



CSE 331: Algorithms & Complexity “Matchings”

Prof. Charlie Anne Carlson (She/Her)

Lecture 3

Monday September 3rd, 2025



University at Buffalo®



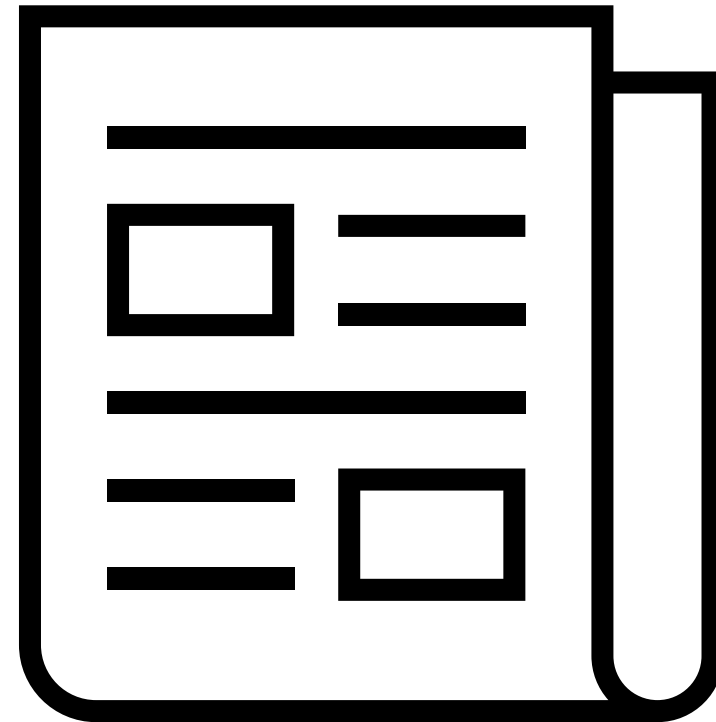
Schedule

1. Course Updates
2. Group Project
3. Pet Assignments
4. Matchings
5. Gale-Shapely



Course Updates

- Office Hours Posted
- Complete Syllabus Quiz
- HW 0 Solutions Up
- HW 1 Posted Before Next Tuesday



Group Project

CSE 331 Syllabus Piazza Schedule Homeworks ▾ Autolab Project ▾ Support Pages ▾

CSE 331 Project

Fall 2025

Details and motivations for the project.

Motivation

[CSE 331](#) is primarily concerned with the technical aspects of algorithms: how to design them and then how to analyze their correctness and runtime. However, algorithms are pervasive in our world and are commonplace in many aspects of society. The main aim of the project is to have you explore some of the social implications of algorithms.

Just to give some examples for such implications:

- Big data is hot these days and there is a (not-uncommon) belief that by running (mainly machine learning) algorithms on big data, we can detect patterns and use those to potentially make policy decisions. Here is a cautionary talk:

Group Project: 3 Parts

1. Do **five programming problems** that involve making tradeoffs between various choices, some of which have ethical dimensions.
2. Each programming question will be paired with (a series of) **reflection questions**.
3. At the end of the project, each group member will fill in a **survey** rating their own and the other group members' contribution to the project.

Group Project: Timeline

Deliverable	Due Date
Team Composition	September 19
Team Registration (on Autolab)	September 29
Problem 1 Coding	October 31
Problem 1 Reflection	November 3
Problem 2 Coding	October 31
Problem 2 Reflection	November 3
Problem 3 Coding	November 24
Problem 3 Reflection	December 1
Problem 4 Coding	December 5
Problem 4 Reflection	December 9
Problem 5 Coding	December 5
Problem 5 Reflection	December 9
Survey	December 9

Group Project: Teams

1. Teams will consist of **3 people**.
2. You can form and register your own team, or you can request to be formed into a team.
3. Teams will not change.
4. There is a **google form** under the project tab on the website to register team or request being added randomly to a team.
5. Register yourself or your team by **September 19th** at midnight or get a zero!

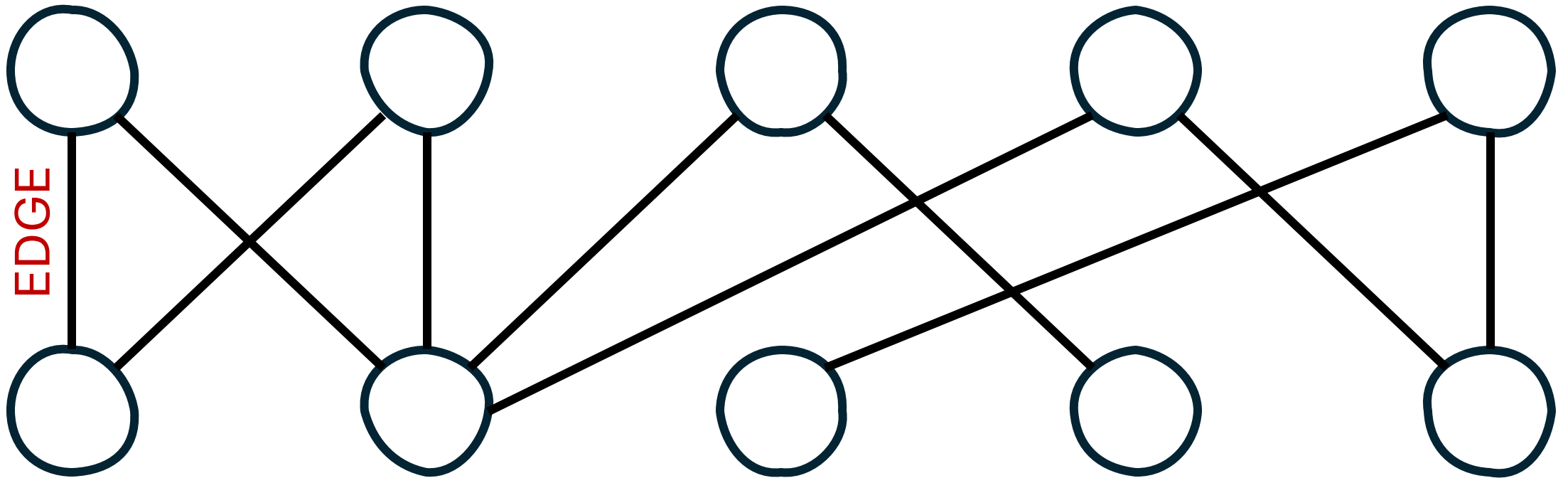
Group Project: Teams

1. Teams will consist of **3 people**.
2. You can form and register your own team, or you can request to be formed into a team.
3. Teams will not change.
4. There is a **google form** under the project tab on the website to register team or request being added randomly to a team.
5. Register yourself or your team by **September 19th** at midnight or get a zero!

