Human review packet: HT26EFXChores

Generated from byte-pinned source excerpts, the expanded PaperInterface specifications, and hash-bound raw-source-to-Spec screening records.

Paper: HT26EFXChores
Paper id: HT26EFXChores
Source version: arXiv v2 (2026-07-09)
Source record: audit/paper_statement_map.json
Rows: 18
Generated: 2026-09-06
Source URL: official source

How to use this packet

For each source claim, compare the exact source excerpts with the expanded PaperInterface specification. Mark up this PDF or fill the blank reviewer box in the TeX source; return the annotated file for reconciliation into the paper-local human-review log.

Open the interactive dashboard

The interactive dashboard is an optional alternative to using this PDF; it presents the same review surface rather than an additional review requirement. After forking and cloning (or pulling) AppliedModelingLib, run the following from the repository root. The cache command is needed only the first time in a new clone.

```
cd <your-repository-checkout>
lake exe cache get
python3 scripts/review_dashboard.py --paper HT26EFXChores --serve
```

Open http://127.0.0.1:8765/ in a browser and keep that terminal running while reviewing. The dashboard records saved annotations in this paper's local reviewer-owned review record.

Contents

Material library prerequisites

AppliedModelingLib.FairDivision.EFXForChores

Source locator: cited publication, lines 260-275

Verbatim paper-source connection

```
\begin{definition}[EFX for chores]
An allocation $X=(X_1,\ldots,X_n)$ is \emph{EFX} if for every pair of agents $i,j$ we have either
\begin{itemize}
    \item $X_i=\emptyset$, or
    \item for every item $g\in X_i$, $c_i(X_i\setminus\{g\})\le c_i(X_j).$
\end{itemize}
Equivalently, if
\[
    \tau_i(X):=\tau_i(X_i):=
    \begin{cases}
        0, & X_i=\emptyset,\\
        c_i(X_i)-\min_{g\in X_i}c_i(g), & X_i\ne\emptyset,
    \end{cases}
\]
then $X$ is EFX if and only if $\tau_i(X)\le c_i(X_j)$ for all $i,j$.
\end{definition}

[Next verbatim source excerpt]

We will use ``item'' and ``chore'' interchangeably in this paper to refer to an element in $M$.
Each agent $i$'s cost function $c_i:2^M\to\mathbb{R}_{\geq0}$ is additive:
$$c_i(S)=\sum_{g\in S}c_i(\{g\}).$$
We write $c_i(g_j)$ in short for $c_i(\{g_j\})$ for notation simplicity.
We will use ``cost function'' and ``disutility function'' interchangeably in this paper.
```

Lean-expanded library semantic target

```
type:
{Agent : Type u_1} →
  {Item : Type u_2} →
    [DecidableEq Item] →
      (Agent → AppliedModelingLib.FairDivision.Bundle Item → ℝ) →
        AppliedModelingLib.FairDivision.Allocation Agent Item → Prop

value:
fun {Agent : Type u_1} {Item : Type u_2} [DecidableEq Item]
    (cost : Agent → AppliedModelingLib.FairDivision.Bundle Item → ℝ)
    (allocation : AppliedModelingLib.FairDivision.Allocation Agent Item) =>
  ∀ (i j : Agent), allocation i = ∅ ∨ ∀ item ∈ allocation i, cost i (allocation i \ {item}) ≤ cost i (allocation j)
```

Exact Lean library declaration

```
/-- Envy-freeness up to any item for chores: after removing any item from an
agent's own bundle, that agent incurs no more cost than for another bundle. -/
def EFXForChores [DecidableEq Item] (cost : Agent → Bundle Item → ℝ)
    (allocation : Allocation Agent Item) : Prop :=
  ∀ i j, allocation i = ∅ ∨
    ∀ item ∈ allocation i, cost i (allocation i \ {item}) ≤ cost i (allocation j)
```

Recorded source-to-library screening **Verdict:** matches

Reason: The target states that every agent pair either has an empty own bundle or satisfies the source all-item removal inequality.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.FairDivision.EnvyFreeForChores

Source locator: cited publication, lines 253-255

Verbatim paper-source connection

```
\begin{definition}[EF for chores]
An allocation $X=(X_1,\ldots,X_n)$ is \emph{envy-free} if for every pair of agents $i,j$ we have $c_i(X_i)\leq c_i(X_j)$.
\end{definition}
```

Lean-expanded library semantic target

```
type:
{Agent : Type u_1} →
  {Item : Type u_2} →
    (Agent → AppliedModelingLib.FairDivision.Bundle Item → ℝ) →
      AppliedModelingLib.FairDivision.Allocation Agent Item → Prop

value:
fun {Agent : Type u_1} {Item : Type u_2} (cost : Agent → AppliedModelingLib.FairDivision.Bundle Item → ℝ)
    (allocation : AppliedModelingLib.FairDivision.Allocation Agent Item) =>
  ∀ (i j : Agent), cost i (allocation i) ≤ cost i (allocation j)
```

Exact Lean library declaration

```
/-- An allocation is envy-free for chores when each agent's own cost is no
greater than their cost for any other agent's bundle. -/
def EnvyFreeForChores (cost : Agent → Bundle Item → ℝ)
    (allocation : Allocation Agent Item) : Prop :=
  ∀ i j, cost i (allocation i) ≤ cost i (allocation j)
```

Recorded source-to-library screening **Verdict:** matches

Reason: The target states exactly that every agent’s own bundle cost is at most that agent’s cost for every other bundle.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.FairDivision.ParetoOptimalForChores

Source locator: cited publication, lines 280-287

Verbatim paper-source connection

```
\begin{definition}[Pareto-Optimality]
    An allocation $Y=(Y_1,\ldots,Y_n)$ \emph{Pareto-dominates} an allocation $X=(X_1,\ldots,X_n)$ if
    \begin{itemize}
        \item $c_i(Y_i)\leq c_i(X_i)$ holds for all $i\in\{1,\ldots,n\}$, and
        \item there exists $i\in\{1,\ldots,n\},\;$ $c_i(Y_i)<c_i(X_i)$.
    \end{itemize}
    An allocation $X=(X_1,\ldots,X_n)$ is \emph{Pareto-optimal} if it is not Pareto-dominated by any allocation.
\end{definition}
```

Lean-expanded library semantic target

```
type:
{Agent : Type u_1} →
  {Item : Type u_2} →
    [DecidableEq Item] →
      (Agent → AppliedModelingLib.FairDivision.Bundle Item → ℝ) →
        Finset Item → AppliedModelingLib.FairDivision.Allocation Agent Item → Prop

value:
fun {Agent : Type u_1} {Item : Type u_2} [DecidableEq Item]
    (cost : Agent → AppliedModelingLib.FairDivision.Bundle Item → ℝ) (chores : Finset Item)
    (allocation : AppliedModelingLib.FairDivision.Allocation Agent Item) =>
  AppliedModelingLib.FairDivision.IsAllocationOf allocation chores ∧
    ¬∃ (improvement : AppliedModelingLib.FairDivision.Allocation Agent Item),
        AppliedModelingLib.FairDivision.IsAllocationOf improvement chores ∧
          AppliedModelingLib.FairDivision.ParetoDominatesForChores cost allocation improvement
```

Exact Lean library declaration

```
/-- A complete chore allocation is Pareto-optimal if no other complete
allocation Pareto-dominates it in the cost direction. -/
def ParetoOptimalForChores [DecidableEq Item] (cost : Agent → Bundle Item → ℝ)
    (chores : Finset Item) (allocation : Allocation Agent Item) : Prop :=
  IsAllocationOf allocation chores ∧
    ¬ ∃ improvement : Allocation Agent Item,
      IsAllocationOf improvement chores ∧
        ParetoDominatesForChores cost allocation improvement
```

Recorded source-to-library screening **Verdict:** matches

Reason: The surrounding preliminaries fix a chore set allocated to the agents, so the source's candidate and comparator allocations are complete allocations of that same chore set, as encoded by IsAllocationOf.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Paper-specific semantic prerequisites

HT26EFXChores.canonicalShortLongLabelsSourceDefinition

Paper-local declaration:
HT26EFXChores.canonicalShortLongLabelsSourceDefinition
Source locator: cited publication, lines 753-754

Verbatim paper-source connection

In a canonical allocation with $b>0$, agents receiving a items are called \emph{short} agents, and agents receiving $a+1$ items are called
↪ \emph{long} agents.
For $b=0$ and $a>0$, every agent in a canonical is both long and short.

[Next verbatim source excerpt]

We will assume $r>2$ in the remaining part of this section.
For a chore g, let
\[
S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom g is small. We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in M_2, we write (i,j) for an item small exactly for agents
i and j. Thus the items in M_2 form a multigraph on the vertex set
N, where an edge (i,j) is a chore small for precisely agents i and j.

[Next verbatim source excerpt]

For each agent i, let
\[
n_i=\left|\{g\in M_{01}:S(g)=\{i\}\}\right|
\]
be the number of M_{01} items uniquely small for i. Items in M_{01}
with $S(g)=\emptyset$ are all-large items.
Let $|M_{01}|=4a+b$ with $b\in\{0,1,2,3\}$.
As mentioned before, we will make sure each agent receives either a items or $a+1$ items.

Fix quotas $q_i\in\{a,a+1\}$ with $\sum_i q_i=|M_{01}|$.
We consider allocations of M_{01} with quotas $q=(q_1,q_2,q_3,q_4)$ in which each agent i first receives as many of her uniquely-small items as
↪ possible, up to quota q_i, and the remaining slots are filled with the remaining items.
We will show that all such allocations satisfy EFX.
It is also important that we have the freedom to choose which agents receive a items and which agents receive $a+1$ items.
This freedom enables us to append the allocation of M_2 after the allocation of M_{01}.

We begin by defining this type of allocations.

\begin{definition}
An allocation $P=(P_1,P_2,P_3,P_4)$ of M_{01} is \emph{canonical} if, for each agent i, $|P_i|\in\{a,a+1\}$ and P_i contains $\min\{|P_i|, n_i\}$
↪ items that are small for agent i.

[Next verbatim source excerpt]

be the set of agents for whom g is small. We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in M_2, we write (i,j) for an item small exactly for agents
i and j. Thus the items in M_2 form a multigraph on the vertex set
N, where an edge (i,j) is a chore small for precisely agents i and j.

