COL866: Special Topics in Algorithms

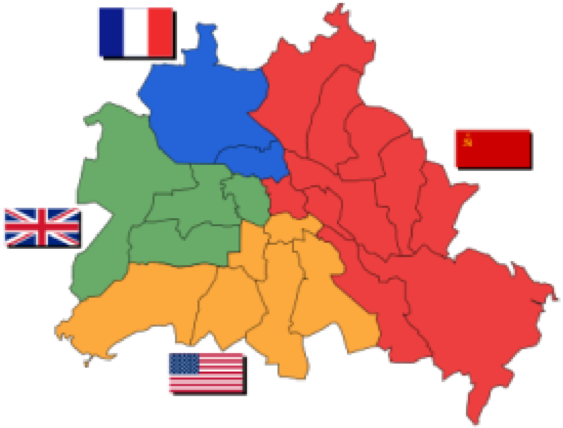
Lecture 7

Cake Cutting

Aug 25, 2023 | 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

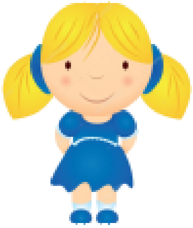
Cake Cutting

How to fairly divide a cake

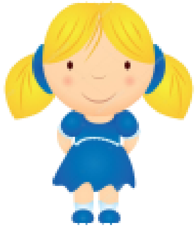
among agents with differing preferences?

Why is this problem interesting?

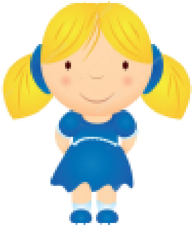
Why is this problem interesting?

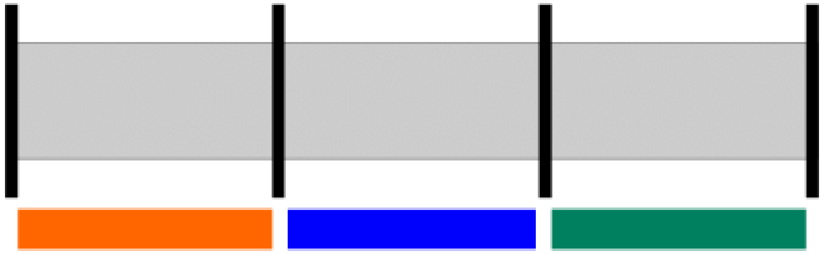


Why is this problem interesting?

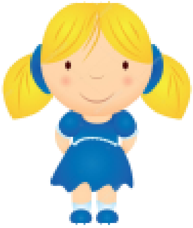


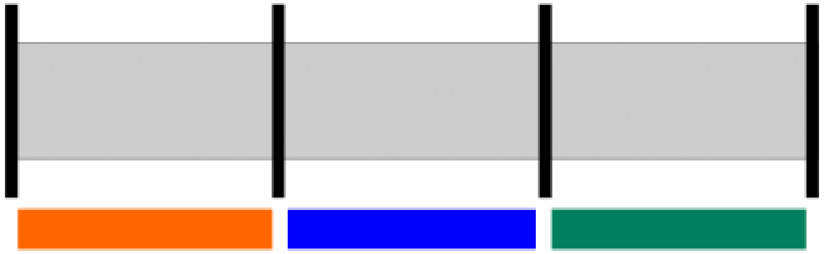
Why is this problem interesting?





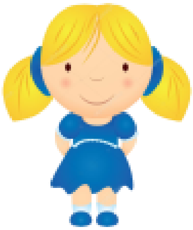
Why is this problem interesting?



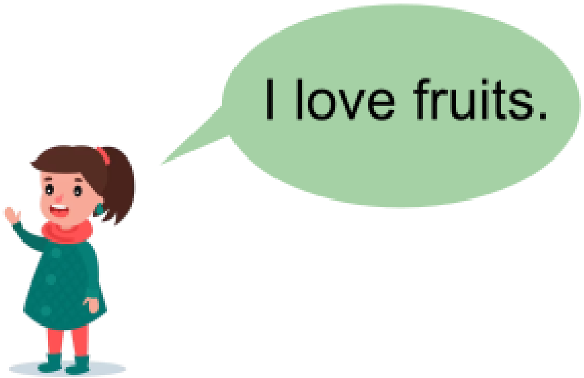


Fair

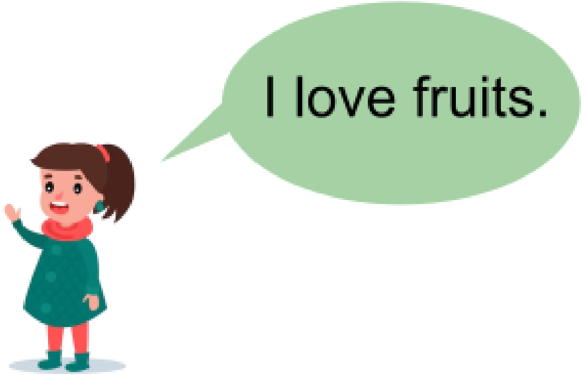
Why is this problem interesting?



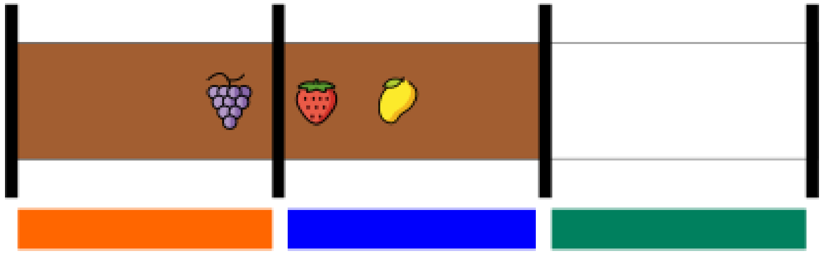
Why is this problem interesting?



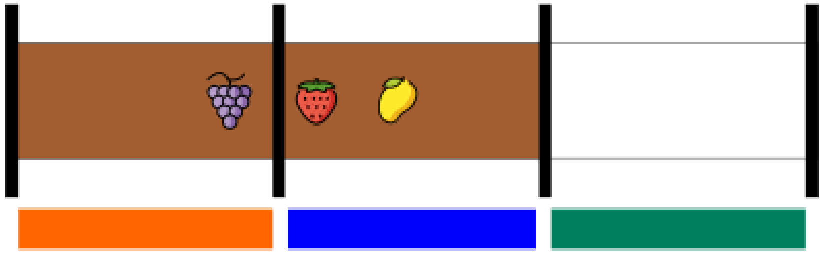
Why is this problem interesting?



Why is this problem interesting?

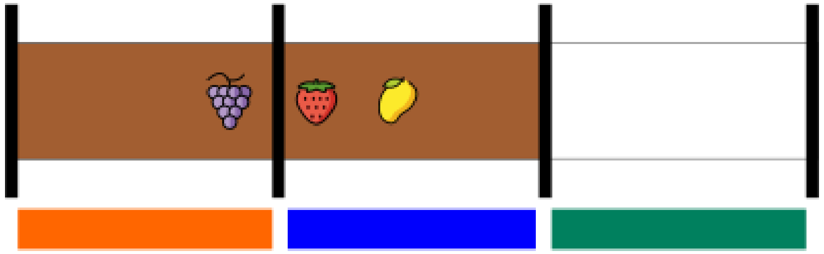


Why is this problem interesting?



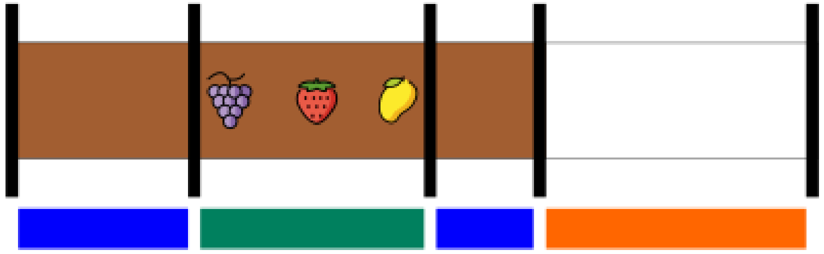
Is this division fair?

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

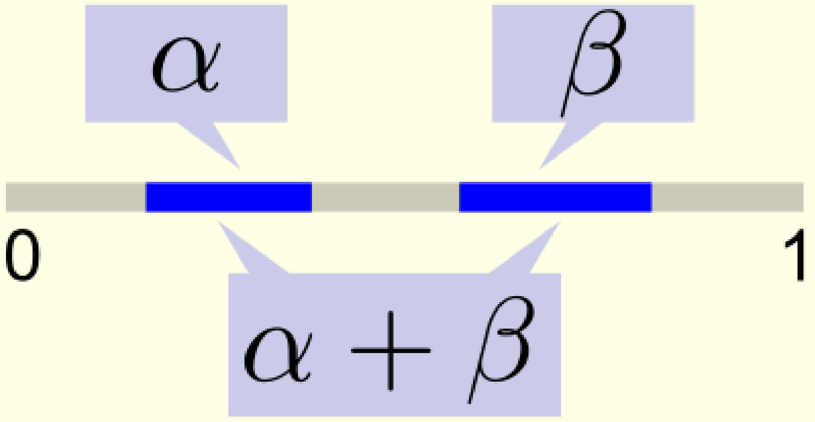
for disjoint $X, Y \subseteq [0,1]$,
$$v_i(X \cup Y) = v_i(X) + v_i(Y)$$

Preferences of Agents

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

Additivity

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

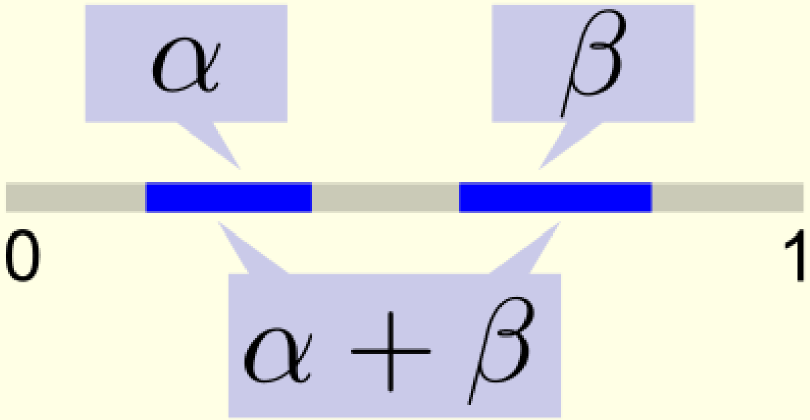


Preferences of Agents

- 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

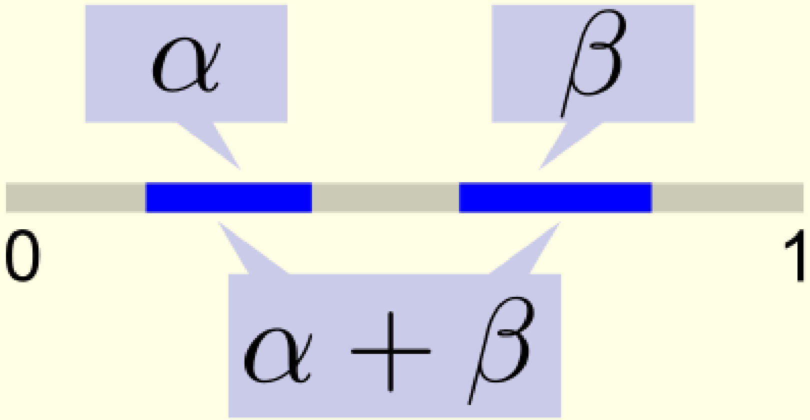
for any $X \subseteq [0,1]$ and any $\lambda \in [0,1]$,
there exists $Y \subseteq X$ s.t. $v_i(Y) = \lambda v_i(X)$

Preferences of Agents

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

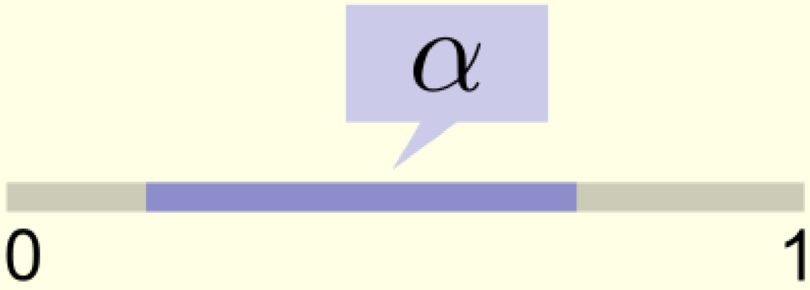
Additivity

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. $v_i(Y) = \lambda v_i(X)$

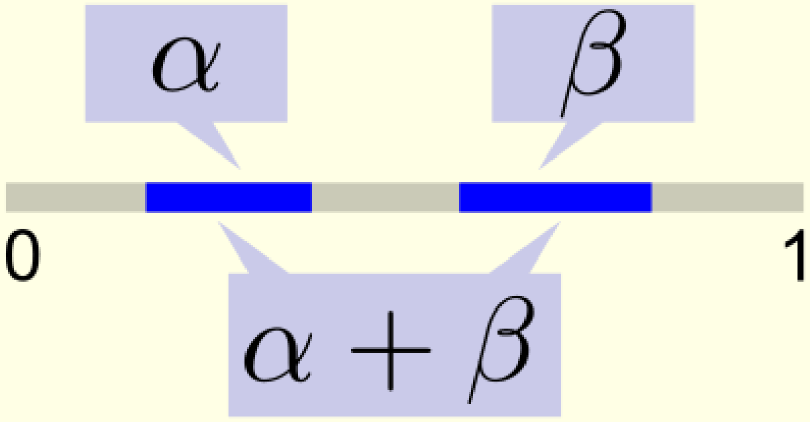


Preferences of Agents

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

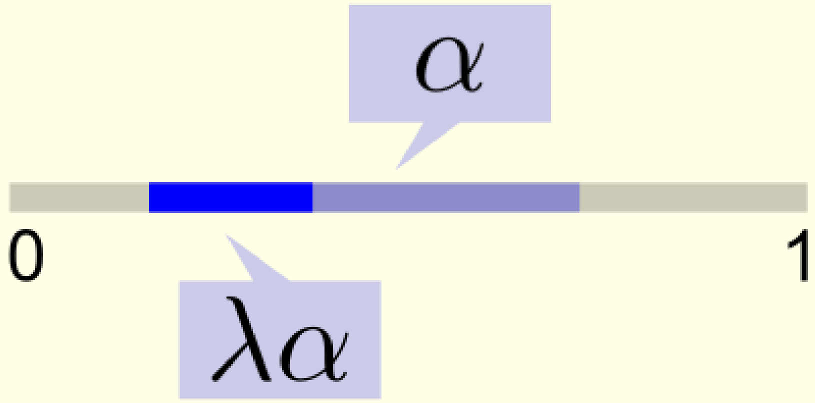
Additivity

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. $v_i(Y) = \lambda v_i(X)$

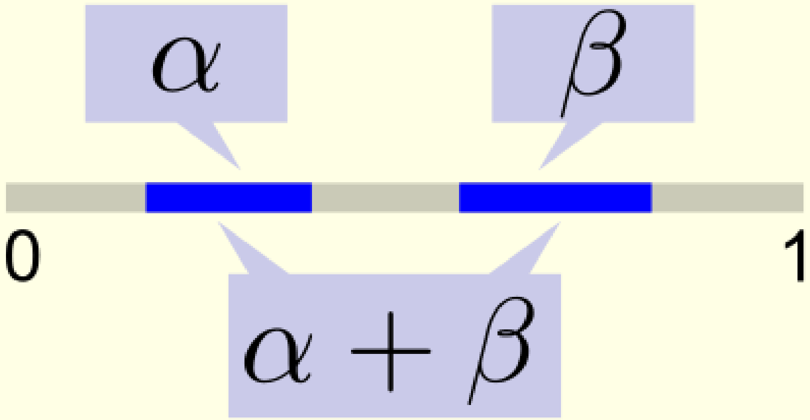


Preferences of Agents

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

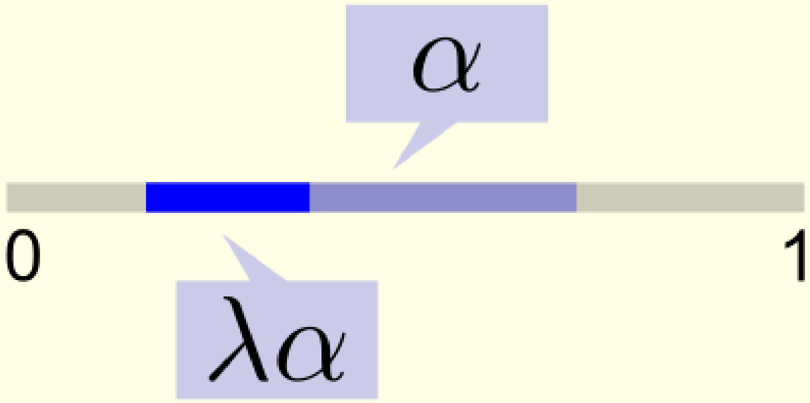
Additivity

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. $v_i(Y) = \lambda v_i(X)$

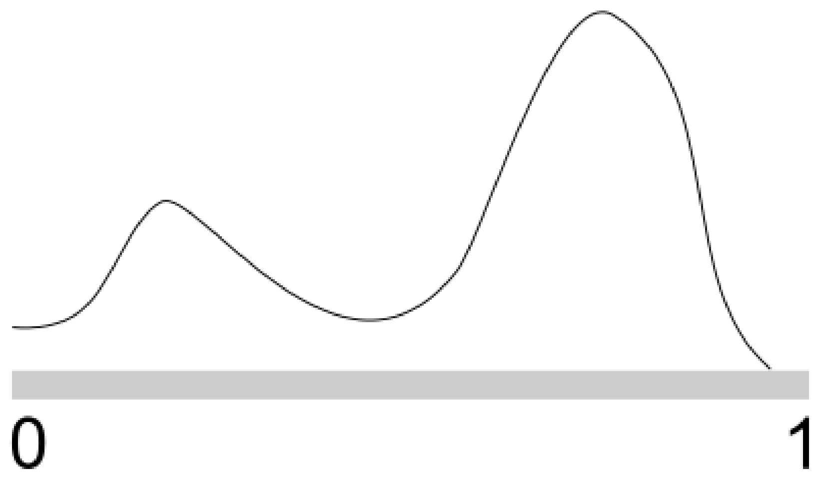


Normalization: for each agent i, $v_i([0,1]) = 1$.

Preferences of Agents

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

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

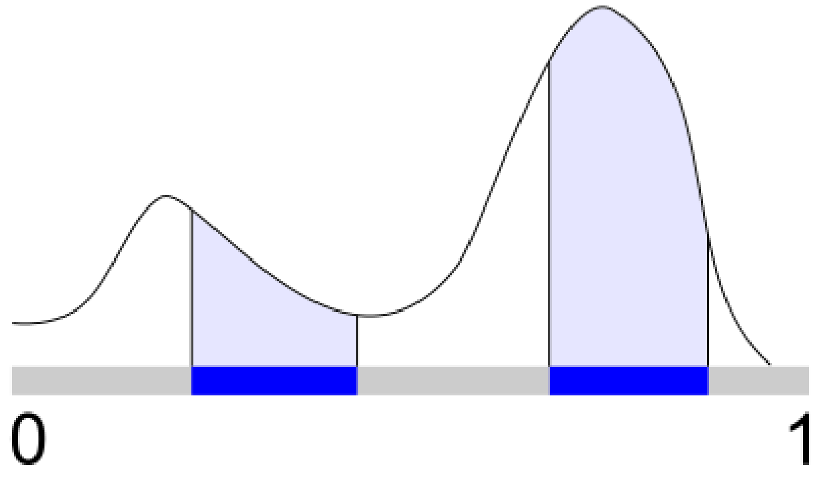


Preferences of Agents

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

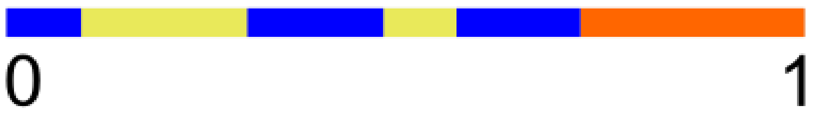
$$v_i(X) = \int_{x \in X} f_i(x) dx$$

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

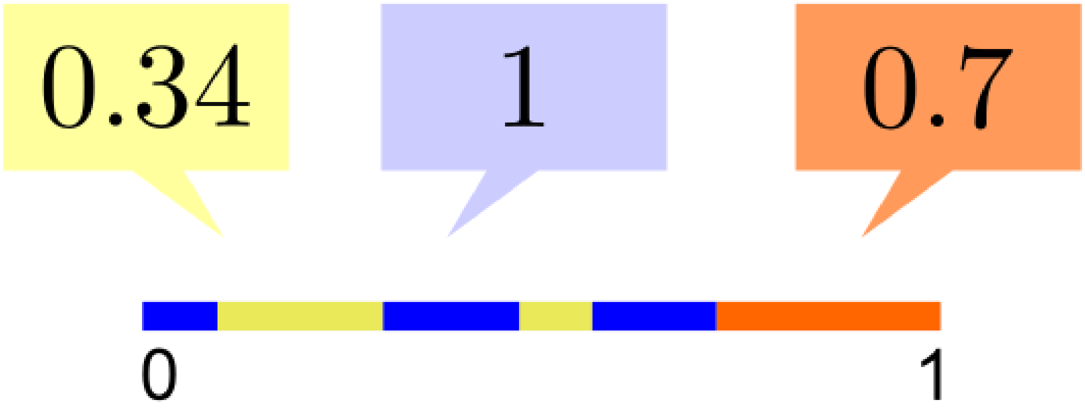
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