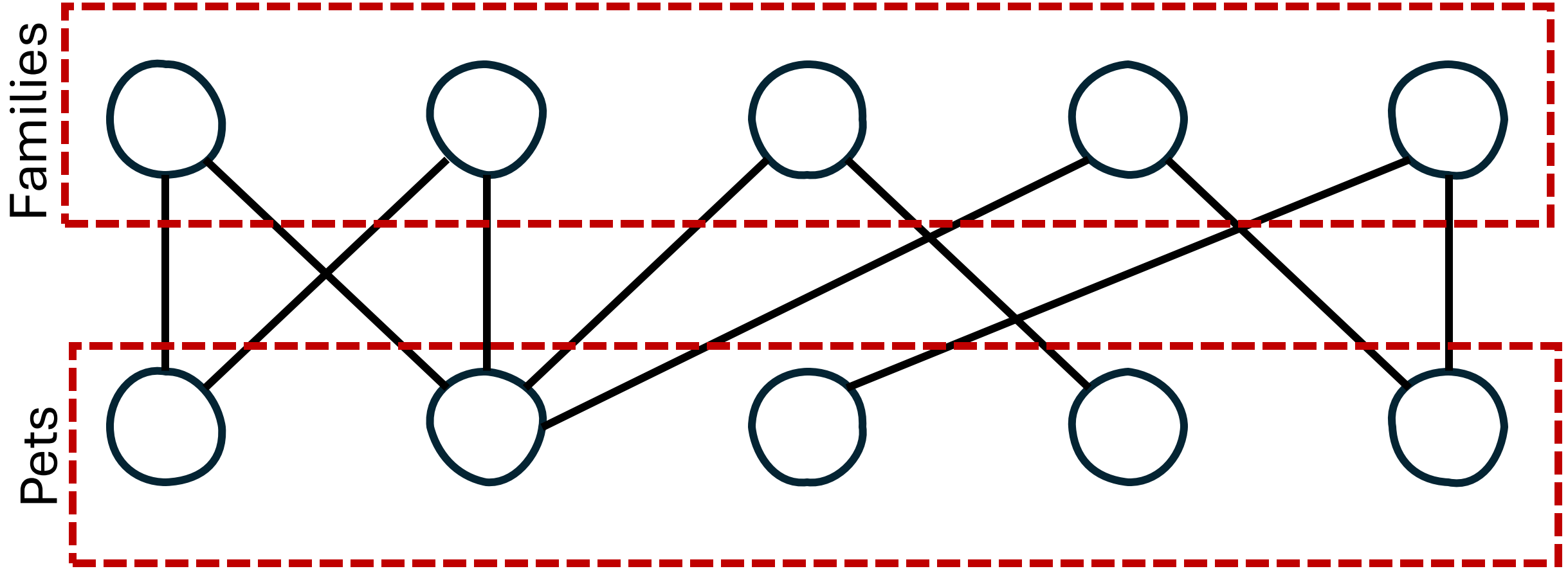
Graphs

NODE

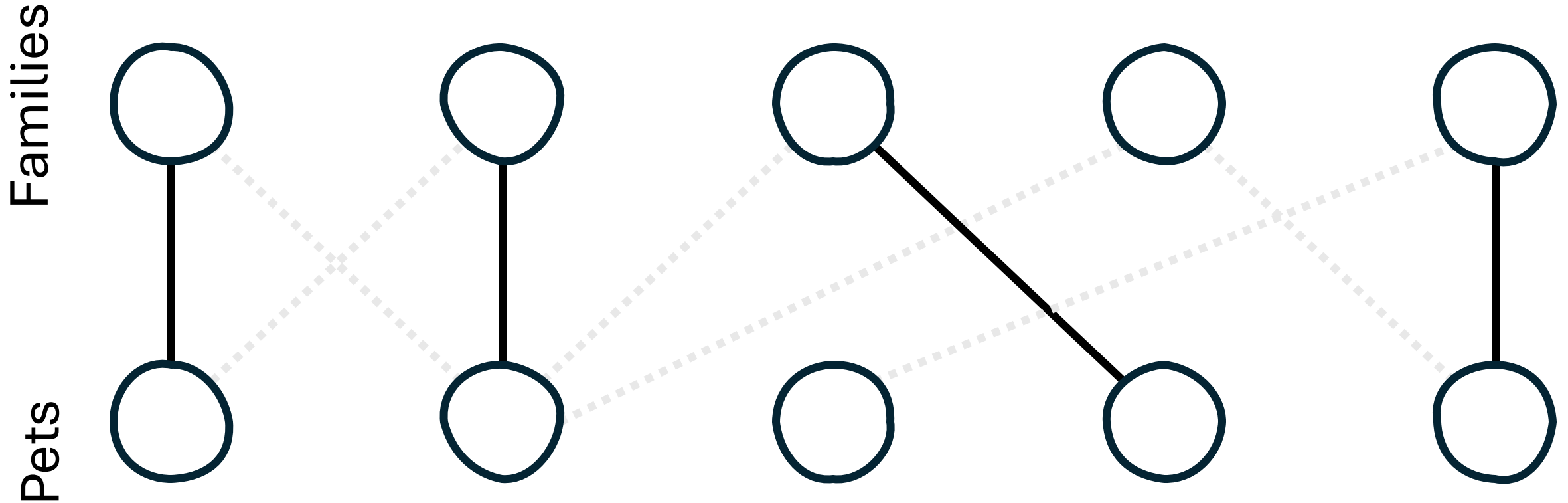
EDGE



Devs + Testers = Success

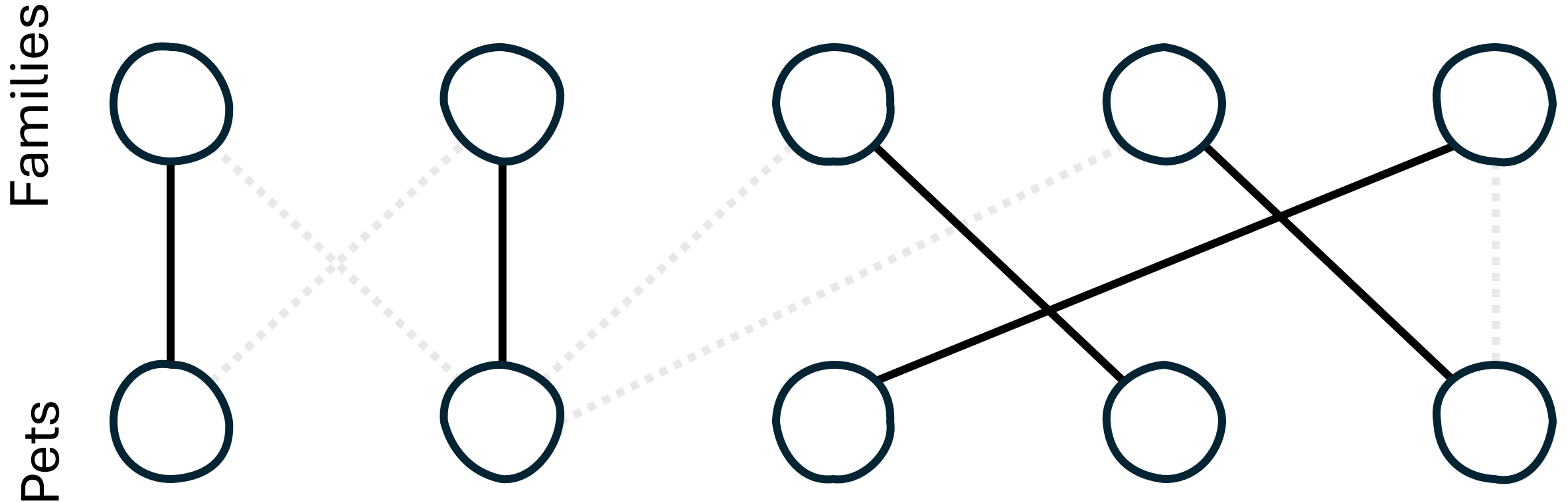


Matching



Definition: A ***matching*** is a collection of edges such that no two share a node.

Perfect Matching



Definition: A ***perfect matching*** is a matching such that every node is incident to an edge.