Lean-elaborated paper signature

```
type:
(Item : Type) →
  AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item →
    Finset Item →
      ℕ →
        ℕ →
          (Fin 4 → ℕ) → AppliedModelingLib.FairDivision.Allocation (Fin 4) Item → Finset (Fin 4) → Finset (Fin 4) → Prop

value:
fun (Item : Type) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item) (a b : ℕ)
    (quota : Fin 4 → ℕ) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item)
    (shortAgents longAgents : Finset (Fin 4)) =>
  HT26EFXChores.canonicalSmallChoreAllocationSourceDefinition Item cost chores a b quota allocation ∧
    (0 < b →
       (∀ (agent : Fin 4), agent ∈ shortAgents ↔ quota agent = a) ∧
         ∀ (agent : Fin 4), agent ∈ longAgents ↔ quota agent = a + 1) ∧
      (b = 0 → 0 < a → shortAgents = Finset.univ ∧ longAgents = Finset.univ)
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The target identifies short and long agents by the source quota convention, including the b=0, a>0 convention and the source-undefined b=0, a=0 case.

Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

HT26EFXChores.canonicalSmallChoreAllocationSourceDefinition

Paper-local declaration:
HT26EFXChores.canonicalSmallChoreAllocationSourceDefinition
Source locator: cited publication, lines 752

Verbatim paper-source connection

An allocation $P=(P_1,P_2,P_3,P_4)$ of M_{01} is \emph{canonical} if, for each agent i, $|P_i|\in\{a,a+1\}$ and P_i contains $\min\{|P_i|, n_i\}$
↪ items that are small for agent i.

[Next verbatim source excerpt]

We will assume $r>2$ in the remaining part of this section.
For a chore g, let
\[
S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom g is small. We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in M_2, we write (i,j) for an item small exactly for agents
i and j. Thus the items in M_2 form a multigraph on the vertex set
N, where an edge (i,j) is a chore small for precisely agents i and j.

[Next verbatim source excerpt]

For each agent i, let
\[
n_i=\left|\{g\in M_{01}:S(g)=\{i\}\}\right|
\]
be the number of M_{01} items uniquely small for i. Items in M_{01}
with $S(g)=\emptyset$ are all-large items.
Let $|M_{01}|=4a+b$ with $b\in\{0,1,2,3\}$.
As mentioned before, we will make sure each agent receives either a items or $a+1$ items.

Fix quotas $q_i\in\{a,a+1\}$ with $\sum_i q_i=|M_{01}|$.
We consider allocations of M_{01} with quotas $q=(q_1,q_2,q_3,q_4)$ in which each agent i first receives as many of her uniquely-small items as
↪ possible, up to quota q_i, and the remaining slots are filled with the remaining items.
We will show that all such allocations satisfy EFX.
It is also important that we have the freedom to choose which agents receive a items and which agents receive $a+1$ items.
This freedom enables us to append the allocation of M_2 after the allocation of M_{01}.

We begin by defining this type of allocations.

[Next verbatim source excerpt]

be the set of agents for whom g is small. We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in M_2, we write (i,j) for an item small exactly for agents
i and j. Thus the items in M_2 form a multigraph on the vertex set
N, where an edge (i,j) is a chore small for precisely agents i and j.

Lean-elaborated paper signature

```
type:
(Item : Type) →
  AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item →
    Finset Item → ℕ → ℕ → (Fin 4 → ℕ) → AppliedModelingLib.FairDivision.Allocation (Fin 4) Item → Prop

value:
fun (Item : Type) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item) (a b : ℕ)
    (quota : Fin 4 → ℕ) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item) =>
  b ≤ 3 ∧
    chores.card = 4 * a + b ∧
      (∀ item ∈ chores, AppliedModelingLib.FairDivision.IsSmallForAtMostOne cost item) ∧
        (∀ (agent : Fin 4), quota agent = a ∨ quota agent = a + 1) ∧
          Finset.univ.sum quota = chores.card ∧
            AppliedModelingLib.FairDivision.IsAllocationOf allocation chores ∧
              ∀ (agent : Fin 4),
                Finset.card (allocation agent) = quota agent ∧
                  (AppliedModelingLib.FairDivision.ownSmallChoreSet cost (allocation agent) agent).card =
                    min (quota agent) (AppliedModelingLib.FairDivision.ownSmallChoreSet cost chores agent).card
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The target encodes b at most three, the 4a+b chore count, balanced quotas, complete allocation, and exactly the source maximum number of own-small chores per bundle.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

HT26EFXChores.superCanonicalSmallChoreAllocationSourceDefinition

Paper-local declaration:
HT26EFXChores.superCanonicalSmallChoreAllocationSourceDefinition
Source locator: cited publication, lines 755

Verbatim paper-source connection

For $b>0$, an allocation $P=(P_1,P_2,P_3,P_4)$ is \emph{super-canonical} if it is canonical and satisfies $c_i(P_i)\leq c_i(P_j)-r$ for every short agent i ↪ and every long agent j.

[Next verbatim source excerpt]

We will assume $r>2$ in the remaining part of this section.
For a chore g, let
\[
 S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom g is small. We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in M_2, we write (i,j) for an item small exactly for agents
i and j. Thus the items in M_2 form a multigraph on the vertex set
N, where an edge (i,j) is a chore small for precisely agents i and j.

[Next verbatim source excerpt]

For each agent i, let
\[
 n_i=\left|\{g\in M_{01}:S(g)=\{i\}\}\right|
\]
be the number of M_{01} items uniquely small for i. Items in M_{01}
with $S(g)=\emptyset$ are all-large items.
Let $|M_{01}|=4a+b$ with $b\in\{0,1,2,3\}$.
As mentioned before, we will make sure each agent receives either a items or $a+1$ items.

Fix quotas $q_i\in\{a,a+1\}$ with $\sum_i q_i=|M_{01}|$.
We consider allocations of M_{01} with quotas $q=(q_1,q_2,q_3,q_4)$ in which each agent i first receives as many of her uniquely-small items as ↪ possible, up to quota q_i, and the remaining slots are filled with the remaining items.
We will show that all such allocations satisfy EFX.
It is also important that we have the freedom to choose which agents receive a items and which agents receive $a+1$ items.
This freedom enables us to append the allocation of M_2 after the allocation of M_{01}.

We begin by defining this type of allocations.

[Next verbatim source excerpt]

\begin{definition}
 An allocation $P=(P_1,P_2,P_3,P_4)$ of M_{01} is \emph{canonical} if, for each agent i, $|P_i|\in\{a,a+1\}$ and P_i contains $\min\{|P_i|, n_i\}$ ↪ items that are small for agent i.
 In a canonical allocation with $b>0$, agents receiving a items are called \emph{short} agents, and agents receiving $a+1$ items are called ↪ \emph{long} agents.
 For $b=0$ and $a>0$, every agent in a canonical is both long and short.

[Next verbatim source excerpt]

be the set of agents for whom g is small. We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in M_2, we write (i,j) for an item small exactly for agents
i and j. Thus the items in M_2 form a multigraph on the vertex set
N, where an edge (i,j) is a chore small for precisely agents i and j.

Lean-elaborated paper signature

type:
(Item : Type) →
  ℝ →
   AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item →
    Finset Item → ℕ → ℕ → (Fin 4 → ℕ) → AppliedModelingLib.FairDivision.Allocation (Fin 4) Item → Prop

value:
fun (Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item)
   (a b : ℕ) (quota : Fin 4 → ℕ) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item) =>
  0 < b ∧
   ∃ (shortAgents : Finset (Fin 4)) (longAgents : Finset (Fin 4)),
    HT26EFXChores.canonicalShortLongLabelsSourceDefinition Item cost chores a b quota allocation shortAgents
      longAgents ∧
     ∀ shortAgent ∈ shortAgents,
      ∀ longAgent ∈ longAgents,
       AppliedModelingLib.FairDivision.additiveChoreCost cost shortAgent (allocation shortAgent) ≤
        AppliedModelingLib.FairDivision.additiveChoreCost cost shortAgent (allocation longAgent) - r

Recorded source-to-declaration screening **Verdict:** matches
Reason: The target requires b positive, canonicality with the source short/long convention, and exactly the stated r cost gap from every short agent to every long agent.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Main-text source claims

Review row 1

Source locator: cited publication, lines 304-306

Verbatim source input

```
\begin{theorem}\label{thm:tri}
    For any $n\geq 4$, there exists a tri-valued instance with $n$ agents where no EFX allocation exists.
\end{theorem}
```

[Next verbatim source excerpt]

```
\section{Preliminaries}
A set $M$ of $m$ chores $g_1,\ldots,g_m$ is allocated to a set $N$ of $n$ agents $1,\ldots,n$.
We will use ``item'' and ``chore'' interchangeably in this paper to refer to an element in $M$.
Each agent $i$'s cost function $c_i:2^M\to\mathbb{R}_{\geq0}$ is additive:
$$c_i(S)=\sum_{g\in S}c_i(\{g\}).$$
We write $c_i(g_j)$ in short for $c_i(\{g_j\})$ for notation simplicity.
We will use ``cost function'' and ``disutility function'' interchangeably in this paper.
```

[Next verbatim source excerpt]

```
We say that the cost function is tri-valued if there exist constants $p,q,r\geq 0$ such that $c_i(g_j)\in\{p,q,r\}$ for any $i\in N$ and $g_j\in M$.
The cost function is bi-valued if there exist constants $p,q\geq 0$ such that $c_i(g_j)\in\{p,q\}$ for any $i\in N$ and $g_j\in M$.
For bi-valued cost functions, we will focus on the case $0<p<q$ (the case $0=p<q$ corresponds to the setting with binary cost functions and we know an
↪  allocation that is EFX and Pareto-optimal always exists), and we will assume $c_i(g_j)\in\{1,r\}$ (for any $i\in N$ and $g_j\in M$) instead.
We say that the item $g_j$ is ``large'' for agent $i$ if $c_i(g_j)=r$ and it is ``small'' for agent $i$ if $c_i(g_j)=1$.
```

Expanded PaperInterface specification

```
∀ (n : ℕ),
  4 ≤ n →
    ∃ (m : ℕ) (choreInstance : AppliedModelingLib.FairDivision.AdditiveChoreInstance (Fin n) (Fin m)),
      AppliedModelingLib.FairDivision.IsTriValuedChoreCost choreInstance.cost ∧
        ∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) (Fin m)),
          choreInstance.IsFeasible allocation → ¬choreInstance.IsEFX allocation
```

Lean proof endpoint (built separately)

HT26EFXChores.tri_valued_nonexistence

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact n-at-least-four existential tri-valued additive nonnegative instance with no feasible EFX allocation.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 2

Source locator: cited publication, lines 340-342

Verbatim source input

```
\begin{proposition}
   No agent can receive more than one item in $A$.
\end{proposition}

[Next verbatim source excerpt]

The instance consists of $4$ agents and $13$ items.
The items are partitioned into three groups:
$$A=\{a_1,a_2,a_3\},\quad B=\{b_1,b_2,b_3,b_4,b_5\},\quad\mbox{and}\quad C=\{c_1,c_2,c_3,c_4,c_5\}.$$
The items group $A$ consists of ``large'' items where each agent values each of them at $20$.
For each item in $B$, agents $1$ and $2$ value it at $1$ and agents $3$ and $4$ value it at $7$.
For each item in $C$, agents $1$ and $2$ value it at $7$ and agents $3$ and $4$ value it at $1$.

[Next verbatim source excerpt]

In the remaining part of this section, we will show that no EFX allocation exists in this instance.
Suppose for contradiction an EFX allocation $X=(X_1,X_2,X_3,X_4)$ exists.
Firstly, we will show that no one can receive more than one large item in $A$.

[Next verbatim source excerpt]

\begin{definition}[EFX for chores]
An allocation $X=(X_1,\ldots,X_n)$ is \emph{EFX} if for every pair of agents $i,j$ we have either
\begin{itemize}
   \item $X_i=\emptyset$, or
   \item for every item $g\in X_i$, $c_i(X_i\setminus\{g\})\le c_i(X_j).$
\end{itemize}
Equivalently, if
\[
   \tau_i(X):=\tau_i(X_i):=
   \begin{cases}
      0, & X_i=\emptyset,\\
      c_i(X_i)-\min_{g\in X_i}c_i(g), & X_i\ne\emptyset,
   \end{cases}
\]
then $X$ is EFX if and only if $\tau_i(X)\le c_i(X_j)$ for all $i,j$.
\end{definition}
```

Expanded PaperInterface specification

```
∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) (Fin 13)),
  AppliedModelingLib.FairDivision.IsAllocationOf allocation Finset.univ →
    AppliedModelingLib.FairDivision.EFXForChores
      (AppliedModelingLib.FairDivision.additiveChoreCost fun (agent : Fin 4) (item : Fin 13) =>
        if ↑item < 3 then 20 else if ↑agent < 2 then if ↑item < 8 then 1 else 7 else if ↑item < 8 then 7 else 1)
      allocation →
    ∀ (agent : Fin 4), {item ∈ allocation agent | ↑item < 3}.card ≤ 1
```

Lean proof endpoint (built separately)

```
HT26EFXChores.tri_four_no_two_large
```

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact concrete four-agent, 13-item instance and at-most-one-A-item EFX consequence.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 3

Source locator: cited publication, lines 363-365

Verbatim source input

```
\begin{proposition}
    For any agent $i$, we have $c_i(X_1)\geq 20$.
\end{proposition}
```