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



Fairness notions

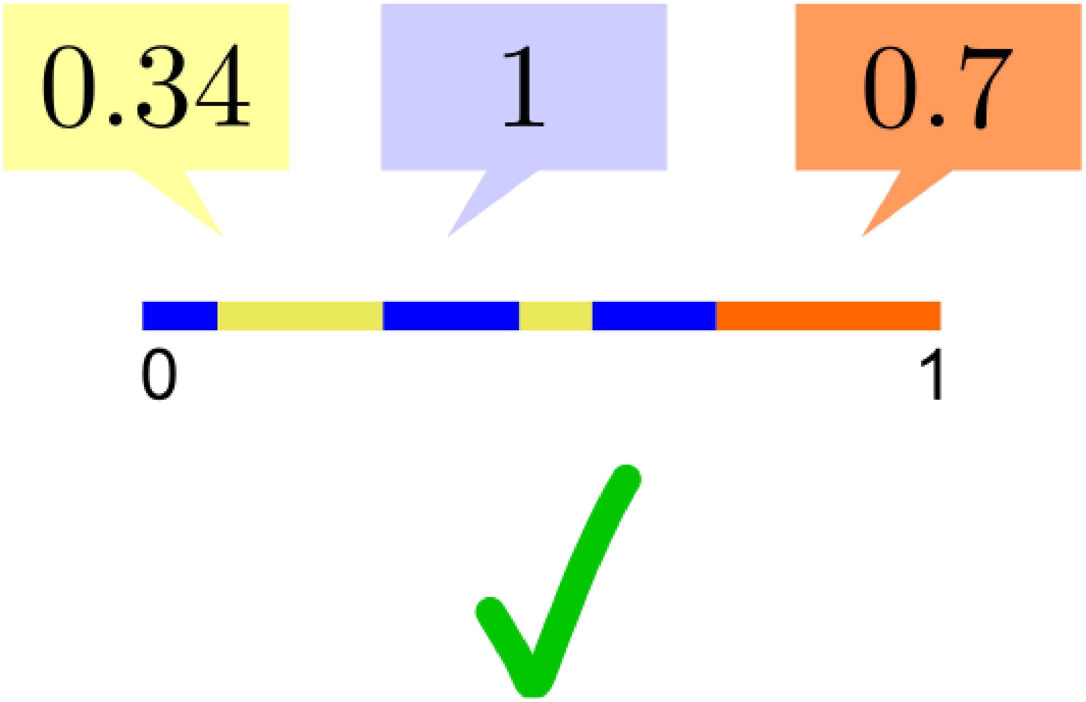
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

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



Fairness notions

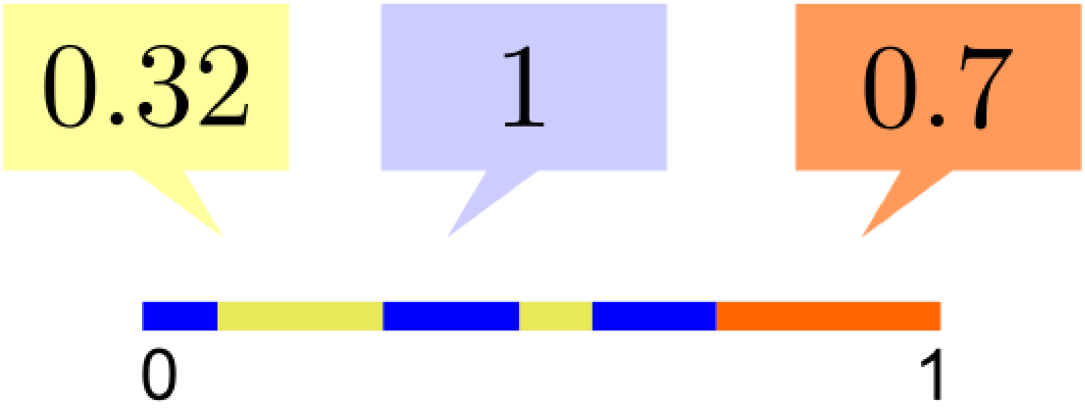
- *Allocation/Division*: A partition $(A_1, A_2, \ldots, A_n)$ of the cake [0,1] where each A_i is a piece of cake.

Proportionality

[Steinhaus, 1948]

for each agent i,

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



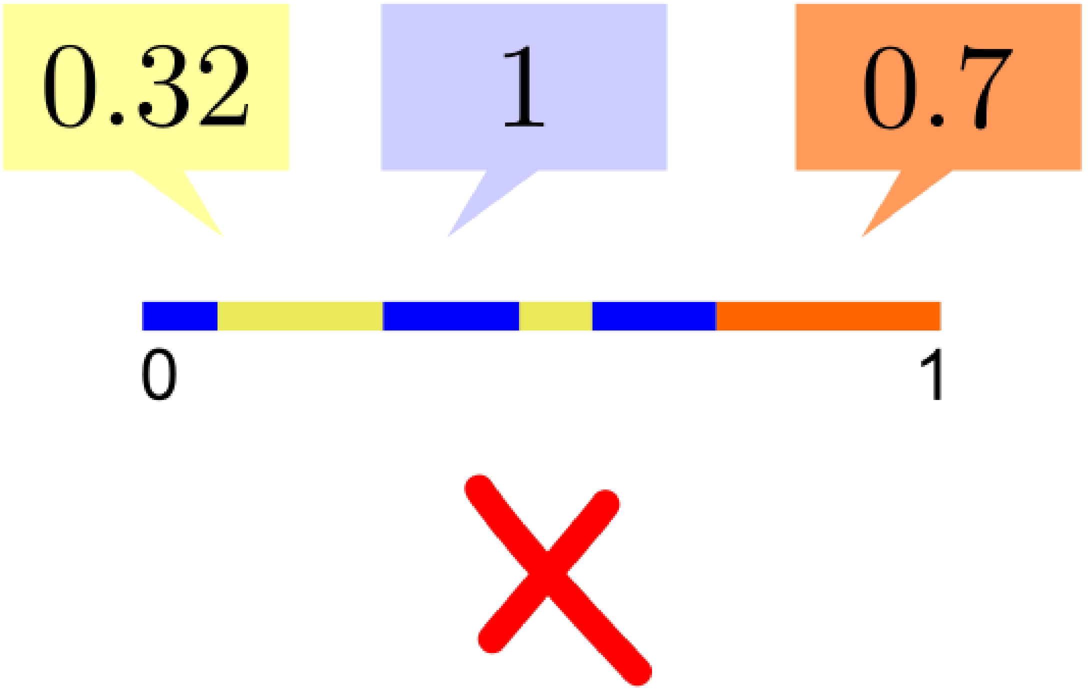
Fairness notions

- *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

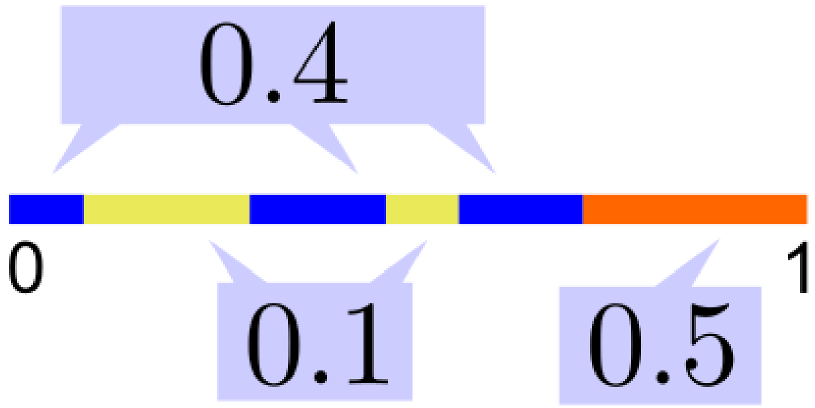
[Gamow and Stern, 1958; Foley, 1967]

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

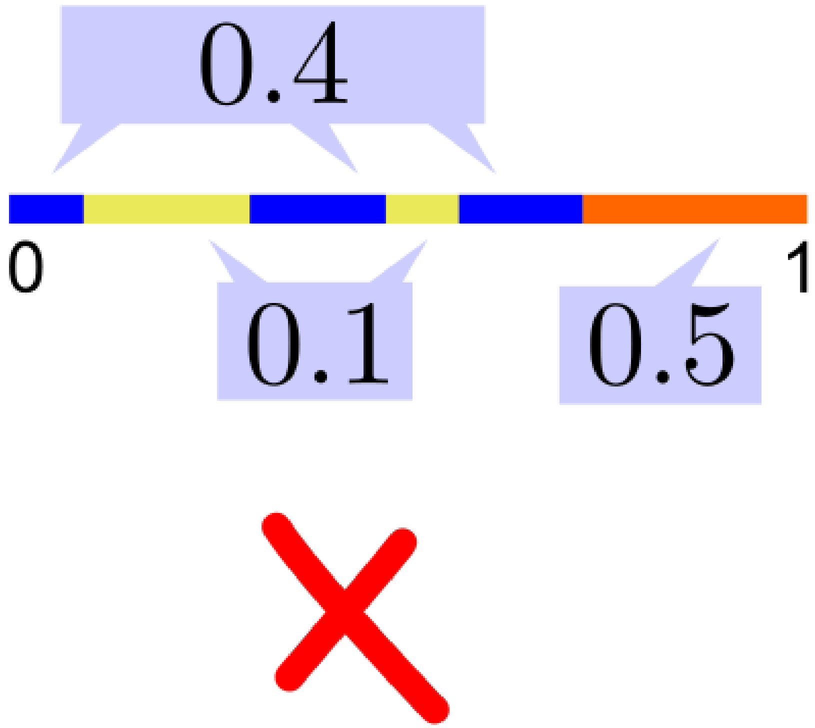
[Gamow and Stern, 1958; Foley, 1967]

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 $v_i([x, y])$

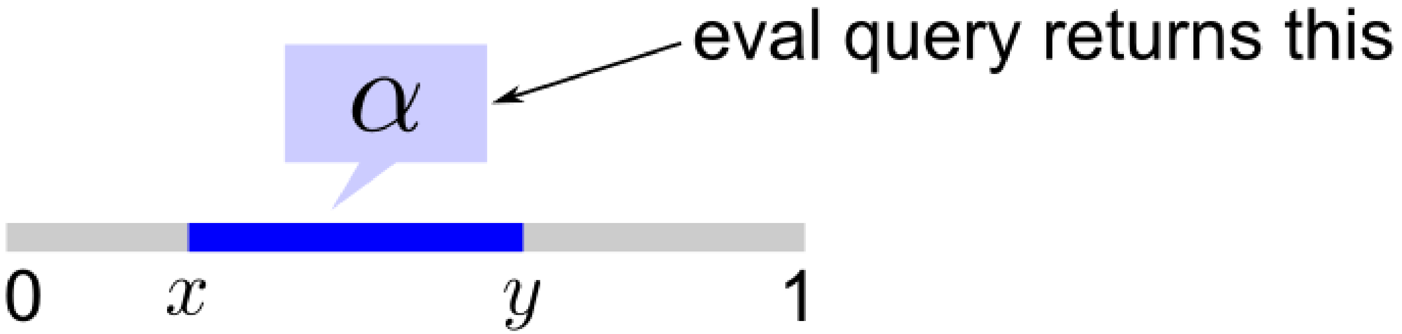
$\texttt{cut}_i(x, \alpha)$: returns y such that $v_i([x, y]) = \alpha$

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 $v_i([x, y]) = \alpha$

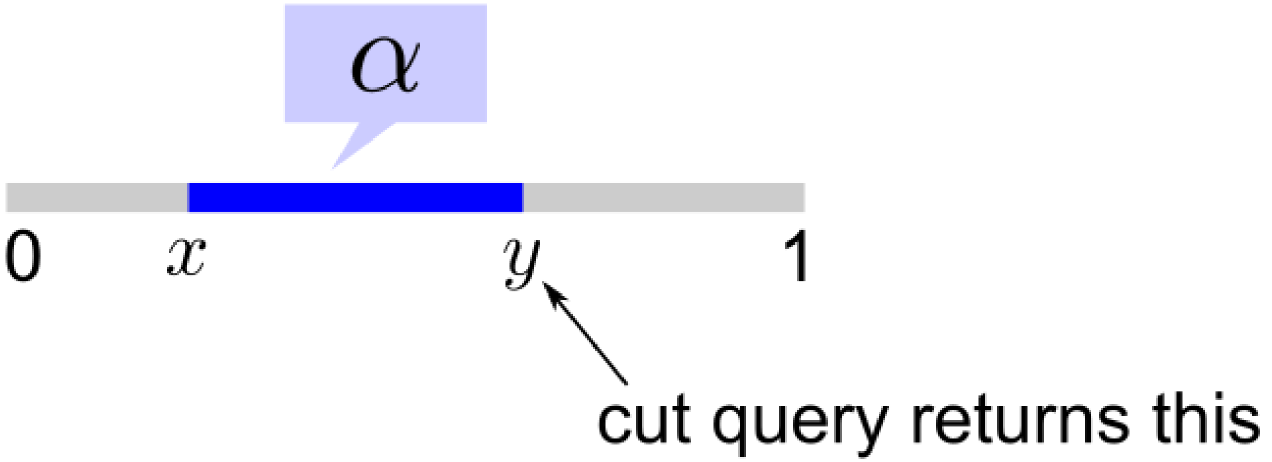


Robertson-Webb Query Model

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

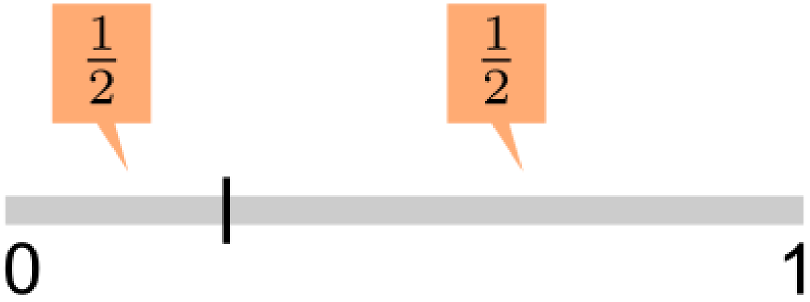
1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).

Cut and Choose

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

Cut and Choose

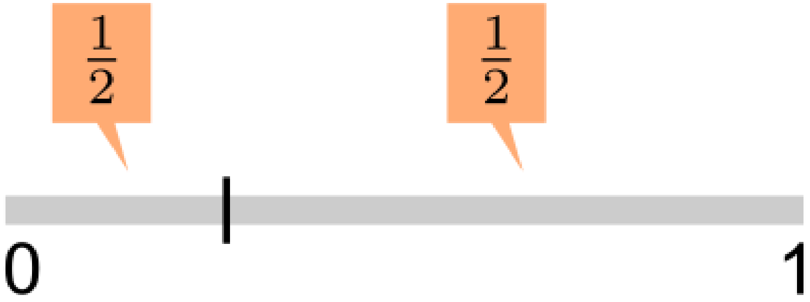
1. Agent 1 cuts the cake into two equally-valued pieces (as per v_1).



Cut and Choose

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

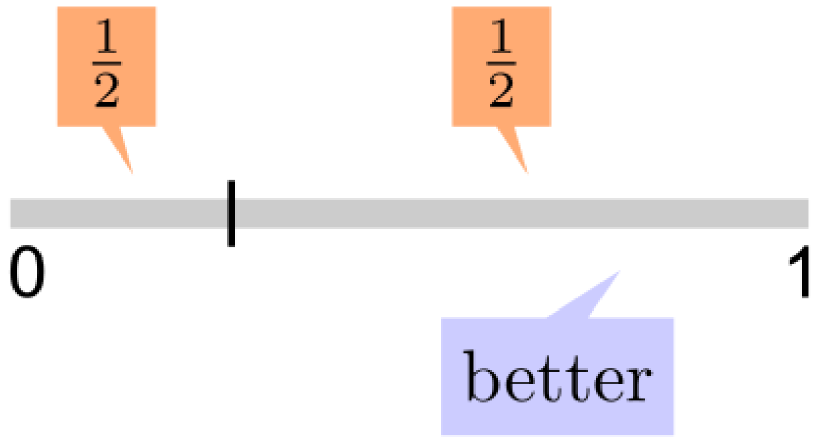
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Cut and Choose

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

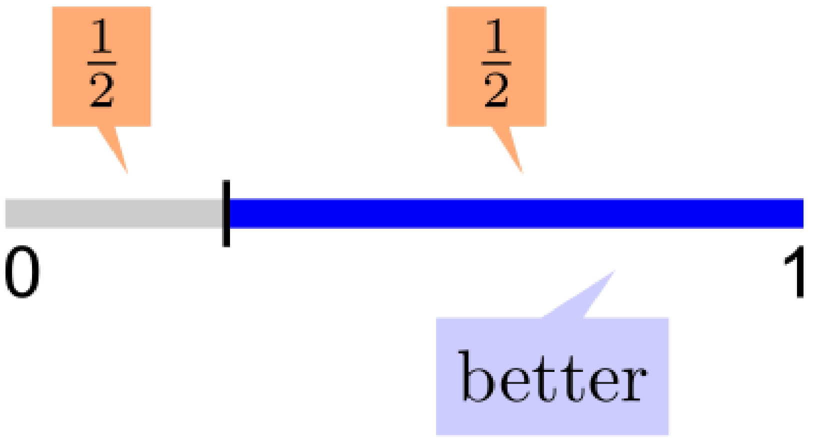
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Cut and Choose

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

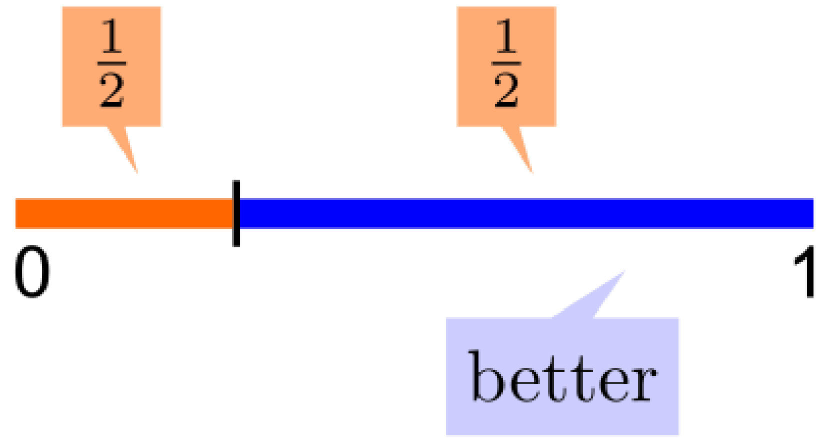
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



Cut and Choose

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

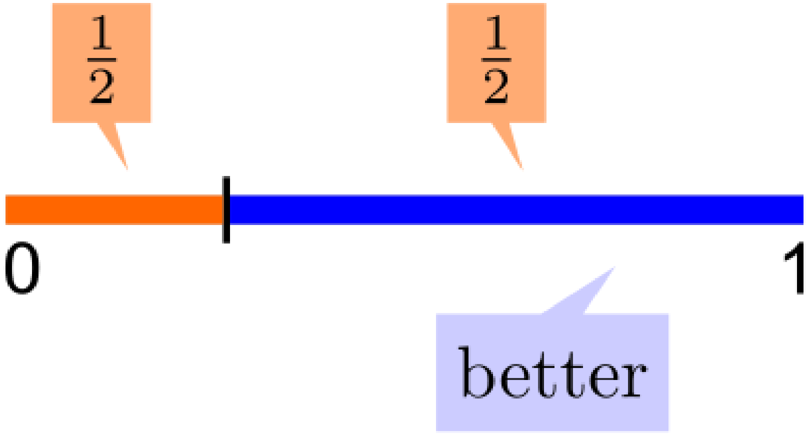
2. Agent 2 chooses its preferred piece (as per v_2), and agent 1 gets the remaining piece.



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.

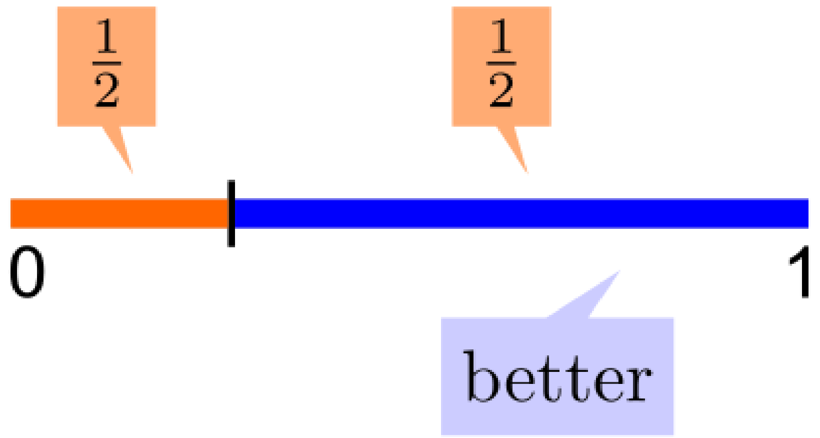


Is the cut-and-choose outcome proportional?

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.



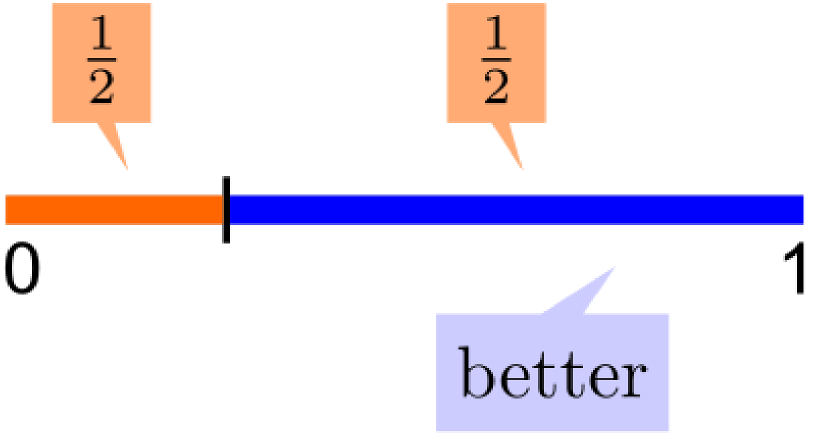
Is the cut-and-choose outcome proportional?

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.

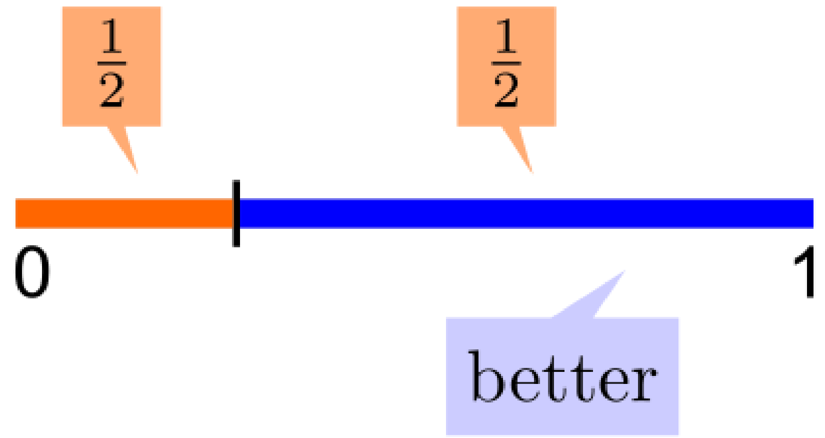


Is the cut-and-choose outcome envy-free?