Pet Assignments

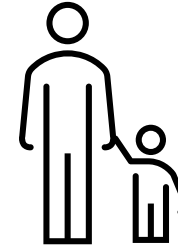
Peter



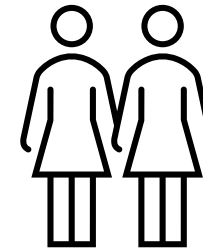
Kat



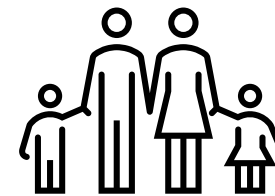
Avery



Family A



Family B



Family C

Pet Assignments

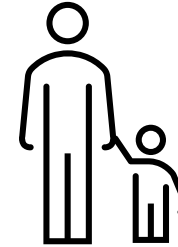
Peter



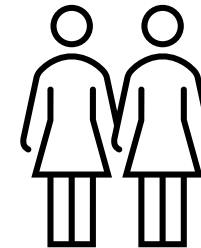
Kat



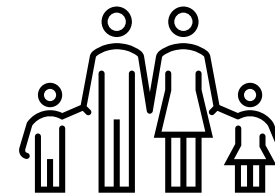
Avery



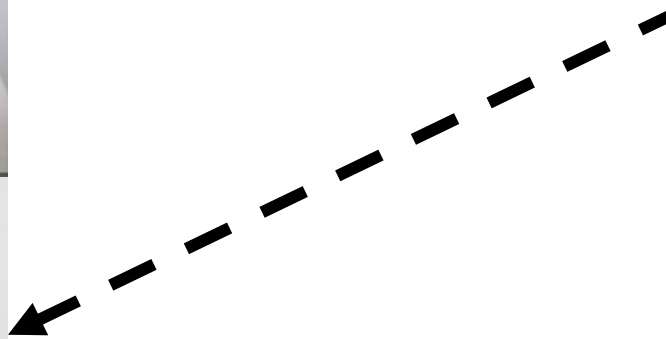
Family A



Family B



Family C



Pet Assignments

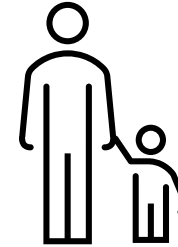
Peter



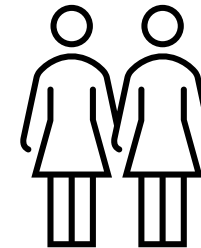
Kat



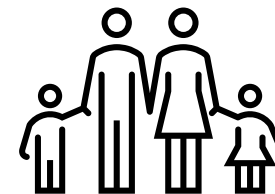
Avery



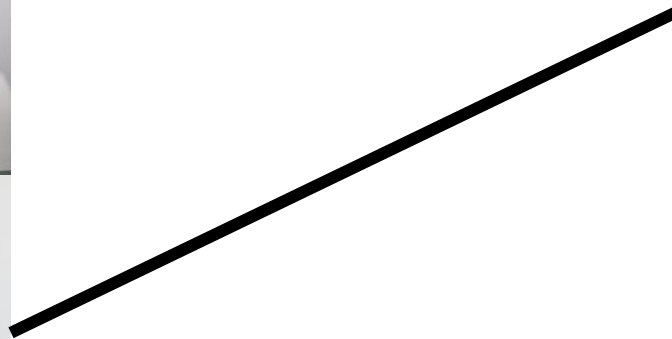
Family A



Family B



Family C



Pet Assignments

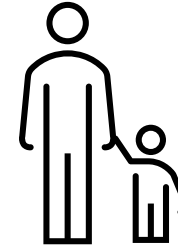
Peter



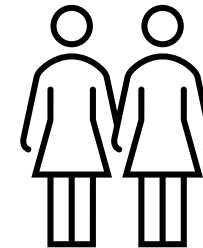
Kat



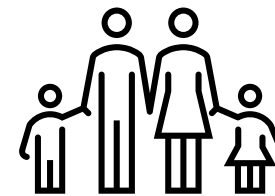
Avery