[Next verbatim source excerpt]

```
The instance consists of $4$ agents and $13$ items.
The items are partitioned into three groups:
$$A=\{a_1,a_2,a_3\},\quad B=\{b_1,b_2,b_3,b_4,b_5\},\quad\mbox{and}\quad C=\{c_1,c_2,c_3,c_4,c_5\}.$$
The items group $A$ consists of ``large'' items where each agent values each of them at $20$.
For each item in $B$, agents $1$ and $2$ value it at $1$ and agents $3$ and $4$ value it at $7$.
For each item in $C$, agents $1$ and $2$ value it at $7$ and agents $3$ and $4$ value it at $1$.
```

[Next verbatim source excerpt]

```
In the remaining part of this section, we will show that no EFX allocation exists in this instance.
Suppose for contradiction an EFX allocation $X=(X_1,X_2,X_3,X_4)$ exists.
Firstly, we will show that no one can receive more than one large item in $A$.
```

[Next verbatim source excerpt]

```
With the above proposition, it must be that one agent does not receive any large item in $A$ and each of the remaining three agents receives exactly one
↪  item in $A$.
Suppose without loss of generality that agent $1$ does not receive any item from $A$.
```

[Next verbatim source excerpt]

```
\begin{definition}[EFX for chores]
An allocation $X=(X_1,\ldots,X_n)$ is \emph{EFX} if for every pair of agents $i,j$ we have either
\begin{itemize}
    \item $X_i=\emptyset$, or
    \item for every item $g\in X_i$, $c_i(X_i\setminus\{g\})\le c_i(X_j).$
\end{itemize}
Equivalently, if
\[
    \tau_i(X):=\tau_i(X_i):=
    \begin{cases}
        0, & X_i=\emptyset,\\
        c_i(X_i)-\min_{g\in X_i}c_i(g), & X_i\ne\emptyset,
    \end{cases}
\]
then $X$ is EFX if and only if $\tau_i(X)\le c_i(X_j)$ for all $i,j$.
\end{definition}
```

Expanded PaperInterface specification

```
∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) (Fin 13)),
  AppliedModelingLib.FairDivision.IsAllocationOf allocation Finset.univ →
    AppliedModelingLib.FairDivision.EFXForChores
      (AppliedModelingLib.FairDivision.additiveChoreCost fun (agent : Fin 4) (item : Fin 13) =>
        if ↑item < 3 then 20 else if ↑agent < 2 then if ↑item < 8 then 1 else 7 else if ↑item < 8 then 7 else 1)
      allocation →
     {item ∈ allocation 0 | ↑item < 3}.card = 0 →
      (∀ (agent : Fin 4), agent ≠ 0 → {item ∈ allocation agent | ↑item < 3}.card = 1) →
        ∀ (agent : Fin 4),
          20 ≤
            AppliedModelingLib.FairDivision.additiveChoreCost
              (fun (agent : Fin 4) (item : Fin 13) =>
                if ↑item < 3 then 20
                else if ↑agent < 2 then if ↑item < 8 then 1 else 7 else if ↑item < 8 then 7 else 1)
              agent (allocation 0)
```

Lean proof endpoint (built separately)

HT26EFXChores.tri_four_no_a_bundle_expensive

Recorded source-to-Spec screening **Verdict:** matches

Reason: Under the supplied WLOG distribution of the three A-items, asserts every agent values agent 1’s A-free bundle at least 20.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 4

Source locator: cited publication, lines 416-418

Verbatim source input

```
\begin{theorem}\label{thm:EFXPO}
    For every $n\geq 4$ and $r>\lceil n/2\rceil+1$, there exists a $(1,r)$-bi-valued instance with $n$ agents where every EFX allocation is not Pareto-optimal.
\end{theorem}

[Next verbatim source excerpt]

\section{Preliminaries}
A set $M$ of $m$ chores $g_1,\ldots,g_m$ is allocated to a set $N$ of $n$ agents $1,\ldots,n$.
We will use ``item'' and ``chore'' interchangeably in this paper to refer to an element in $M$.
Each agent $i$'s cost function $c_i:2^M\to\mathbb{R}_{\geq0}$ is additive:
$$c_i(S)=\sum_{g\in S}c_i(\{g\}).$$
We write $c_i(g_j)$ in short for $c_i(\{g_j\})$ for notation simplicity.
We will use ``cost function'' and ``disutility function'' interchangeably in this paper.

[Next verbatim source excerpt]

We say that the cost function is tri-valued if there exist constants $p,q,r\geq 0$ such that $c_i(g_j)\in\{p,q,r\}$ for any $i\in N$ and $g_j\in M$.
The cost function is bi-valued if there exist constants $p,q\geq 0$ such that $c_i(g_j)\in\{p,q\}$ for any $i\in N$ and $g_j\in M$.
For bi-valued cost functions, we will focus on the case $0<p<q$ (the case $0=p<q$ corresponds to the setting with binary cost functions and we know an
↪  allocation that is EFX and Pareto-optimal always exists), and we will assume $c_i(g_j)\in\{1,r\}$ (for any $i\in N$ and $g_j\in M$) instead.
We say that the item $g_j$ is ``large'' for agent $i$ if $c_i(g_j)=r$ and it is ``small'' for agent $i$ if $c_i(g_j)=1$.
```

Expanded PaperInterface specification

```
∀ (n : ℕ),
  4 ≤ n →
    ∀ (r : ℝ),
      ↑((n + 1) / 2) + 1 < r →
        ∃ (m : ℕ) (choreInstance : AppliedModelingLib.FairDivision.AdditiveChoreInstance (Fin n) (Fin m)),
          AppliedModelingLib.FairDivision.IsOneOrRChoreCost choreInstance.cost r ∧
            ∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) (Fin m)),
              choreInstance.IsFeasible allocation →
                choreInstance.IsEFX allocation → ¬choreInstance.IsParetoOptimal allocation
```

Lean proof endpoint (built separately)

HT26EFXChores.efx_pareto_incompatibility

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact existence of a normalized one-or-r bi-valued feasible instance whose EFX allocations are not Pareto-optimal.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 5

Source locator: cited publication, lines 448-452

Verbatim source input

```
Notice that an EFX allocation exists in this instance.
For each agent $i\in T_1$, she receives a large item from $A$ and a small item from $B$.
For all but one agent in $T_2$, each of them receives a large item from $A$ and a small item from $C$.
The remaining agent in $T_2$ receives one item from $B$ and two items from $C$.
The readers can easily verify this is an EFX allocation.

[Next verbatim source excerpt]

Agents are partitioned into two groups $T_1$ and $T_2$ with $|T_1|=t_1$ and $|T_2|=t_2$.
There are $m=2n+1$ items which are partitioned into three groups $A,B$, and $C$.
$A$ is the set of large items containing $n-1$ items which have cost $r$ to every agent.
$B$ contains $t_1+1$ items.
Each agent in $T_1$ values each item in $B$ at $1$ and each item in $C$ at $r$.
$C$ contains $t_2+1$ items.
Each agent in $T_2$ values each item in $B$ at $r$ and each item in $C$ at $1$.
Recall that $r$ can be any real number greater than $t_2+1$.
Table~\ref{tab:po} illustrates the instance.

\begin{table}[h]
   \centering
   \caption{A bi-valued instance where every EFX allocation fails to be Pareto-optimal.  Each entry is the cost of every item in the corresponding item class.}
   \begin{tabular}{cccc}
   \hline
       & $A=\{a_1,\ldots,a_{n-1}\}$ & $B=\{b_1,\ldots,b_{t_1+1}\}$ & $C=\{c_1,\ldots,c_{t_2+1}\}$\\
   \hline
     Agents in $T_1$ & $r$ & $1$ & $r$\\
     Agents in $T_2$ & $r$ & $r$ & $1$\\
   \hline
   \end{tabular}

   \label{tab:po}
\end{table}
```

Expanded PaperInterface specification

```
∀ (n : ℕ),
  4 ≤ n →
    ∀ (r : ℝ),
      ↑((n + 1) / 2) + 1 < r →
       ∃ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) (Fin (2 * n + 1))),
         AppliedModelingLib.FairDivision.IsAllocationOf allocation Finset.univ ∧
           AppliedModelingLib.FairDivision.EFXForChores
             (AppliedModelingLib.FairDivision.additiveChoreCost fun (agent : Fin n) (item : Fin (2 * n + 1)) =>
               if ↑item < n - 1 then r
               else if ↑agent < n / 2 then if ↑item < n + n / 2 then 1 else r else if ↑item < n + n / 2 then r else 1)
             allocation
```

Lean proof endpoint (built separately)

HT26EFXChores.efx_pareto_construction_has_efx

Recorded source-to-Spec screening **Verdict:** matches

Reason: Finite encoding has the stated group sizes, A/B/C costs, r threshold, and EFX existence.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 6

Source locator: cited publication, lines 461-463

Verbatim source input

```
\begin{proposition}\label{prop:PO-exist-large}
   For every $X_i$, there exists $g\in X_i$ with $c_i(g)=r$.
\end{proposition}

[Next verbatim source excerpt]

\begin{theorem}\label{thm:EFXPO}
   For every $n\geq 4$ and $r>\lceil n/2\rceil+1$, there exists a $(1,r)$-bi-valued instance with $n$ agents where every EFX allocation is not Pareto-optimal.
\end{theorem}

[Next verbatim source excerpt]

Let $t_1=\lfloor n/2\rfloor$ and $t_2=\lceil n/2\rceil$.
Agents are partitioned into two groups $T_1$ and $T_2$ with $|T_1|=t_1$ and $|T_2|=t_2$.
There are $m=2n+1$ items which are partitioned into three groups $A,B$, and $C$.
$A$ is the set of large items containing $n-1$ items which have cost $r$ to every agent.
$B$ contains $t_1+1$ items.
Each agent in $T_1$ values each item in $B$ at $1$ and each item in $C$ at $r$.
$C$ contains $t_2+1$ items.
Each agent in $T_2$ values each item in $B$ at $r$ and each item in $C$ at $1$.
Recall that $r$ can be any real number greater than $t_2+1$.
```

Expanded PaperInterface specification

```
∀ (n : ℕ),
  4 ≤ n →
   ∀ (r : ℝ),
     ↑((n + 1) / 2) + 1 < r →
      ∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) (Fin (2 * n + 1))),
        AppliedModelingLib.FairDivision.IsAllocationOf allocation Finset.univ →
          AppliedModelingLib.FairDivision.EFXForChores
              (AppliedModelingLib.FairDivision.additiveChoreCost fun (agent : Fin n) (item : Fin (2 * n + 1)) =>
                if ↑item < n - 1 then r
                else
                  if ↑agent < n / 2 then if ↑item < n + n / 2 then 1 else r else if ↑item < n + n / 2 then r else 1)
              allocation →
            ∀ (agent : Fin n),
              ∃ item ∈ allocation agent,
                (if ↑item < n - 1 then r
                  else
                    if ↑agent < n / 2 then if ↑item < n + n / 2 then 1 else r
                    else if ↑item < n + n / 2 then r else 1) =
                  r
```

Lean proof endpoint (built separately)

HT26EFXChores.efx_po_every_agent_large

Recorded source-to-Spec screening **Verdict:** matches

Reason: Uses the stated instance and asserts every agent's bundle contains an item of own cost r.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 7

Source locator: cited publication, lines 502-504

Verbatim source input

```
\begin{proposition}\label{prop:r+1r+2}
    For every agent $i$, we have $c_i(X_i)\geq r+1$. Moreover, there exists an agent $i$ with $c_i(X_i)\geq r+2$.
\end{proposition}

[Next verbatim source excerpt]

\begin{theorem}\label{thm:EFXPO}
    For every $n\geq 4$ and $r>\lceil n/2\rceil+1$, there exists a $(1,r)$-bi-valued instance with $n$ agents where every EFX allocation is not Pareto-optimal.
\end{theorem}

[Next verbatim source excerpt]

Let $t_1=\lfloor n/2\rfloor$ and $t_2=\lceil n/2\rceil$.
Agents are partitioned into two groups $T_1$ and $T_2$ with $|T_1|=t_1$ and $|T_2|=t_2$.
There are $m=2n+1$ items which are partitioned into three groups $A,B$, and $C$.
$A$ is the set of large items containing $n-1$ items which have cost $r$ to every agent.
$B$ contains $t_1+1$ items.
Each agent in $T_1$ values each item in $B$ at $1$ and each item in $C$ at $r$.
$C$ contains $t_2+1$ items.
Each agent in $T_2$ values each item in $B$ at $r$ and each item in $C$ at $1$.
Recall that $r$ can be any real number greater than $t_2+1$.
```