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.



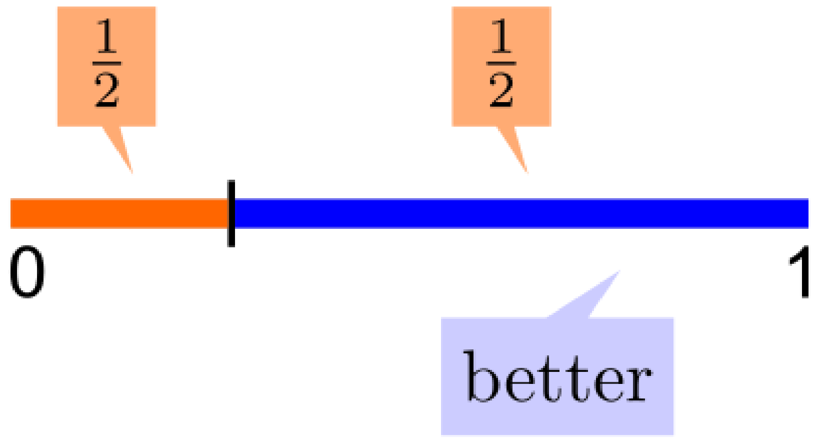
Is the cut-and-choose outcome envy-free?

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.

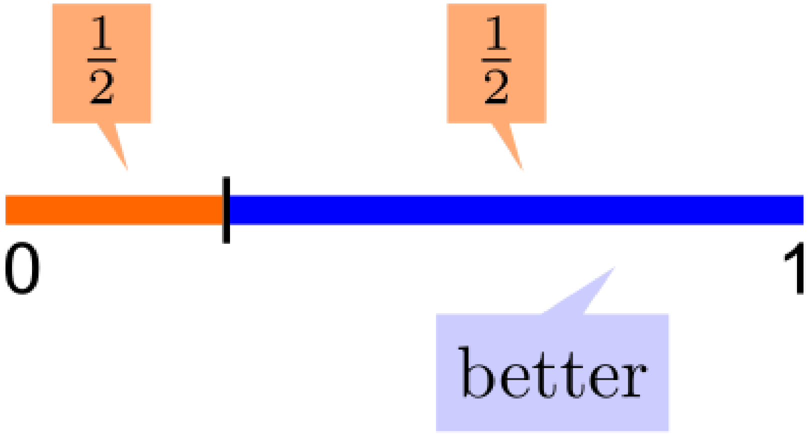


Can cut-and-choose be implemented in the Robertson-Webb model?

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.



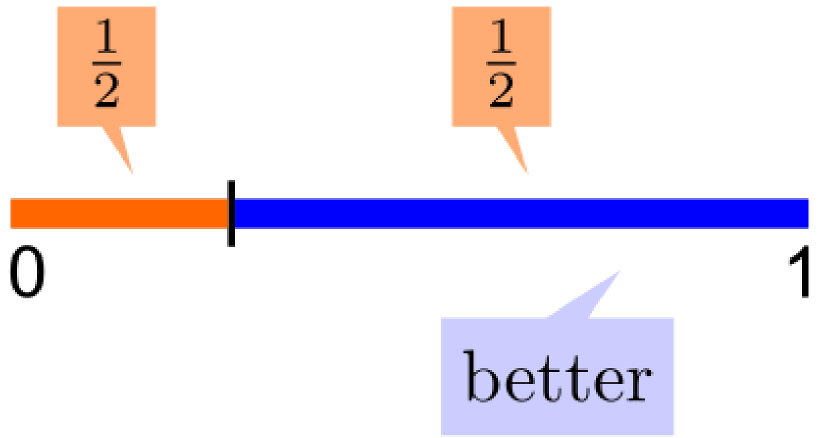
Can cut-and-choose be implemented in the Robertson-Webb model?

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

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?

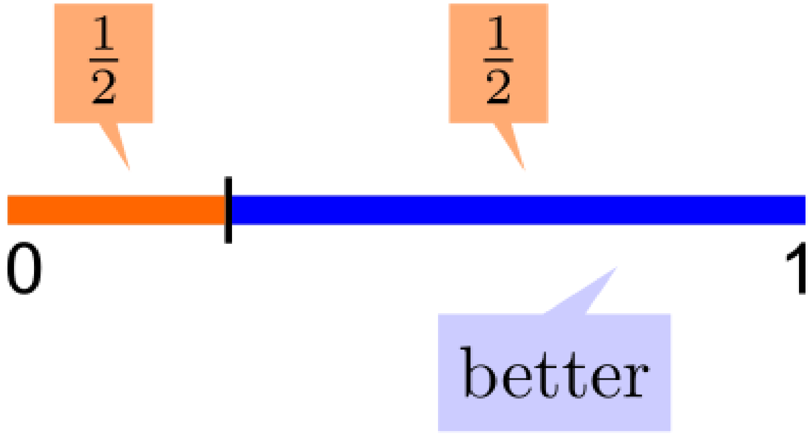
$$y = \texttt{cut}_1(0, 1/2)$$

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

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.

Dubins-Spanier Procedure

A proportional cake division protocol for any number of agents

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Dubins-Spanier Procedure

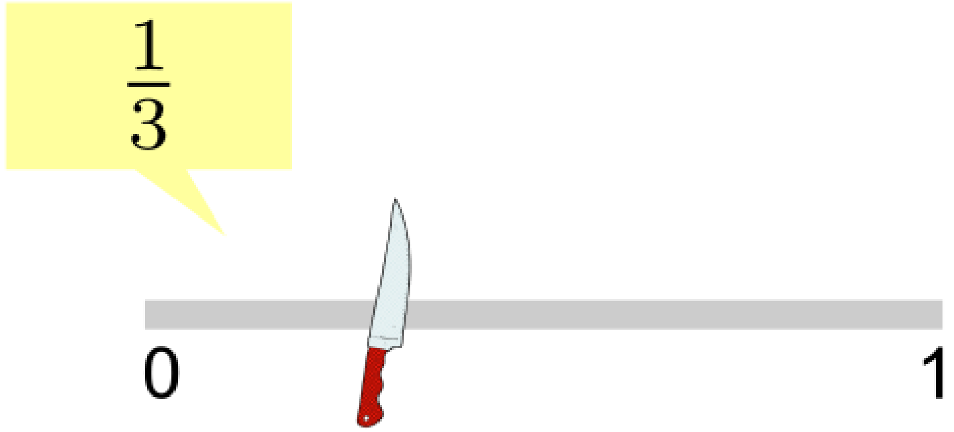
1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

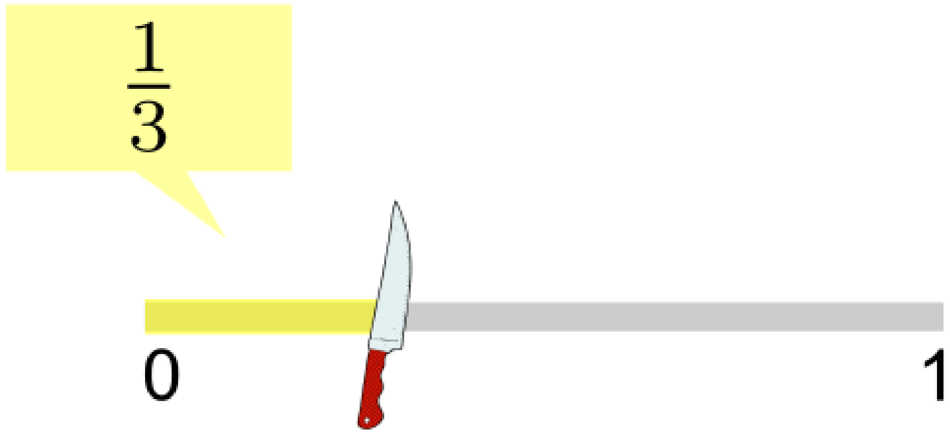
Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.



Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

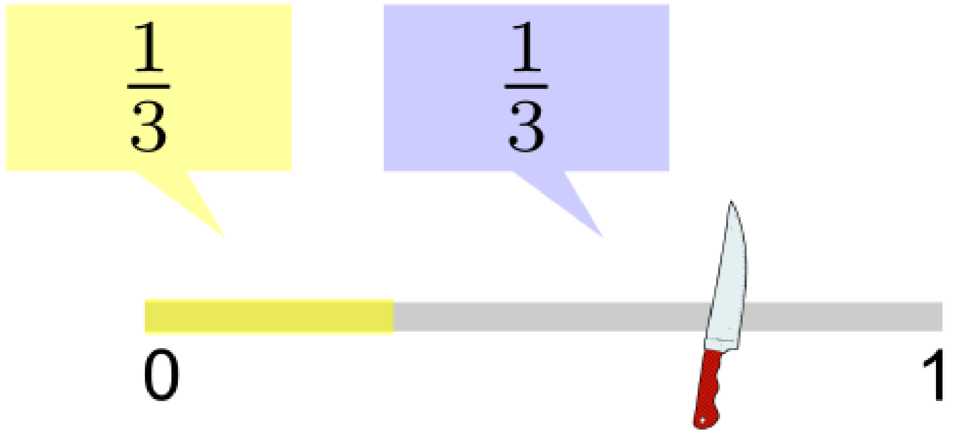


Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

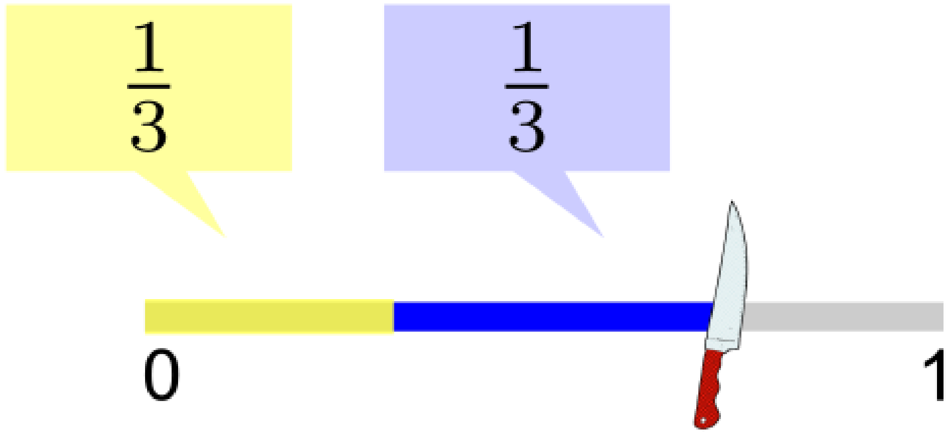
Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.



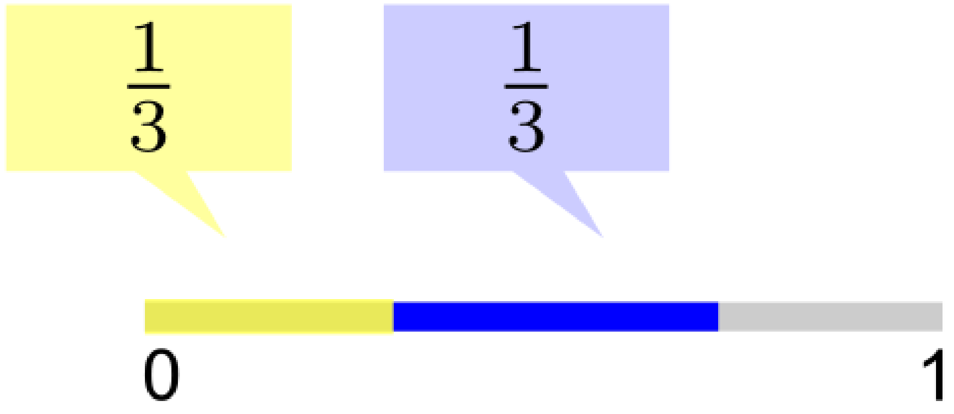
Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.



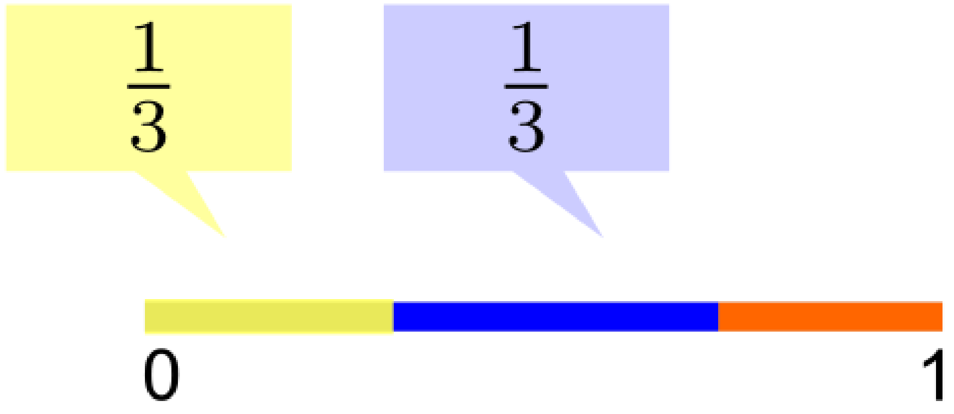
Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.



Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.



Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Why is the resulting allocation proportional?

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Why is the resulting allocation proportional?

Every agent except for the last one gets *exactly* 1/n.
The last agent gets *at least* 1/n.

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Can this procedure be implemented in the Robertson-Webb model?

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Can this procedure be implemented in the Robertson-Webb model?

Yes!

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Query complexity in the Robertson-Webb model?

Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

Query complexity in the Robertson-Webb model?

$\mathcal{O}(n^2)$ queries (Exercise)

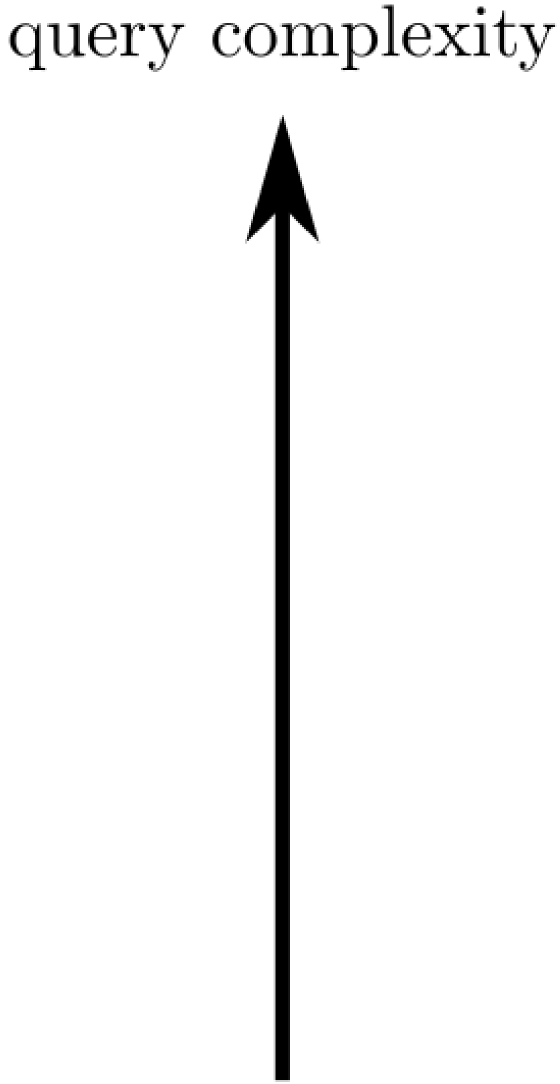
Dubins-Spanier Procedure

1. A referee gradually moves a knife from left to right.
2. As soon as the piece to the left of the knife is worth 1/n to some agent, it shouts "stop".
3. The said agent is assigned the left-side piece and is removed.
4. The procedure repeats with the remaining agents.

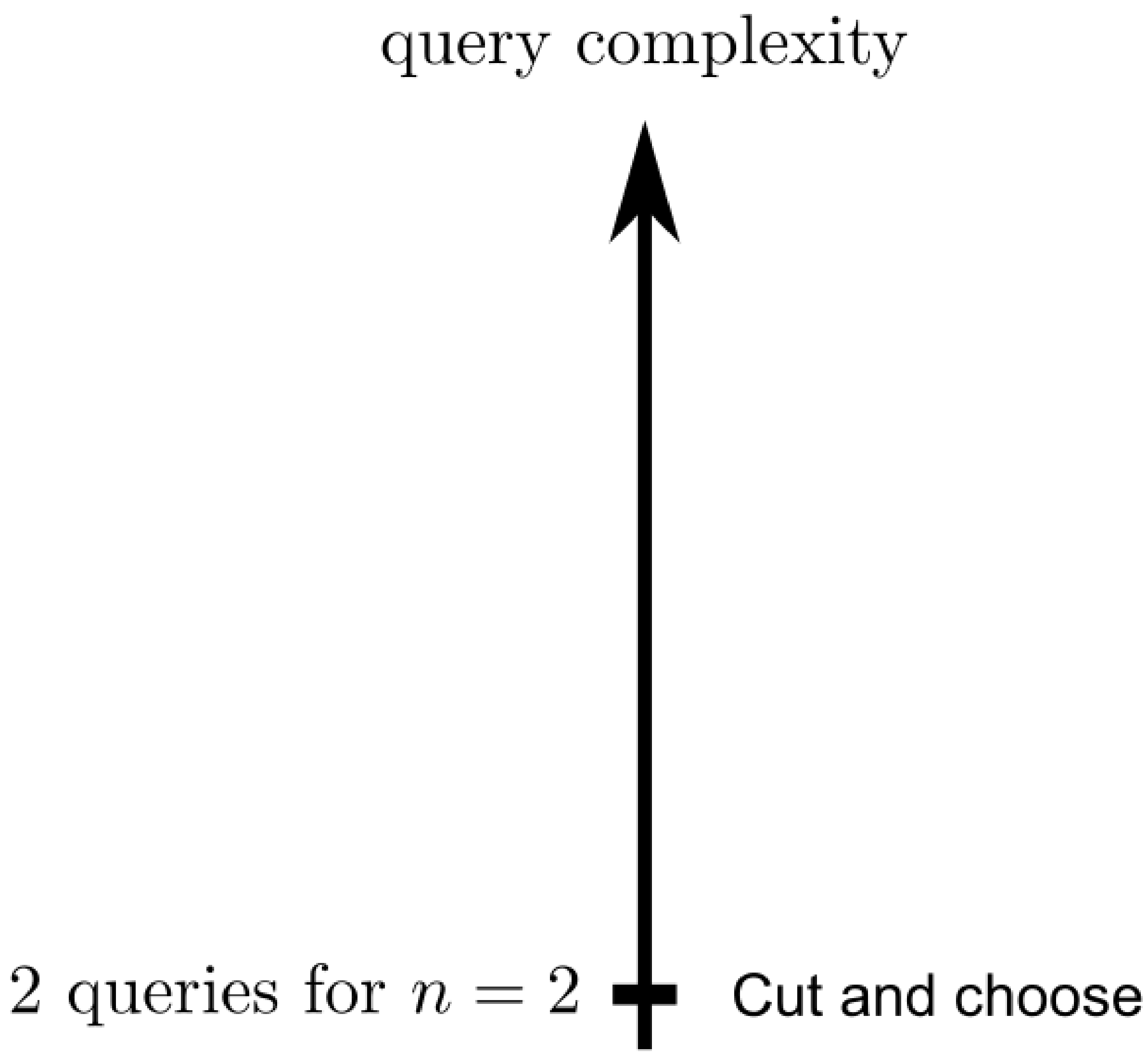
For *n* agents, a proportional cake division can be computed using $O(n^2)$ queries.

