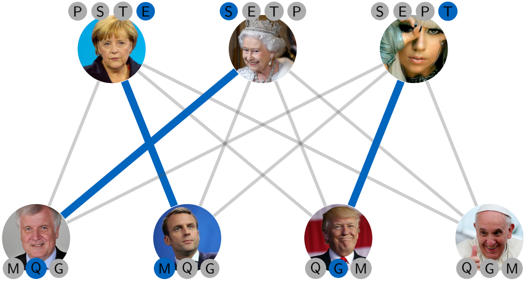
# Stable Matchings

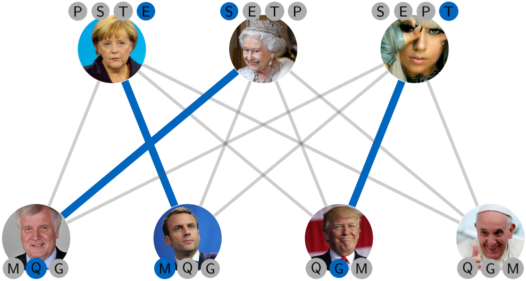
Preferences of market participants are expressed by total orders (preference lists) instead of numerically via costs.

# Matching with Preferences




Preference list: For each node $v \in V$, a total order $\prec_v$ of the neighbors $N_G(v)$ is given.

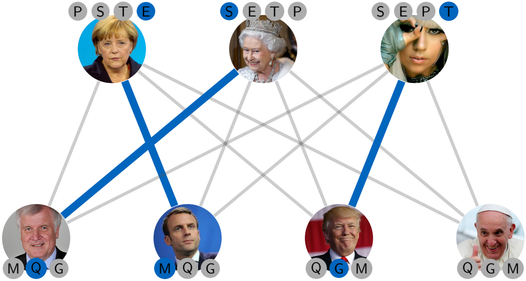
# Matching with Preferences




Preference list: For each node $v \in V$, a total order $\prec_v$ of the neighbors $N_G(v)$ is given.

Assumption: $u \prec_v \emptyset$ for all $u \in N_G(v)$

# Matching with Preferences



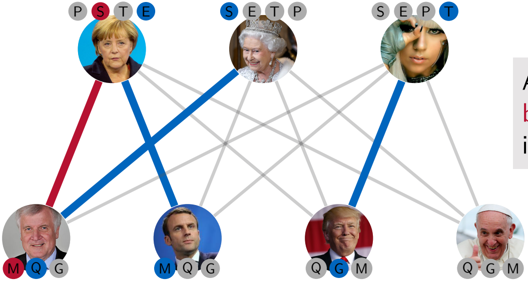

Preference list: For each node $v \in V$, a total order $\prec_v$ of the neighbors $N_G(v)$ is given.

Assumption: $u \prec_v \emptyset$ for all $u \in N_G(v)$

Represent matching $M$ as a mapping $\mu : V \to V \cup \emptyset$ with

$$\mu(v) := \begin{cases} \emptyset & \text{if } v \text{ is not covered by } M, \\ u & \text{if } \{u, v\} \in M. \end{cases}$$

# Matching with Preferences




An edge $\{u, v\} \in E \setminus M$ is called blocking (w.r.t. $M$), if $u \prec_v \mu(v)$ und $v \prec_u \mu(u)$.

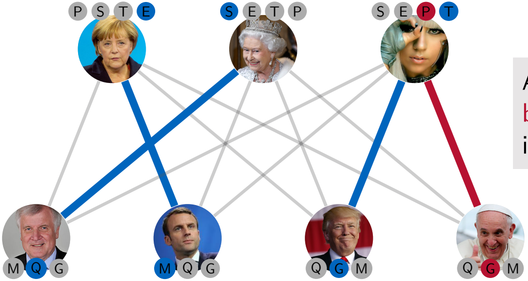
Preference list: For each node $v \in V$, a total order $\prec_v$ of the neighbors $N_G(v)$ is given.

Assumption: $u \prec_v \emptyset$ for all $u \in N_G(v)$

Represent matching $M$ as a mapping $\mu : V \to V \cup \emptyset$ with

$$\mu(v) := \begin{cases} \emptyset & \text{if } v \text{ is not covered by } M, \\ u & \text{if } \{u, v\} \in M. \end{cases}$$

# Matching with Preferences




An edge $\{u, v\} \in E \setminus M$ is called blocking (w.r.t. $M$), if $u \prec_v \mu(v)$ und $v \prec_u \mu(u)$.

Preference list: For each node $v \in V$, a total order $\prec_v$ of the neighbors $N_G(v)$ is given.

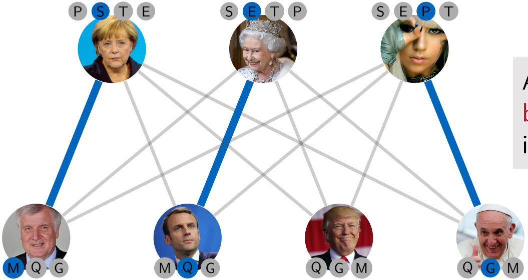
Assumption: $u \prec_v \emptyset$ for all $u \in N_G(v)$

Represent matching $M$ as a mapping $\mu : V \to V \cup \emptyset$ with

$$\mu(v) := \begin{cases} \emptyset & \text{if } v \text{ is not covered by } M, \\ u & \text{if } \{u, v\} \in M. \end{cases}$$

Universität Bremen

# Matching with Preferences




An edge $\{u, v\} \in E \setminus M$ is called blocking (w.r.t. $M$), if $u \prec_v \mu(v)$ und $v \prec_u \mu(u)$.

Preference list: For each node $v \in V$, a total order $\prec_v$ of the neighbors $N_G(v)$ is given.

Assumption: $u \prec_v \emptyset$ for all $u \in N_G(v)$

**Definition.** A matching $M$ is stable, if there is no blocking edge w.r.t. $M$.

# Stable Matching Problem

## Stable Matching Problem

- Input: Graph $G = (V, E)$; every $v \in V$ has a preference list over their neighbors $N_G(v)$.
- Task: Compute a stable matching.

# Stable Matching Problem

## Stable Matching Problem

- Input: Graph $G = (V, E)$; every $v \in V$ has a preference list over their neighbors $N_G(v)$.
- Task: Compute a stable matching.

**Typical Questions**

(1) Does there always exist a stable matching?

(2) Can it be computed in polynomial time?

(3) If there are more, which one is the best?

# Stable Matching Problem

## Stable Matching Problem

- Input: Graph $G = (V, E)$; every $v \in V$ has a preference list over their neighbors $N_G(v)$.
- Task: Compute a stable matching.

### Typical Questions

(1) Does there always exist a stable matching?

Yes, for bipartite graphs.

(2) Can it be computed in polynomial time?

(3) If there are more, which one is the best?

# Stable Matching Problem

## Stable Matching Problem

- Input: Graph $G = (V, E)$; every $v \in V$ has a preference list over their neighbors $N_G(v)$.
- Task: Compute a stable matching.

## Typical Questions

(1) Does there always exist a stable matching?

Yes, for bipartite graphs.

(2) Can it be computed in polynomial time?

Yes, for bipartite graphs.

(3) If there are more, which one is the best?

# Stable Matching Problem

### Stable Matching Problem

- Input: Graph $G = (V, E)$; every $v \in V$ has a preference list over their neighbors $N_G(v)$.
- Task: Compute a stable matching.

**Typical Questions**

(1) Does there always exist a stable matching?

Yes, for bipartite graphs.

(2) Can it be computed in polynomial time?

Yes, for bipartite graphs.

(3) If there are more, which one is the best?

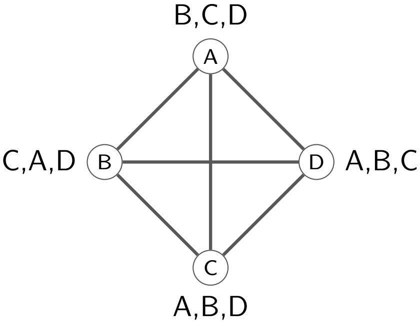
Depends for whom...

# Stable Matching Problem

- In bipartite graphs $G = (A \cup B, E)$ often called stable marriage problem. Usually $|A| = |B|$, so we are looking for a perfect stable matching.