Expanded PaperInterface specification

```
∀ (n : ℕ),
  4 ≤ n →
   ∀ (r : ℝ),
     ↑((n + 1) / 2) + 1 < r →
      ∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) (Fin (2 * n + 1))),
        AppliedModelingLib.FairDivision.IsAllocationOf allocation Finset.univ →
          AppliedModelingLib.FairDivision.EFXForChores
              (AppliedModelingLib.FairDivision.additiveChoreCost fun (agent : Fin n) (item : Fin (2 * n + 1)) =>
                if ↑item < n - 1 then r
                else
                  if ↑agent < n / 2 then if ↑item < n + n / 2 then 1 else r else if ↑item < n + n / 2 then r else 1)
              allocation →
            (∀ (agent : Fin n),
               r + 1 ≤
                 AppliedModelingLib.FairDivision.additiveChoreCost
                   (fun (agent : Fin n) (item : Fin (2 * n + 1)) =>
                     if ↑item < n - 1 then r
                     else
                       if ↑agent < n / 2 then if ↑item < n + n / 2 then 1 else r
                       else if ↑item < n + n / 2 then r else 1)
                   agent (allocation agent)) ∧
              ∃ (agent : Fin n),
                r + 2 ≤
                  AppliedModelingLib.FairDivision.additiveChoreCost
                    (fun (agent : Fin n) (item : Fin (2 * n + 1)) =>
                      if ↑item < n - 1 then r
                      else
                        if ↑agent < n / 2 then if ↑item < n + n / 2 then 1 else r
                        else if ↑item < n + n / 2 then r else 1)
                    agent (allocation agent)
```

Lean proof endpoint (built separately)

HT26EFXChores.efx_po_cost_lower_bounds

Recorded source-to-Spec screening **Verdict:** matches

Reason: Uses the specified EFXPO instance and concludes every own cost is at least r+1, with one at least r+2.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 8

Source locator: cited publication, lines 532-534

Verbatim source input

```
\begin{theorem}\label{thm:four}
    For four agents, there exists an EFX allocation in every bi-valued instance.
\end{theorem}

[Next verbatim source excerpt]

\section{Preliminaries}
A set $M$ of $m$ chores $g_1,\ldots,g_m$ is allocated to a set $N$ of $n$ agents $1,\ldots,n$.
We will use ``item'' and ``chore'' interchangeably in this paper to refer to an element in $M$.
Each agent $i$'s cost function $c_i:2^M\to\mathbb{R}_{\geq0}$ is additive:
$$c_i(S)=\sum_{g\in S}c_i(\{g\}).$$
We write $c_i(g_j)$ in short for $c_i(\{g_j\})$ for notation simplicity.
We will use ``cost function'' and ``disutility function'' interchangeable in this paper.

[Next verbatim source excerpt]

We say that the cost function is tri-valued if there exist constants $p,q,r\geq 0$ such that $c_i(g_j)\in\{p,q,r\}$ for any $i\in N$ and $g_j\in M$.
The cost function is bi-valued if there exist constants $p,q\geq 0$ such that $c_i(g_j)\in\{p,q\}$ for any $i\in N$ and $g_j\in M$.
For bi-valued cost functions, we will focus on the case $0<p<q$ (the case $0=p<q$ corresponds to the setting with binary cost functions and we know an
↪  allocation that is EFX and Pareto-optimal always exists), and we will assume $c_i(g_j)\in\{1,r\}$ (for any $i\in N$ and $g_j\in M$) instead.
We say that the item $g_j$ is ``large'' for agent $i$ if $c_i(g_j)=r$ and it is ``small'' for agent $i$ if $c_i(g_j)=1$.
```

Expanded PaperInterface specification

```
∀ (Item : Type) (choreInstance : AppliedModelingLib.FairDivision.AdditiveChoreInstance (Fin 4) Item),
  AppliedModelingLib.FairDivision.IsBiValuedChoreCost choreInstance.cost →
    ∃ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
      choreInstance.IsFeasible allocation ∧ choreInstance.IsEFX allocation
```

Lean proof endpoint (built separately)

```
HT26EFXChores.four_agent_bi_valued_efx_exists
```

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact four-agent bi-valued EFX-existence claim; instance nonnegativity is carried by AdditiveChoreInstance.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 9

Source locator: cited publication, lines 577-581

Verbatim source input

```
\begin{lemma}[Insertion of an $M_{34}$ item]\label{lem:insertion}
Let $X$ be an EFX allocation of a set of chores $T$.  Let $g\in M_{34}$ be a
chore that is small for at least three agents.  Then there is an EFX
allocation of $T\cup\{g\}$.
\end{lemma}
```

[Next verbatim source excerpt]

```
We say that the cost function is tri-valued if there exist constants $p,q,r\geq 0$ such that $c_i(g_j)\in\{p,q,r\}$ for any $i\in N$ and $g_j\in M$.
The cost function is bi-valued if there exist constants $p,q\geq 0$ such that $c_i(g_j)\in\{p,q\}$ for any $i\in N$ and $g_j\in M$.
For bi-valued cost functions, we will focus on the case $0<p<q$ (the case $0=p<q$ corresponds to the setting with binary cost functions and we know an
↪  allocation that is EFX and Pareto-optimal always exists), and we will assume $c_i(g_j)\in\{1,r\}$ (for any $i\in N$ and $g_j\in M$) instead.
We say that the item $g_j$ is ``large'' for agent $i$ if $c_i(g_j)=r$ and it is ``small'' for agent $i$ if $c_i(g_j)=1$.
```

[Next verbatim source excerpt]

```
We will assume $r>2$ in the remaining part of this section.
For a chore $g$, let
\[
 S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom $g$ is small.  We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in $M_2$, we write $(i,j)$ for an item small exactly for agents
$i$ and $j$.  Thus the items in $M_2$ form a multigraph on the vertex set
$N$, where an edge $(i,j)$ is a chore small for precisely agents $i$ and $j$.
```

Expanded PaperInterface specification

```
∀ (Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item)
  (item : Item),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      item ∉ chores →
        AppliedModelingLib.FairDivision.IsSmallForAtLeastThree cost item →
          ∀ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
            AppliedModelingLib.FairDivision.IsAllocationOf allocation chores →
              AppliedModelingLib.FairDivision.EFXForChores (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                  allocation →
                ∃ (extended : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
                  AppliedModelingLib.FairDivision.IsAllocationOf extended (insert item chores) ∧
                    AppliedModelingLib.FairDivision.EFXForChores
                      (AppliedModelingLib.FairDivision.additiveChoreCost cost) extended
```

Lean proof endpoint (built separately)

HT26EFXChores.m34_insertion

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact four-agent M34 insertion result; the explicit freshness condition is standard nonsemantic insertion presentation.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 10

Source locator: cited publication, lines 673-677

Verbatim source input

```
\begin{remark}
   This idea also works for general numbers of agents.
   If we have a partial EFX allocation such that each of the remaining unallocated items is valued as large for at most one agent, then we can always
   ↪  allocate the remaining items one by one until we get a complete EFX allocation.
   Therefore, under bi-valued setting, items valued as large by at most one agent are never a real obstacle for EFX allocations.
\end{remark}

[Next verbatim source excerpt]

We say that the cost function is tri-valued if there exist constants $p,q,r\geq 0$ such that $c_i(g_j)\in\{p,q,r\}$ for any $i\in N$ and $g_j\in M$.
The cost function is bi-valued if there exist constants $p,q\geq 0$ such that $c_i(g_j)\in\{p,q\}$ for any $i\in N$ and $g_j\in M$.
For bi-valued cost functions, we will focus on the case $0<p<q$ (the case $0=p<q$ corresponds to the setting with binary cost functions and we know an
↪  allocation that is EFX and Pareto-optimal always exists), and we will assume $c_i(g_j)\in\{1,r\}$ (for any $i\in N$ and $g_j\in M$) instead.
We say that the item $g_j$ is ``large'' for agent $i$ if $c_i(g_j)=r$ and it is ``small'' for agent $i$ if $c_i(g_j)=1$.

[Next verbatim source excerpt]

\section{EFX Allocations for Four Agents with Bi-Valued Disutilities}
We continue to study bi-valued instances.
In this section, we show that an EFX allocation always exists for four agents.

\begin{theorem}\label{thm:four}
   For four agents, there exists an EFX allocation in every bi-valued instance.
\end{theorem}

For $r=2$, \citet{lin2025approximately} shows that EFX is compatible with Pareto-optimality for any number of agents.
If we only consider EFX, the case with $r\leq 2$ can be handled by the following simple variant of the round-robin algorithm for any number of agents.
Let $m=an+b$ for $b<n$.
If $b=0$, the standard round-robin algorithm outputs an EFX allocation $X=(X_1,\ldots,X_n)$: for any pair of agents $i,j$ with $i<j$, $i$ will not envy $j$
↪  as $i$ always picks first; we also have $c_j(X_j)-c_j(X_i)\leq r-1$ (if in a round agent $j$ begins to receive an large item whereas agent $i$'s item in
↪  this round is small for agent $j$, then every item in the remaining process of this algorithm is large for agent $j$), $j$ will not envy $i$ up to removing
↪  any item from $j$'s bundle since $r-1\leq1$.
If $b>0$, we introduce a special ``first reverse round'' of the round-robin algorithm where agents $b,b-1,\ldots,1$ iteratively pick favorite items in the order.
After that, we perform the standard round-robin algorithm in the standard order $1,\ldots,n$.
For each pair of agents $i,j$ with $i<j$, agent $i$ will not envy agent $j$ after removing the item picked in the first special round. If this item is a small
↪  item for agent $i$, EFX condition holds; if this item is a large item, then $X_i$ only contains large items, and EFX condition also holds.
The reason agent $j$ will not strongly envy agent $i$ is similar as the $b=0$ case: agent $j$ has an advantage to agent $i$ in the first round, and after
↪  that the disadvantage is at most $r-1$.

We will assume $r>2$ in the remaining part of this section.
For a chore $g$, let
\[
 S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom $g$ is small.  We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in $M_2$, we write $(i,j)$ for an item small exactly for agents
```

Expanded PaperInterface specification

```
∀ (Agent Item : Type) [Fintype Agent] [Nonempty Agent] [DecidableEq Agent] [inst : DecidableEq Item] (r : ℝ)
  (cost : AppliedModelingLib.FairDivision.ChoreCost Agent Item) (remaining chores : Finset Item),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      Disjoint remaining chores →
        (∀ item ∈ remaining, ∀ (first second : Agent), cost first item = r → cost second item = r → first = second) →
          (∀ item ∈ remaining,
              ∀ (allocation : AppliedModelingLib.FairDivision.Allocation Agent Item),
                AppliedModelingLib.FairDivision.IsAllocationOf allocation chores →
                  AppliedModelingLib.FairDivision.EFXForChores (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                      allocation →
                    ∃ (extended : AppliedModelingLib.FairDivision.Allocation Agent Item),
                      AppliedModelingLib.FairDivision.IsAllocationOf extended (insert item chores) ∧
                        AppliedModelingLib.FairDivision.EFXForChores
                          (AppliedModelingLib.FairDivision.additiveChoreCost cost) extended) ∧
            ∀ (allocation : AppliedModelingLib.FairDivision.Allocation Agent Item),
              AppliedModelingLib.FairDivision.IsAllocationOf allocation chores →
                AppliedModelingLib.FairDivision.EFXForChores (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                    allocation →
                  ∃ (extended : AppliedModelingLib.FairDivision.Allocation Agent Item),
                    AppliedModelingLib.FairDivision.IsAllocationOf extended (chores ∪ remaining) ∧
                      AppliedModelingLib.FairDivision.EFXForChores
                        (AppliedModelingLib.FairDivision.additiveChoreCost cost) extended
```

