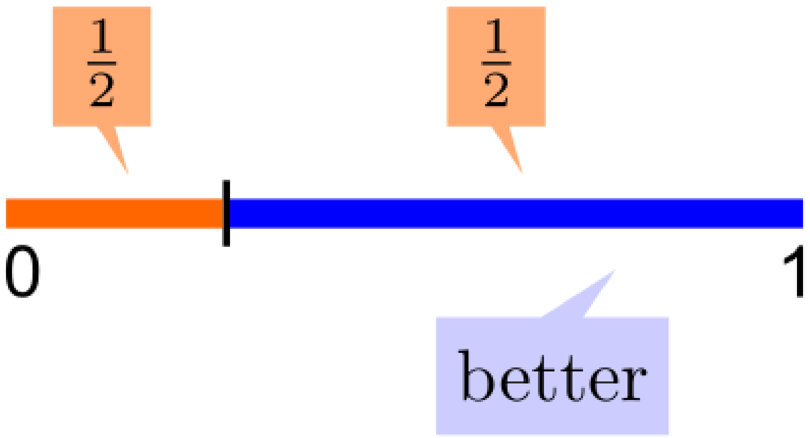


Is the cut-and-choose outcome envy-free?

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.

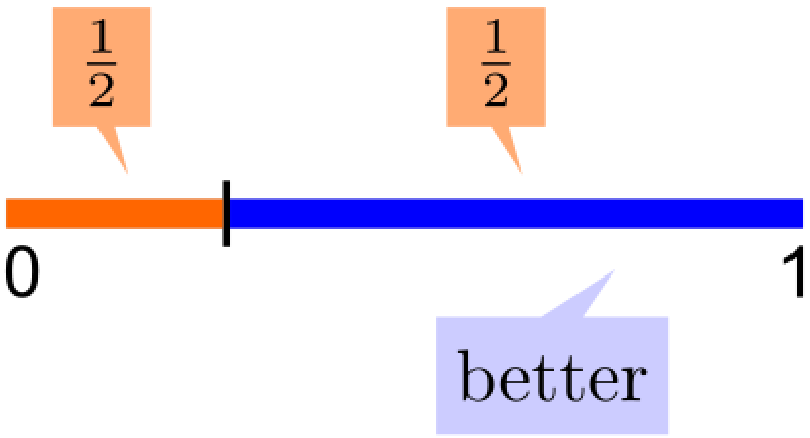


Is the cut-and-choose outcome envy-free?

Yes! EF and Prop are equivalent for two agents.

Cut and Choose

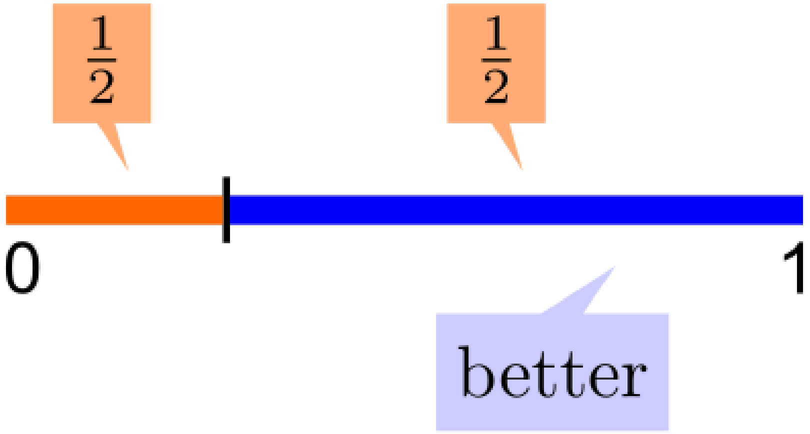
1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Can cut-and-choose be implemented in the Robertson-Webb model?

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



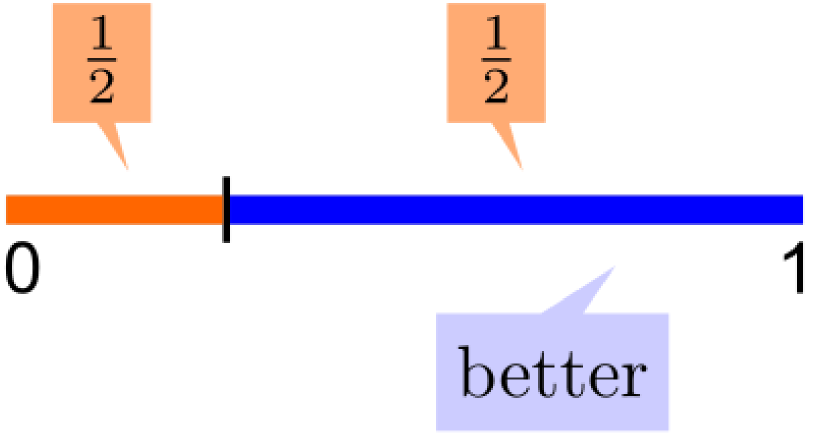
Can cut-and-choose be implemented in the Robertson-Webb model?

$$y = \texttt{cut}_1(0, 1/2)$$

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Can cut-and-choose be implemented in the Robertson-Webb model?

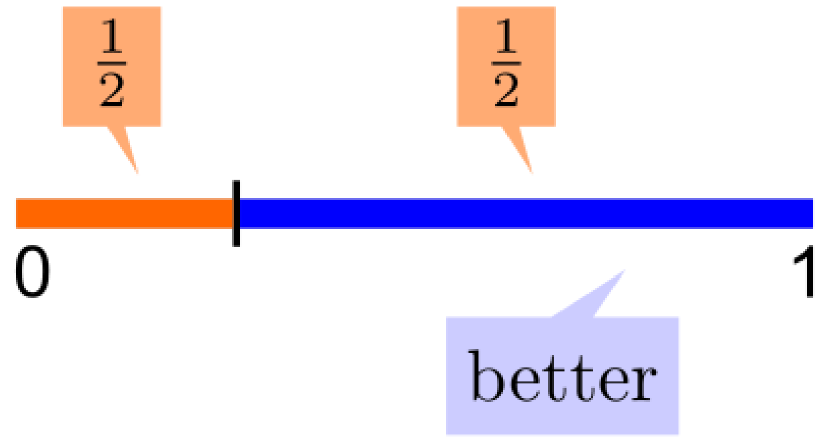
$$y = \texttt{cut}_1(0, 1/2)$$

$$\texttt{eval}_2(0, y)$$

Cut and Choose

1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



For two agents, an envy-free/proportional cake division can be computed using two queries.