COST IC1205 Summer School on Fair Division
Grenoble, France, 13-17 July, 2015

Mechanism Design and Fair Allocation Problems

Gianluigi Greco

Social Choice Theory

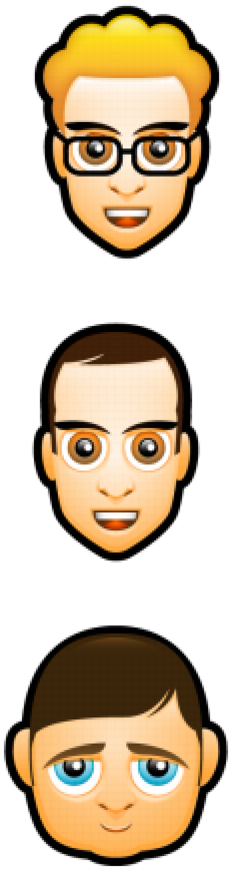
Rule for breaking ties: A > B > C

Alternatives

- { A, B, C }

Social Choice Function:

- Compute the alternative that is top-ranked by the majority



A > C > B

B > A > C

C > B > A

→ A

Social Choice Theory → Mechanism Design

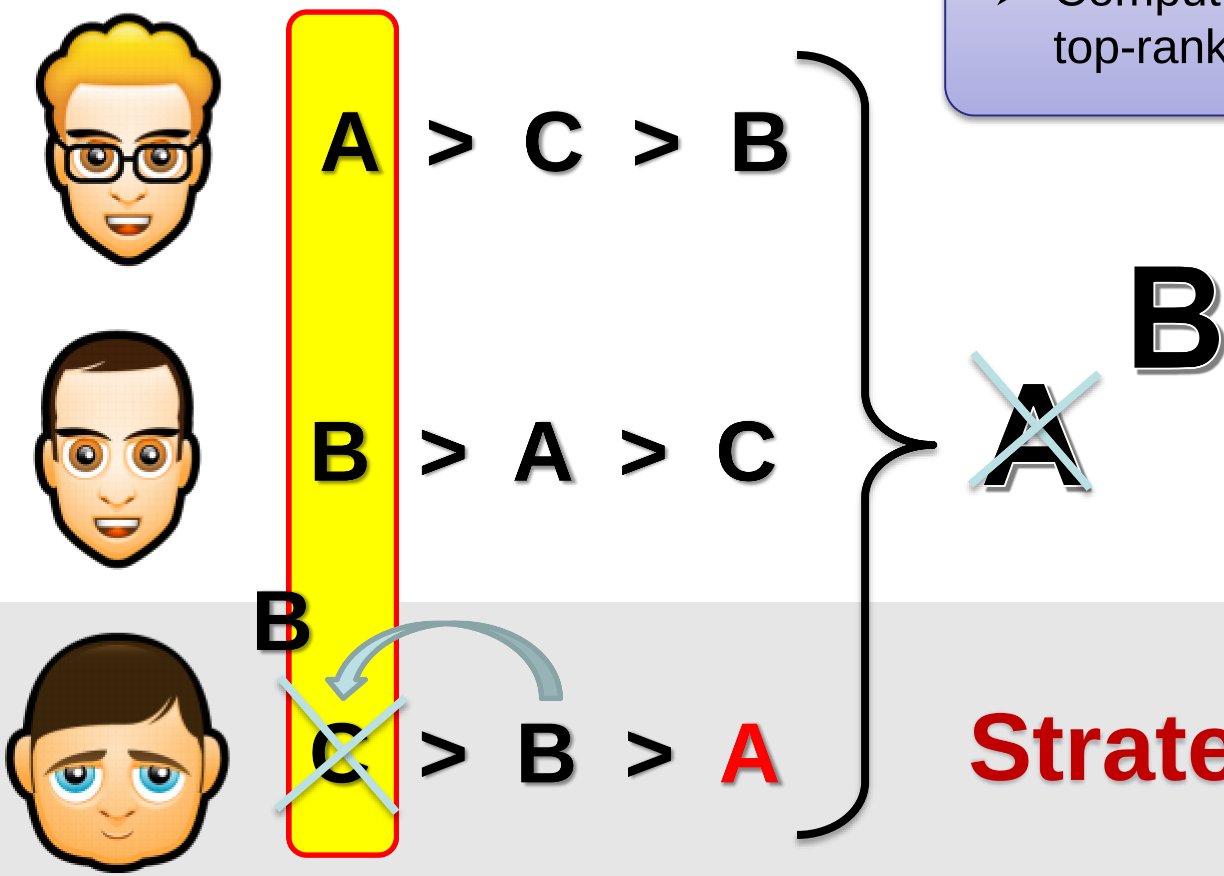
Rule for breaking ties: A > B > C

Alternatives

- { A, B, C }

Social Choice Function*:*

- Compute the alternative that is top-ranked by the majority



Strategic issues!

Mechanism Design

- Social Choice Theory is *non-strategic*
- In practice, agents **declare** their preferences
  - They are self interested
  - They might not reveal their true preferences
- We want to find optimal outcomes w.r.t. true preferences
- Optimizing w.r.t. the declared preferences might not achieve the goal

How to build a mechanism where agents find convenient to report their true preferences?

Outline

- Game Theory
- Mechanism Design
- Mechanisms with Verification
- Mechanisms and Allocation Problems
- Complexity Analysis

Basic Concepts (1/2)

- Each agent i is associated with a **type** $\theta_i \in \Theta_i$

private knowledge, preferences,…

C > B > A

- Each agent i has a **strategy** $s_i(\theta_i) \in \Sigma_i$

the action manifested

C > B > A

Basic Concepts (2/2)

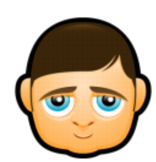
- Consider the vector of the **joint strategies** $s = (s_1, \ldots, s_I)$

(A, B, C) ➡ A

- Each agent i gets some **utility** $u_i(s_1, \ldots, s_I, \theta_i)$

$$u_i(s_i, s_{-i}, \theta_i)$$

(A, B, C) ➡ A ➡ 1

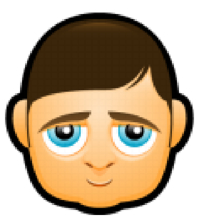


C > B > A

3 2 1

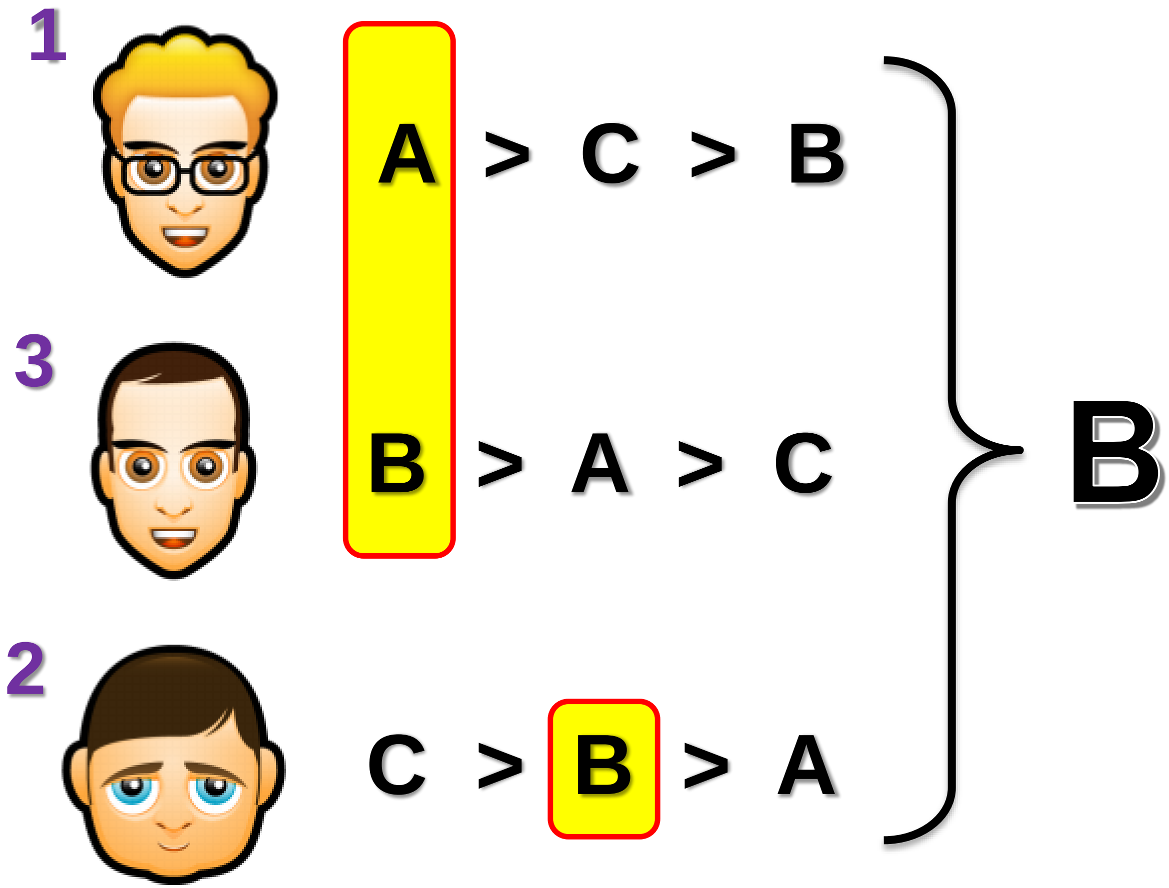
Game Theory (by Example)

- Consider the utility function of agent  (C > B > A ; 3 2 1)
- Let us reason on the case where
  -  selects **A**
  -  selects **B**

 will select **B**

						
A	B	A	➡	A	➡	1
A	B	B	➡	B	➡	2
A	B	C	➡	A	➡	1

Game Theory (by Example)



No agents can benefit by deviating!

Solution Concepts

- A **Nash equilibrium** is a strategy profile $s = (s_1, \ldots, s_I)$

  such that, for every agent i and for every $s_i' \neq s_i$,

$$u_i(s_i, s_{-i}, \theta_i) \geq u_i(s_i', s_{-i}, \theta_i)$$

The strategies of the other agents are fixed…

Solution Concepts

- A **Nash equilbrium** is a strategy profile $s = (s_1, \ldots, s_I)$ such that, for every agent i and for every $s_i' \neq s_i$,

$$u_i(s_i, s_{-i}, \theta_i) \geq u_i(s_i', s_{-i}, \theta_i)$$

Bob	John goes *out*	John stays at *home*
out	2	0
home	0	1

John	Bob goes *out*	Bob stays at *home*
out	1	1
home	0	0

A Closer Look

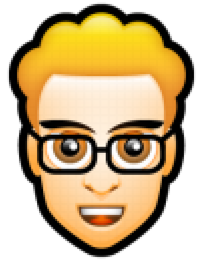
- To play a Nash equilibrium,
  - every agent must have perfect information
  - rationality is common knowledge
  - all agents must select the same Nash equilibrium

Bob	John goes *out*	John stays at *home*
out	2	0
home	0	1

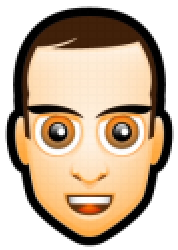
Dominant strategy →

John	Bob goes *out*	Bob stays at *home*
out	1	1
home	0	0

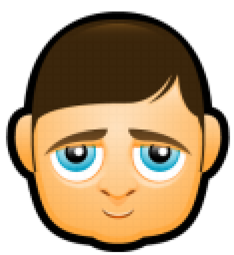
Dominant Strategies (by Example)



A > C > B



B > A > C



C > B > A

For , A is a dominant strategy. Why?

Solution Concepts

- A **Nash equilbrium** is a strategy profile $s = (s_1, \ldots, s_I)$ such that, for every agent i and for every $s_i' \neq s_i$,

$$u_i(s_i, s_{-i}, \theta_i) \geq u_i(s_i', s_{-i}, \theta_i)$$

- A strategy s_i is **dominant** for agent i, if for every $s_i' \neq s_i$ and for every s_{-i},

$$u_i(s_i, s_{-i}, \theta_i) \geq u_i(s_i', s_{-i}, \theta_i)$$

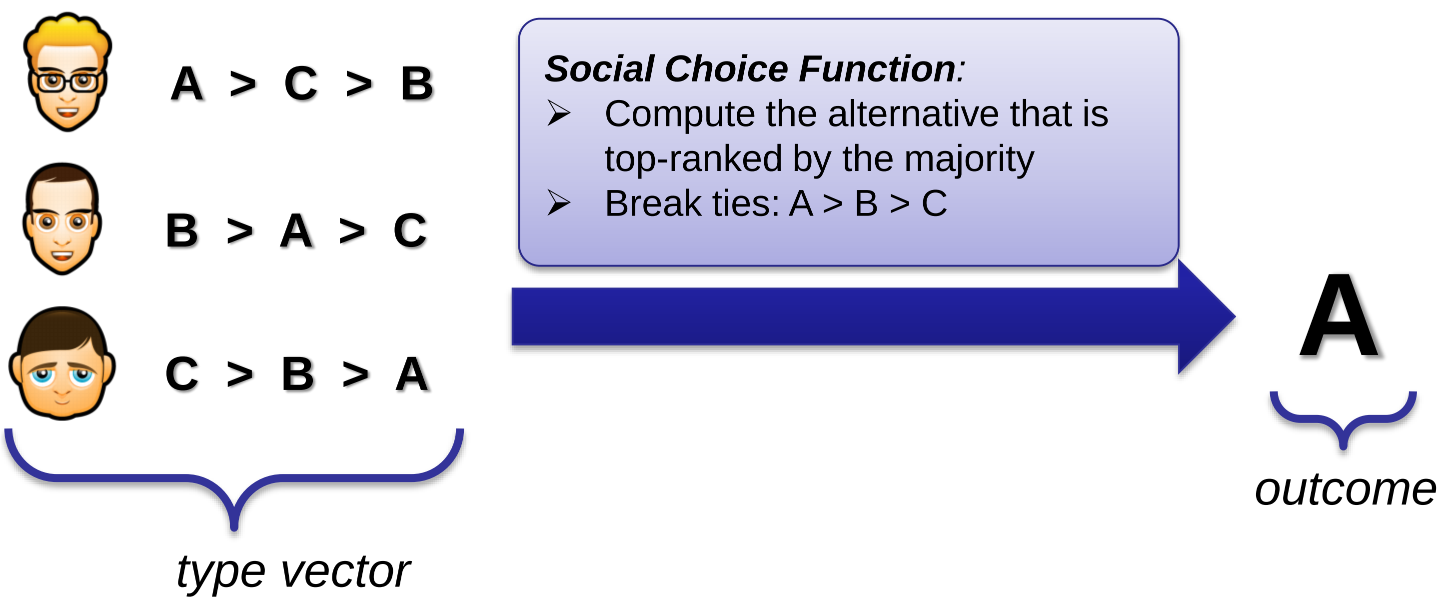
Independently on the other agents…

Outline

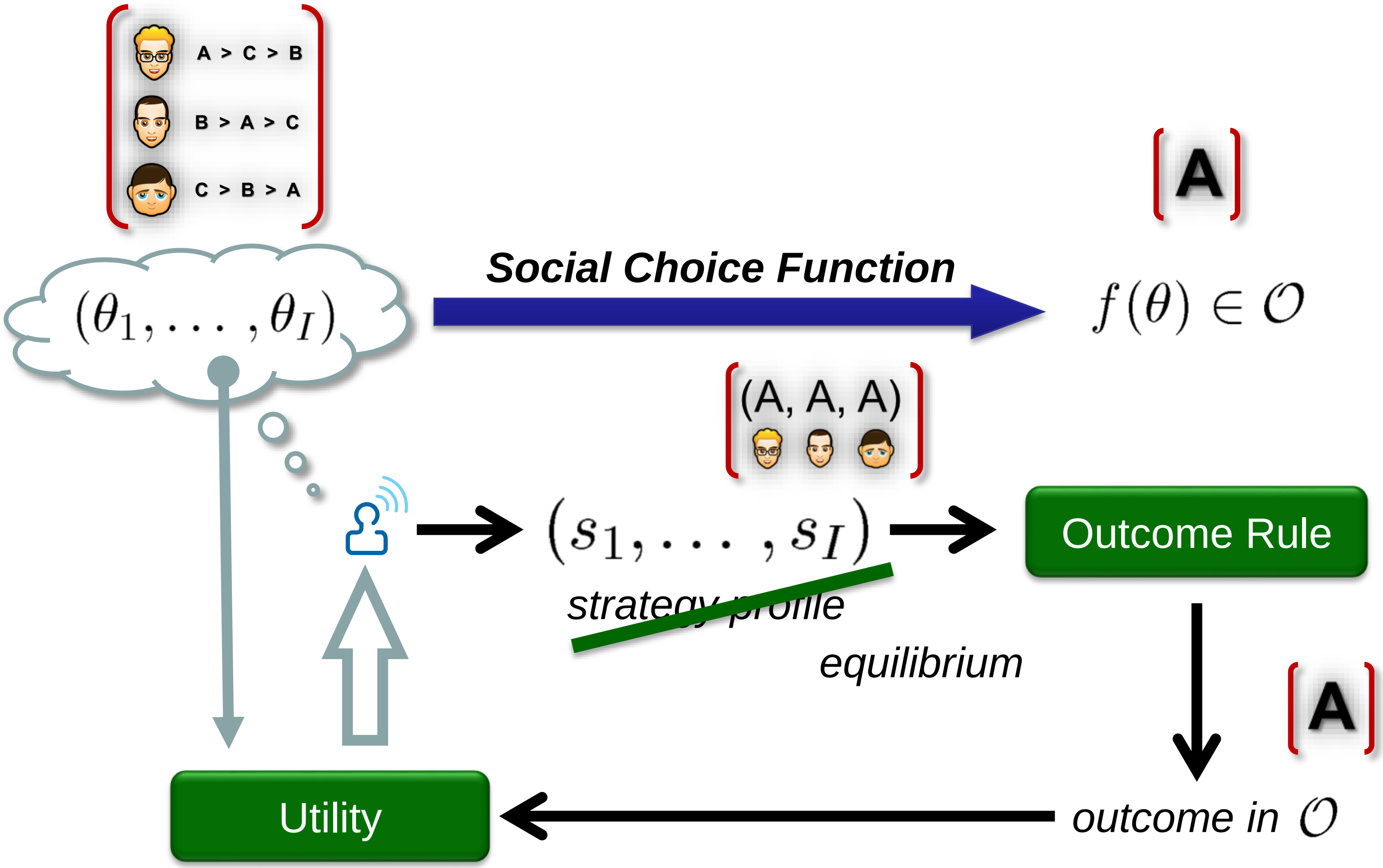
- Game Theory
- Mechanism Design
- Mechanisms with Verification
- Mechanisms and Allocation Problems
- Complexity Analysis

Social Choice Functions

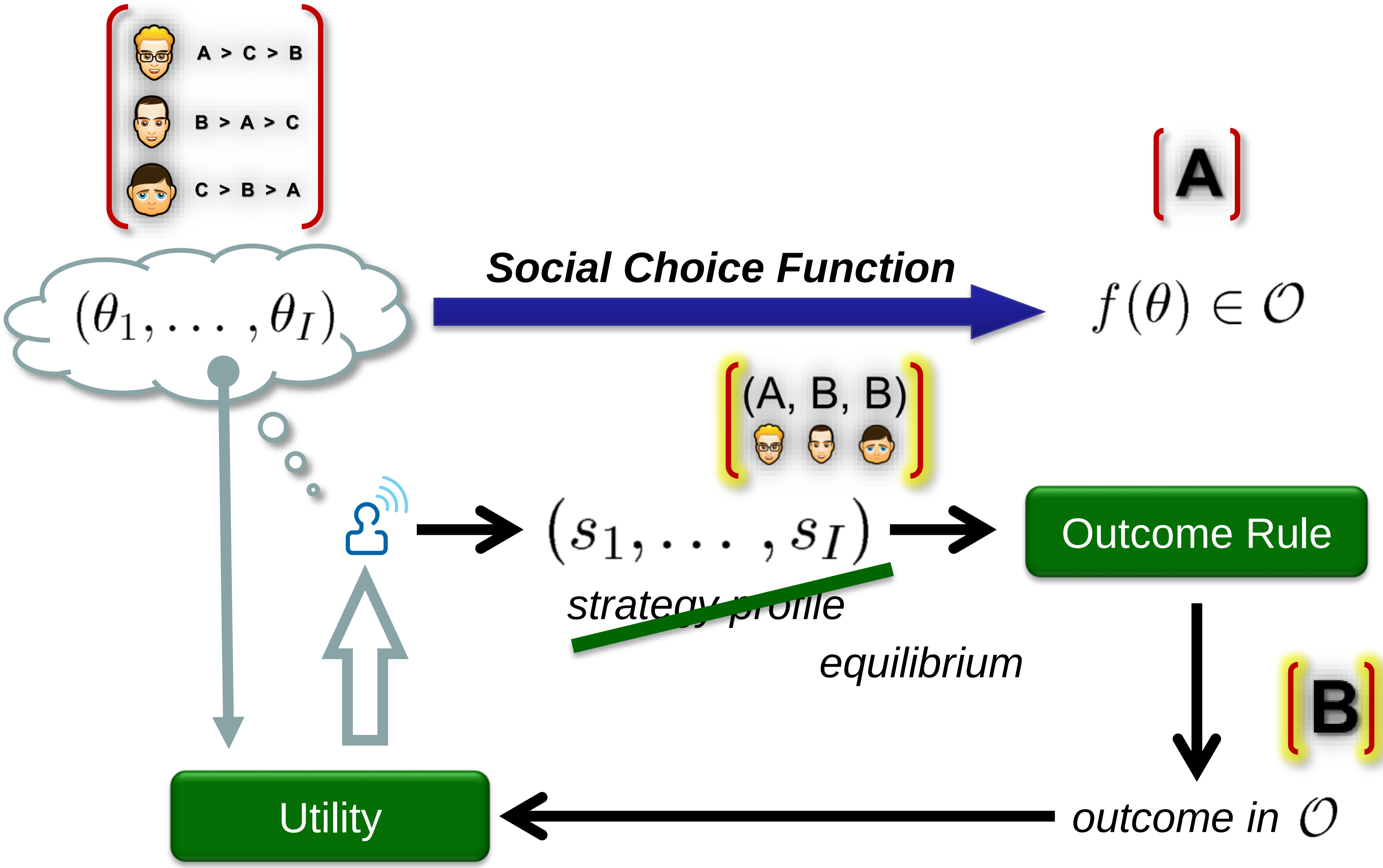
- A **social choice function** $f : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathcal{O}$
  - given a type vector $\theta = (\theta_1, \ldots, \theta_I)$
  - selects an outcome $f(\theta) \in \mathcal{O}$