Lean proof endpoint (built separately)

HT26EFXChores.m34_insertion_general_agents

Recorded source-to-Spec screening **Verdict:** matches
Reason: Exact general finite-agent sequential insertion claim for remaining items large for at most one agent, including final EFX completion.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 11

Source locator: cited publication, lines 706-724

Verbatim source input

```
\begin{lemma}[Composition]\label{lem:composition}
Let $P=(P_1,\ldots,P_n)$ be an allocation of some items and $B=(B_1,\ldots,B_n)$ an allocation of disjoint
items.  If
\[
 c_i(P_i)\le c_i(P_j)\qquad\forall i,j
\]
and
\[
 c_i(B_i)-1\le c_i(B_j)\qquad\forall i,j,
\]
then the combined allocation $X_i=P_i\cup B_i$ is EFX. %provided every agent whose second inequality is tight has a small item in her final bundle.
More generally, $X$ is EFX whenever
\[
 c_i(P_i)-c_i(P_j)
 \le c_i(B_j)-c_i(B_i)+\min_{g\in X_i}c_i(g)
 \qquad\forall i,j.
\]
If the inequalities above hold only for a particular pair of agents $(i,j)$, then $i$ does not strongly envy $j$.
\end{lemma}
```

[Next verbatim source excerpt]

```
We say that the cost function is tri-valued if there exist constants $p,q,r\geq 0$ such that $c_i(g_j)\in\{p,q,r\}$ for any $i\in N$ and $g_j\in M$.
The cost function is bi-valued if there exist constants $p,q\geq 0$ such that $c_i(g_j)\in\{p,q\}$ for any $i\in N$ and $g_j\in M$.
For bi-valued cost functions, we will focus on the case $0<p<q$ (the case $0=p<q$ corresponds to the setting with binary cost functions and we know an
↪  allocation that is EFX and Pareto-optimal always exists), and we will assume $c_i(g_j)\in\{1,r\}$ (for any $i\in N$ and $g_j\in M$) instead.
We say that the item $g_j$ is ``large'' for agent $i$ if $c_i(g_j)=r$ and it is ``small'' for agent $i$ if $c_i(g_j)=1$.
```

[Next verbatim source excerpt]

```
We will assume $r>2$ in the remaining part of this section.
```

Expanded PaperInterface specification

```
∀ (Agent Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost Agent Item)
  (leftChores rightChores : Finset Item)
  (leftAllocation rightAllocation : AppliedModelingLib.FairDivision.Allocation Agent Item),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      Disjoint leftChores rightChores →
        AppliedModelingLib.FairDivision.IsAllocationOf leftAllocation leftChores →
          AppliedModelingLib.FairDivision.IsAllocationOf rightAllocation rightChores →
            AppliedModelingLib.FairDivision.IsAllocationOf
              (fun (agent : Agent) => leftAllocation agent ∪ rightAllocation agent) (leftChores ∪ rightChores) ∧
            ((AppliedModelingLib.FairDivision.EnvyFreeForChores
                  (AppliedModelingLib.FairDivision.additiveChoreCost cost) leftAllocation ∧
                ∀ (i j : Agent),
                  AppliedModelingLib.FairDivision.additiveChoreCost cost i (rightAllocation i) - 1 ≤
                    AppliedModelingLib.FairDivision.additiveChoreCost cost i (rightAllocation j)) →
              AppliedModelingLib.FairDivision.EFXForChores (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                fun (agent : Agent) => leftAllocation agent ∪ rightAllocation agent) ∧
              ((∀ (i j : Agent) (hnonempty : Finset.Nonempty (leftAllocation i ∪ rightAllocation i)),
                  AppliedModelingLib.FairDivision.additiveChoreCost cost i (leftAllocation i) -
                      AppliedModelingLib.FairDivision.additiveChoreCost cost i (leftAllocation j) ≤
                    AppliedModelingLib.FairDivision.additiveChoreCost cost i (rightAllocation j) -
                      AppliedModelingLib.FairDivision.additiveChoreCost cost i (rightAllocation i) +
                    (Finset.image (cost i) (leftAllocation i ∪ rightAllocation i)).min'
                      (HT26EFXChores.compositionSpec._proof_1 Agent Item cost leftAllocation rightAllocation i
                        hnonempty)) →
                AppliedModelingLib.FairDivision.EFXForChores
                  (AppliedModelingLib.FairDivision.additiveChoreCost cost) fun (agent : Agent) =>
                  leftAllocation agent ∪ rightAllocation agent) ∧
                ∀ (i j : Agent) (hnonempty : Finset.Nonempty (leftAllocation i ∪ rightAllocation i)),
                  AppliedModelingLib.FairDivision.additiveChoreCost cost i (leftAllocation i) -
                      AppliedModelingLib.FairDivision.additiveChoreCost cost i (leftAllocation j) ≤
                    AppliedModelingLib.FairDivision.additiveChoreCost cost i (rightAllocation j) -
                      AppliedModelingLib.FairDivision.additiveChoreCost cost i (rightAllocation i) +
                    (Finset.image (cost i) (leftAllocation i ∪ rightAllocation i)).min'
                      (HT26EFXChores.compositionSpec._proof_1 Agent Item cost leftAllocation rightAllocation i
                        hnonempty) →
                  AppliedModelingLib.FairDivision.DoesNotStronglyEnvyForChores
                    (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                    (fun (agent : Agent) => leftAllocation agent ∪ rightAllocation agent) i j
```

Lean proof endpoint (built separately)

HT26EFXChores.composition

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact disjoint-allocation composition, envy-free-plus-minus-one clause, minimum-item clause, and no-strong-envy consequence.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 12

Source locator: cited publication, lines 760-767

Verbatim source input

```
\begin{lemma}\label{lem:canonical}
   We have the following properties regarding canonical allocations.
   \begin{enumerate}[label=(\alph*)]
      \item Every canonical allocation is EFX. For $b=0$, every canonical allocation is envy-free.
      \item For any $b>0$, there always exists a super-canonical allocation.
      \item If $a>0$ and $b=2$, for any partition $N_1,N_2$ of $N$ with $|N_1|=|N_2|=2$, there exist $i\in N_1$ and $j\in N_2$ such that there exists a
      ↪  super-canonical allocation $P$ with $|P_i|=|P_j|=a$.
   \end{enumerate}
\end{lemma}
```

[Next verbatim source excerpt]

```
We will assume $r>2$ in the remaining part of this section.
For a chore $g$, let
\[
 S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom $g$ is small.  We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in $M_2$, we write $(i,j)$ for an item small exactly for agents
$i$ and $j$.  Thus the items in $M_2$ form a multigraph on the vertex set
$N$, where an edge $(i,j)$ is a chore small for precisely agents $i$ and $j$.
```

[Next verbatim source excerpt]

```
For each agent $i$, let
\[
 n_i=\left|\{g\in M_{01}:S(g)=\{i\}\}\right|
\]
be the number of $M_{01}$ items uniquely small for $i$.  Items in $M_{01}$
with $S(g)=\emptyset$ are all-large items.
Let $|M_{01}|=4a+b$ with $b\in\{0,1,2,3\}$.
As mentioned before, we will make sure each agent receives either $a$ items or $a+1$ items.

Fix quotas $q_i\in\{a,a+1\}$ with $\sum_i q_i=|M_{01}|$.
We consider allocations of $M_{01}$ with quotas $q=(q_1,q_2,q_3,q_4)$ in which each agent $i$ first receives as many of her uniquely-small items as
↪  possible, up to quota $q_i$, and the remaining slots are filled with the remaining items.
We will show that all such allocations satisfy EFX.
It is also important that we have the freedom to choose which agents receive $a$ items and which agents receive $a+1$ items.
This freedom enables us to append the allocation of $M_2$ after the allocation of $M_{01}$.

We begin by defining this type of allocations.
```

[Next verbatim source excerpt]

```
be the set of agents for whom $g$ is small.  We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in $M_2$, we write $(i,j)$ for an item small exactly for agents
$i$ and $j$.  Thus the items in $M_2$ form a multigraph on the vertex set
$N$, where an edge $(i,j)$ is a chore small for precisely agents $i$ and $j$.
```

[Next verbatim source excerpt]

```
\begin{definition}
   An allocation $P=(P_1,P_2,P_3,P_4)$ of $M_{01}$ is \emph{canonical} if, for each agent $i$, $|P_i|\in\{a,a+1\}$ and $P_i$ contains $\min\{|P_i|, n_i\}$
   ↪  items that are small for agent $i$.
   In a canonical allocation with $b>0$, agents receiving $a$ items are called \emph{short} agents, and agents receiving $a+1$ items are called
   ↪  \emph{long} agents.
   For $b=0$ and $a>0$, every agent in a canonical is both long and short.
   For $b>0$, an allocation $P=(P_1,P_2,P_3,P_4)$ is \emph{super-canonical} if it is canonical and satisfies $c_i(P_i)\leq c_i(P_j)-r$ for every short agent
   ↪  $i$ and every long agent $j$.
\end{definition}
```

Expanded PaperInterface specification

```
∀ (Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item)
  (a b : ℕ),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      b ≤ 3 →
        chores.card = 4 * a + b →
          (∀ item ∈ chores, AppliedModelingLib.FairDivision.IsSmallForAtMostOne cost item) →
            (∀ (quota : Fin 4 → ℕ) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
               (∀ (agent : Fin 4), quota agent = a ∨ quota agent = a + 1) →
                 AppliedModelingLib.FairDivision.IsCanonicalSmallChoreAllocation cost chores quota allocation →
                   AppliedModelingLib.FairDivision.EFXForChores
                     (AppliedModelingLib.FairDivision.additiveChoreCost cost) allocation) ∧
```

```
(b = 0 →
   ∀ (quota : Fin 4 → ℕ) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
    (∀ (agent : Fin 4), quota agent = a) →
     AppliedModelingLib.FairDivision.IsCanonicalSmallChoreAllocation cost chores quota allocation →
      AppliedModelingLib.FairDivision.EnvyFreeForChores
       (AppliedModelingLib.FairDivision.additiveChoreCost cost) allocation) ∧
 (0 < b →
    ∃ (quota : Fin 4 → ℕ) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
     (∀ (agent : Fin 4), quota agent = a ∨ quota agent = a + 1) ∧
      AppliedModelingLib.FairDivision.IsCanonicalSmallChoreAllocation cost chores quota allocation ∧
       ∀ (i j : Fin 4),
        quota i = a →
         quota j = a + 1 →
          AppliedModelingLib.FairDivision.additiveChoreCost cost i (allocation i) ≤
           AppliedModelingLib.FairDivision.additiveChoreCost cost i (allocation j) - r) ∧
  (0 < a →
    b = 2 →
     ∀ (N1 N2 : Finset (Fin 4)),
      N1 ∪ N2 = Finset.univ →
       Disjoint N1 N2 →
        N1.card = 2 →
         N2.card = 2 →
          ∃ i ∈ N1,
           ∃ j ∈ N2,
            ∃ (quota : Fin 4 → ℕ) (allocation :
              AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
             (∀ (agent : Fin 4), quota agent = a ∨ quota agent = a + 1) ∧
              AppliedModelingLib.FairDivision.IsCanonicalSmallChoreAllocation cost chores
                 quota allocation ∧
               (∀ (x y : Fin 4),
                  quota x = a →
                   quota y = a + 1 →
                    AppliedModelingLib.FairDivision.additiveChoreCost cost x
                       (allocation x) ≤
                     AppliedModelingLib.FairDivision.additiveChoreCost cost x
                       (allocation y) -
                      r) ∧
                quota i = a ∧ quota j = a)
```

Lean proof endpoint (built separately)

HT26EFXChores.canonical_allocation_properties

Recorded source-to-Spec screening **Verdict:** matches

Reason: Captures all canonical and super-canonical clauses; feasibility and quota-cardinality are within the expanded canonical predicate.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 13