# Stable Matching Problem

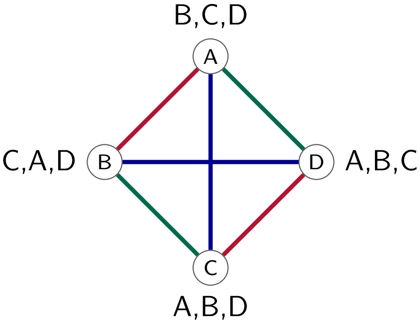
- In bipartite graphs $G = (A \cup B, E)$ often called stable marriage problem. Usually $|A| = |B|$, so we are looking for a perfect stable matching.
- In general graphs also called stable roommate problem.




There is not always a perfect stable matching.

# Stable Matching Problem

- In bipartite graphs $G = (A \cup B, E)$ often called stable marriage problem. Usually $|A| = |B|$, so we are looking for a perfect stable matching.
- In general graphs also called stable roommate problem.

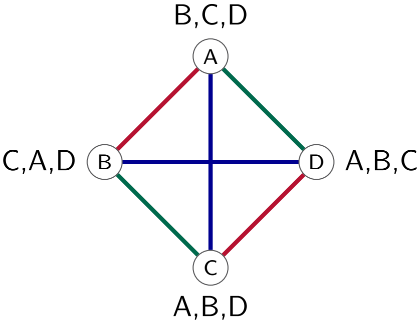



There is not always a perfect stable matching.

red: B,C

# Stable Matching Problem

- In bipartite graphs $G = (A \cup B, E)$ often called stable marriage problem. Usually $|A| = |B|$, so we are looking for a perfect stable matching.
- In general graphs also called stable roommate problem.



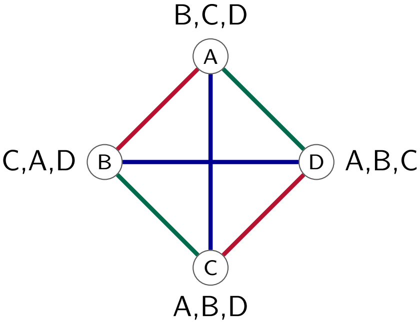

There is not always a perfect stable matching.

red: B,C

blue: A,B

# Stable Matching Problem

- In bipartite graphs $G = (A \cup B, E)$ often called stable marriage problem. Usually $|A| = |B|$, so we are looking for a perfect stable matching.
- In general graphs also called stable roommate problem.




There is not always a perfect stable matching.

red: B,C

blue: A,B

green: C,A

# Stable Matchings in (complete) bipartite graphs

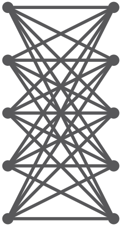
stable marriage problem

# Approach 1: Enumeration

- Algorithm: Consider all possible (maximum) matchings and check every matching for stability.
  - How to find? How to check?
- Advantage: The algorithm is correct.
- Disadvantage: There are too many perfect matching ($n!$), hence the algorithm is not efficient.

# Approach 1: Enumeration

- Algorithm: Consider all possible (maximum) matchings and check every matching for stability.
  - How to find? How to check?
- Advantage: The algorithm is correct.
- Disadvantage: There are too many perfect matching ($n!$), hence the algorithm is not efficient.

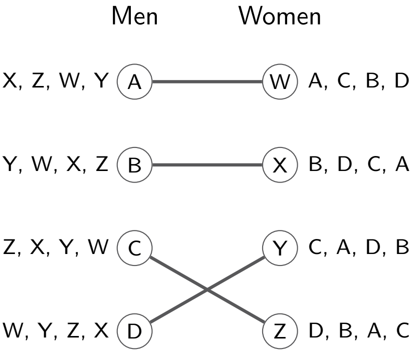


**Example**

A complete bipartite graph with 5 vertices on each side has $5! = 5 * 4 * 3 * 2 * 1 = 120$ distinct perfect matchings.

# Approach 2: Greedy

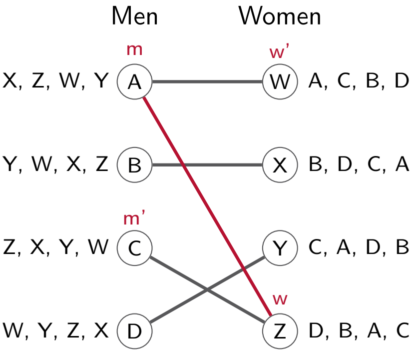
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




Universität Bremen

# Approach 2: Greedy

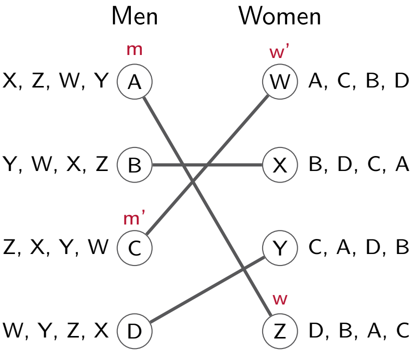
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{A,Z} unhappy

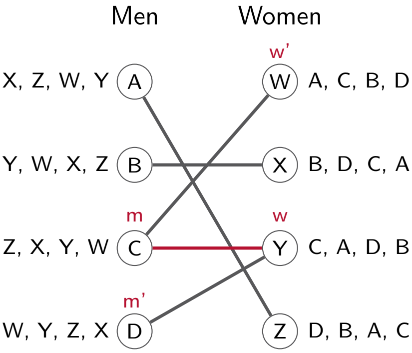
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

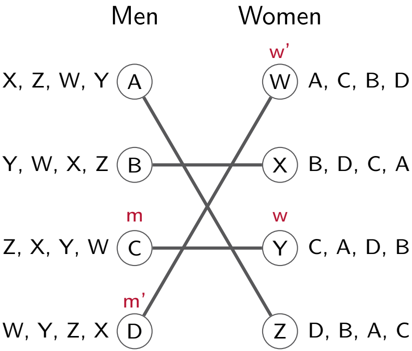
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{C,Y} unhappy

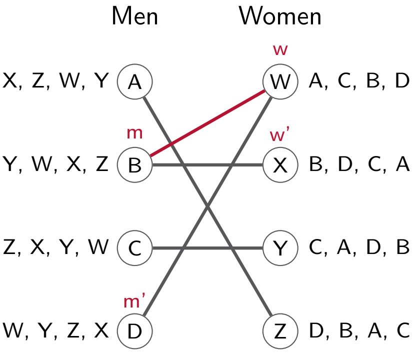
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

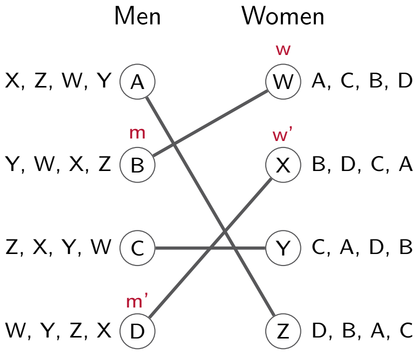
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{B,W} unhappy

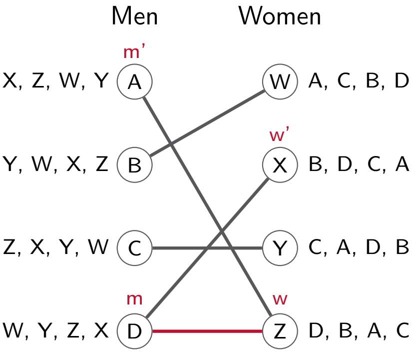
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

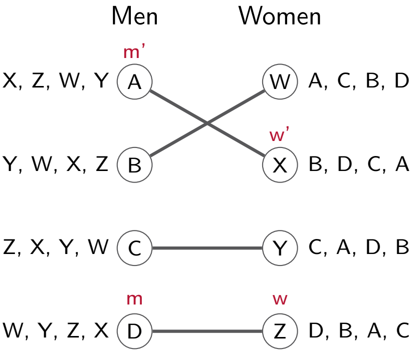
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{D,Z} unhappy

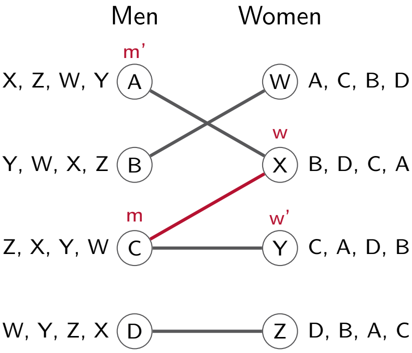
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

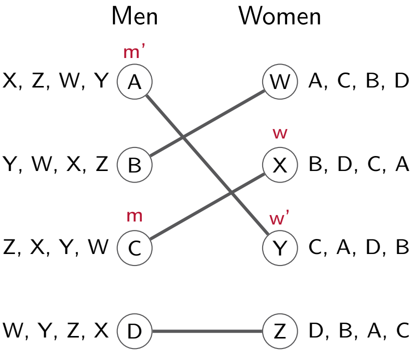
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{C,X} unhappy