Mechanism Design



Mechanism Design



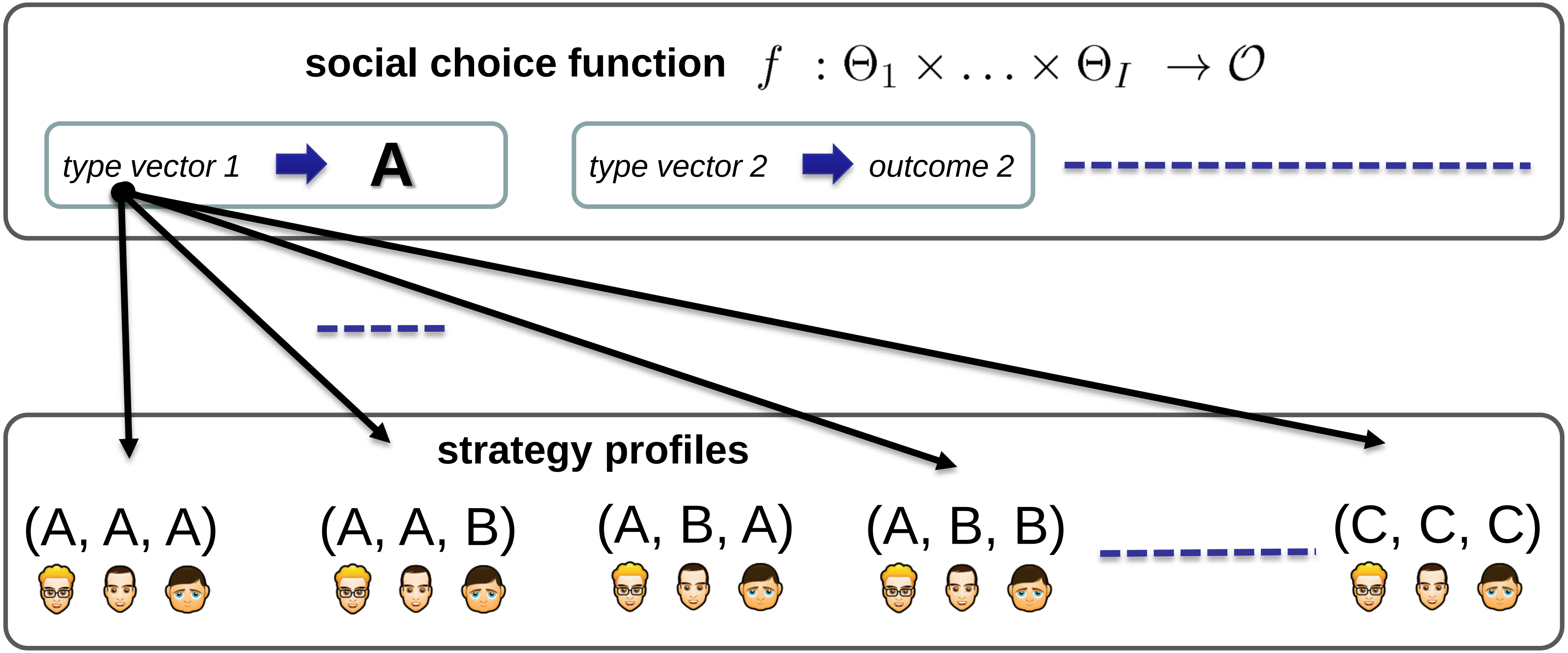
Mechanism and Implementation

social choice function $f : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathcal{O}$

type vector 1 → *outcome 1*

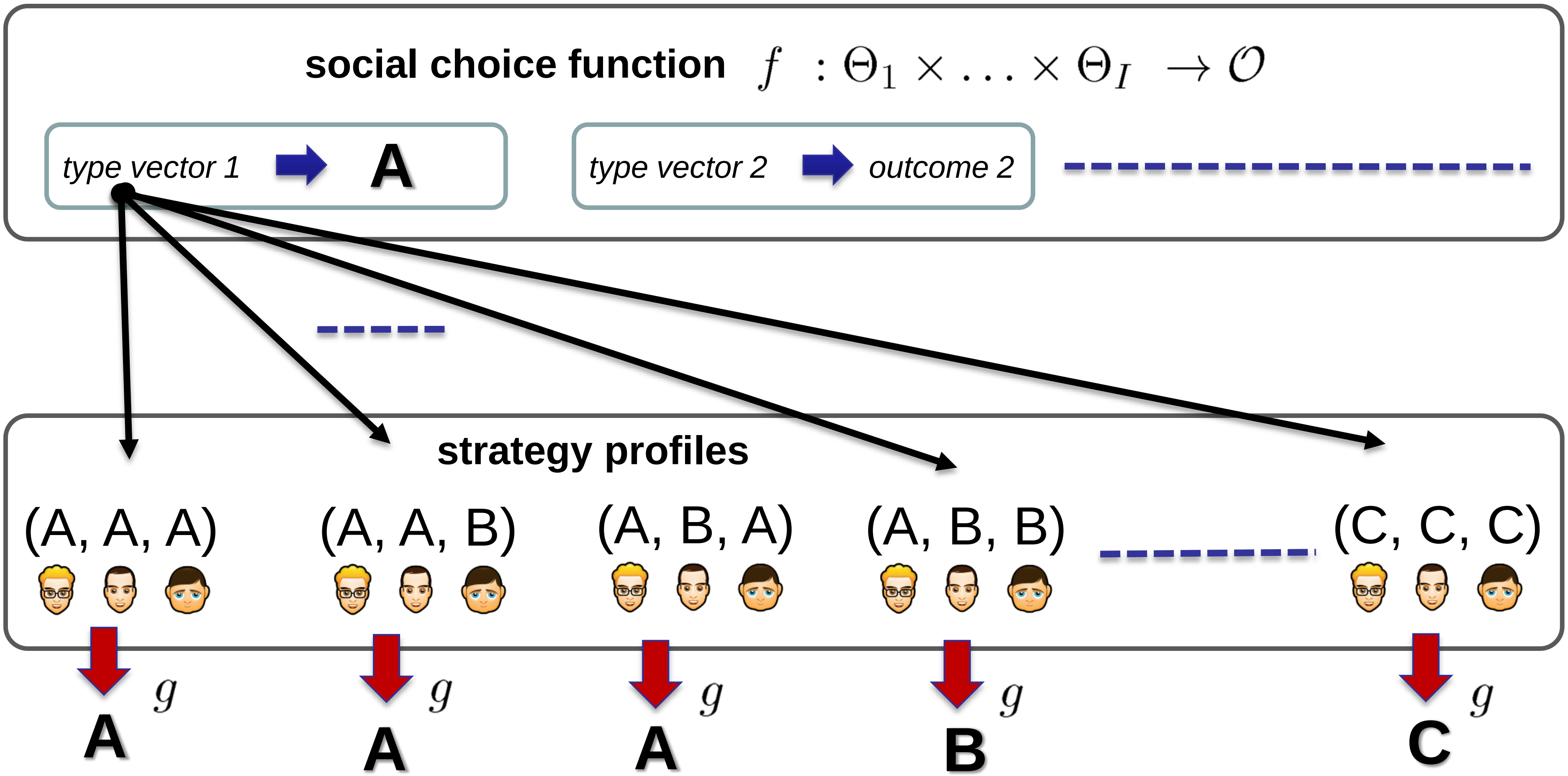
type vector 2 → *outcome 2*

Mechanism and Implementation



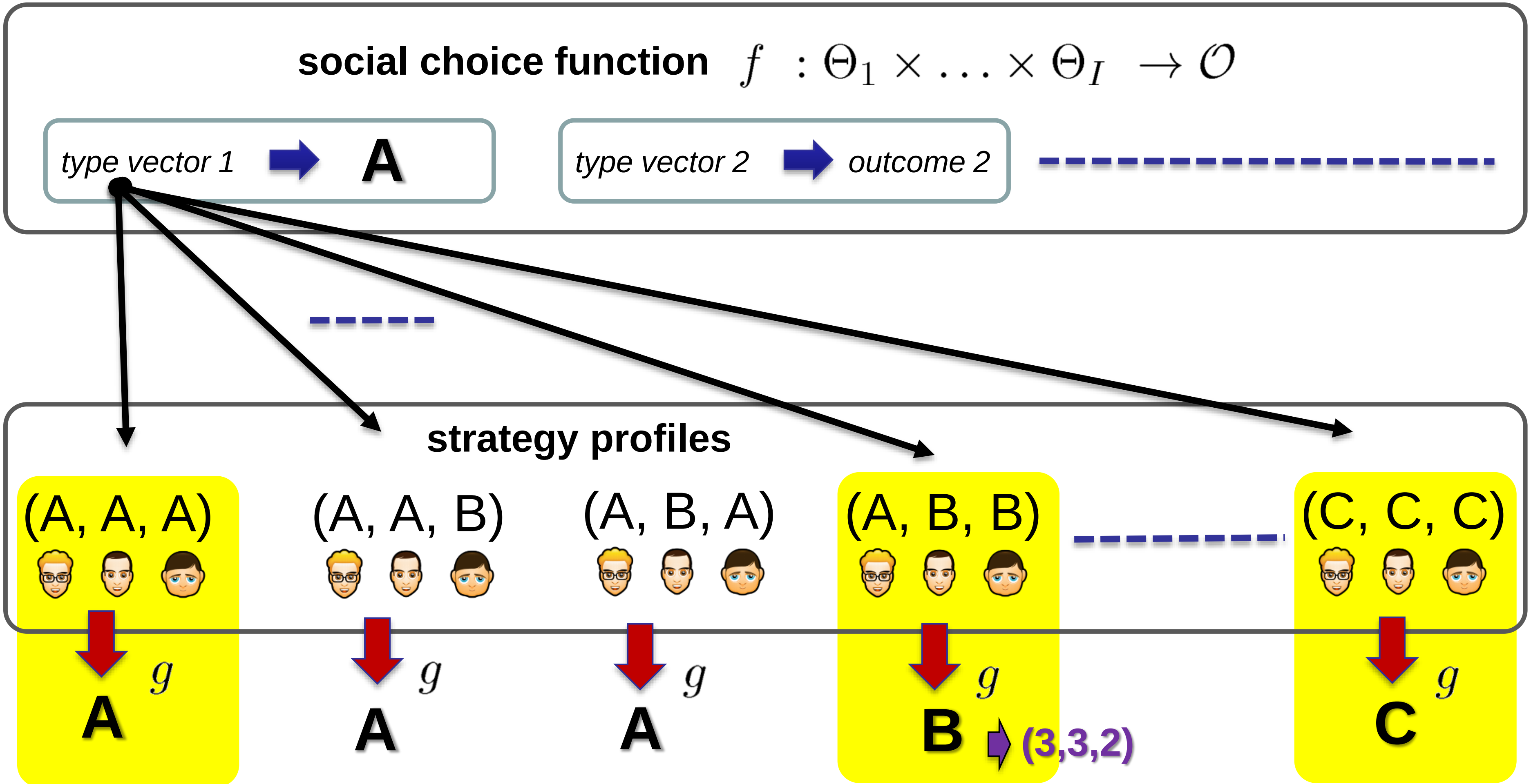
- *For a given type vector, all startegy profiles are in principle admissible*

Mechanism and Implementation



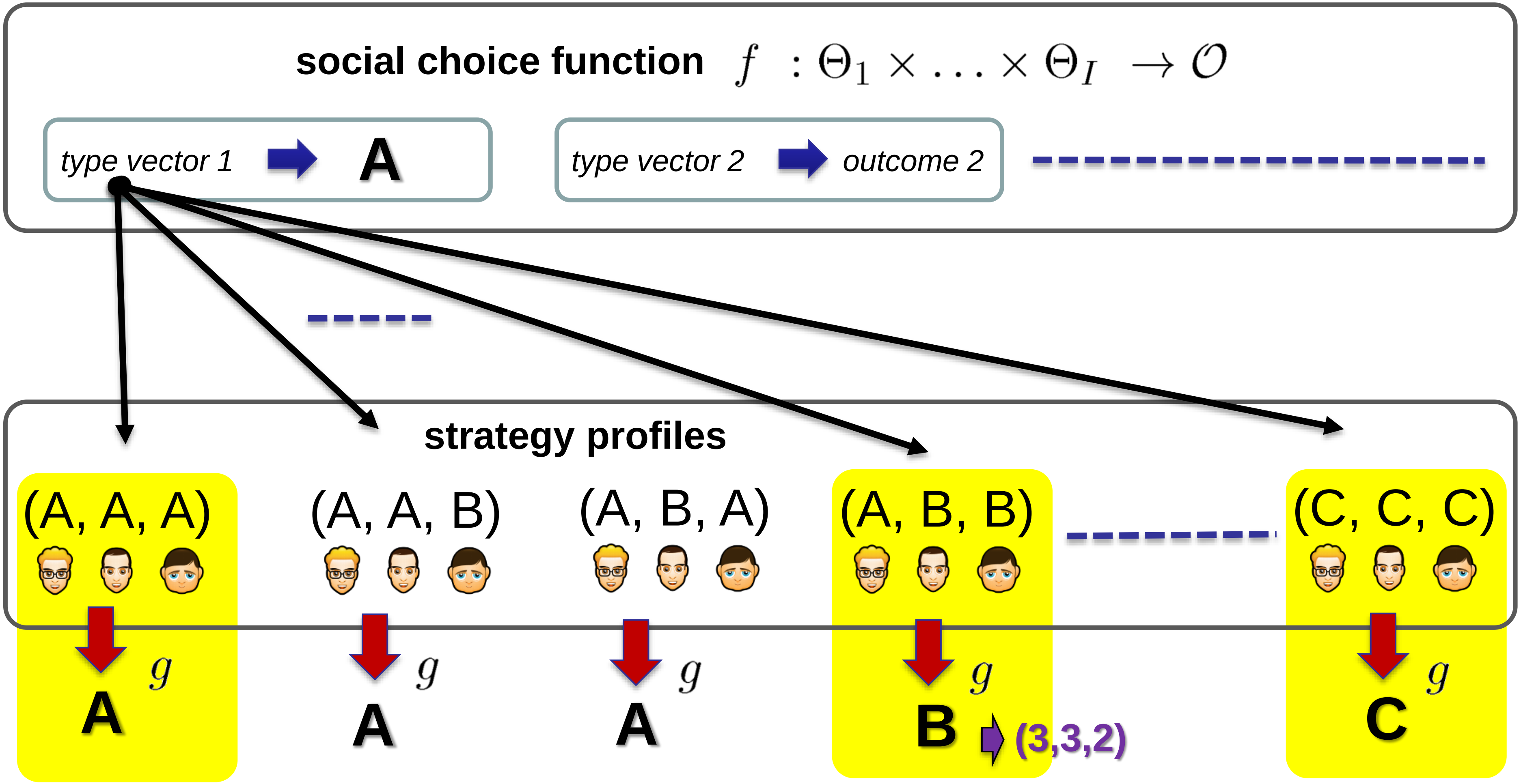
- ➢ *For a given type vector, all startegy profiles are in principle admissible*
- ➢ *An outcome rule is applied*

Mechanism and Implementation



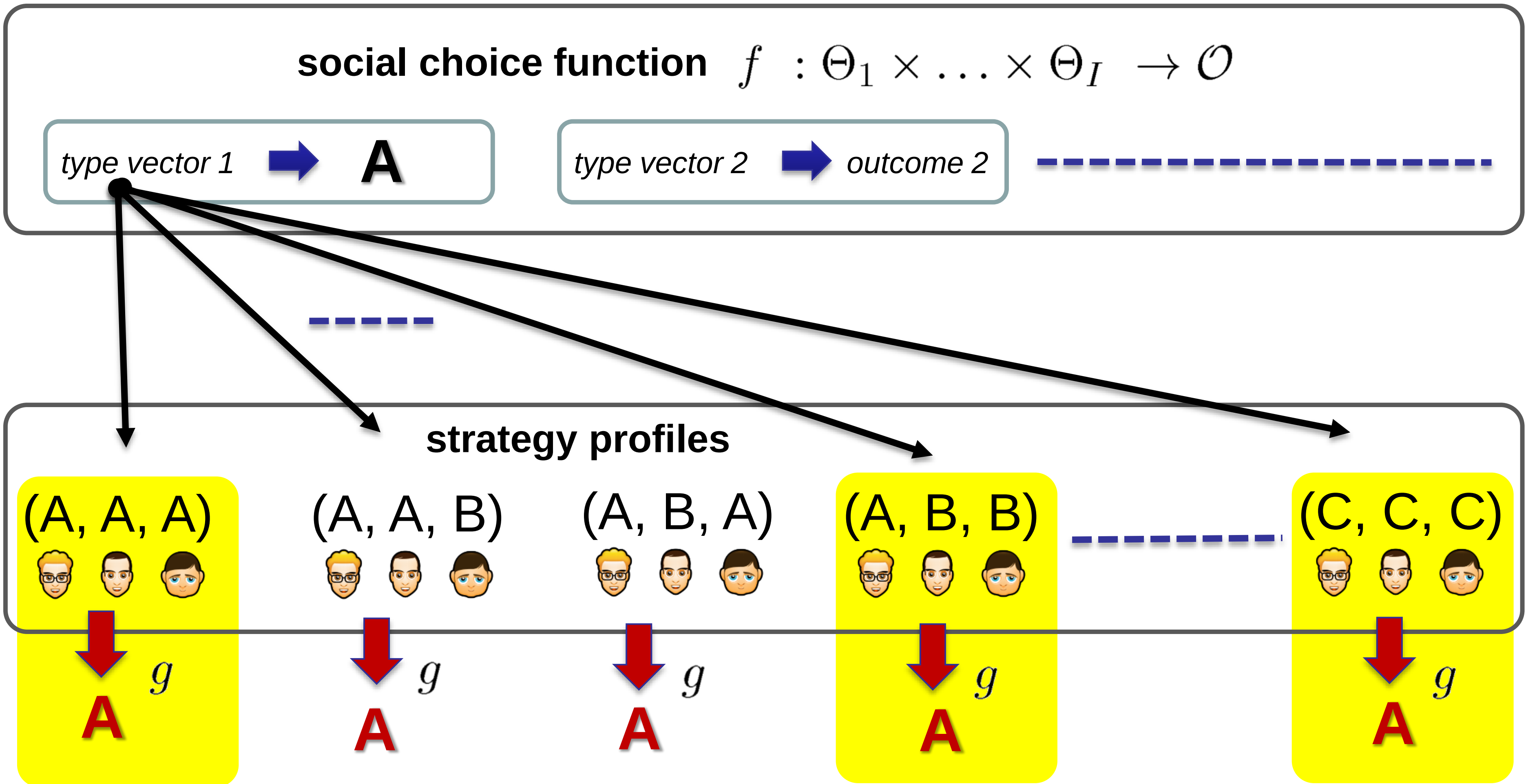
- *For a given type vector, all startegy profiles are in principle admissible*
- *An outcome rule is applied*
- *So, utilities can be computed and equilibria can be selected*

Mechanism and Implementation



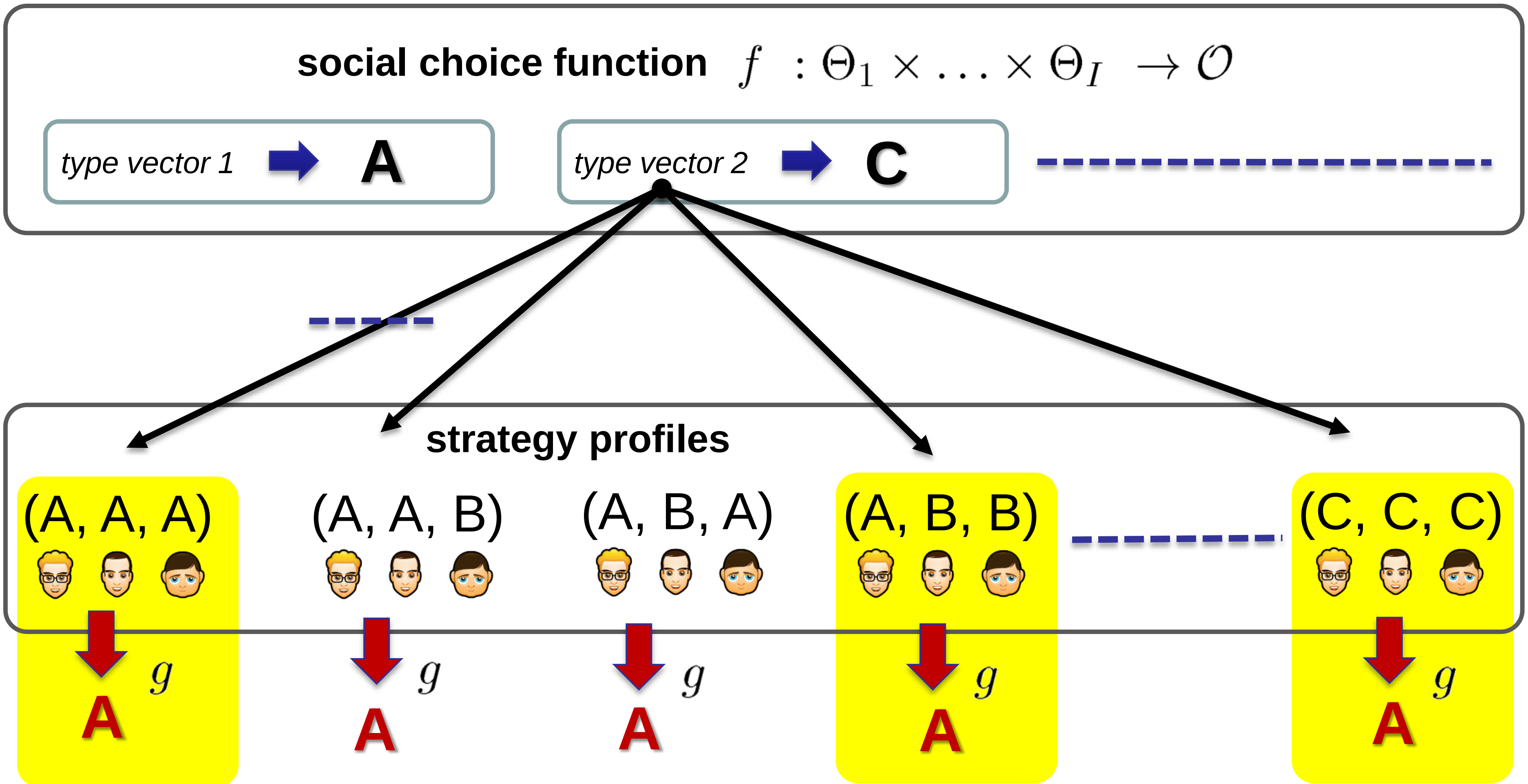
GOAL: In all equlibria, the rule must select the outcome of the social choice function

Mechanism and Implementation



GOAL: In all equlibria, the rule must select the outcome of the social choice function

Mechanism and Implementation



GOAL: and this must happen with any type vector!