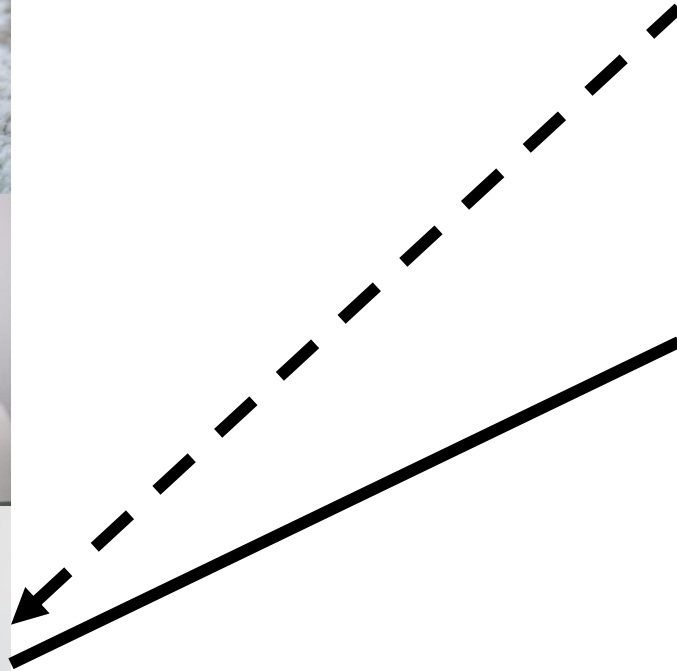
Family A



Family B



Family C



Pet Assignments

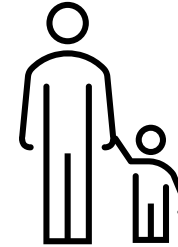
Peter



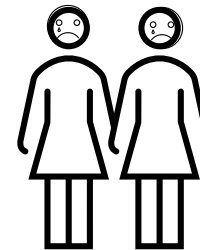
Kat



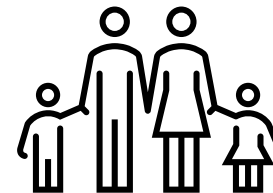
Avery



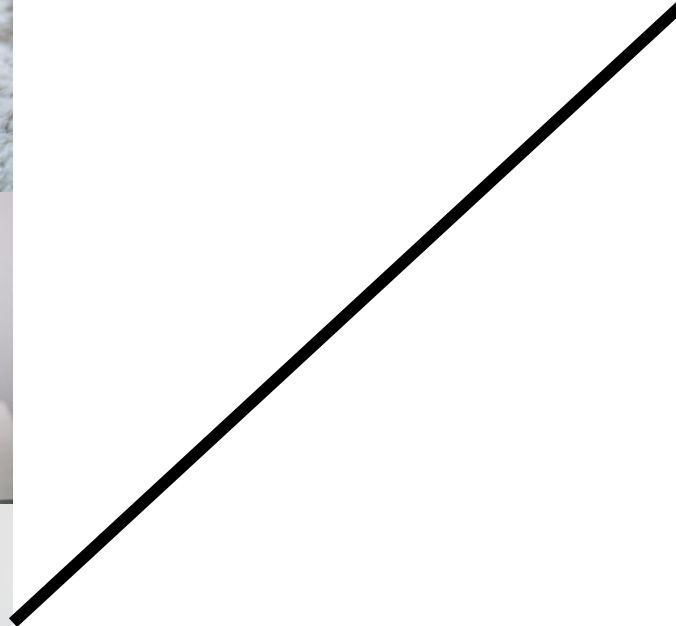
Family A



Family B



Family C



Pet Assignments

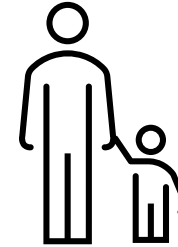
Peter



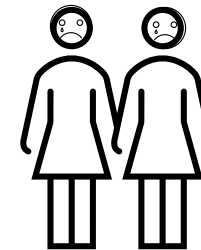
Kat



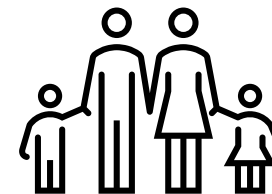
Avery



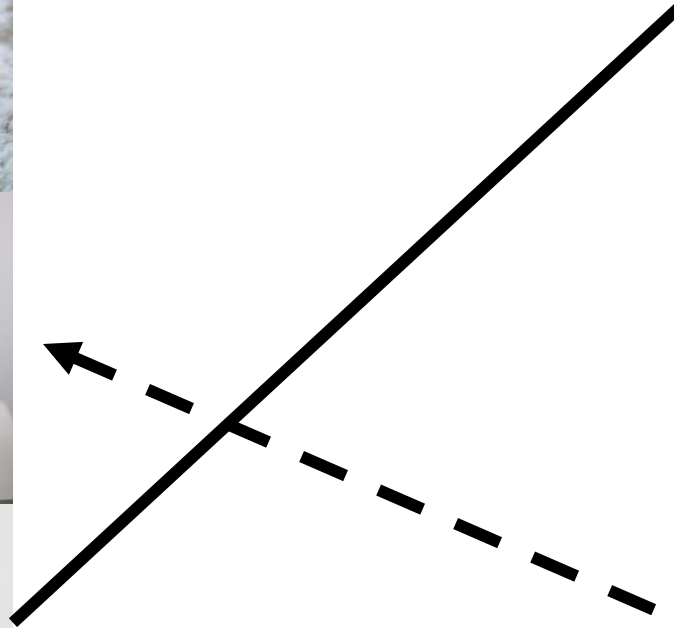
Family A



Family B



Family C



Pet Assignments

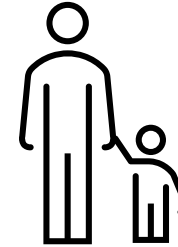
Peter



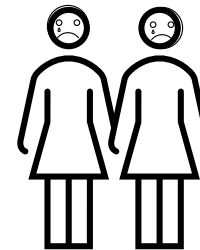
Kat



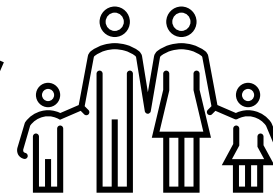
Avery



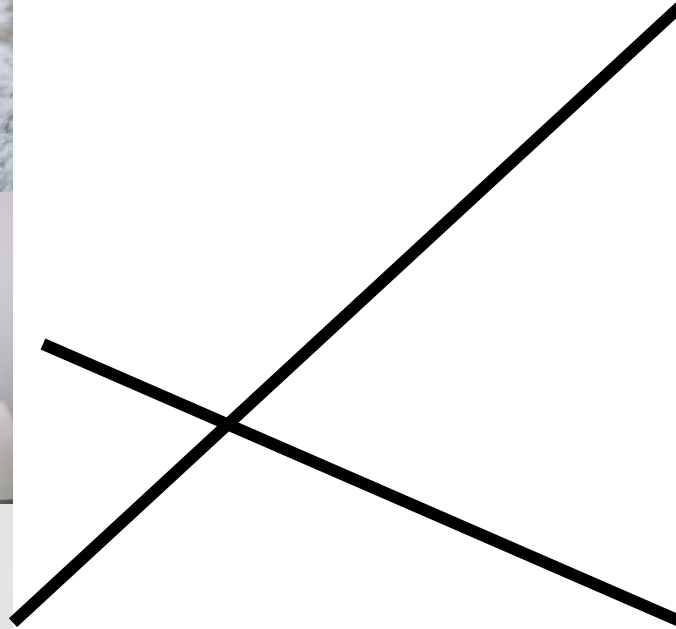
Family A



Family B



Family C



Pet Assignments

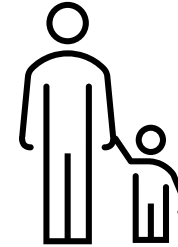
Peter