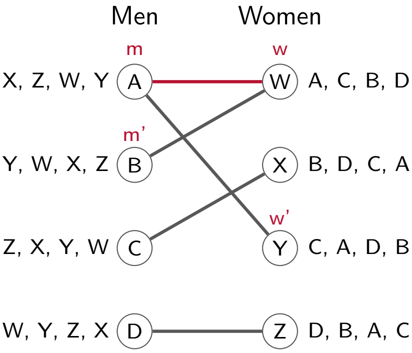
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

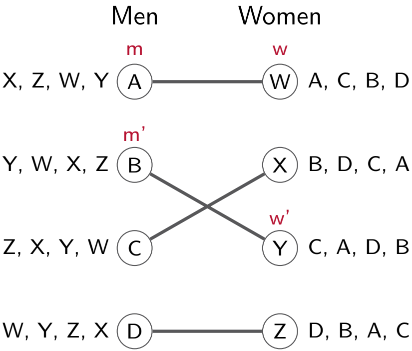
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{A,W} unhappy

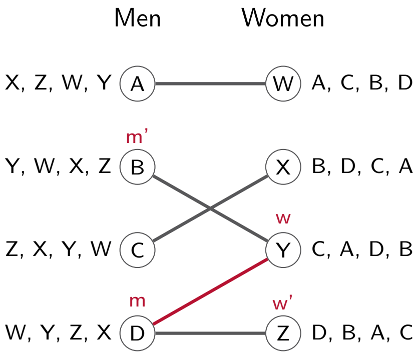
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

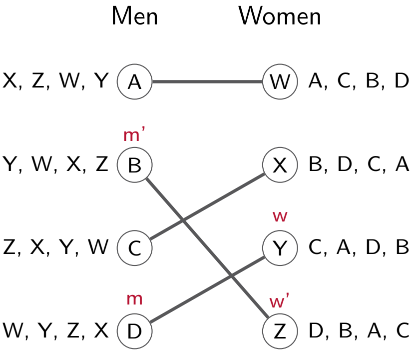
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{D,Y} unhappy

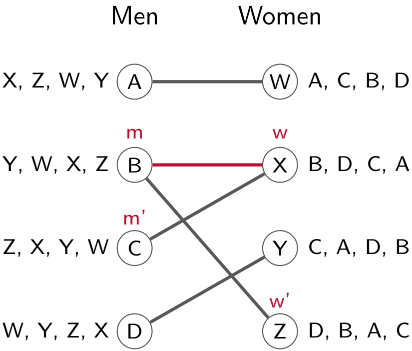
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

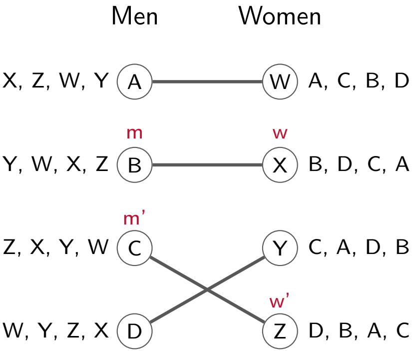
**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




{B,X} unhappy

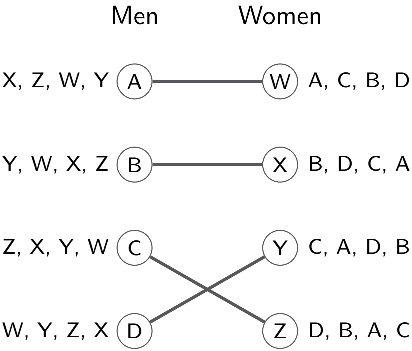
# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.



# Approach 2: Greedy

**Greedy**: Given a matching $M$. As long as there is a blocking edge $\{m, w\} \notin M$, exchange the matching edges $\{m, w'\}$ and $\{m', w\}$ by $\{m, w\}$ and $\{m', w'\}$.




Alert: We are again at the beginning! Infinite loop!

# Gale-Shapley Algorithm (1962)

(aka Deferred-Acceptance Algorithm)

Originally for the more general College Admission Problem.

# 3rd Approach: Gale-Shapley Algorithm

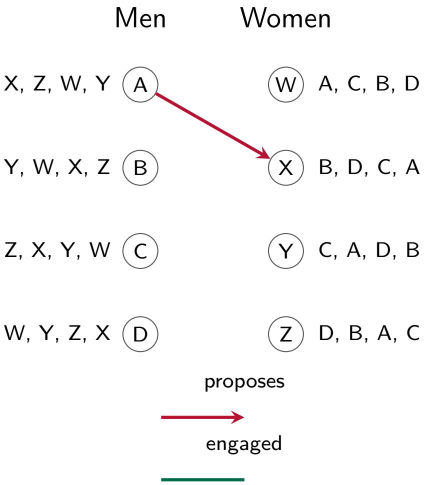
**Input** : Bipartite graph $G = (A \dot\cup B, E)$, preference lists $\prec_v$, $v \in V = A \cup B$.
**Output** : A stable matching in $G$.

Initialization: All men $A$ and women $B$ are unengaged $\mu(v) = \emptyset$, for all $v \in V$.
**while** There exists a man $m \in A$ who is currently unengaged $(\mu(m) = \emptyset)$ and has not yet proposed to all women **do**
  Choose any such man $m$.
  $m$ proposes to the best woman $w = \min_{\prec_m}\{w' \in B\}$ on his list.
  **if** woman $w$ is not yet engaged **then**
    Engage $m$ and $w$, i.e., set $\mu(m) = w, \mu(w) = m$.
  **else**
    **if** $w$ prefers $m$ over her current fiancé $m'$, $m \prec_w \mu(w)$ **then**
      Break the engagement between $w$ and $m'$ $(\mu(w) = \mu(m') = \emptyset)$.
      Engage $m$ and $w$ $(\mu(m) := w, \mu(w) = m)$.
      Remove $w$ from $m'$'s list.
    **else**
      Remove $w$ from $m$'s list.