Mechanism and Implementation

- A **mechanism** is a tuple $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_I, g(\cdot))$, where
  - for each agent i, Σ_i is the set of available strategies
  - $g : \Sigma_1 \times \ldots \times \Sigma_I \to \mathcal{O}$ is an outcome rule that
    - given a strategy profile $s = (s_1, \ldots, s_I)$
    - selects an outcome $g(s)$

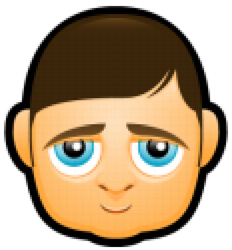
$\mathcal{M}$ **implements** in dominant strategy the social choice function f if,

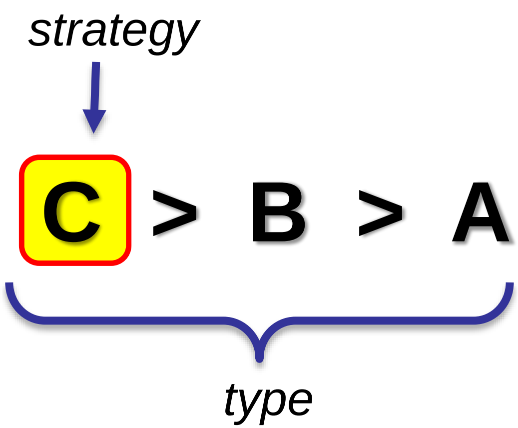
for each type vector $\theta = (\theta_1, \ldots, \theta_I)$,

$$g(s_1^*(\theta_1), \ldots, s_I^*(\theta_I)) = f(\theta)$$

where $(s_1^*, \ldots, s_I^*)$ is a dominant strategy.

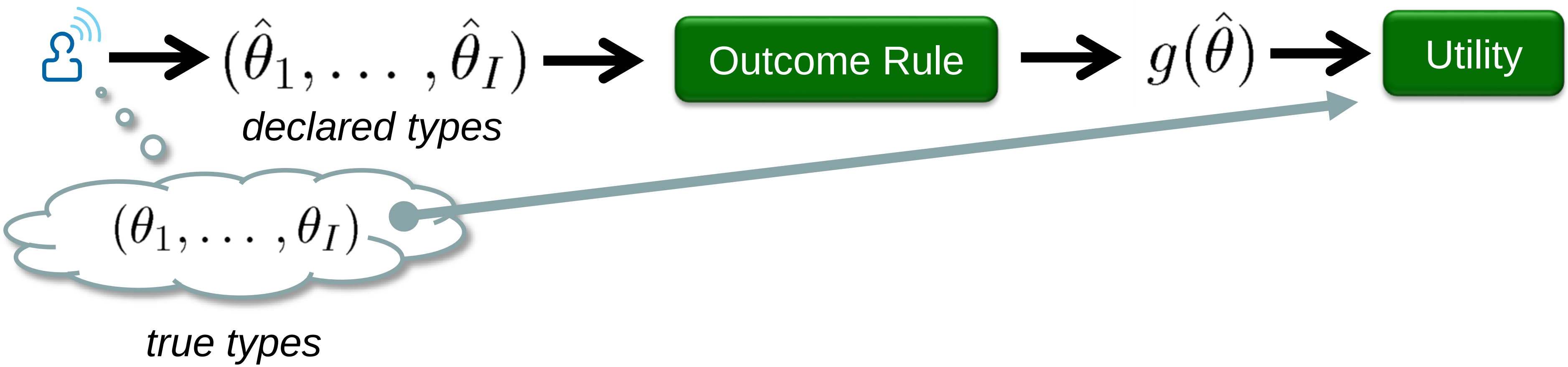
Types VS Strategies





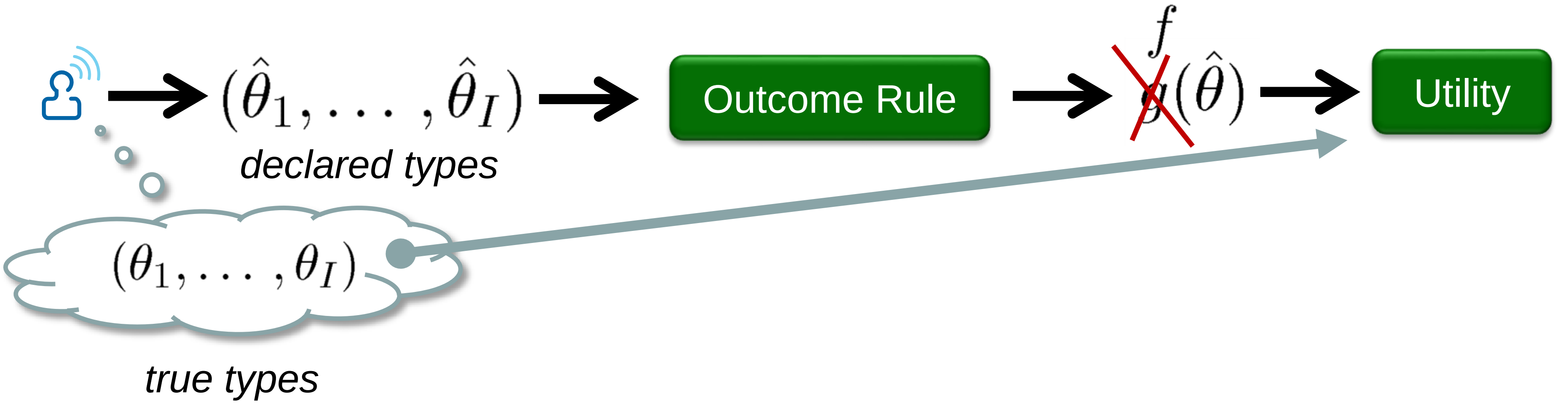
- In a **direct revelation** mechanism, each strategy is restricted to a declaration about the private type

Types VS Strategies



- In a **direct revelation** mechanism, each strategy is restricted to a declaration about the private type

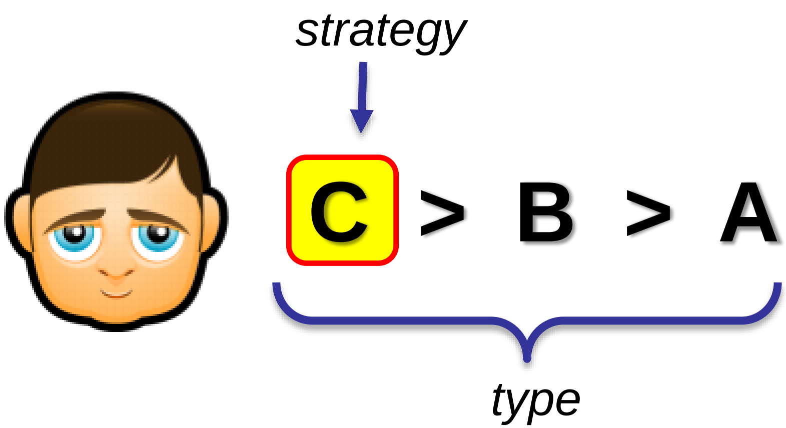
Types VS Strategies



Definition. A direct-revelation mechanism is **strategy-proof** (dominant-strategy incentive-compatible) if truth-revelation is a dominant strategy for each agent.

- If the mechanism implements a function f, then $g = f$

Revelation Principle



THEOREM. If a social choice function can be implemented in dominant strategies, then it can be implemented by a strategy-proof **direct-revelation** mechanism.

- It is a central theoretical tool in mechanism design
  - [Gibbard, 1973]
  - [Green and Laffont, 1977]
  - [Mayerson, 1979]

Impossibility Result

- A social choice function is **dictatorial** if one agent always receives one of its most preferred alternatives

A > C > B

B > A > C

C > B > A

Which functions can be implemented in dominant strategies?

Impossibility Result

- A social choice function is **dictatorial** if one agent always receives one of its most preferred alternatives

- A preference relation is **general** when it defines a complete and transitive ordering over the alternatives

Which functions can be implemented in dominant strategies?

Impossibility Result

THEOREM. Assume general preferences, at least two agents, and at least three optimal outcomes. A social choice function can be **implemented in dominant strategies** if, and only if, it is **dictatorial**.

- Very bad news...
  - [Gibbard, 1973] and [Satterthwaite, 1975]
- …, but must be interpreted with care

The result does not necessarily hold in restricted environments

Which functions can be implemented in dominant strategies?

Payments

Monetary compensation to induce **truthfulness**

- A utility is **quasi-linear** if it has the following form

$$u_i(o, \theta_i) = v_i(o, \theta_i) - p_i$$

valuation function
cardinal preferences

payment by the agent

Payments

Monetary compensation to induce **truthfulness**

- A utility is **quasi-linear** if it has the following form

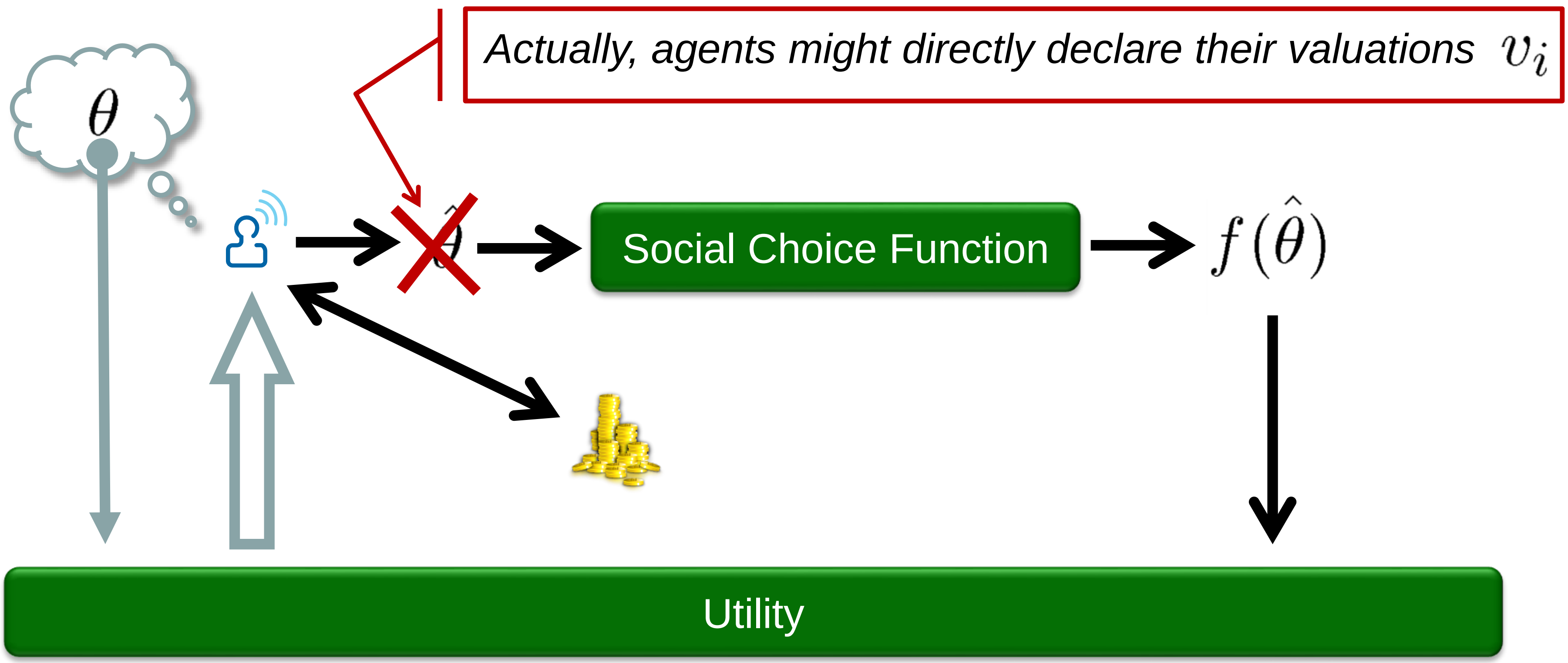
$$u_i(o, \theta_i) = v_i(o, \theta_i) - p_i$$

valuation function
cardinal preferences

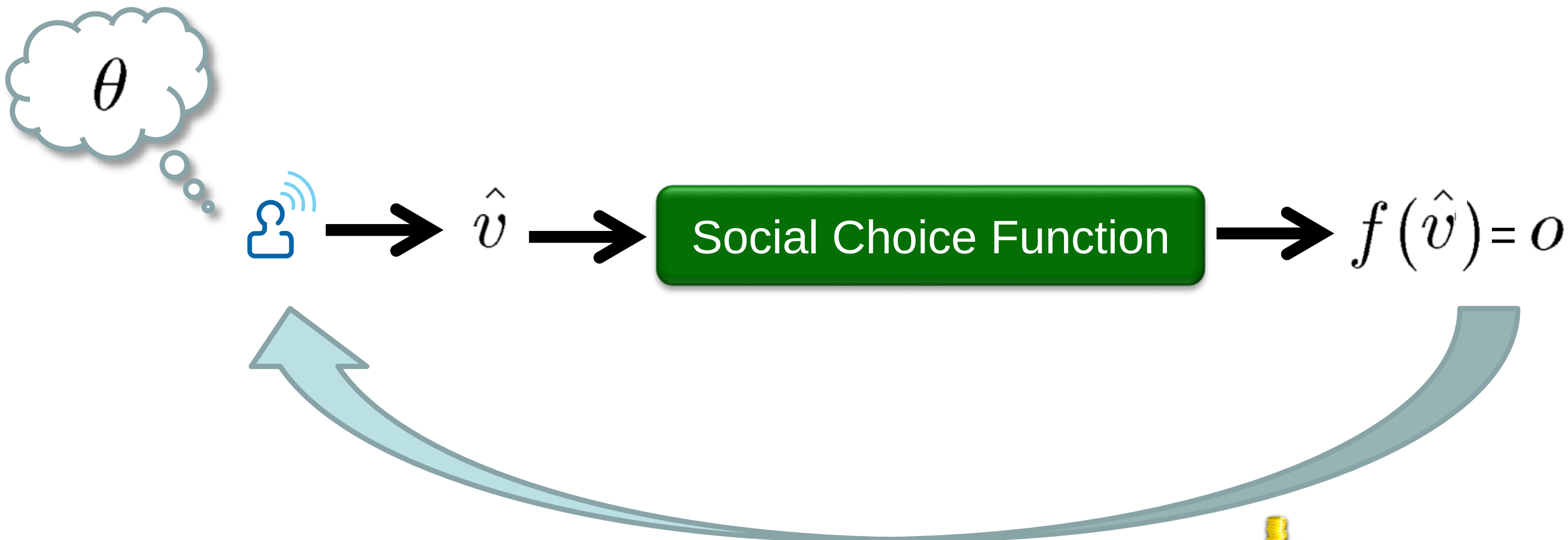
payment by the agent

- Payments are defined by the mechanism

Direct Mechanisms with Payments



Direct Mechanisms with Payments



$$u_i(o, \theta_i) = v_i(o, \theta_i) - p_i$$

Vickrey-Clarke-Groves (VCG) Mechanisms

- Consider **quasi-linear** utilities: $u_i(o, \theta_i) = v_i(o, \theta_i) - p_i$
- Consider social choice functions that are **efficient**:
  - Given v, $f(v)$ maximizes the sum of the valuations

$$\sum_i v_i(f(v)), \theta_i)$$

(1) The mechanism selects the outcome o^* maximizing $\sum_i \hat{v}_i(o, \theta_i)$.

(2) Payments are such that $p_i = h_i - \sum_{j \neq i} \hat{v}_j(o^*, \theta_j)$

Family of mechanisms (e.g., the value of the optimal outcome without the agent)

Vickrey-Clarke-Groves (VCG) Mechanisms

- An auction with one item
- We have bids: $b_1 > b_2 > \cdots > b_n$

Agent *1* receives the item

Agent *1* pays b_2

(1) **The r** … **ne o^* maximizing** $\sum_i \hat{v}_i(o, \theta_i)$.

(2) **Payments are such that** $p_i = h_i - \sum_{j \neq i} \hat{v}_j(o^*, \theta_j)$

Family of mechanisms (e.g., the value of the optimal outcome without the agent)

Payment Rules (Again…)

- Monetary compensation to induce **truthfulness**

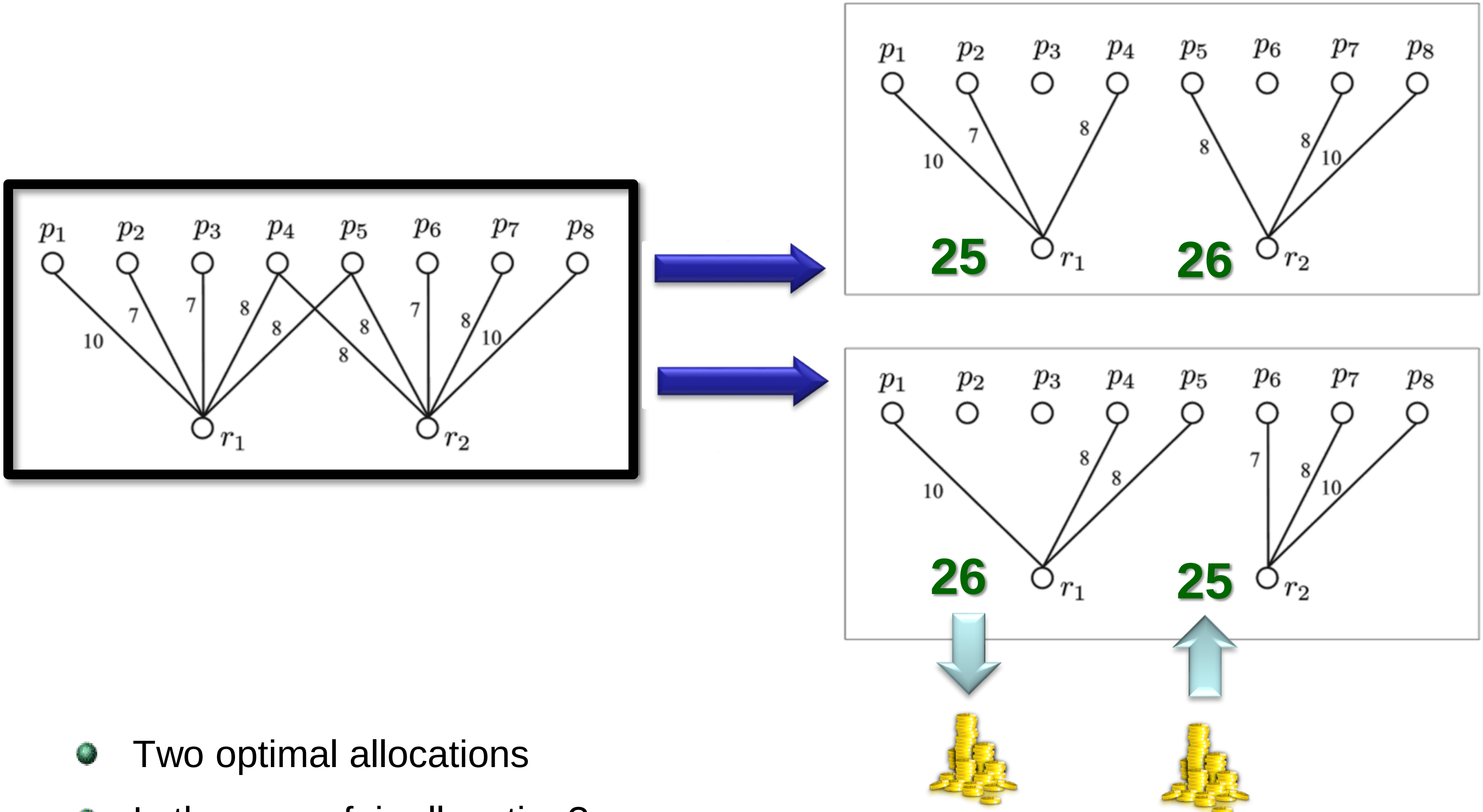
GOAL: Budget Balance

✓ The algebraic sum of the monetary transfers is zero
✓ In particular, mechanisms cannot run into deficit

- Monetary compensation to induce **fairness**

✓ For instance, it is desirable that ***no agent envies*** the allocation of any another agent, or that
✓ The outcome is ***Pareto efficient***, i.e., there is no different allocation such that every agent gets at least the same utility and one of them improves.

Fairness vs Efficiency



- Two optimal allocations
- Is there any fair allocation?