The Story of Proportionality

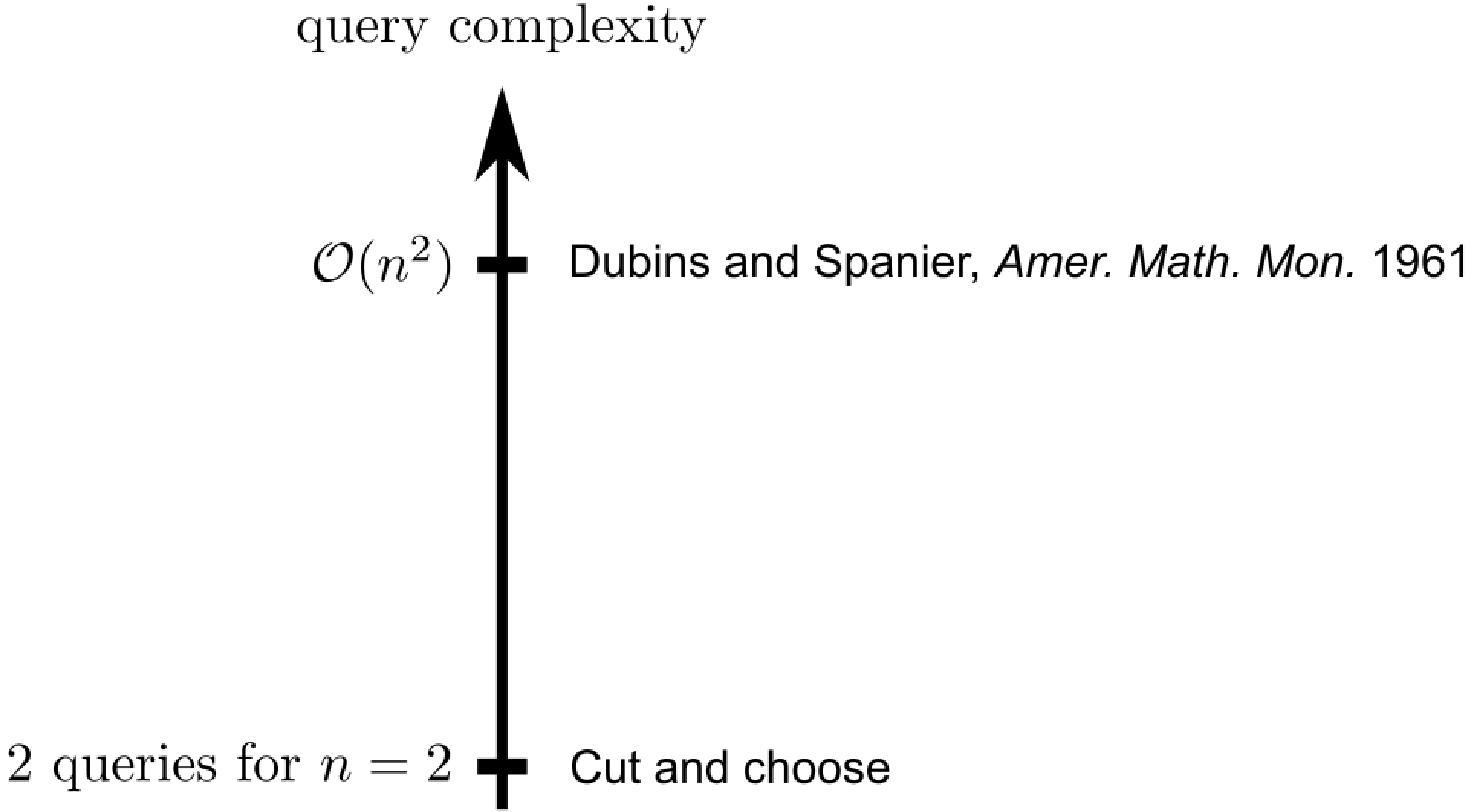
The Story of Proportionality



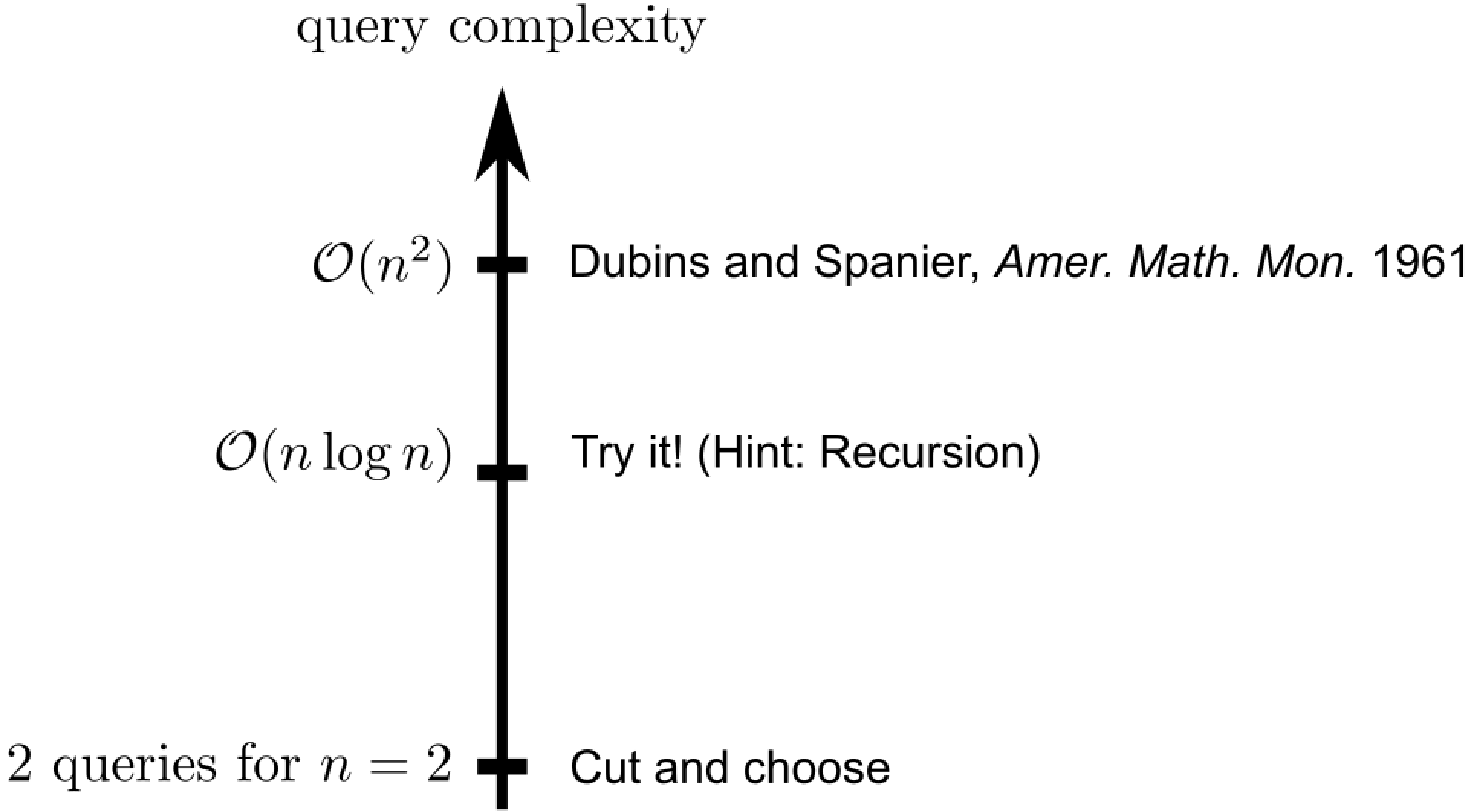
The Story of Proportionality



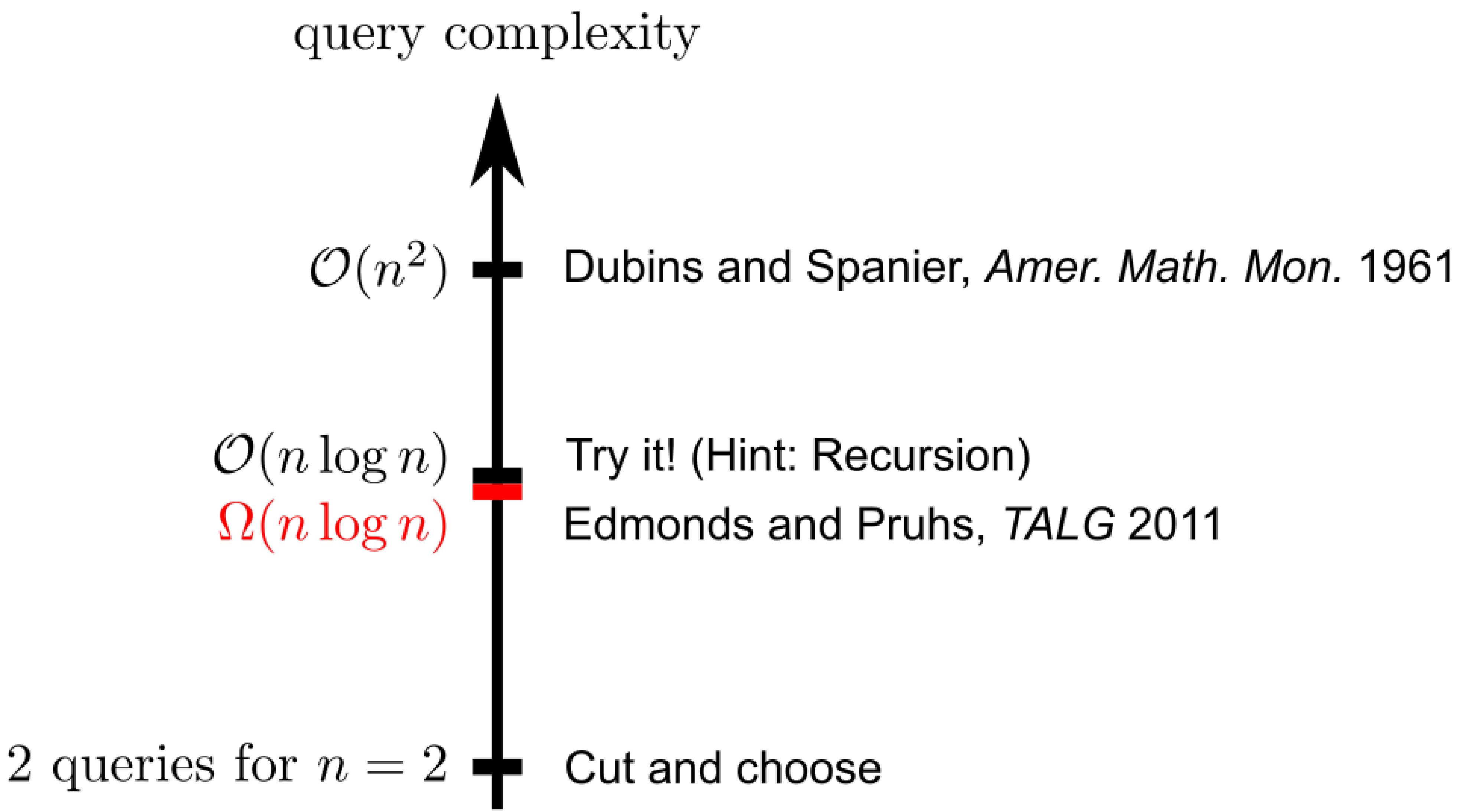
The Story of Proportionality



The Story of Proportionality



The Story of Proportionality



The Story of Envy-freeness

Selfridge-Conway Procedure

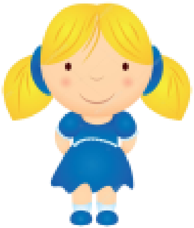
An envy-free cake division protocol for three agents

