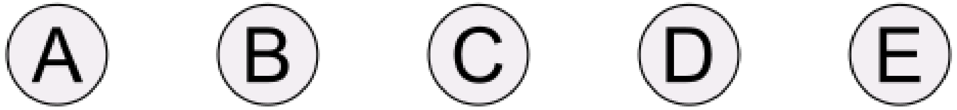
fair allocation of indivisible Chores

Rohit Vaish

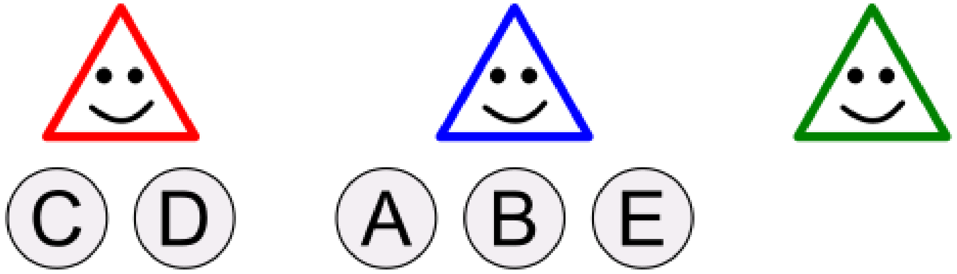
The Model

Set of agents

Set of indivisible items



Allocation



Valuation Function

For three items A B C

{ } = 0

{A} = 1

{B} = 0

{C} = 2

{A B} = 1

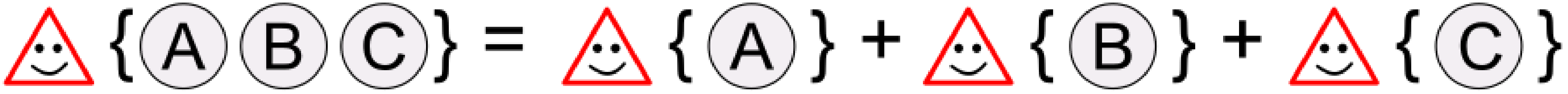
{B C} = 2

{A C} = 3

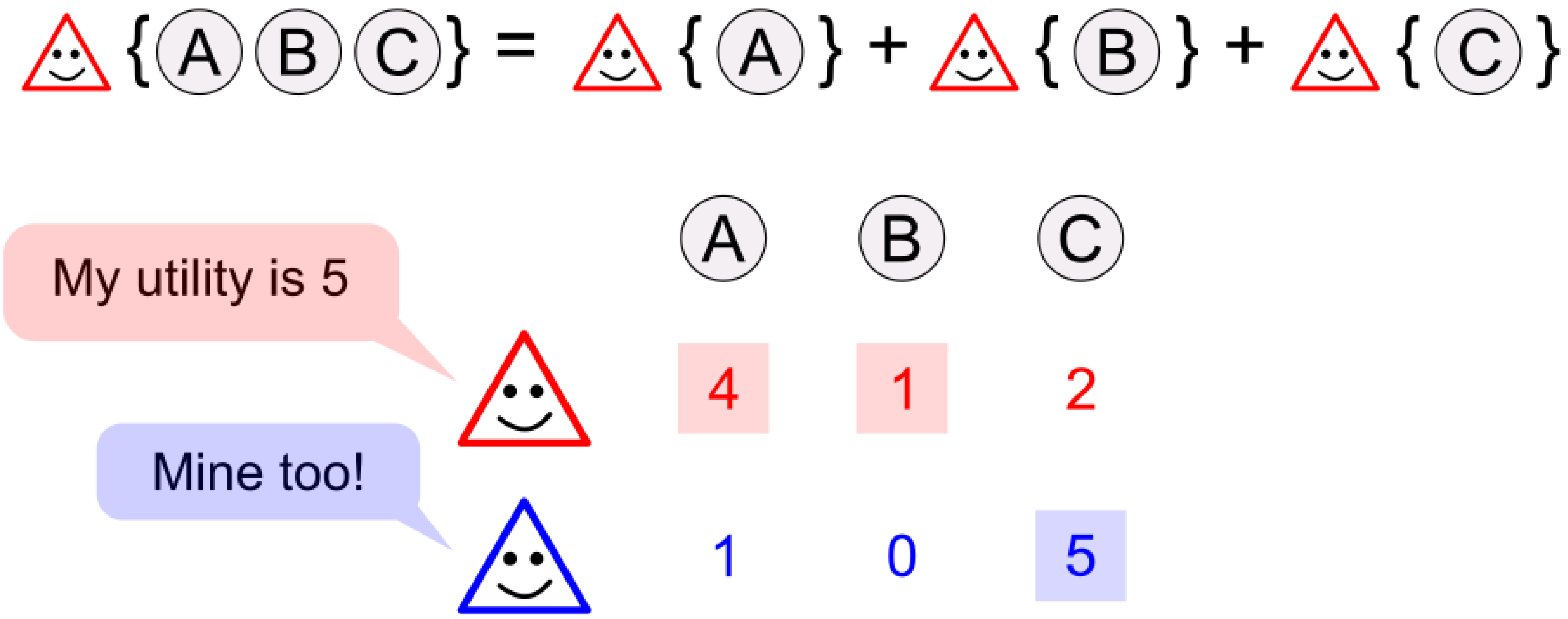
{A B C} = 3

Description grows *exponentially* with the number of items!

Additive Valuations



Additive Valuations



Description grows *linearly* with the number of items.

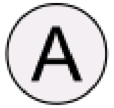
Marginal Value

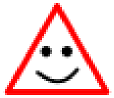
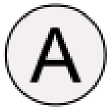
Marginal value of Ⓐ for △ with respect to { Ⓑ Ⓒ }

△ Ⓐ | { Ⓑ Ⓒ } = △ { Ⓐ Ⓑ Ⓒ } - △ { Ⓑ Ⓒ }

Types of Resources

Types of Resources

The item  is a good for  if for *all* subsets of items S

  $\mid S \geq 0$

Types of Resources

The item A is a good for △ if for *all* subsets of items S

$$\triangle\ A \mid S \geq 0$$

E.g., an extra GB of cloud storage

2GB 5GB 15GB 10GB

Types of Resources

The item Ⓑ is a chore for △ if for *all* subsets of items S

$$\triangle\ \text{Ⓑ} \mid S \leq 0$$

E.g., a dish that you forgot to wash

Types of Resources

Good for one agent, chore for another: Mixed items

E.g., service charge in restaurant bills

Types of Resources

If all items are goods for all agents: Goods instance

If all items are chores for all agents: Chores instance

Otherwise: Mixed instance

Types of Resources

Goods

Chores

Mixed

Types of Valuation Functions

Goods

Chores

Mixed

Types of Valuation Functions

Goods

=

Monotone ↑

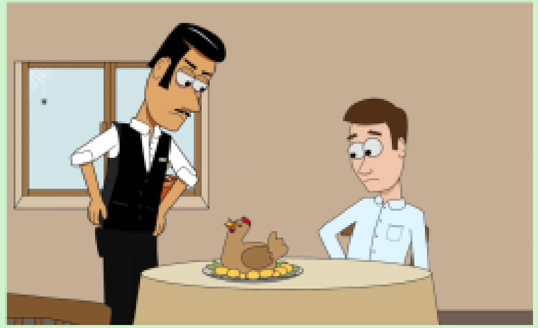
 $S \geq$  T

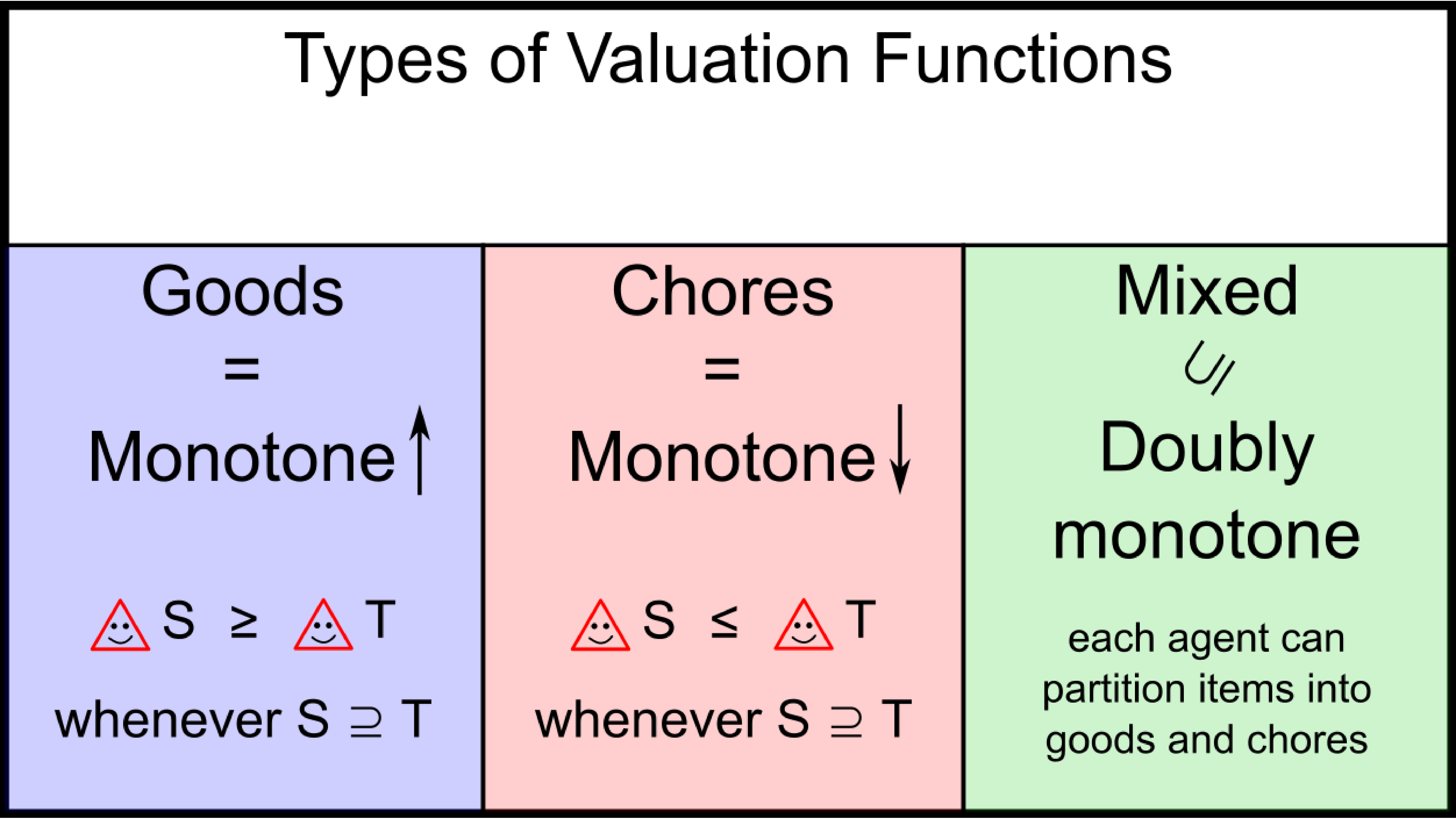
whenever $S \supseteq T$

Chores

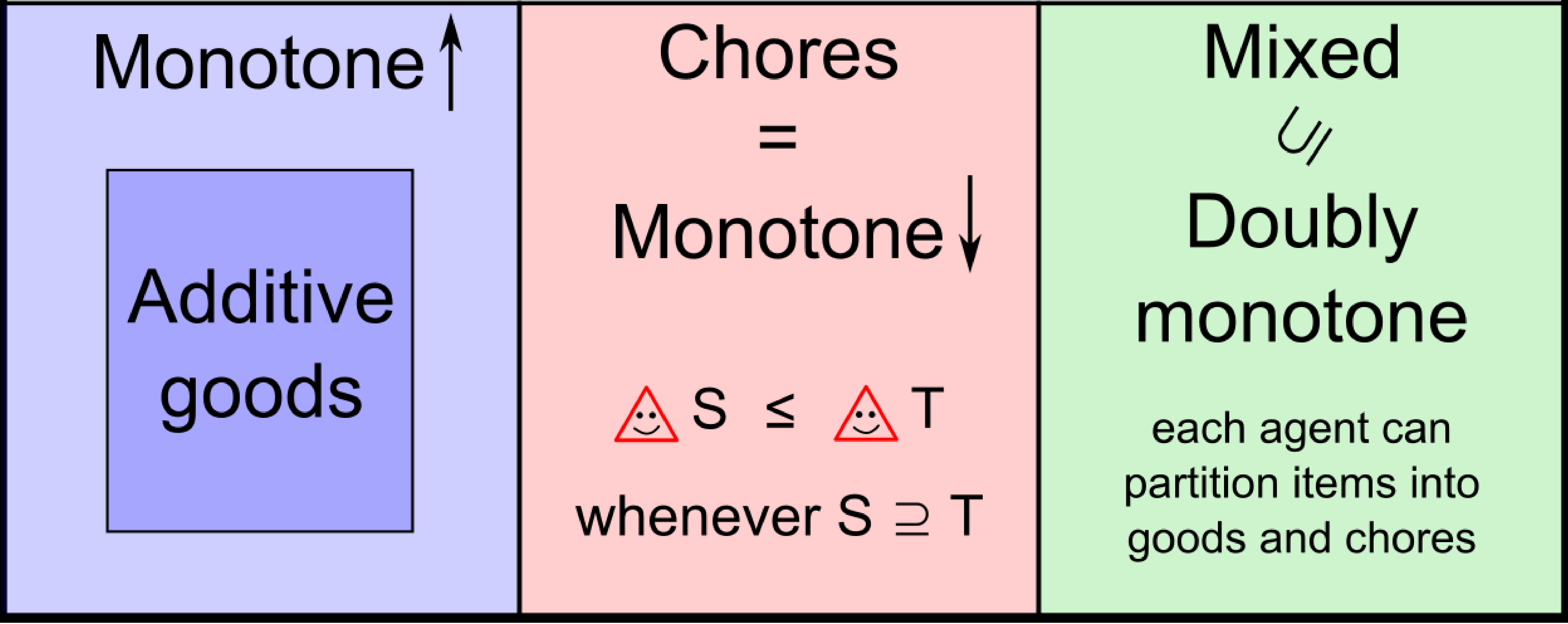
Mixed

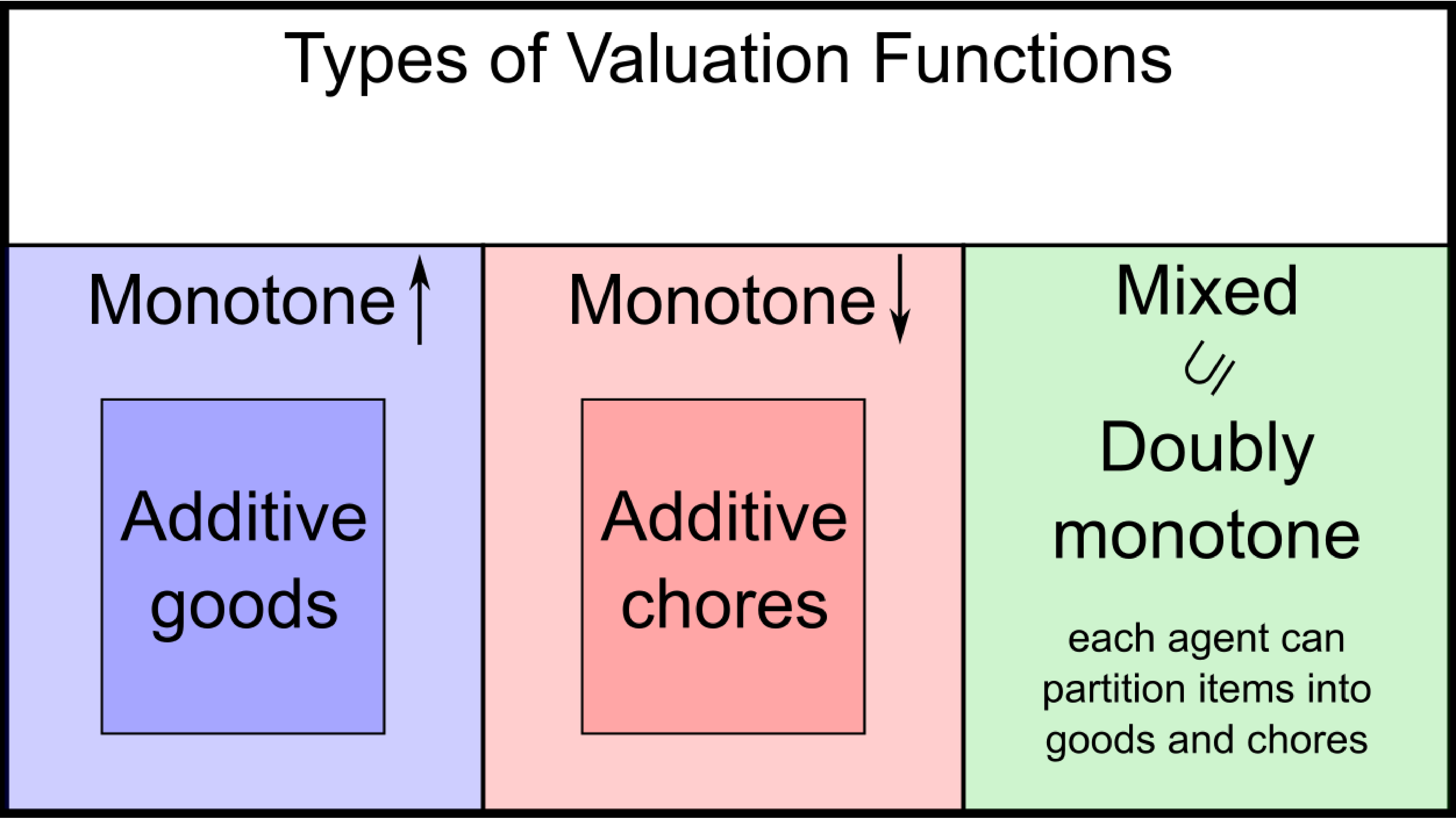
Types of Valuation Functions

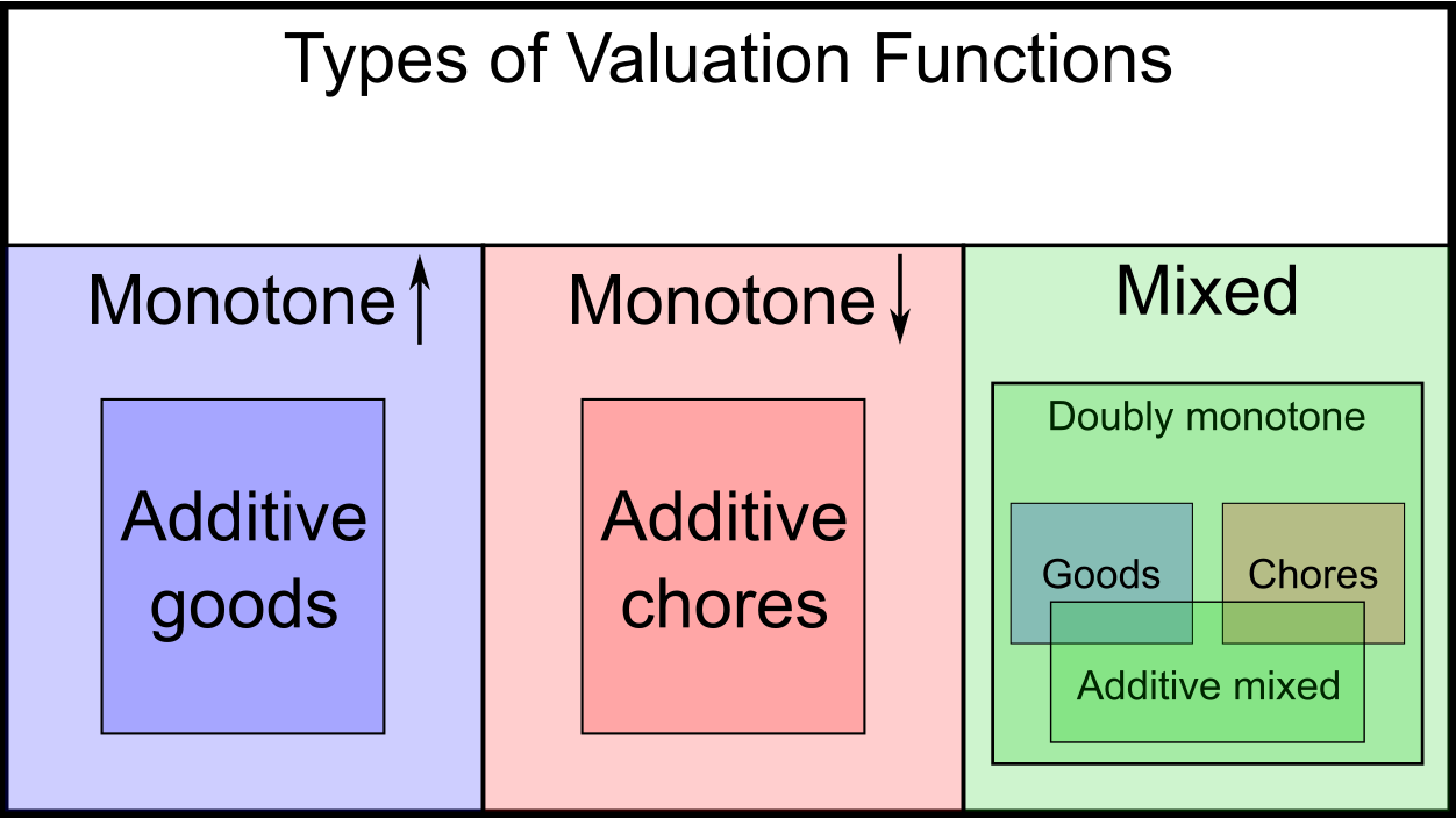
Goods	Chores	Mixed
=	=	
Monotone ↑	Monotone ↓	
$\triangle S \geq \triangle T$	$\triangle S \leq \triangle T$	
whenever $S \supseteq T$	whenever $S \supseteq T$	


Types of Valuation Functions
Goods
=
Monotone↑
S ≥ T
whenever S ⊇ T
Chores
=
Monotone↓
S ≤ T
whenever S ⊇ T
Mixed
⊆
Doubly monotone
each agent can partition items into goods and chores

Types of Valuation Functions




Types of Valuation Functions
Monotone↑
Additive goods
Monotone↓
Additive chores
Mixed
⊇
Doubly monotone
each agent can partition items into goods and chores


Types of Valuation Functions
Monotone↑
Additive goods
Monotone↓
Additive chores
Mixed
Doubly monotone
Goods
Chores
Additive mixed

Types of Valuation Functions

Under additive valuations

Goods

	A	B	C
	4	1	2
	1	0	5

Chores

	A	B	C
	-1	0	-2
	-5	-1	-1

Mixed

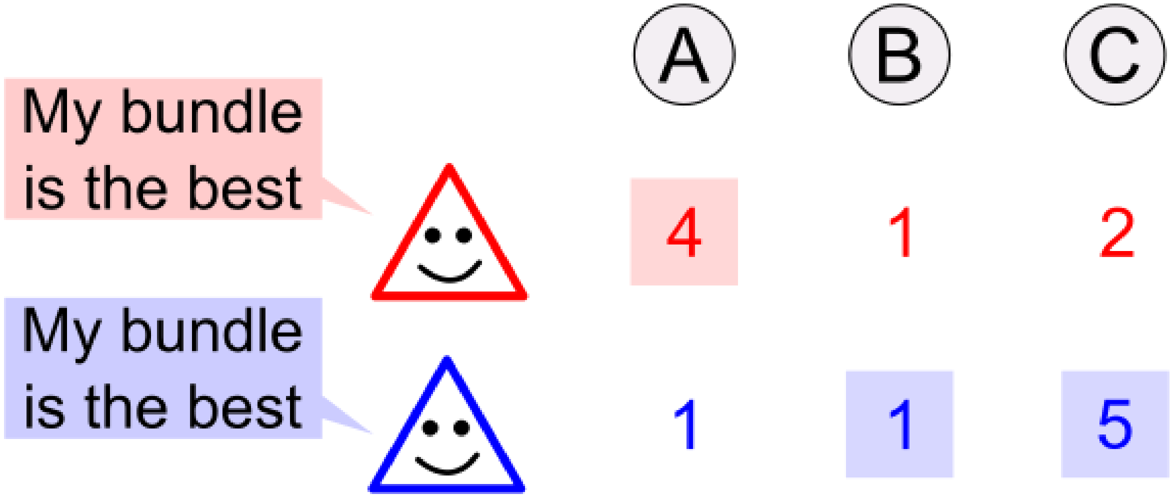
	A	B	C
	1	1	-1
	-2	0	-2

Fairness Notions

Envy-Freeness

[Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.