**return** Matching $M^*$ consisting of all currently engaged pairs.

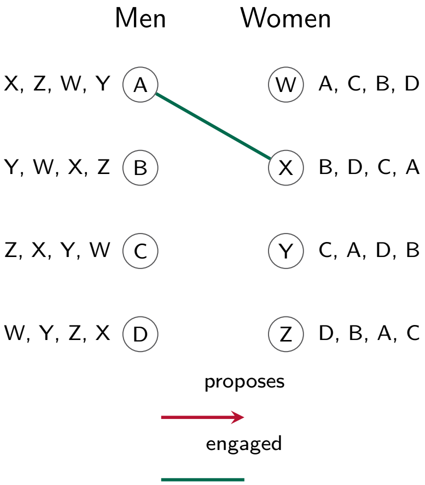
Universität Bremen

# Gale-Shapley Algorithm: Example 1



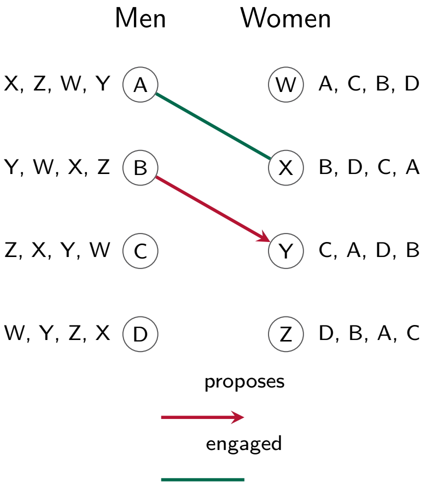

- $A$ proposes to $X$

# Gale-Shapley Algorithm: Example 1



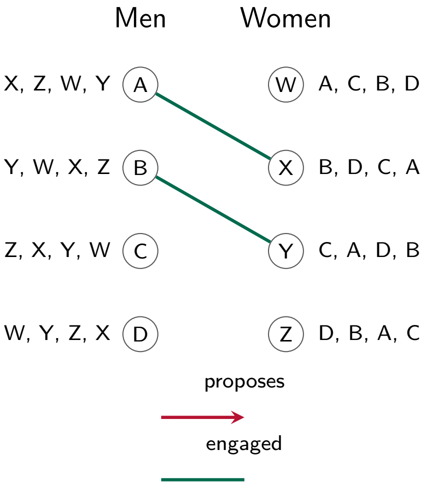

- $A$ proposes to $X$
- $X$ accepts and gets engaged to $A$

# Gale-Shapley Algorithm: Example 1



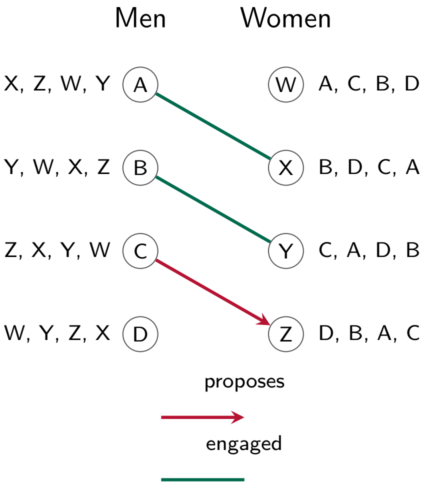

- $B$ proposes to $Y$

# Gale-Shapley Algorithm: Example 1



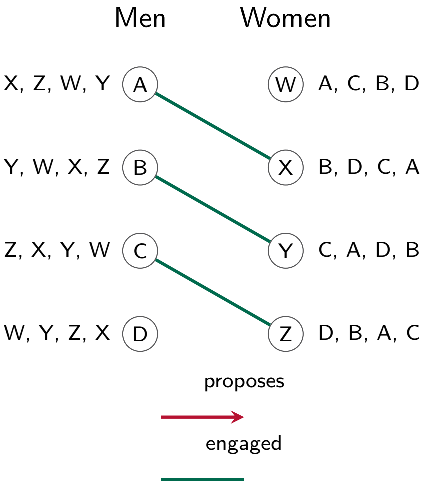

- $B$ proposes to $Y$
- $Y$ accepts and gets engaged to $B$

# Gale-Shapley Algorithm: Example 1



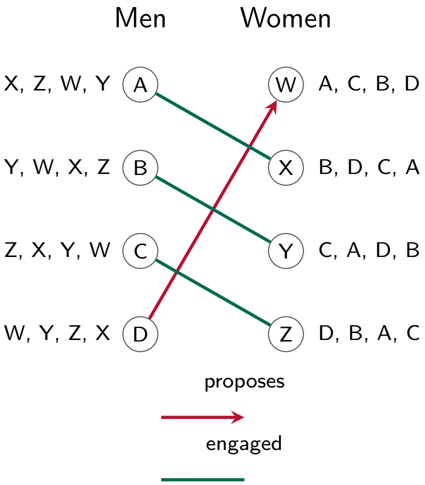

- $C$ proposes to $Z$

# Gale-Shapley Algorithm: Example 1



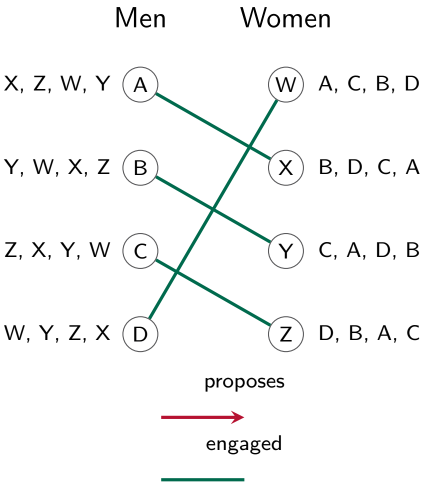

- $C$ proposes to $Z$
- $Z$ accepts and gets engaged to $C$

# Gale-Shapley Algorithm: Example 1



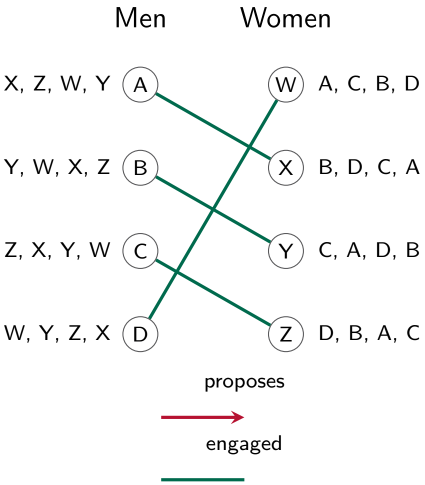

- *D* proposes to *W*

# Gale-Shapley Algorithm: Example 1



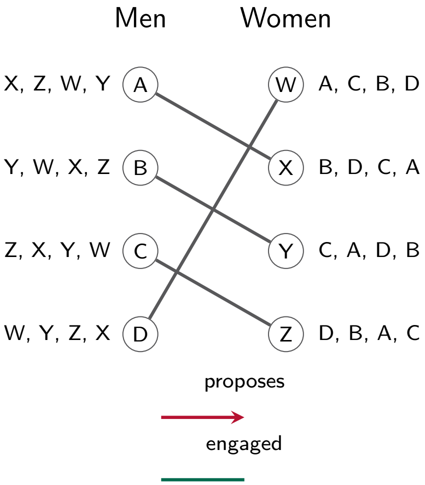

- *D* proposes to *W*
- *W* accepts and gets engaged to *D*

# Gale-Shapley Algorithm: Example 1



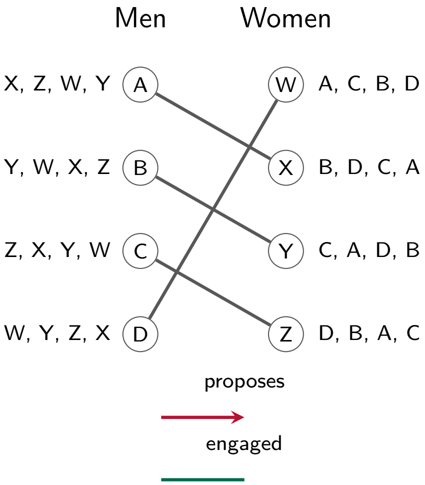

- Alle men are engaged.

# Gale-Shapley Algorithm: Example 1



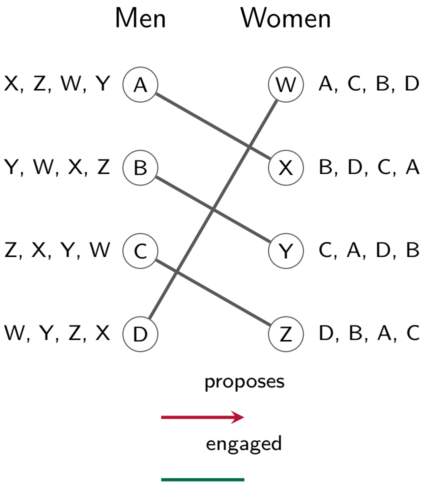

- Alle men are engaged.
- The resulting matching is perfect and stable.

# Gale-Shapley Algorithm: Example 1




- Alle men are engaged.
- The resulting matching is perfect and stable.
- Every man has his 1. priority; not the women.

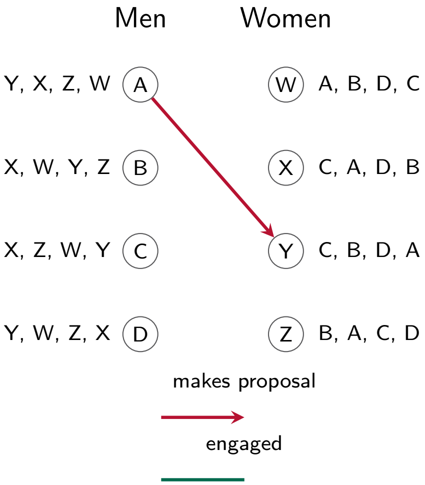
# Gale-Shapley Algorithm: Example 1




- The resulting matching is perfect and stable.

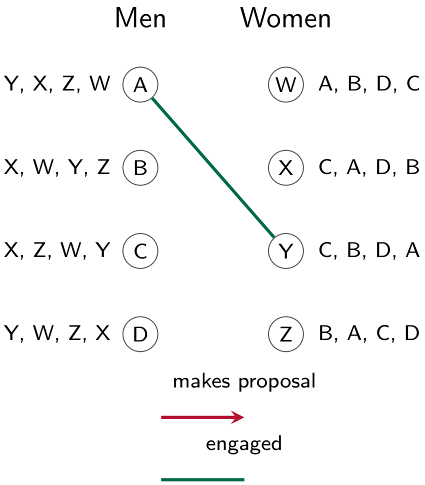
Boring Example (one round).