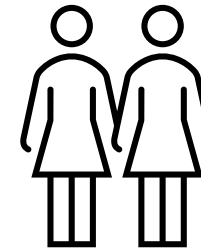
Kat



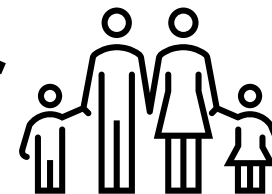
Avery



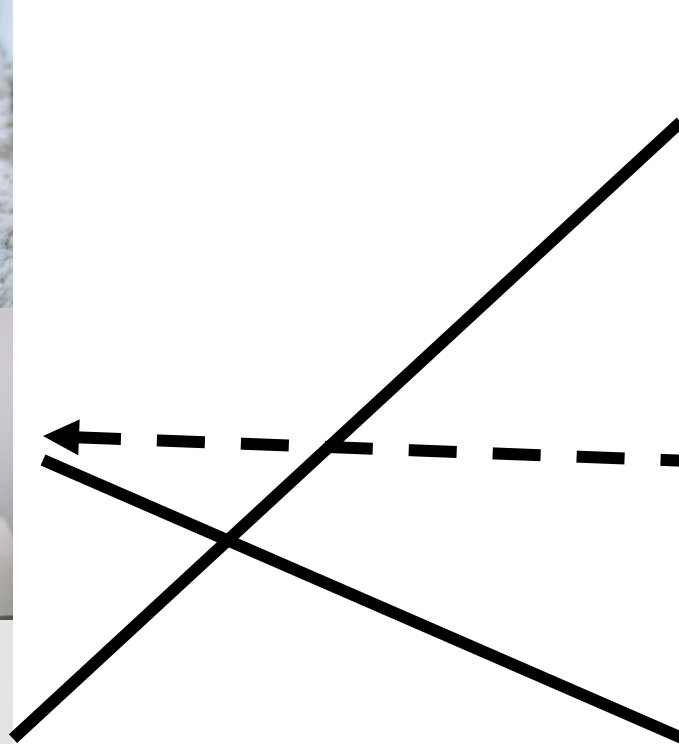
Family A



Family B



Family C



Pet Assignments

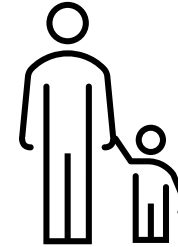
Peter



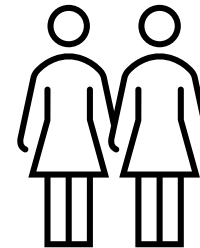
Kat



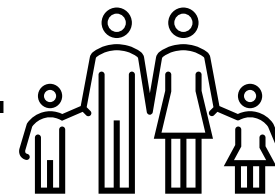
Avery



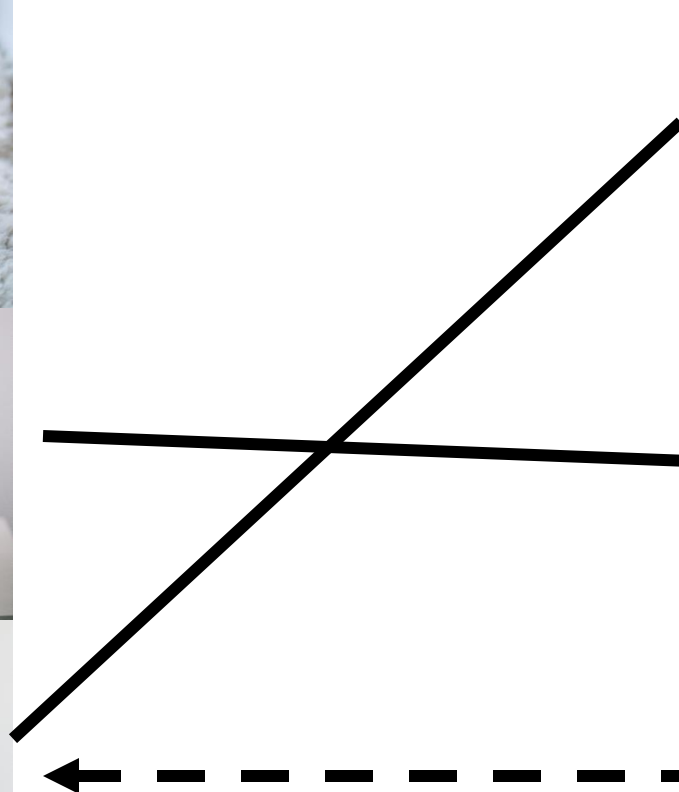
Family A



Family B



Family C



Pet Assignments

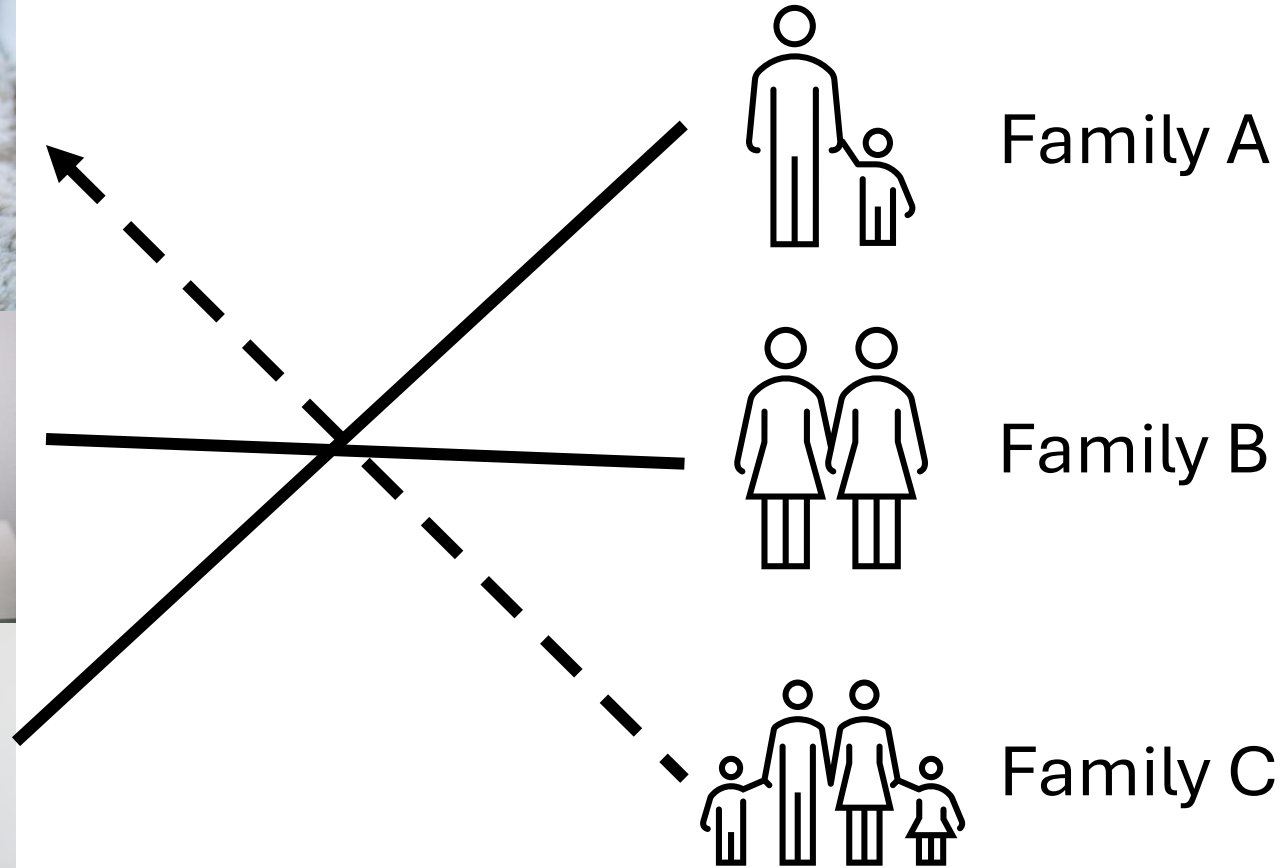
Peter



Kat



Avery



Pet Assignments

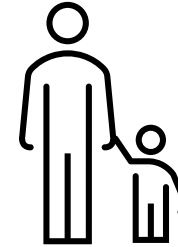
Peter



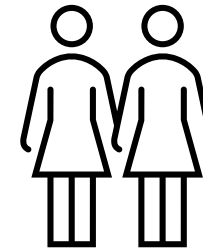
Kat



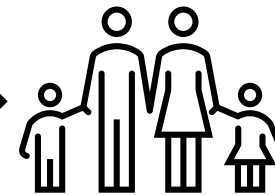
Avery



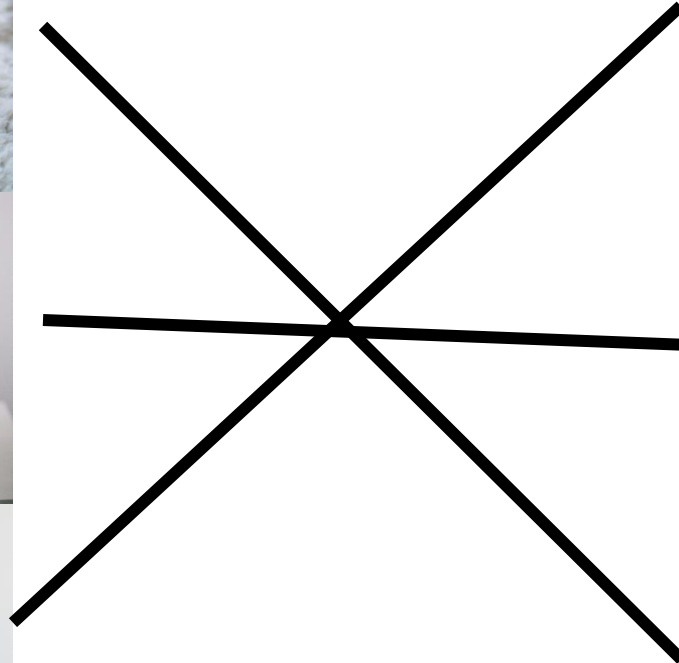
Family A



Family B



Family C



Pet Assignments (Stable State?)

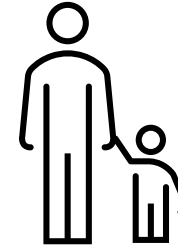
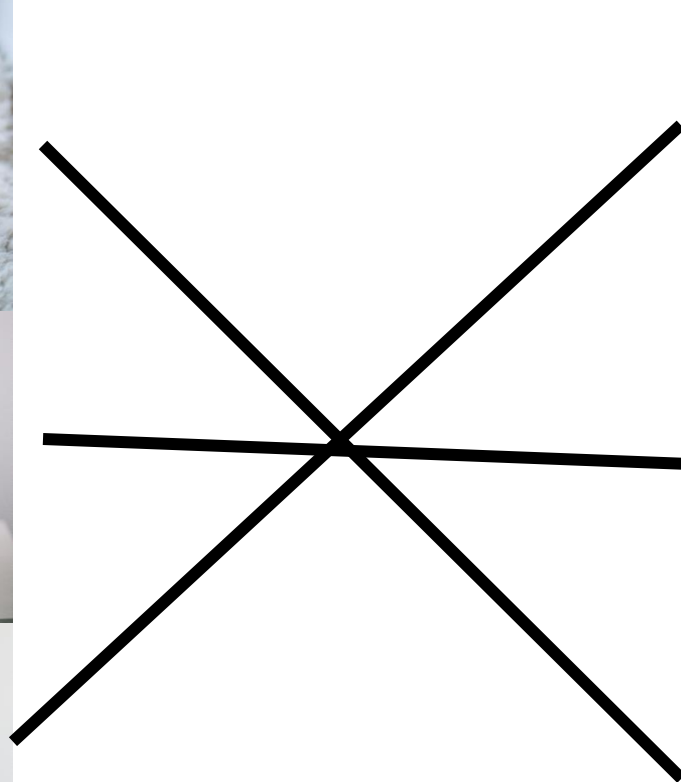
Peter