Envy-free for three

Envy-free for three

A

B



C

Envy-free for three

A

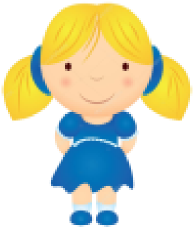
B

C

Envy-free for three

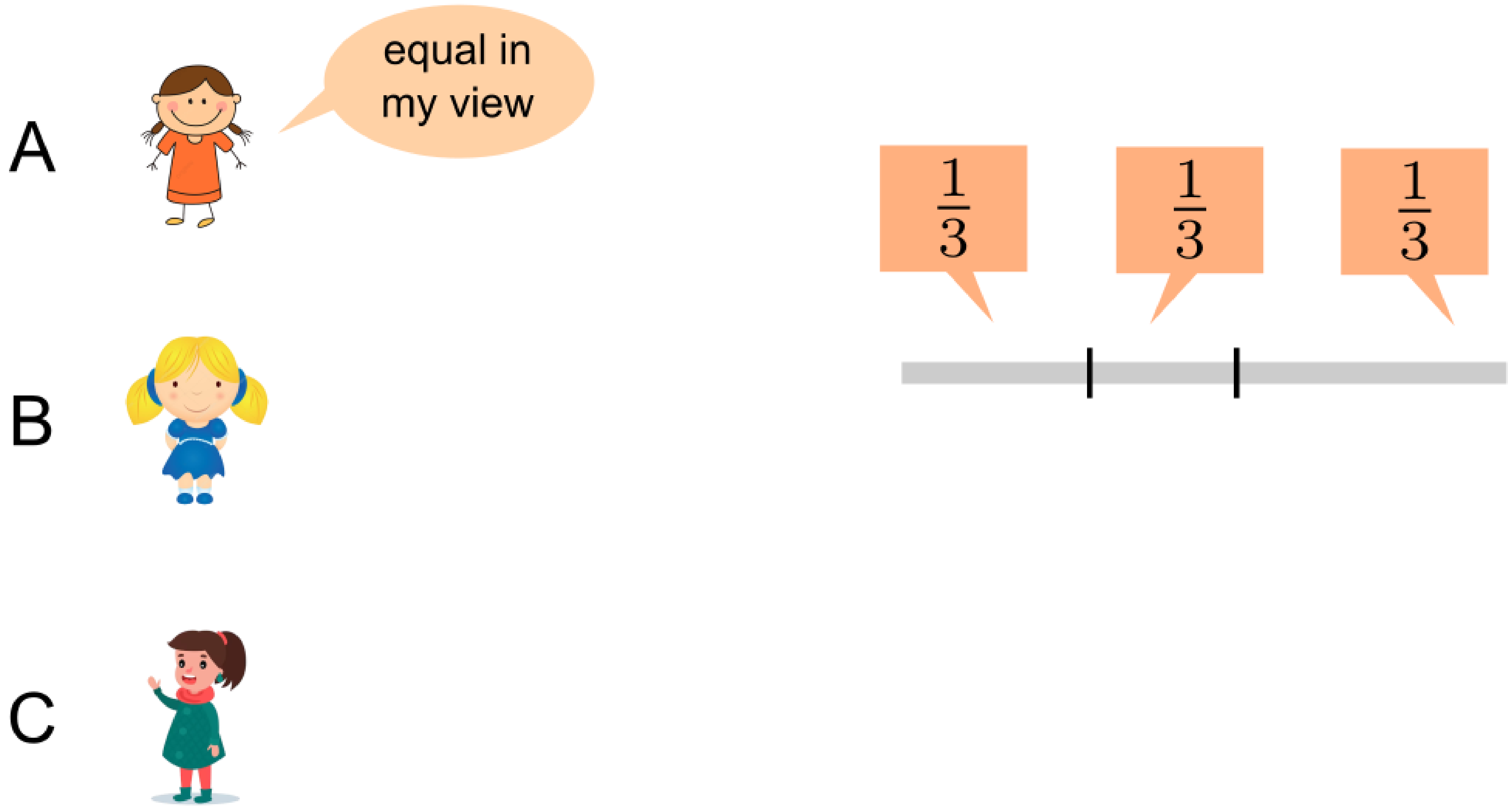
A

B



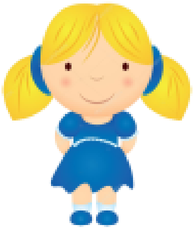
C

Envy-free for three



Envy-free for three

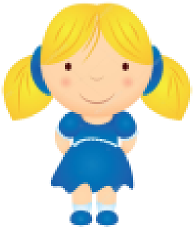
A 

B 

C 

Envy-free for three

A 

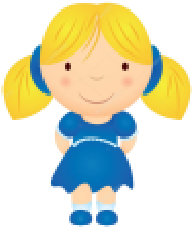
B 

C 

Envy-free for three

A

B



two-way tie

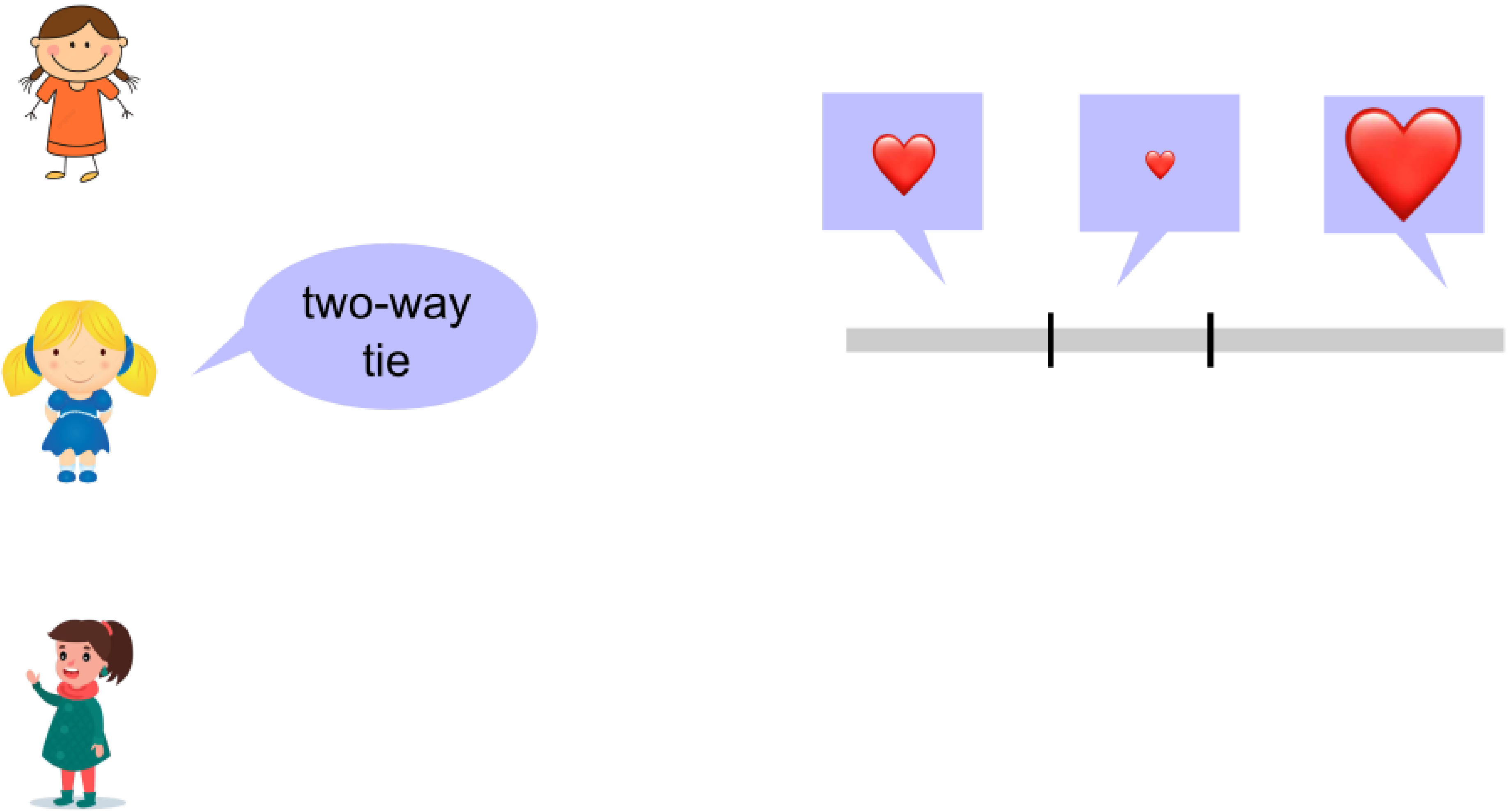
C

Envy-free for three

A

B

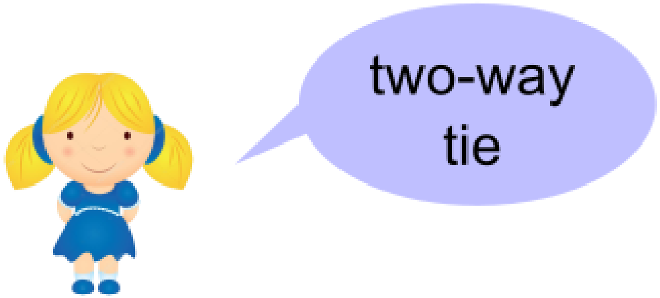
C



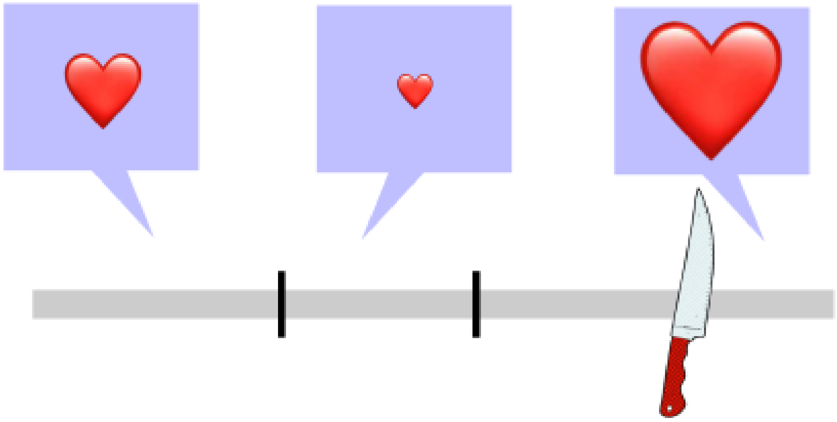
Envy-free for three

A

B



C

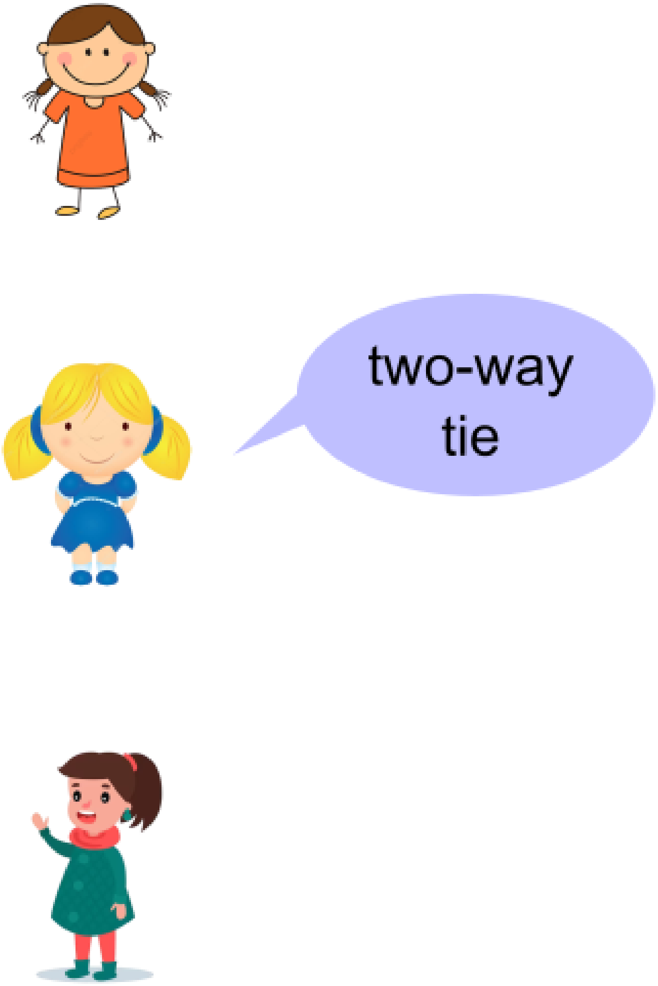


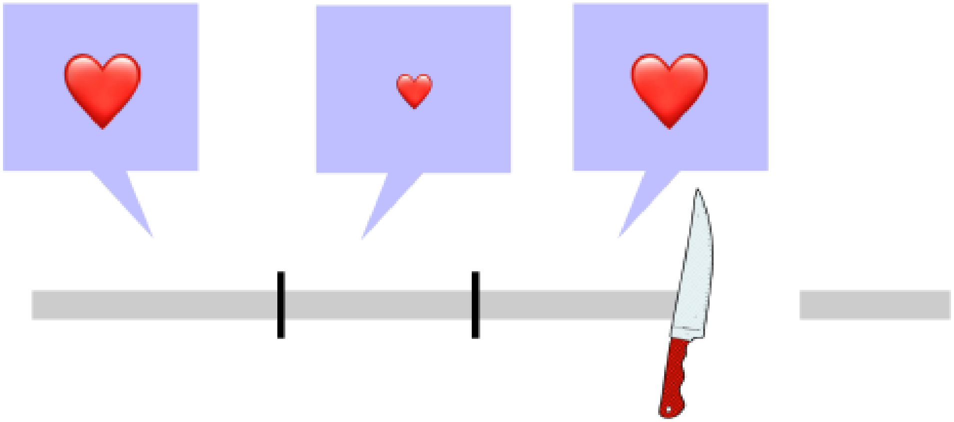
Envy-free for three

A

B

C

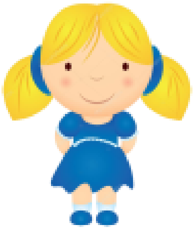




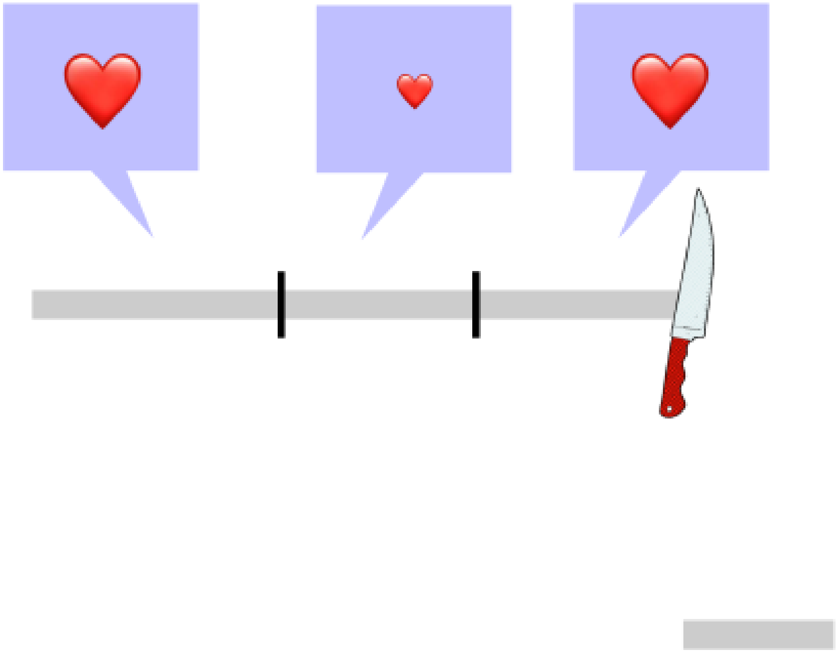
Envy-free for three

A

B



two-way tie



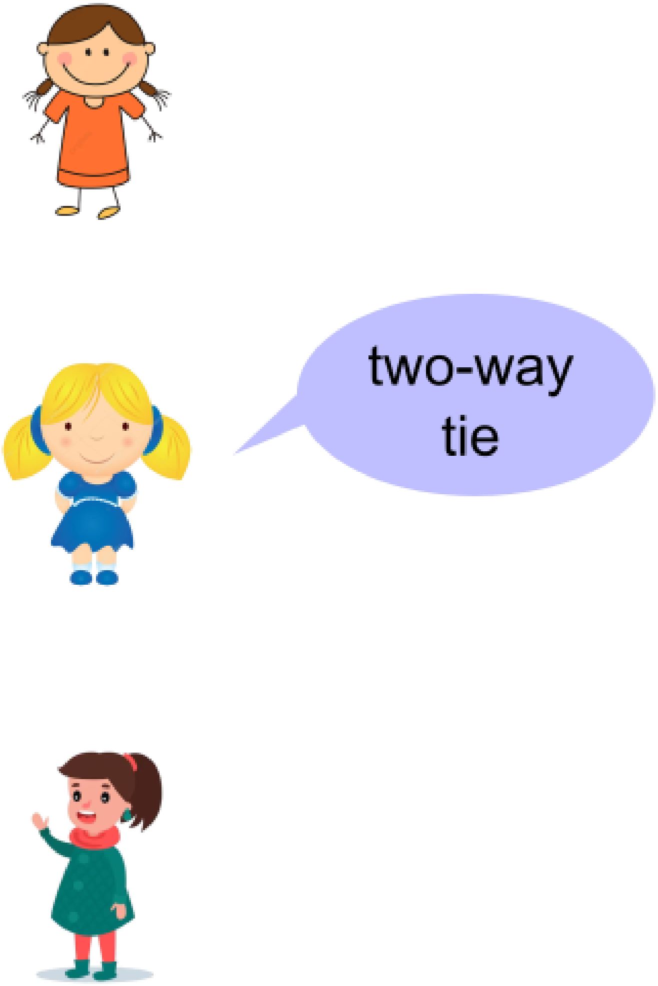
C

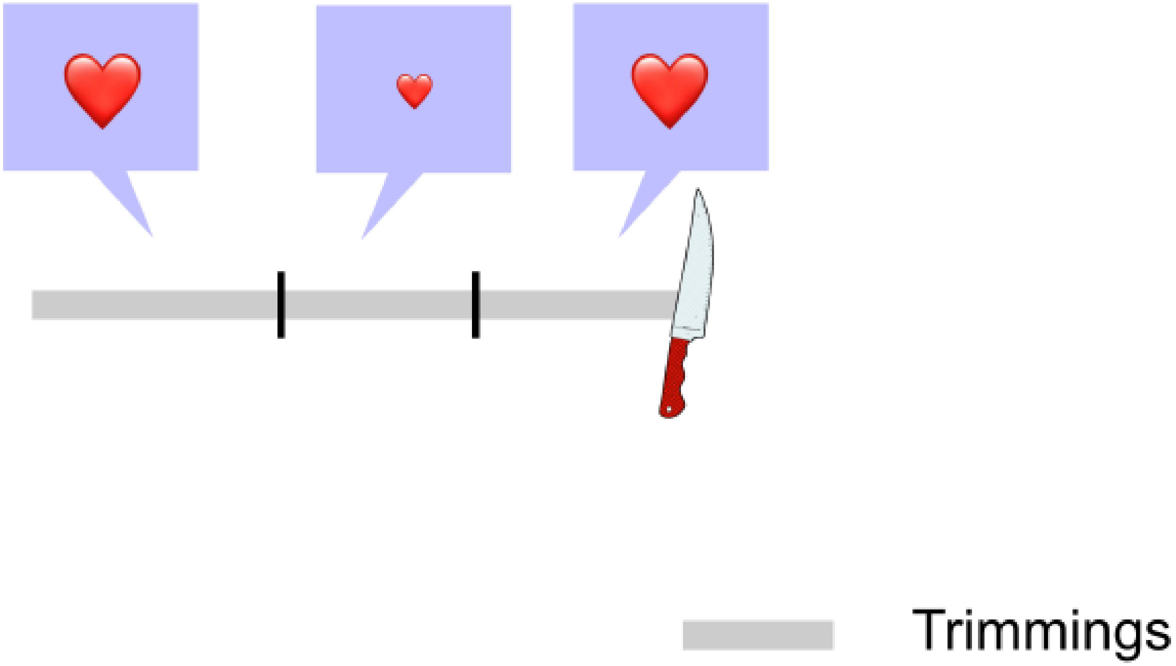
Envy-free for three

A

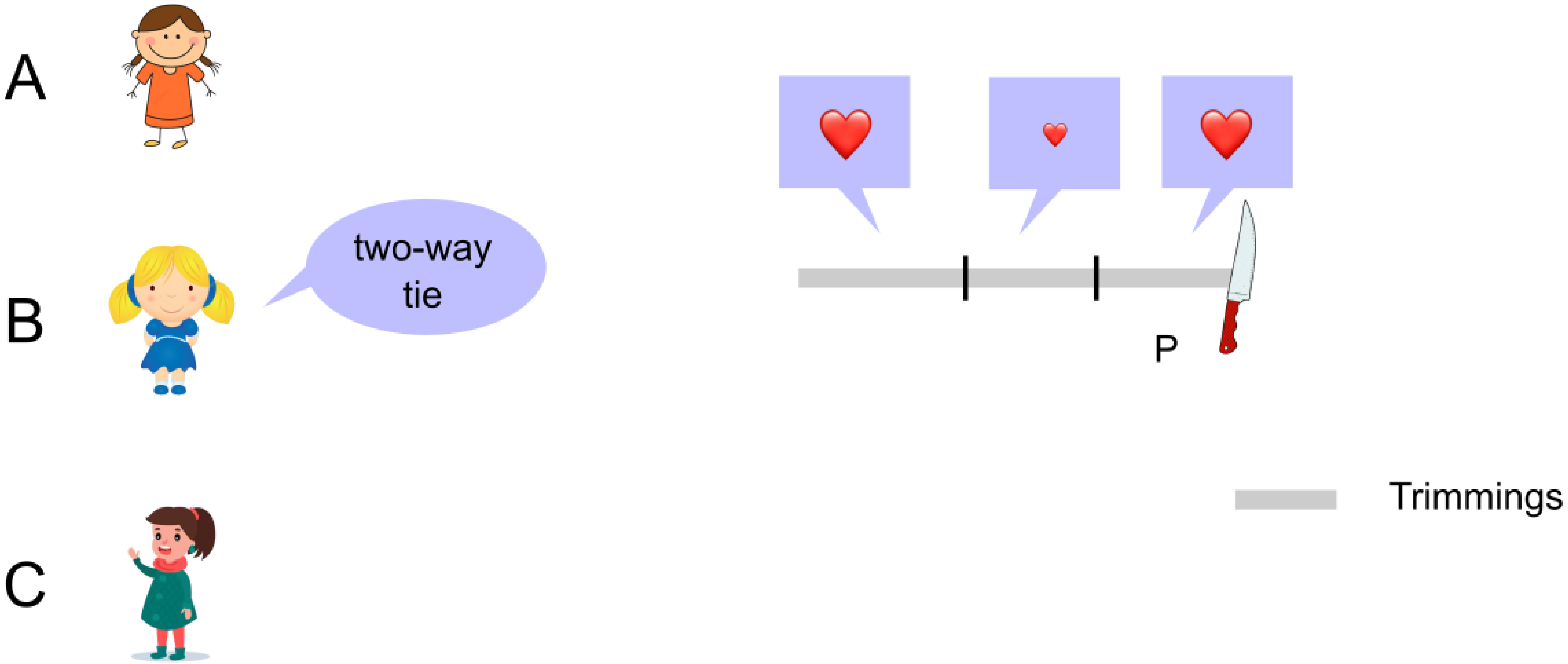
B

C