Source locator: cited publication, lines 1019-1031

Verbatim source input

```
\begin{lemma}[Balanced orientation]\label{lem:balancedorientation}
Let $G=(V,E)$ be a multigraph and let $(b_i)_{i\in V}$ be nonnegative
integers satisfying $\sum_{i\in V}b_i=|E|$.  There exists an assignment of
every edge to one of its endpoints such that vertex $i$ receives exactly
$b_i$ edges if and only if
\begin{equation}\label{eqn:orientation}
    \sum_{i\in U}b_i\le \operatorname{inc}_G(U)
 \qquad\text{for every }U\subseteq V.
\end{equation}

If we have \(\sum_{i\in V}b_i\le |E|\) instead, and \eqref{eqn:orientation} is satisfied, then one can assign some edges to their endpoints so that vertex \(i\) receives exactly \(b_i\) edges, leaving the remaining edges unassigned.
\end{lemma}
```

Expanded PaperInterface specification

```
∀ (Vertex Edge : Type) [inst : Fintype Vertex] (edges : Finset Edge) (endpoints : Edge → Finset Vertex)
  (quota : Vertex → ℕ),
  (∀ edge ∈ edges, (endpoints edge).card = 2) →
    (Finset.univ.sum quota = edges.card →
       ((∃ (owner : Edge → Option Vertex),
          (∀ edge ∈ edges, ∃ (vertex : Vertex), owner edge = some vertex ∧ vertex ∈ endpoints edge) ∧
            ∀ (vertex : Vertex), {edge ∈ edges | owner edge = some vertex}.card = quota vertex) ↔
         ∀ (vertices : Finset Vertex),
          vertices.sum quota ≤ {edge ∈ edges | (endpoints edge ∩ vertices).Nonempty}.card)) ∧
     (Finset.univ.sum quota ≤ edges.card →
       (∀ (vertices : Finset Vertex),
          vertices.sum quota ≤ {edge ∈ edges | (endpoints edge ∩ vertices).Nonempty}.card) →
        ∃ (owner : Edge → Option Vertex),
          (∀ edge ∈ edges, ∀ (vertex : Vertex), owner edge = some vertex → vertex ∈ endpoints edge) ∧
            ∀ (vertex : Vertex), {edge ∈ edges | owner edge = some vertex}.card = quota vertex)
```

Lean proof endpoint (built separately)

HT26EFXChores.balanced_orientation

Recorded source-to-Spec screening **Verdict:** matches

Reason: The expanded Spec gives the exact all-edges endpoint-assignment equivalence when total demand equals the edge count and the partial endpoint-assignment conclusion when demand is at most the edge count, with the same incident-edge inequalities for every vertex subset.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 14

Source locator: cited publication, lines 1045-1047

Verbatim source input

```
\begin{lemma}[EFX Allocation of $M_2$]\label{lem:M2}
There is an EFX allocation of $M_2$.
\end{lemma}

[Next verbatim source excerpt]

We will assume $r>2$ in the remaining part of this section.
For a chore $g$, let
\[
 S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom $g$ is small.  We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in $M_2$, we write $(i,j)$ for an item small exactly for agents
$i$ and $j$.  Thus the items in $M_2$ form a multigraph on the vertex set
$N$, where an edge $(i,j)$ is a chore small for precisely agents $i$ and $j$.

[Next verbatim source excerpt]

\subsection{The $M_2$ Multigraph Algorithm}
\label{subsec:2}
We now give a complete allocation procedure for the items in $M_2$.
Each item is small for some agents $i$ and $j$ and
large for the other two agents, and we say that the type of this item is $(i,j)$, or $ij$ for short.
In this section, we consider the case where the instance only has $M_{2}$ items.
We represent the instance by a
multigraph $R$ on $N=\{1,2,3,4\}$, where an item of type $(i,j)$ is an edge $(i,j)$.
We slightly abuse the notation by letting $R$ also represent the edge set, i.e., $R$ exactly corresponds to items in $M_2$.
Let $x_{ij}$ denote the multiplicity of edge $(i,j)$ and let $m_R=|R|$.

For $U\subseteq N$, define
\[
 \operatorname{inc}_R(U)=\left|\{e\in R:e\cap U\neq\emptyset\}\right|
\]
and
\[
 \Delta_R(U)=\frac{|U|m_R}{4}-\operatorname{inc}_R(U).
\]
We call $U$ \emph{deficient} if $\Delta_R(U)>0$.
```

Expanded PaperInterface specification

```
∀ (Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      (∀ item ∈ chores, AppliedModelingLib.FairDivision.IsSmallForExactlyTwo cost item) →
        ∃ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
          AppliedModelingLib.FairDivision.IsAllocationOf allocation chores ∧
            AppliedModelingLib.FairDivision.EFXForChores (AppliedModelingLib.FairDivision.additiveChoreCost cost)
              allocation
```

Lean proof endpoint (built separately)

```
HT26EFXChores.m2_efx_allocation
```

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact four-agent one-or-r, r-greater-than-two, exactly-two-small-agent M2 premise and EFX allocation conclusion.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 15

Source locator: cited publication, lines 1419-1462

Verbatim source input

```
\begin{lemma}[Properties of the EFX allocations in Lemma~\ref{lem:M2}]\label{lem:M2properties}
The EFX allocation $B=(B_1,\ldots,B_4)$ ensured by Lemma~\ref{lem:M2} can be chosen to satisfy the
following additional properties.
\begin{enumerate}[label=(\roman*)]
\item Except in the two configurations described in~(ii), every possible envy is certified by removing a small item.  More precisely,
whenever $B_i$ contains an item that is small for agent $i$,
\[
 c_i(B_i)-1\le c_i(B_j)\qquad\forall j\in N,
\]
whereas every agent whose bundle contains only large items is envy-free:
\[
 c_i(B_i)\le c_i(B_j)\qquad\forall j\in N.
\]

\item Up to relabeling, the only configurations in which an agent whose
bundle contains only large items may fail to be envy-free, and hence
must use a large item as the EFX-removal item, are the following.
\begin{enumerate}[label=(\alph*)]
\item All items have type $(3,4)$ and
\[
 |R|=4q+3.
\]
Then agents $1,2$ receive $q$ and $q+1$ items, respectively, in either
order, while agents $3,4$ receive $q+1$ items each.  We are free to choose
which of agents $1,2$ receives $q+1$ items, and this agent is the unique agent who use a large item as the EFX-removal item.

\item The multiset $R$ consists of one item of type $(1,2)$ and
$4q+3$ items of type $(3,4)$.  One of agents $1,2$ receives the unique
type-$(1,2)$ item together with $q$ type-$(3,4)$ items, the other receives
$q+1$ type-$(3,4)$ items, and agents $3,4$ receive $q+1$ type-$(3,4)$
items each.  We are free to choose which of agents $1,2$ receives the unique
type-$(1,2)$ item.  The other agent is the unique agent who must use a large
item as the EFX-removal item.
\end{enumerate}

\item If every edge of $R$ has type $(1,2)$ or $(3,4)$, with
$x_{34}\ge x_{12}\ge 0$, then the allocation can be chosen so that agent $1$
is envy-free:
\[
 c_1(B_1)\le c_1(B_j)\qquad\forall j\in N.
\]
The same statement holds with agent $2$ in place of agent $1$.
\end{enumerate}
\end{lemma}
```

[Next verbatim source excerpt]

```
We will assume $r>2$ in the remaining part of this section.
For a chore $g$, let
\[
 S(g)=\{i\in N:c_i(g)=1\}
\]
be the set of agents for whom $g$ is small.  We partition the chores into
\[
M_{01}=\{g:|S(g)|\le 1\},\qquad
M_2=\{g:|S(g)|=2\},\qquad
M_{34}=\{g:|S(g)|\ge 3\}.
\]
For an item in $M_2$, we write $(i,j)$ for an item small exactly for agents
$i$ and $j$.  Thus the items in $M_2$ form a multigraph on the vertex set
$N$, where an edge $(i,j)$ is a chore small for precisely agents $i$ and $j$.
```

[Next verbatim source excerpt]

```
\subsection{The $M_2$ Multigraph Algorithm}
\label{subsec:2}
We now give a complete allocation procedure for the items in $M_2$.
Each item is small for some agents $i$ and $j$ and
large for the other two agents, and we say that the type of this item is $(i,j)$, or $ij$ for short.
In this section, we consider the case where the instance only has $M_{2}$ items.
We represent the instance by a
multigraph $R$ on $N=\{1,2,3,4\}$, where an item of type $(i,j)$ is an edge $(i,j)$.
We slightly abuse the notation by letting $R$ also represent the edge set, i.e., $R$ exactly corresponds to items in $M_2$.
Let $x_{ij}$ denote the multiplicity of edge $(i,j)$ and let $m_R=|R|$.

For $U\subseteq N$, define
\[
 \operatorname{inc}_R(U)=\left|\{e\in R:e\cap U\neq\emptyset\}\right|
\]
and
\[
 \Delta_R(U)=\frac{|U|m_R}{4}-\operatorname{inc}_R(U).
\]
We call $U$ \emph{deficient} if $\Delta_R(U)>0$.
```