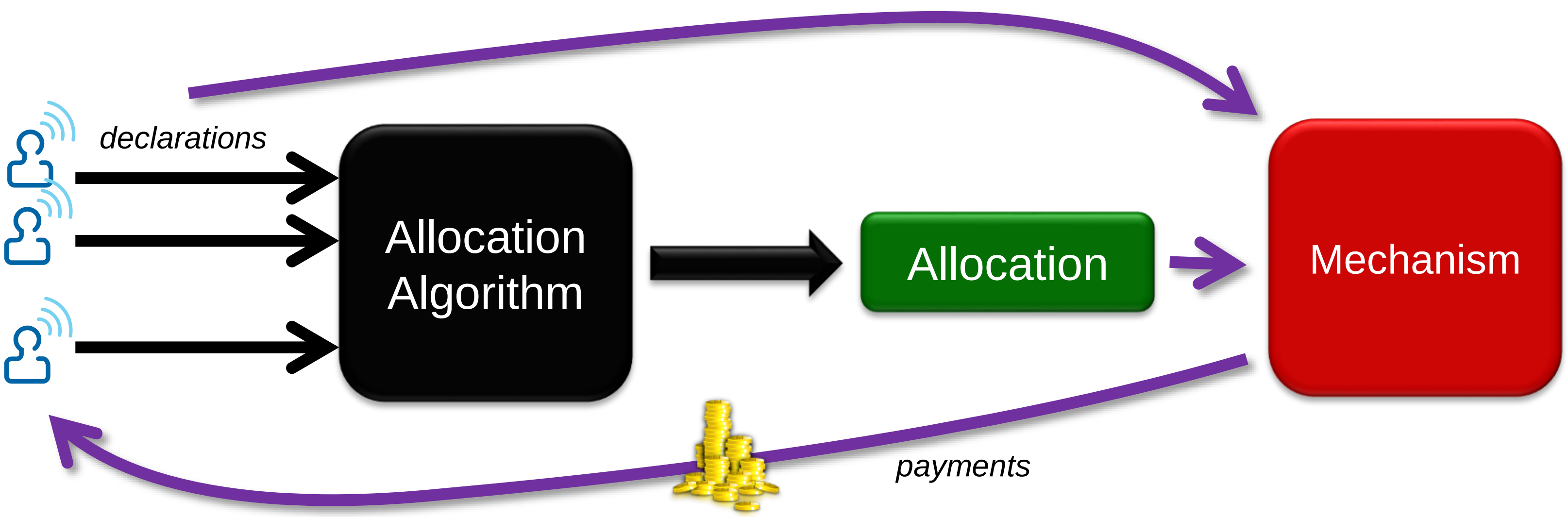
(A Few…) Impossibility Results

Efficiency **+** Truthfulness **+** Budget Balance

[Green, Laffont; 1977]
[Hurwicz; 1975]

Fairness **+** Truthfulness **+** Budget Balance

[Tadenuma, Thomson;1995]
[Alcalde, Barberà; 1994]
[Andersson, Svensson, Ehlers; 2010]



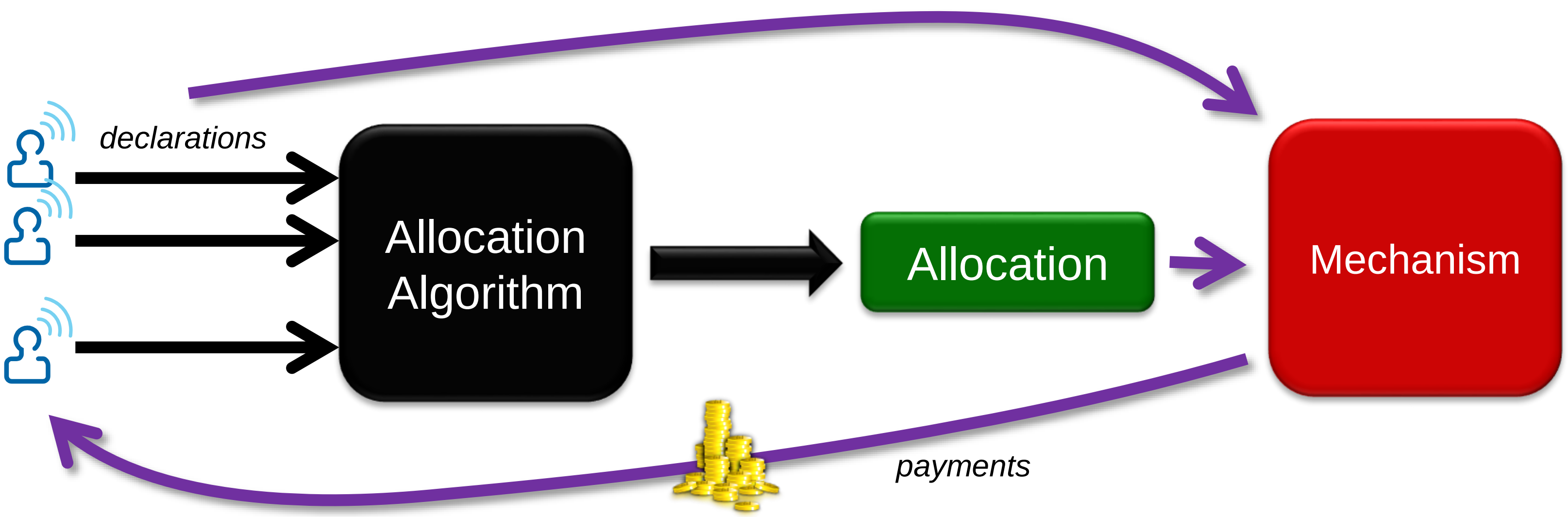
Outline

- Game Theory
- Mechanism Design
- Mechanisms with Verification
- Mechanisms and Allocation Problems
- Complexity Analysis

(A Few…) Impossibility Results

Efficiency **+** Truthfulness **+** Budget Balance

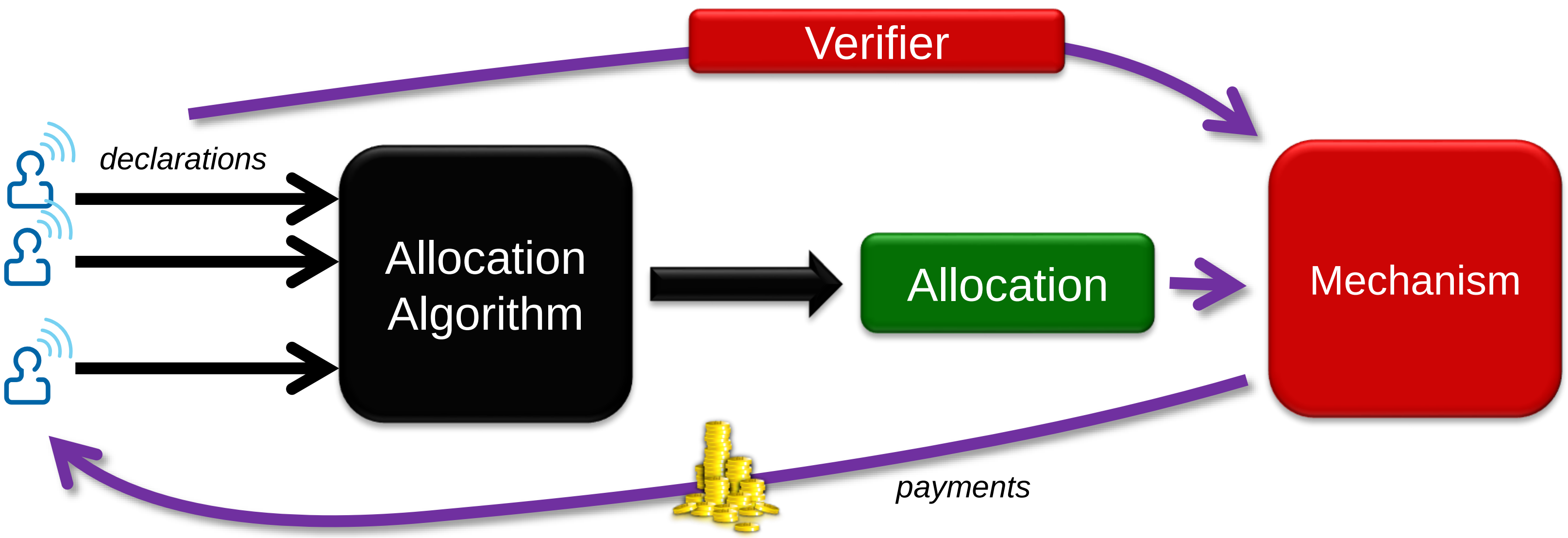
Fairness **+** Truthfulness **+** Budget Balance



(A Few…) Impossibility Results

- Efficiency **+** Truthfulness **+** Budget Balance
- Fairness **+** Truthfulness **+** Budget Balance

- Verification on «selected» declarations



Approaches to Verification

(1) Partial Verification

(2) Probabilistic Verification

Approaches to Verification

(1) Partial Verification

[Green, Laffont; 1986]
[Nisan, Ronen; 2001]

(2) Probabilistic Verification

Approaches to Verification

(1) Partial Verification

[Auletta, De Prisco, Ferrante, Krysta, Parlato, Penna, Persiano, Sorrentino, Ventre]

(2) Probabilistic Verification

Approaches to Verification

(1) Partial Verification

[Auletta, De Prisco, Ferrante, Krysta, Parlato, Penna, Persiano, Sorrentino, Ventre]

(2) Probabilistic Verification

[Caragiannis, Elkind, Szegedy, Yu; 2012]

Approaches to Verification

(1) Partial Verification

(2) Probabilistic Verification

Punishments are used to enforce truthfulness

Approaches to Verification

(1) Partial Verification

(2) Probabilistic Verification

Punishments are used to enforce truthfulness

- Verification is performed via **sensing**
  - Hence, it is subject to errors; for instance, because of the limited precision of the measurement instruments.
  - It might be problematic to decide whether an observed discrepancy between verified values and declared ones is due to a strategic behavior or to such sensing errors.

[Greco, Scarcello; 2014]

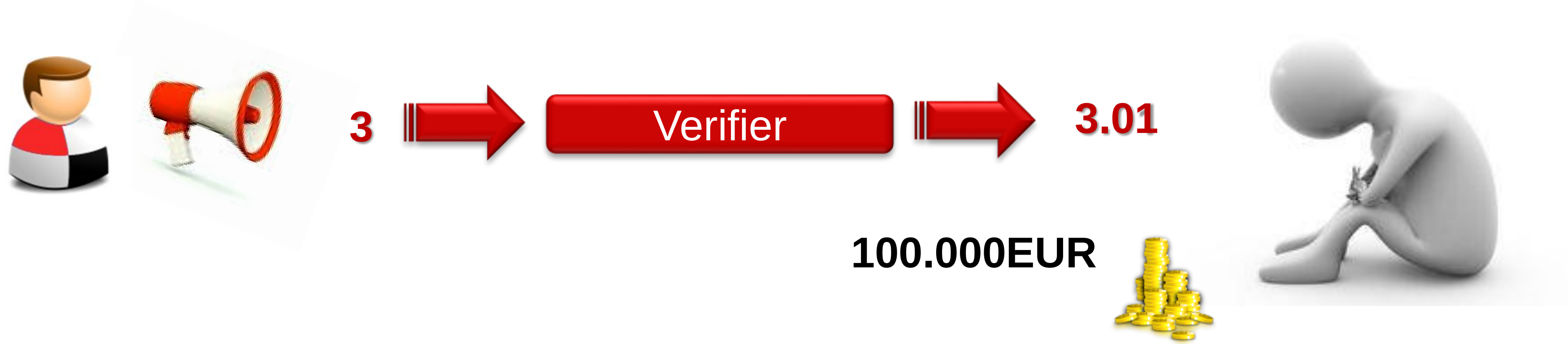
Approaches to Verification

- Verification is performed via **sensing**
  - Hence, it is subject to errors; for instance, because of the limited precision of the measurement instruments.
  - It might be problematic to decide whether an observed discrepancy between verified values and declared ones is due to a strategic behavior or to such sensing errors.

Approaches to Verification (bis)

- Agents might be uncertain of their private features; for instance, due to limited computational resources
  - There might be no strategic issues

Approaches to Verification (ter)



- Punishments enforce truthfulness
  - They might be disproportional to the harm done by misreporting
  - Inappropriate in real life situations in which uncertainty is inherent due to measurements errors or uncertain inputs.

[Feige, Tennenholtz; 2011]

Approaches to Verification

(1) Partial Verification

(2) Probabilistic Verification

Punishments are used to enforce truthfulness

(3) Full Verification

The verifier returns a value.

Approaches to Verification

(1) Partial Verification

(2) Probabilistic Verification

Punishments are used to enforce truthfulness

(3) Full Verification

The verifier returns a value. But,…

- ***no punishment***
  - payments are always computed under the presumption of innocence, where incorrect declared values do not mean manipulation attempts by the agents
- **error tolerance**
  - the consequences of errors in the declarations produce a linear "distorting effect" on the various properties of the mechanism

Payment Rules

- Monetary compensation to induce **truthfulness**

GOAL: Budget Balance

✓ The algebraic sum of the monetary transfers is zero
✓ In particular, mechanisms cannot run into deficit

- Monetary compensation to induce **fairness**

✓ For instance, it is desirable that ***no agent envies*** the allocation of any another agent, or that
✓ The outcome is ***Pareto efficient***, i.e., there is no different allocation such that every agent gets at least the same utility and one of them improves.

Payment Rules & Full Verification

- Monetary compensation to induce **truthfulness**

GOAL: Budget Balance

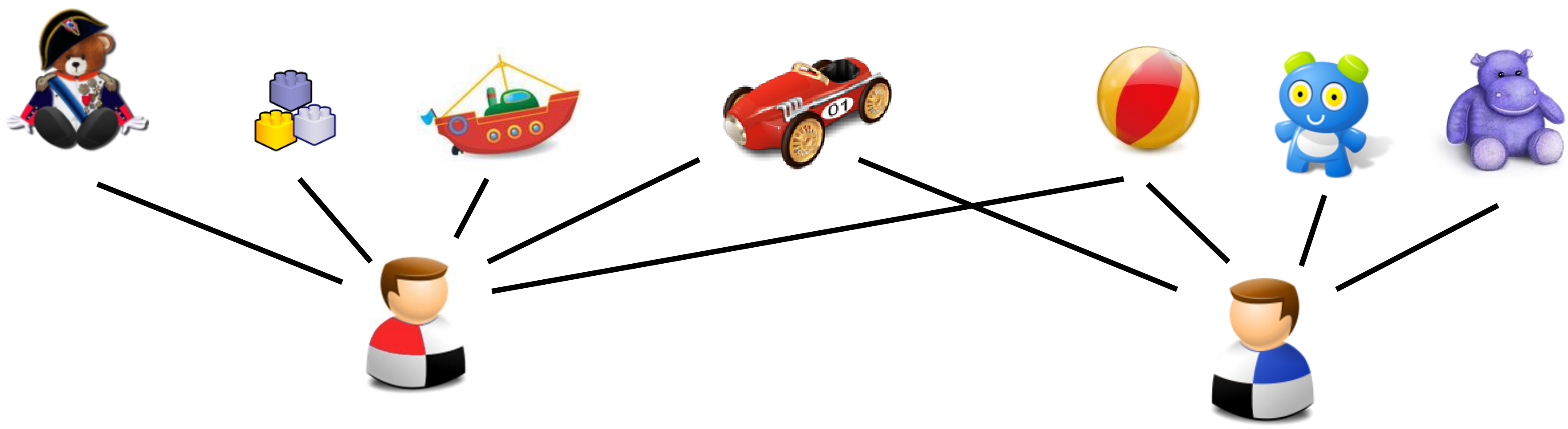
- ✓ The algebraic sum of the monetary transfers is zero
- ✓ In particular, mechanisms cannot run into deficit

- Monetary compensation to induce **fairness**
  - ✓ For instance, it is desirable that ***no agent envies*** the allocation of any another agent, or that
  - ✓ The outcome is ***Pareto efficient***, i.e., there is no different allocation such that every agent gets at least the same utility and one of them improves.

Outline

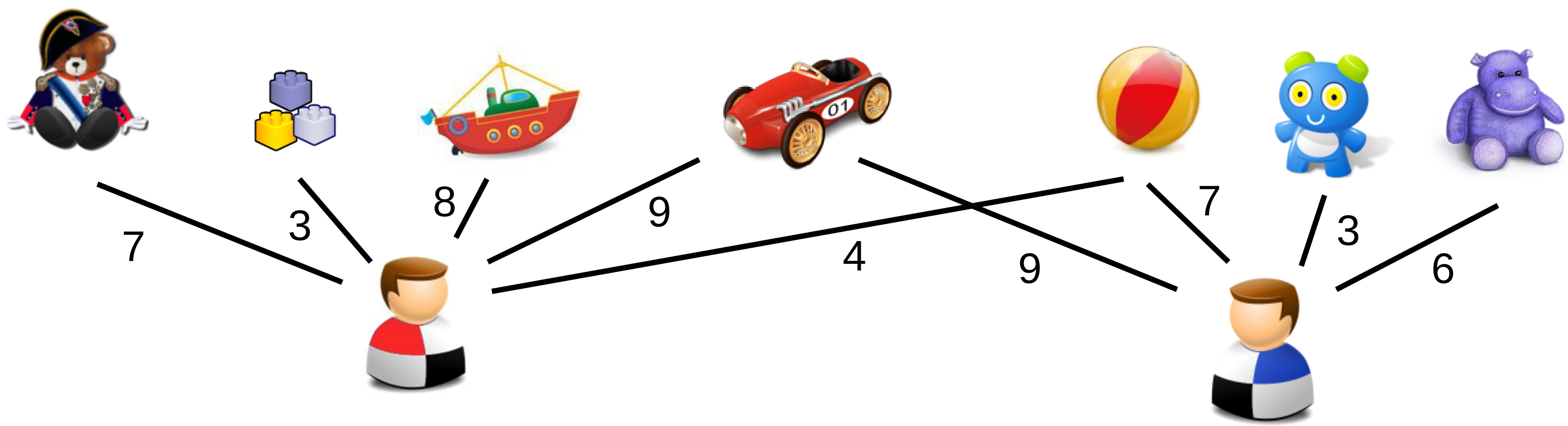
- Game Theory
- Mechanism Design
- Mechanisms with Verification
- Mechanisms and Allocation Problems
- Complexity Analysis

The Model



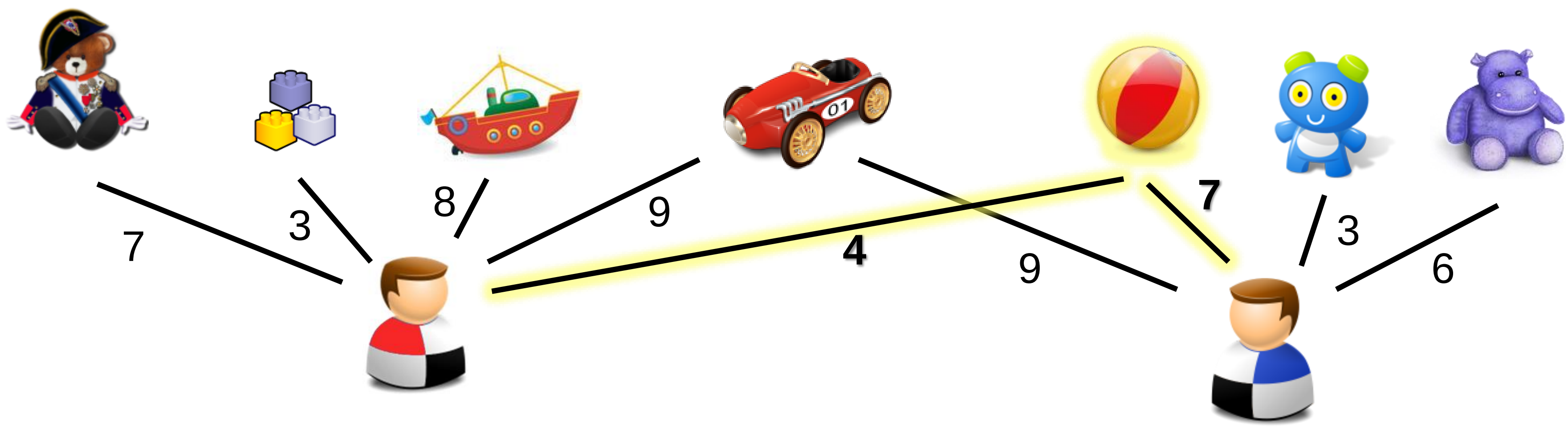
- Goods are indivisible and non-sharable
- Constraints on the maximum number of goods to be allocated to each agent
- Cardinal preferences: *Utility functions*

The Model



- Goods are indivisible and non-sharable
- Constraints on the maximum number of goods to be allocated to each agent
- Cardinal preferences: *Utility functions*

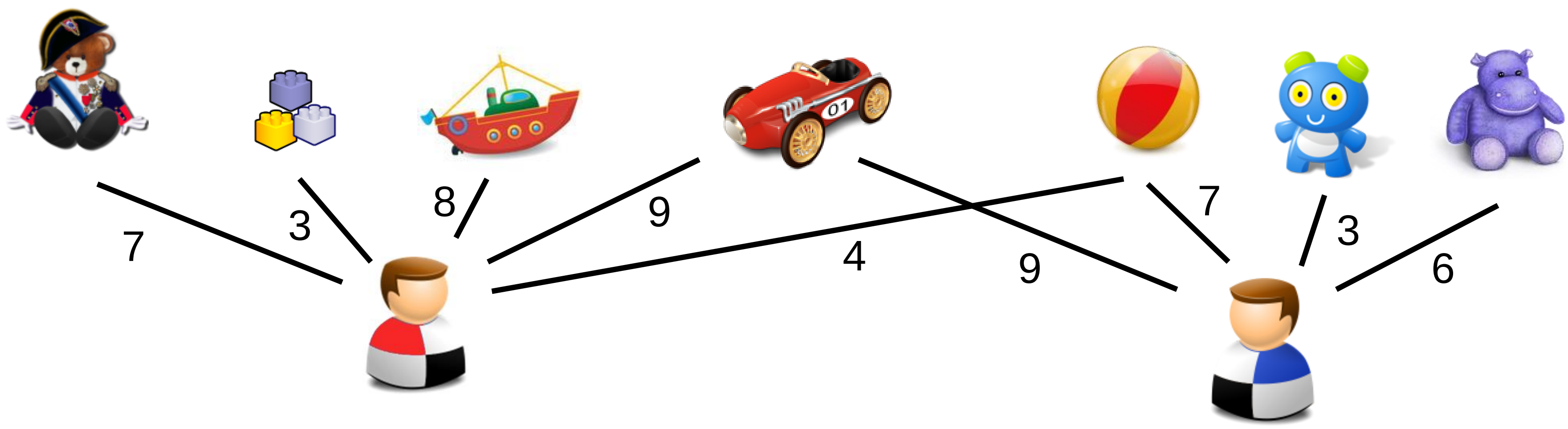
The Model



- Goods are indivisible and non-sharable
- Constraints on the maximum number of goods to be allocated to each agent
- Cardinal preferences: *Utility functions*

Different agents might have different valuations for the same good

The Model

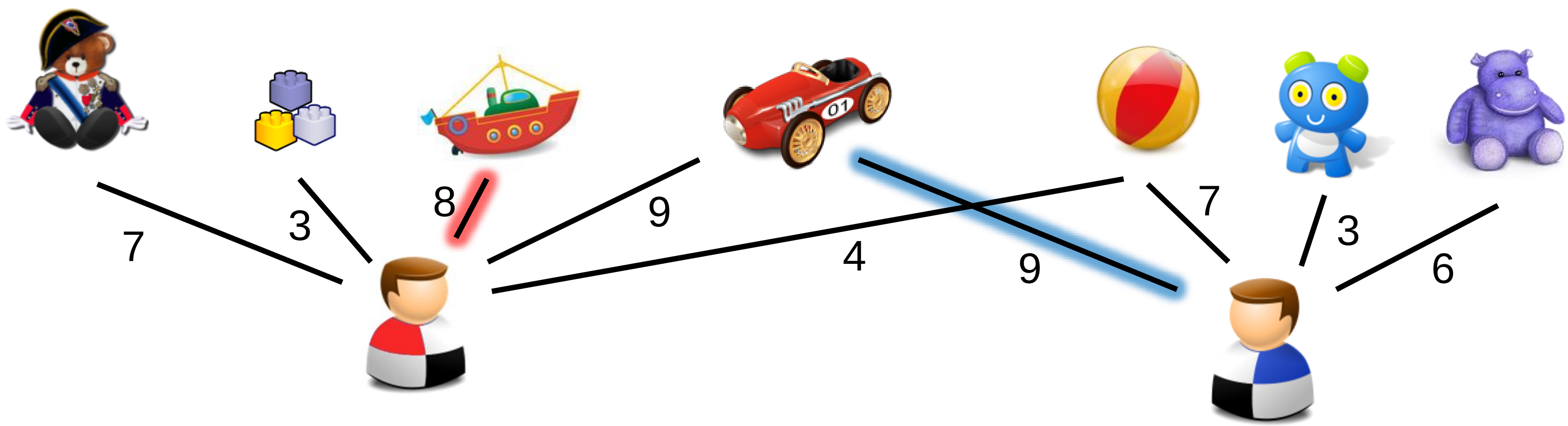


- Goods are indivisible and non-sharable
- Constraints on the maximum number of goods to be allocated to each agent
- Cardinal preferences: *Utility functions*