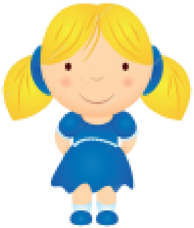
Envy-free for three



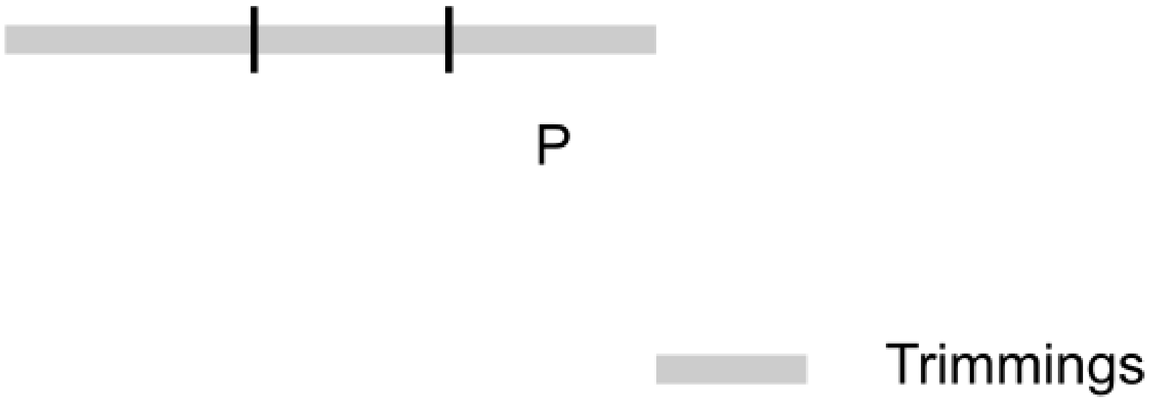
Envy-free for three

A

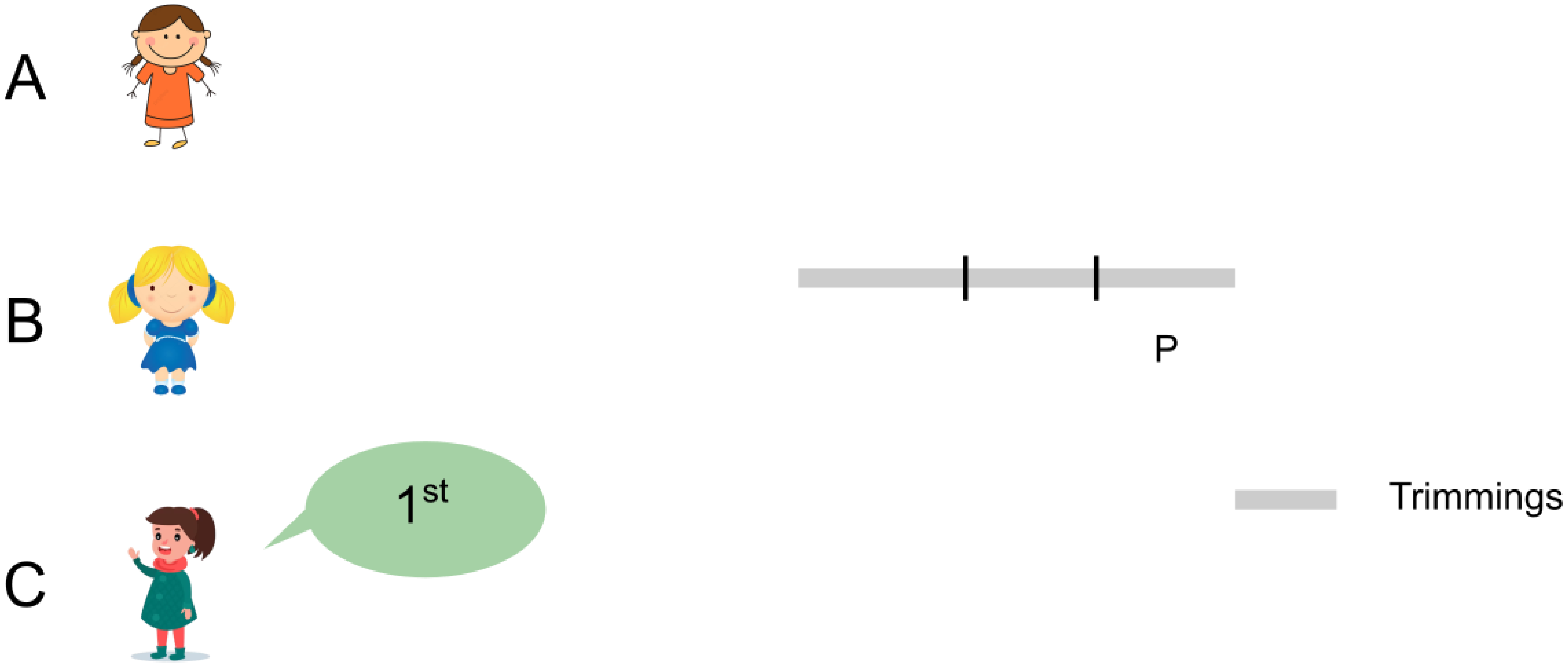
B



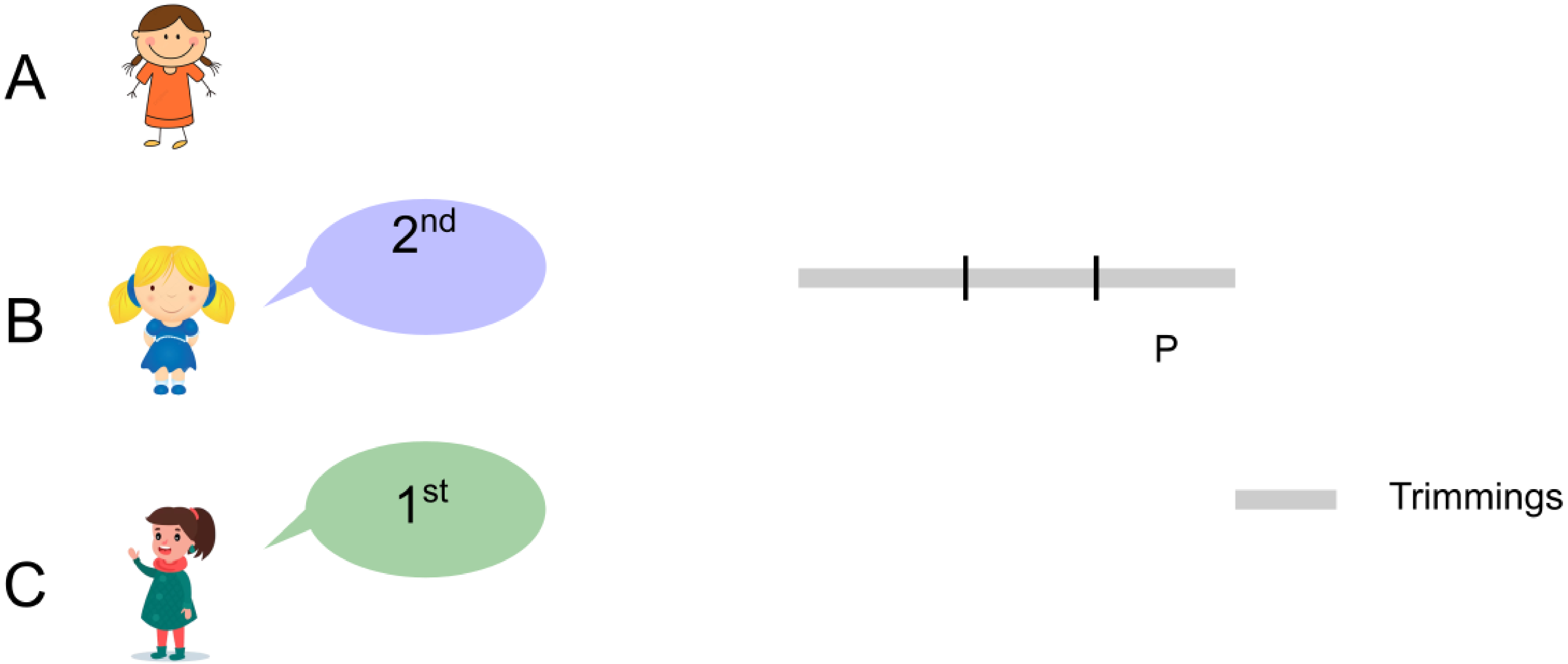
C



Envy-free for three



Envy-free for three

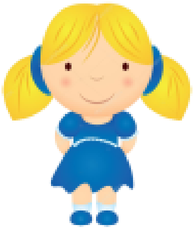


Envy-free for three

A

3rd

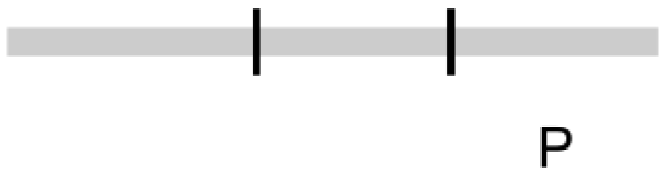
B



2nd

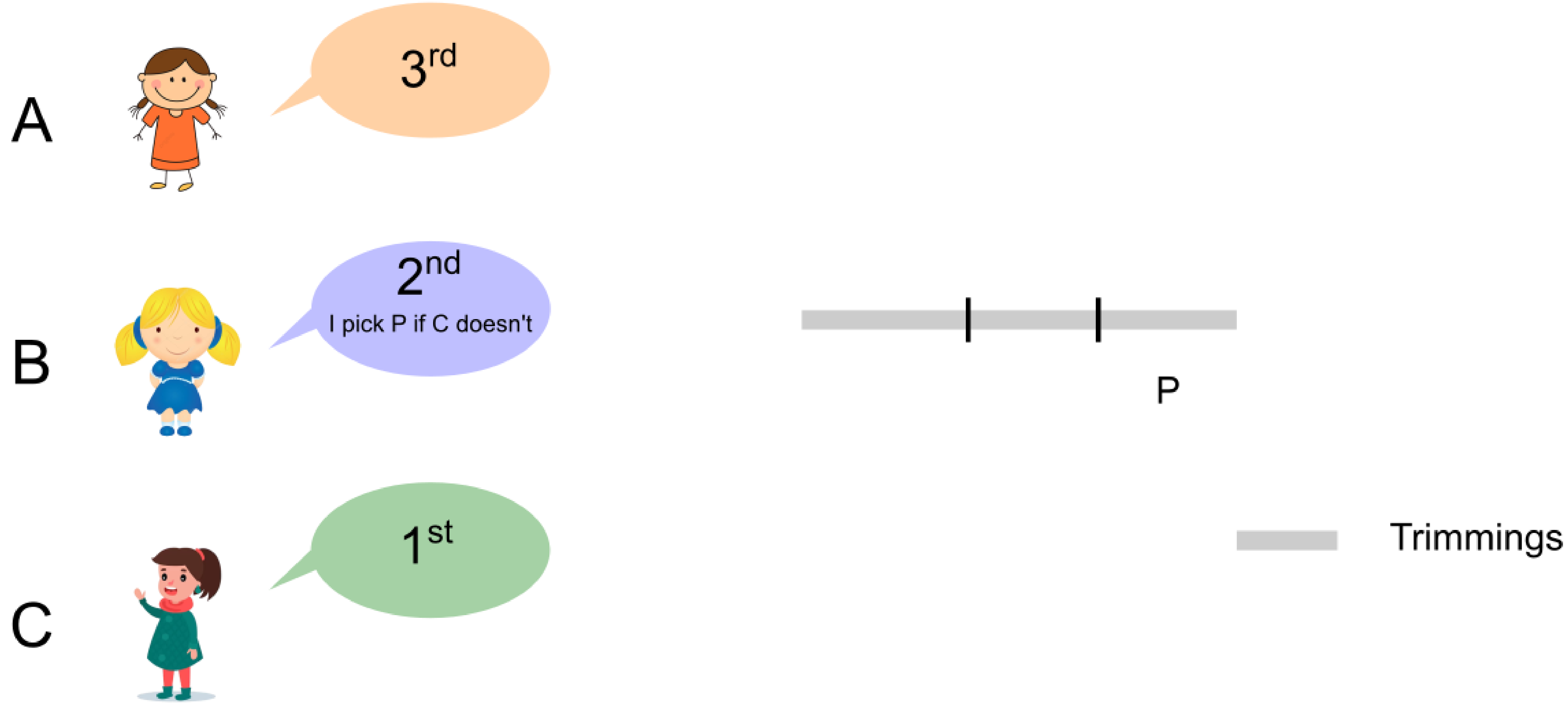
C

1st



Trimmings

Envy-free for three

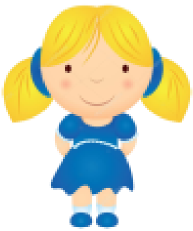


Envy-free for three

A

3rd

B

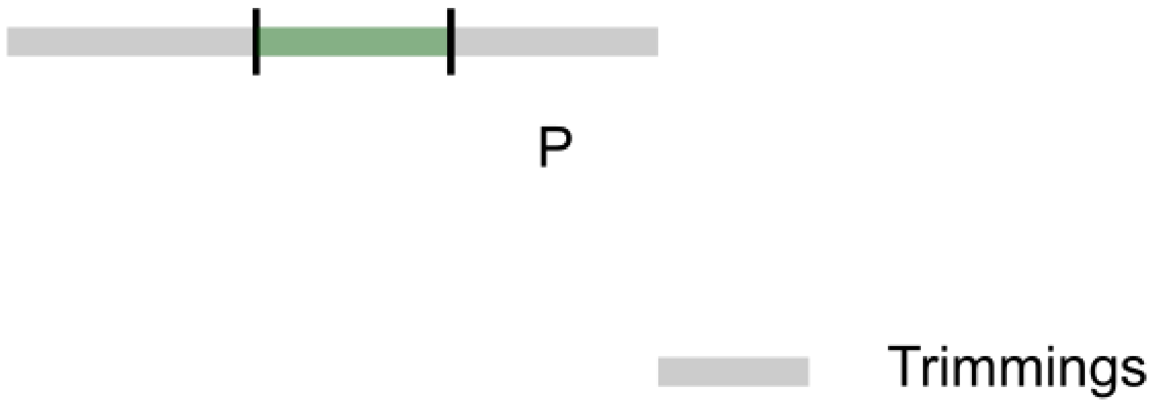


2nd

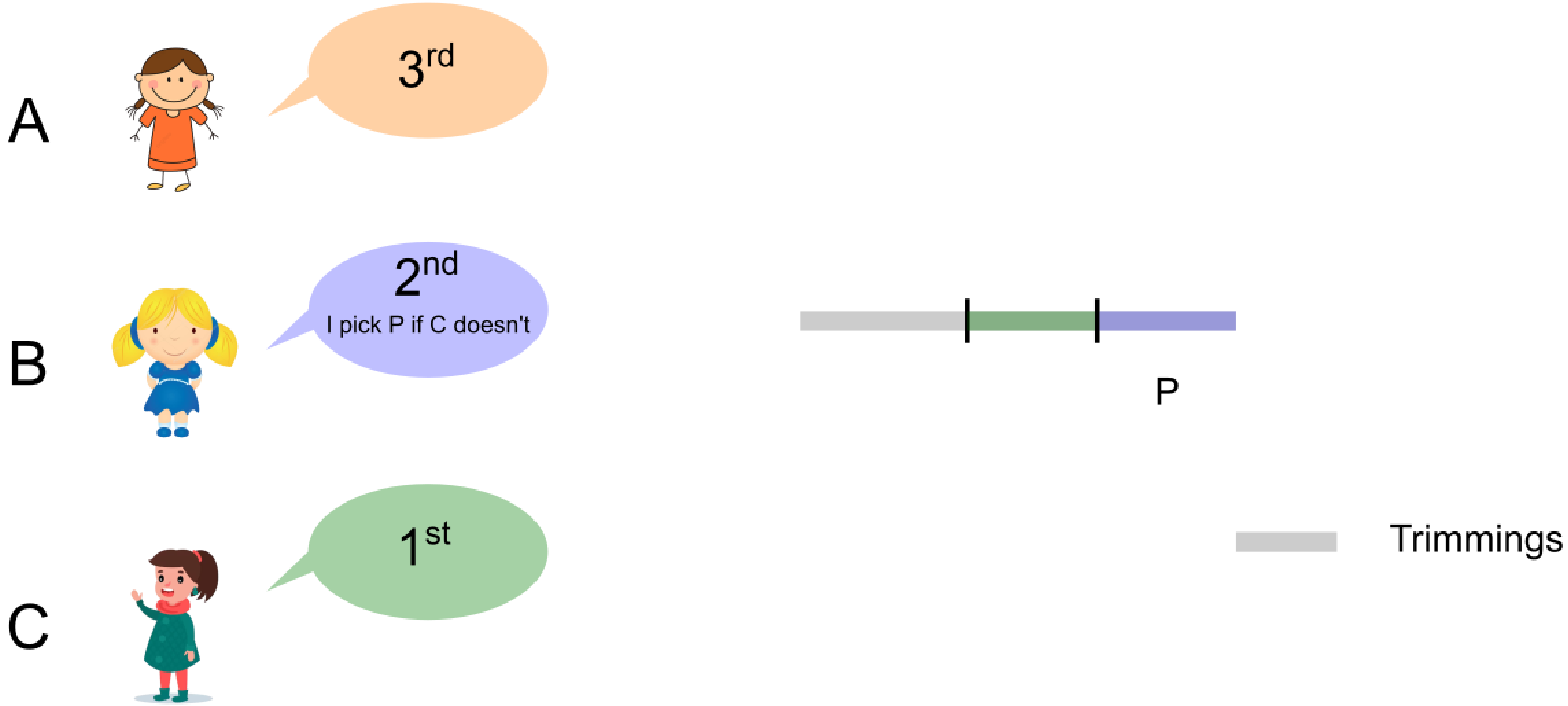
I pick P if C doesn't

C

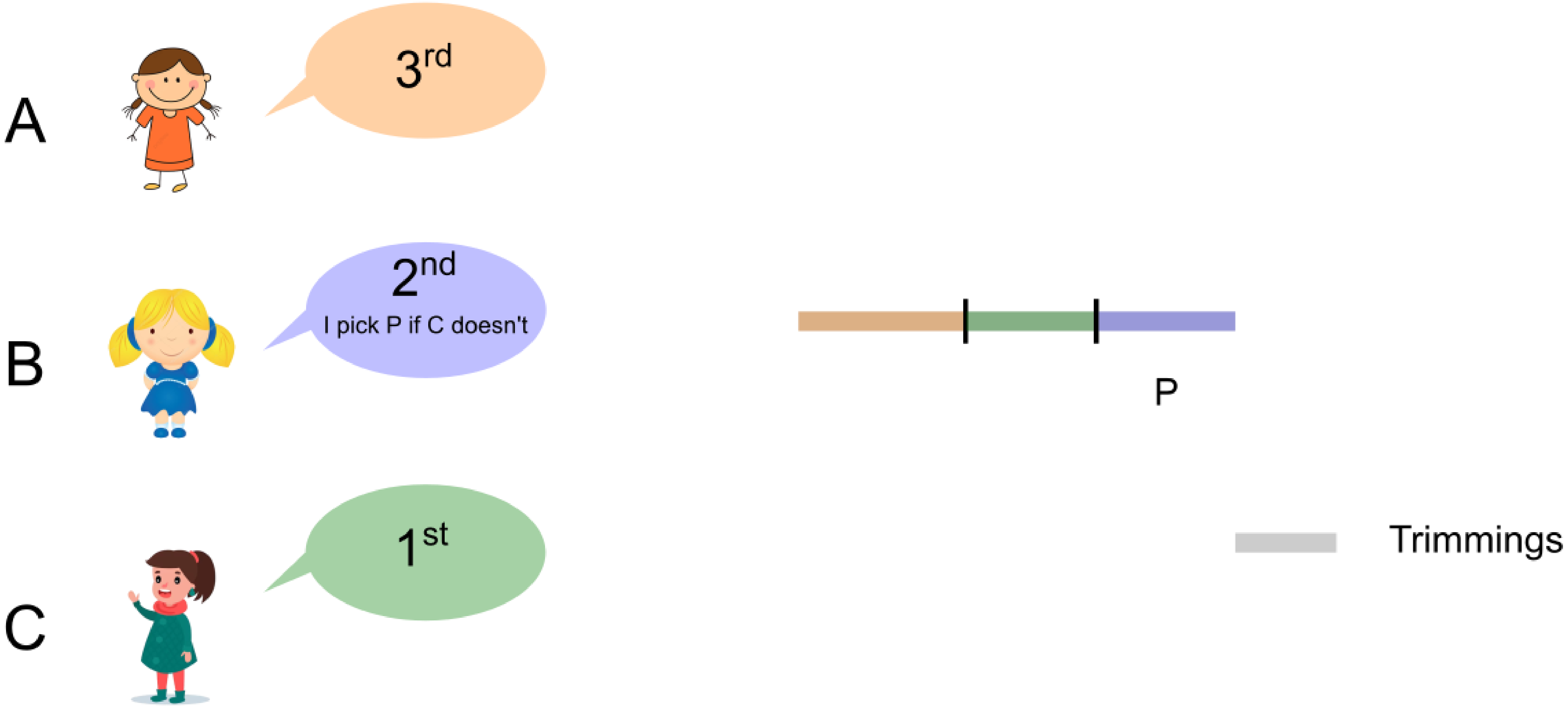
1st



Envy-free for three



Envy-free for three

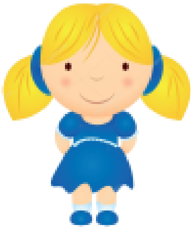


Envy-free for three

A

EF because of equal cuts

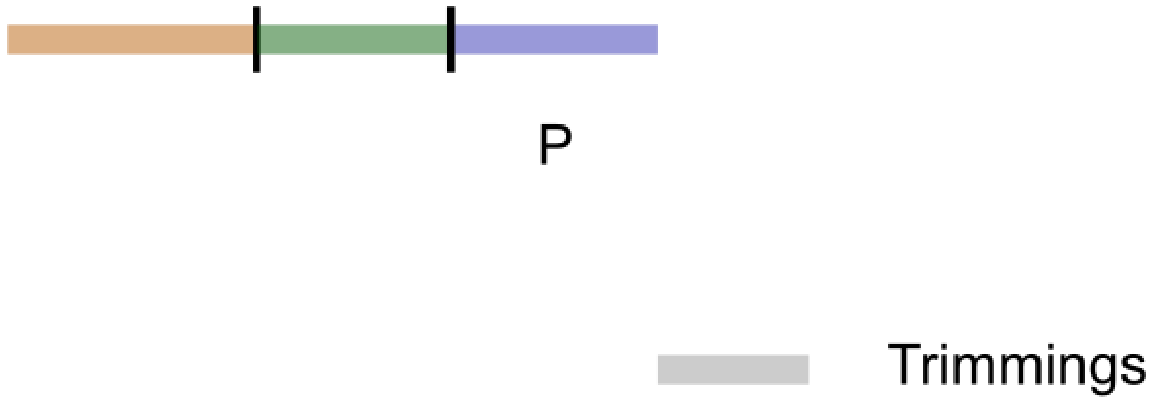
B



EF because of two-way tie

C

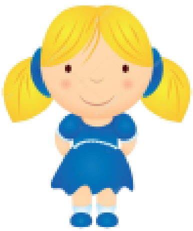
EF because I picked first



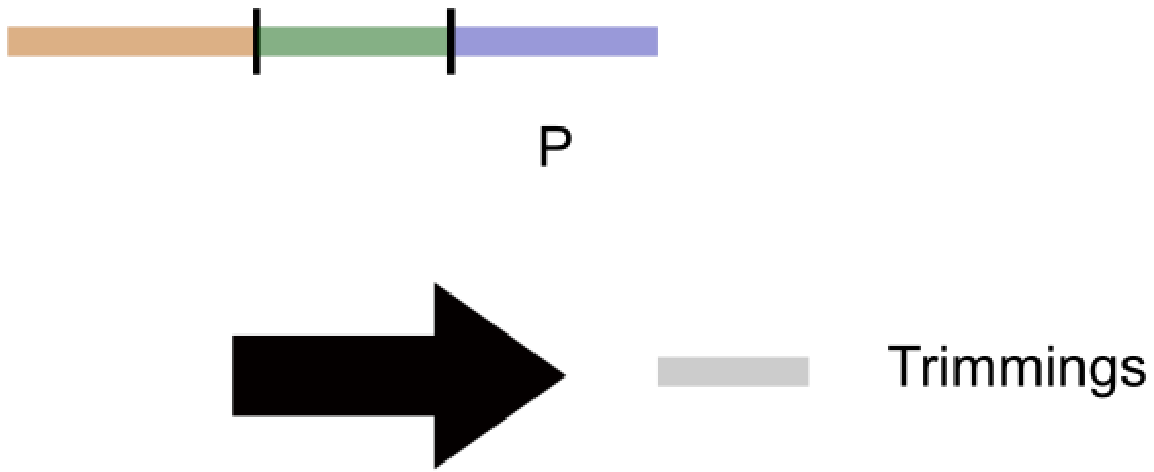
Envy-free for three

A