# Gale-Shapley Algorithm: 2nd Example



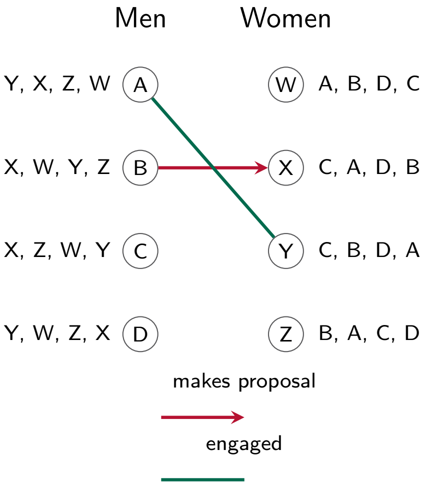

- $A$ proposes to $Y$

# Gale-Shapley Algorithm: 2nd Example



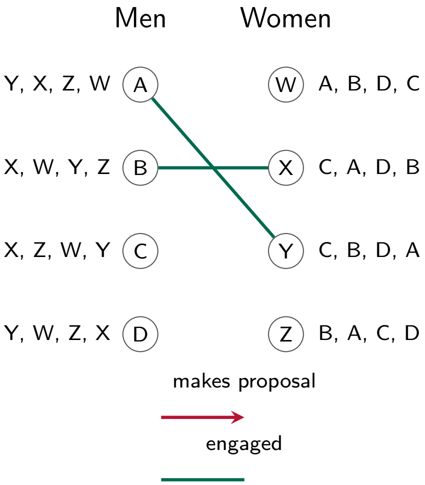

- *A* proposes to *Y*
- *Y* accepts and gets engaged to *A*

# Gale-Shapley Algorithm: 2nd Example



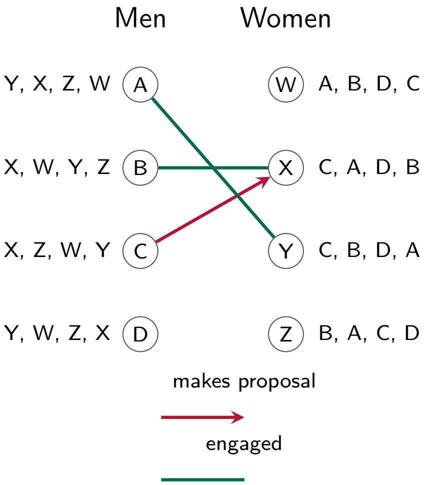

- $B$ proposes to $X$

# Gale-Shapley Algorithm: 2nd Example




- *B* proposes to *X*
- *X* accepts and gets engaged to *B*

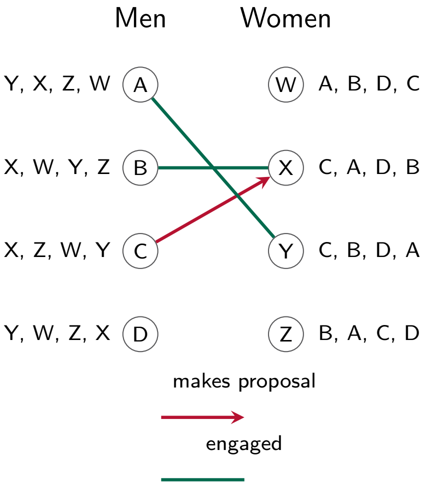
# Gale-Shapley Algorithm: 2nd Example




- $C$ proposes to $X$

Universität Bremen

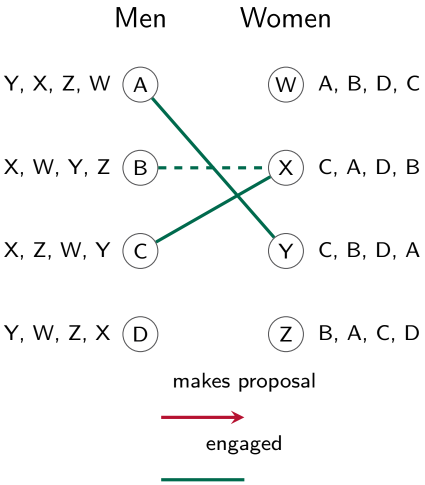
# Gale-Shapley Algorithm: 2nd Example




- $C$ proposes to $X$
- $X$ compares $C$ and $B$: she prefers $C$ over $B$

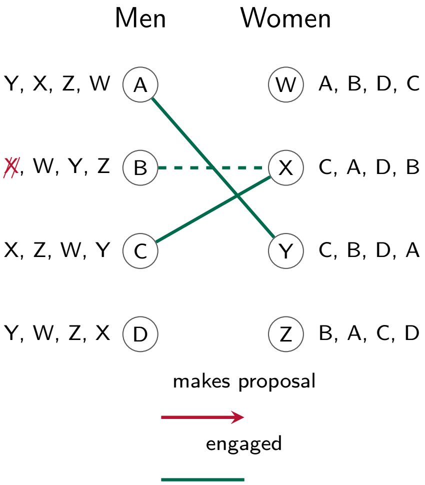
Universität Bremen

# Gale-Shapley Algorithm: 2nd Example



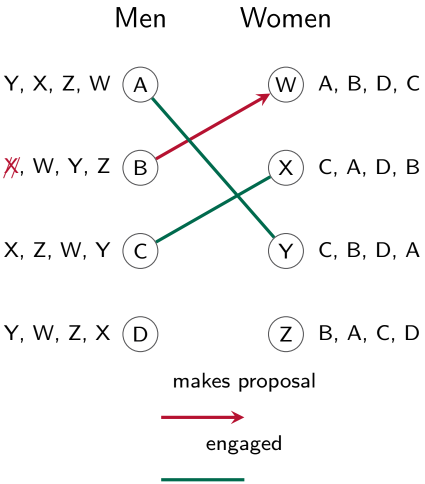

- *C* proposes to *X*
- *X* compares *C* and *B*: she prefers *C* over *B*
- *X* breaks engagement with *B* and gets engaged to *C*

# Gale-Shapley Algorithm: 2nd Example



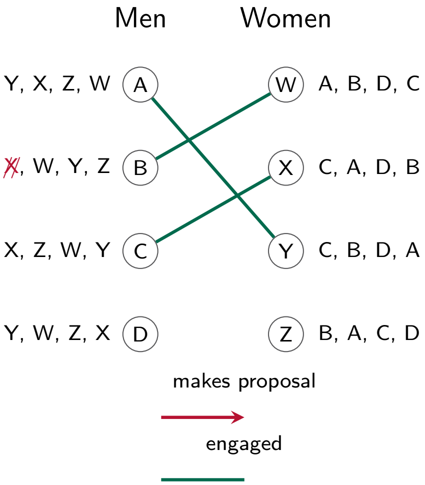

- *C* proposes to *X*
- *X* compares *C* and *B*: she prefers *C* over *B*
- *X* breaks engagement with *B* and gets engaged to *C*
- *B* removes *X* from his list

# Gale-Shapley Algorithm: 2nd Example



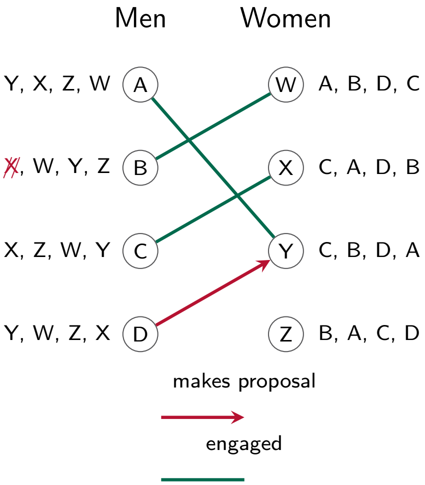

- *B* proposes to *W*

# Gale-Shapley Algorithm: 2nd Example



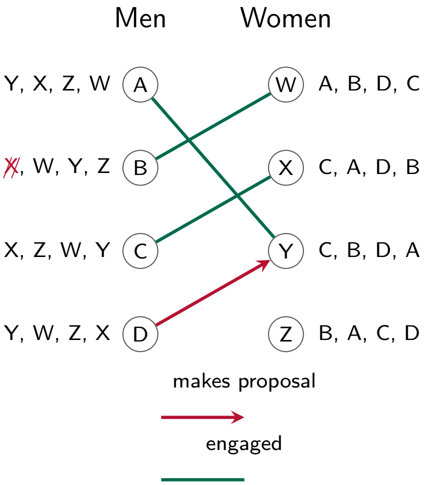

- *B* proposes to *W*
- *W* accepts and gets engaged to *B*

# Gale-Shapley Algorithm: 2nd Example



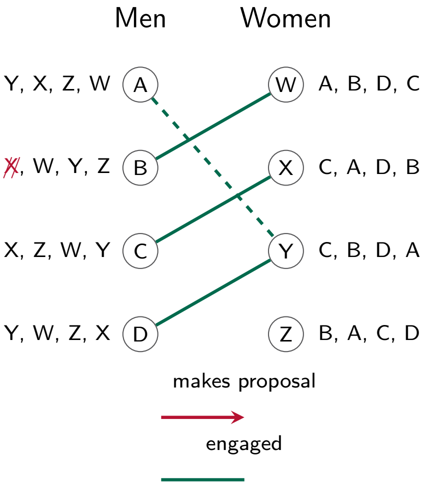

- *D* proposes to *Y*

# Gale-Shapley Algorithm: 2nd Example




- *D* proposes to *Y*
- *Y* compares *D* and *A*: she prefers *D* over *A*

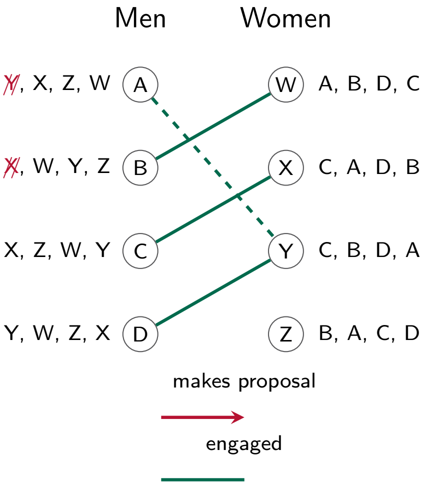
# Gale-Shapley Algorithm: 2nd Example




- *D* proposes to *Y*
- *Y* compares *D* and *A*: she prefers *D* over *A*
- *Y* breaks engagement with *A* and gets engaged to *D*

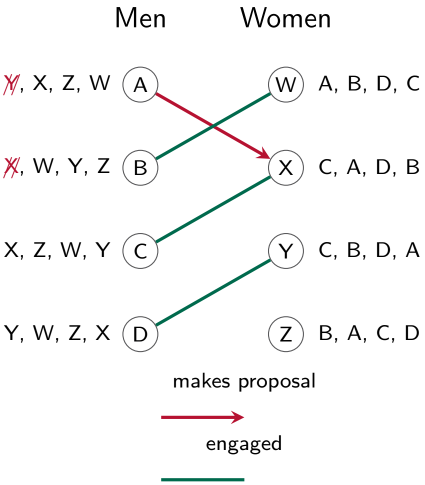
Universität Bremen

# Gale-Shapley Algorithm: 2nd Example



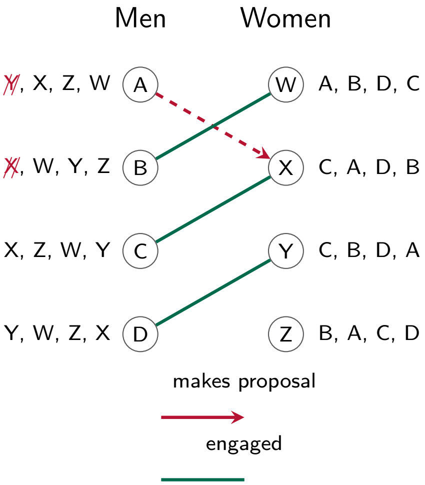

- *D* proposes to *Y*
- *Y* compares *D* and *A*: she prefers *D* over *A*
- *Y* breaks engagement with *A* and gets engaged to *D*
- *A* removes *Y* from his list

# Gale-Shapley Algorithm: 2nd Example



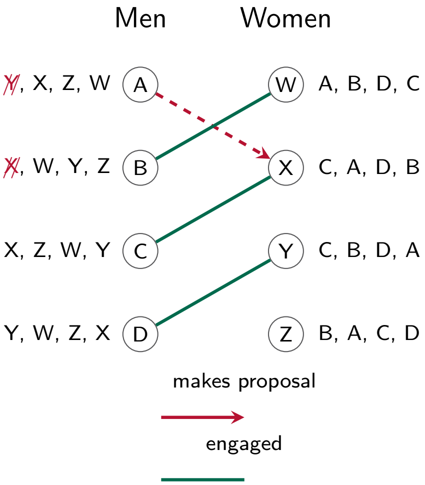

- $A$ proposes to $X$

# Gale-Shapley Algorithm: 2nd Example



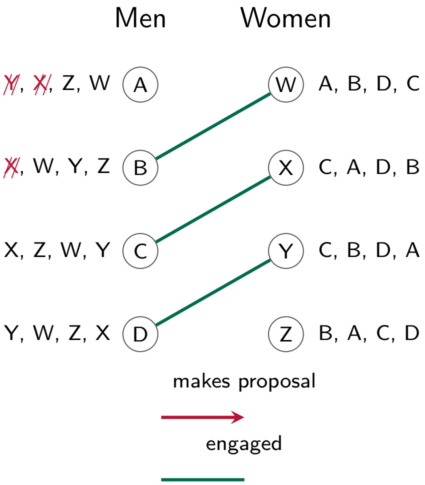

- *A* proposes to *X*
- *X* prefers *C* over *A*

# Gale-Shapley Algorithm: 2nd Example



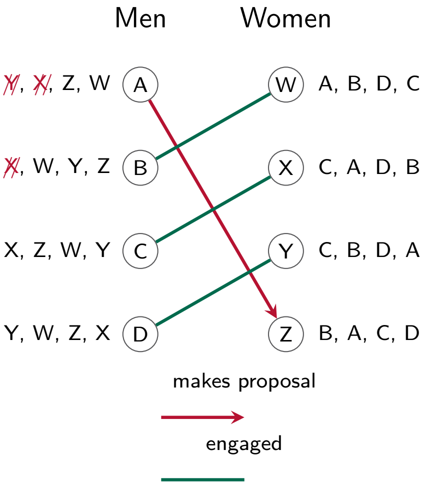

- $A$ proposes to $X$
- $X$ prefers $C$ over $A$
- $X$ rejects $A$

# Gale-Shapley Algorithm: 2nd Example



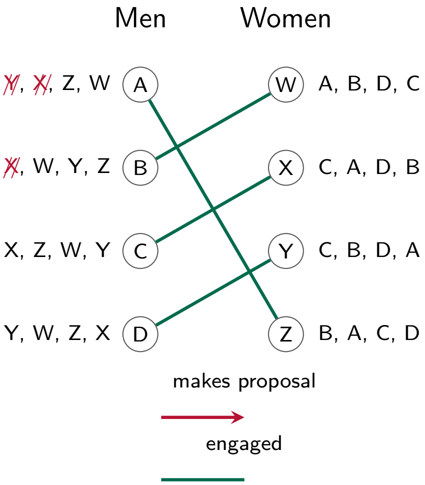

- $A$ proposes to $X$
- $X$ prefers $C$ over $A$
- $X$ rejects $A$
- $A$ removes $X$ from his list

# Gale-Shapley Algorithm: 2nd Example



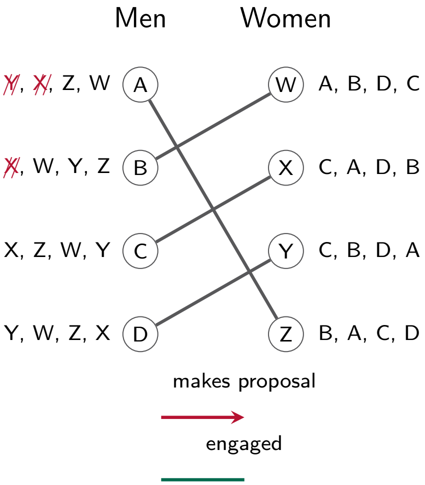

- $A$ proposes to $Z$

# Gale-Shapley Algorithm: 2nd Example



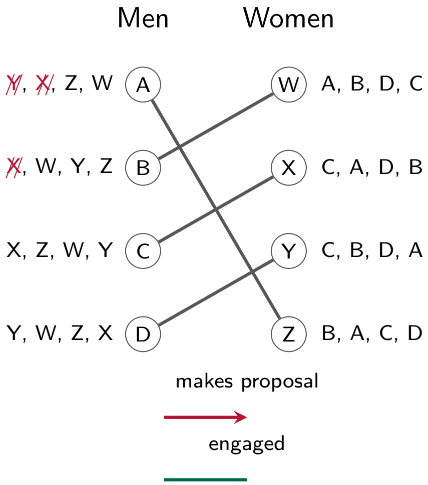

- *A* proposes to *Z*
- *Z* is overjoyed and accepts
- *Z* and *A* get engaged

# Gale-Shapley Algorithm: 2nd Example



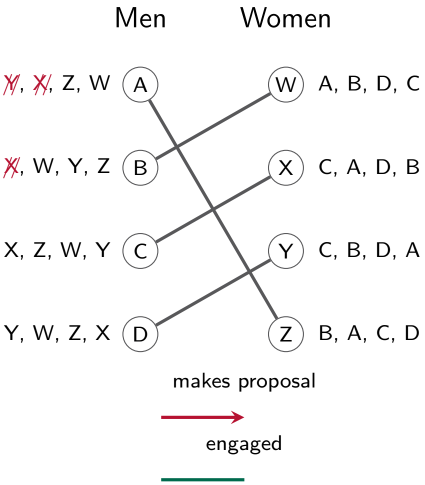

- Termination condition reached: all men are engaged

# Gale-Shapley Algorithm: 2nd Example




- Termination condition reached: all men are engaged
- The algorithm outputs the current matching

# Gale-Shapley Algorithm: 2nd Example




## Observation

- The resulting matching is perfect and stable
- Neither all men nor all women are matched with their first choice

# Correctness – Termination

**Lemma**

The algorithm terminates after $\mathcal{O}(|A||B|)$ steps.