GOAL: Optimal Allocations

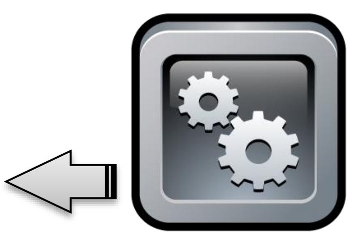
✓ Social Welfare
✓ Efficiency

The Model



- Goods are indivisible and non-sharable
- Constraints on the maximum number of goods to be allocated to each agent
- Cardinal preferences: *Utility functions*

GOAL: Optimal Allocations



✓ Social Welfare
✓ Efficiency

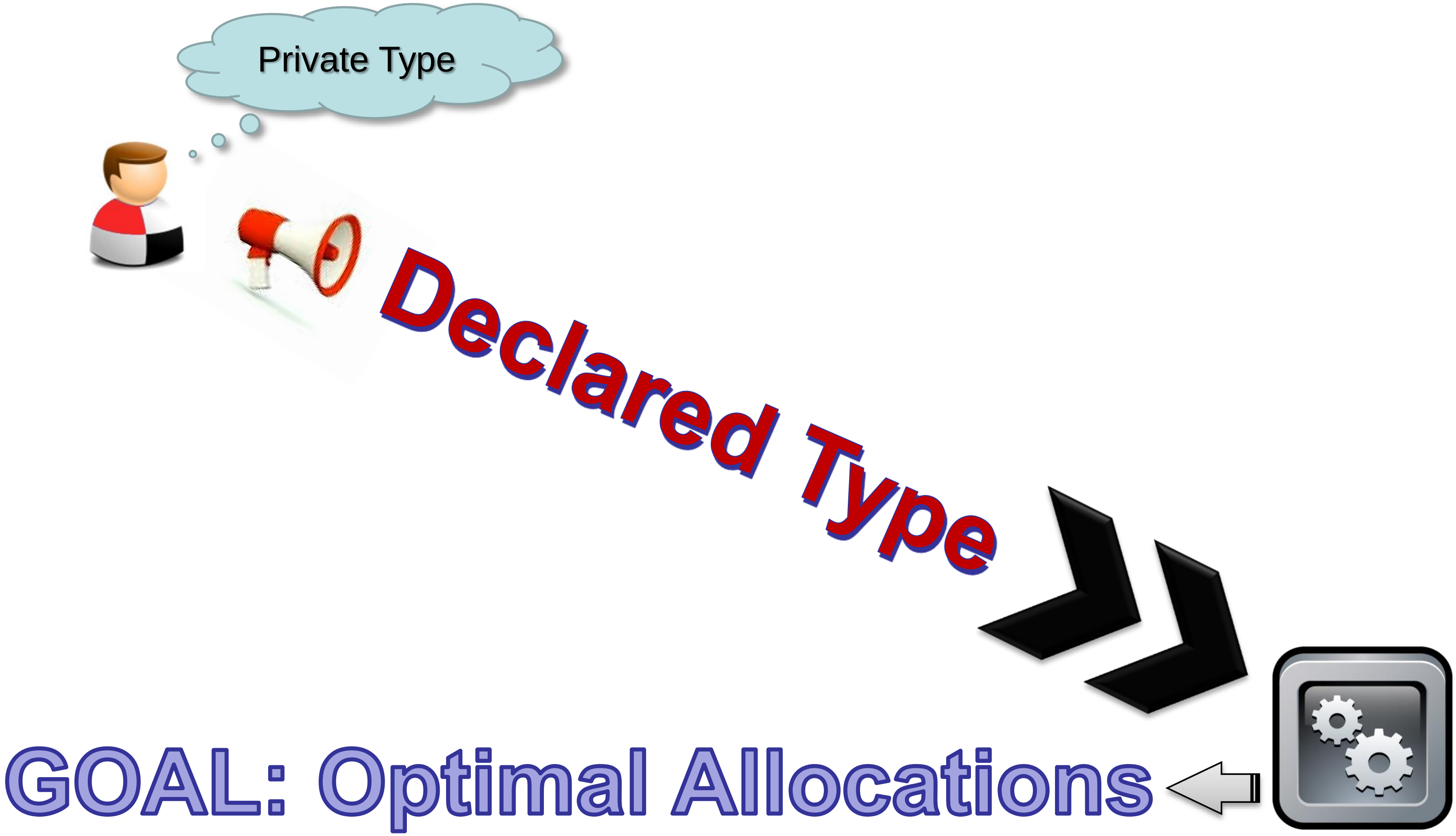
Strategic Issues

Private Type

GOAL: Optimal Allocations

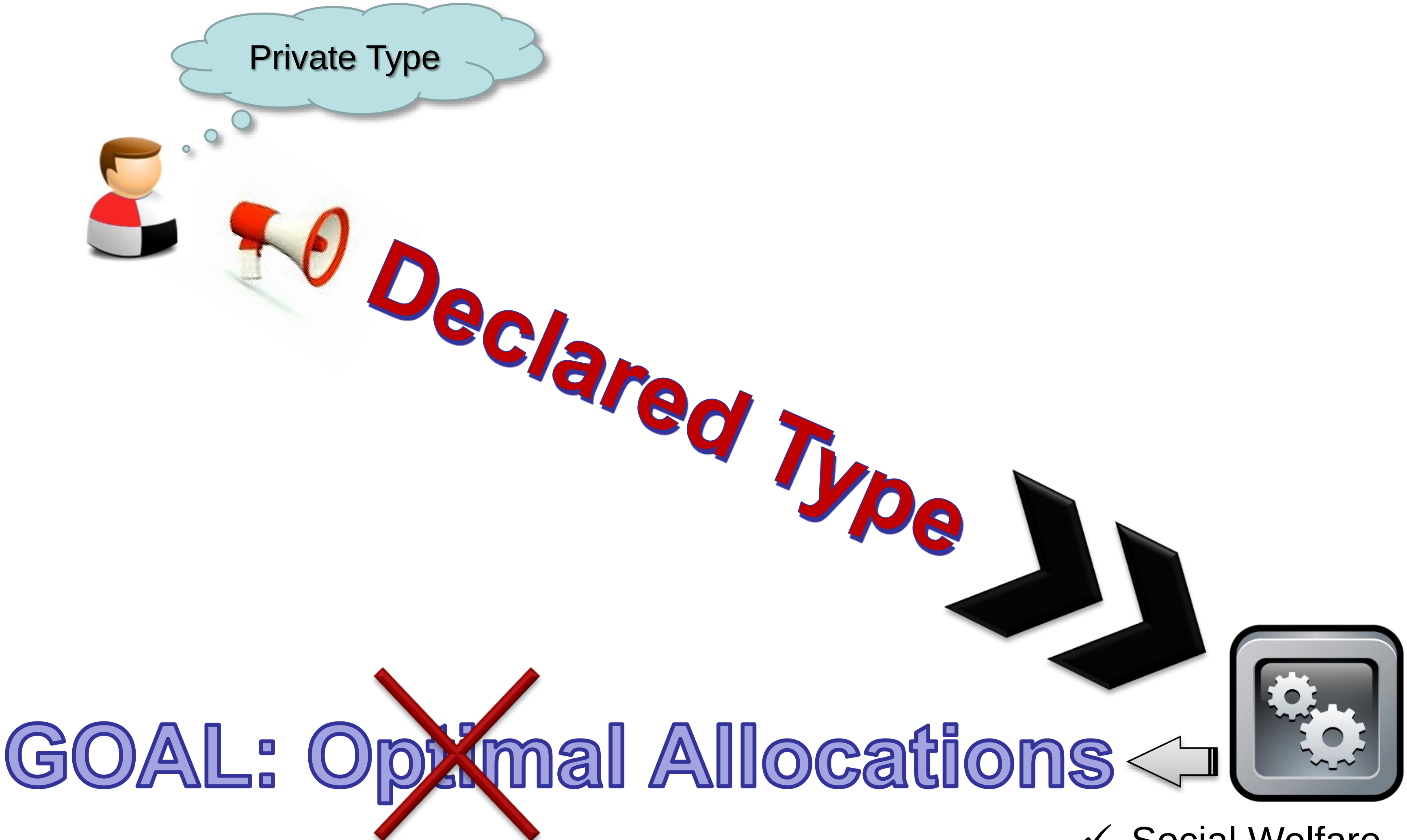
- ✓ Social Welfare
- ✓ Efficiency

Strategic Issues



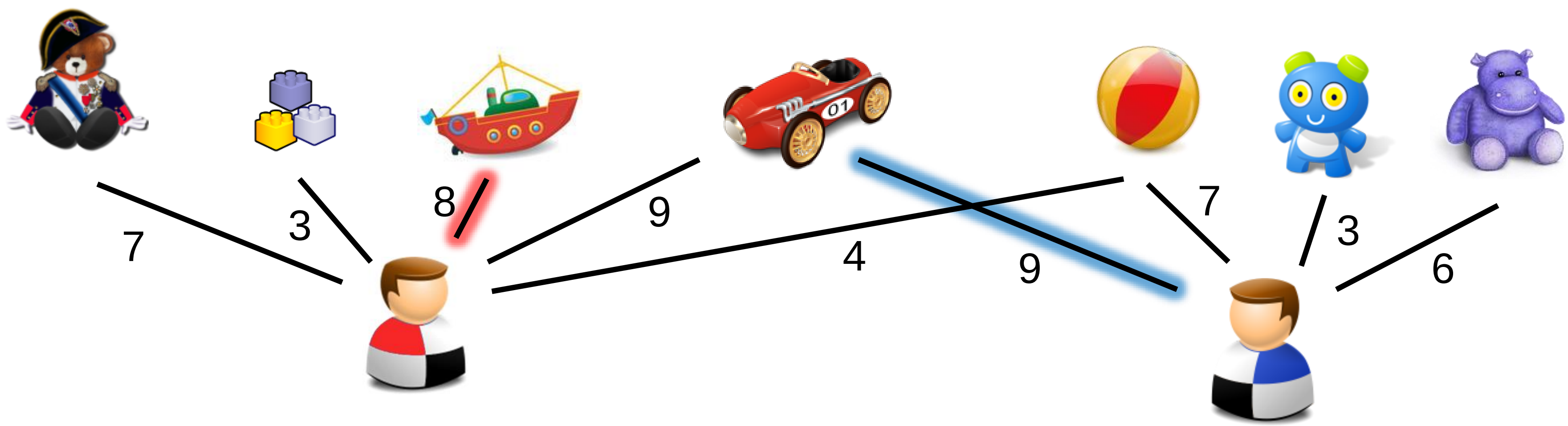
- ✓ Social Welfare
- ✓ Efficiency

Strategic Issues



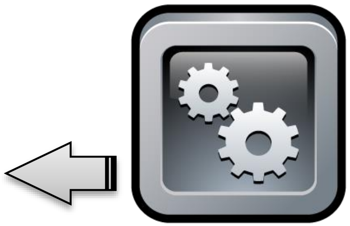
- ✓ Social Welfare
- ✓ Efficiency

Strategic Issues: Example



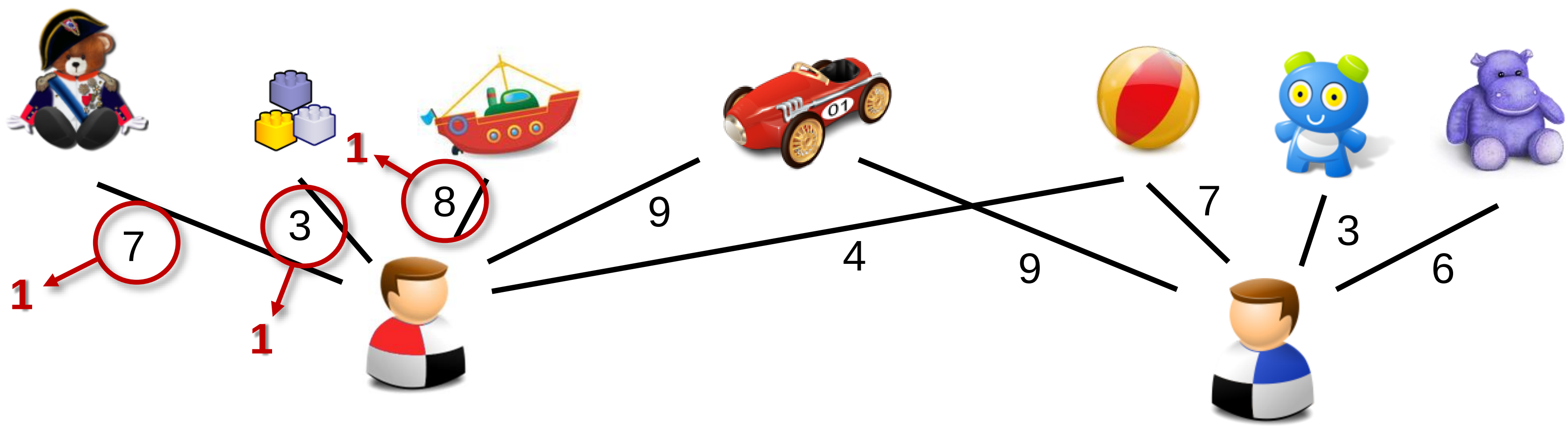
Before: 8+9=17

GOAL: Optimal Allocations



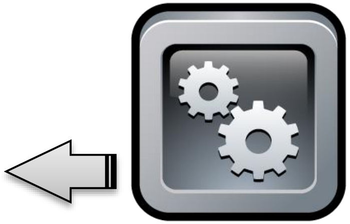
- ✓ Social Welfare
- ✓ Efficiency

Strategic Issues: Example



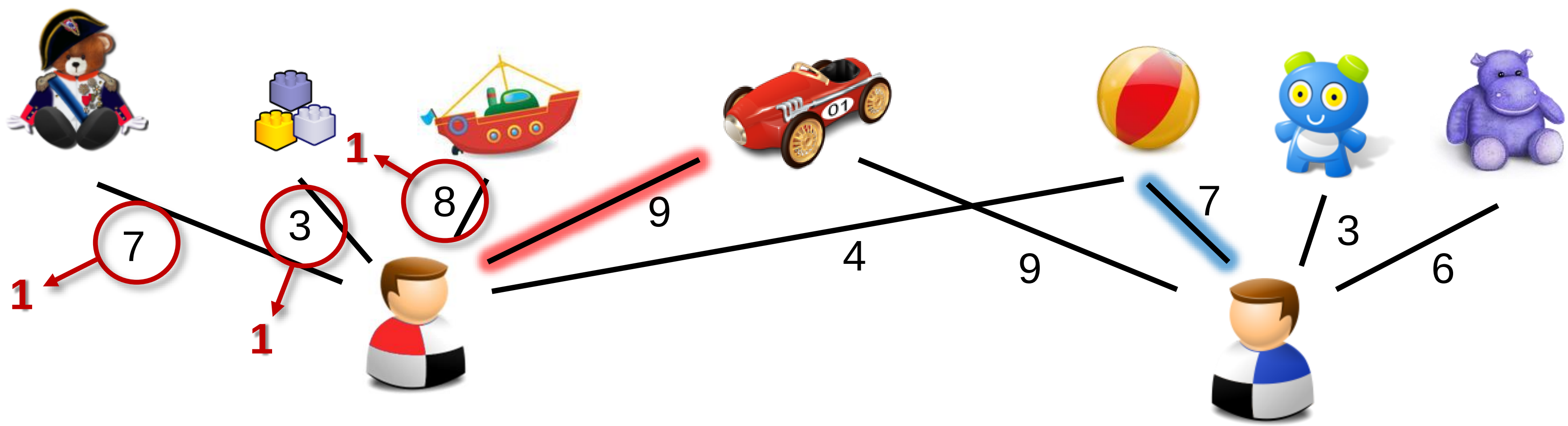
✓ Before: 8+9=17

GOAL: Optimal Allocations



- ✓ Social Welfare
- ✓ Efficiency

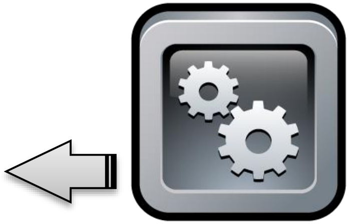
Strategic Issues: Example



Before: 8+9=17

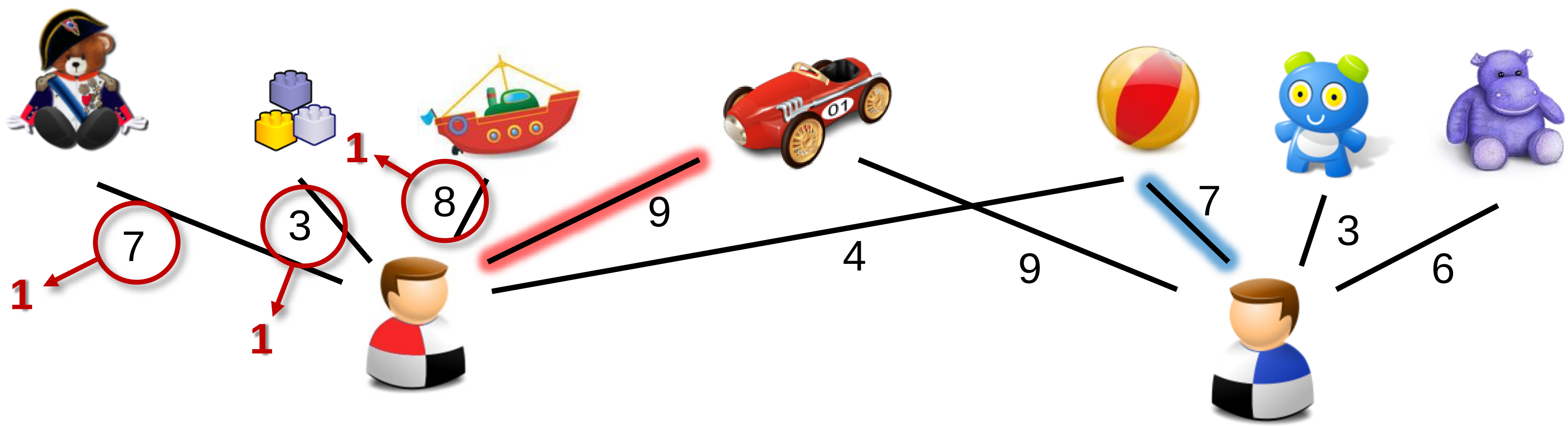
After: 9+7=16

GOAL: Optimal Allocations



- ✓ Social Welfare
- ✓ Efficiency

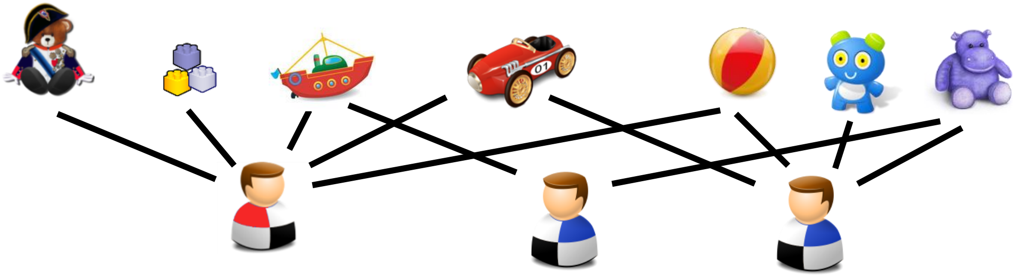
Strategic Issues: Verification



We assume full-verification.

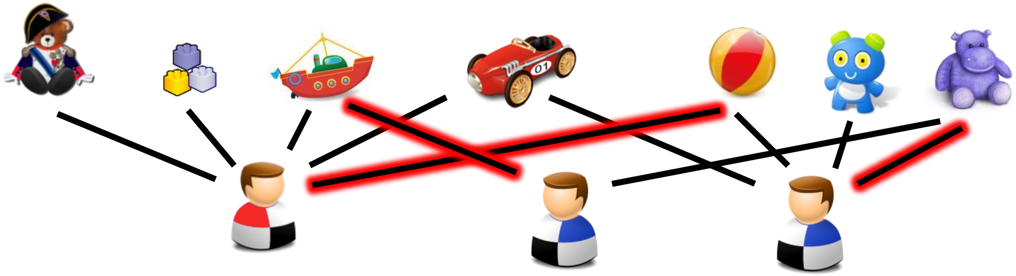
But, of course, we can verify only the goods that are selected.