Expanded PaperInterface specification

```
∀ (Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item) (chores : Finset Item),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      (∀ item ∈ chores, AppliedModelingLib.FairDivision.IsSmallForExactlyTwo cost item) →
        ∃ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
          AppliedModelingLib.FairDivision.IsAllocationOf allocation chores ∧
            AppliedModelingLib.FairDivision.EFXForChores (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                allocation ∧
              ((¬∃ (dominant : Finset (Fin 4)) (auxiliary : Finset (Fin 4)) (q : ℕ),
                    dominant.card = 2 ∧
                      auxiliary.card = 2 ∧
                        Disjoint dominant auxiliary ∧
                          ((∀ item ∈ chores, AppliedModelingLib.FairDivision.smallAgentSet cost item = dominant) ∧
                              chores.card = 4 * q + 3 ∨
                            ∃ exceptionalItem ∈ chores,
                              AppliedModelingLib.FairDivision.smallAgentSet cost exceptionalItem = auxiliary ∧
                                (∀ item ∈ chores.erase exceptionalItem,
                                    AppliedModelingLib.FairDivision.smallAgentSet cost item = dominant) ∧
                                  (chores.erase exceptionalItem).card = 4 * q + 3)) →
                  (∀ (i : Fin 4),
                      (∃ item ∈ allocation i, AppliedModelingLib.FairDivision.IsSmallChore cost i item) →
                        ∀ (j : Fin 4),
                          AppliedModelingLib.FairDivision.additiveChoreCost cost i (allocation i) - 1 ≤
                            AppliedModelingLib.FairDivision.additiveChoreCost cost i (allocation j)) ∧
                    ∀ (i : Fin 4),
                      (∀ item ∈ allocation i, AppliedModelingLib.FairDivision.IsLargeChore cost r i item) →
                        ∀ (j : Fin 4),
                          AppliedModelingLib.FairDivision.additiveChoreCost cost i (allocation i) ≤
                            AppliedModelingLib.FairDivision.additiveChoreCost cost i (allocation j)) ∧
                (∀ (firstType secondType : Finset (Fin 4)),
                    firstType.card = 2 →
                      secondType.card = 2 →
                        Disjoint firstType secondType →
                          (∀ item ∈ chores,
                              AppliedModelingLib.FairDivision.smallAgentSet cost item = firstType ∨
                                AppliedModelingLib.FairDivision.smallAgentSet cost item = secondType) →
                            {item ∈ chores |
                                AppliedModelingLib.FairDivision.smallAgentSet cost item = secondType}.card ≥
                              {item ∈ chores |
                                AppliedModelingLib.FairDivision.smallAgentSet cost item = firstType}.card →
                            ∀ agent ∈ firstType,
                              ∃ (preferredAllocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
                                AppliedModelingLib.FairDivision.IsAllocationOf preferredAllocation chores ∧
                                  AppliedModelingLib.FairDivision.EFXForChores
                                      (AppliedModelingLib.FairDivision.additiveChoreCost cost) preferredAllocation ∧
                                    ∀ (other : Fin 4),
                                      AppliedModelingLib.FairDivision.additiveChoreCost cost agent
                                          (preferredAllocation agent) ≤
                                        AppliedModelingLib.FairDivision.additiveChoreCost cost agent
                                          (preferredAllocation other)) ∧
                  ∀ (dominant auxiliary : Finset (Fin 4)) (q : ℕ),
                    dominant.card = 2 →
                      auxiliary.card = 2 →
                        Disjoint dominant auxiliary →
                          ∀ special ∈ auxiliary,
                            ∀ companion ∈ auxiliary,
                              special ≠ companion →
                                ((∀ item ∈ chores, AppliedModelingLib.FairDivision.smallAgentSet cost item = dominant) ∧
                                    chores.card = 4 * q + 3 ∨
                                  ∃ exceptionalItem ∈ chores,
                                    AppliedModelingLib.FairDivision.smallAgentSet cost exceptionalItem = auxiliary ∧
                                      (∀ item ∈ chores.erase exceptionalItem,
                                          AppliedModelingLib.FairDivision.smallAgentSet cost item = dominant) ∧
                                        (chores.erase exceptionalItem).card = 4 * q + 3) →
                                  ∃ (exceptionalAllocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
                                    AppliedModelingLib.FairDivision.IsAllocationOf exceptionalAllocation chores ∧
                                      AppliedModelingLib.FairDivision.EFXForChores
                                          (AppliedModelingLib.FairDivision.additiveChoreCost cost)
                                          exceptionalAllocation ∧
                                        (∀ (agent : Fin 4),
                                            agent ≠ special →
                                              ∀ (other : Fin 4),
                                                AppliedModelingLib.FairDivision.additiveChoreCost cost agent
                                                    (exceptionalAllocation agent) -
                                                  1 ≤
                                                AppliedModelingLib.FairDivision.additiveChoreCost cost agent
                                                  (exceptionalAllocation other)) ∧
                                          (∀ item ∈ exceptionalAllocation special,
                                              AppliedModelingLib.FairDivision.IsLargeChore cost r special item) ∧
                                            (∀ (other : Fin 4),
                                                other ≠ companion →
                                                  AppliedModelingLib.FairDivision.additiveChoreCost cost special
                                                      (exceptionalAllocation special) ≤
                                                    AppliedModelingLib.FairDivision.additiveChoreCost cost special
                                                      (exceptionalAllocation other)) ∧
                                              AppliedModelingLib.FairDivision.additiveChoreCost cost special
                                                  (exceptionalAllocation special) -
                                                r ≤
                                                AppliedModelingLib.FairDivision.additiveChoreCost cost special
                                                  (exceptionalAllocation companion) ∧
                                              (((∀ item ∈ chores,
                                                    AppliedModelingLib.FairDivision.smallAgentSet cost item =
                                                      dominant) ∧
                                                  chores.card = 4 * q + 3 ∧
                                                    ∀ (agent : Fin 4),
                                                      Finset.card (exceptionalAllocation agent) =
                                                        if agent = companion then q else q + 1) ∨
```

```
∃ exceptionalItem ∈ chores,
  AppliedModelingLib.FairDivision.smallAgentSet cost exceptionalItem =
    auxiliary ∧
   (∀ item ∈ chores.erase exceptionalItem,
      AppliedModelingLib.FairDivision.smallAgentSet cost item =
        dominant) ∧
     (chores.erase exceptionalItem).card = 4 * q + 3 ∧
      exceptionalItem ∈ exceptionalAllocation companion ∧
       ∀ (agent : Fin 4),
         Finset.card (exceptionalAllocation agent) = q + 1)
```

Lean proof endpoint (built separately)

HT26EFXChores.m2_efx_allocation_properties

Recorded source-to-Spec screening **Verdict:** matches

Reason: Encodes the ordinary-case small-removal/envy-free guarantees, both relabelled exceptional forms, selectable exceptional agent, and two-type preferred-agent clause.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 16

Source locator: cited publication, lines 1709-1712

Verbatim source input

\begin{lemma}\label{lem:easycombine}
Suppose $P=(P_1,P_2,P_3,P_4)$ is a canonical allocation of M_{01}. Suppose some items of \(M_2\) are used for gap-filling, all assigned to short agents, such that \(P\) is extended to an envy-free allocation.
If the residual of M_2 is of an exceptional configuration with exceptional agents i and j and both agents are long agents in the canonical allocation, then there exists an EFX allocation for $M_{01}\cup M_2$.
\end{lemma}

[Next verbatim source excerpt]

For each agent i, let
\[
 n_i=\left|\{g\in M_{01}:S(g)=\{i\}\}\right|
\]
be the number of M_{01} items uniquely small for i. Items in M_{01}
with $S(g)=\emptyset$ are all-large items.
Let $|M_{01}|=4a+b$ with $b\in\{0,1,2,3\}$.
As mentioned before, we will make sure each agent receives either a items or $a+1$ items.

Fix quotas $q_i\in\{a,a+1\}$ with $\sum_i q_i=|M_{01}|$.
We consider allocations of M_{01} with quotas $q=(q_1,q_2,q_3,q_4)$ in which each agent i first receives as many of her uniquely-small items as possible, up to quota q_i, and the remaining slots are filled with the remaining items.
We will show that all such allocations satisfy EFX.
It is also important that we have the freedom to choose which agents receive a items and which agents receive $a+1$ items.
This freedom enables us to append the allocation of M_2 after the allocation of M_{01}.

We begin by defining this type of allocations.

[Next verbatim source excerpt]

\subsection{The M_2 Multigraph Algorithm}
\label{subsec:2}
We now give a complete allocation procedure for the items in M_2.
Each item is small for some agents i and j and
large for the other two agents, and we say that the type of this item is (i,j), or ij for short.
In this section, we consider the case where the instance only has M_{2} items.
We represent the instance by a
multigraph R on $N=\{1,2,3,4\}$, where an item of type (i,j) is an edge (i,j).
We slightly abuse the notation by letting R also represent the edge set, i.e., R exactly corresponds to items in M_2.
Let x_{ij} denote the multiplicity of edge (i,j) and let $m_R=|R|$.

For $U\subseteq N$, define
\[
 \operatorname{inc}_R(U)=\left|\{e\in R:e\cap U\neq\emptyset\}\right|
\]
and
\[
 \Delta_R(U)=\frac{|U|m_R}{4}-\operatorname{inc}_R(U).
\]
We call U \emph{deficient} if $\Delta_R(U)>0$.

[Next verbatim source excerpt]

\subsection{Concatenating the M_{01} prefix and the M_2 allocation}
\label{subsec:combine}
We now finish the proof for $M_{01}\cup M_2$. Write
\[
 |M_{01}|=4a+b,
 \qquad b\in\{0,1,2,3\}.
\]
We discuss the four values of b.
In each case and each of the sub-case, we will discuss how the concatenation is done and show that the resultant allocation is EFX.
We will exploit many features discussed earlier: the freedom to choose those agents that receive one more item when allocating M_{01}, the freedom to choose envy-free agents suggested in Lemma~\ref{lem:M2properties}, etc.

At a high level, we will try to use some items in M_2 for gap-filling such that the allocation of M_{01} in the first phase becomes envy-free after gap-filling.
If the residual of M_2 is not of the two configurations in Lemma~\ref{lem:M2properties}(ii), we can allocate this residue so that every envy is up to a small item.
Lemma~\ref{lem:composition} then implies the combined allocation is EFX.
The tricky part is to try to make the residue of M_2 after gap-filling avoid the two configurations in Lemma~\ref{lem:M2properties}(ii).
However, there are hard cases where this is not always possible, and we need to handle these cases separately.

If, after the gap-filling, the residual M_2 items form one of the two configurations in Lemma~\ref{lem:M2properties}(ii), we will call this configuration \emph{exceptional}.
In an exceptional configuration, one edge (i,j) has multiplicity at least 3, and, other than those (i,j)-items, there can be at most one other item of type (i',j') where (i',j') is disjoint from (i,j).
We will call the two agents i',j' \emph{exceptional agents}.
By Lemma~\ref{lem:M2properties}(ii), for the two exceptional agents i',j', one of them receives only large item in the residual of M_2 and envies the other by a large item.
We call this agent the \emph{disadvantageous agent}.
Lemma~\ref{lem:M2properties}(ii) tells us we are free to choose the disadvantageous agent from the two exceptional agents.

[Next verbatim source excerpt]

\begin{lemma}[Properties of the EFX allocations in Lemma~\ref{lem:M2}]\label{lem:M2properties}
The EFX allocation $B=(B_1,\ldots,B_4)$ ensured by Lemma~\ref{lem:M2} can be chosen to satisfy the

following additional properties.
\begin{enumerate}[label=(\roman*)]
\item Except in the two configurations described in~(ii), every possible envy is certified by removing a small item. More precisely,
whenever B_i contains an item that is small for agent i,
\[
 c_i(B_i)-1\le c_i(B_j)\qquad\forall j\in N,
\]
whereas every agent whose bundle contains only large items is envy-free:
\[
 c_i(B_i)\le c_i(B_j)\qquad\forall j\in N.
\]

\item Up to relabeling, the only configurations in which an agent whose
bundle contains only large items may fail to be envy-free, and hence
must use a large item as the EFX-removal item, are the following.
\begin{enumerate}[label=(\alph*)]
\item All items have type $(3,4)$ and
\[
 |R|=4q+3.
\]
Then agents $1,2$ receive q and $q+1$ items, respectively, in either
order, while agents $3,4$ receive $q+1$ items each. We are free to choose
which of agents $1,2$ receives $q+1$ items, and this agent is the unique agent who use a large item as the EFX-removal item.

\item The multiset R consists of one item of type $(1,2)$ and
$4q+3$ items of type $(3,4)$. One of agents $1,2$ receives the unique
type-$(1,2)$ item together with q type-$(3,4)$ items, the other receives
$q+1$ type-$(3,4)$ items, and agents $3,4$ receive $q+1$ type-$(3,4)$
items each. We are free to choose which of agents $1,2$ receives the unique
type-$(1,2)$ item. The other agent is the unique agent who must use a large
item as the EFX-removal item.
\end{enumerate}

\item If every edge of R has type $(1,2)$ or $(3,4)$, with

[Next verbatim source excerpt]

If, after the gap-filling, the residual M_2 items form one of the two configurations in Lemma~\ref{lem:M2properties}(ii), we will call this configuration
↪ \emph{exceptional}.
In an exceptional configuration, one edge (i,j) has multiplicity at least 3, and, other than those (i,j)-items, there can be at most one other item of
↪ type (i',j') where (i',j') is disjoint from (i,j).
We will call the two agents i',j' \emph{exceptional agents}.
By Lemma~\ref{lem:M2properties}(ii), for the two exceptional agents i',j', one of them receives only large item in the residual of M_2 and envies the
↪ other by a large item.
We call this agent the \emph{disadvantageous agent}.
Lemma~\ref{lem:M2properties}(ii) tells us we are free to choose the disadvantageous agent from the two exceptional agents.