# Correctness – Termination

**Lemma**

The algorithm terminates after $\mathcal{O}(|A||B|)$ steps.

- Men propose to women in decreasing order of preference. $\mu(a)$ never decreases with respect to $\prec_a$ for each $a \in A$ (i.e., it never improves)

# Correctness – Termination

**Lemma**

The algorithm terminates after $\mathcal{O}(|A||B|)$ steps.

- Men propose to women in decreasing order of preference. $\mu(a)$ never decreases with respect to $\prec_a$ for each $a \in A$ (i.e., it never improves)
- A woman only switches to a better partner for her. $\mu(b)$ never increases with respect to $\prec_b$ for each $b \in B$ (i.e., it never worsens)

# Correctness – Termination

**Lemma**

The algorithm terminates after $\mathcal{O}(|A||B|)$ steps.

- Men propose to women in decreasing order of preference. $\mu(a)$ never decreases with respect to $\prec_a$ for each $a \in A$ (i.e., it never improves)
- A woman only switches to a better partner for her. $\mu(b)$ never increases with respect to $\prec_b$ for each $b \in B$ (i.e., it never worsens)
- In each iteration of the while loop, a man proposes to a woman to whom he has not yet proposed. ($\mu(a)$ advances for some $a \in A$.)

# Correctness – Termination

**Lemma**

The algorithm terminates after $\mathcal{O}(|A||B|)$ steps.

- Men propose to women in decreasing order of preference. $\mu(a)$ never decreases with respect to $\prec_a$ for each $a \in A$ (i.e., it never improves)
- A woman only switches to a better partner for her. $\mu(b)$ never increases with respect to $\prec_b$ for each $b \in B$ (i.e., it never worsens)
- In each iteration of the while loop, a man proposes to a woman to whom he has not yet proposed. ($\mu(a)$ advances for some $a \in A$.)
- Since there are only $|A||B|$ possible man-woman pairs, the statement follows. $\square$

# Correctness

**Lemma**

The algorithm terminates with a stable matching.

# Correctness

**Lemma**

The algorithm terminates with a stable matching.

- At any point, every man is matched to at most one woman, and every woman to at most one man. Thus, the pairs always form a matching.

# Correctness

**Lemma**

The algorithm terminates with a stable matching.

- At any point, every man is matched to at most one woman, and every woman to at most one man. Thus, the pairs always form a matching.
- Suppose there exists a blocking pair $\{a, b\} \in E \setminus M$, that is, a man $a$ and a woman $b$ who are not matched at termination and

$$(i)\ b \prec_a \mu(a) \quad \text{and} \quad (ii)\ a \prec_b \mu(b)$$

# Correctness

**Lemma**

The algorithm terminates with a stable matching.

- At any point, every man is matched to at most one woman, and every woman to at most one man. Thus, the pairs always form a matching.
- Suppose there exists a blocking pair $\{a, b\} \in E \setminus M$, that is, a man $a$ and a woman $b$ who are not matched at termination and

$$(i)\ b \prec_a \mu(a) \quad \text{and} \quad (ii)\ a \prec_b \mu(b)$$

- Because of $(i)$, there was an iteration where man $a$ proposed to woman $b$. However, $b$ rejected $a$ either immediately or later, thus $\mu(b) \preceq_b a$; contradiction to $(ii)$. $\square$

# Correctness

**Lemma**

The algorithm terminates with a stable matching.

- At any point, every man is matched to at most one woman, and every woman to at most one man. Thus, the pairs always form a matching.
- Suppose there exists a blocking pair $\{a, b\} \in E \setminus M$, that is, a man $a$ and a woman $b$ who are not matched at termination and

$$(i)\ b \prec_a \mu(a) \quad \text{and} \quad (ii)\ a \prec_b \mu(b)$$

- Because of $(i)$, there was an iteration where man $a$ proposed to woman $b$. However, $b$ rejected $a$ either immediately or later, thus $\mu(b) \preceq_b a$; contradiction to $(ii)$. $\square$

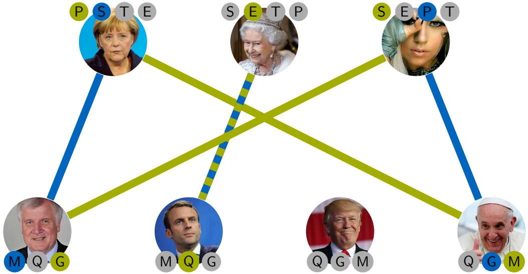
**Theorem**

Gale-Shapley's algorithm computes a stable matching in $\mathcal{O}(|A||B|)$.

# The Best Stable Matching?

Suppose there are multiple stable matchings in a graph — which one does the Gale-Shapley algorithm find?

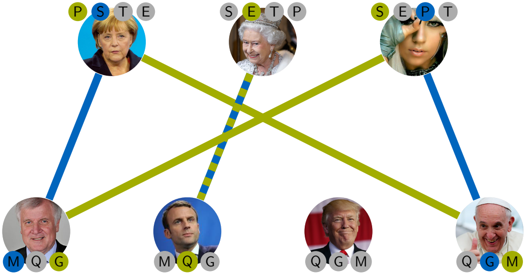
# Best stable Matching?




Blue Matching is better for  and .

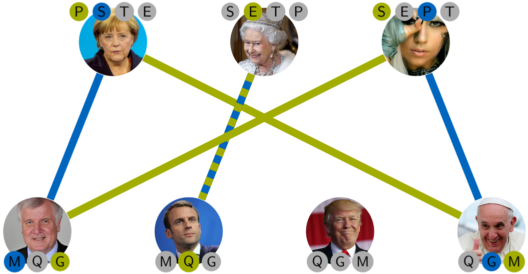
Green Matching is better for  and .

# Best stable Matching?




**Definition.** We call a woman $w$ a valid partner of man $m$, if there is a stable matching $M$ with $(w, m) \in M$.

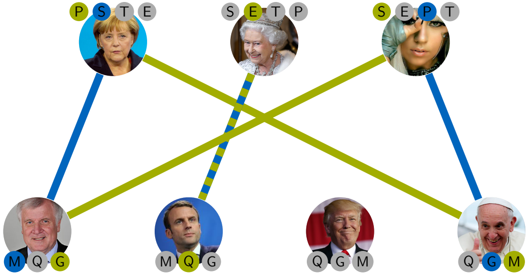
# Best stable Matching?




**Definition.** We call a woman $w$ a valid partner of man $m$, if there is a stable matching $M$ with $(w, m) \in M$.

A matching is men-optimal, if every man is matched with his best valid partner.

# Best stable Matching?




**Definition.** We call a woman $w$ a valid partner of man $m$, if there is a stable matching $M$ with $(w, m) \in M$.

A matching is men-optimal, if every man is matched with his best valid partner.

**Theorem**

The Gale-Shapley Algorithm computes a man-optimal matching.

# Man-optimal Matching

**Theorem**

The Gale-Shapley Algorithm computes a man-optimal matching.

# Man-optimal Matching

**Theorem**

The Gale-Shapley Algorithm computes a man-optimal matching.

**Proof**: Suppose the algorithm finds a matching which is not man-optimal.

- Then there exists an earlier time point $t$, in which man $m$ was rejected from his preferred valid partner $w$.

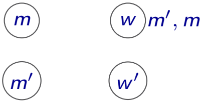
# Man-optimal Matching

**Theorem**

The Gale-Shapley Algorithm computes a man-optimal matching.