B



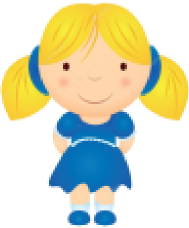
C



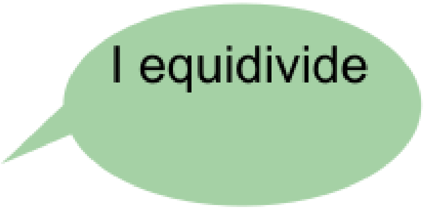
Envy-free for three

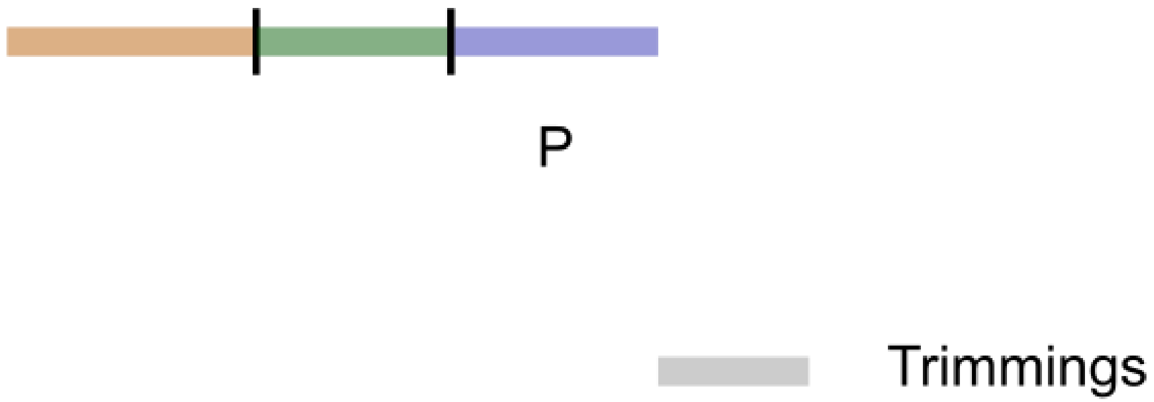
A

B

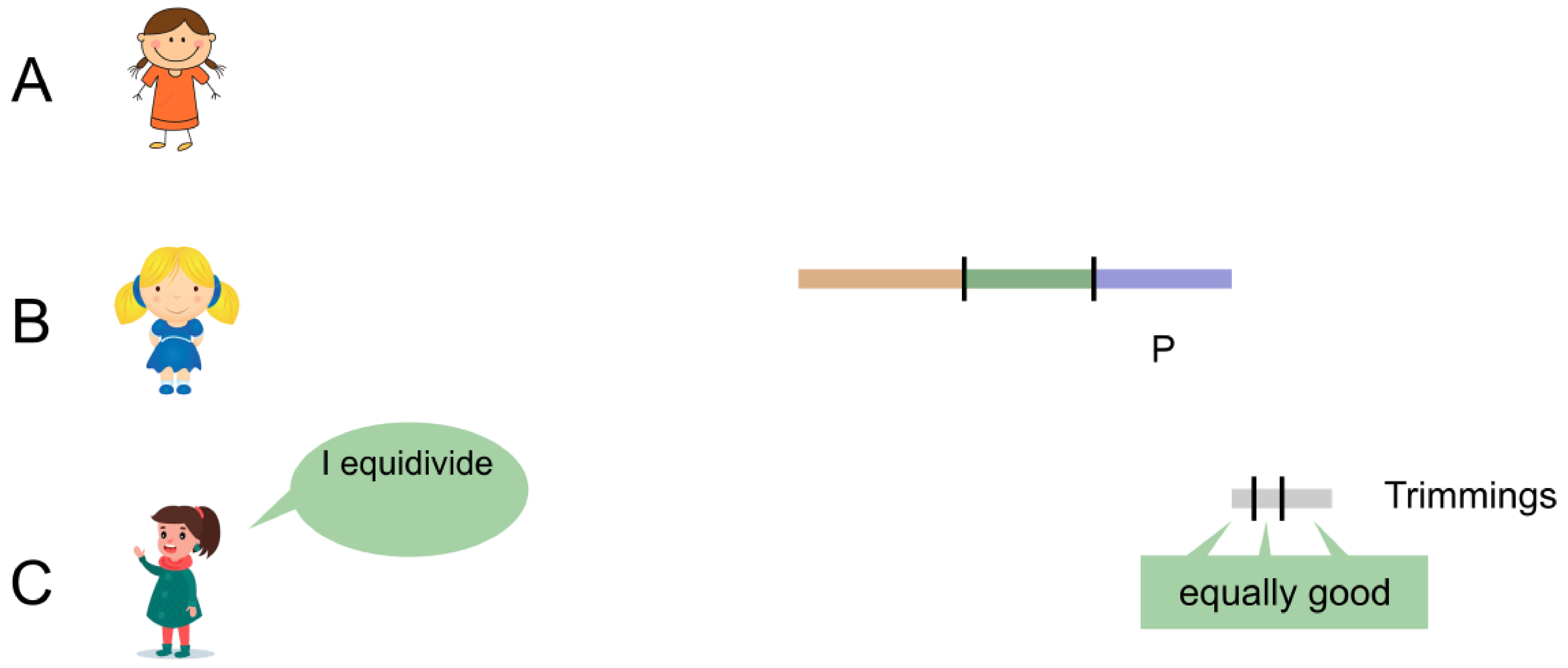


C

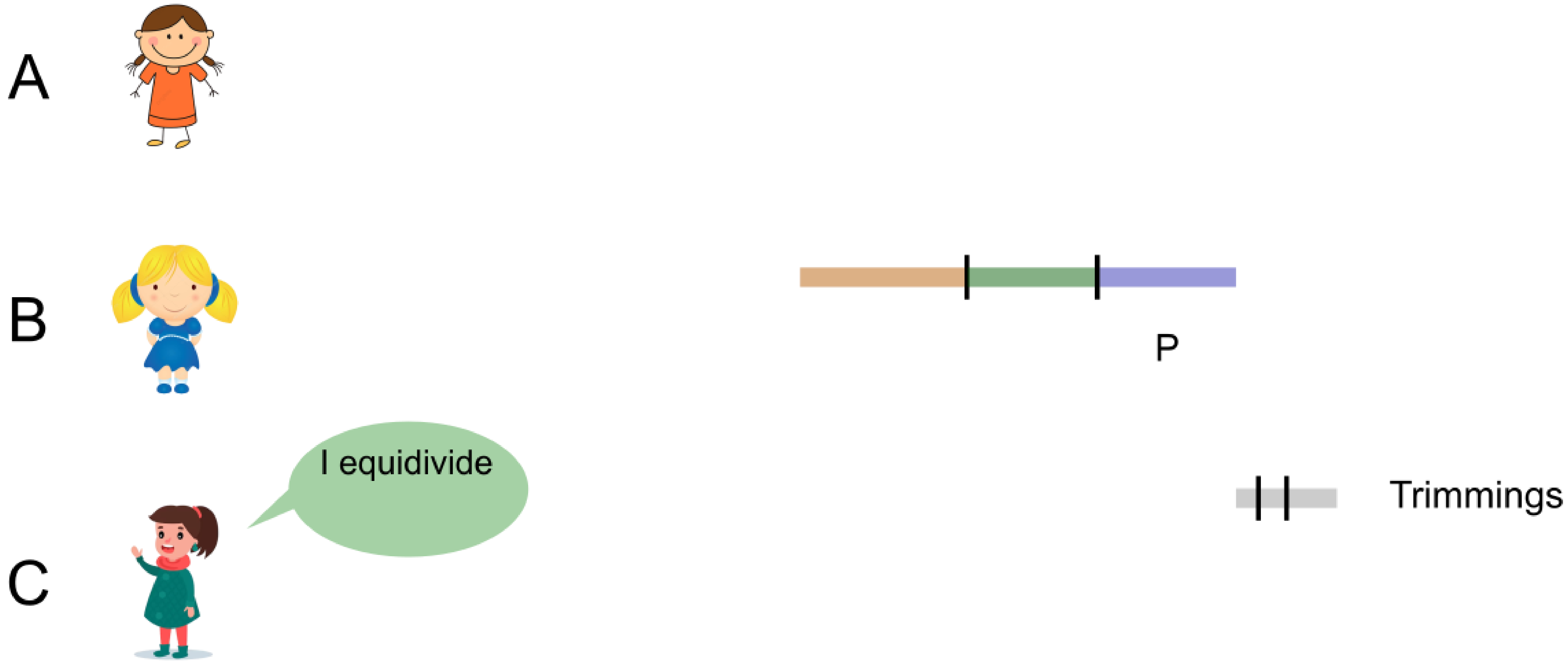




Envy-free for three



Envy-free for three

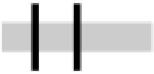


Envy-free for three

A

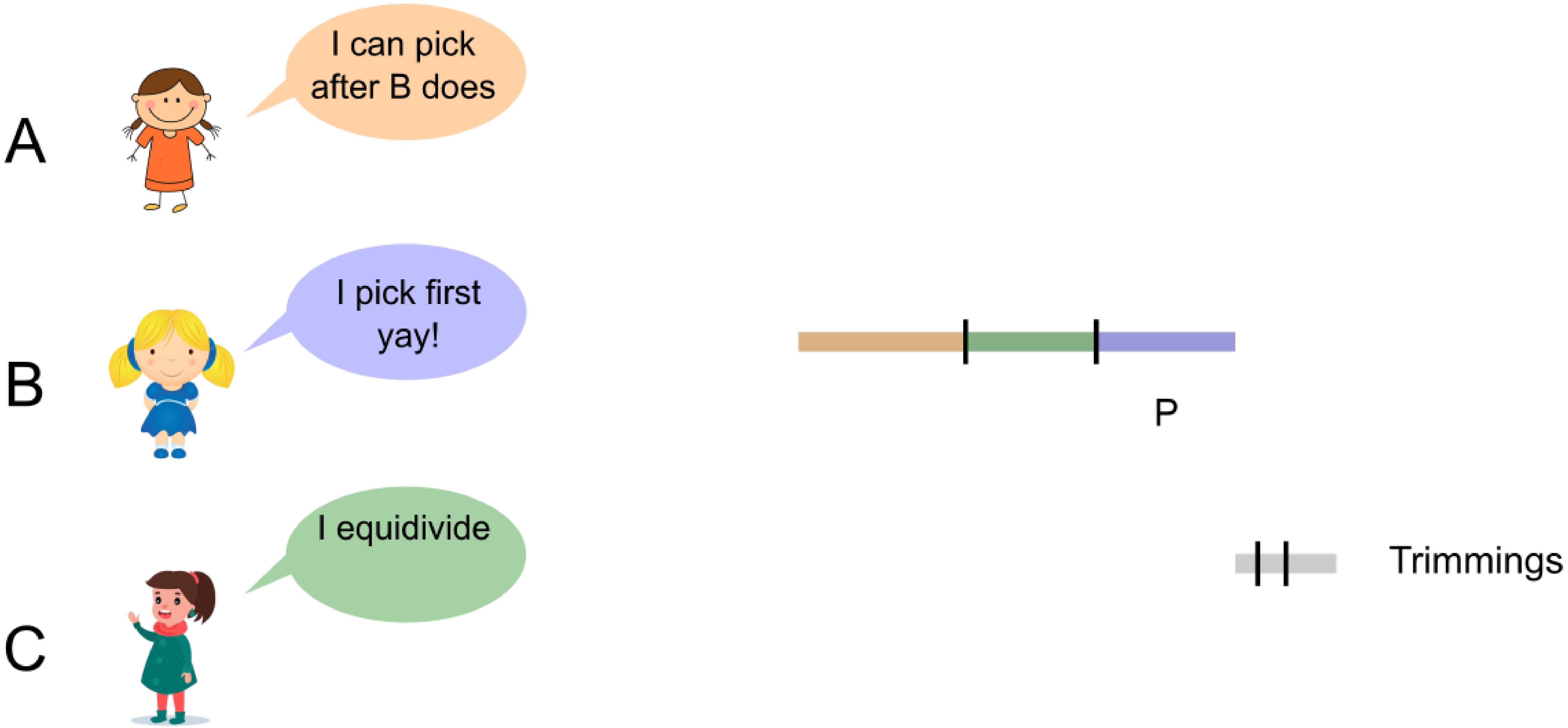
B

C



Trimmings

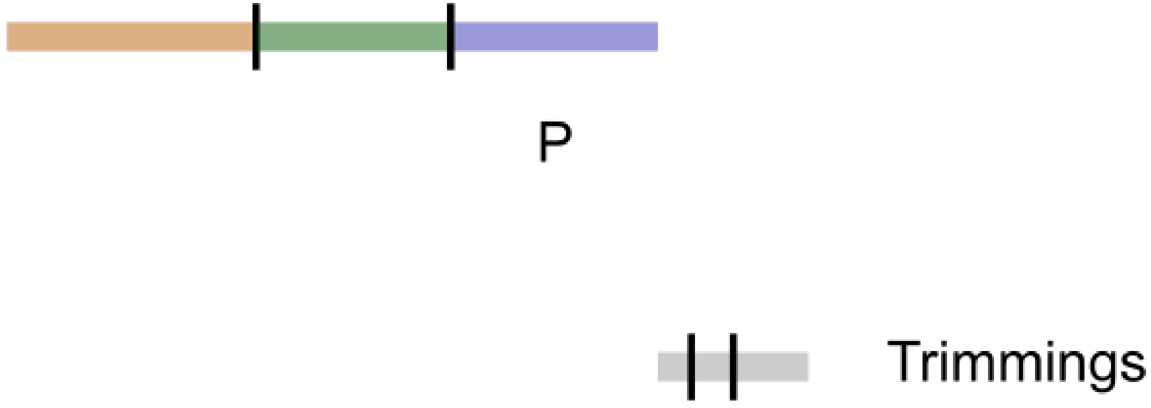
Envy-free for three



Envy-free for three

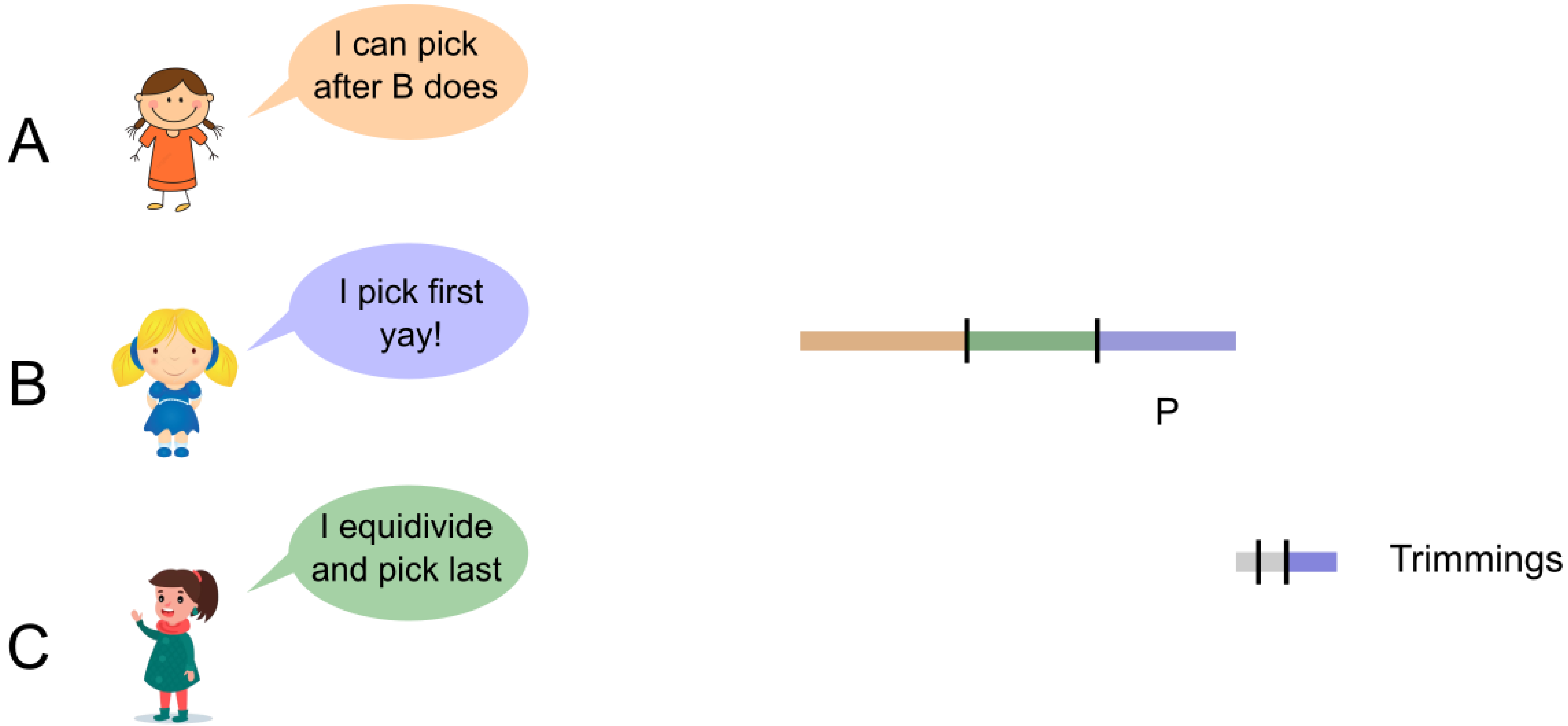
A

B

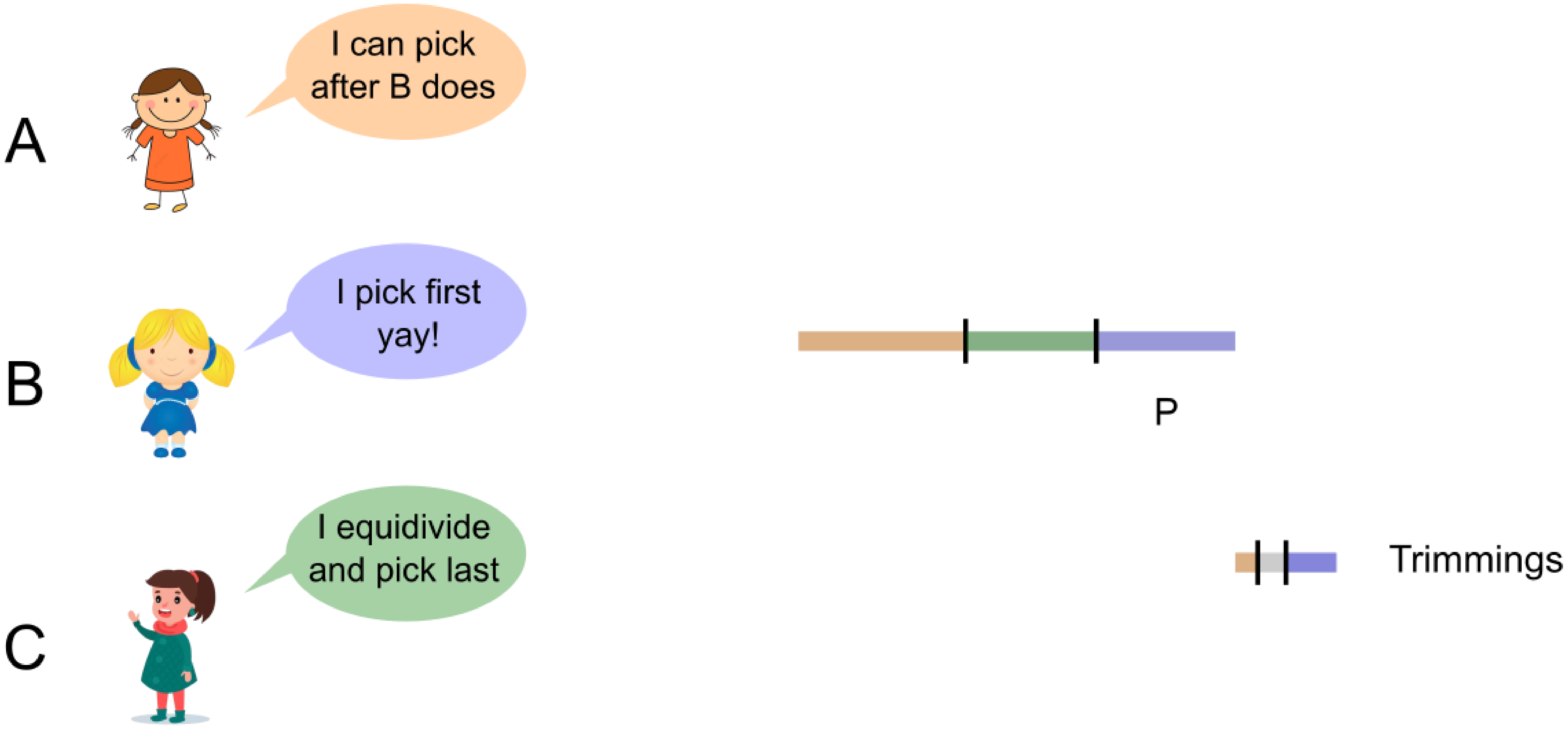


C

Envy-free for three



Envy-free for three

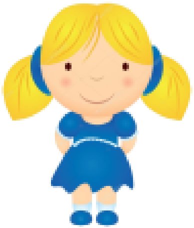


Envy-free for three

A

I can pick after B does

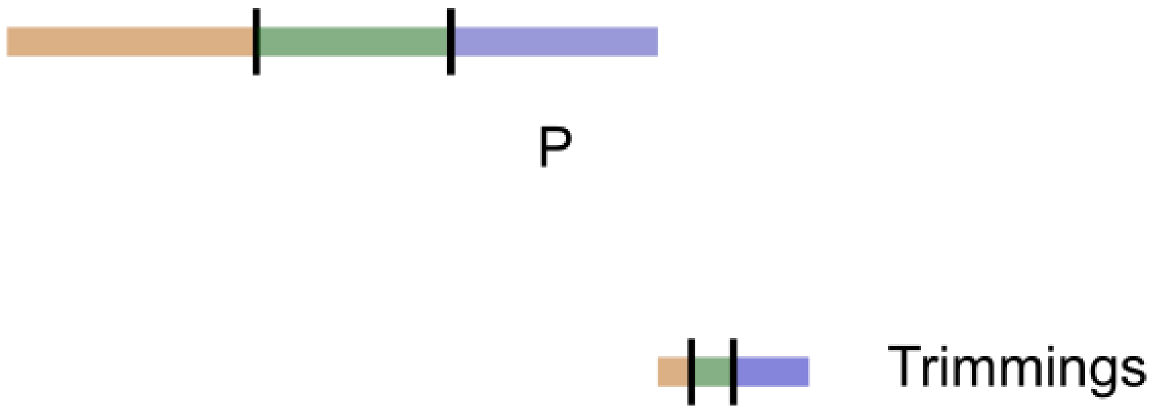
B



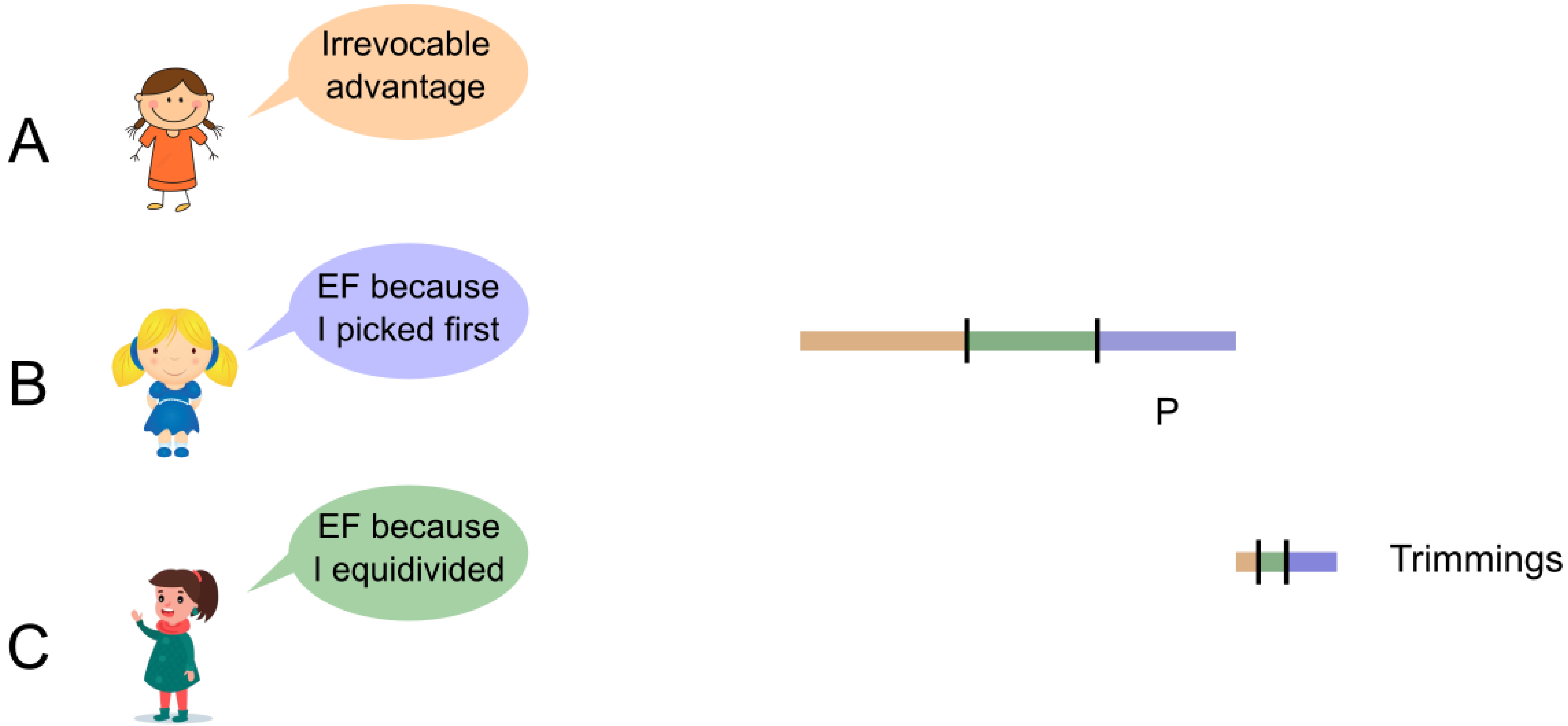
I pick first yay!

C

I equidivide and pick last



Envy-free for three

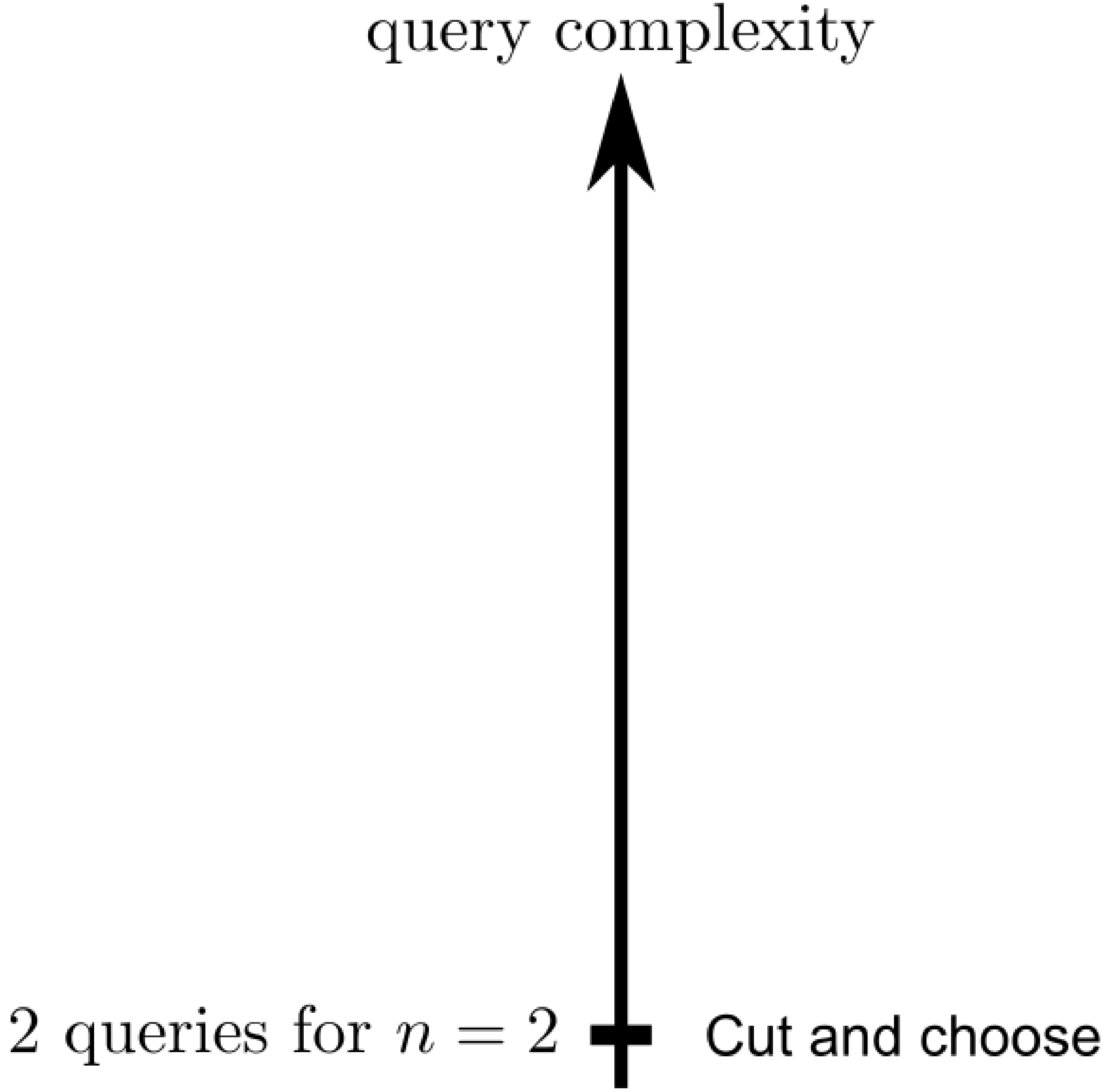


Exercise

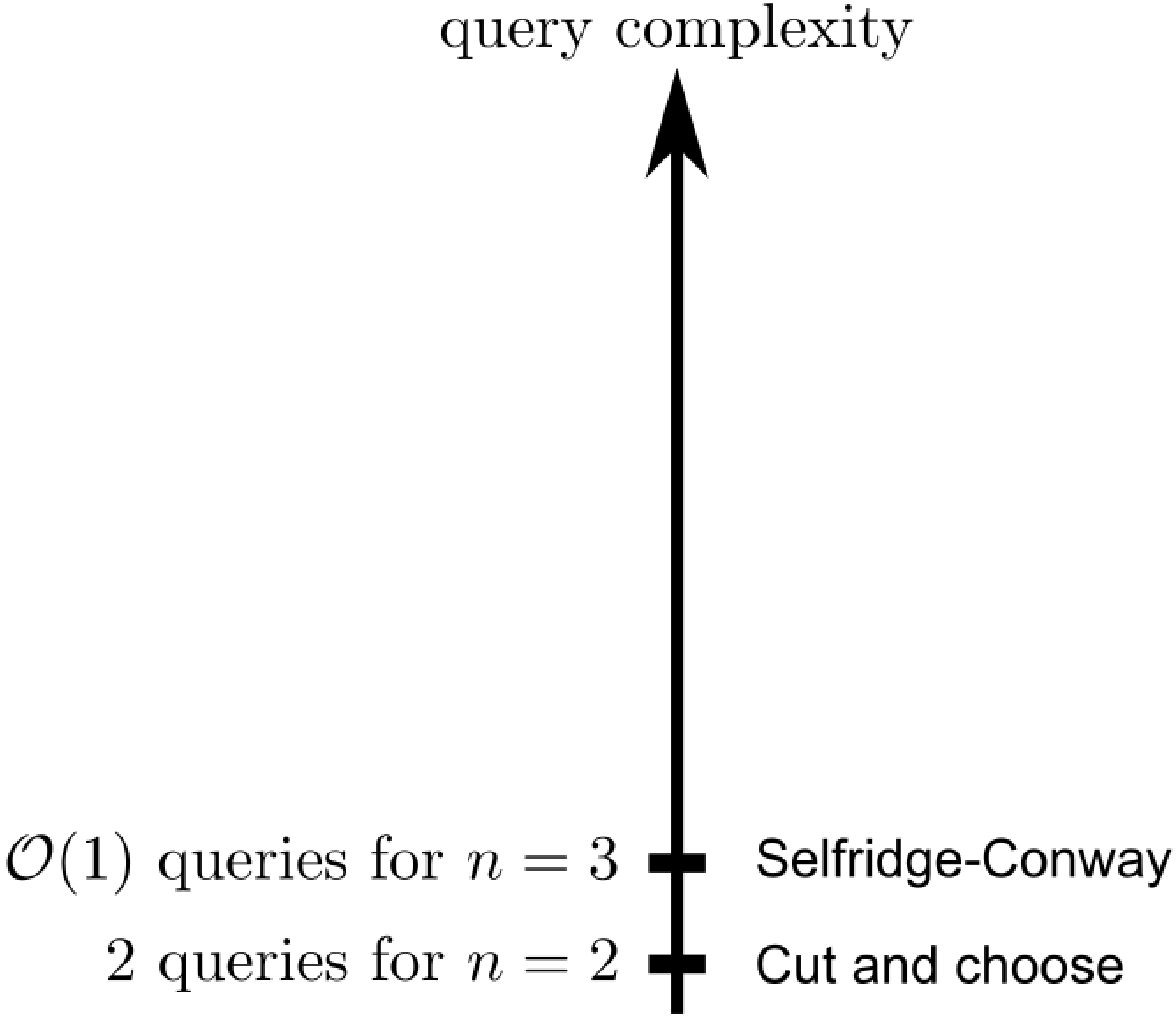
How many queries does the three-person EF protocol require?