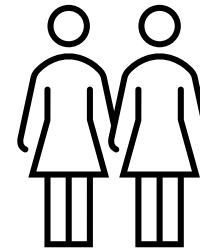
Kat



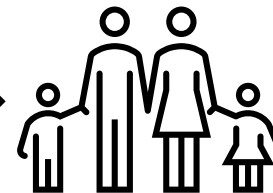
Avery



Family A

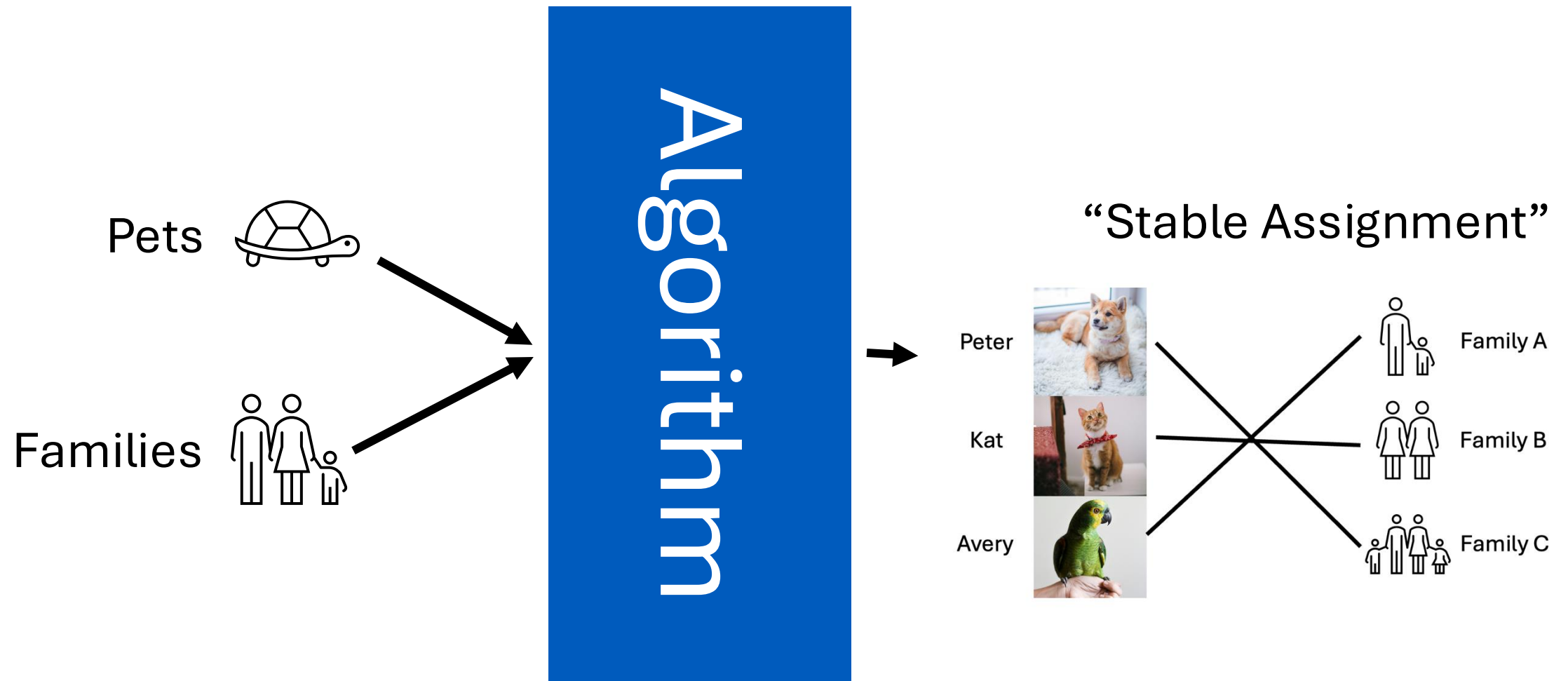


Family B

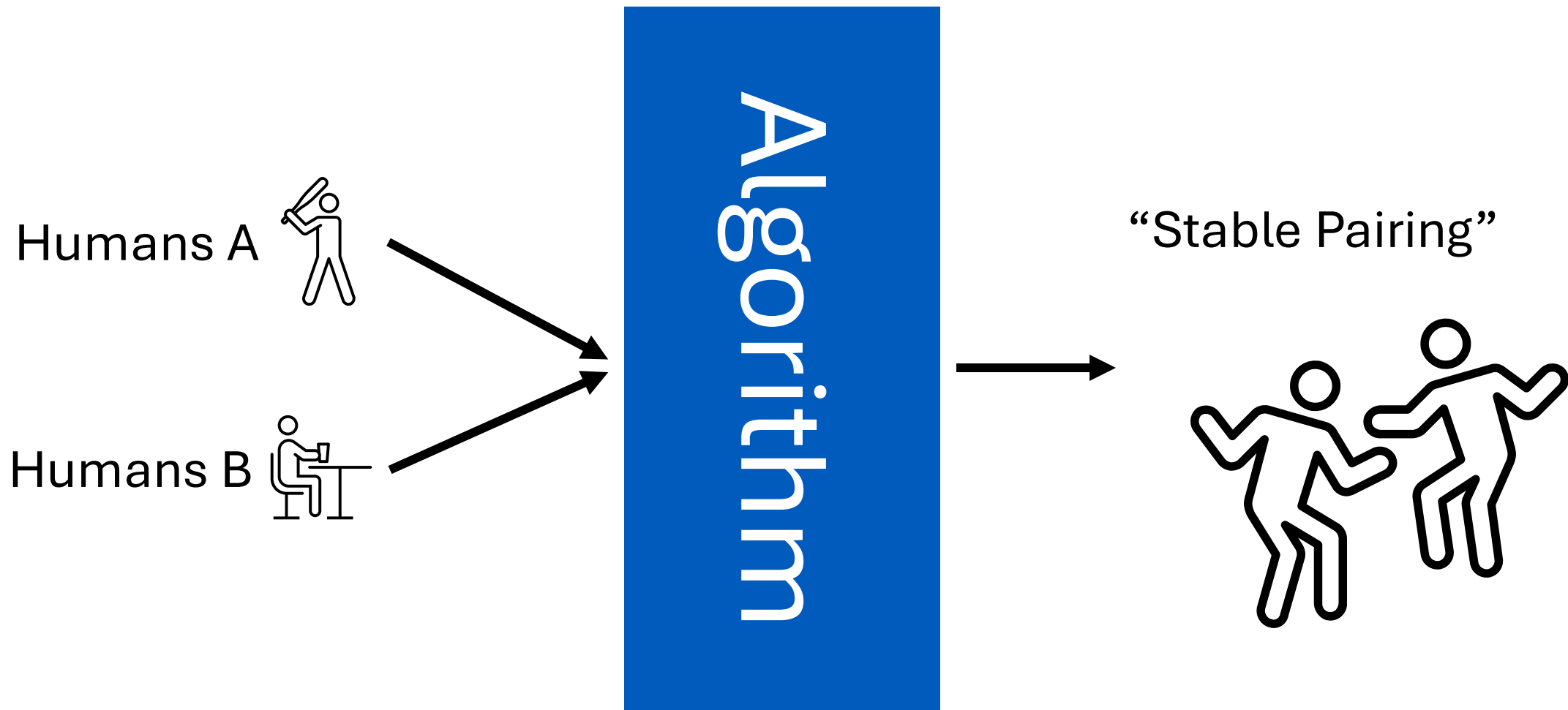


Family C

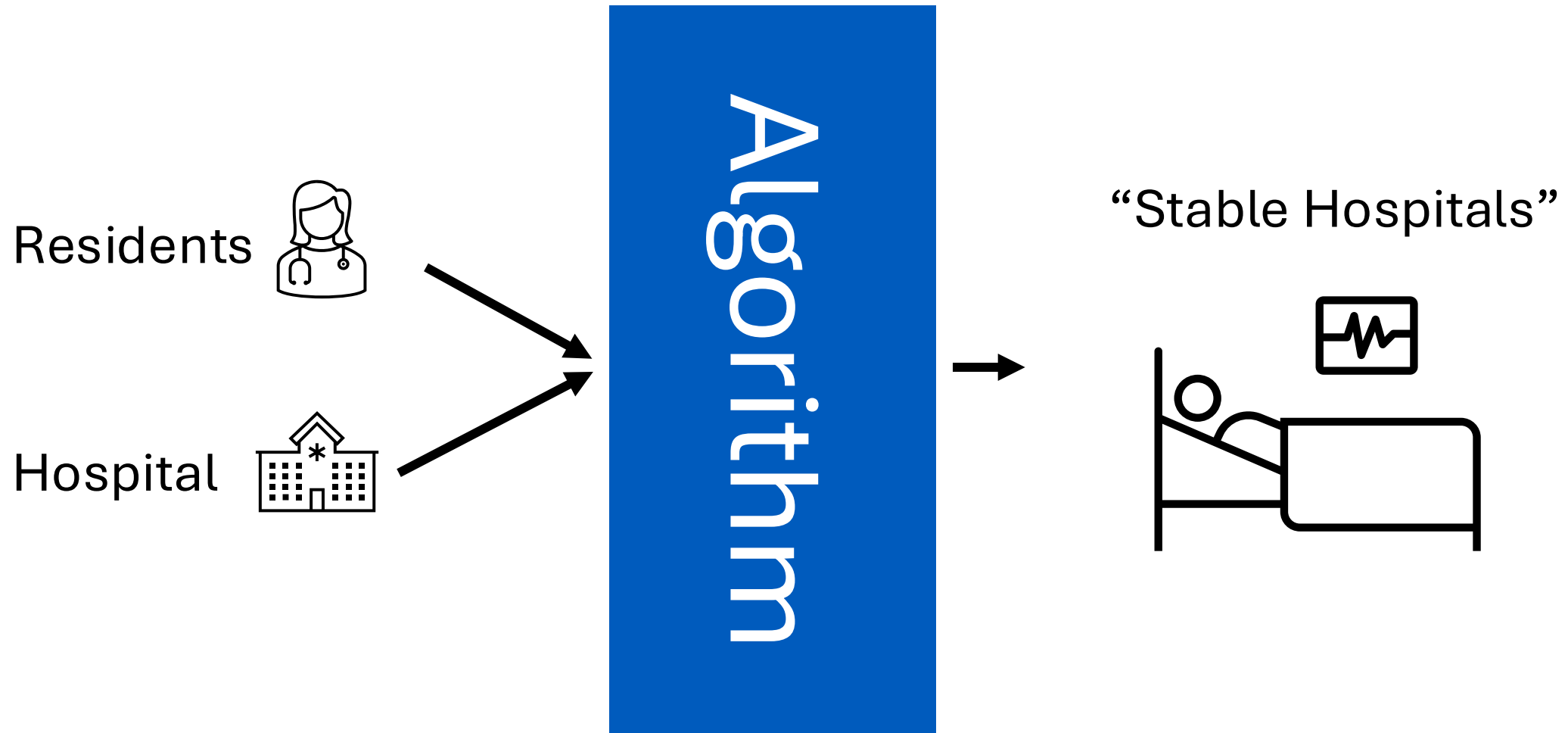
Pet Assignments Problem



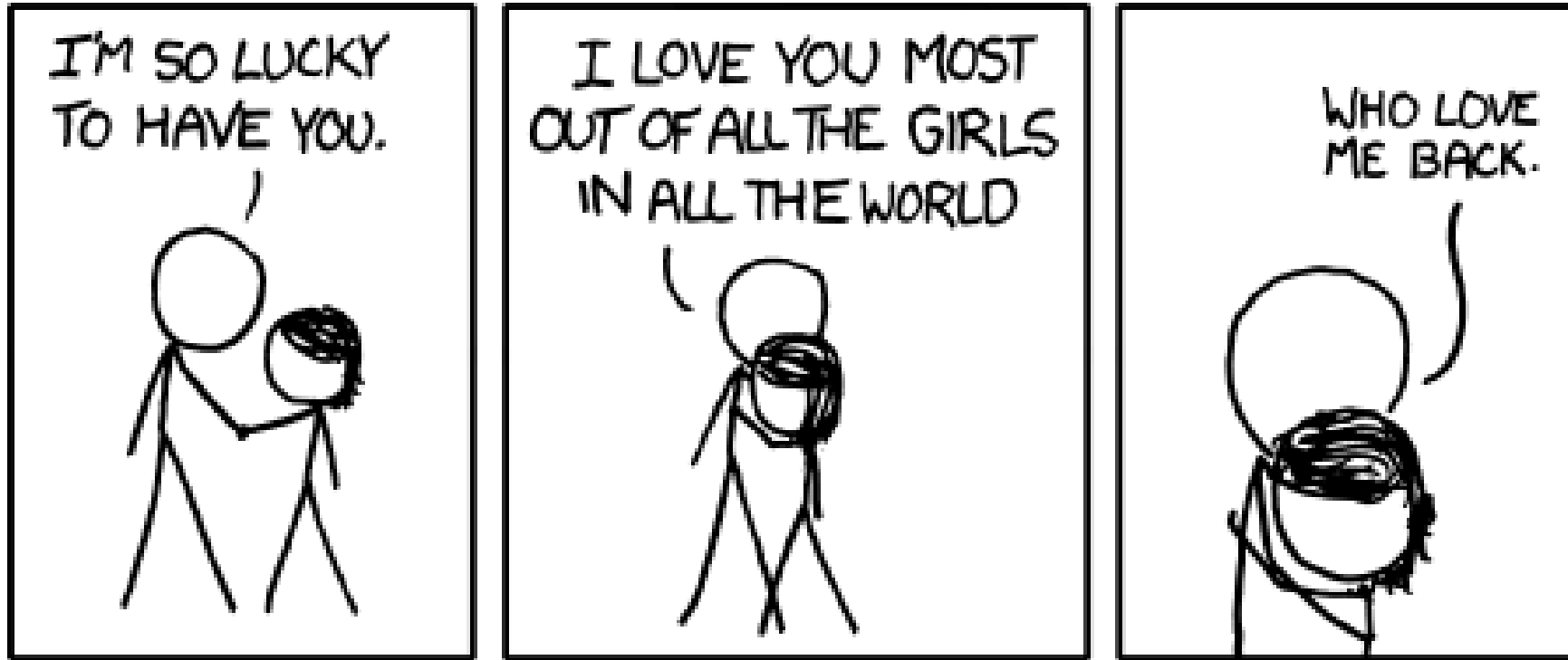
“Stable Marriage Problem”



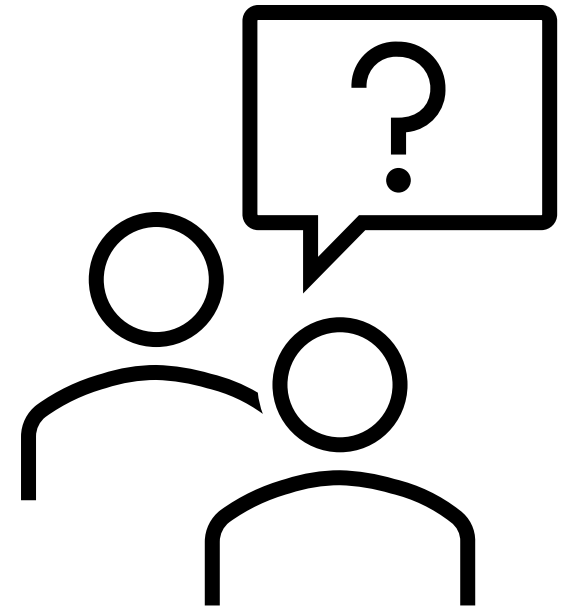