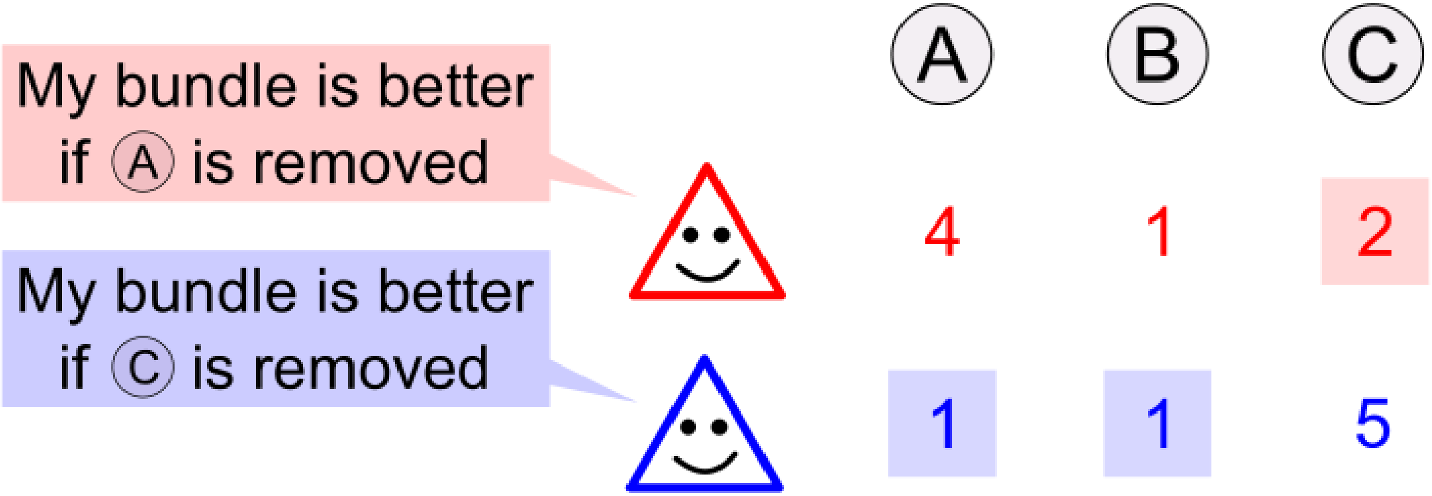
Not guaranteed to exist (two agents, one good)

Checking whether an EF allocation exists is NP-complete

Envy-Freeness Up To One Good

[Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.

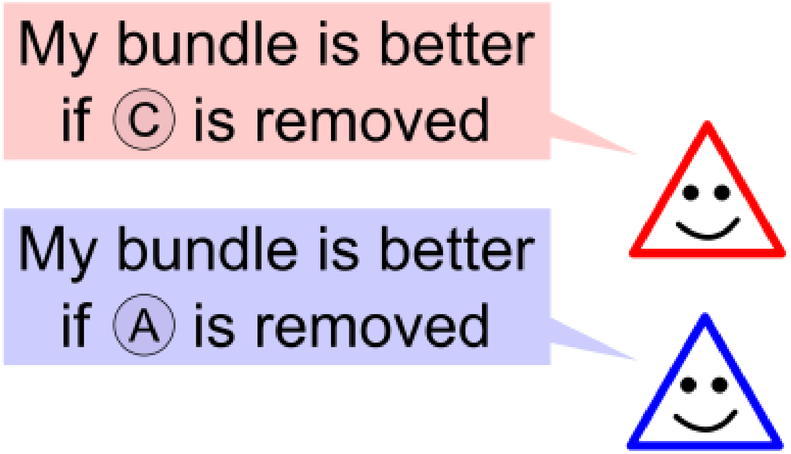


Guaranteed to exist and efficiently computable

Envy-Freeness Up To One Chore

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

Eliminate envy by removing some chore in the envious agent's bundle.



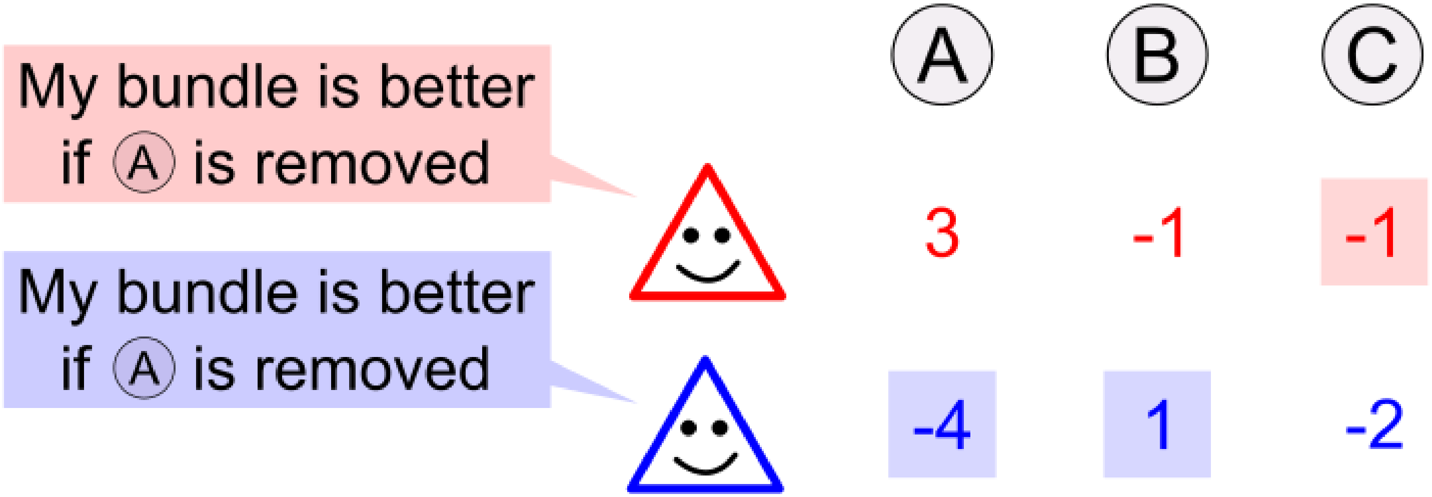
	A	B	C
	-1	-1	-3
	-4	-1	-2

Allocation $A = (A_1, \ldots, A_n)$ is EF1 if for every pair of agents i, k, there exists a chore $j \in A_i$ such that $v_i(A_i \setminus \{j\}) \geq v_i(A_k)$.

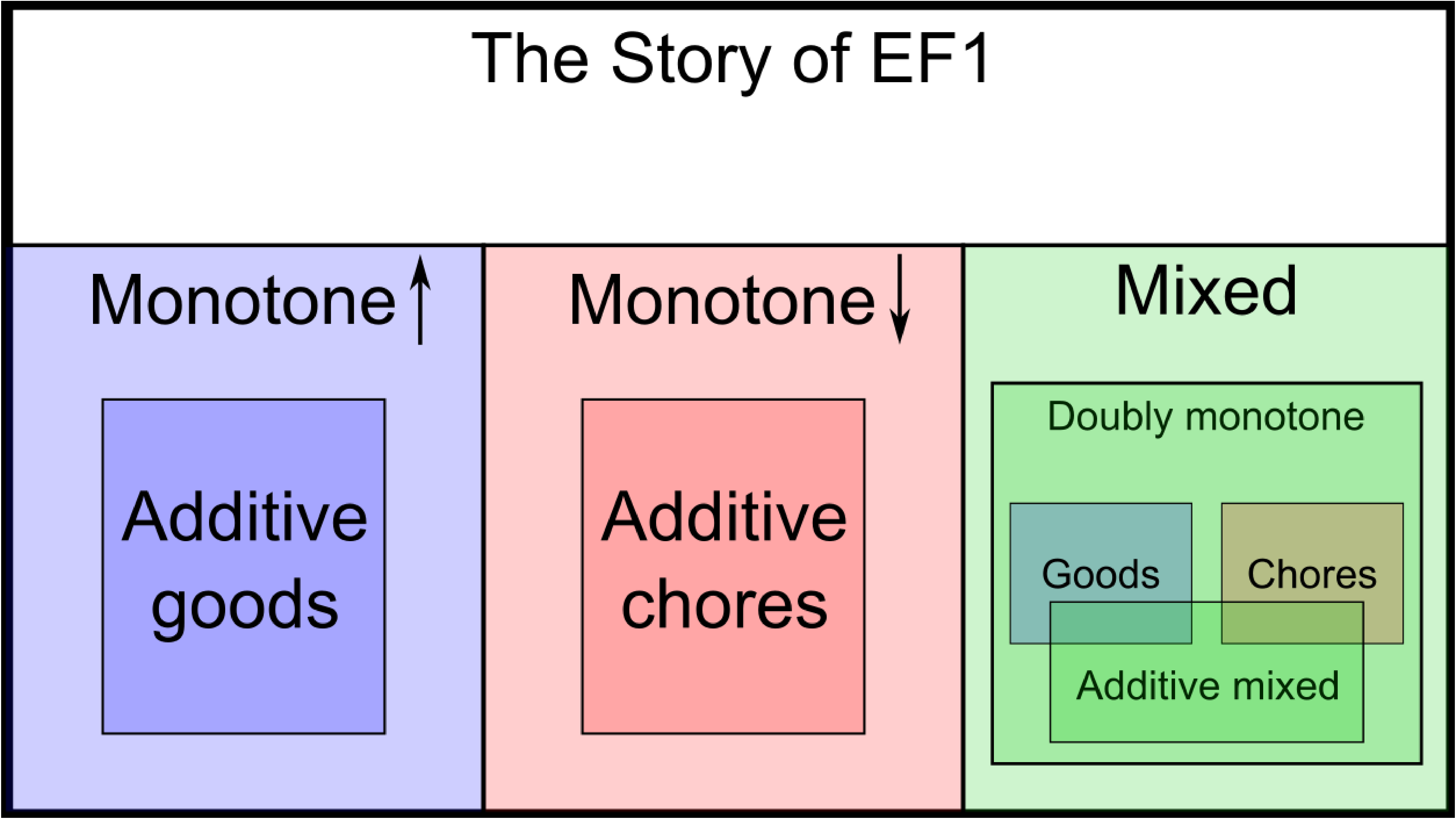
Envy-Freeness Up To One Item

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

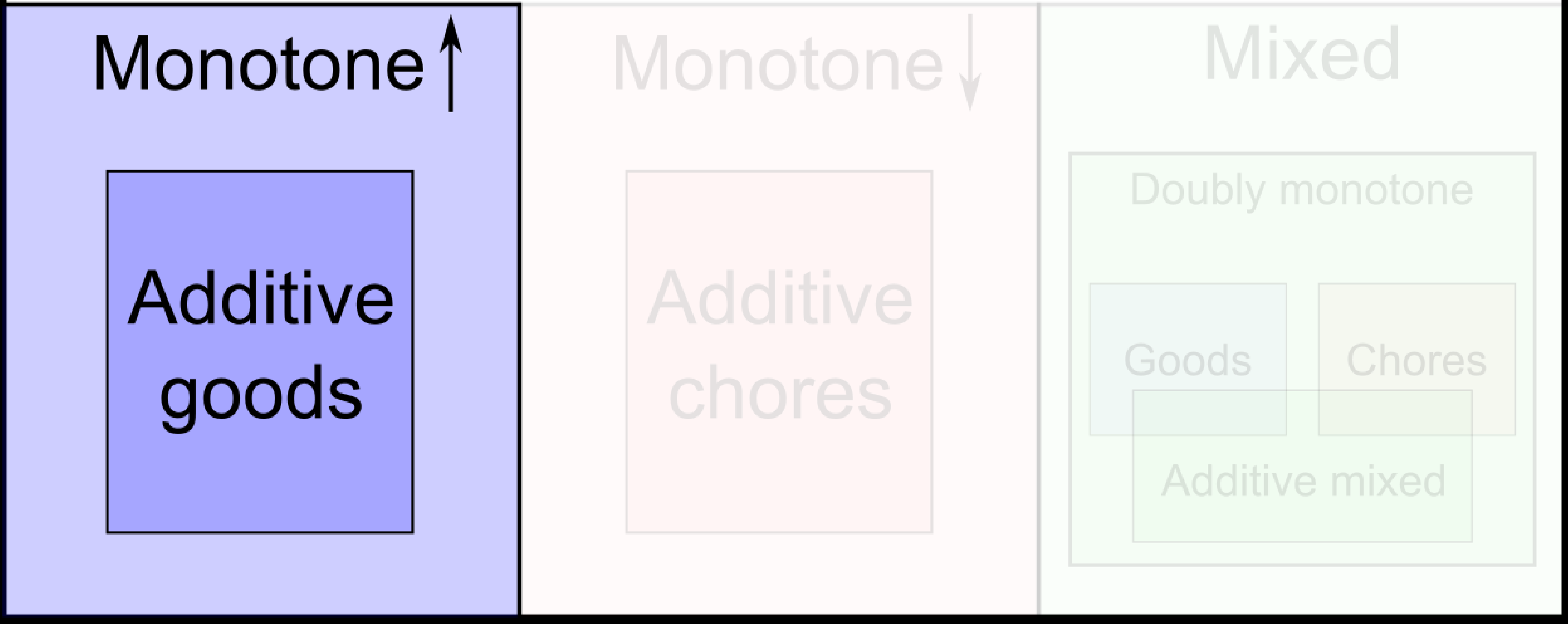
Eliminate envy by removing some "good" in the envied bundle or some "chore" in the envious agent's bundle.



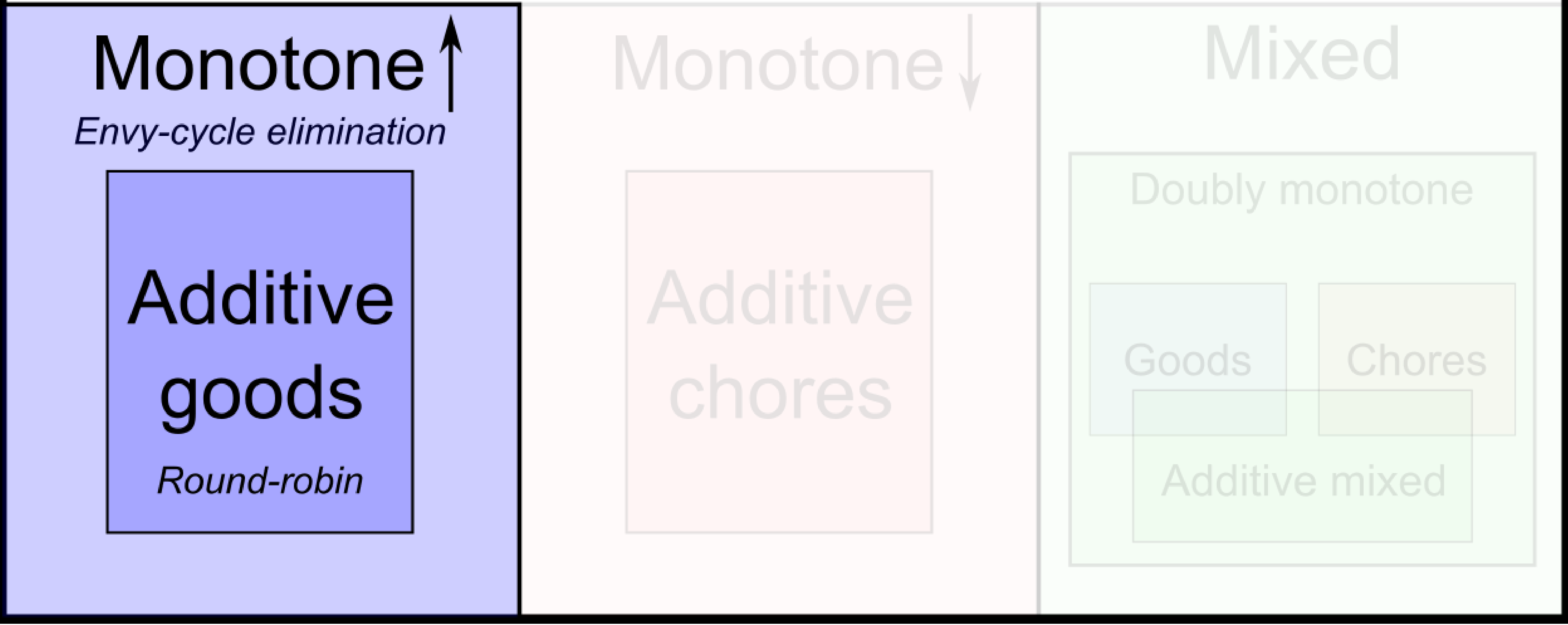
Allocation $A = (A_1, \ldots, A_n)$ is EF1 if for every pair of agents i, k, there exists an item $j \in A_i \cup A_k$ s.t. $v_i(A_i \setminus \{j\}) \geq v_i(A_k \setminus \{j\})$.


The Story of EF1
Monotone↑
Additive goods
Monotone↓
Additive chores
Mixed
Doubly monotone
Goods
Chores
Additive mixed

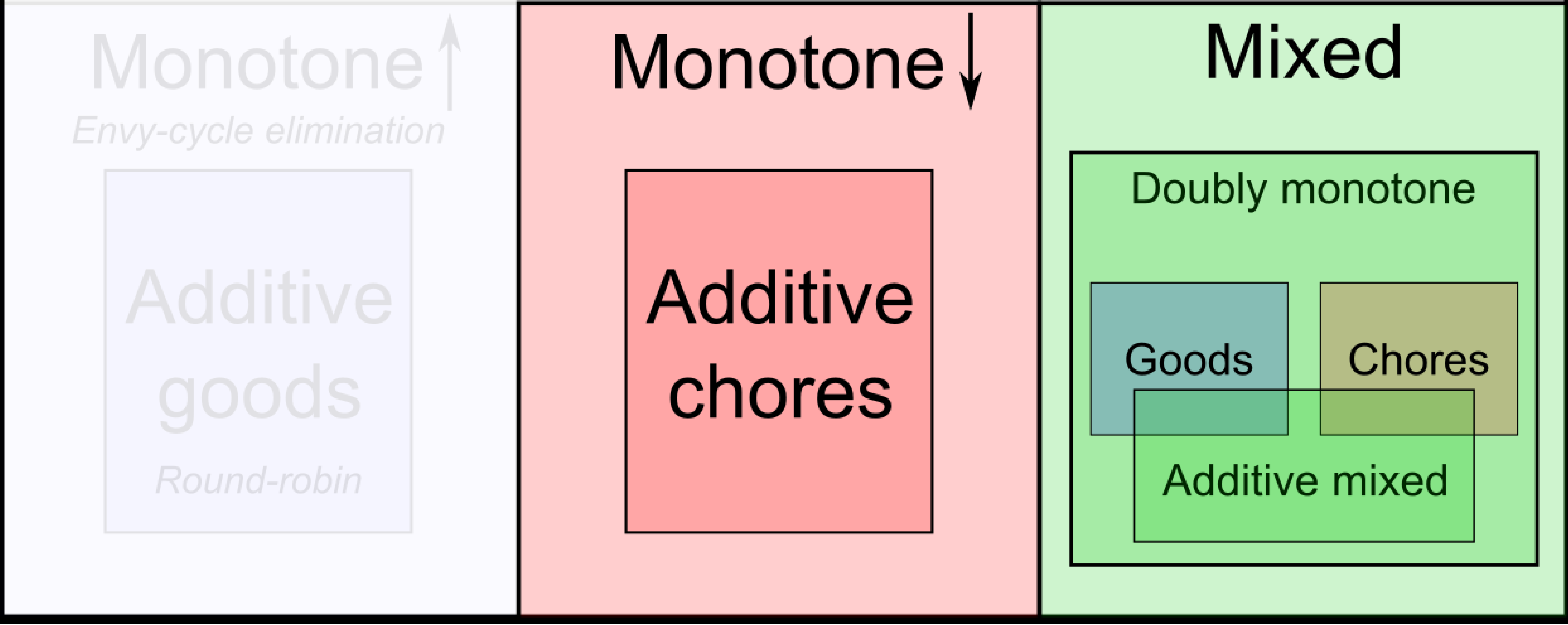
The Story of EF1



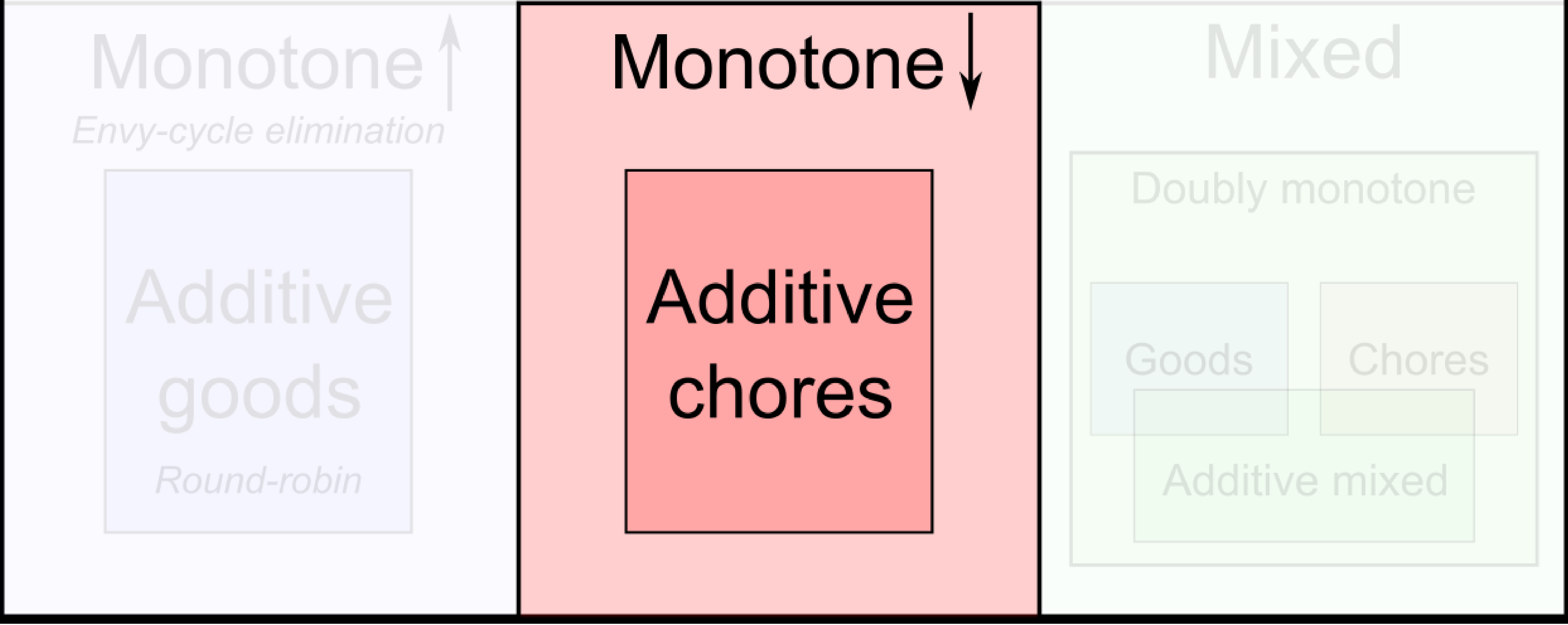
The Story of EF1



The Story of EF1



The Story of EF1

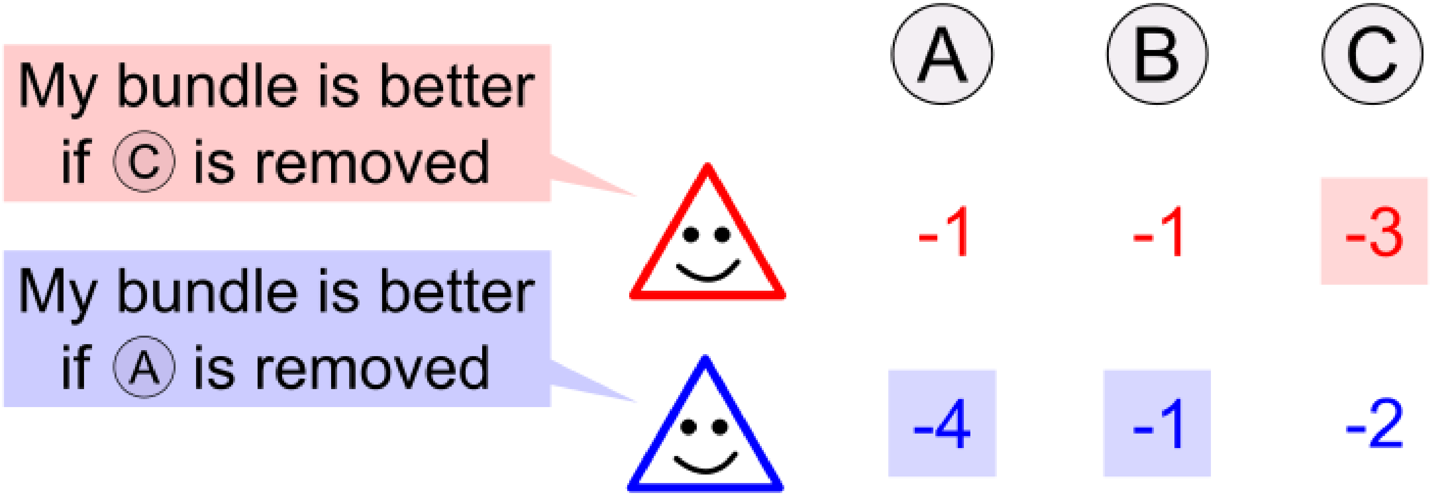


The Story of EF1

Envy-Freeness Up To One Chore

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

Eliminate envy by removing some chore in the envious agent's bundle.



Allocation $A = (A_1, \ldots, A_n)$ is EF1 if for every pair of agents i, k, there exists a chore $j \in A_i$ such that $v_i(A_i \setminus \{j\}) \geq v_i(A_k)$.

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

	A	B	C	D	E
	-4	-1	-3	-2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	-3	-1

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

	A	B	C	D	E
	-4	-1	-3	-2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	-3	-1

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

	A	B	C	D	E
	-4	-1	-3	-2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	-3	-1

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

	A	B	C	D	E
	-4	-1	-3	-2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	-3	-1

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

	A	B	C	D	E
	-4	-1	-3	-2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	-3	-1

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

	A	B	C	D	E
	-4	-1	-3	-2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	-3	-1

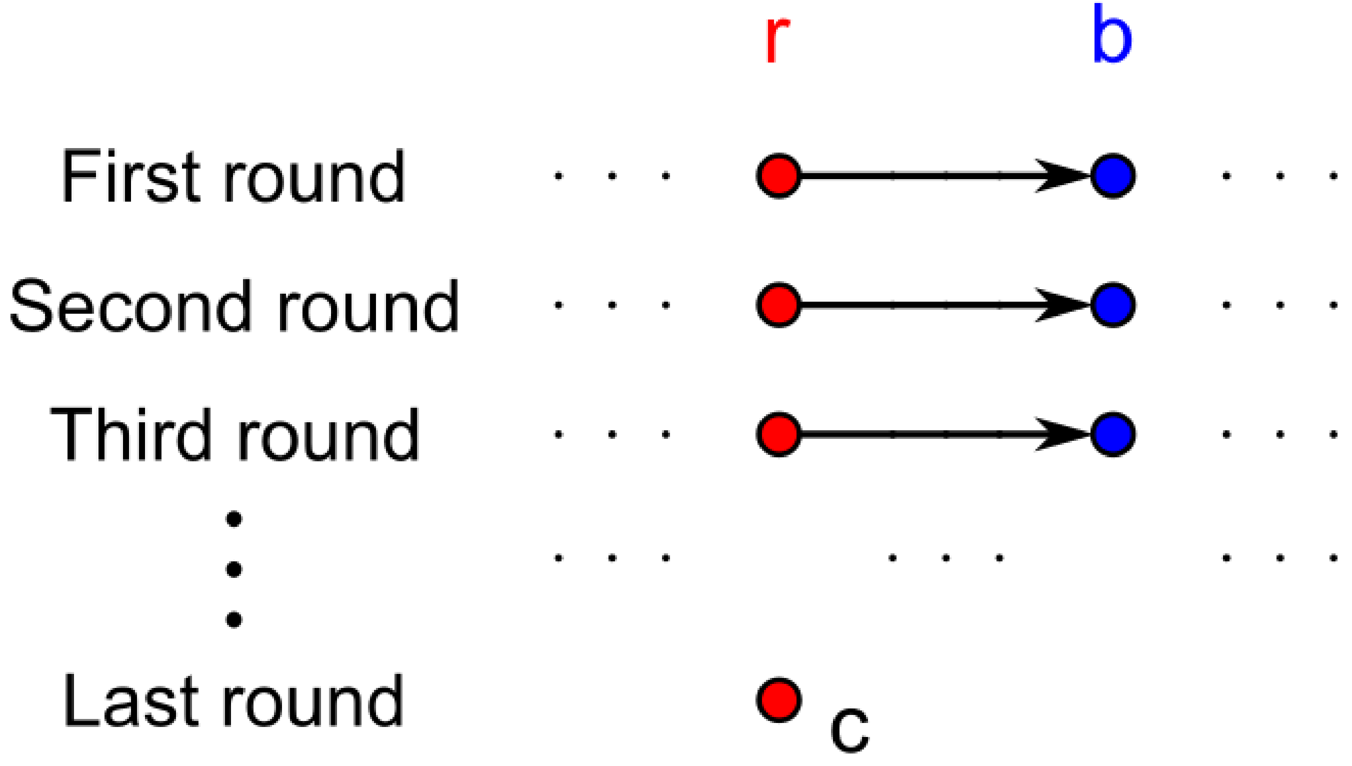
For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.

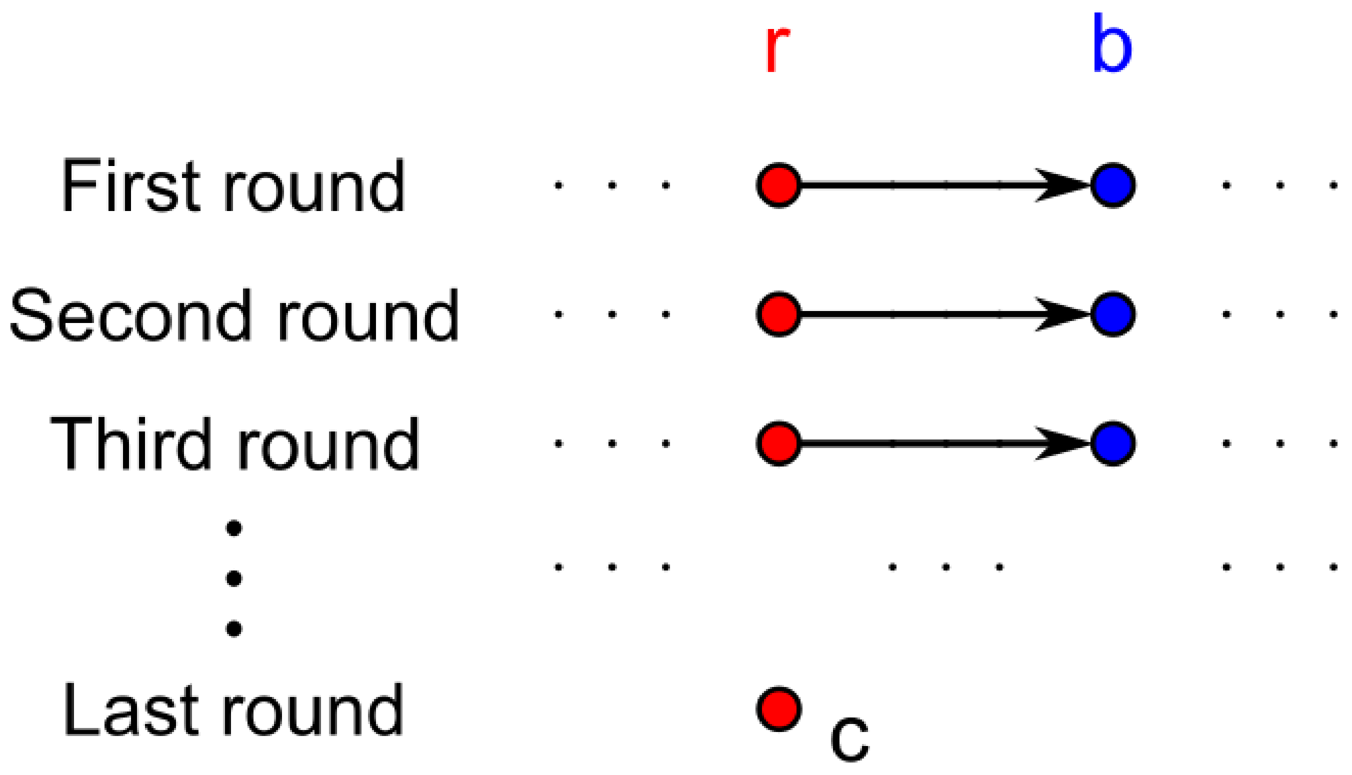
For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



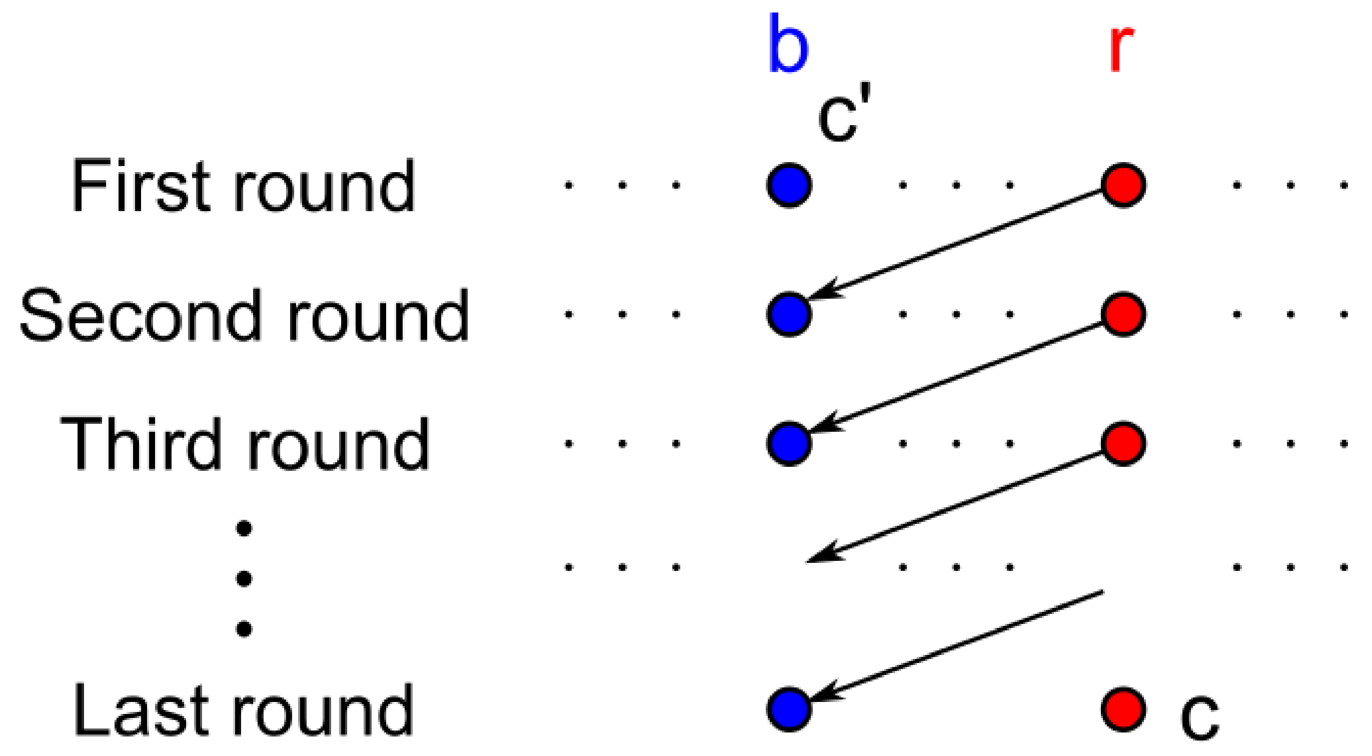
If r precedes b: Then, by additivity, $v_r(A_r \setminus \{c\}) \geq v_r(A_b)$.

For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.

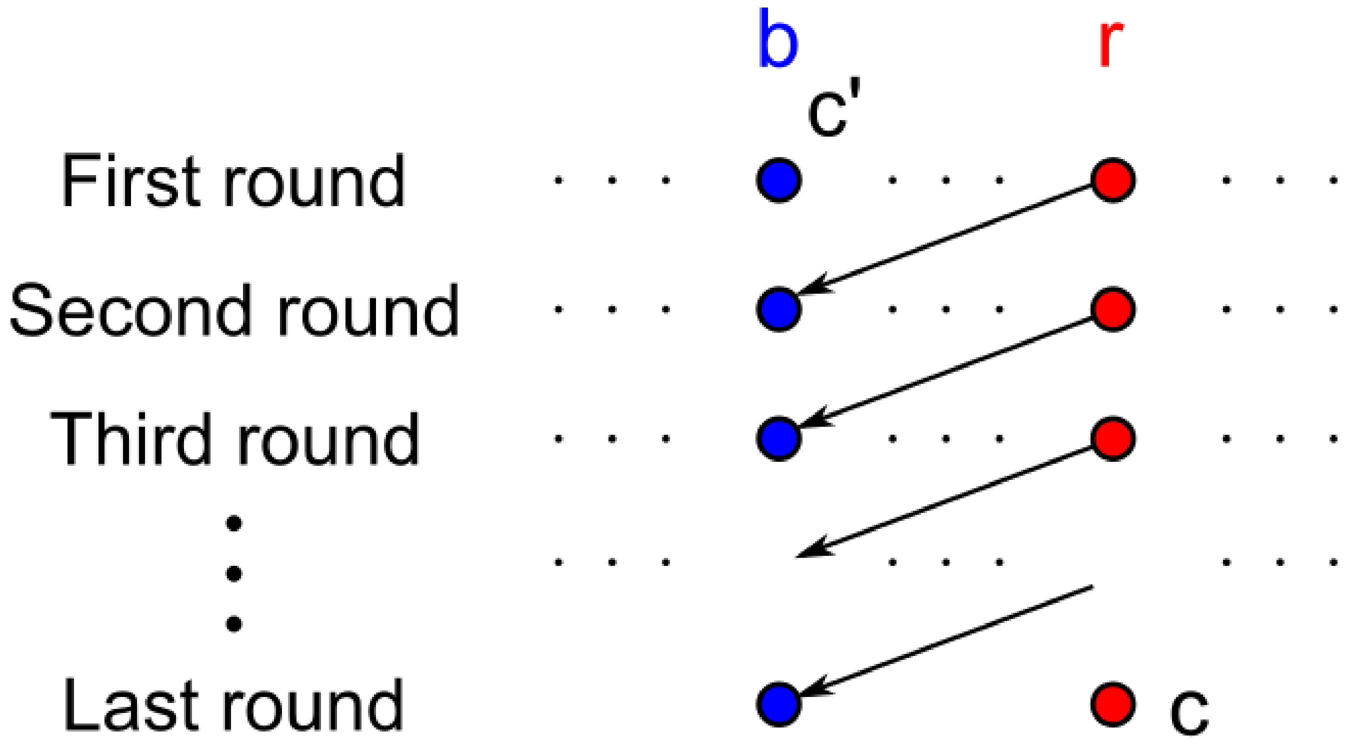
For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

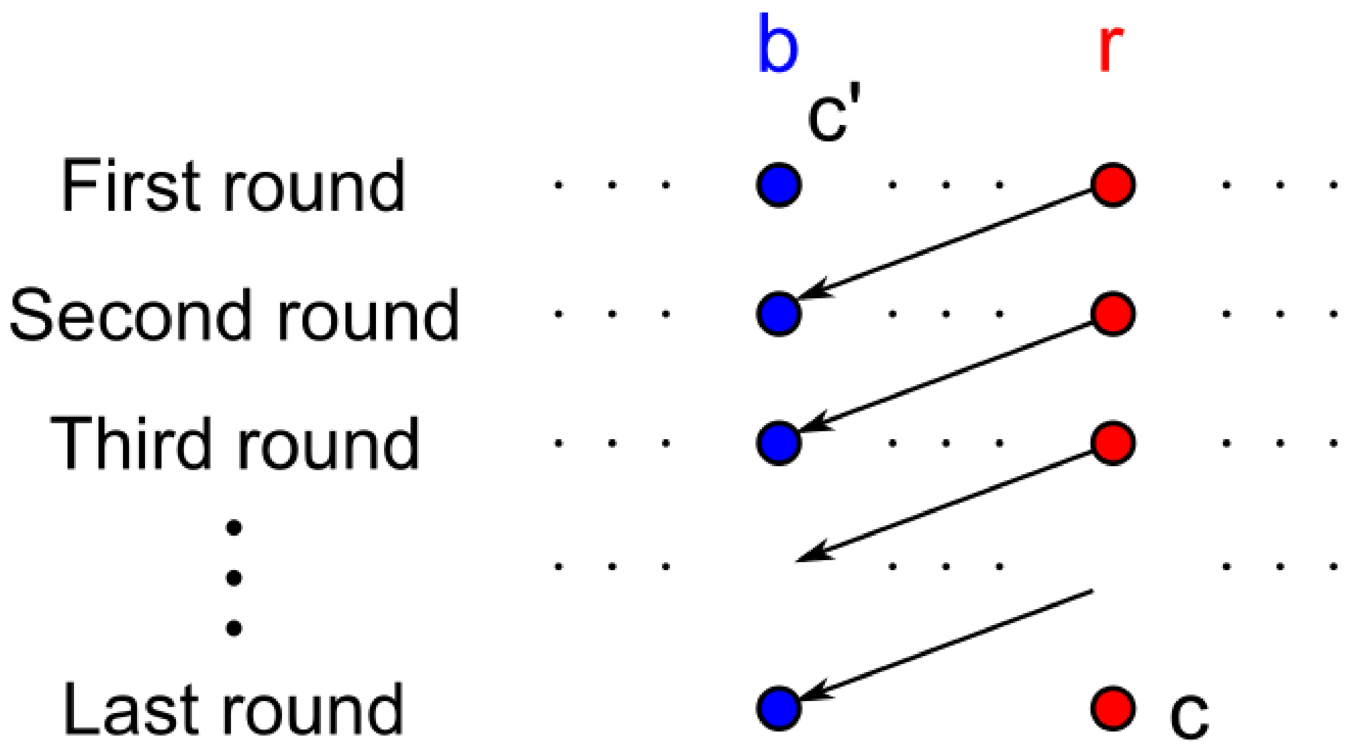
Fix a pair of agents (r,b). Analyze envy of r towards b.