Expanded PaperInterface specification

```
∀ (Item : Type) (r : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin 4) Item)
  (prefixChores m2Chores gapChores residueChores : Finset Item) (a : ℕ) (quota : Fin 4 → ℕ)
  (prefixAllocation gap : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
  2 < r →
    AppliedModelingLib.FairDivision.IsOneOrRChoreCost cost r →
      Disjoint prefixChores m2Chores →
        (∀ item ∈ m2Chores, AppliedModelingLib.FairDivision.IsSmallForExactlyTwo cost item) →
          Disjoint gapChores residueChores →
            gapChores ∪ residueChores = m2Chores →
              AppliedModelingLib.FairDivision.IsCanonicalSmallChoreAllocation cost prefixChores quota prefixAllocation →
                (∀ item ∈ prefixChores, AppliedModelingLib.FairDivision.IsSmallForAtMostOne cost item) →
                  (∀ (agent : Fin 4), quota agent = a ∨ quota agent = a + 1) →
                    AppliedModelingLib.FairDivision.IsAllocationOf gap gapChores →
                      (∀ (agent : Fin 4), quota agent ≠ a → gap agent = ∅) →
                        (AppliedModelingLib.FairDivision.EnvyFreeForChores
                           (AppliedModelingLib.FairDivision.additiveChoreCost cost) fun (agent : Fin 4) =>
                           prefixAllocation agent ∪ gap agent) →
                          ∀ (exceptionalI exceptionalJ : Fin 4),
                            (∃ (dominant : Finset (Fin 4)) (auxiliary : Finset (Fin 4)) (q : ℕ),
                               dominant.card = 2 ∧
                                 auxiliary.card = 2 ∧
                                   Disjoint dominant auxiliary ∧
                                     exceptionalI ∈ auxiliary ∧
                                       exceptionalJ ∈ auxiliary ∧
                                         ((∀ item ∈ residueChores,
                                              AppliedModelingLib.FairDivision.smallAgentSet cost item = dominant) ∧
                                            residueChores.card = 4 * q + 3 ∨
                                          ∃ exceptionalItem ∈ residueChores,
                                            AppliedModelingLib.FairDivision.smallAgentSet cost exceptionalItem =
                                              auxiliary ∧
                                              (∀ item ∈ residueChores.erase exceptionalItem,
                                                 AppliedModelingLib.FairDivision.smallAgentSet cost item =
                                                   dominant) ∧
                                              (residueChores.erase exceptionalItem).card = 4 * q + 3)) →
                              exceptionalI ≠ exceptionalJ →
                                quota exceptionalI = a + 1 →
                                  quota exceptionalJ = a + 1 →
                                    ∃ (allocation : AppliedModelingLib.FairDivision.Allocation (Fin 4) Item),
                                      AppliedModelingLib.FairDivision.IsAllocationOf allocation
                                        (prefixChores ∪ m2Chores) ∧
                                        AppliedModelingLib.FairDivision.EFXForChores
                                          (AppliedModelingLib.FairDivision.additiveChoreCost cost) allocation
```

Lean proof endpoint (built separately)

HT26EFXChores.exceptional_residue_combination

Recorded source-to-Spec screening **Verdict:** matches

Reason: Formalizes the canonical prefix, short-only envy-free gap filling, the two exceptional residual forms, two long exceptional agents, and EFX completion.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Appendix and supplement source claims

Review row 17

Source locator: cited publication, lines 2074-2076

Verbatim source input

```
\begin{proposition}
   In an EFX allocation, no bundle contains two or more items from $A$.
\end{proposition}

[Next verbatim source excerpt]

In this section, we prove Theorem~\ref{thm:tri} for an arbitrary fixed $n\geq 4$.

\subsection{The construction}
Fix an integer $n\ge 4$. Partition the agents into two groups $T_1$ and $T_2$ such that
\[
   |T_1|=\left\lfloor \frac n2\right\rfloor,
   \qquad
   |T_2|=\left\lceil \frac n2\right\rceil.
\]
Let $t_1=|T_1|$ and $t_2=|T_2|$.
Let $s=2t_2+1$.
The instance contains $n-1+2s$ items that are partitioned into three groups $A,B,C$ where $A$ contains $n-1$ items and each of $B$ and $C$ contains
↪   $s$ items.

The three values in the tri-valued instances are given by
$$p=1,\quad q=s+2=2t_2+3,\quad\mbox{and}\quad r=\frac{s(q+1)}2=(2t_2+1)(t_2+2).$$


The costs are defined as follows. For every agent $i\in T_1$,
\[
   c_i(a)=r \quad(a\in A),
   \qquad
   c_i(b)=p \quad(b\in B),
   \qquad
   c_i(c)=q \quad(c\in C),
\]
and for every agent $i\in T_2$,
\[
   c_i(a)=r \quad(a\in A),
   \qquad
   c_i(b)=q \quad(b\in B),
   \qquad
   c_i(c)=p \quad(c\in C).
\]

For $n=4$, we have $t_1=t_2=2$, $s=5$, $q=7$, and $r=20$.
This resembles our construction in Sect.~\ref{sec:tri}.

\subsection{Non-existence of EFX allocations}
Let $X=(X_1,\ldots,X_n)$ be an arbitrary allocation. For a bundle $Y\subseteq A\cup B\cup C$, write
\[
   a(Y)=|Y\cap A|,
   \qquad
   b(Y)=|Y\cap B|,
   \qquad
   c(Y)=|Y\cap C|.
\]
All agents in $T_1$ have the same cost for a bundle $Y$, namely
\[
   P_1(Y)=r\cdot a(Y)+b(Y)+q\cdot c(Y),
\]
and all agents in $T_2$ have the same cost for $Y$, namely
\[
   P_2(Y)=r\cdot  a(Y)+q\cdot b(Y)+c(Y).
\]
Define
\[
   p_1=\min_{i\in \{1,\ldots,n\}} P_1(X_i),
   \qquad\mbox{and}\qquad
   p_2=\min_{i\in \{1,\ldots,n\}} P_2(X_i).
\]
```

Expanded PaperInterface specification

```
∀ (n : ℕ) (Item : Type) (r q : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin n) Item)
  (chores A B C : Finset Item) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) Item),
  4 ≤ n →
    q = ↑(2 * ((n + 1) / 2) + 1) + 2 →
      r = ↑(2 * ((n + 1) / 2) + 1) * (q + 1) / 2 →
        A.card = n - 1 →
          B.card = 2 * ((n + 1) / 2) + 1 →
            C.card = 2 * ((n + 1) / 2) + 1 →
              Disjoint A B →
                Disjoint A C →
                  Disjoint B C →
                    A ∪ B ∪ C = chores →
                      (∀ (agent : Fin n) (item : Item),
```

```
  (item ∈ A → cost agent item = r) ∧
    (item ∈ B → cost agent item = if ↑agent < n / 2 then 1 else q) ∧
      (item ∈ C → cost agent item = if ↑agent < n / 2 then q else 1)) →
AppliedModelingLib.FairDivision.IsAllocationOf allocation chores →
  AppliedModelingLib.FairDivision.EFXForChores
      (AppliedModelingLib.FairDivision.additiveChoreCost cost) allocation →
    ∀ (agent : Fin n), Finset.card (allocation agent ∩ A) ≤ 1
```

Lean proof endpoint (built separately)

HT26EFXChores.appendix_no_two_a_items

Recorded source-to-Spec screening **Verdict:** matches

Reason: Exact Appendix construction, EFX allocation, and at-most-one-A-item conclusion.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 18

Source locator: cited publication, lines 2104-2106

Verbatim source input

```
\begin{proposition}
   In an EFX allocation, $p_1\ge r$ and $p_2\ge r$.
\end{proposition}

[Next verbatim source excerpt]

In this section, we prove Theorem~\ref{thm:tri} for an arbitrary fixed $n\geq 4$.

\subsection{The construction}
Fix an integer $n\ge 4$. Partition the agents into two groups $T_1$ and $T_2$ such that
\[
   |T_1|=\left\lfloor \frac n2\right\rfloor,
   \qquad
   |T_2|=\left\lceil \frac n2\right\rceil.
\]
Let $t_1=|T_1|$ and $t_2=|T_2|$.
Let $s=2t_2+1$.
The instance contains $n-1+2s$ items that are partitioned into three groups $A,B,C$ where $A$ contains $n-1$ items and each of $B$ and $C$ contains
↪   $s$ items.

The three values in the tri-valued instances are given by
$$p=1,\quad q=s+2=2t_2+3,\quad\mbox{and}\quad r=\frac{s(q+1)}2=(2t_2+1)(t_2+2).$$


The costs are defined as follows. For every agent $i\in T_1$,
\[
   c_i(a)=r \quad(a\in A),
   \qquad
   c_i(b)=p \quad(b\in B),
   \qquad
   c_i(c)=q \quad(c\in C),
\]
and for every agent $i\in T_2$,
\[
   c_i(a)=r \quad(a\in A),
   \qquad
   c_i(b)=q \quad(b\in B),
   \qquad
   c_i(c)=p \quad(c\in C).
\]

For $n=4$, we have $t_1=t_2=2$, $s=5$, $q=7$, and $r=20$.
This resembles our construction in Sect.~\ref{sec:tri}.

\subsection{Non-existence of EFX allocations}
Let $X=(X_1,\ldots,X_n)$ be an arbitrary allocation. For a bundle $Y\subseteq A\cup B\cup C$, write
\[
   a(Y)=|Y\cap A|,
   \qquad
   b(Y)=|Y\cap B|,
   \qquad
   c(Y)=|Y\cap C|.
\]
All agents in $T_1$ have the same cost for a bundle $Y$, namely
\[
   P_1(Y)=r\cdot a(Y)+b(Y)+q\cdot c(Y),
\]
and all agents in $T_2$ have the same cost for $Y$, namely
\[
   P_2(Y)=r\cdot  a(Y)+q\cdot b(Y)+c(Y).
\]
Define
\[
   p_1=\min_{i\in \{1,\ldots,n\}} P_1(X_i),
   \qquad\mbox{and}\qquad
   p_2=\min_{i\in \{1,\ldots,n\}} P_2(X_i).
\]
```

Expanded PaperInterface specification

```
∀ (n : ℕ) (Item : Type) (r q : ℝ) (cost : AppliedModelingLib.FairDivision.ChoreCost (Fin n) Item)
  (chores A B C : Finset Item) (allocation : AppliedModelingLib.FairDivision.Allocation (Fin n) Item)
  (P1 P2 : AppliedModelingLib.FairDivision.Bundle Item → ℝ) (p1 p2 : ℝ),
  4 ≤ n →
    q = ↑(2 * ((n + 1) / 2) + 1) + 2 →
      r = ↑(2 * ((n + 1) / 2) + 1) * (q + 1) / 2 →
        A.card = n - 1 →
          B.card = 2 * ((n + 1) / 2) + 1 →
            C.card = 2 * ((n + 1) / 2) + 1 →
              Disjoint A B →
                Disjoint A C →
                  Disjoint B C →
                    A ∪ B ∪ C = chores →
                      (∀ (agent : Fin n) (item : Item),
                          (item ∈ A → cost agent item = r) ∧
                            (item ∈ B → cost agent item = if ↑agent < n / 2 then 1 else q) ∧
```

```
      (item ∈ C → cost agent item = if ↑agent < n / 2 then q else 1)) →
AppliedModelingLib.FairDivision.IsAllocationOf allocation chores →
  AppliedModelingLib.FairDivision.EFXForChores
      (AppliedModelingLib.FairDivision.additiveChoreCost cost) allocation →
    (∀ (bundle : AppliedModelingLib.FairDivision.Bundle Item),
       P1 bundle =
         r * ↑(Finset.card (bundle ∩ A)) + ↑(Finset.card (bundle ∩ B)) +
           q * ↑(Finset.card (bundle ∩ C))) →
     (∀ (bundle : AppliedModelingLib.FairDivision.Bundle Item),
        P2 bundle =
          r * ↑(Finset.card (bundle ∩ A)) + q * ↑(Finset.card (bundle ∩ B)) +
            ↑(Finset.card (bundle ∩ C))) →
      (∀ (agent : Fin n),
         ↑agent < n / 2 →
           AppliedModelingLib.FairDivision.additiveChoreCost cost agent = P1) →
       (∀ (agent : Fin n),
          n / 2 ≤ ↑agent →
            AppliedModelingLib.FairDivision.additiveChoreCost cost agent = P2) →
        (∃ (agent : Fin n), P1 (allocation agent) = p1) →
          (∀ (agent : Fin n), p1 ≤ P1 (allocation agent)) →
            (∃ (agent : Fin n), P2 (allocation agent) = p2) →
              (∀ (agent : Fin n), p2 ≤ P2 (allocation agent)) → p1 ≥ r ∧ p2 ≥ r
```

Lean proof endpoint (built separately)

HT26EFXChores.appendix_p1_p2_lower_bounds

Recorded source-to-Spec screening **Verdict:** matches

Reason: Encodes the stated P1/P2 definitions and concludes p1,p2 are each at least r.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation