Fair Allocation Problems

Lecturer: Cosimo Vinci
Course: Social Network Analysis
M.D. in Computer Engineering, University of Salerno

April 4, 2022

From Welfare Maximisation to Fairness

The goal in designing auctions for markets is optimising a social welfare function (e.g., the sum of the agents' valuations)

From Welfare Maximisation to Fairness

The goal in designing auctions for markets is optimising a social welfare function (e.g., the sum of the agents' valuations)

Example:

- We have two agents {1,2} and two goods $\{g_1, g_2\}$ that can assigned to the agents, and the valuations of the agents for the goods are:
  $v_1(g_1) = 3,\ v_1(g_2) = 2,\ v_2(g_1) = 0.5,\ v_2(g_2) = 1$.
- If we aim at maximising the total utility, we should assign all the goods to agent 1.
- However, agent 2 will desire at least one piece, that is, agent 2 will be envious of agent 1.
- A fairer allocation is obtained by assigning good g_1 to agent 1 and good g_2 to agent 2.

From Welfare Maximisation to Fairness

In this lecture, we will focus on fairness aspects of markets/allocations.

From Welfare Maximisation to Fairness

In this lecture, we will focus on fairness aspects of markets/allocations.

Our new question/problem is the following:

How to fairly allocate (divisible or indivisible) goods among people?

Fair Cake Cutting Problem

- A set of n agents $N = 1,2,\ldots,n$.

- A cake (or divisible good) $C = [0,1]$

Fair Cake Cutting Problem

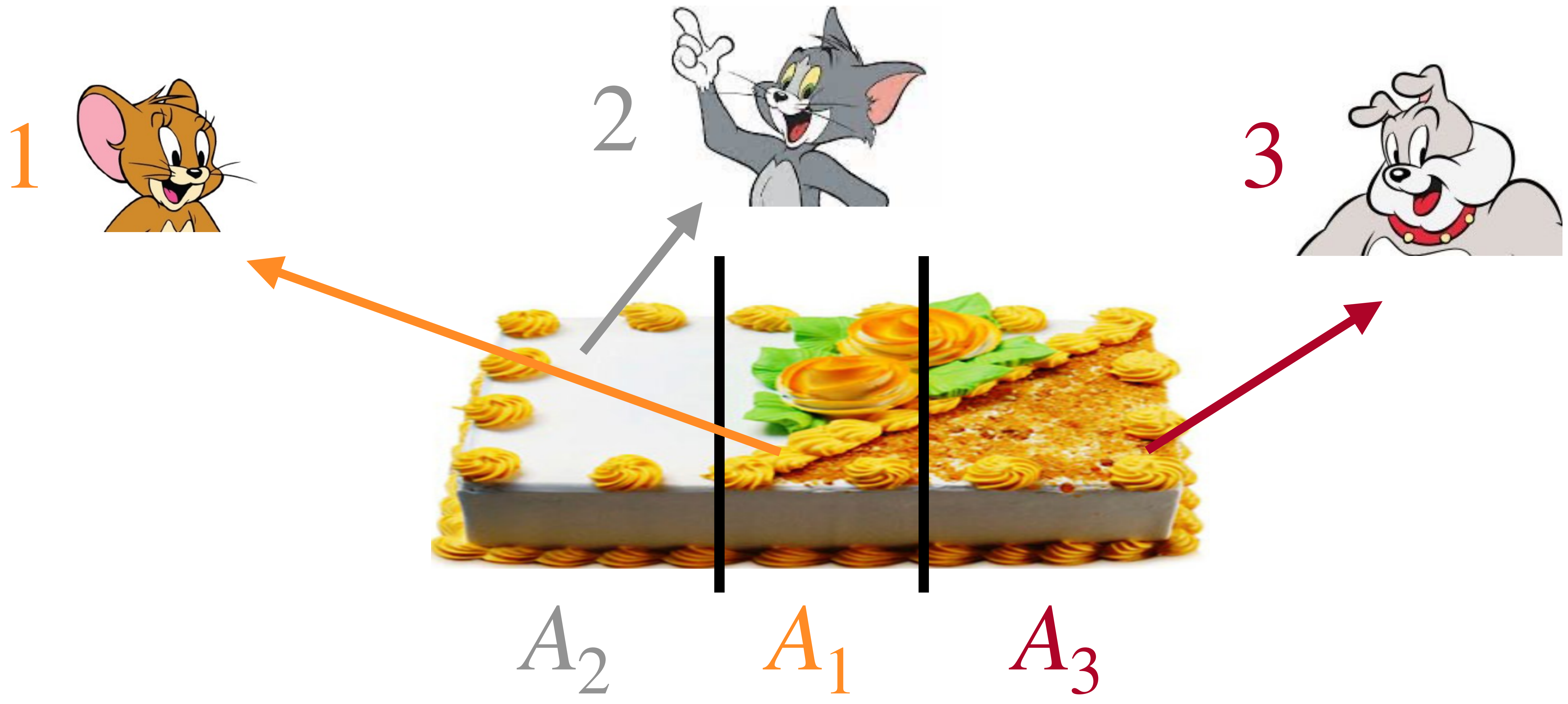
- A set of n agents $N = 1,2,\ldots,n$.

- A cake (or divisible good) $C = [0,1]$

- Each agent has valuation function v_i, that assigns real value $v_i(P)$ to each piece/interval $P = [a, b] \subseteq [0,1]$.
- Valuations are monotone: $0 = v_i(\emptyset) \leq v_i(A) \leq v_i(B)$, if $A \subseteq B$
- Valuations are continuous: small variations of the pieces imply small variations of the values.

Fair Cake Cutting Problem

- A connected division of the cake is a partition $\mathscr{A} = \{A_1, \ldots, A_n\}$ of C in n pieces/intervals , and each piece A_i is assigned to a distinct agent i.
- A connected division $\mathscr{A}$ is envy-free if $v_i(A_i) \geq v_i(A_j)$ for any agents i, j.



Real-life Applications

- The "cake" is only a metaphor.
- Procedures for fair cake-cutting can be used to divide various kinds of resources, such as:
    - land estates;
    - advertisement space;
    - broadcast time.

A look into the past

- The prototypical procedure for fair cake-cutting is Divide (or Cut) and Choose.
- The Divide and Choose Protocol is mentioned already in the book of Genesis.
- During World War II, Hugo Steinhaus, who was hiding from the Nazis, occupied himself with the question of how to divide resources fairly.
- Inspired by the "Divide and Choose" procedure for dividing a cake between two brothers, he asked his students (Stefan Banach and Bronisław Knaster), to find a procedure that can work for any number of people…they obtained the following seminal work:
    - Steinhaus, Hugo (1948). "The problem of fair division". Econometrica. 16 (1): 101–4.

Envy-free divisions

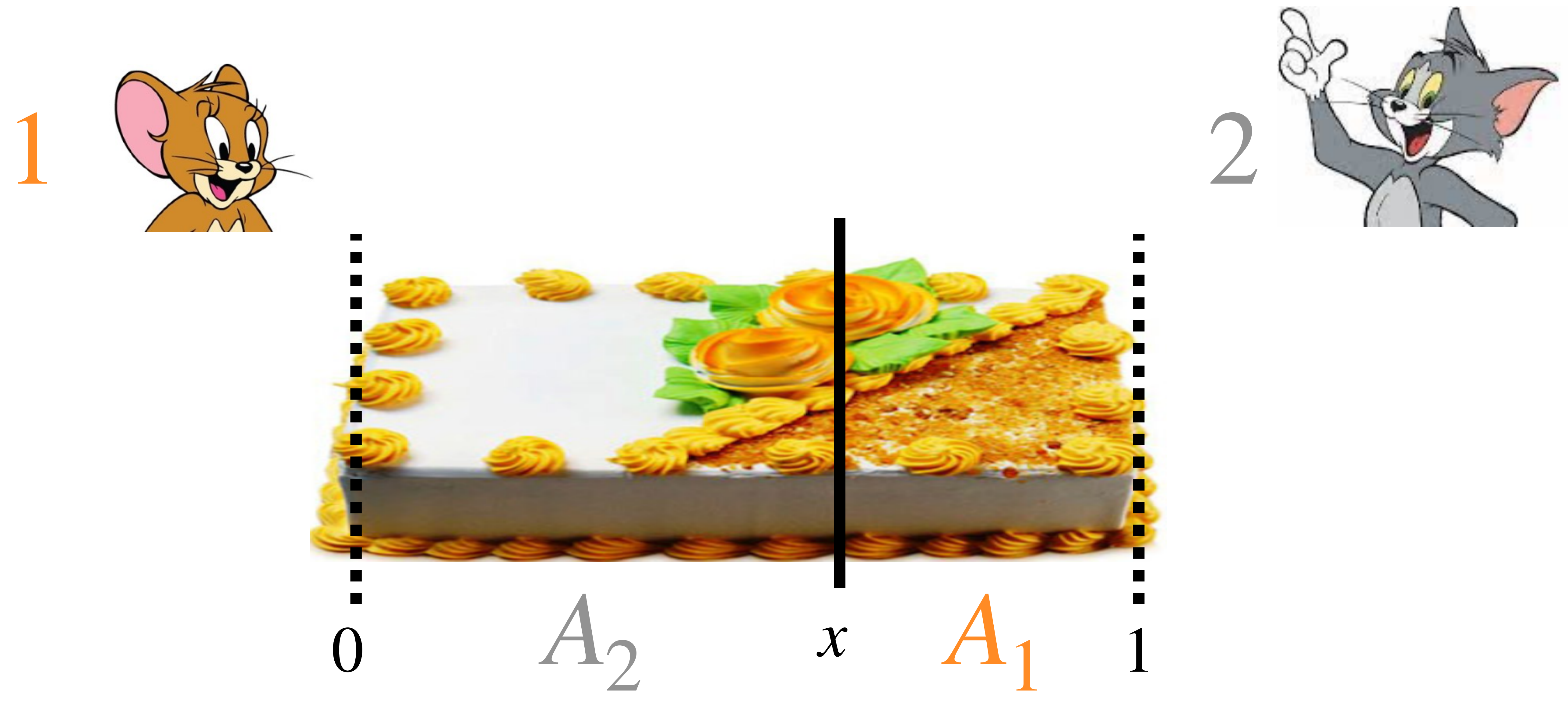
- Does a connected envy-free division always exist?

Envy-free divisions

- Does a connected envy-free division always exist? YES.
- For 2 agents case: Cut and Choose Protocol (we will see it soon)
- For 3 agents: Stromquist moving-knives procedure
    - Brams, Steven J.; Taylor, Alan D. (1996). Fair division: from cake-cutting to dispute resolution. Cambridge University Press.
- For more agents: A non-constructive proof based on Sperner's Lemma
    - Francis Edward Su. Rental harmony: Sperner's lemma in fair division. Amer. Math. Monthly, 106(10):930–942, 1999.)
    - Se also the seminal work: Steinhaus, Hugo (1948). "The problem of fair division". Econometrica. 16 (1): 101–4.
- Other protocols if we allow that each agent can get disconnected pieces
    - Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, Ariel D. Procaccia: Handbook of Computational Social Choice. Cambridge University Press 2016, ISBN 9781107446984)

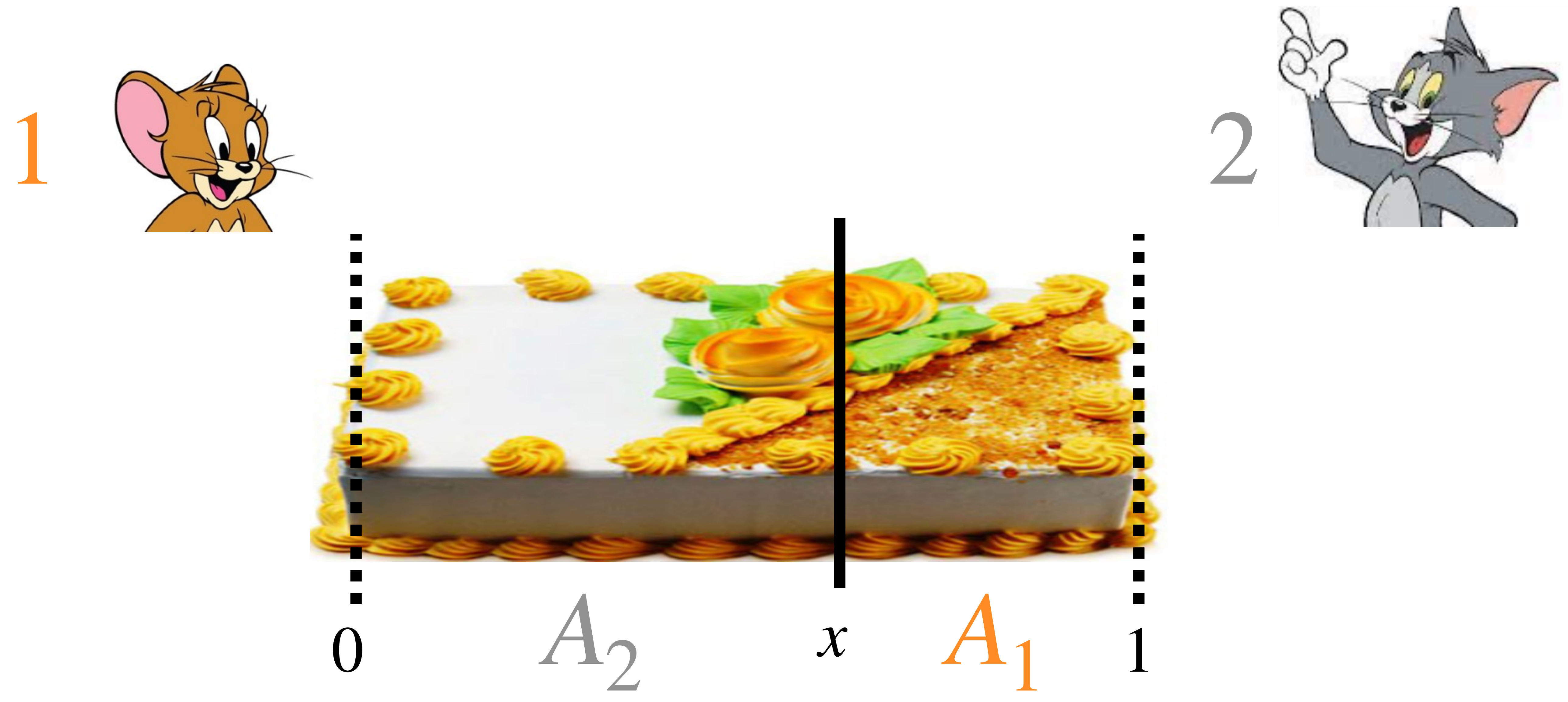
Cut and Choose Protocol (2 agents)

1. Agent 1 cuts: Let $x \in [0,1]$ be the first point such that $v_1([0,x]) = v_1([x,1])$
2.



Cut and Choose Protocol (2 agents)

1. Agent 1 cuts: Let $x \in [0,1]$ be the first point such that $v_1([0,x]) = v_1([x,1])$
2. Agent 2 chooses: Agent 2 takes the best pieces among $A = [0,x]$ and $B = [x,1]$ (i.e., the piece $P \in \{A, B\}$ maximising $v_2(P)$.



Cut and Choose Protocol (2 agents)

Example

- $v_1([a, b]) = b^2 - a^2$, $v_2([a, b]) = b - a$
- Agent 1 cuts in $x = 1/\sqrt{2} \approx 0.7$
- Agent 2 chooses piece/interval $A = \left[0,1/\sqrt{2}\right]$

From divisible to indivisible items

- Assume that the cake is made of some indivisible pieces.
- Does a fair allocation always exist?

From divisible to indivisible items

- Assume that the cake is made of some indivisible pieces.
- Does a fair allocation always exist? NO

Example (in which envy cannot be avoided)

- We have three (indivisible) cookies C_1, C_2, C_3.
- Two agents having the following valuations:

$$1 = v_1(A_1) = v_1(A_2) = v_1(A_3) = v_2(A_1) = v_2(A_2) = v_2(A_3)$$

Fair Allocation of Indivisible Goods

- A set of n agents $N = 1,2,\ldots,n$.
- A set of m indivisible goods/items $M = \{g_1, \ldots, g_m\}$.
- The valuation of each agent i over bundles/subsets of goods is monotone:

  $0 = v_i(\emptyset) \leq v_i(A) \leq v_i(B)$ if $A \subseteq B$.
- The valuations are additive if $v_i(A) = \sum_{g_h \in A} v_{i,h}$ for any bundle $A \subseteq M$, where

  $v_{i,h} := v_i(g_h)$ is the valuation of agent i for good g_h.
- The valuations are identical if $v_i(A) = v_j(A) =: v(A)$ holds for any $A \subseteq M$ and agents i, j.

Envy-free Allocations

- An allocation is a partition $\mathscr{A} = (A_1, \ldots, A_n)$ of all the goods, where A_i is the bundle/subset assigned to agent i.
- An allocation $\mathscr{A}$ is envy-free if $v_i(A_i) \geq v_i(A_j)$ for any agents i, j.

Real-life Applications

- By resorting to the “indivisible setting”, we can model several real-life scenarios. For instance:
  - Several heirs want to divide the inherited property, which contains e.g. a house, a car, a piano and several paintings.
  - Several lecturers want to divide the courses given in their faculty, and each lecturer can teach one or more whole courses.

Envy-free Allocations (cont)

- An allocation is a partition $\mathscr{A} = (A_1, \ldots, A_n)$ of all the goods, where A_i is the bundle/subset assigned to agent i.
- An allocation $\mathscr{A}$ is envy-free if $v_i(A_i) \geq v_i(A_j)$ for any agents i, j.
- We already know that envy-free allocations cannot be guaranteed when dealing with indivisible items.
- Can we hope for some weaker notion of fairness?

Relaxed Notions of Envy-Freeness

An allocation $\mathscr{A} = (A_1, \ldots, A_n)$ is:

- Envy-free allocation-up-to-1-good (EF1) if, for any agents i, j, there exists a good $g_h \in A_j$ such that $v_i(A_i) \geq v_i(A_j \backslash g_h)$.
- Envy-free allocation-up-to-any-good (EFX): if, for any agents i, j, $v_i(A_i) \geq v_i(A_j \backslash g_h)$ holds for any $g_h \in A_j$.
- α-approximate EFX (α-EFX): if, there exists $\alpha \in [0,1]$ such that, for any agents i, j, $v_i(A_i) \geq \alpha \cdot v_i(A_j \backslash g_h)$ holds for any $g_h \in A_j$.

Relaxed Notions of Envy-Freeness

- Other relaxed notions of fairness are (exact and approximate) proportional allocations, maximin fair allocations, etc…
- Refer to Brandt et al. 2016 for a detailed discussion of fair allocation problems.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, Ariel D. Procaccia: Handbook of Computational Social Choice. Cambridge University Press 2016, ISBN 9781107446984

EF1 allocations: existence and computation

- Does an EF1 allocation always exist?

EF1 allocations: existence and computation

- Does an EF1 allocation always exist? YES
- Can an EF1 allocation be efficiently computed?

EF1 allocations: existence and computation

- Does an EF1 allocation always exist? YES
- Can an EF1 allocation be efficiently computed? YES

We will consider the following two algorithms to compute EF1 allocations (and implicitly, to show their existence):

- Round-Robin Algorithm (for additive valuations)
- Envy-Cycle Elimination Algorithm (for monotone valuations)

For further details, see: Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, Amin Saberi: On approximately fair allocations of indivisible goods. EC 2004: 125-131

Round-Robin Algorithm

- Consider an arbitrary order of the agents (we consider w.l.o.g. the increasing order w.r.t. the agents' indexes).
- For k=1,2...,m
  - Agent i=k mod n takes her most valuable good, among all the unallocated good.
- Return the obtained allocation

Each time agent 1 takes a new good, a new round starts
(thus, there are $\lceil m/n \rceil$ rounds)

Round-Robin Algorithm (Example)

							
	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Initial instance

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Round s=1 Iteration k=1

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Round s=1 Iteration k=2

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Round s=1 Iteration k=3

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Round s=2 Iteration k=4

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Round s=2 Iteration k=5

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

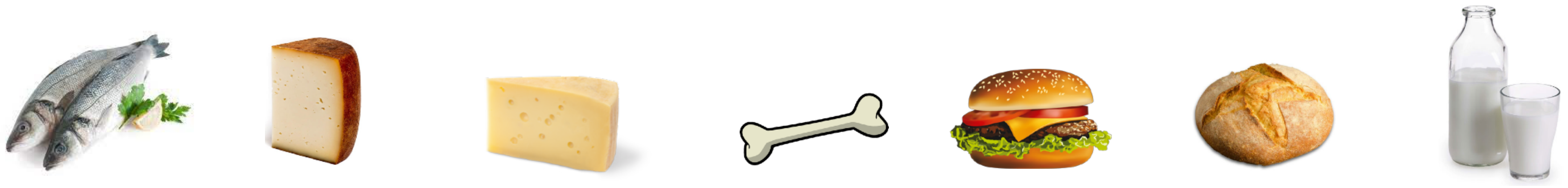
Round s=2 Iteration k=6

Round-Robin Algorithm (Example)

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	13
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

Round s=3 Iteration k=7

Round-Robin Algorithm (Example)



	g_1	g_2	g_3	g_4	g_5	g_6	g_7
	3	10	12	1	5	4	20
	40	9	10	30	8	4	50
	1	0	0	3	1	0	5

$A_1 = \{g_3, g_6, g_7\}, v_1(A_1) = 36$

$A_2 = \{g_1, g_2\}, v_2(A_2) = 49$

$A_3 = \{g_4, g_5\}, v_3(A_3) = 4$

Round-Robin Algorithm

Theorem: If valuations are additive, Round-Robin returns an EF1-allocation

Proof:

- For simplicity, we add dummy items of null value so that n divides m.
- Let $\mathscr{A} = \{A_1, \ldots, A_n\}$ be the allocation returned by Round-Robin.
- Let $g^{i,s}$ denote the s-th good picked by each agent j (i.e., at round s), so that $A_i = \{g^{i,1}, g^{i,2}, \ldots, g^{i,m/n}\}$ for any agent i.

Round-Robin Algorithm

Proof (cont):

For any fixed agent i, we have two cases:

$v_i(g^{i,s}) \geq v_i(g^{j,s})$ for any round s (see next table)

1. For j>i, we have: $v_i(A_i) = \sum_{s=1}^{m/n} v_i(g^{i,s}) \geq \sum_{s=1}^{m/n} v_i(g^{j,s}) = v_i(A_j)$.

2.

Round-Robin Algorithm

A_1	...	A_i	...	A_j	...	A_n
g^{1,1}	....	g^{i,1}	...	g^{j,1}	...	g^{n,1}
g^{1,2}	...	g^{i,2}	...	g^{j,2}	...	g^{n,2}
...	...	...	...	...	...	...
g^{1,s-1}	...	g^{i,s-1}	...	g^{j,s-1}	...	g^{n,s-1}
g^{1,s}	...	g^{i,s}	...	g^{j,s}	...	g^{n,s}
g^{1,s+1}	...	g^{i,s+1}	...	g^{j,s+1}	...	g^{n,s+1}
...	...	...	...	...	...	...
g^{1,m/n}	...	g^{i,m/n}	...	g^{j,m/n}	...	g^{n,m/n}

An arrow from a to b means that $a \geq b$

Round-Robin Algorithm

Proof (cont):

For any fixed agent i, we have two cases:

$v_i(g^{i,s}) \geq v_i(g^{j,s})$ for any round s (see next table)

1. For j>i, we have: $v_i(A_i) = \sum_{s=1}^{m/n} v_i(g^{i,s}) \geq \sum_{s=1}^{m/n} v_i(g^{j,s}) = v_i(A_j)$.

2. For j<i, we have: $v_i(A_i) \geq \sum_{s=1}^{m/n-1} v_i(g^{i,s}) \geq \sum_{s=2}^{m/n} v_i(g^{j,s}) = v_i(A_j) - v_i(g^{j,1})$

$v_i(g^{i,s}) \geq v_i(g^{j,s+1})$ for any round s (see next table)

Round-Robin Algorithm

A_1	...	A_j	...	A_i	...	A_n
g^{1,1}	....	g^{j,1}	...	g^{i,1}	...	g^{n,1}
g^{1,2}	...	g^{j,2}	...	g^{i,2}	...	g^{n,2}
...	...	...	...	...	...	...
g^{1,s-1}	...	g^{j,s-1}	...	g^{i,s-1}	...	g^{n,s-1}
g^{1,s}	...	g^{j,s}	...	g^{i,s}	...	g^{n,s}
g^{1,s+1}	...	g^{j,s+1}	...	g^{i,s+1}	...	g^{n,s+1}
...	...	...	...	...	...	...
g^{1,m/n}	...	g^{j,m/n}	...	g^{i,m/n}	...	g^{n,m/n}

An arrow from a to b means that $a \geq b$

Round-Robin Algorithm

Proof (cont):

For any fixed agent i, we have two cases:

1. For $j>i$, we have: $v_i(A_i) = \sum_{s=1}^{m/n} v_i(g^{i,s}) \geq \sum_{s=1}^{m/n} v_i(g^{j,s}) = v_i(A_j)$.

2. For $j<i$, we have: $v_i(A_i) \geq \sum_{s=1}^{m/n-1} v_i(g^{i,s}) \geq \sum_{s=2}^{m/n} v_i(g^{j,s}) = v_i(A_j) - v_i(g^{j,1})$

We conclude that $\mathscr{A}$ is an EF1 allocation. Q.E.D.

Envy-Cycle Elimination Algorithm

Envy-graph and Envy-Cycle-Elimination Subroutine

Given a partial allocation $\mathscr{A} = (A_1, \ldots, A_n)$ (i.e., there are some unallocated goods), the envy-graph associated to $\mathscr{A}$ is a graph $G = (V, E)$ defined as follows:

- V is the set of agents;
- we put a directed edge (u, z) in E iff agent u is envious of z under $\mathscr{A}$ (that is, $v_u(A_u) < v_u(A_z)$).

Envy-Cycle Elimination Algorithm

Envy-graph (cont)

- If there is a cycle $C = (u_1, u_2), (u_2, u_2)\ldots, (u_{t-1}, u_1)$ in G, we assign the bundle of u_2 to u_1, that of u_3 to u_2 ... that of u_1 to u_{t-1}.

Envy-Cycle Elimination Algorithm

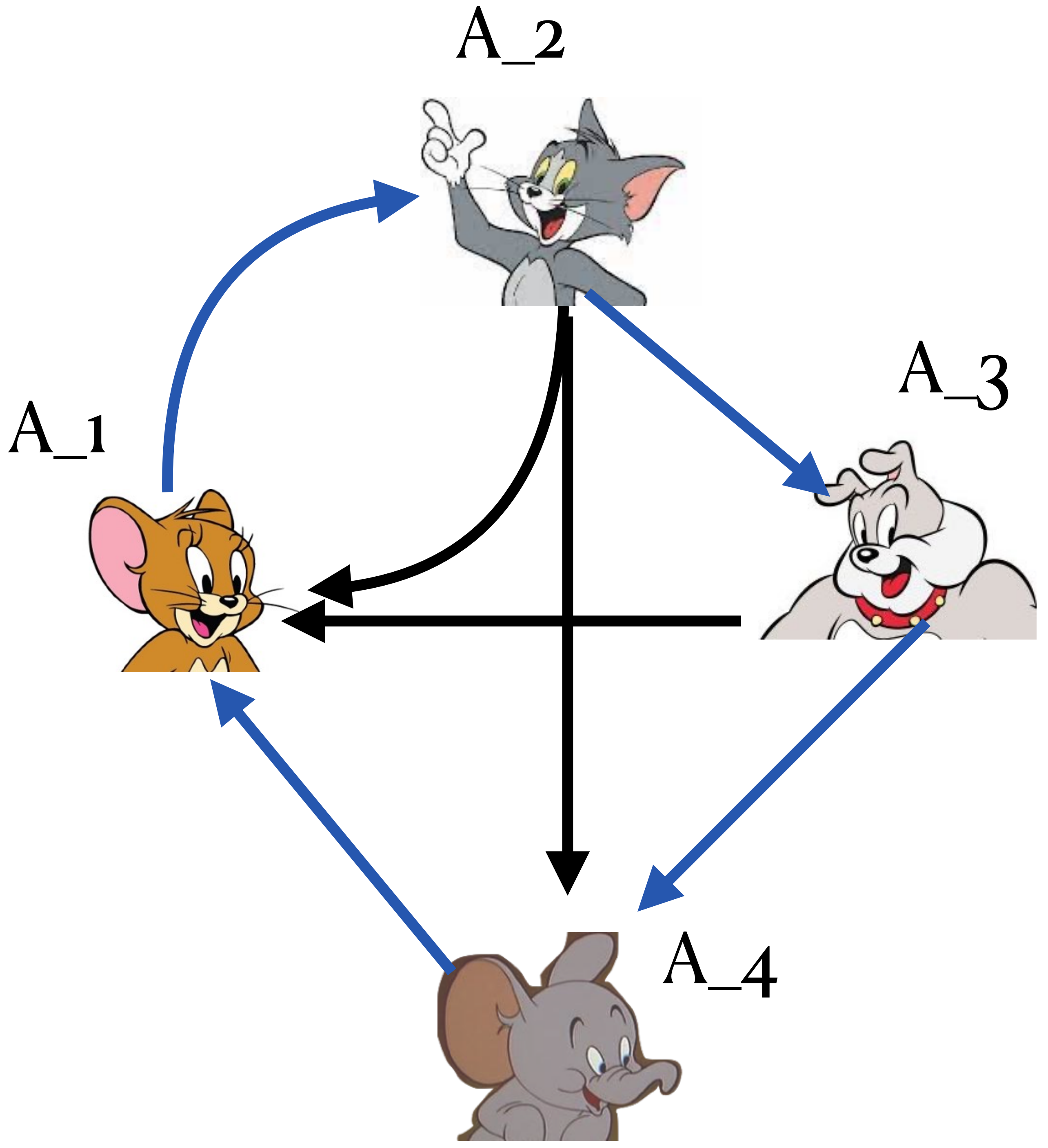
Envy-graph (cont)

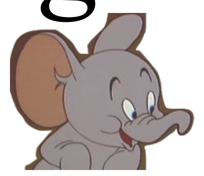
- If there is a cycle $C = (u_1, u_2), (u_2, u_2)\ldots, (u_{t-1}, u_1)$ in G, we assign the bundle of u_2 to u_1, that of u_3 to u_2 ... that of u_1 to u_{t-1}. After this reallocation, the following facts hold:
    - the envy-graph $G' = (V, E')$ associated with the new partial allocation $\mathscr{A}'$ does not contain cycle C;
    - new edges are not included (either we delete edges or "adjust" existing edges).
- We conclude that $|E'| < |E|$. By iterating such process (Q: how many times?) we necessarily reach a partial allocation $\mathscr{A}^*$ whose envy-graph G^* is acyclic.
- The above procedure is called Envy-Cycle-Elimination Subroutine.

Envy-Cycle Elimination Algorithm

Example (Envy-Cycle-Elimination Subroutine)

	v(A_1)	v(A_2)	v(A_3)	v(A_4)
	20	24	5	10
	30	0	35	10
	70	0	40	60
	20	3	10	11

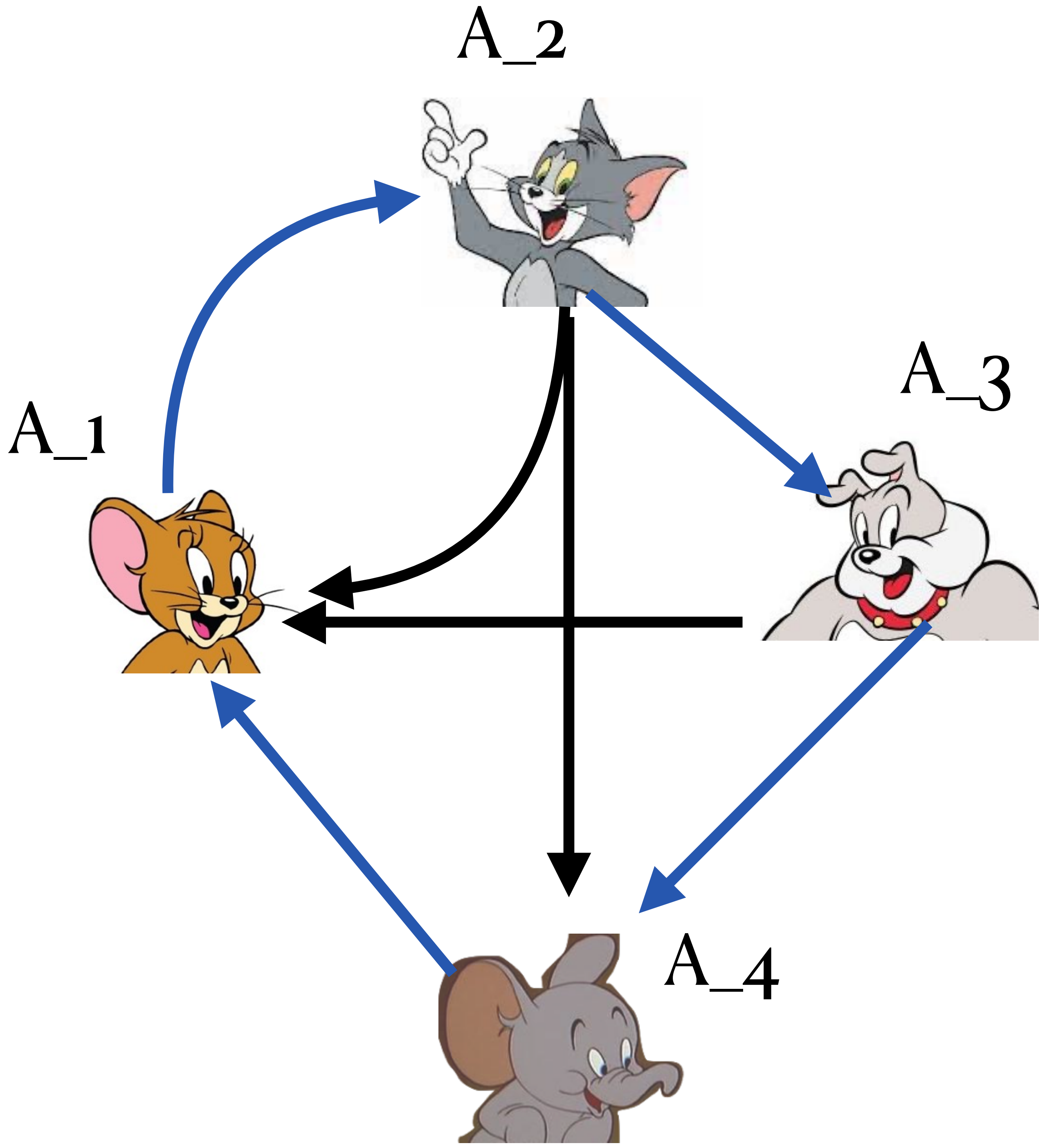


We initially consider the allocation assigning:
A_1 to , A_2 to , A_3 to , A_4 to 

Envy-Cycle Elimination Algorithm

Example (Envy-Cycle-Elimination Subroutine)

	v(A_1)	v(A_2)	v(A_3)	v(A_4)
	20	24	5	10
	30	0	35	10
	70	0	40	60
	20	3	10	11

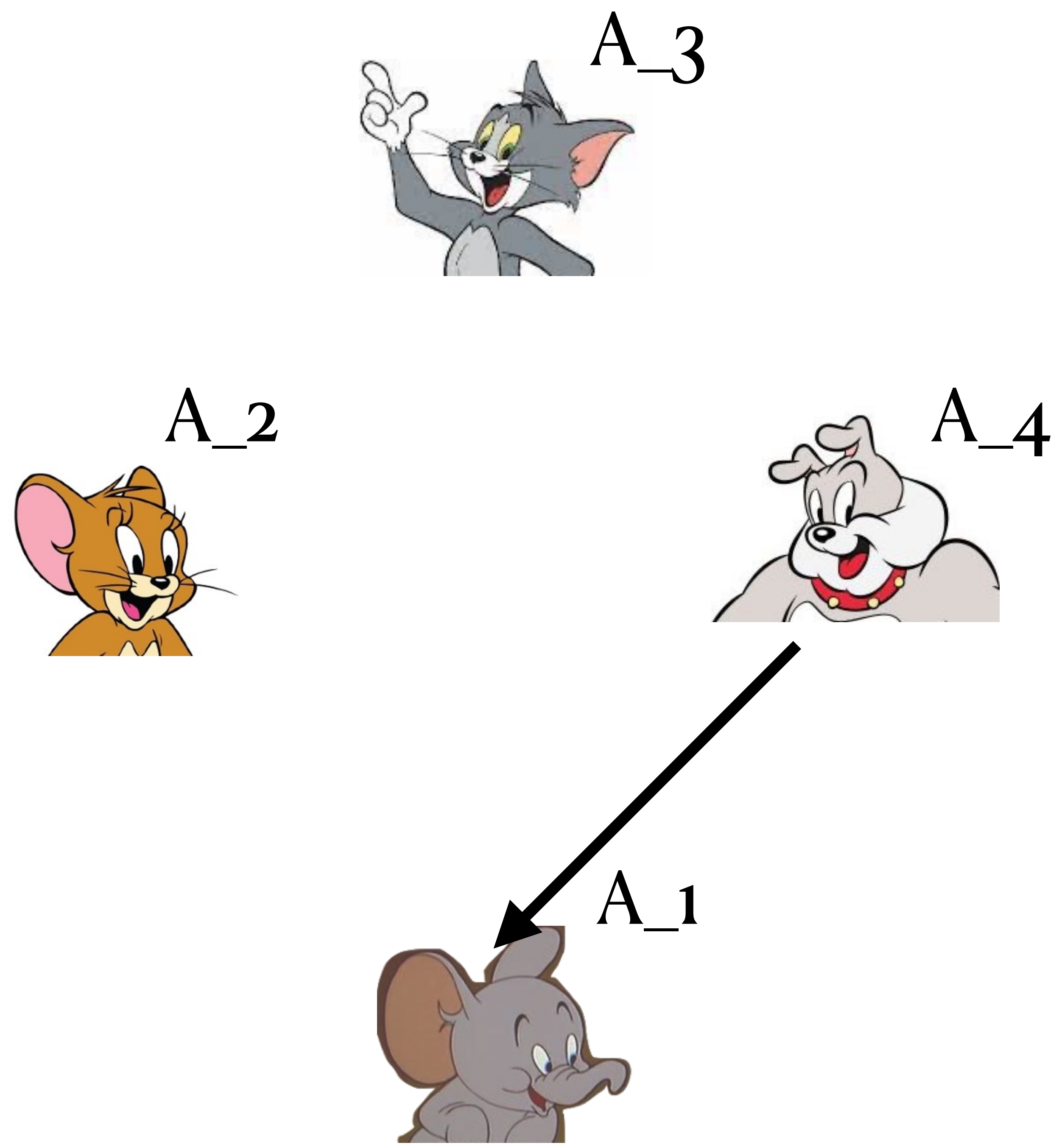


There is (at least) one cycle (the blue one) in the envy-graph

Envy-Cycle Elimination Algorithm

Example (Envy-Cycle-Elimination Subroutine)

	v(A_1)	v(A_2)	v(A_3)	v(A_4)
	20	24	5	10
	30	0	35	10
	70	0	40	60
	20	3	10	11

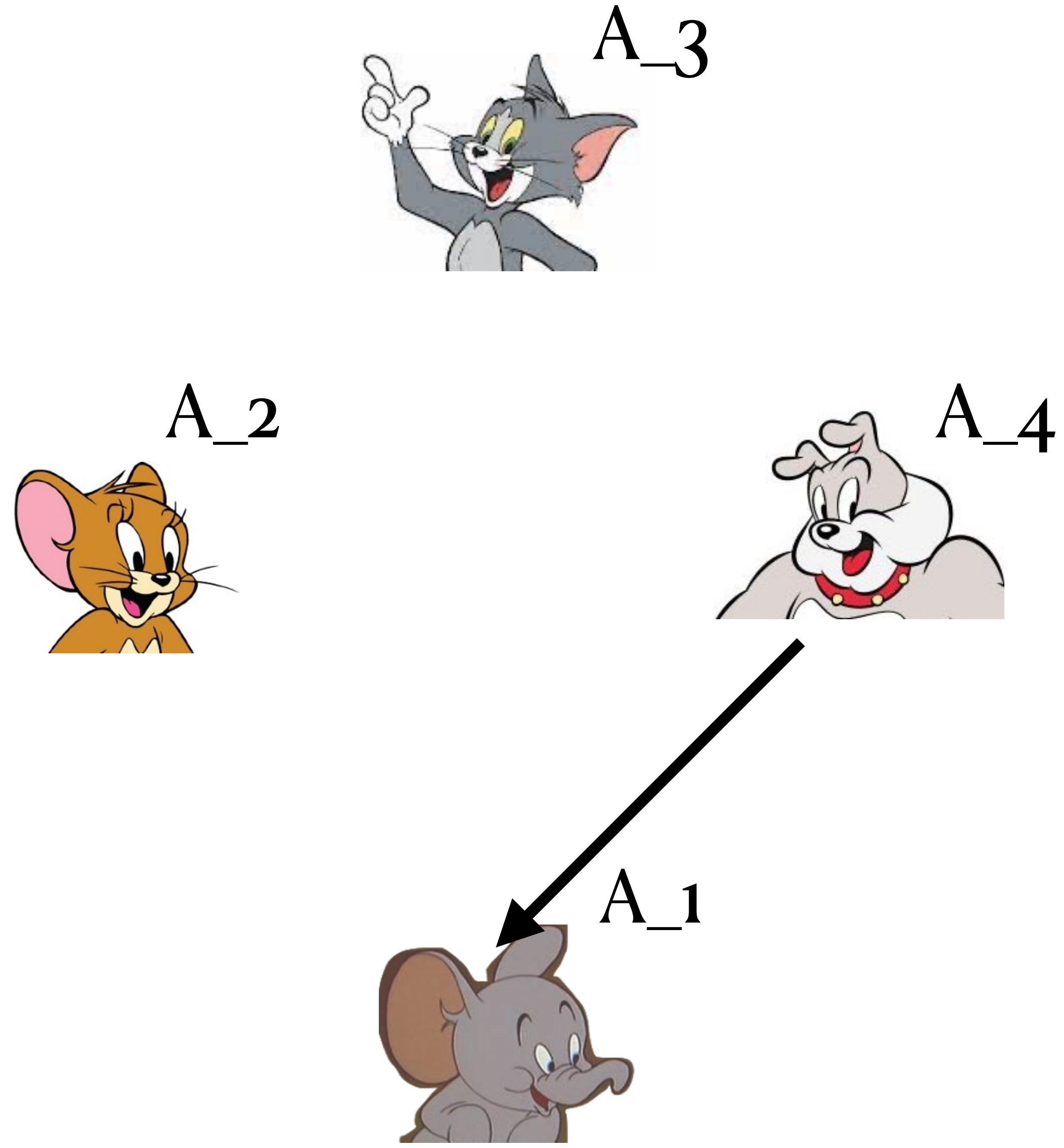


We construct a new envy-graph
(by reallocating the bundles in the cycle)

Envy-Cycle Elimination Algorithm

Example (Envy-Cycle-Elimination Subroutine)

	v(A_1)	v(A_2)	v(A_3)	v(A_4)
	20	24	5	10
	30	0	35	10
	70	0	40	60
	20	3	10	11



The new graph is acyclic :)

(otherwise, we would have had to repeat the procedure)

Envy-Cycle Elimination Algorithm

We are ready to present the Envy-Cycle Elimination Algorithm:

1. Start with an empty partial allocation $\mathscr{A} = (A_1, \ldots, A_n)$ (i.e., $A_i = \emptyset$)
2. Consider an arbitrary order of the goods (e.g., $g_1, g_2, \ldots, g_m$)
3. For $k = 1, 2, \ldots, m$ {
   A. $\mathscr{A} \leftarrow$ Partial all. returned by Envy-cycle Elimination Subroutine applied to $\mathscr{A}$
   B. let i be an agent that is not envied by anyone (Q: why does such agent always exist? See the above example for a hint)
   C. assign g_k to i, and let $\mathscr{A}$ be the resulting partial allocation}
4. Return $\mathscr{A}$

Envy-Cycle Elimination Algorithm

Theorem: The Envy-Cycle Elimination Algorithm always returns an EF1 allocation (even, for general monotone valuation).

Proof: We will inductively show that the partial allocation $\mathscr{A}_k$ obtained at the end of each iteration k=0,1…,m (k=0 refers to the initialisation) is EF1.

- **k=0:** Trivial (the empty allocation is trivially EF1).

Envy-Cycle Elimination Algorithm

Proof (cont):

- **k=1:**
    - We know that $\mathscr{A}_0$ is EF1 (from the previous step).
    - By applying the envy-cycle elimination subroutine to $\mathscr{A}_0$ we get again an EF1 allocation (indeed, "new envy is not created" and "old envy could be removed").
    - Item g_1 is assigned to an agent that is not envied (in the strong sense) by anyone. Thus, after receiving g_1, other agents, in the worst-case, can be "envious of that agent up to item g_1".
    - Thus, the new partial allocation $\mathscr{A}_1$ is EF1.

Envy-Cycle Elimination Algorithm

Proof (cont):

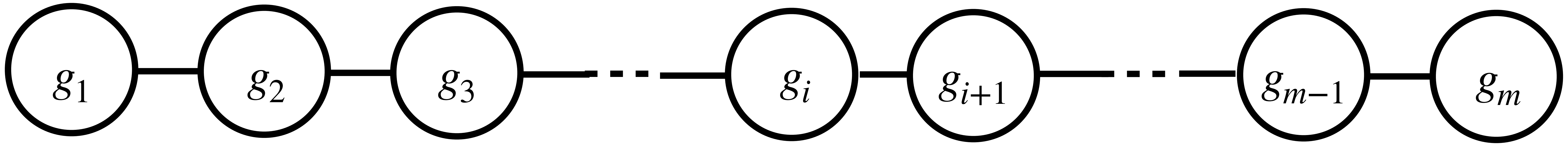
- **Any k:**
  - Analogously, by using that $\mathscr{A}_1$ is EF1, we can show that the partial allocation $\mathscr{A}_2$ obtained from $\mathscr{A}_1$ is EF1, and so on…
  - More formally, we can show the claim by induction: we assume that $\mathscr{A}_{k-1}$ is EF1 by the inductive hypothesis, and by using the same proof arguments of case k=1 we show that partial allocation $\mathscr{A}_k$ is EF1.

We conclude that $\mathscr{A}_k$ is EF1 for any k, and even more so the final allocation $\mathscr{A}$ (i.e., $\mathscr{A}_m$) is EF1.

Q.E.D.

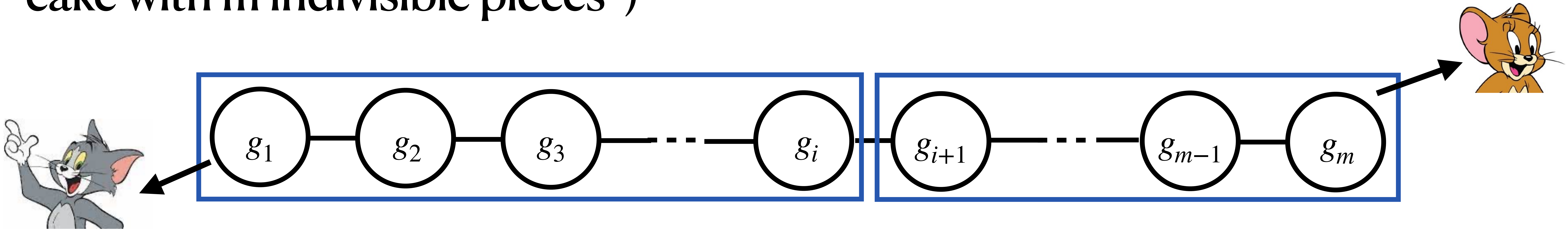
Exercise: Connected EF1 with 2 agents

- We have 2 agents, and assume that the goods are organised as a "path" (that is, a "cake with m indivisible pieces")



Exercise: Connected EF1 with 2 agents

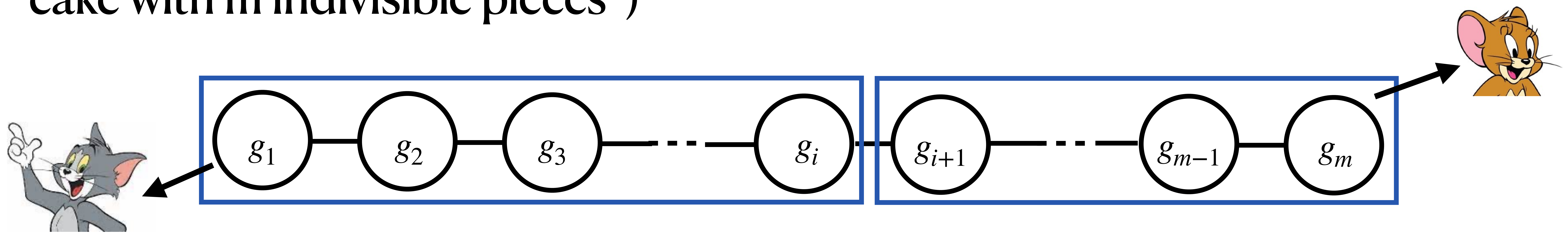
- We have 2 agents, and assume that the goods are organised as a "path" (that is, a "cake with m indivisible pieces")



- You are asked to partition the goods into 2 bundles having the structure of "sub-paths" (that is, each bundle must be of type $\{g_1, \ldots, g_i\}$ or $\{g_{i+1}, \ldots, g_m\}$ for some i), and assign each bundle to a distinct agent.

Exercise: Connected EF1 with 2 agents

- We have 2 agents, and assume that the goods are organised as a “path” (that is, a “cake with m indivisible pieces”)



- You are asked to partition the goods into 2 bundles having the structure of “sub-paths” (that is, each bundle must be of type $\{g_1, \ldots, g_i\}$ or $\{g_{i+1}, \ldots, g_m\}$ for some i), and assign each bundle to a distinct agent.
- The resulting allocation must be EF1. (Hint: Provide a discrete version of the Cut and Choose protocol)

EFX allocations: existence and computation

- From now on, we implicitly focus on additive valuations.
- Does an EFX allocation always exist?
    - We don't know (the conjecture is YES)
    - This is one of the most important open problems in fair allocation.

For further details, see:

- Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, Junxing Wang:The Unreasonable Fairness of Maximum Nash Welfare. ACM Trans. Economics and Comput. 7(3): 12:1-12:32 (2019)
- Benjamin Plaut, Tim Roughgarden: Almost Envy-Freeness with General Valuations. SODA 2018: 2584-2603

EFX allocations: existence and computation

- There are some interesting cases in which they always exist?

EFX allocations: existence and computation

- There are some interesting cases for which EF1 alloc. always exist? YES.
    - agents having the same ranking on the goods, despite possibly different values: by using a variant of cycle-elimination (we will see it)
    - 2 agents: discrete variant of cut and choose (see: Benjamin Plaut, Tim Roughgarden: Almost Envy-Freeness with General Valuations. SODA 2018: 2584-2603)
    - 3 agents (see: Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn: EFX Exists for Three Agents. EC 2020: 1-19)
    - and so on…

Revisited Envy-cycle Elimination Algorithm

1. Start with an empty partial allocation $\mathscr{A} = (A_1, \ldots, A_n)$ (i.e., $A_i = \emptyset$)
2. While there exists an unallocated good {
   A. $\mathscr{A} \leftarrow$ Envy-cycle Elimination applied to $\mathscr{A}$
   B. let i be an agent that is not envied by anyone

Novelty

   C. let g the most valuable good for i among all the unallocated ones, and assign g to i
   D. set $\mathscr{A}$ equal to the resulting allocation }
3. Return $\mathscr{A}$

Revisited Envy-cycle Elimination Algorithm

Theorem: If the agents have the same ranking on the goods, the Revisited Envy-Cycle Elimination Algorithm always returns an EFX allocation.

Proof (sketch): Reconsider the proof for EF1.

- Some agent i becomes "envious because of 1 good" only if she becomes "envious because of the last good g" added to some bundle A.
- Such good g is necessarily the least valuable among all the goods of bundle A.
- Thus, agent i is "envious because of the least valuable good of A", that is, she is "not-envious-up-to-any-good".

We conclude that, at each iteration, the reached partial allocation is EFX. Q.E.D.

Revisited Envy-cycle Elimination Algorithm

What would we have achieved by applying the revisited Envy-cycle algorithm to general instances?

Theorem: The Revisited Envy-Cycle Elimination Algorithm always returns a 1/2-EFX allocation.

The proof follows the same high-level structure of the previous one, but it is more sophisticated.