If b precedes r: Again, by additivity, $v_r(A_r\setminus\{c\}) \geq v_r(A_b\setminus\{c'\}) \geq v_r(A_b)$.

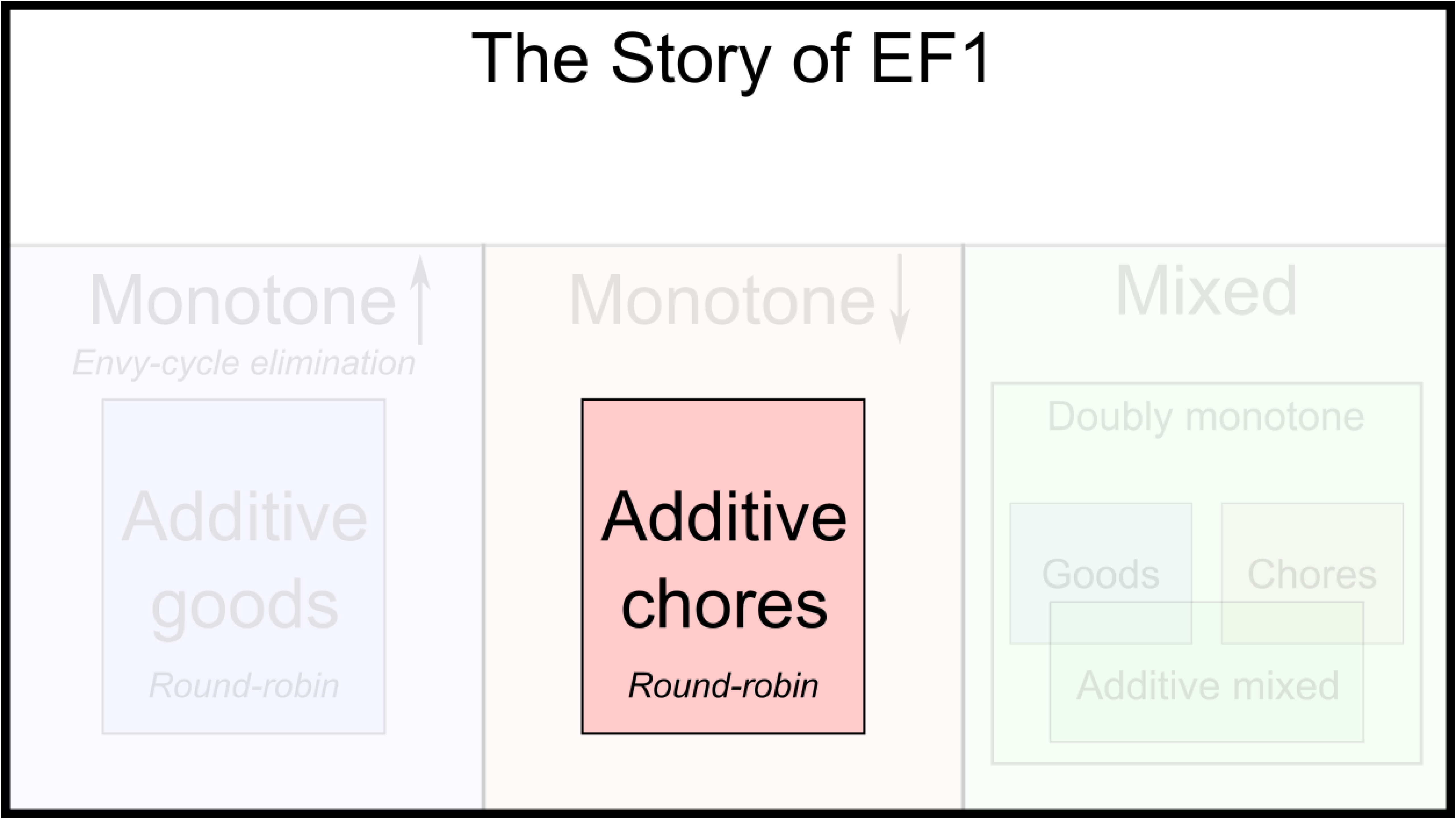
For additive chores, the allocation computed by round-robin algorithm satisfies EF1.

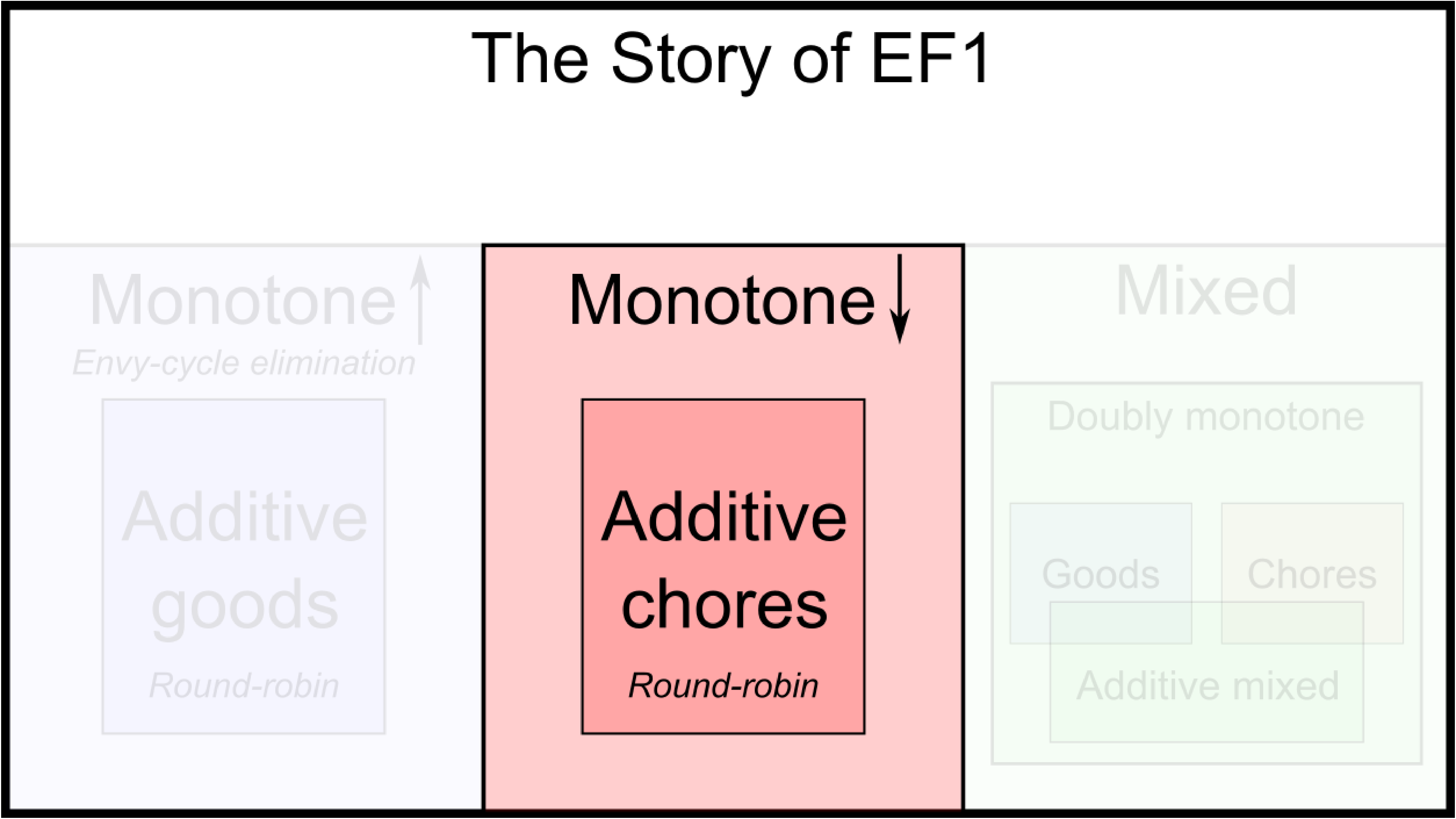
Fix a pair of agents (r,b). Analyze envy of r towards b.



If b precedes r: Again, by additivity, $v_r(A_r \setminus \{c\}) \geq v_r(A_b \setminus \{c'\}) \geq v_r(A_b)$.

The Story of EF1


The Story of EF1
Monotone↑
Envy-cycle elimination
Additive goods
Round-robin
Monotone↓
Additive chores
Round-robin
Mixed
Doubly monotone
Goods
Chores
Additive mixed



The Story of EF1
Monotone↑
Envy-cycle elimination
Additive goods
Round-robin
Monotone↓
Additive chores
Round-robin
Mixed
Doubly monotone
Goods
Chores
Additive mixed

Adapting envy-cycle elimination to chores

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Adapting envy-cycle elimination to chores

While there is an unallocated ~~good~~ chore

- If the envy graph has a ~~source~~ sink vertex, assign the ~~good~~ chore to that agent.
- Otherwise, resolve envy cycles until a ~~source~~ sink vertex shows up, and then assign the ~~good~~ chore to it.

Adapting envy-cycle elimination to chores

While there is an unallocated chore

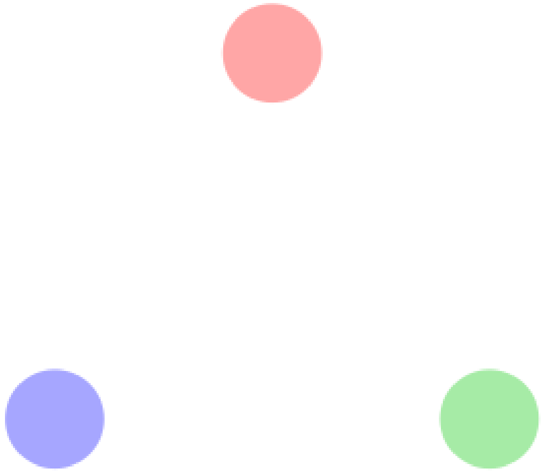
- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