A Key Lemma



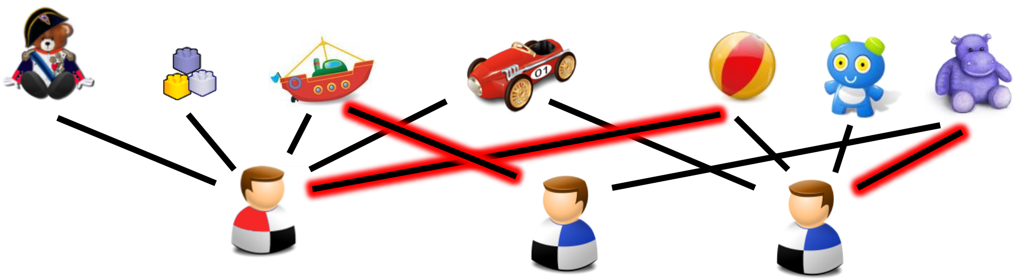
- Consider an optimal allocation (w.r.t. some declared types)

A Key Lemma



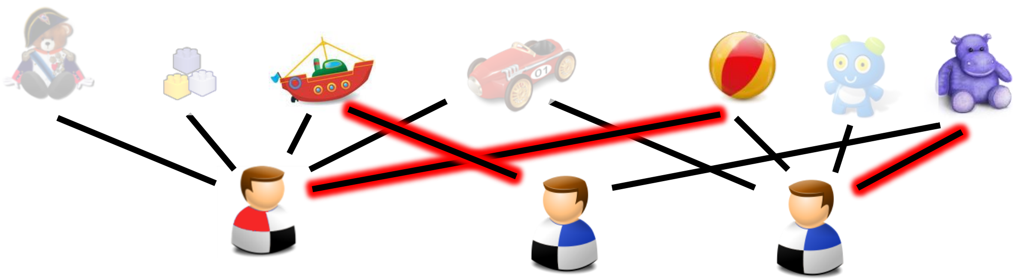
- Consider an optimal allocation (w.r.t. some declared types)

A Key Lemma



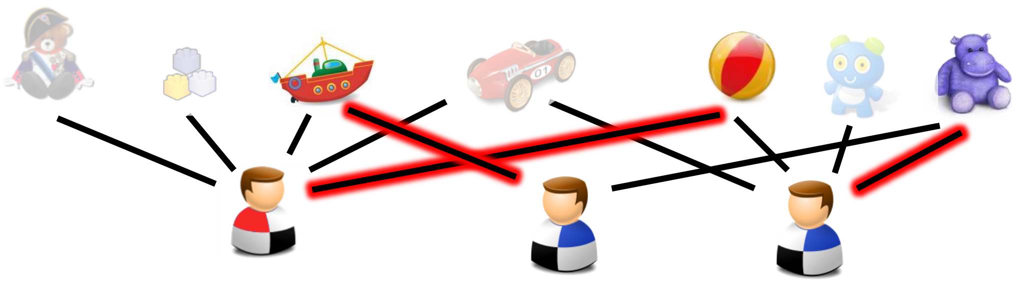
- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…

A Key Lemma



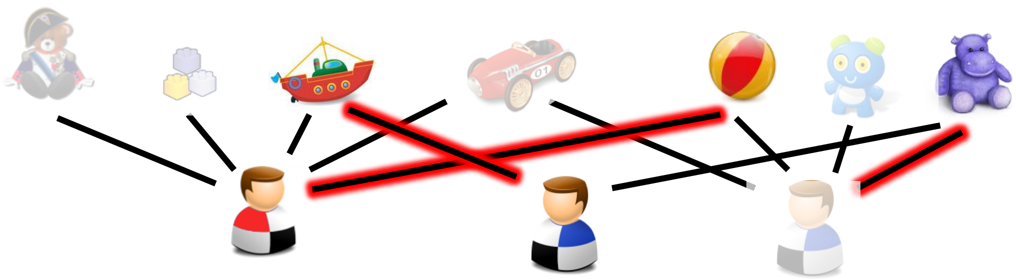
- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…

A Key Lemma



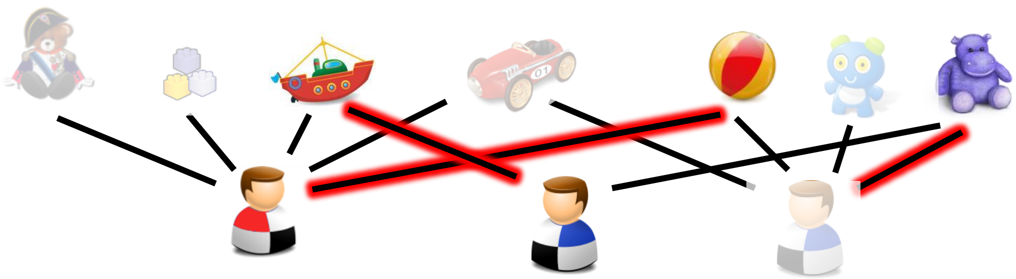
- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…
- Focus on an arbitrary coalition of agents

A Key Lemma



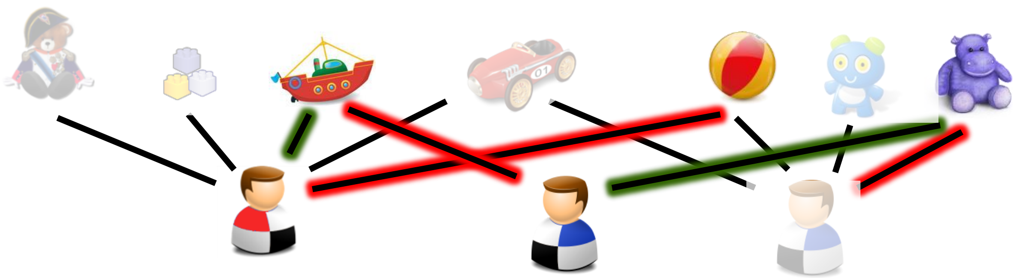
- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…
- Focus on an arbitrary coalition of agents

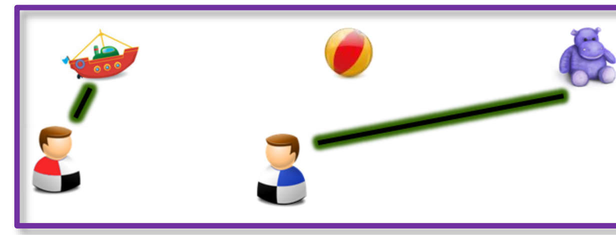
A Key Lemma



- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…
- Focus on an arbitrary coalition of agents
- In this novel setting, compute an optimal allocation

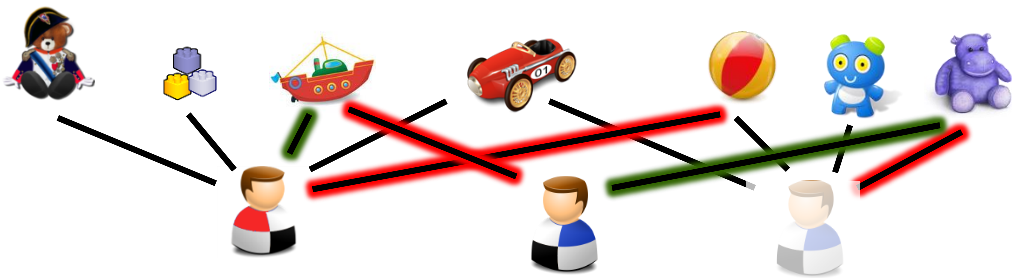
A Key Lemma

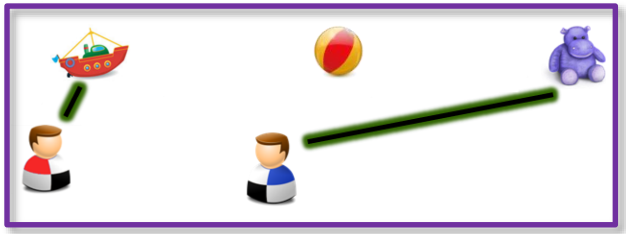




- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…
- Focus on an arbitrary coalition of agents
- In this novel setting, compute an optimal allocation

A Key Lemma





- Consider an optimal allocation (w.r.t. some declared types)
- Ignore the goods that are not allocated,
  - and hence that cannot be verified later…
- Focus on an arbitrary coalition of agents
- In this novel setting, compute an optimal allocation

❖ **The allocation is also optimal for that coalition, even if all goods were actually available**

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;

Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
For each set $\mathcal{C} \in \mathbb{C}$,
 Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \mathtt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
For each agent $i \in \mathcal{A}$,
 For each set $\mathcal{C} \in \mathbb{C}$,
  Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $\quad (= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
  Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $\quad (= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
 Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
 Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

---

1. Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
2. For each set $\mathcal{C} \in \mathbb{C}$,
3. $\lfloor$ Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \mathtt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
4. For each agent $i \in \mathcal{A}$,
5. | For each set $\mathcal{C} \in \mathbb{C}$,
6. | | Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $(= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
7. | $\lfloor$ Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $(= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
8. | Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}| - 1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
9. $\lfloor$ Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
For each set $\mathcal{C} \in \mathbb{C}$,
  Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \texttt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
For each agent $i \in \mathcal{A}$,
  For each set $\mathcal{C} \in \mathbb{C}$,
    Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \texttt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $(= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
    Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \texttt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $(= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
  Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
  Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

By the previous lemma, this is without loss of generality.
In fact, allocated goods are the only ones that we verify.

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

1. Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
2. For each set $\mathcal{C} \in \mathbb{C}$,
3. Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \mathtt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
4. For each agent $i \in \mathcal{A}$,
5. For each set $\mathcal{C} \in \mathbb{C}$,
6. Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $\quad (= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
7. Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $\quad (= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
8. Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
9. Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

«Bonus and Compensation»,
by Nisan and Ronen (2001)

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
For each set $\mathcal{C} \in \mathbb{C}$,
  Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \mathtt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
For each agent $i \in \mathcal{A}$,
  For each set $\mathcal{C} \in \mathbb{C}$,
    Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $\quad (= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
    Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $\quad (= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
  Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}|-|\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
  Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

«Bonus and Compensation», by Nisan and Ronen (2001)

No punishments!

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

1. Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
2. For each set $\mathcal{C} \in \mathbb{C}$,
3. | Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \mathtt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
4. For each agent $i \in \mathcal{A}$,
5. | For each set $\mathcal{C} \in \mathbb{C}$,
6. | | Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $(= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
7. | | Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $(= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
8. | Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}|-|\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
9. | Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

«Bonus and Compensation», by Nisan and Ronen (2001)

❖ **Truth-telling is a dominant strategy for each agent**

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, \ldots, v_n)$;

1. Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
2. For each set $\mathcal{C} \in \mathbb{C}$,
3. ⌊ Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \mathtt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
4. For each agent $i \in \mathcal{A}$,
5. | For each set $\mathcal{C} \in \mathbb{C}$,
6. | | Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $\quad (= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
7. | ⌊ Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \mathtt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $\quad (= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
8. | Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}| - 1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
9. ⌊ Define $p^{\xi}_i(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

Does not depend on *i*

Is maximized when the declared type coincides with the verified one

«Bonus and Compensation», by Nisan and Ronen (2001)

❖ **Truth-telling is a dominant strategy for each agent**

The Mechanism…

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, \ldots, v_n)$;

---

1. Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
2. For each set $\mathcal{C} \in \mathbb{C}$,
3. ⌊ Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \texttt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
4. For each agent $i \in \mathcal{A}$,
5. | For each set $\mathcal{C} \in \mathbb{C}$,
6. | | Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \texttt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $(= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
7. | ⌊ Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \texttt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $(= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
8. | Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
9. ⌊ Define $p_i^{\xi}(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Allocated goods are considered only

«Bonus and Compensation», by Nisan and Ronen (2001)

❖ **Truth-telling is a dominant strategy for each agent**

Coalitional Games

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\mathcal{G} = \langle N, \varphi \rangle, \; \varphi : 2^N \mapsto \mathbb{R}$$

- **Solution Concepts** characterize outcomes in terms of
  - Fairness
  - Stability

Coalitional Games: Shapley Value

$$\phi_i(\mathcal{G}) = \sum_{C \subseteq N} \frac{(|N| - |C|)!(|C| - 1)!}{|N|!}(\varphi(C) - \varphi(C \setminus \{i\}))$$

- **Solution Concepts** characterize outcomes in terms of
  - Fairness
  - Stability

Relevant Properties of the Shapley Value

(I) $\sum_{i \in N} \phi_i(\mathcal{G}) = \varphi(N)$;

(II) If φ is *supermodular* (resp., *submodular*), then $\sum_{i \in R} \phi_i(\mathcal{G}) \geq \varphi(R)$ (resp., $\sum_{i \in R} \phi_i(\mathcal{G}) \leq \varphi(R)$), for each coalition $R \subseteq N$.

(III) If $\mathcal{G}' = \langle N, \varphi' \rangle$ is a game such that $\varphi'(R) \geq \varphi(R)$, for each $R \subseteq N$, then $\phi_i(\mathcal{G}') \geq \phi_i(\mathcal{G})$, for each agent $i \in N$.

Core Allocation

$$\varphi(R \cup T) + \varphi(R \cap T) \geq \varphi(R) + \varphi(T) \text{ (resp., } \varphi(R \cup T) + \varphi(R \cap T) \leq \varphi(R) + \varphi(T))$$

The Mechanism

$$\mathcal{G} = \langle N, \varphi \rangle,\ \varphi : 2^N \mapsto \mathbb{R}$$

- $\varphi(C)$ is the *contribution* of the coalition **w.r.t.** $\left\{ \begin{array}{l} \textbf{selected products} \\ \quad\quad \textit{and} \\ \textbf{verified values} \end{array} \right.$

The Mechanism

$\mathcal{G} = \langle N, \varphi \rangle, \ \varphi : 2^N \mapsto \mathbb{R}$

- $\varphi(C)$ is the *contribution* of the coalition **w.r.t.** $\left\{ \begin{array}{l} \textbf{selected products} \\ \textit{and} \\ \textbf{verified values} \end{array} \right.$

⬇

Best possible allocation,
assuming that agents in C are the only ones in the game

The Mechanism

$$\mathcal{G} = \langle N, \varphi \rangle,\ \varphi : 2^N \mapsto \mathbb{R}$$

- $\varphi(C)$ is the *contribution* of the coalition **w.r.t.** $\left\{ \begin{array}{l} \textbf{selected products} \\ \textit{and} \\ \textbf{verified values } (\pi) \end{array} \right.$

→ Each agent gets the Shapley value $\phi_i(\mathcal{G})$

The Mechanism

$$\mathcal{G} = \langle N, \varphi \rangle,\ \varphi : 2^N \mapsto \mathbb{R}$$

- $\varphi(C)$ is the *contribution* of the coalition **w.r.t.** { **selected products** *and* **verified values (π)** }

Each agent gets the Shapley value $\phi_i(\mathcal{G})$

Properties

The resulting mechanism is «fair» and «buget balanced»

The Mechanism

$\mathcal{G} = \langle N, \varphi \rangle$, $\varphi : 2^N \mapsto \mathbb{R}$

- $\varphi(C)$ is the *contribution* of the coalition **w.r.t.**
  - **selected products** *and* **verified values (π)**

Each agent gets the Shapley value $\phi_i(\mathcal{G})$

Properties

The resulting mechanism is «fair» and «buget balanced»

$$\sum_{i \in N} \phi_i(\mathcal{G}) = \varphi(N)$$

The Mechanism

$$\mathcal{G} = \langle N, \varphi \rangle,\ \varphi : 2^N \mapsto \mathbb{R}$$

- $\varphi(C)$ is the *contribution* of the coalition **w.r.t.** { **selected products** *and* **verified values (π)** }

Each agent gets the Shapley value $\phi_i(\mathcal{G})$

Properties

The resulting mechanism is «fair» and «buget balanced»

The game is supermodular;
so the Shapley value is stable

Further Observations for Fairness

- Let π be an optimal allocation
- Let π' be an allocation

Further Observations for Fairness

- Let π be an optimal allocation
- Let π' be an allocation

$$\mathcal{G} \longleftarrow \pi$$
$$\downarrow$$
$$\varphi(C)$$

(best allocation for the coalition with products in π)

As π is optimal, then $\varphi(C)$ is in fact optimal even by considering all possible products as available

$$\pi' \rightarrow \mathcal{G}' \rightarrow \varphi'(C)$$

$$\varphi(C) \geq \varphi'(C)$$

Further Observations for Fairness

- Let π be an optimal allocation
- Let π' be an allocation

$$\mathcal{G} \leftarrow \pi$$

$$\downarrow$$

$$\varphi(C)$$

(best allocation for the coalition with products in π)

As π is optimal, then $\varphi(C)$ is in fact optimal even by considering all possible products as available

$$\pi' \rightarrow \mathcal{G}' \rightarrow \varphi'(C)$$

$$\varphi(C) \geq \varphi'(C)$$

By the monotonicity of the Shapley value, $\phi_i \geq \phi_i'$

Further Observations for Fairness

- Let π be an optimal allocation
- Let π' be an allocation

$$\pi \geq \pi'$$

❖ **Optimal allocations are always preferred by ALL agents**
❖ **There is no difference between two different optimal allocations**

Further Observations for Fairness

- Let π be an optimal allocation
- Let π' be an allocation

$$\pi \geq \pi'$$

❖ **Optimal allocations are always preferred by ALL agents**
❖ **There is no difference between two different optimal allocations**

Efficiency  **Fairness**

Other Solution Concepts?

Other Solution Concepts?

- How fairness/stability can be measured?

$$e(S, x) = v(S) - x(S)$$

- The excess is a measure of the dissatisfaction of S

Other Solution Concepts?

- How fairness/stability can be measured?

$$e(S, x) = v(S) - x(S)$$

- The excess is a measure of the dissatisfaction of S

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1,2\})) = v(\{1,3\}) = v(\{2,3\}) = 1$$
$$v(\{1,2,3\}) = 3$$

Excess...

- How fairness/stability can be measured?

$$e(S, x) = v(S) - x(S)$$

- The excess is a measure of the dissatisfaction of S

$x = (0, 0, 3)$ → $e(\{1, 2\}, x) = v(\{1, 2\}) - (x_1 + x_2) = 1 - 0 = 1$

$x = (1, 2, 0)$ → $e(\{1, 2\}, x) = v(\{1, 2\}) - (x_1 + x_2) = 1 - 3 = -2$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

…and the Nucleolus

- Arrange excess values in non-increasing order

...and the Nucleolus

- Arrange excess values in non-increasing order

$$x = (1, 2, 0) \qquad \theta(x) = (0, 0, -1, -1, -2, -2)$$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

Core Imputation

$x = (1, 2, 0)$ $\theta(x) = (0, 0, -1, -1, -2, -2)$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

$x^* = (1, 1, 1)$ $\theta(x^*) = (-1, -1, -1, -1, -1, -1)$

$x = (1, 2, 0)$ $\theta(x) = (0, 0, -1, -1, -2, -2)$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

$x^* = (1, 1, 1)$ $\theta(x^*) = (-1, -1, -1, -1, -1, -1)$

$x = (1, 2, 0)$ $\theta(x) = (0, 0, -1, -1, -2, -2)$

$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$
$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$
$v(\{1, 2, 3\}) = 3$

...and the Nucleolus

- Arrange excess values in non-increasing order

Definition [Schmeidler]

The *nucleolus* $\mathcal{N}(\mathcal{G})$ of a game $\mathcal{G}$ is the set
$\mathcal{N}(\mathcal{G}) = \{x \in X(\mathcal{G}) \mid \nexists y \in X(\mathcal{G}) \text{ s.t. } \theta(y) \prec \theta(x)\}$

$x^* = (1, 1, 1)$ $\qquad$ $\theta(x^*) = (-1, -1, -1, -1, -1, -1)$

$x = (1, 2, 0)$ $\qquad$ $\theta(x) = (0, 0, -1, -1, -2, -2)$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

Outline

- Game Theory
- Mechanism Design
- Mechanisms with Verification
- Mechanisms and Allocation Problems
- **Complexity Analysis**

Complexity Issues

- For many classes of «compact games» (e.g., *graph games*), the Shapley-value can be efficiently calculated
- Here, the problem emerges to be #P-complete

Complexity Issues

- For many classes of «compact games» (e.g., *graph games*), the Shapley-value can be efficiently calculated
- Here, the problem emerges to be #P-complete

- #P is the class the class of all functions that can be computed by *counting Turing machines* in polynomial time.
- A counting Turing machine is a standard nondeterministic Turing machine with an auxiliary output device that prints in binary notation the number of accepting computations induced by the input.
- Prototypical problem: to count the number of truth variable assignments that satisfy a Boolean formula.

Complexity Issues

- For many classes of «compact games» (e.g., *graph games*), the Shapley-value can be efficiently calculated
- Here, the problem emerges to be #P-complete

Reduction from the problem of counting the number of perfect matchings in certain bipartite graphs [Valiant, 1979]

- #P is the class the class of all functions that can be computed by *counting Turing machines* in polynomial time.
- A counting Turing machine is a standard nondeterministic Turing machine with an auxiliary output device that prints in binary notation the number of accepting computations induced by the input.
- Prototypical problem: to count the number of truth variable assignments that satisfy a Boolean formula.

Complexity Issues

- #P-complete
- However…

Probabilistic Computation

- #P-complete
- However…

Fully Polynomial-Time Randomized Approximation Scheme

- Always Efficient and Budget Balanced
- All other properties in expectation (with high probability)

Coupling of the algorithm with a sampling strategy for the coalitions by [Liben-Nowell,Sharp, Wexler, Woods; 2012]

Probabilistic Computation

Input: An allocation π for $\langle \mathcal{A}, G, \omega \rangle$, and a vector $\mathbf{w} \in \mathbf{D}$;
Assumption: A verifier $\mathbf{v}$ is available. Let $\mathbf{v}(\pi) = (v_1, ..., v_n)$;

Let $\mathbb{C}$ denote the set of all possible subsets of $\mathcal{A}$;
For each set $\mathcal{C} \in \mathbb{C}$,
⌊ Compute an optimal allocation $\pi_{\mathcal{C}}$ for $\langle \mathcal{C}, \texttt{img}(\pi), \omega \rangle$ w.r.t. $\mathbf{w}$;
For each agent $i \in \mathcal{A}$,
| For each set $\mathcal{C} \in \mathbb{C}$,
| | Let $\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) := \texttt{val}(\pi_{\mathcal{C}}, (v_i, \mathbf{w}_{-i}))$; $\quad (= v_i(\pi_{\mathcal{C}}) + \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C}}))$;
| ⌊ Let $\Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}) := \texttt{val}(\pi_{\mathcal{C} \setminus \{i\}}, \mathbf{w})$; $\quad (= \sum_{j \in \mathcal{C} \setminus \{i\}} w_j(\pi_{\mathcal{C} \setminus \{i\}}))$;
| Let $\xi_i(\pi, \mathbf{w}) := \sum_{\mathcal{C} \in \mathbb{C}} \frac{(|\mathcal{A}| - |\mathcal{C}|)!(|\mathcal{C}|-1)!}{|\mathcal{A}|!} (\Delta^1_{\mathcal{C},i}(\pi, \mathbf{w}) - \Delta^2_{\mathcal{C},i}(\pi, \mathbf{w}))$;
⌊ Define $p_i^{\xi}(\pi, \mathbf{w}) := \xi_i(\pi, \mathbf{w}) - v_i(\pi)$;

Use sampling, rather than exaustive search.

Coupling of the algorithm with a sampling strategy for the coalitions by [Liben-Nowell,Sharp, Wexler, Woods; 2012]

Back to Exact Computation: Islands of Tractability

- Can we find classes of instances for «allocation games» over which the Shapley value can be efficiently computed?

Back to Exact Computation: Islands of Tractability

- Can we find classes of instances for «allocation games» over which the Shapley value can be efficiently computed?

Restrictions [G., Lupia and Scarcello; 2015]

#P-complete, even for k=2

- Utility functions
  - Values taken from specific domains
  - For instance, use *k* values at most

Back to Exact Computation: Islands of Tractability

- Can we find classes of instances for «allocation games» over which the Shapley value can be efficiently computed?