**Proof**: Suppose the algorithm finds a matching which is not man-optimal.

- Then there exists an earlier time point $t$, in which man $m$ was rejected from his preferred valid partner $w$.
- $w$ rejects $m$ at time point $t$ only because she is engaged with a preferred man $m'$, i.e., $w$ prefers $m'$ over $m$.



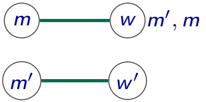
# Man-optimal Matching

## Theorem

The Gale-Shapley Algorithm computes a man-optimal matching.

**Proof**: Suppose the algorithm finds a matching which is not man-optimal.

- Then there exists an earlier time point $t$, in which man $m$ was rejected from his preferred valid partner $w$.
- $w$ rejects $m$ at time point $t$ only because she is engaged with a preferred man $m'$, i.e., $w$ prefers $m'$ over $m$.
- Consider a stable matching $M$, in which $m$ and $w$ are a couple (has to exist since $w$ is a valid partner for $m$). In $M$ let $m'$ be matched to woman $w'$.




stable Matching $M$

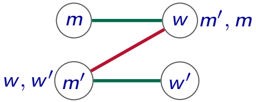
# Man-optimal Matching

**Theorem**

The Gale-Shapley Algorithm computes a man-optimal matching.

**Proof**: Suppose the algorithm finds a matching which is not man-optimal.

- Then there exists an earlier time point $t$, in which man $m$ was rejected from his preferred valid partner $w$.
- $w$ rejects $m$ at time point $t$ only because she is engaged with a preferred man $m'$, i.e., $w$ prefers $m'$ over $m$.
- Consider a stable matching $M$, in which $m$ and $w$ are a couple (has to exist since $w$ is a valid partner for $m$). In $M$ let $m'$ be matched to woman $w'$.
- Man $m'$ prefers $w$ over $w'$, since $m'$ at time point $t$ is engaged with $w$ and before was not rejected from a valid woman, in particular not rejected by $w'$. Therefore, $m'$ and $w$ are a blocking pair w.r.t. $M$.




stable Matching $M$

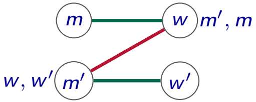
# Man-optimal Matching

## Theorem

The Gale-Shapley Algorithm computes a man-optimal matching.

**Proof**: Suppose the algorithm finds a matching which is not man-optimal.

- Then there exists an earlier time point $t$, in which man $m$ was rejected from his preferred valid partner $w$.
- $w$ rejects $m$ at time point $t$ only because she is engaged with a preferred man $m'$, i.e., $w$ prefers $m'$ over $m$.
- Consider a stable matching $M$, in which $m$ and $w$ are a couple (has to exist since $w$ is a valid partner for $m$). In $M$ let $m'$ be matched to woman $w'$.
- Man $m'$ prefers $w$ over $w'$, since $m'$ at time point $t$ is engaged with $w$ and before was not rejected from a valid woman, in particular not rejected by $w'$. Therefore, $m'$ and $w$ are a blocking pair w.r.t. $M$.
- Contradiction! $M$ is not a stable matching. □




stable Matching $M$
aber blocking pair $\{m', w\}$ in $M$

# Women can not do worse

**Satz**

The Gale-Shapley Algorithm matches to each woman her worst valid partner.

# Women can not do worse

**Satz**

The Gale-Shapley Algorithm matches to each woman her worst valid partner.

**Beweis.**

- Suppose there is a woman $w$ who is engaged with a man $m'$, who is not the worst valid partner $m$ for her.

# Women can not do worse

**Satz**

The Gale-Shapley Algorithm matches to each woman her worst valid partner.

**Beweis.**

- Suppose there is a woman $w$ who is engaged with a man $m'$, who is not the worst valid partner $m$ for her.
- We consider a stable matching $M$, in which $w$ is matched to $m$. Hence, woman $w$ prefers $m'$ over $m$.

# Women can not do worse

**Satz**

The Gale-Shapley Algorithm matches to each woman her worst valid partner.

**Beweis.**

- Suppose there is a woman $w$ who is engaged with a man $m'$, who is not the worst valid partner $m$ for her.
- We consider a stable matching $M$, in which $w$ is matched to $m$. Hence, woman $w$ prefers $m'$ over $m$.
- Man $m'$ is in matching $M$ matched to a woman $w'$.

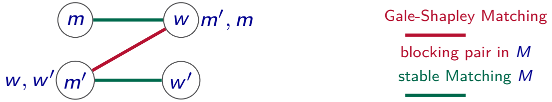
# Women can not do worse

## Satz

The Gale-Shapley Algorithm matches to each woman her worst valid partner.

**Beweis.**

- Suppose there is a woman $w$ who is engaged with a man $m'$, who is not the worst valid partner $m$ for her.
- We consider a stable matching $M$, in which $w$ is matched to $m$. Hence, woman $w$ prefers $m'$ over $m$.
- Man $m'$ is in matching $M$ matched to a woman $w'$.
- Since the Gale-Shapley Matching is man-optimal, man $m'$ prefers his Gale-Shapley-partner $w$ over $w'$.



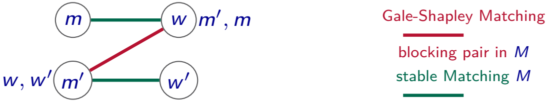
# Women can not do worse

**Satz**

The Gale-Shapley Algorithm matches to each woman her worst valid partner.

**Beweis.**

- Suppose there is a woman $w$ who is engaged with a man $m'$, who is not the worst valid partner $m$ for her.
- We consider a stable matching $M$, in which $w$ is matched to $m$. Hence, woman $w$ prefers $m'$ over $m$.
- Man $m'$ is in matching $M$ matched to a woman $w'$.
- Since the Gale-Shapley Matching is man-optimal, man $m'$ prefers his Gale-Shapley-partner $w$ over $w'$.
- Contradiction to the stability of $M$, since $\{m', w\}$ is a blocking pair. □



# Strategically manipul manipulable?

**Frage:**

Can it be strategically worth to give false preferences?

Universität Bremen

# Strategically manipul manipulable?

**Frage:**

Can it be strategically worth to give false preferences?

Suppose the participants know that

- the matching is computed by Gale-Shapley
- and know the preferences of the remaining participants.

| | | | |
|---|---|---|---|
| X, Y, Z | A | X | B, A, C |
| Y, X, Z | B | Y | A, B, C |
| X, Y, Z | C | Z | A, B, C |

# Strategically manipul manipulable?

**Frage:**

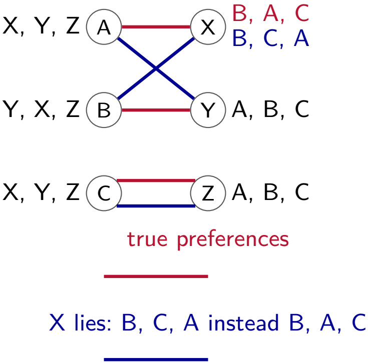
Can it be strategically worth to give false preferences?

Suppose the participants know that

- the matching is computed by Gale-Shapley
- and know the preferences of the remaining participants.

**Answer:**

„No“ for the men, „yes“ for the women.



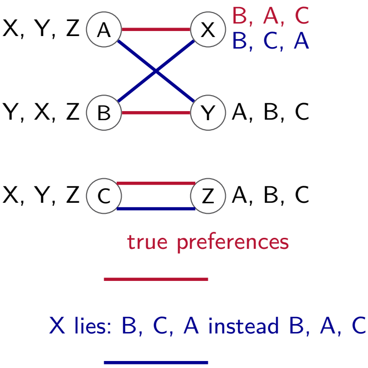
# Strategically manipul manipulable?

**Frage:**

Can it be strategically worth to give false preferences?

Suppose the participants know that

- the matching is computed by Gale-Shapley
- and know the preferences of the remaining participants.



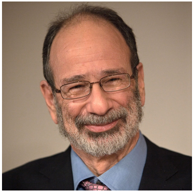

**Answer:**

„No“ for the men, „yes“ for the women.

The algorithm is “one-sided” strategy-proof.

# Nobel-price for Economics 2012

- Lloyd S. Shapley (1923-2016) together with Alvin E. Roth
- „. . . for the theory of stable allocations and the practice of market design."
- Shapley: theoretical foundations, Roth: Exchange-market for kidney exchange, assignment of students to High Schools



Alvin E. Roth, 2012

Lloyd Shapley, 1980

# Recap Matchings

- Weighted bipartite matching
  - Weighted directed graph to compute augmenting paths
- Non-bipartite matchings
  - Augmenting paths vs. blossoms
- Edmonds-Gallai Decomposition
  - Structure of maximum matchings
- Stable matchings
  - preference list for matching participants