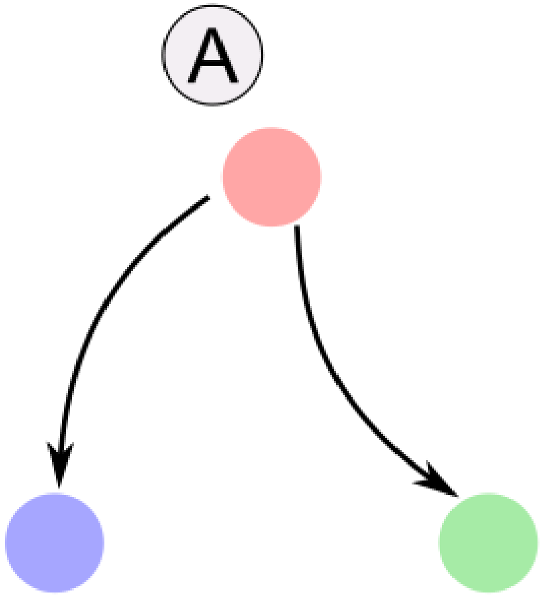


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

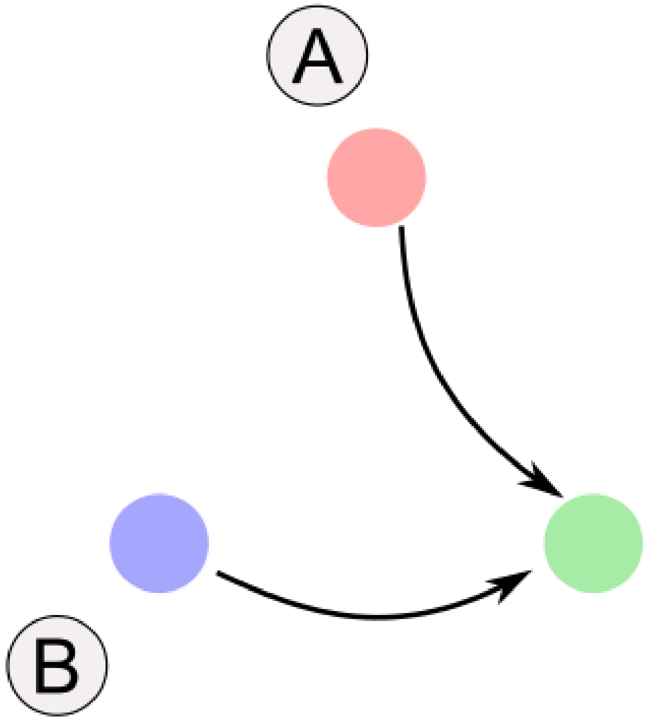


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

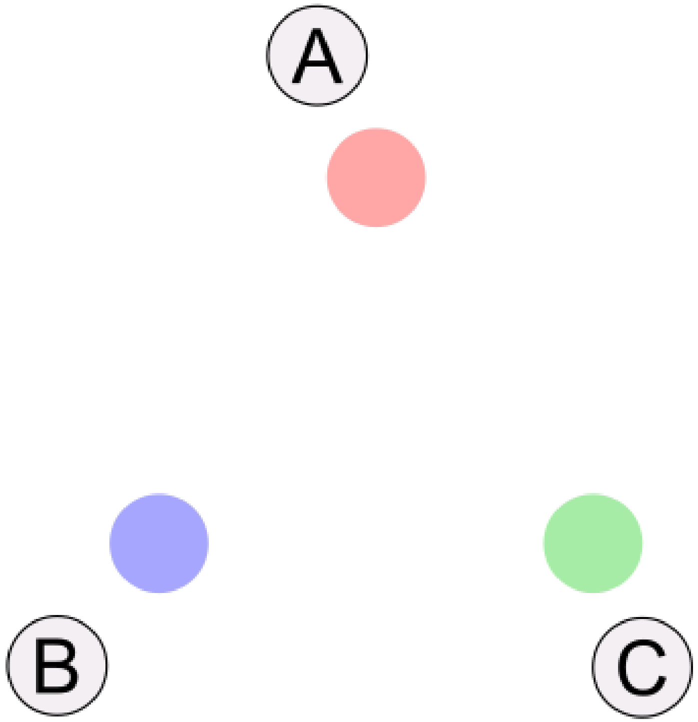


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

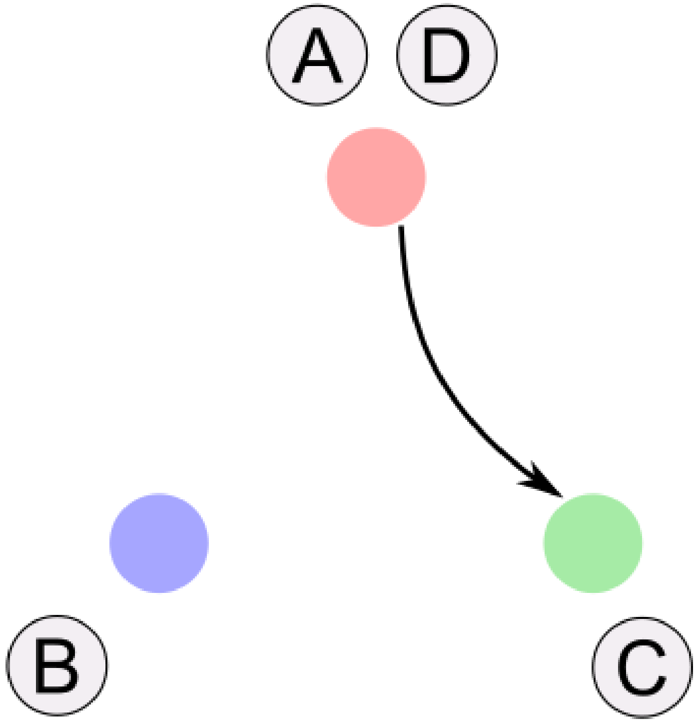


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

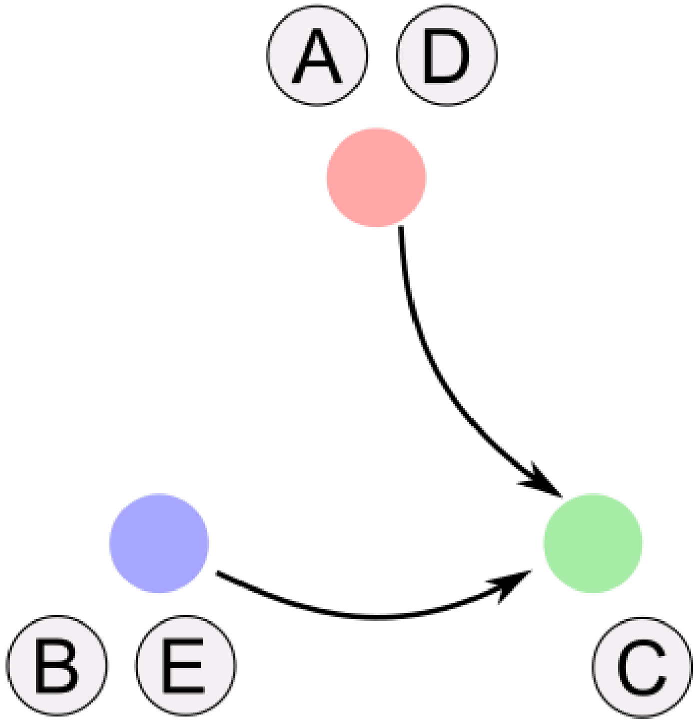


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

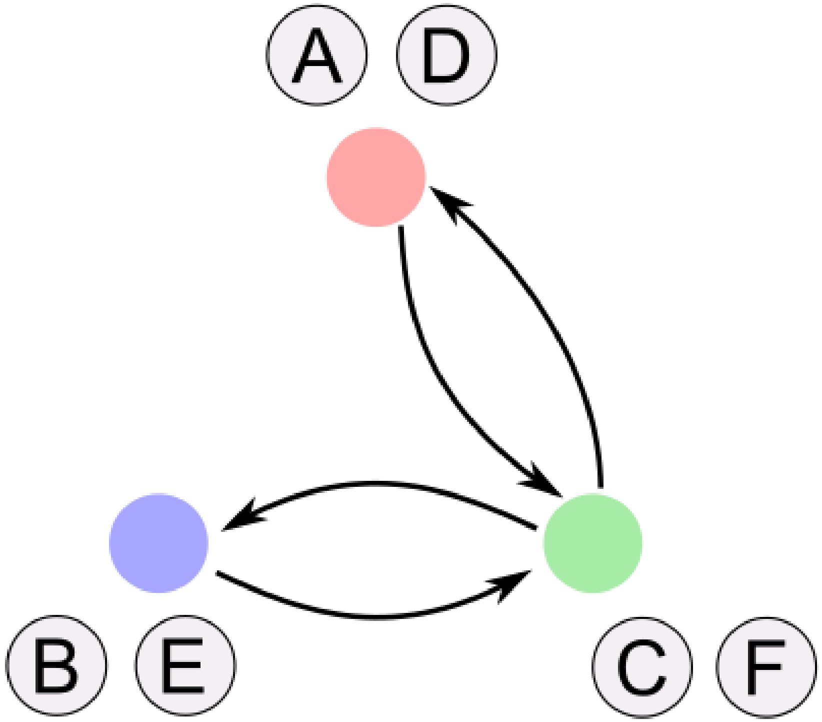


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

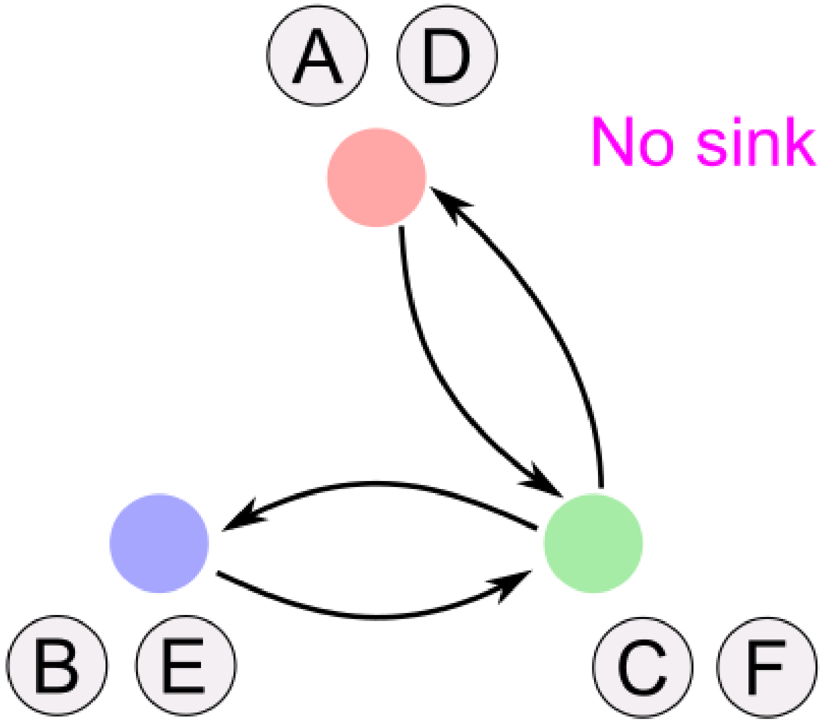


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

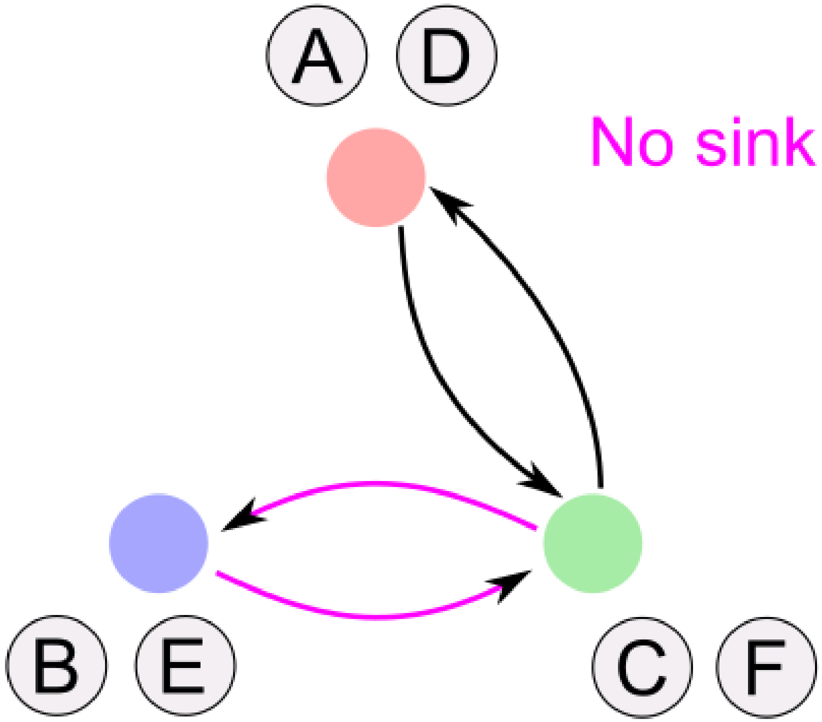


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

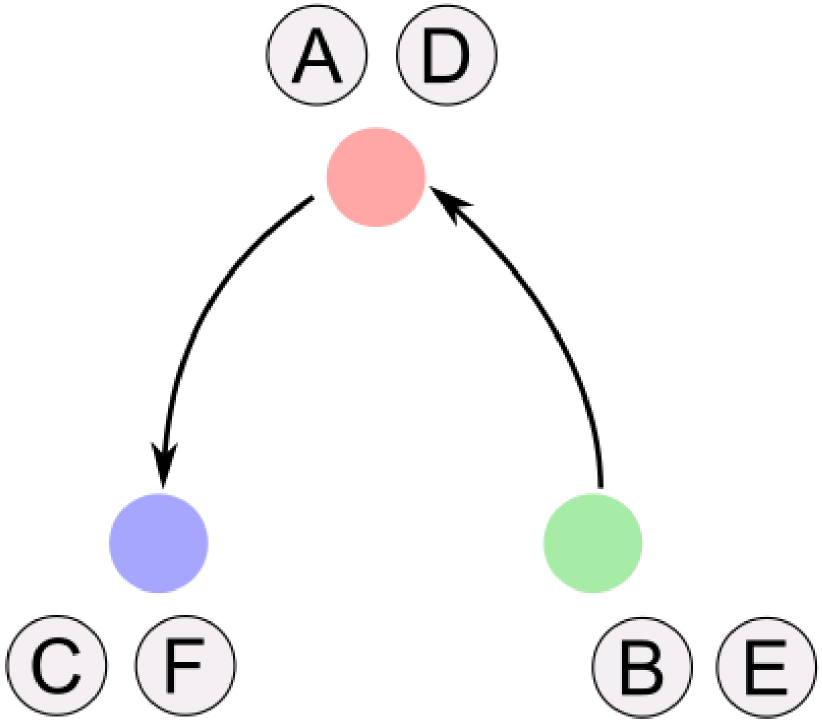


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10

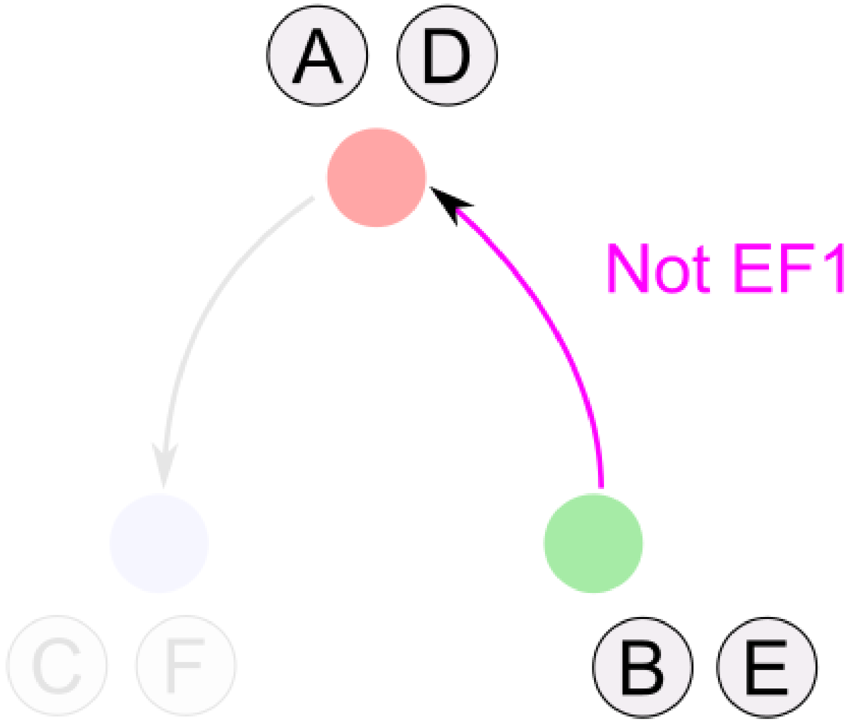


Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

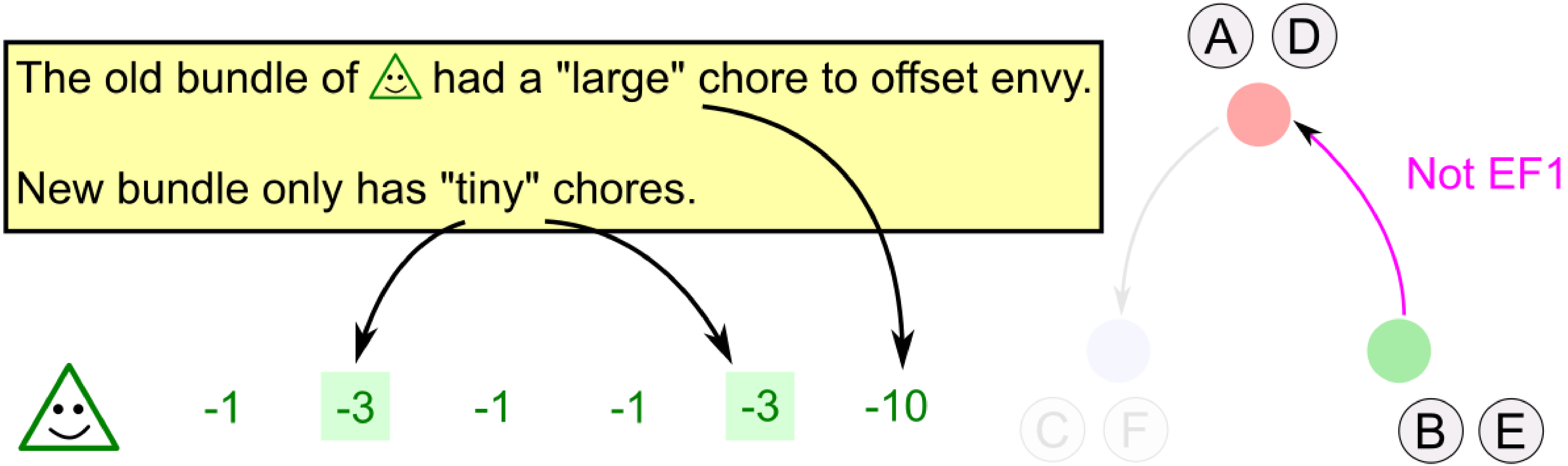
	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10



Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.



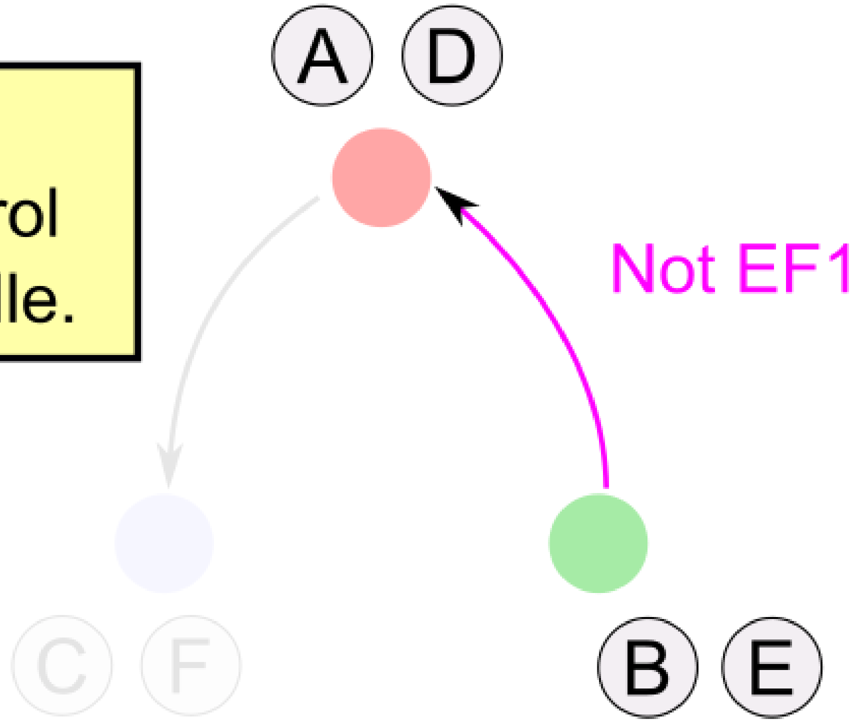
Adapting envy-cycle elimination to chores

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

Source of the problem

Resolving arbitrary envy cycles gives us no control over the size of individual chores in the new bundle.



-1 -3 -1 -1 -3 -10

Adapting envy-cycle elimination to chores

While there is an unallocated chore

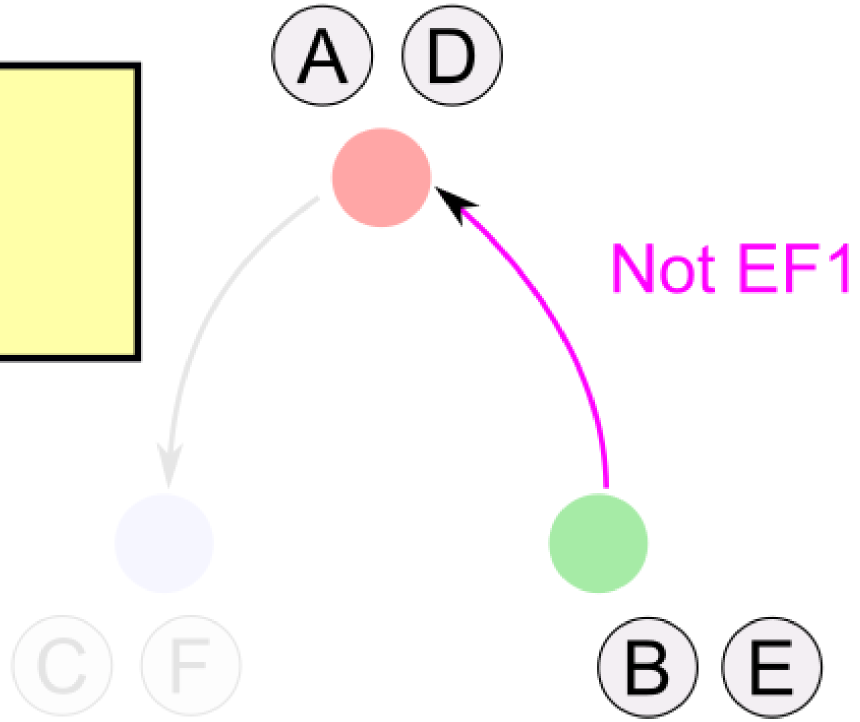
- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve envy cycles until a sink vertex shows up, and then assign the chore to it.

Solution

Resolve top-trading envy cycle

Each agent points to its *favorite* envied bundle

-1 -3 -1 -1 -3 -10



Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

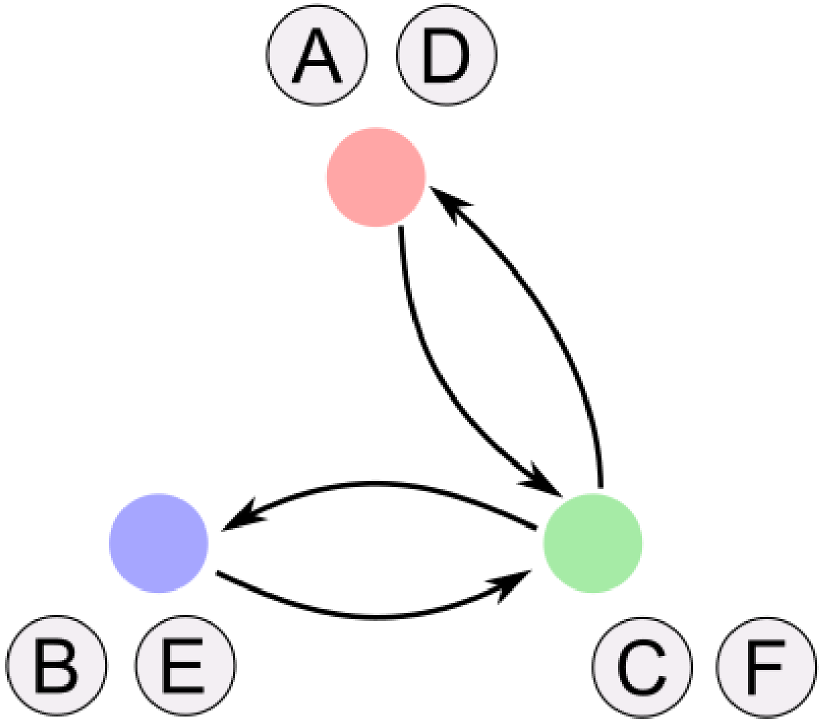
Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10



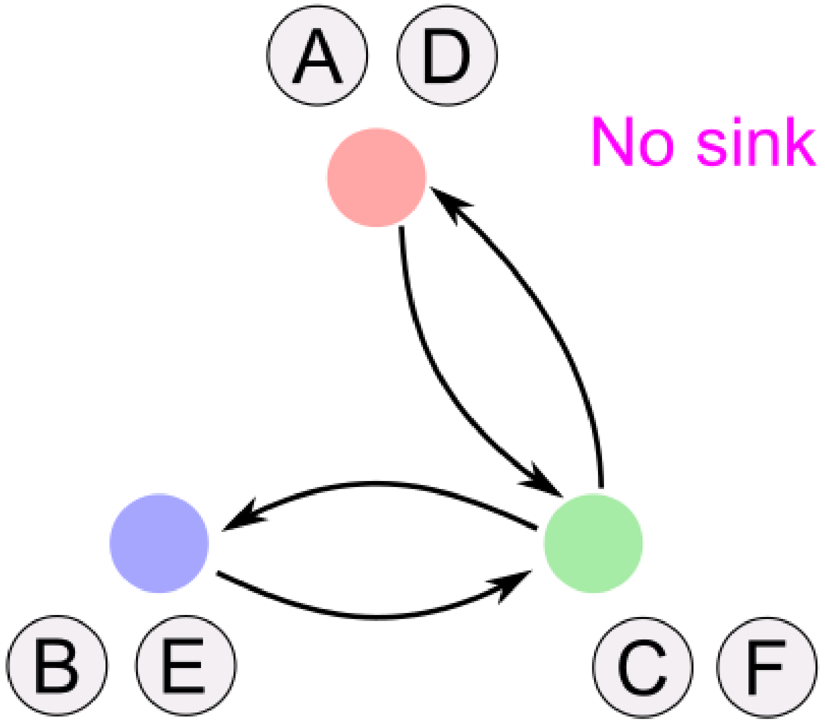
Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10



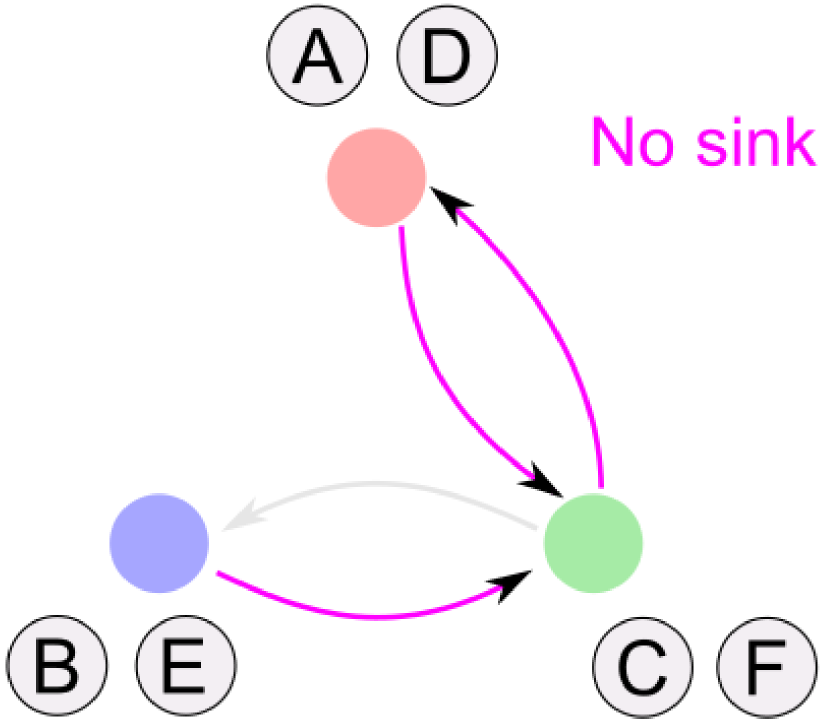
Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10



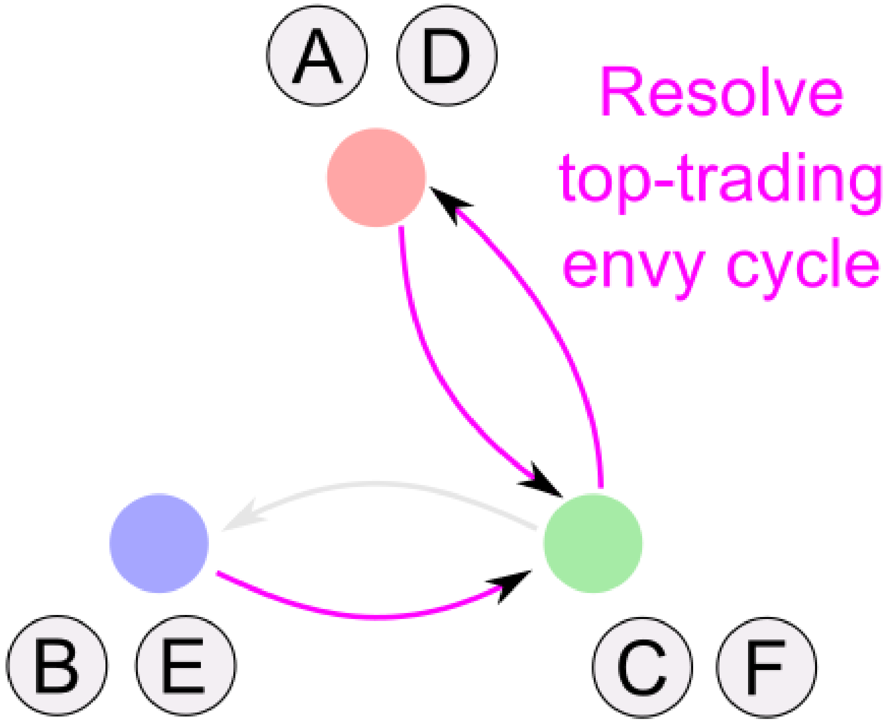
Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10



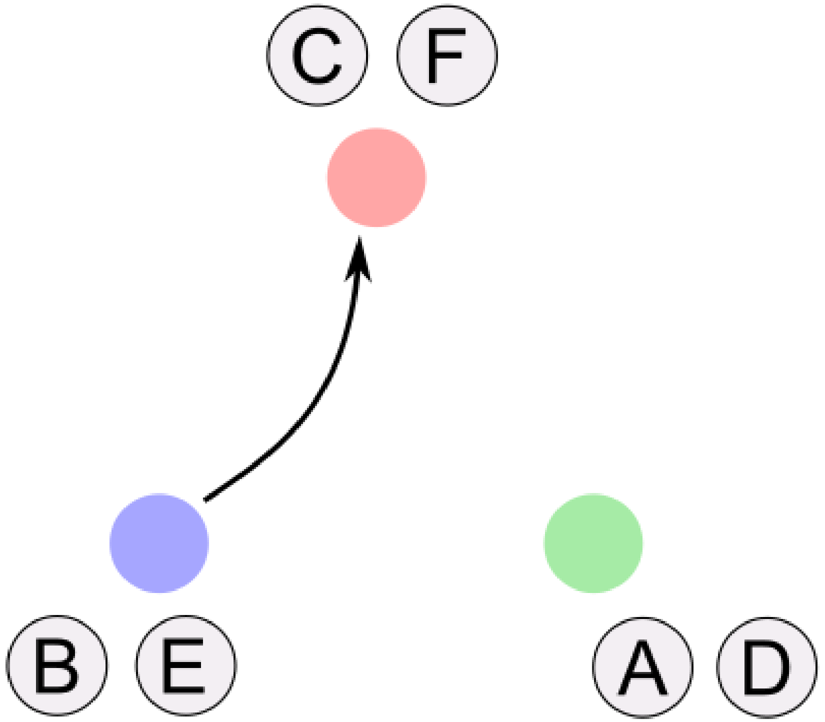
Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

	A	B	C	D	E	F
	-1	-4	-2	-3	0	-1
	-2	-1	-2	-2	-3	-1
	-1	-3	-1	-1	-3	-10



Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Why does a top-trading envy cycle exist when there is no sink?

Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Why does a top-trading envy cycle exist when there is no sink?

No sink ⇒ Every vertex has an outgoing envy edge

Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Why does a top-trading envy cycle exist when there is no sink?

No sink ⇒ Every vertex has ~~an~~ a favorite outgoing envy edge

Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Why does a top-trading envy cycle exist when there is no sink?

No sink ⇒ Every vertex has ~~an~~ a favorite outgoing envy edge

⇒ There is a cycle of "most envied" edges

Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Why does top-trading envy cycle algorithm satisfy EF1?

Top-trading envy-cycle elimination

[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

While there is an unallocated chore

- If the envy graph has a sink vertex, assign the chore to that agent.
- Otherwise, resolve a top-trading envy cycle until a sink vertex shows up, and then assign the chore to it.

Why does top-trading envy cycle algorithm satisfy EF1?

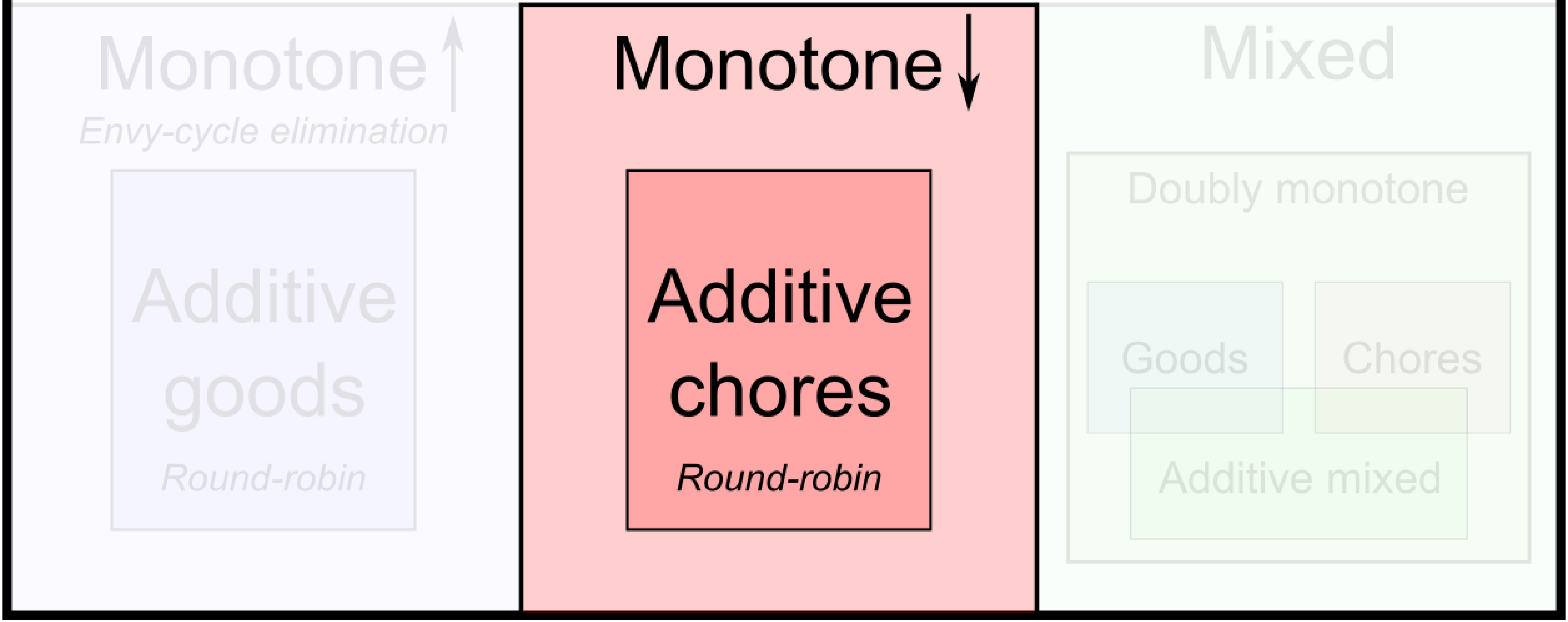
Every vertex in the top-trading cycle becomes envy-free.

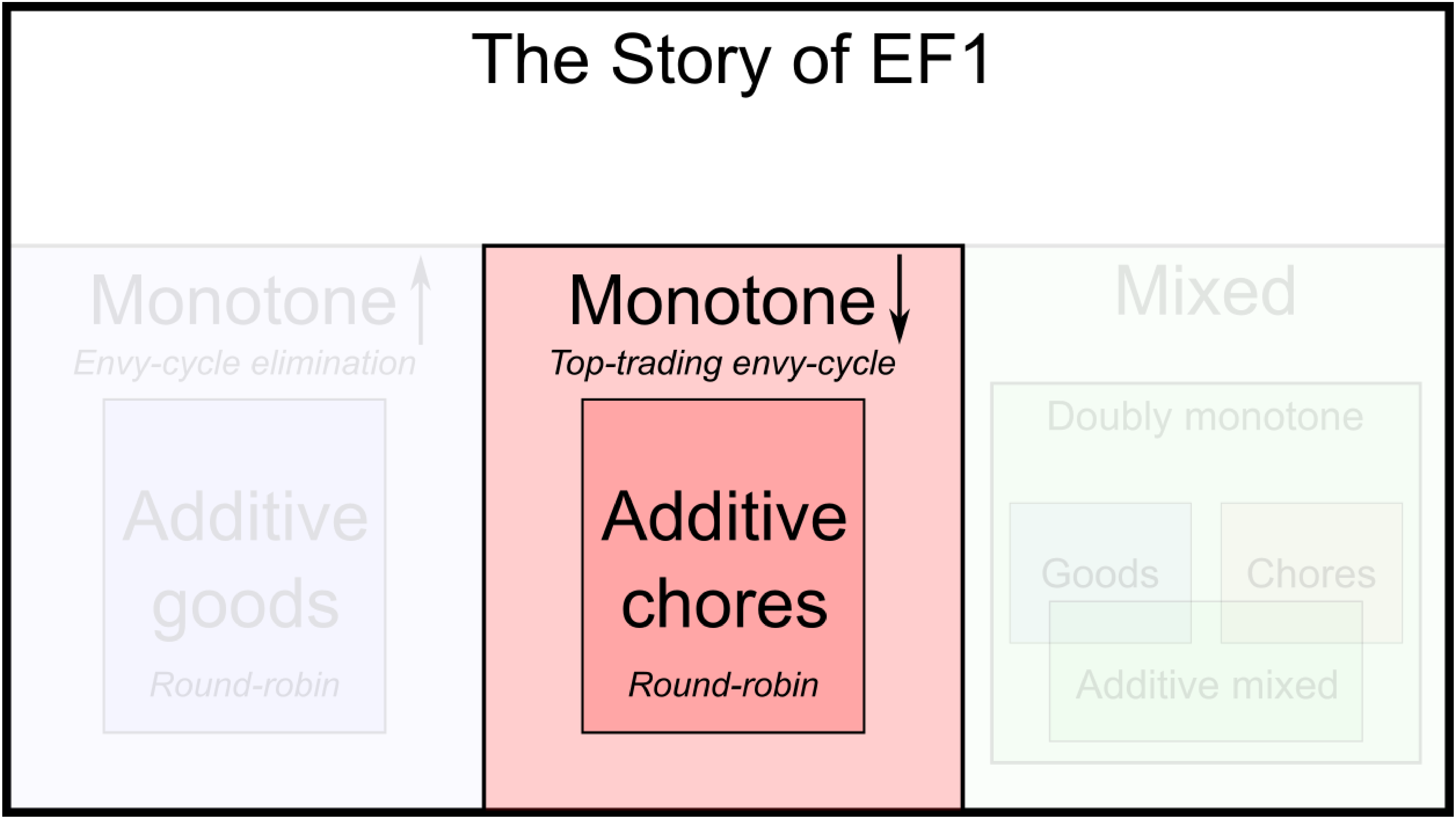
The problem of "new bundle with tiny chores" does not arise.

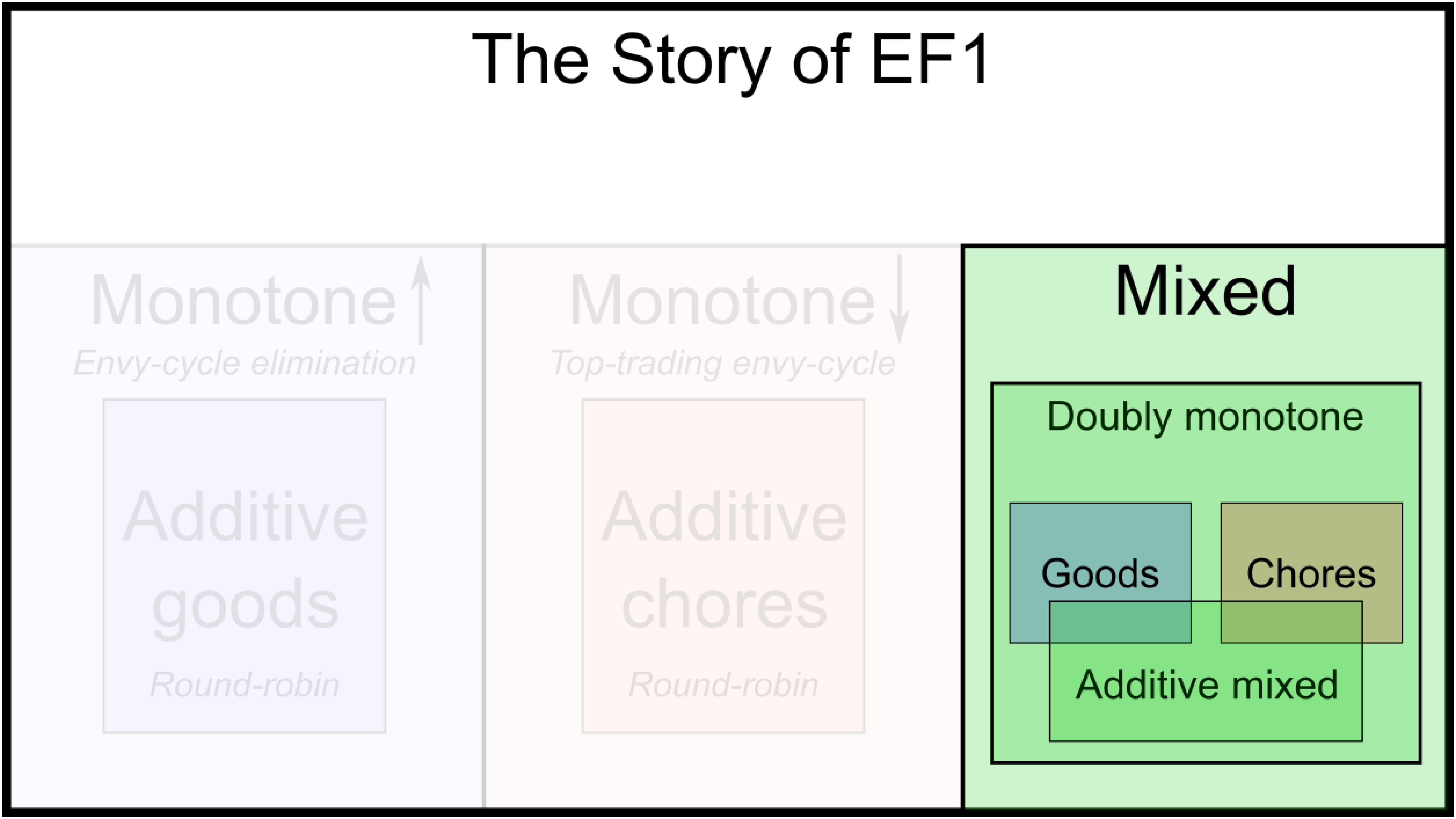
[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

For monotone chores, the allocation computed by the top-trading envy-cycle elimination algorithm satisfies EF1.

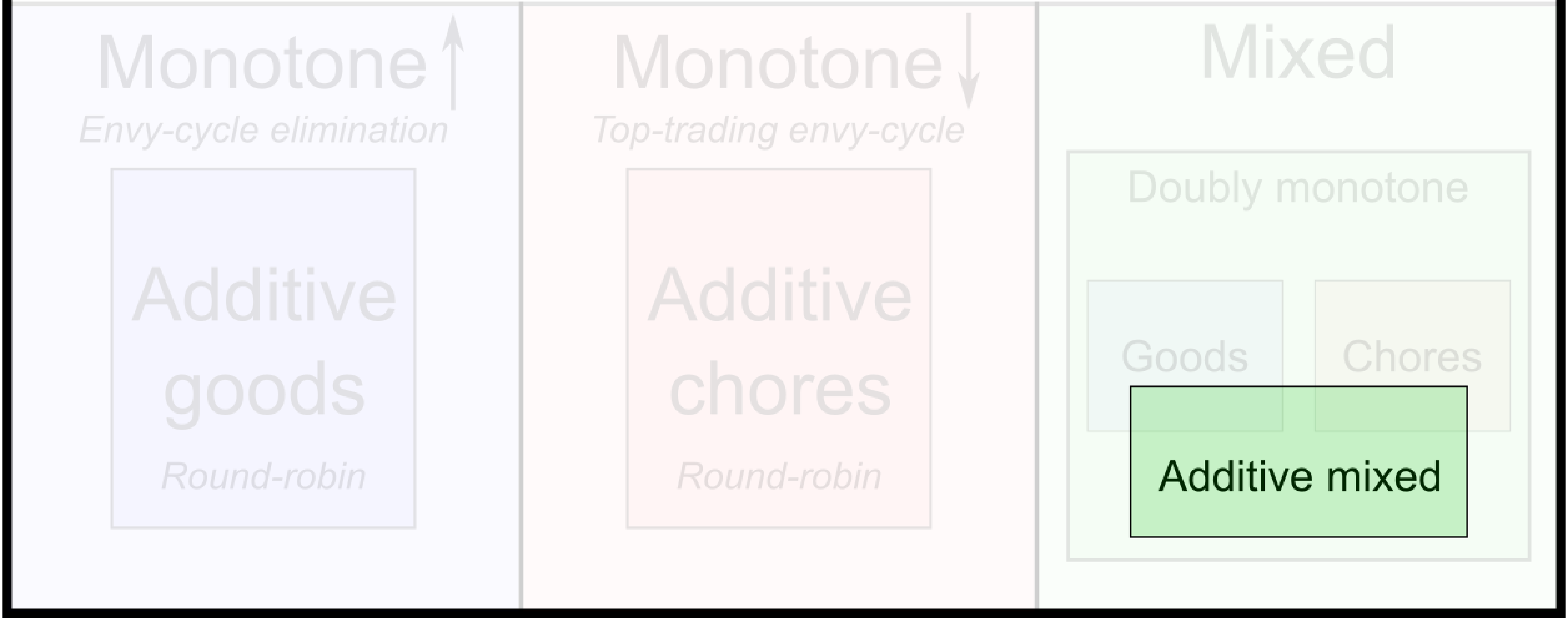
The Story of EF1




The Story of EF1
Monotone↑
Envy-cycle elimination
Additive goods
Round-robin
Monotone↓
Top-trading envy-cycle
Additive chores
Round-robin
Mixed
Doubly monotone
Goods
Chores
Additive mixed


The Story of EF1
Monotone↑
Envy-cycle elimination
Additive goods
Round-robin
Monotone↓
Top-trading envy-cycle
Additive chores
Round-robin
Mixed
Doubly monotone
Goods
Chores
Additive mixed

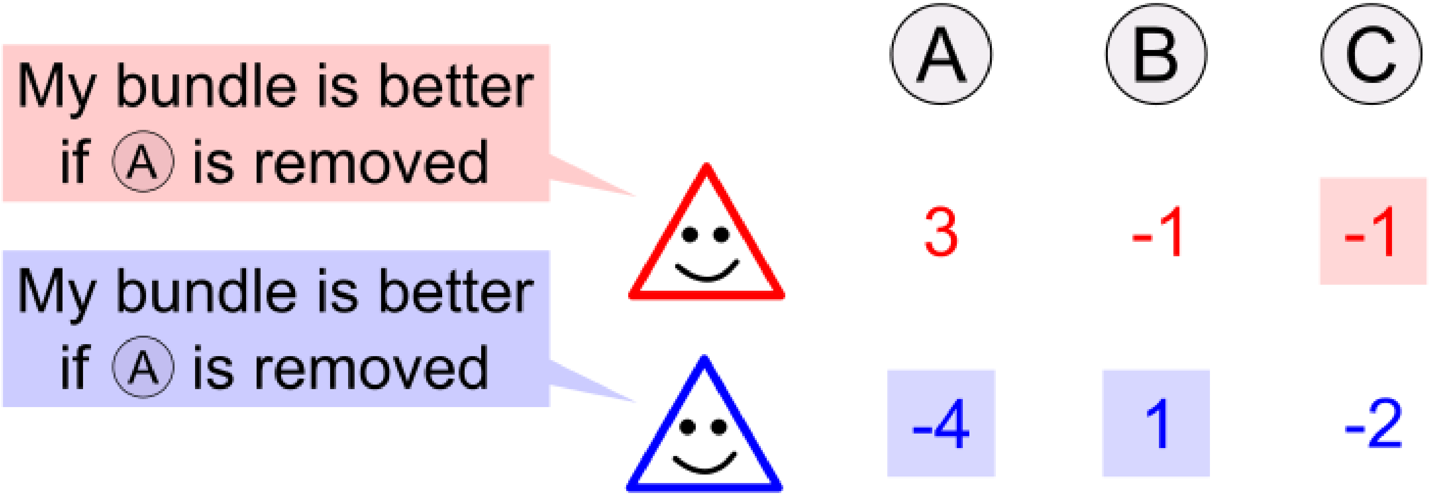
The Story of EF1



Envy-Freeness Up To One Item

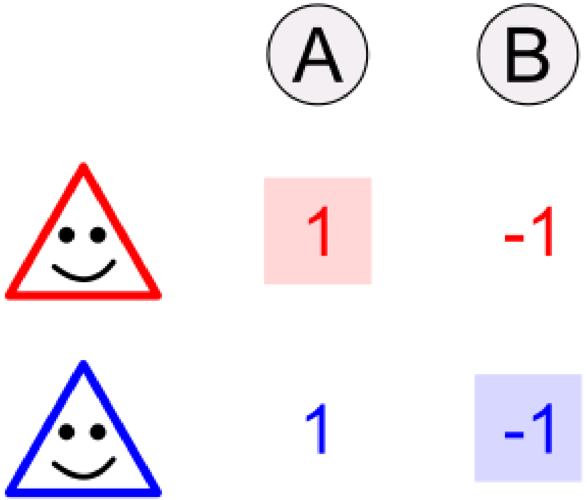
[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

Eliminate envy by removing some "good" in the envied bundle
or some "chore" in the envious agent's bundle.



Allocation $A = (A_1, \ldots, A_n)$ is EF1 if for every pair of agents i, k, there exists an item $j \in A_i \cup A_k$ s.t. $v_i(A_i \setminus \{j\}) \geq v_i(A_k \setminus \{j\})$.

For goods+chores, naive round-robin fails EF1.



Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

Partition the items into two sets: positive and negative

Positive: items with strictly positive value for at least one agent

(considered to be a "good" by at least one agent)

Negative: all other items

(considered a "chore" by all agents)

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

	A	B	C	D	E
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

	A	B	C	D	E
	-	-	-	+	+
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

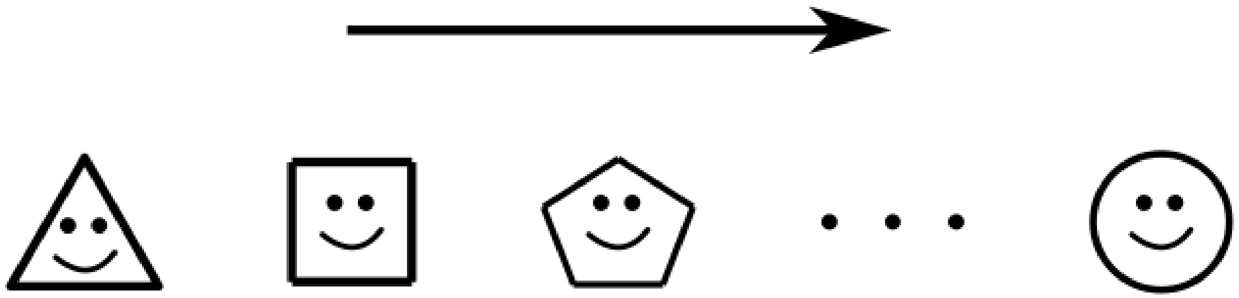
Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

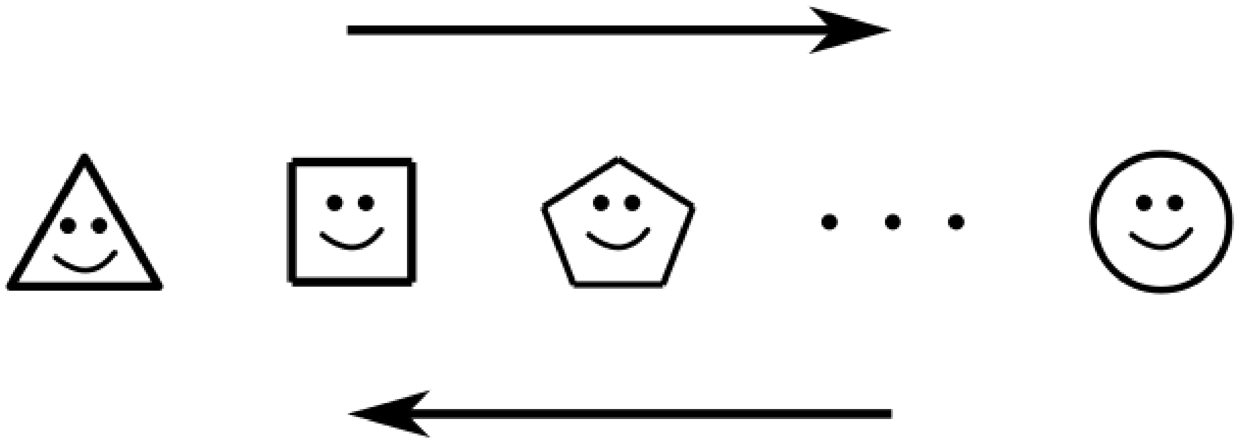
allocate negative items in this order



Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

allocate negative items in this order



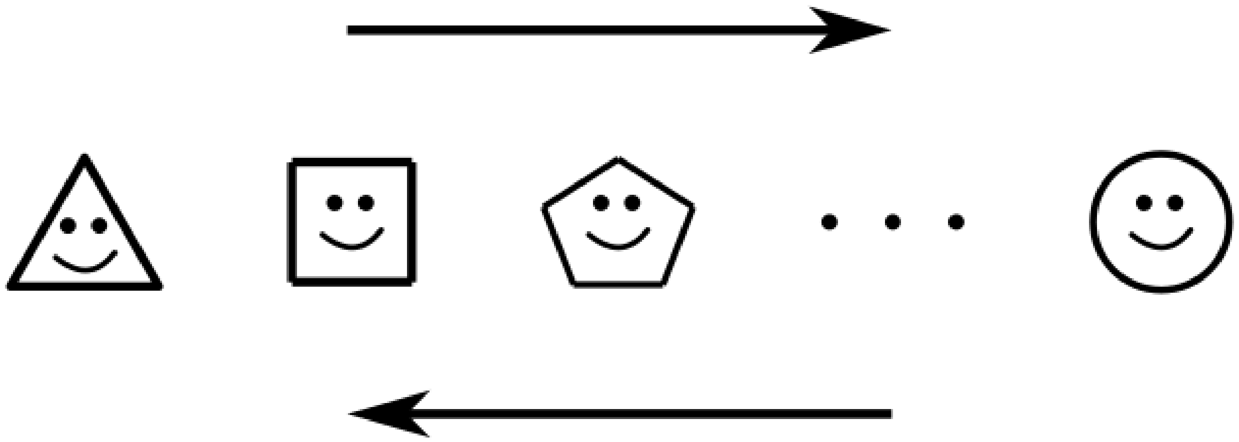
and positive items in the opposite order

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

No. of negative items is an integer multiple of n (add zero valued items)

allocate negative items in this order



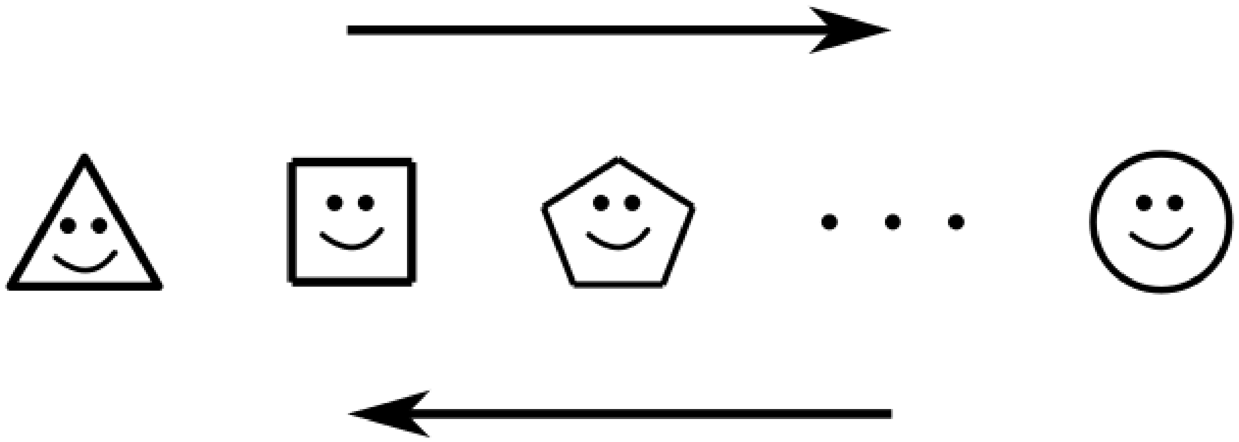
and positive items in the opposite order

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

allocate negative items in this order

No. of negative items is an integer multiple of n (add zero valued items)



and positive items in the opposite order

Picking with skipping

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

	-	-	-	+	+
	A	B	C	D	E
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

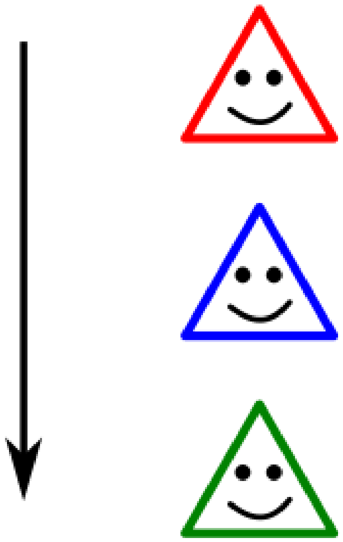
Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

	-	-	-	+	+
	A	B	C	D	E
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

Double round-robin algorithm

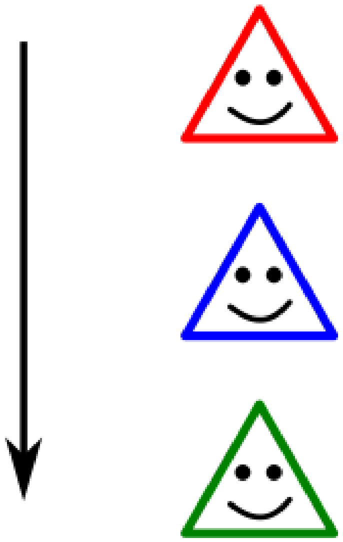
[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]



	-	-	-	+	+
	A	B	C	D	E
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

Double round-robin algorithm

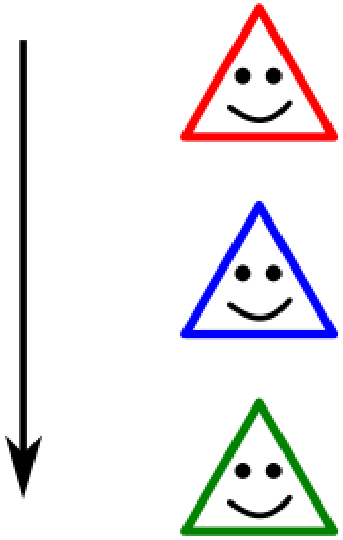
[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]



-	-	-	+	+
A	B	C	D	E
-4	-1	-2	2	-4
0	-1	-5	-2	-1
-4	-2	-5	0	2

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]



	A	B	C	D	E
	-	-	-	+	+
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

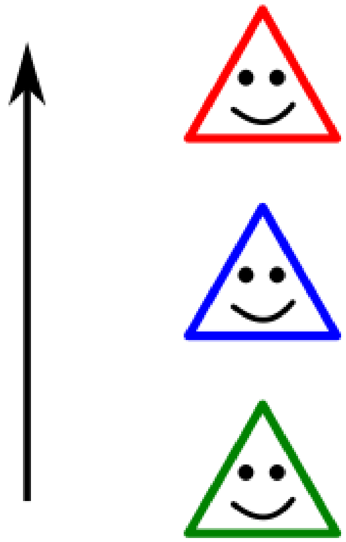
Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]

	-	-	-	+	+
	A	B	C	D	E
	-4	-1	-2	2	-4
	0	-1	-5	-2	-1
	-4	-2	-5	0	2

Double round-robin algorithm

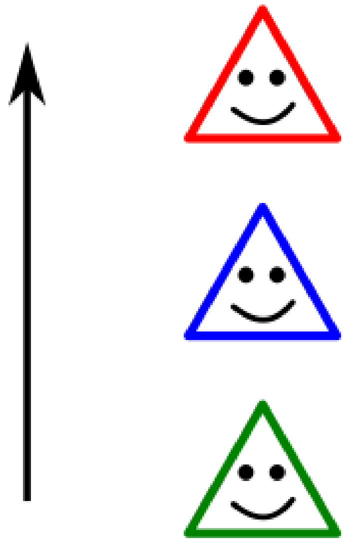
[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]



-	-	-	+	+
A	B	C	D	E
-4	-1	-2	2	-4
0	-1	-5	-2	-1
-4	-2	-5	0	2

Double round-robin algorithm

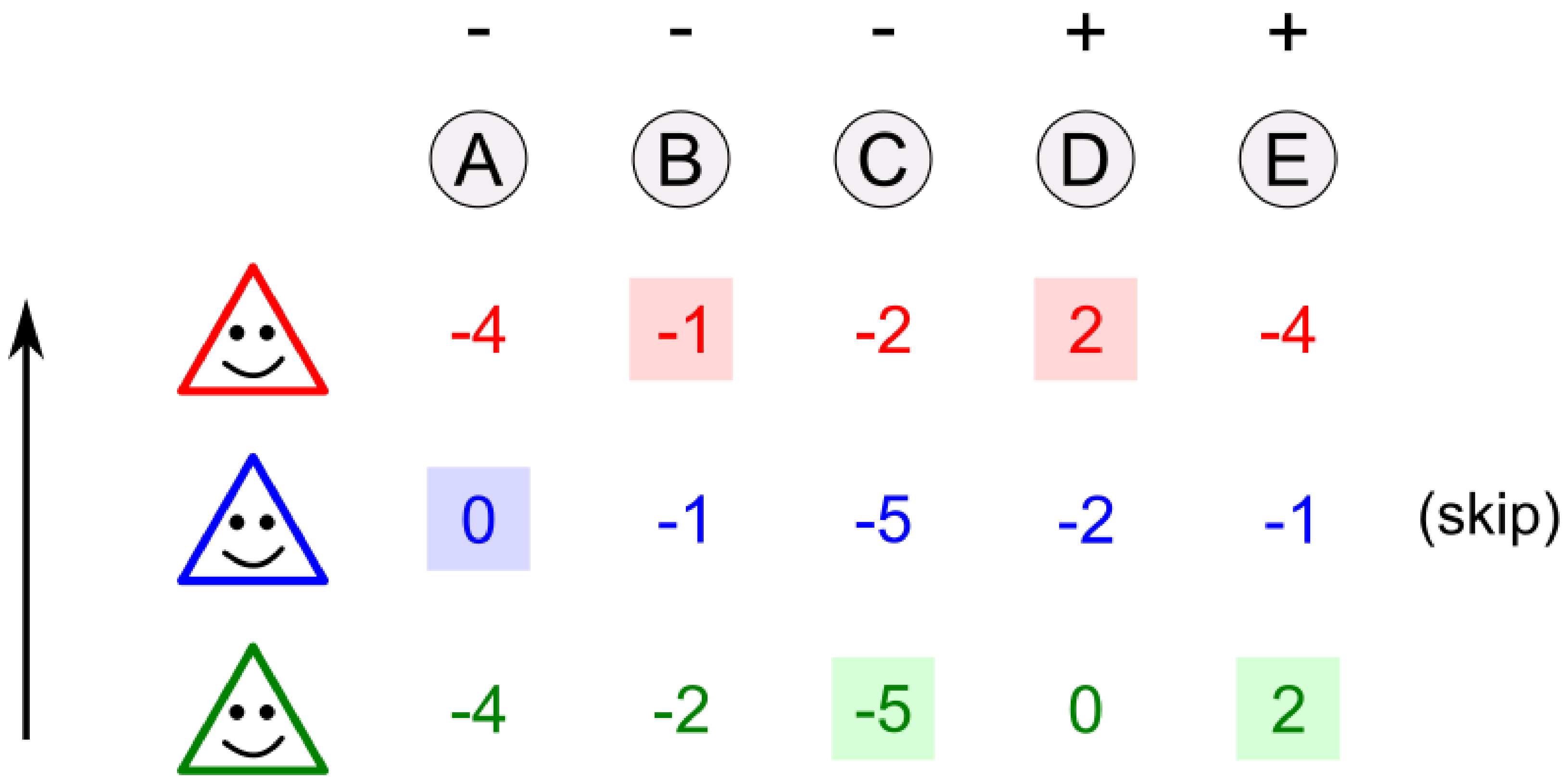
[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]



	A	B	C	D	E	
	-	-	-	+	+	
red agent	-4	-1	-2	2	-4	
blue agent	0	-1	-5	-2	-1	(skip)
green agent	-4	-2	-5	0	2	

Double round-robin algorithm

[Aziz, Caragiannis, Igarashi, and Walsh; *IJCAI* 2019; *JAAMAS* 2022]



Why does double round-robin algorithm satisfy EF1?

Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.

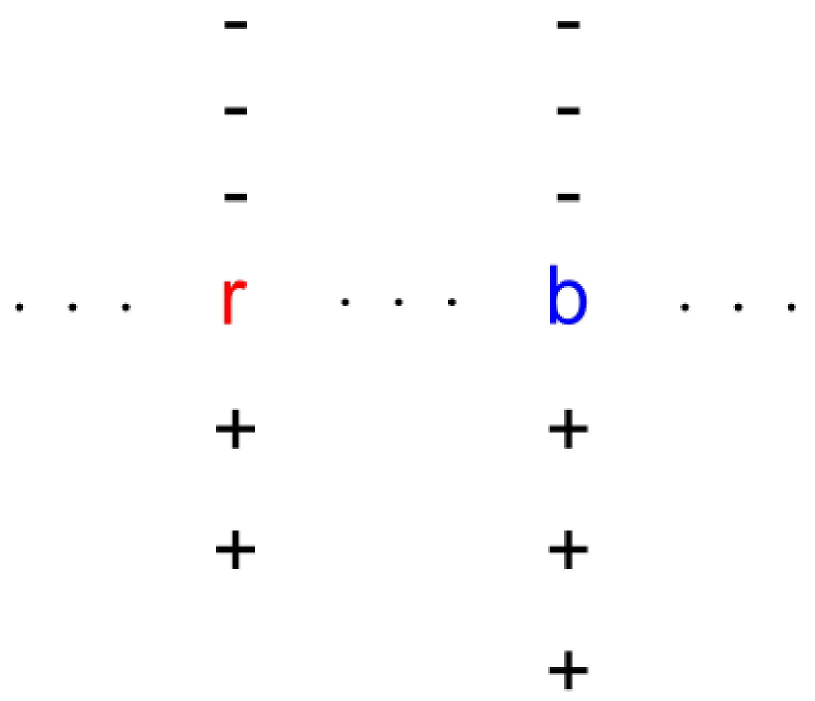
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.

. . . r . . . b . . .

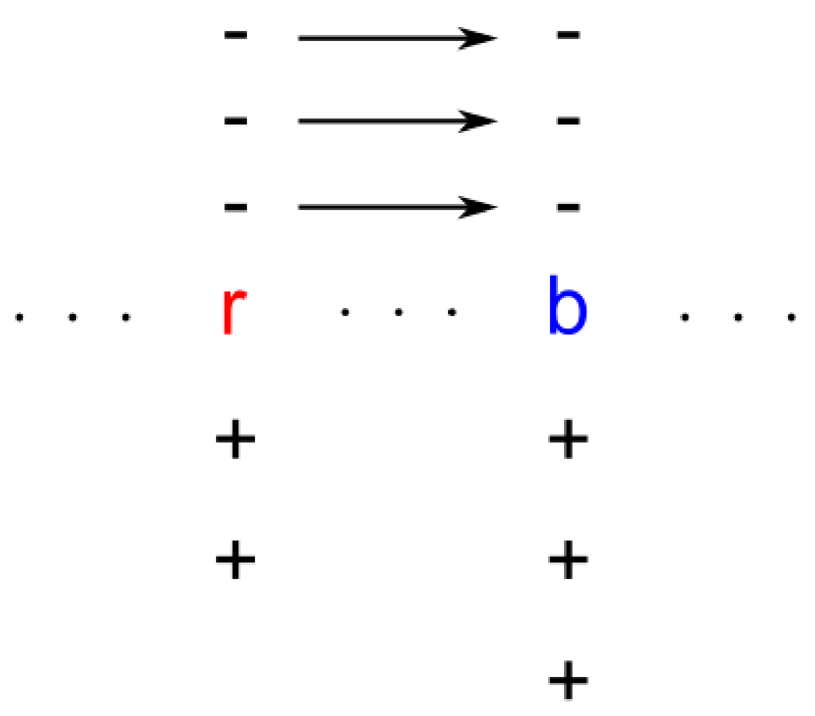
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.



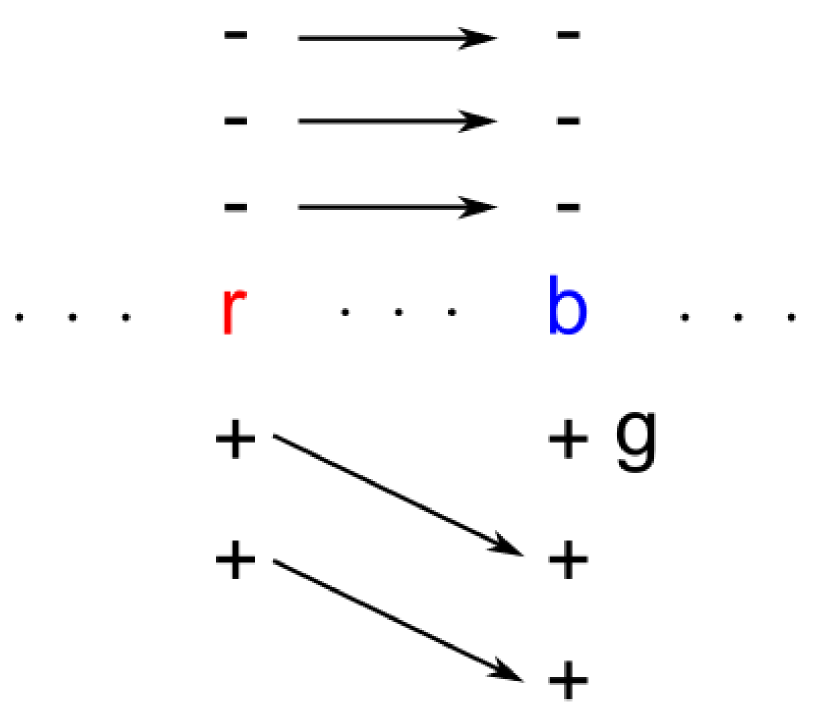
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.



Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.



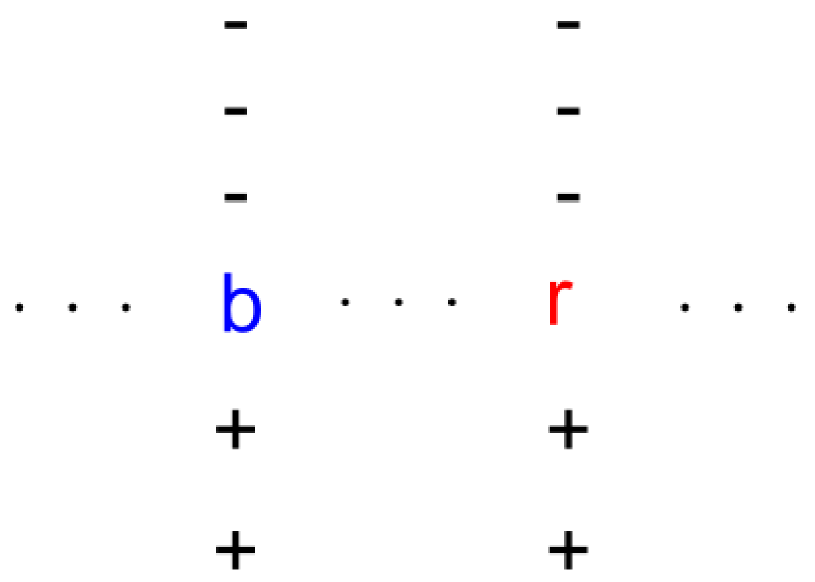
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.

. . . b . . . r . . .

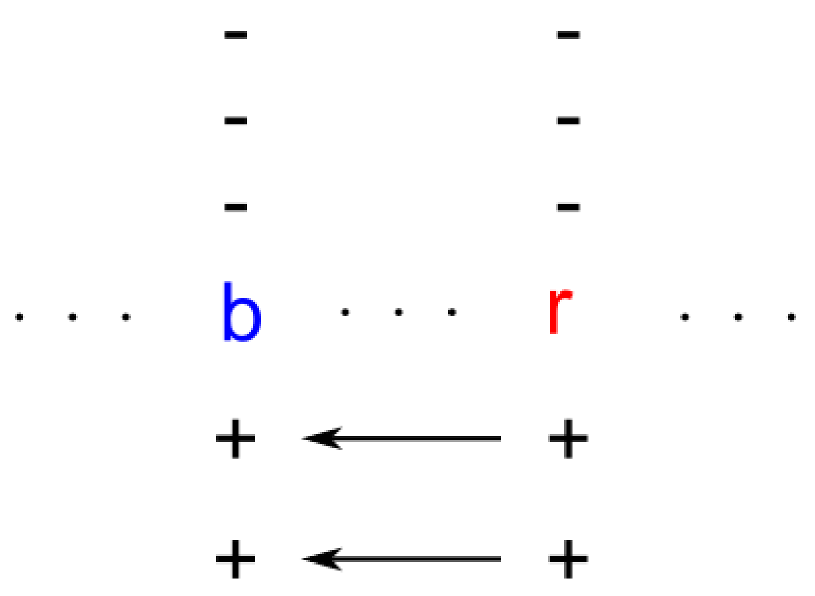
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.



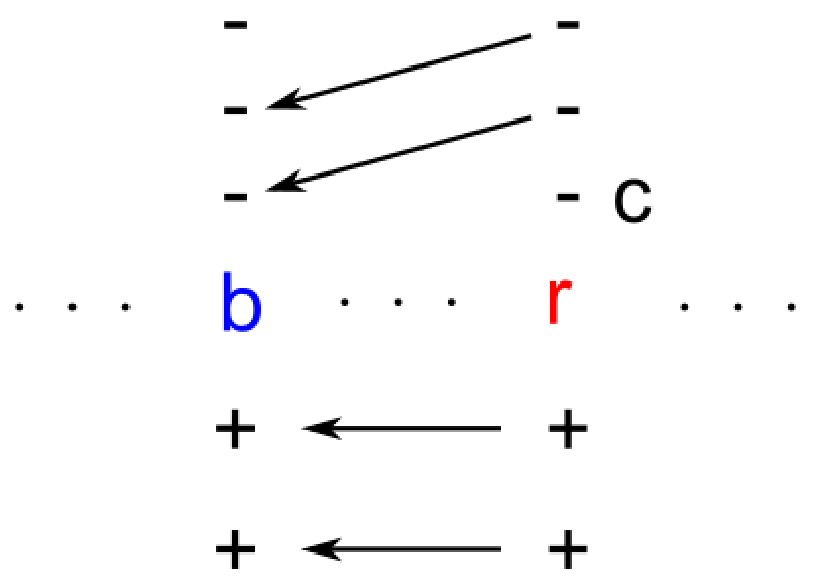
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.



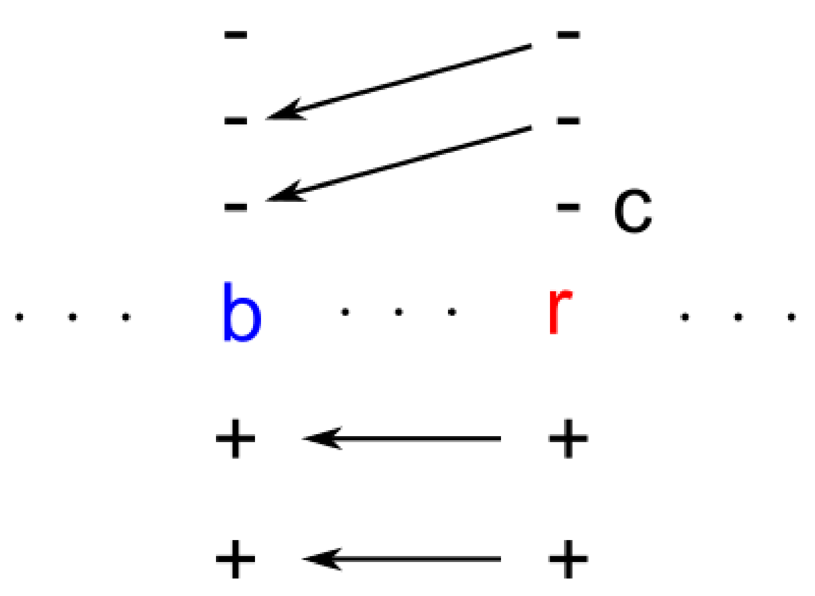
Why does double round-robin algorithm satisfy EF1?

Fix a pair of agents (r,b). Analyze envy of r towards b.

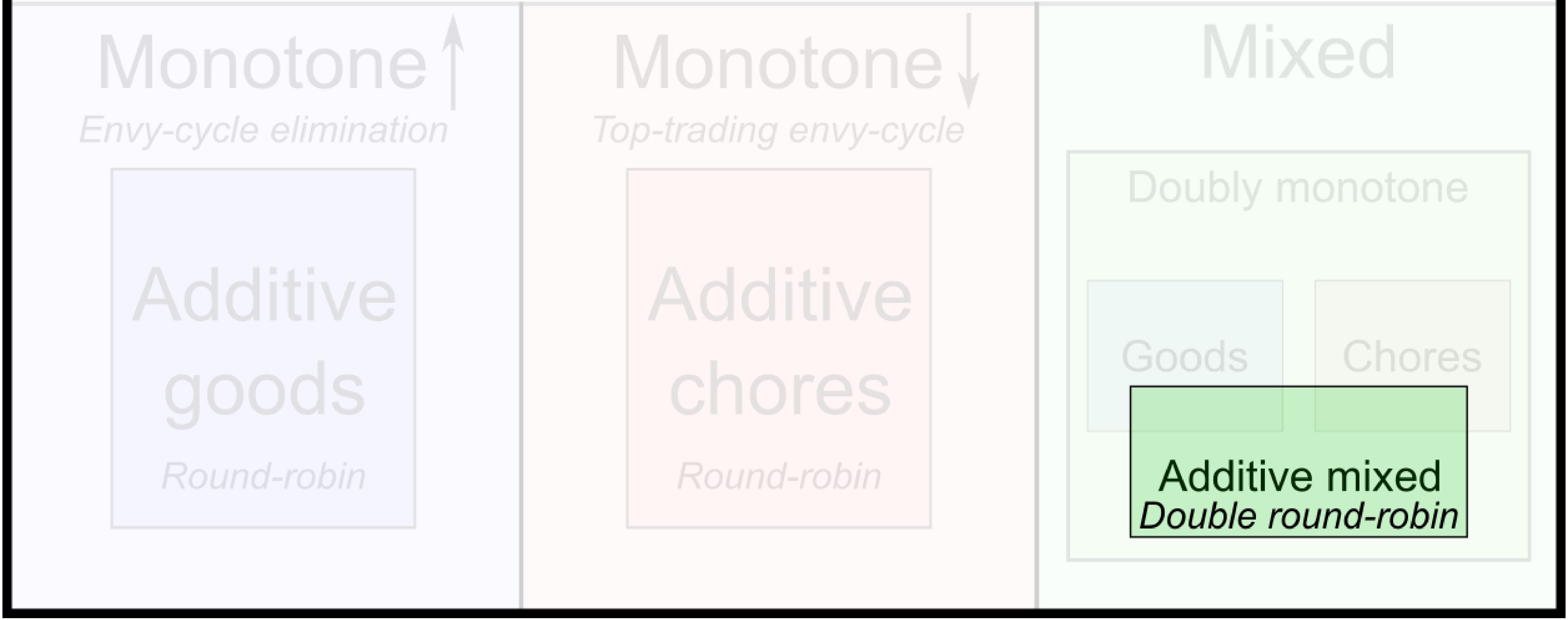


Why does double round-robin algorithm satisfy EF1?

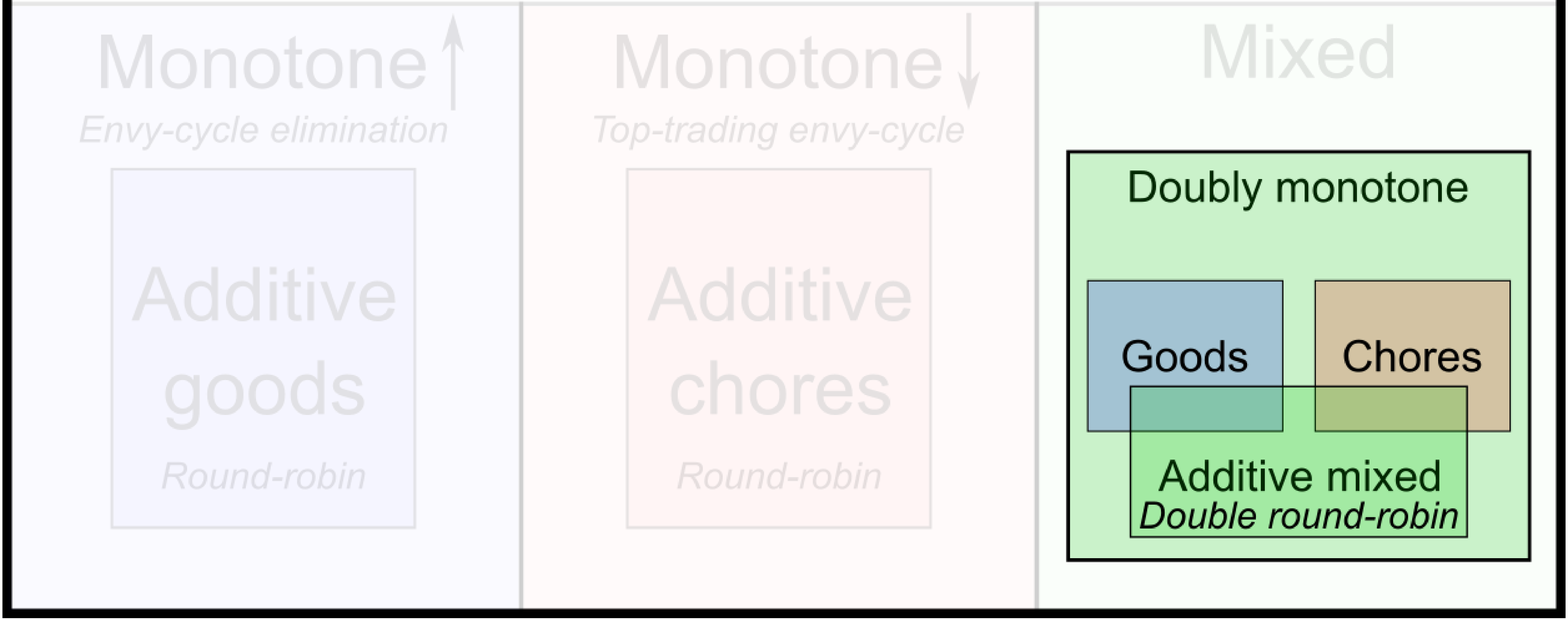
Fix a pair of agents (r,b). Analyze envy of r towards b.



The Story of EF1



The Story of EF1



Doubly Monotone Valuations

Doubly Monotone Valuations

Each agent can partition the items into "goods" and "chores".

marginal ≥ 0

marginal ≤ 0

Doubly Monotone Valuations

Each agent can partition the items into "goods" and "chores".

marginal ≥ 0

marginal ≤ 0

	A	B	C	D	E
	-	-	-	+	+
	-	+	+	+	+
	-	+	-	+	-

EF1 for Doubly Monotone Valuations

Partition the items into two sets: positive and negative

Positive: items considered "good" by at least one agent

Negative: items considered "chore" by everyone

EF1 for Doubly Monotone Valuations

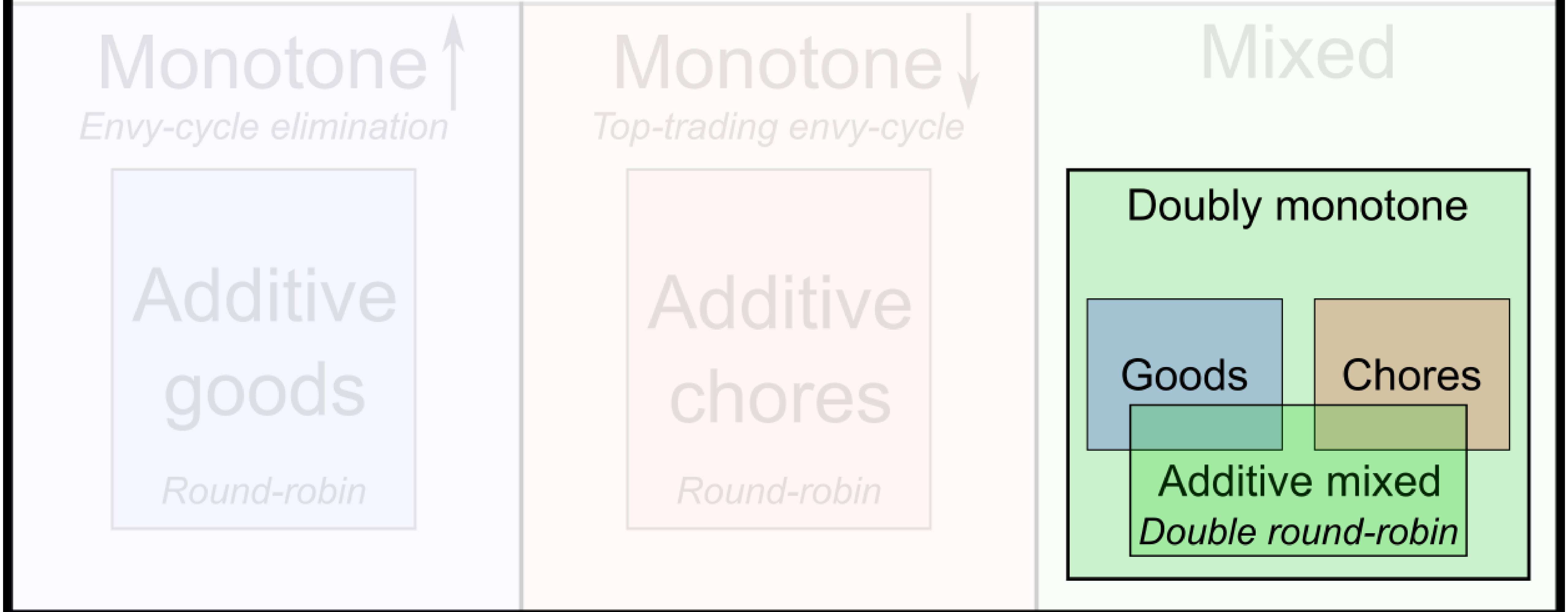
- Assign positive items via envy-cycle elimination
(envy graph defined w.r.t. agents who consider the item a "good")

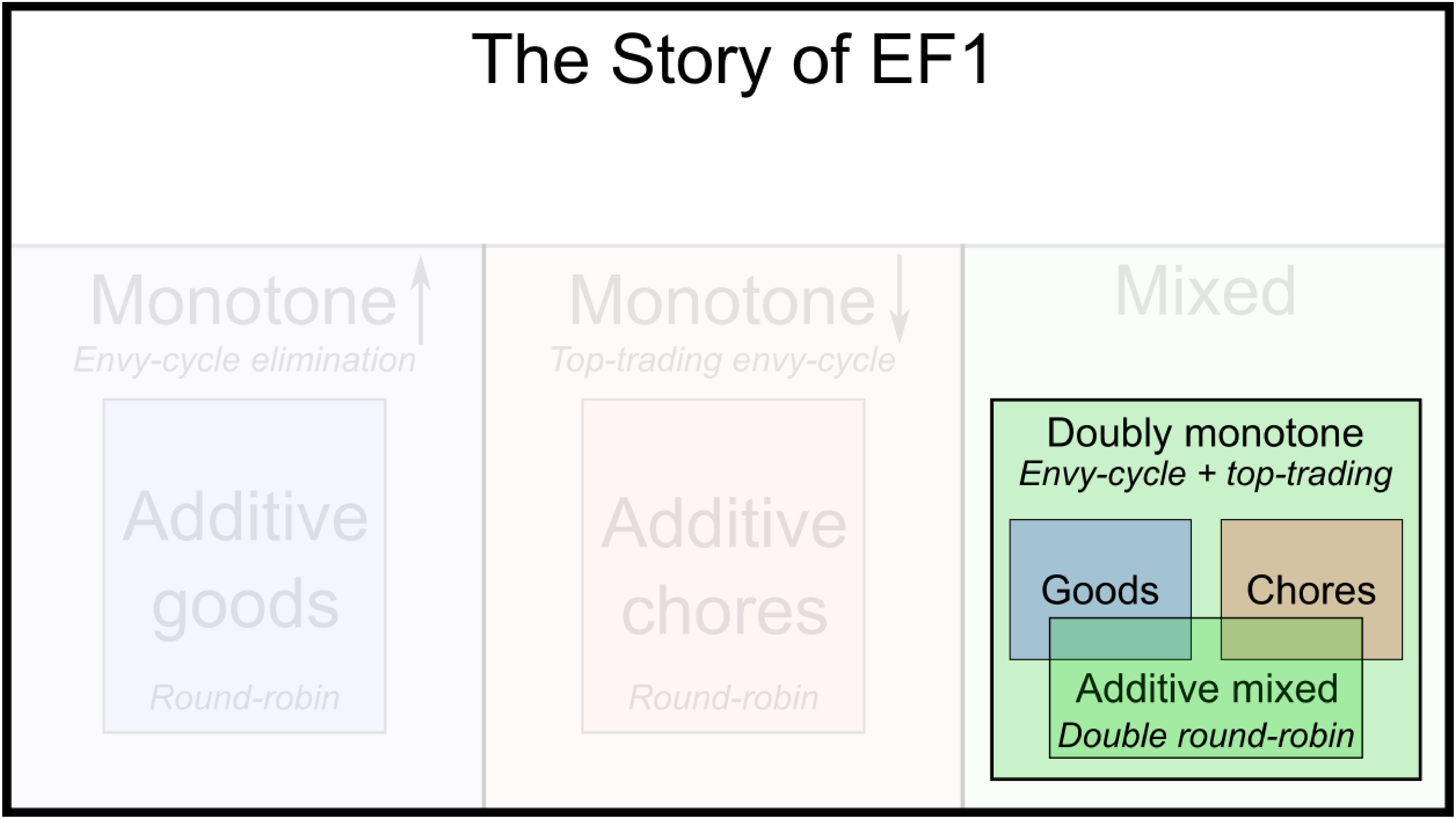
- Assign negative items via top-trading envy-cycle elimination

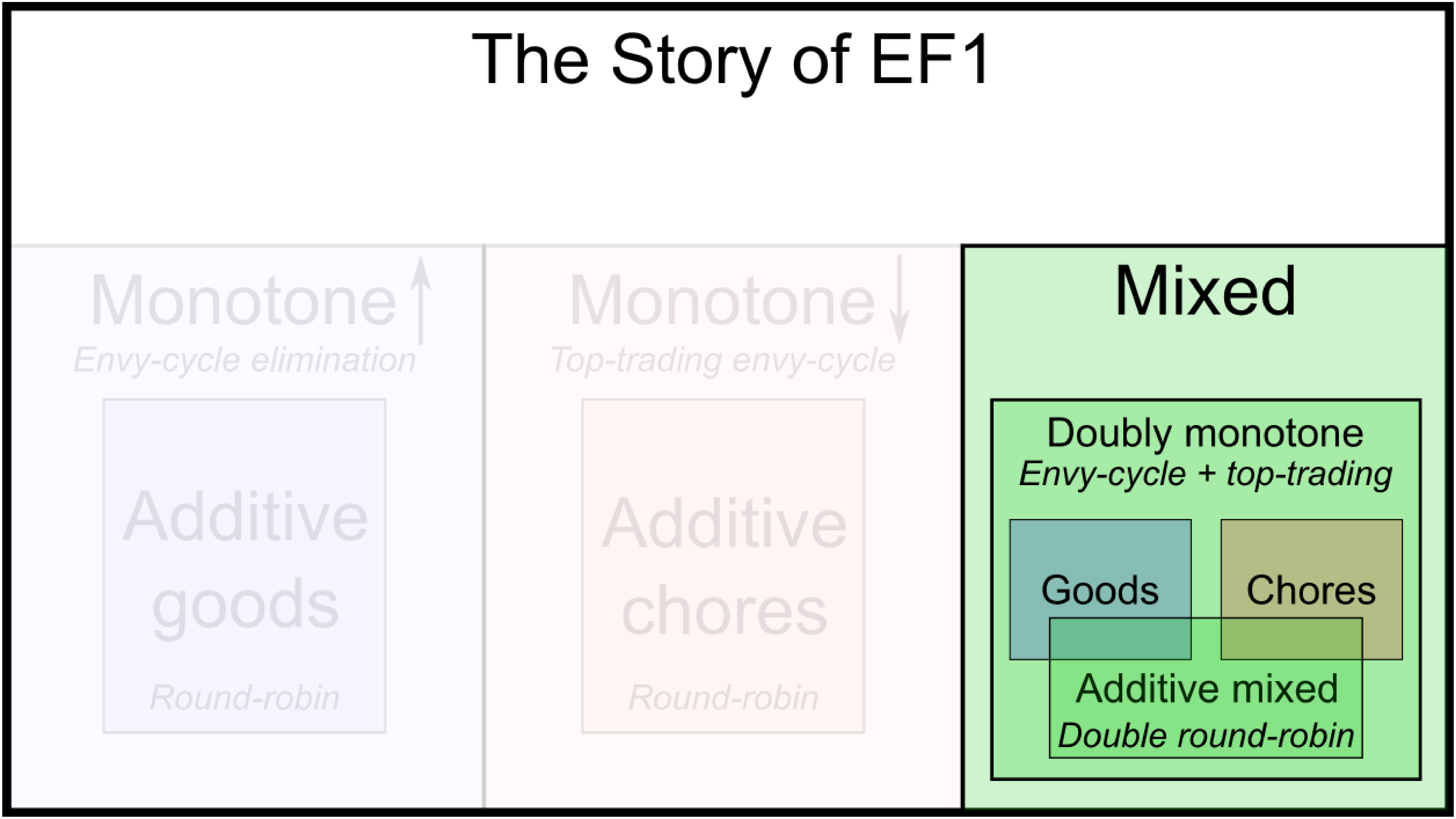
[Bhaskar, Sricharan, and Vaish, *APPROX* 2021]

For doubly monotone items, the above algorithm returns an EF1 allocation.

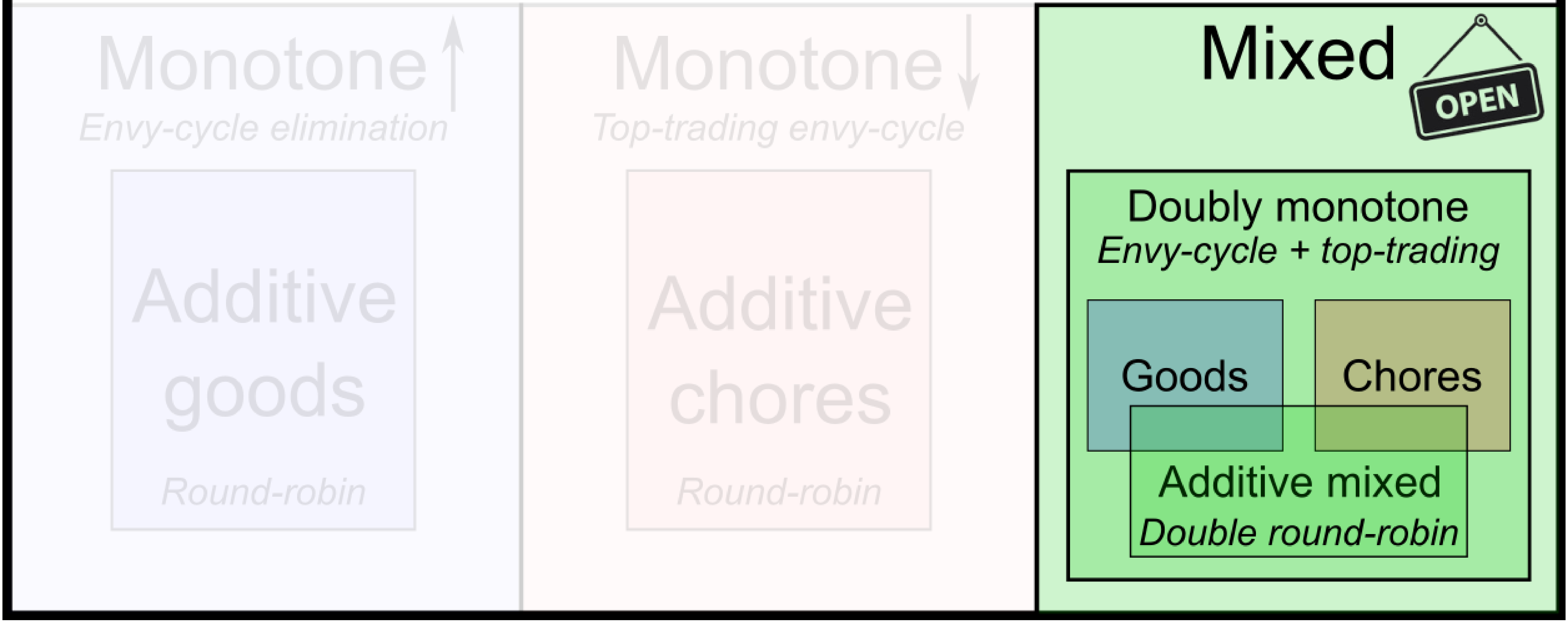
The Story of EF1




The Story of EF1
Monotone↑
Envy-cycle elimination
Additive goods
Round-robin
Monotone↓
Top-trading envy-cycle
Additive chores
Round-robin
Mixed
Doubly monotone
Envy-cycle + top-trading
Goods
Chores
Additive mixed
Double round-robin


The Story of EF1
Monotone↑
Envy-cycle elimination
Additive goods
Round-robin
Monotone↓
Top-trading envy-cycle
Additive chores
Round-robin
Mixed
Doubly monotone
Envy-cycle + top-trading
Goods
Chores
Additive mixed
Double round-robin

The Story of EF1



Next Time

Fairness via Randomness

Quiz

Quiz

Prove or disprove:

For n identical agents with additive valuations over mixed items, an EFX allocation always exists.

References

- Double round-robin algorithm

  Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi and Toby Walsh
  "Fair allocation of indivisible goods and chores"
  Autonomous Agents and Multiagent Systems, 36(3), 2022 pg 1-21
  https://link.springer.com/article/10.1007/s10458-021-09532-8

- Top-trading envy-cycle elimination

  Umang Bhaskar, A R Sricharan, and Rohit Vaish
  "*On approximate envy-freeness for indivisible chores and mixed resources*"
  APPROX 2021
  https://drops.dagstuhl.de/opus/volltexte/2021/14694/