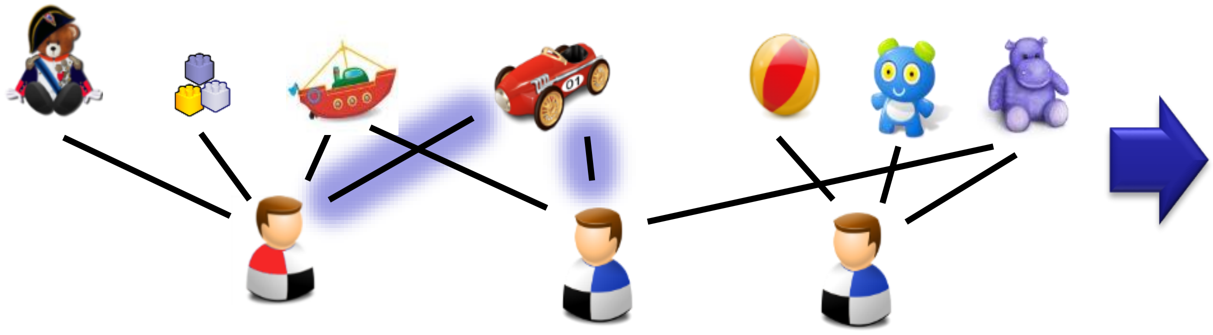
Restrictions

#P-complete, even for k=2

- Utility functions
  - Values taken from specific domains
  - For instance, use *k* values at most

- Structural restrictions…

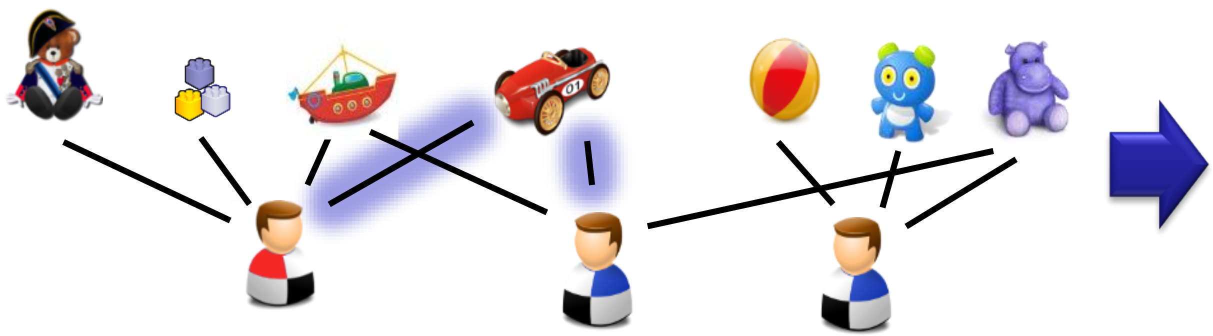
Bounded Sharing Degree



Sharing degree = 2

- Sharing degree
  - Maximum number of agents competing for the same good

Bounded Sharing Degree



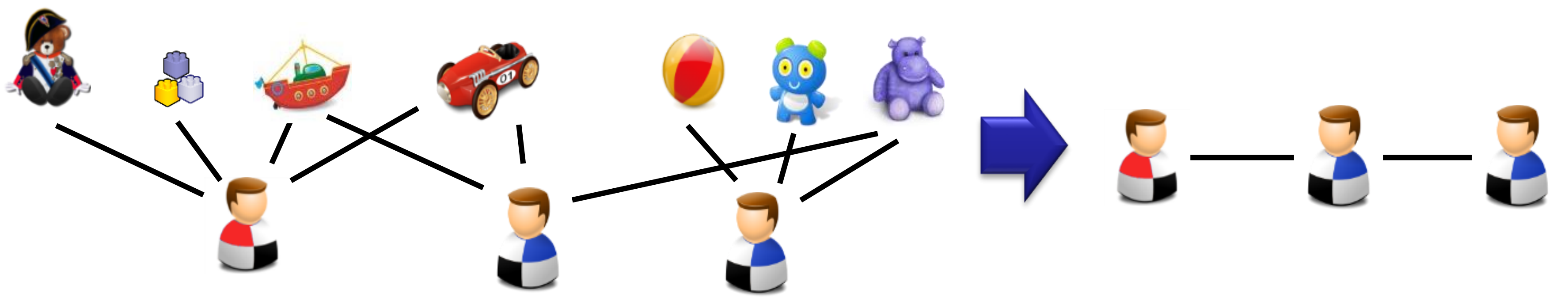
Sharing degree = 2

- Sharing degree
  - Maximum number of agents competing for the same good

The Shapley value can be computed in polynomial time whenever the sharing degree is 2 at most.

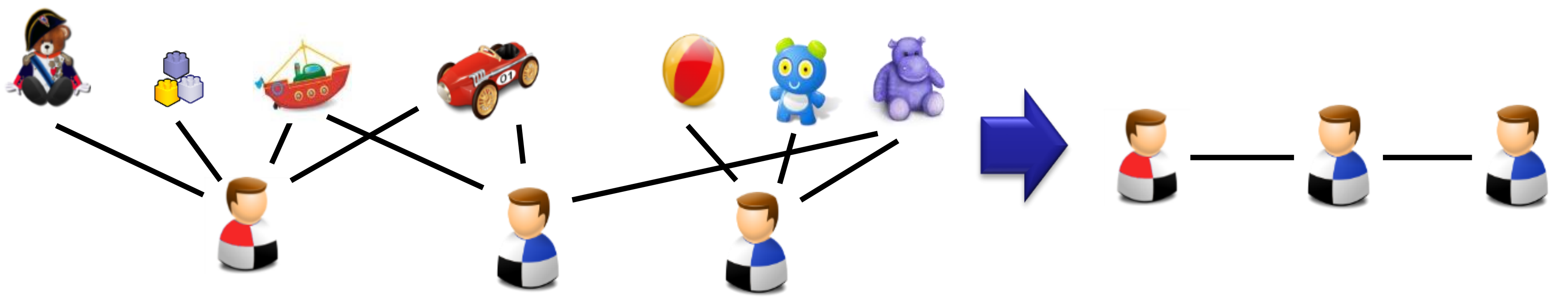
Bounded Interactions

Bounded Interactions



- Interaction graph
  - There is an edge between any pair of agents competing for the same good

Bounded Interactions

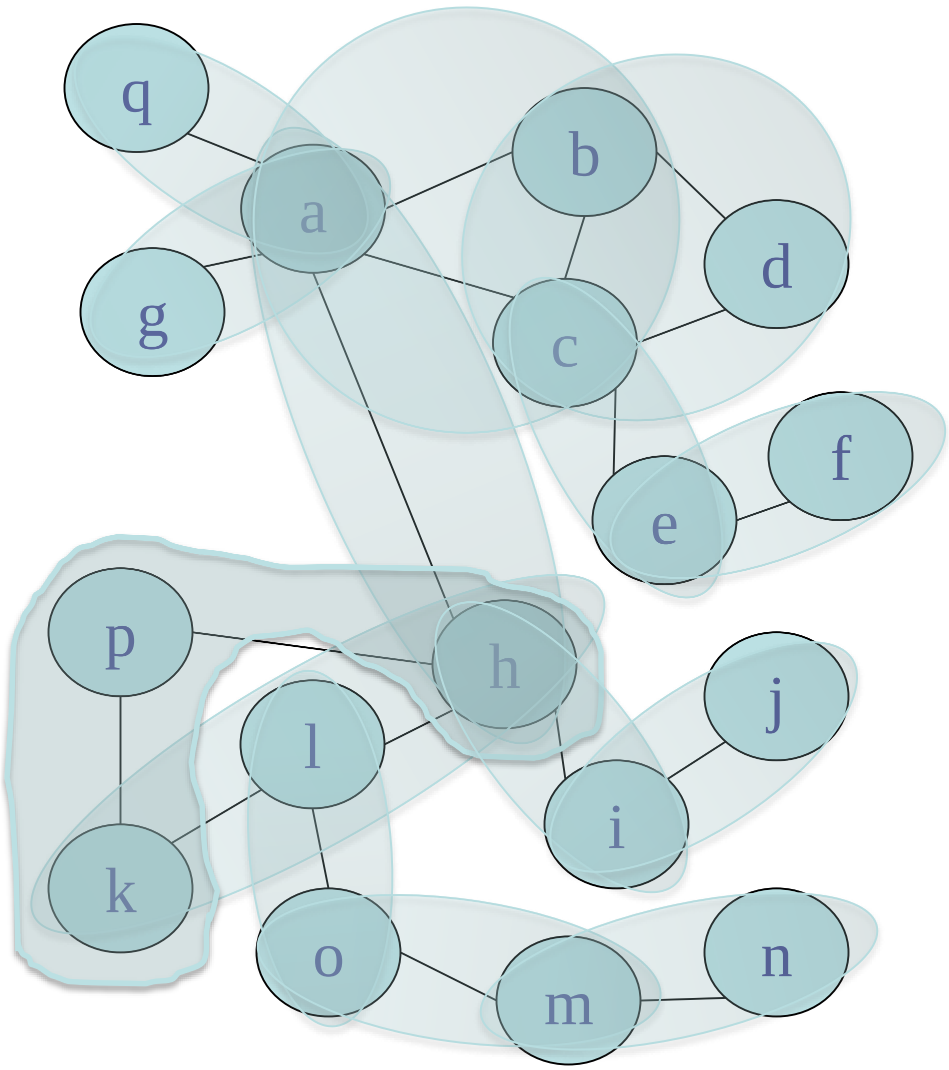


- Interaction graph
  - There is an edge between any pair of agents competing for the same good

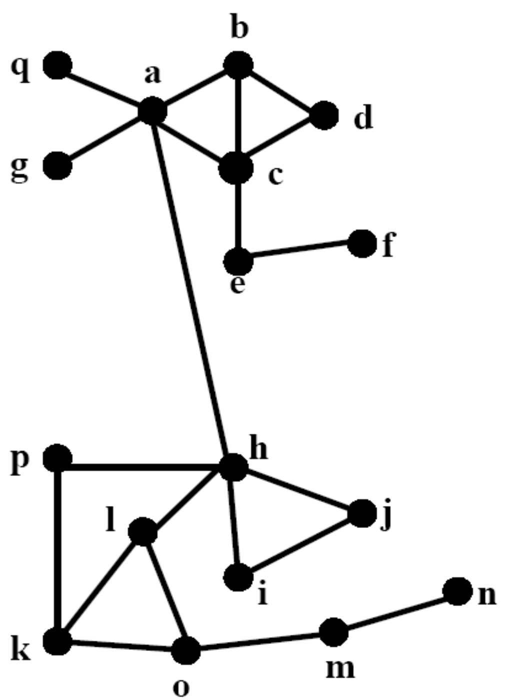
The Shapley value can be computed in polynomial time whenever the interaction graph is a tree.

or, more generally, if it has bounded treewidth

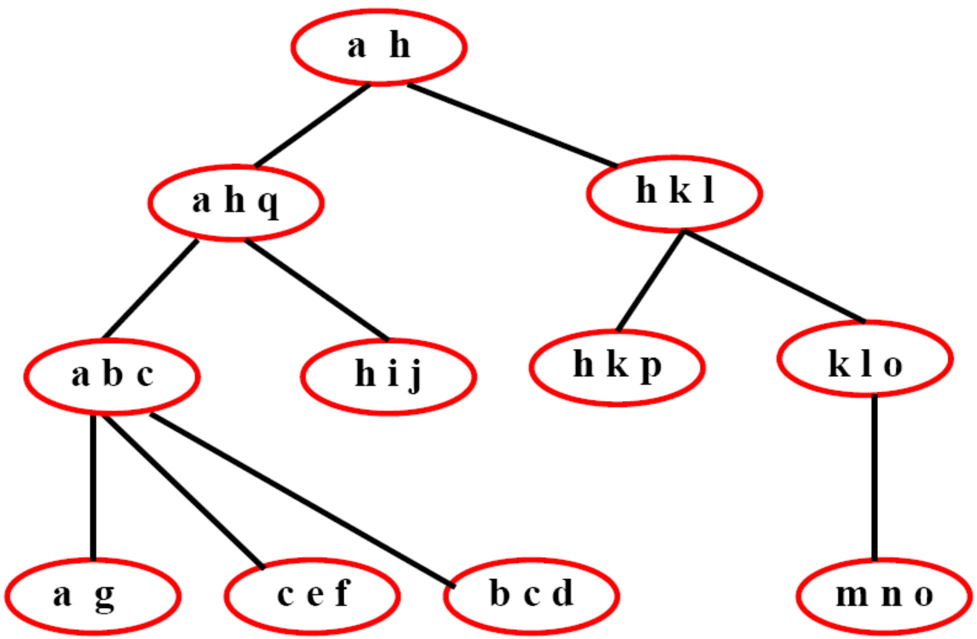
Tree Decompositions [Robertson & Seymour ‘86]



Tree Decompositions [Robertson & Seymour '86]

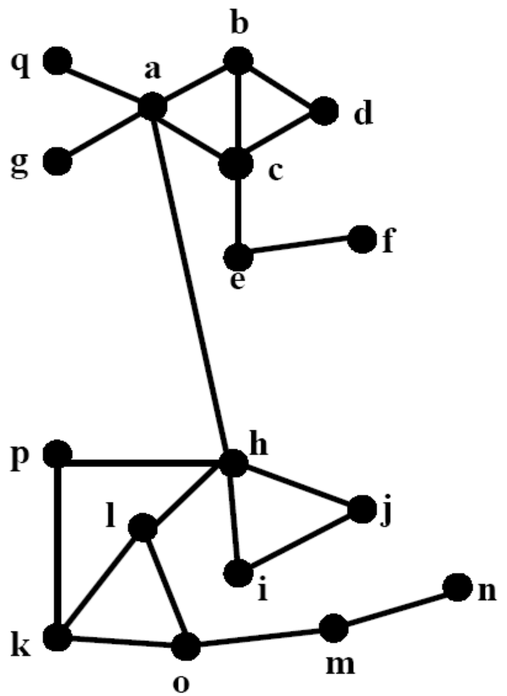


Graph G

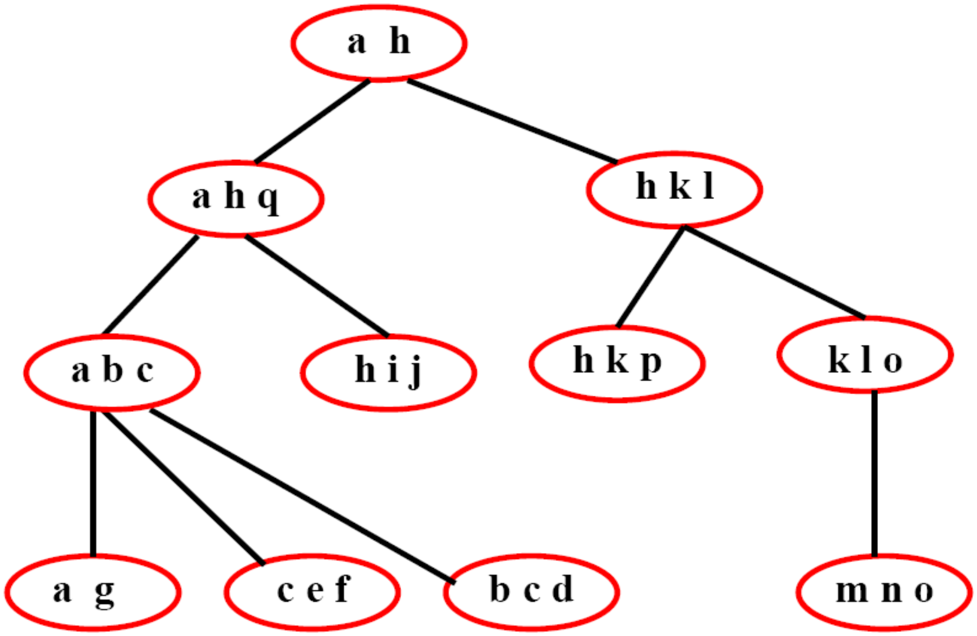


Tree decomposition of width 2 of G

Tree Decompositions [Robertson & Seymour '86]



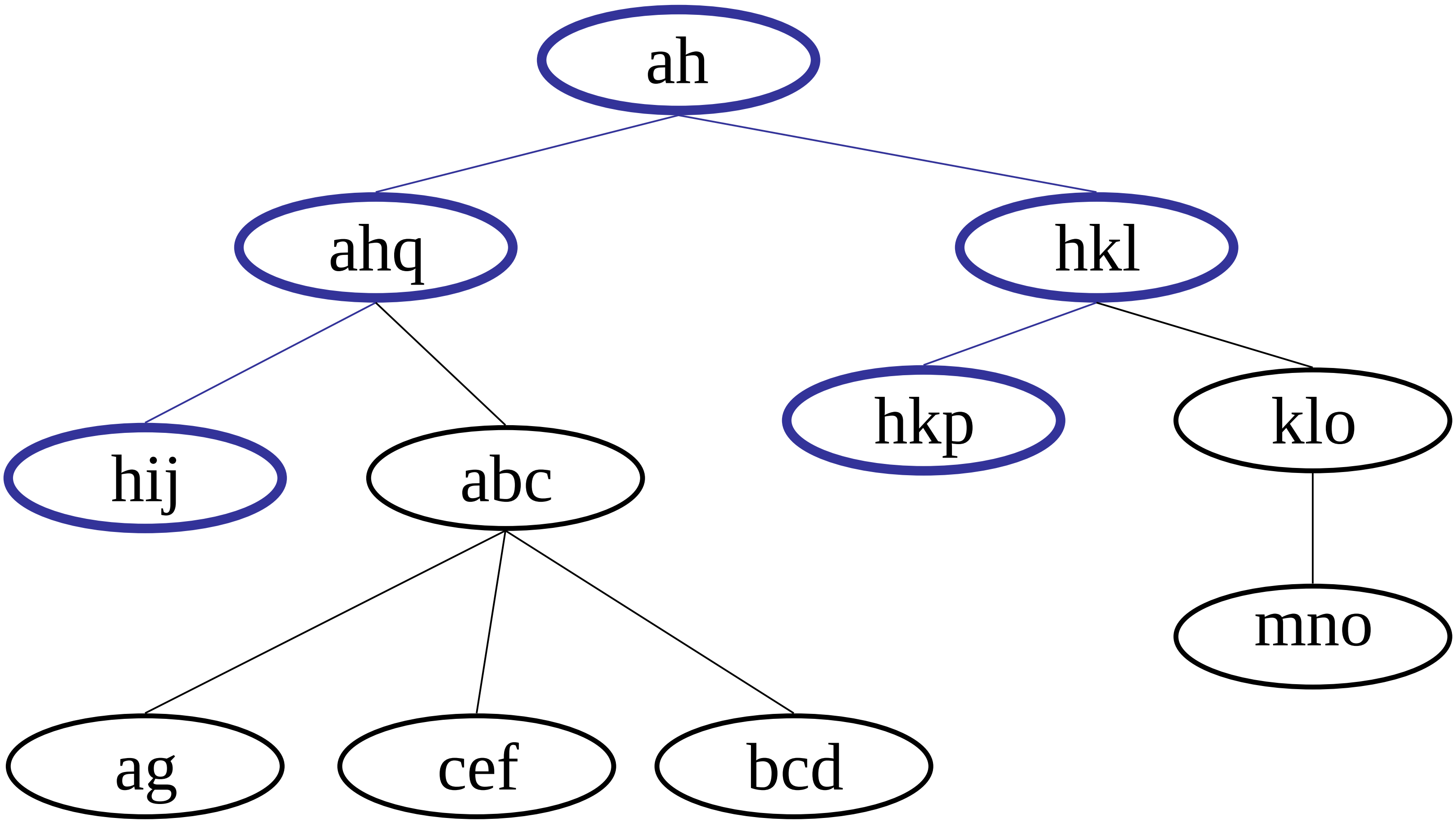
Graph G



Tree decomposition of width 2 of G

- Every edge realized in some bag
- Connectedness condition

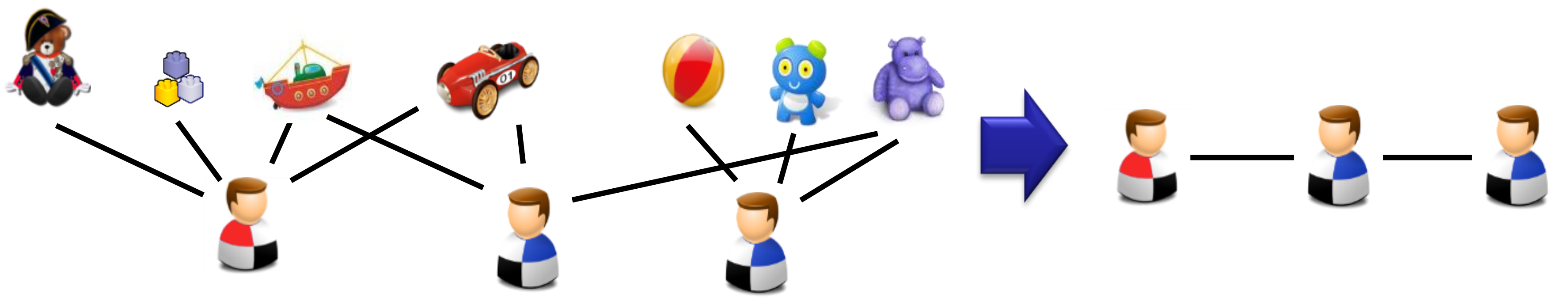
Connectedness condition for *h*



Properties of Treewidth

- tw(acyclic graph)=1
- tw(cycle) = 2
- tw(G+v) $\leq$ tw(G)+1
- tw(G+e) $\leq$ tw(G)+1
- tw(K_n) = n-1
- tw is fixed-parameter tractable (parameter: treewidth)

Bounded Interactions



- Interaction graph
  - There is an edge between any pair of agents competing for the same good

The Shapley value can be computed in polynomial time whenever the interaction graph is a tree.

or, more generally, if it has bounded treewidth

Proof Idea: Ingredient 1

$$\phi_i(\mathcal{G}_\mathcal{A}) = \sum_{h=0}^{n-1} \frac{h!(n\text{-}h\text{-}1)!}{n!} \beta_i(\mathcal{G}_\mathcal{A}, h)$$, *where*

$$\beta_i(\mathcal{G}_\mathcal{A}, h) = \sum_{C \subseteq N \setminus \{i\}, |C|=h} (v(C \cup \{i\}) - v(C))$$

- list the values in increasing order: $w_1, \ldots, w_m$

$$\beta_i(\mathcal{G}_\mathcal{A}, h) = w_1 \times \#\mathrm{col}_1^i(\mathcal{G}_\mathcal{A}, h) + \sum_{\ell=2}^{m} w_\ell \times \left(\#\mathrm{col}_\ell^i(\mathcal{G}_\mathcal{A}, h) - \#\mathrm{col}_{\ell-1}^i(\mathcal{G}_\mathcal{A}, h)\right)$$