The Story of Envy-freeness

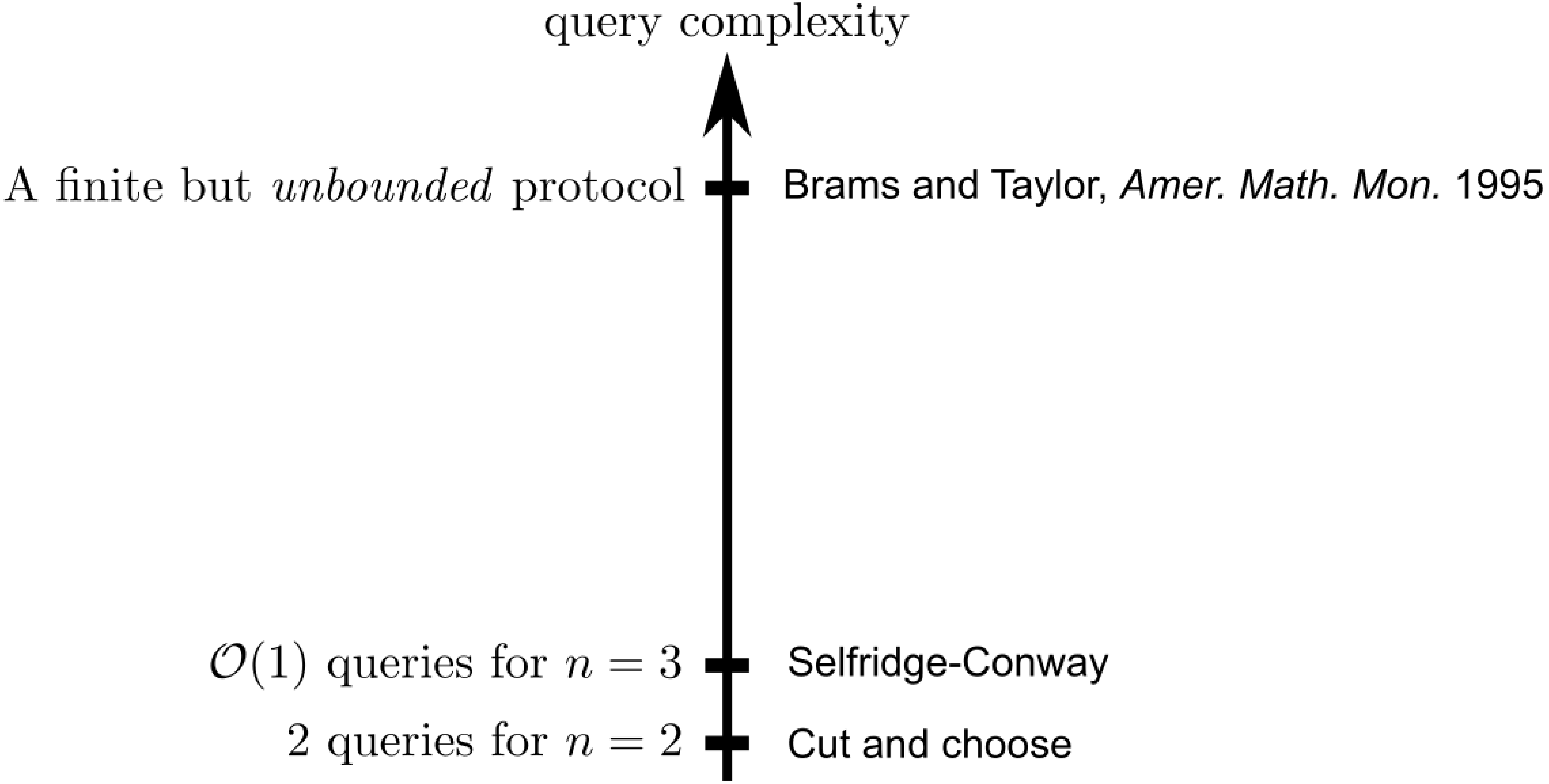
The Story of Envy-freeness



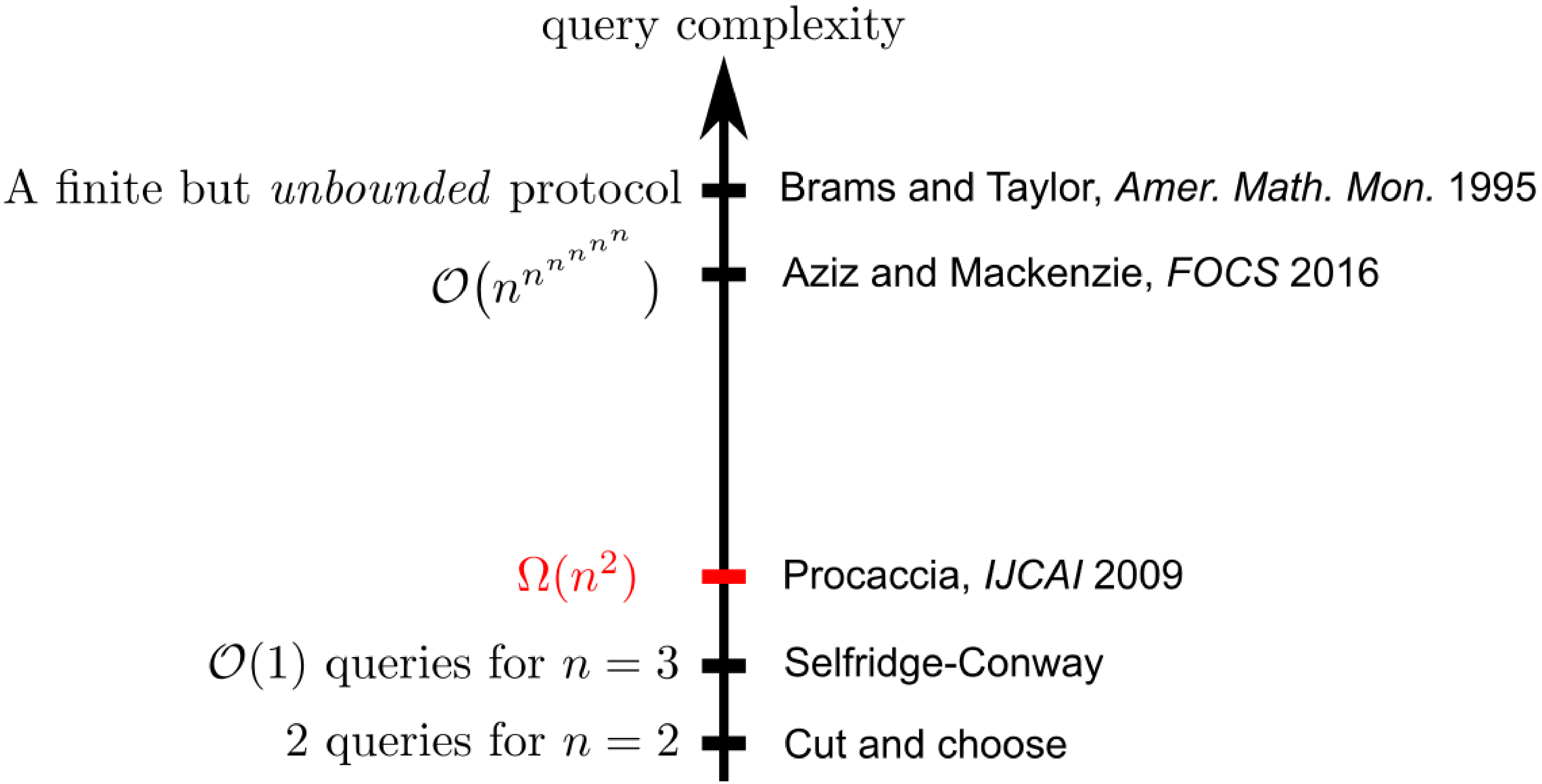
The Story of Envy-freeness



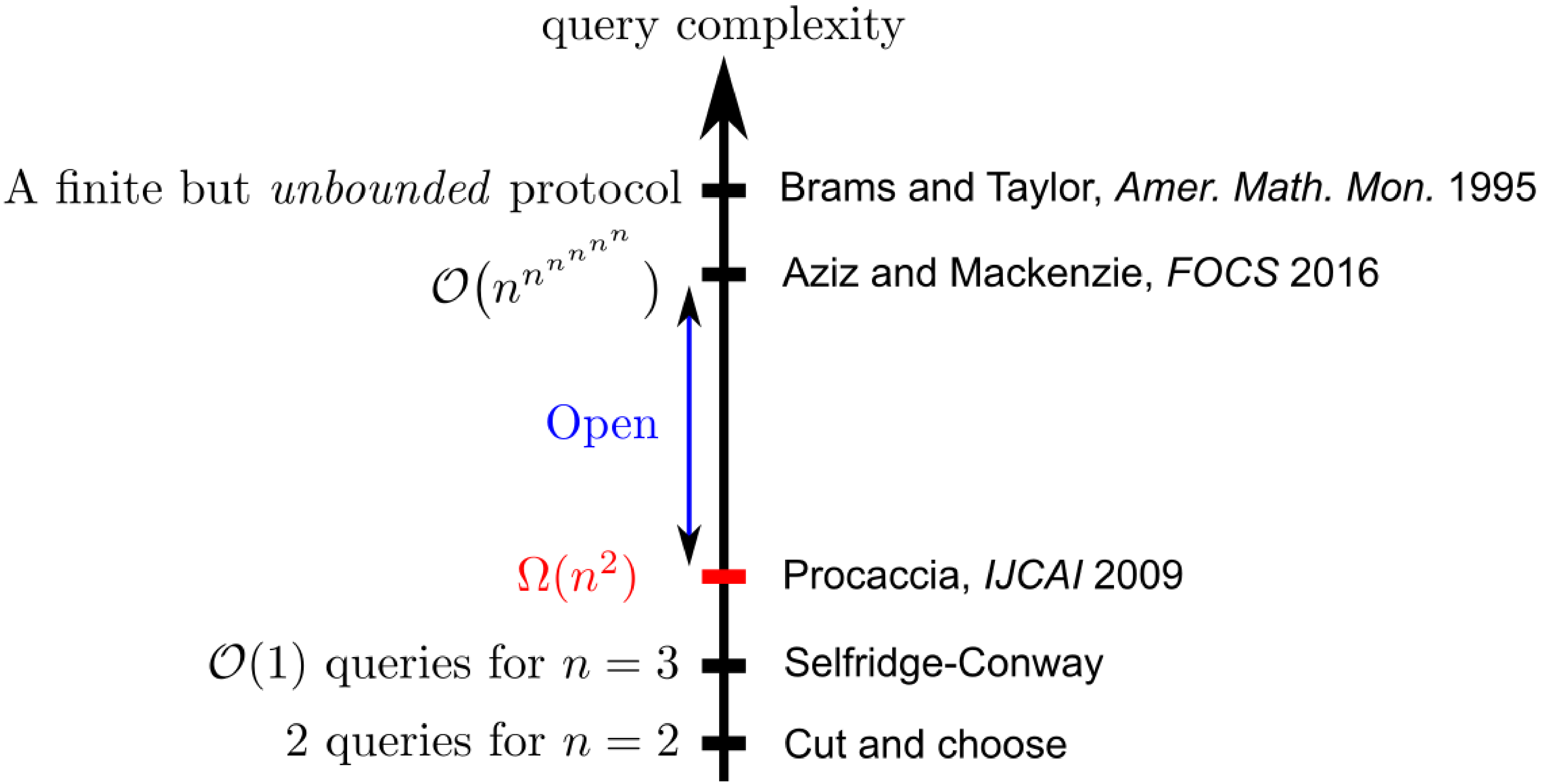
The Story of Envy-freeness



The Story of Envy-freeness



The Story of Envy-freeness



Next Time

Fair Division of Indivisible Items

Quiz

Quiz

Prove or disprove:

The outcome of cut-and-choose is Pareto optimal.

References

- Introduction to cake-cutting algorithms.

  Ariel Procaccia
  "*Cake Cutting Algorithms*"
  Chapter 13 in Handbook of Computational Social Choice

- Lecture by Ariel Procaccia on "Cake cutting" in the *Optimized Democracy* course.
  https://sites.google.com/view/optdemocracy/schedule

Selfridge-Conway Procedure

Phase 1

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).

Selfridge-Conway Procedure

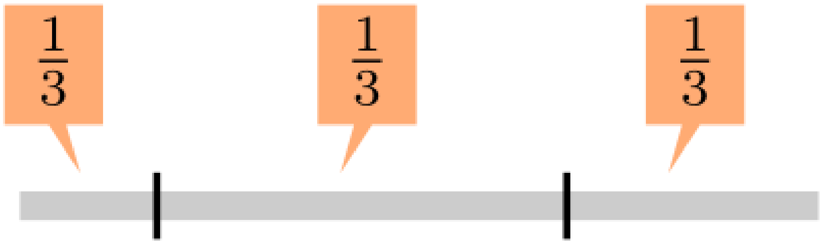
Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).



Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.

Selfridge-Conway Procedure

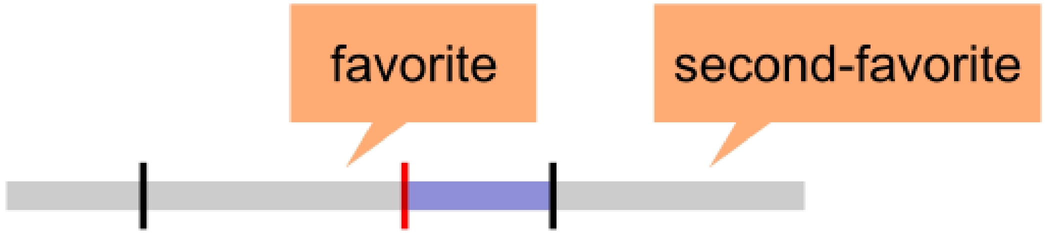
Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.



Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

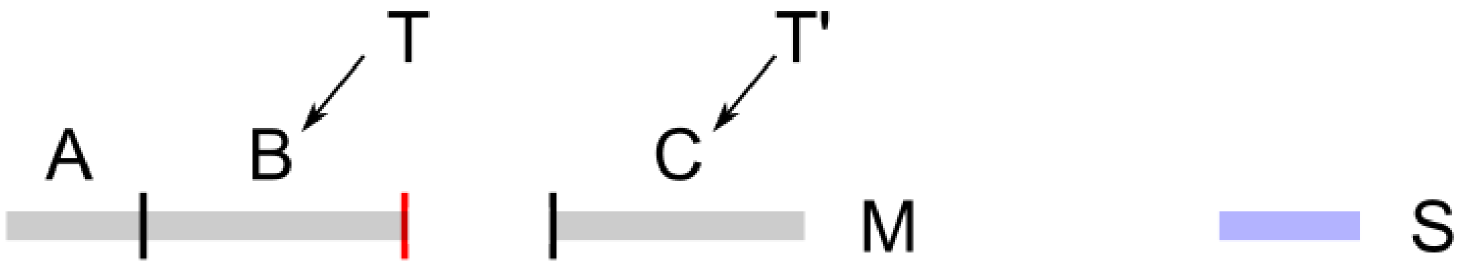
Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.



Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Phase 2

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Phase 2

4. Agent T' divides the trimmings S into three equal pieces (as per $v_{T'}$).

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Phase 2

4. Agent T' divides the trimmings S into three equal pieces (as per $v_{T'}$).

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = M∪S
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Phase 2

4. Agent T' divides the trimmings S into three equal pieces (as per $v_{T'}$).
5. Agent T, then A, then T', in that order, pick a piece each from trimmings S.

Selfridge-Conway Procedure

Phase 1

1. Agent A divides the cake into three equal pieces (as per v_A).
2. Agent B trims its favorite piece to create a two-way tie with second-favorite.
   - Trimmings = S; Main cake M; Original cake = $M \cup S$
3. Agent C, then B, then A, in that order, pick a piece each from main cake M.
   - Agent B must pick the trimmed piece if agent C does not.

Let T = owner of the trimmed piece (T = B or C); let T' = {B,C} \ T.

Phase 2

4. Agent T' divides the trimmings S into three equal pieces (as per $v_{T'}$).
5. Agent T, then A, then T', in that order, pick a piece each from trimmings S.

Selfridge-Conway Procedure

Selfridge-Conway Procedure

- Is any part of the cake left unassigned in the final allocation?

Selfridge-Conway Procedure

- Is any part of the cake left unassigned in the final allocation? No.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective?

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective? Yes.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective? Yes.
  - Within the main cake M, C does not envy A or B because it chooses first.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective? Yes.
  - Within the main cake M, C does not envy A or B because it chooses first.
  - Within the trimmings S, C does not envy A or B because:

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective? Yes.
  - Within the main cake M, C does not envy A or B because it chooses first.
  - Within the trimmings S, C does not envy A or B because:
    - If C is T, then it chooses first in S.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective? Yes.
  - Within the main cake M, C does not envy A or B because it chooses first.
  - Within the trimmings S, C does not envy A or B because:
    - If C is T, then it chooses first in S.
    - If C is T', then it divides S into three equal pieces.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent C's perspective? Yes.
  - Within the main cake M, C does not envy A or B because it chooses first.
  - Within the trimmings S, C does not envy A or B because:
    - If C is T, then it chooses first in S.
    - If C is T', then it divides S into three equal pieces.
  - By additivity across M∪S, C does not envy A or B w.r.t. the entire cake.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective?

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective? Yes.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective? Yes.
    - Within the main cake M, B does not envy A or C because of two-way tie.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective? Yes.
  - Within the main cake M, B does not envy A or C because of two-way tie.
  - Within the trimmings S, B does not envy A or C because:

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective? Yes.
  - Within the main cake M, B does not envy A or C because of two-way tie.
  - Within the trimmings S, B does not envy A or C because:
    - If B is T, then it chooses first in S.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective? Yes.
  - Within the main cake M, B does not envy A or C because of two-way tie.
  - Within the trimmings S, B does not envy A or C because:
    - If B is T, then it chooses first in S.
    - If B is T', then it cuts S into three equal pieces.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent B's perspective? Yes.
  - Within the main cake M, B does not envy A or C because of two-way tie.
  - Within the trimmings S, B does not envy A or C because:
    - If B is T, then it chooses first in S.
    - If B is T', then it cuts S into three equal pieces.
  - By additivity across M∪S, B does not envy A or C w.r.t. the entire cake.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective?

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective? Yes.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective? Yes.
  - Within the main cake M, A does not envy B or C because it was the cutter and it never gets the trimmed piece.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective? Yes.
  - Within the main cake M, A does not envy B or C because it was the cutter and it never gets the trimmed piece.
  - Within the trimmings S, A does not envy:

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective? Yes.
  - Within the main cake M, A does not envy B or C because it was the cutter and it never gets the trimmed piece.
  - Within the trimmings S, A does not envy:
    - T' because it picks before T' does.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective? Yes.
  - Within the main cake M, A does not envy B or C because it was the cutter and it never gets the trimmed piece.
  - Within the trimmings S, A does not envy:
    - T' because it picks before T' does.
    - T because of "irrevocable advantage" from Phase 1.

Selfridge-Conway Procedure

- Is the final allocation envy-free from agent A's perspective? Yes.
  - Within the main cake M, A does not envy B or C because it was the cutter and it never gets the trimmed piece.
  - Within the trimmings S, A does not envy:
    - T' because it picks before T' does.
    - T because of "irrevocable advantage" from Phase 1.
  - By additivity across M∪S, A does not envy B or C w.r.t. the entire cake.