Hospital Resident Assignments Problem



What is stability?



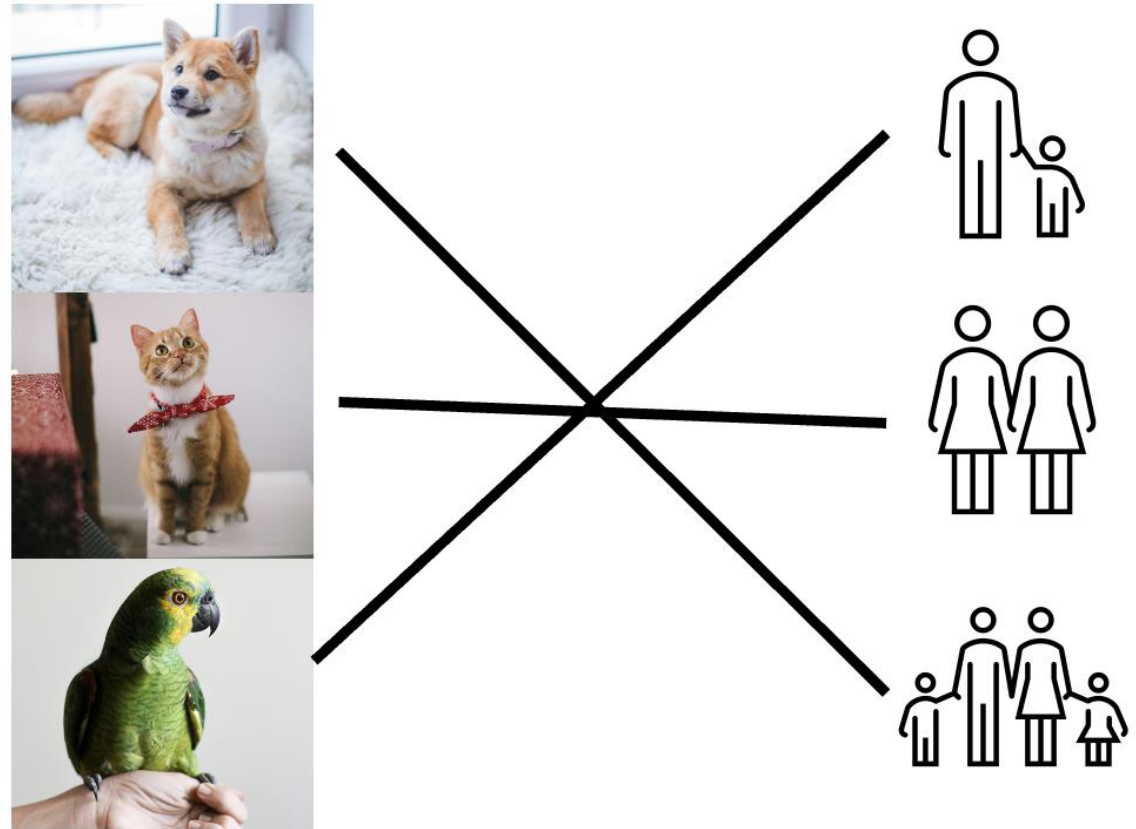
“You know I’ll never leave you. Not as long as she’s with someone else.”



Pet Assignments (Stable State?)

We say that an assignment M is **stable** if there exist no pet p and family f such that

- p prefers f to their current family and
- f prefers p to their current pet.