$\#\mathrm{col}_\ell^i(\mathcal{G}_\mathcal{A}, h)$ *is the number of coalitions* C *such that* $|C| = h$ *and*

$$v_\mathcal{A}(C \cup \{i\}) - v_\mathcal{A}(C) \geq w_\ell$$

Proof Idea: Ingredient 2

$\#\mathrm{col}^i_\ell(\mathcal{G}_A, h)$ *is the number of coalitions* C *such that* $|C| = h$ *and*

$$v_{\mathcal{A}}(C \cup \{i\}) - v_{\mathcal{A}}(C) \geq w_\ell$$

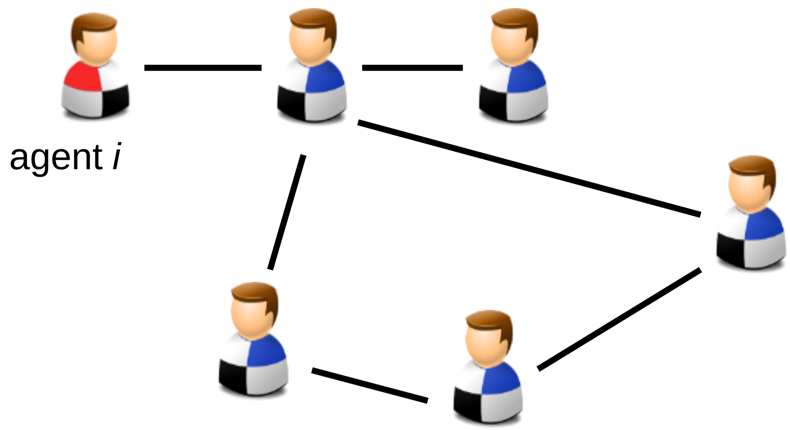
- The marginal contribution can be characterized via the existence of an allocation with certain properties

Proof Idea: Ingredient 2

$\#\mathrm{col}_\ell^i(\mathcal{G}_A, h)$ *is the number of coalitions* C *such that* $|C| = h$ *and*

$$v_{\mathcal{A}}(C \cup \{i\}) - v_{\mathcal{A}}(C) \geq w_\ell$$

- The marginal contribution can be characterized via the existence of an allocation with certain properties

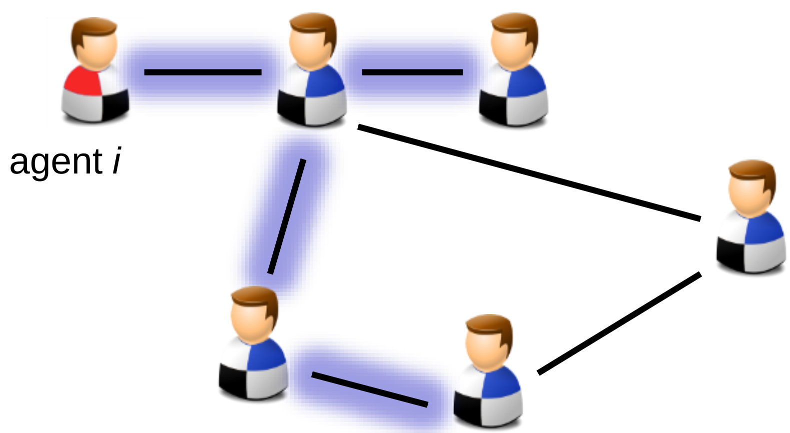


Proof Idea: Ingredient 2

$\#\mathrm{col}^i_\ell(\mathcal{G}_A, h)$ *is the number of coalitions* C *such that* $|C| = h$ *and*

$$v_{\mathcal{A}}(C \cup \{i\}) - v_{\mathcal{A}}(C) \geq w_\ell$$

- The marginal contribution can be characterized via the existence of an allocation with certain properties



interaction graph

✓ *restricted w.r.t.* w_ℓ

$$G_\ell$$

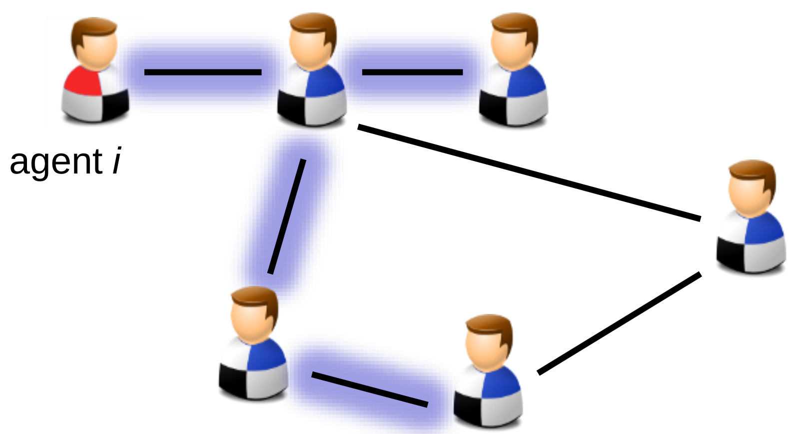
Proof Idea: Ingredient 2

$\#\mathrm{col}_\ell^i(\mathcal{G}_A, h)$ *is the number of coalitions* C *such that* $|C| = h$ *and*

$$v_{\mathcal{A}}(C \cup \{i\}) - v_{\mathcal{A}}(C) \geq w_\ell$$

$$\Updownarrow$$

there is an allocation in the scenario induced over G_ℓ *where each agent gets a good with value at least* w_ℓ



interaction graph

✓ *restricted w.r.t.* w_ℓ

⇓

G_ℓ

Proof Idea: Ingredient 2

$\#\mathrm{col}^i_\ell(\mathcal{G}_A, h)$ *is the number of coalitions* C *such that* $|C| = h$ *and*

$$v_{\mathcal{A}}(C \cup \{i\}) - v_{\mathcal{A}}(C) \geq w_\ell$$

$$\|$$

there is an allocation in the scenario induced over G_ℓ *where each agent gets a good with value at least* w_ℓ

- Keep only goods with the desired value
- Focus on the induced scenario

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

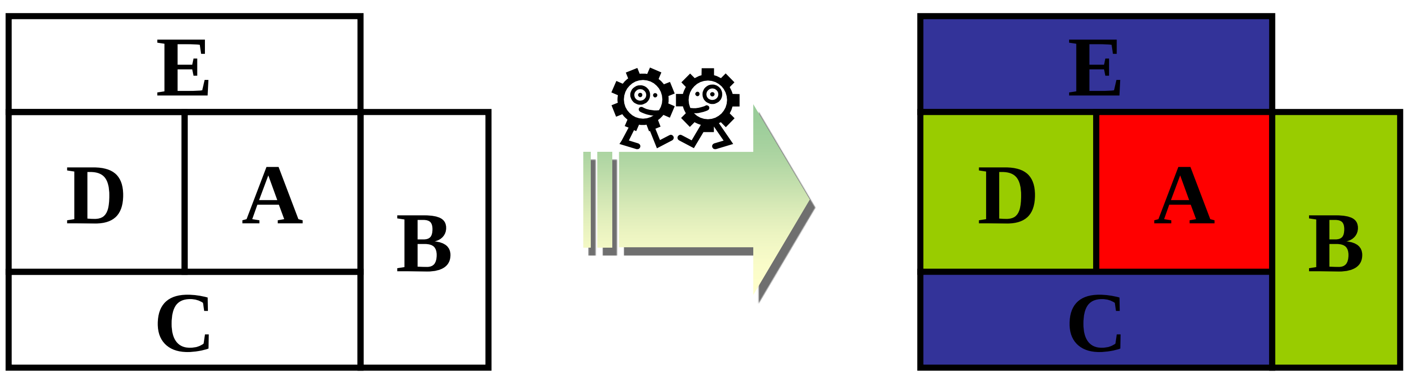
Proof Idea: Ingredient 3

Encode as a CSP

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

CSPs: Informal Definition

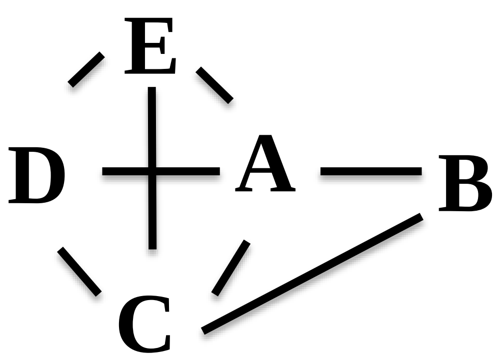
- **Variables:**
  - A, B, C, D, and E
- **Domain:**
  - RGB = {red, green, blue}
- **Constraints:**
  - A≠B, A≠C,A ≠ E, A ≠ D, B ≠ C, C ≠ D, D ≠ E



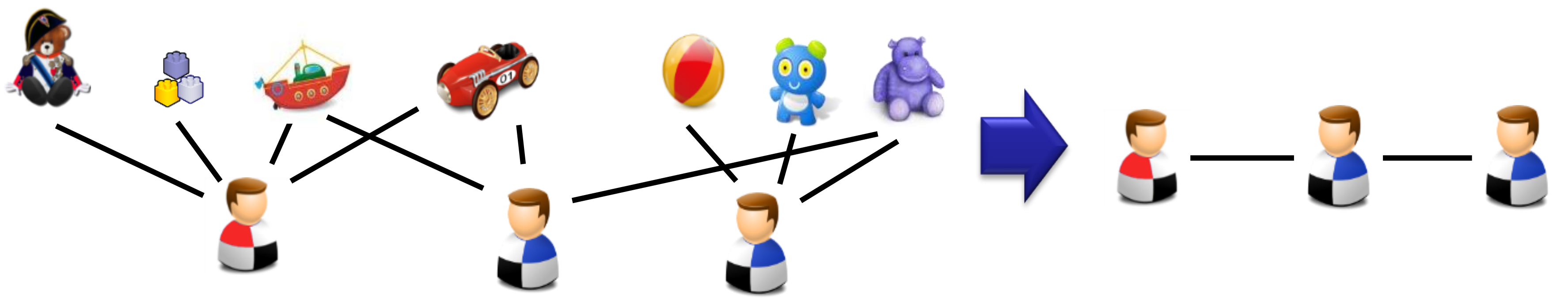
CSPs: Informal Definition

- **Variables:**
  - A, B, C, D, and E
- **Domain:**
  - **D**(A) = **D**(B) = **D**(C) = **D**(D) = **D**(E) = {red, green, blue}
- **Constraints:**
  - A≠B; A≠C; A ≠ E; A ≠ D; B ≠ C; C ≠ D; D ≠ E

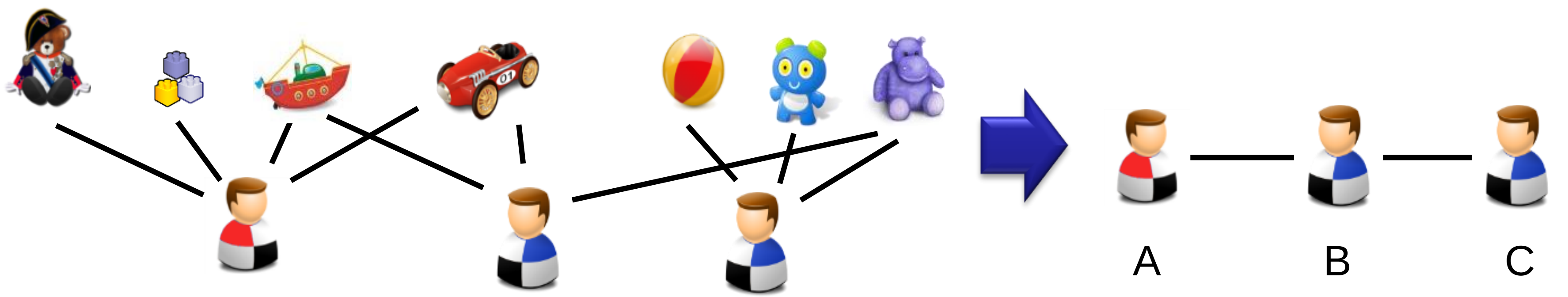
primal graph



Example Encoding



Example Encoding

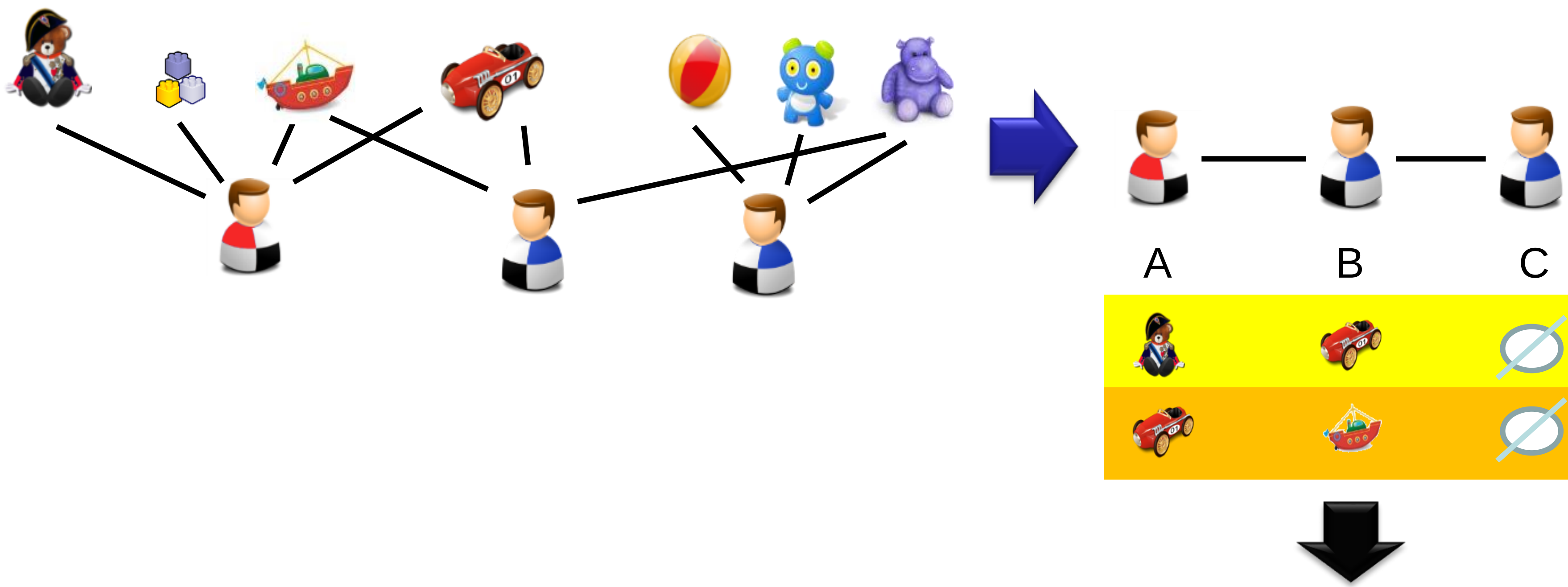


- **Variables:**
  - *Agent* A, *agent* B, and *agent* C + variables IN_A, IN_B, IN_C

boolean: {true, false}

- **Domain:**
  - **D**(A) =  ∅
  - **D**(B) =  ∅
  - **D**(C) =  ∅
- **Constraints:**
  - A≠B; B≠C; X=∅ if, and only if, IN_X=false

Example Encoding



$IN_A = IN_B = \text{true}$

$IN_C = \text{false}$

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Proof Idea: Ingredient 3

Encode as a CSP

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Proof Idea: Ingredient 3

in «Tractability: Practical Approaches to hard Problems»
[Gottlob, Greco, Scarcello, 2013]

Structural tractability results for CSPs

- Decision problems
- Computation Problems
- Counting?

Encode as a CSP

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Proof Idea: Ingredient 3

Structural tractability results for CSPs

- ✓ Solutions projected over a set *W* of output variables
- ✓ Variables not in *W* are auxiliary ones

- Decision problems
- Computation Problems
- Counting?

Theorem (cf. [Pichler and Skritek, 2013; Greco and Scarcello, 2014]). *Counting the number of substitutions in $\Theta(\mathcal{I}, \mathcal{W})$ is feasible in polynomial time, on classes of CSP instances $\mathcal{I}$ such that the treewidth of $G(\mathcal{I})$ is bounded by a constant, and the size of the domain of each variable not in $\mathcal{W}$ is bounded by some constant, too.*

Proof Idea: Ingredient 3

Structural tractability results for CSPs

- ✓ Solutions projected over a set *W* of output variables
- ✓ Variables not in *W* are auxiliary ones

- Decision problems
- Computation Problems
- Counting?

Theorem (cf. [Pichler and Skritek, 2013; Greco and Scarcello, 2014]). *Counting the number of substitutions in* $\Theta(\mathcal{I}, \mathcal{W})$ *is feasible in polynomial time, on classes of CSP instances* $\mathcal{I}$ *such that the treewidth of* $G(\mathcal{I})$ *is bounded by a constant, and the size of the domain of each variable not in* $\mathcal{W}$ *is bounded by some constant, too.*

For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

Proof Idea: Ingredient 3

Structural tractability results for CSPs

- ✓ Solutions projected over a set *W* of output variables
- ✓ Variables not in *W* are auxiliary ones

- Decision problems
- Computation Problems
- Counting?

Theorem (cf. [Pichler and Skritek, 2013; Greco and Scarcello, 2014]). *Counting the number of substitutions in $\Theta(\mathcal{I}, \mathcal{W})$ is feasible in polynomial time, on classes of CSP instances $\mathcal{I}$ such that the treewidth of $G(\mathcal{I})$ is bounded by a constant, and the size of the domain of each variable not in $\mathcal{W}$ is bounded by some constant, too.*

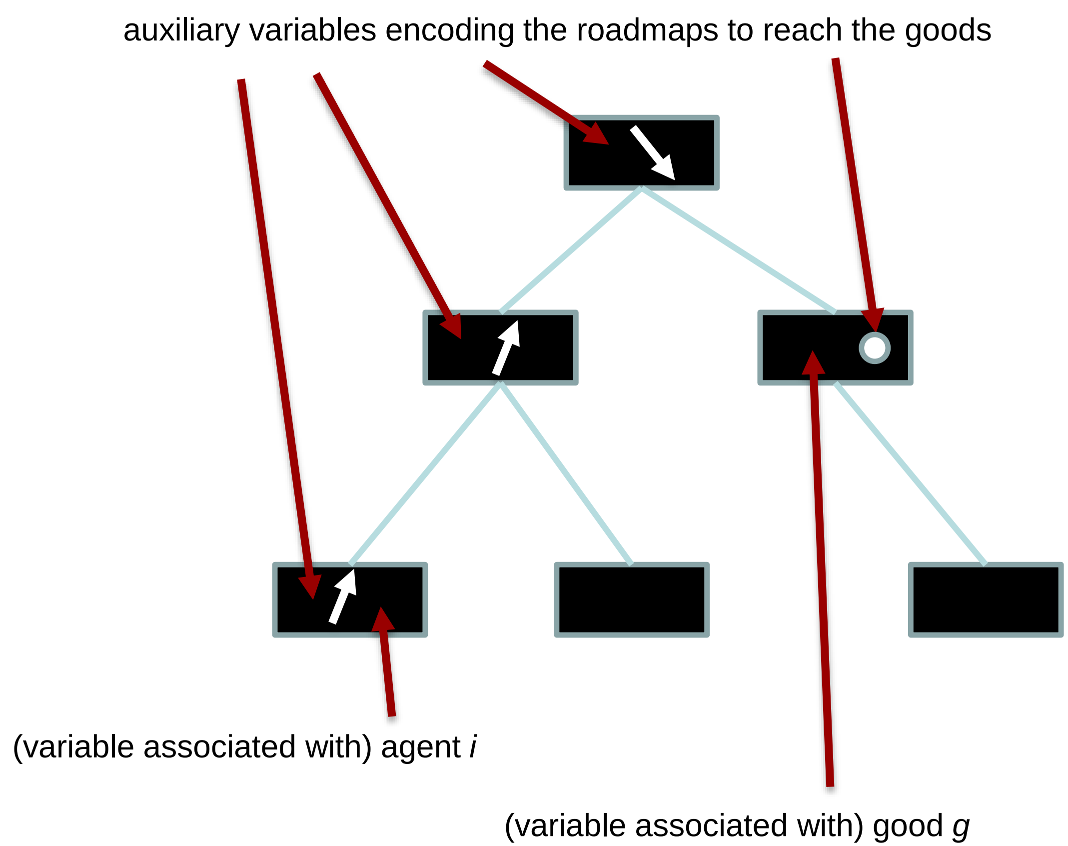
For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

Proof Idea: Ingredient 3

- Usually,
  - Build the CSP
  - Compute a decomposition
  - Use structural tractability results
- Here
  - Compute a decomposition
  - Build the CSP based on the decomposition
  - Recompute the decomposition
  - Use structural tractability results

For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

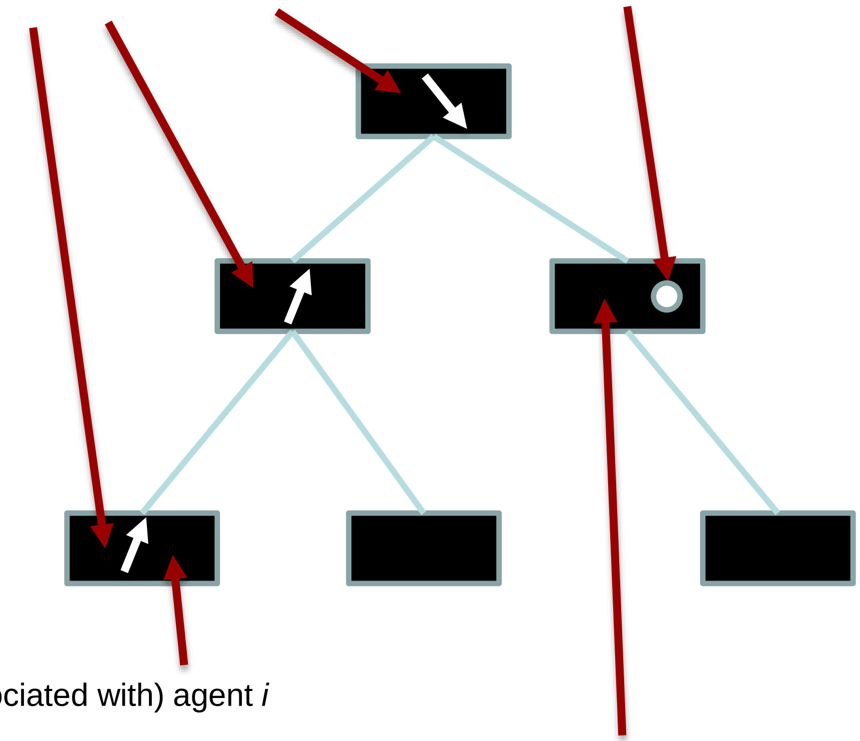
Proof Idea: Ingredient 3



For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

Proof Idea: Ingredient 3

W.l.o.g. the tree is binary.
Hence, a few «road signs» suffices



For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

Thank you!

For references, see the bibliography of *Mechanisms for Fair Allocation Problems* [G. and Scarcello; JAIR 2014]