D > E > F

Environments (Unstable)

Spider > Bunny

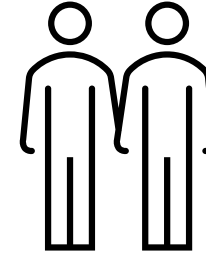
Clover



Dr. Fluffy



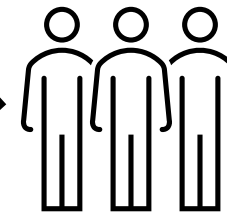
Ronin



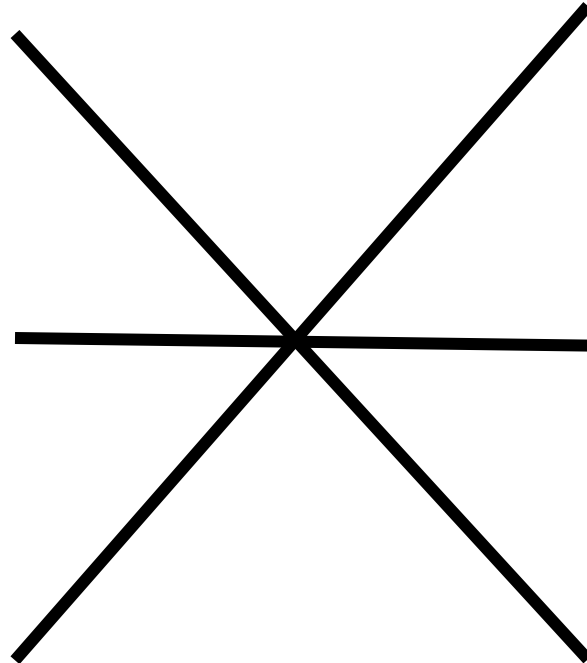
Family D



Family E



Family F



Pet Assignments (Unstable)

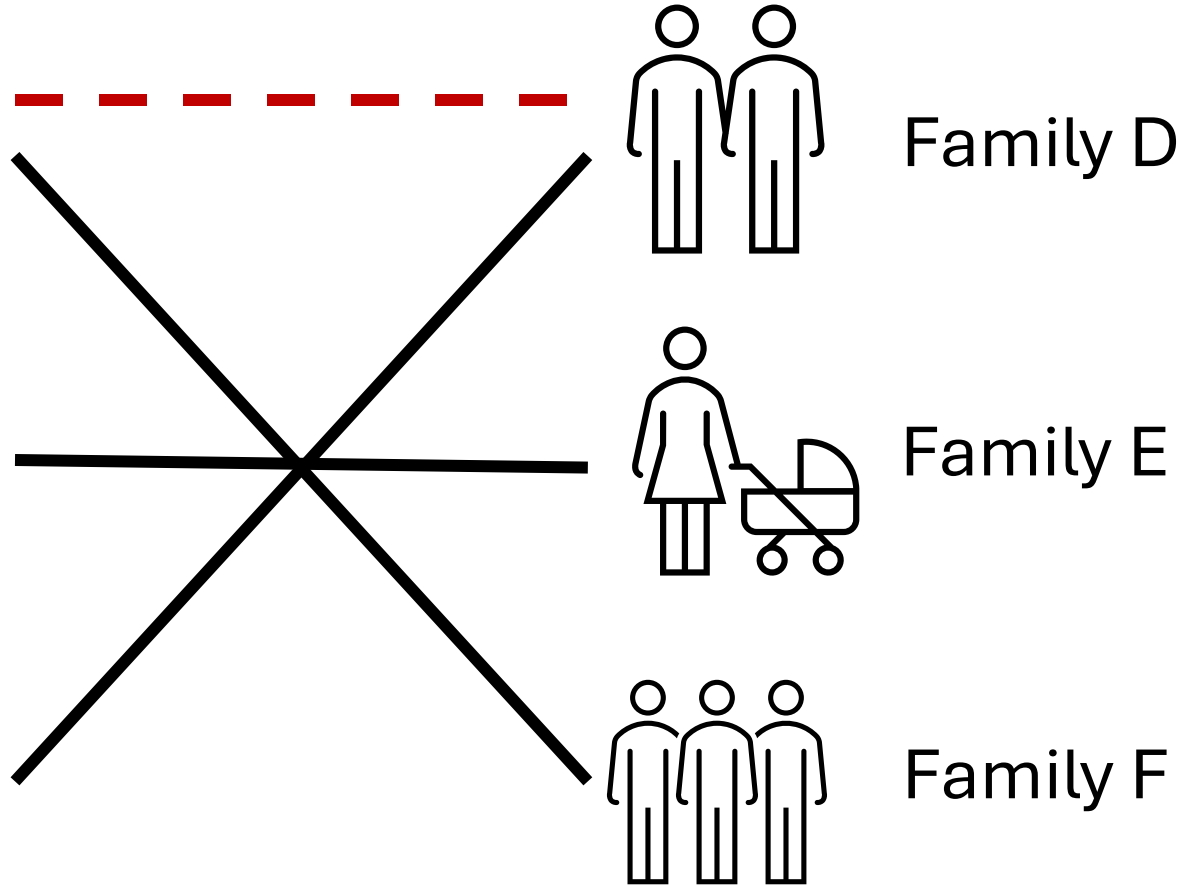
Clover



Dr. Fluffy



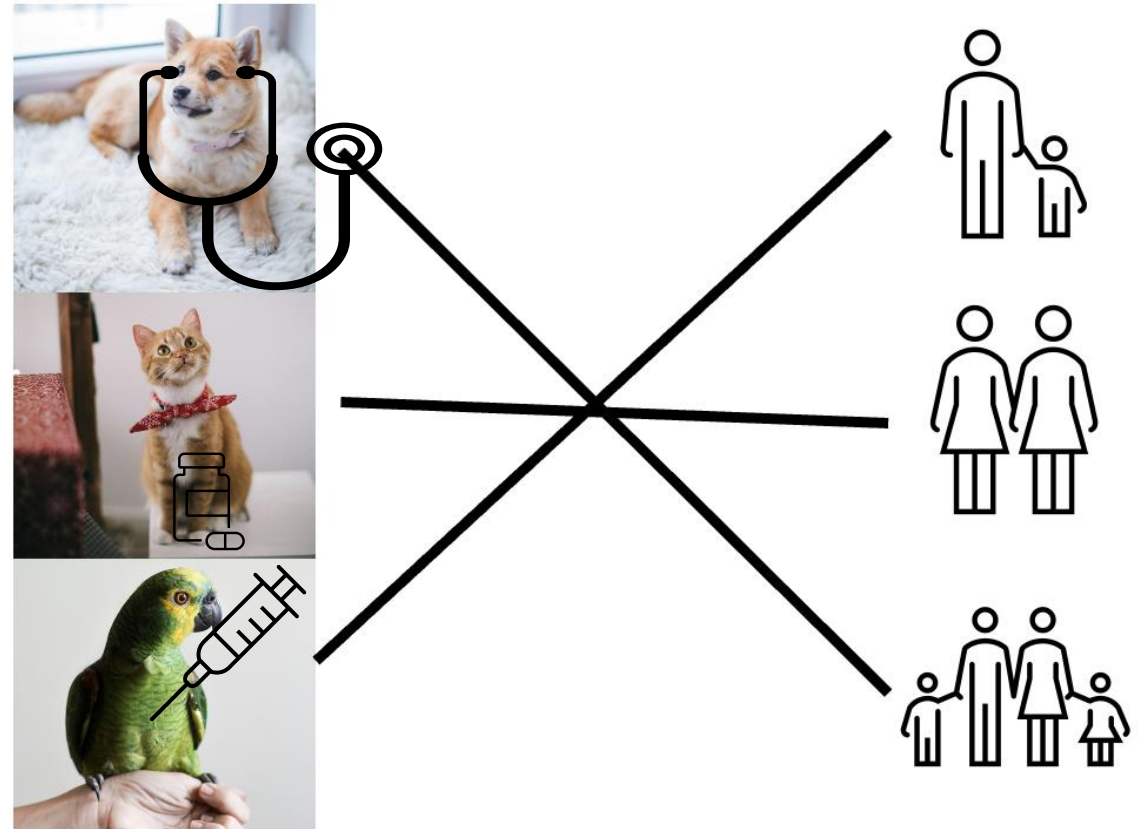
Ronin



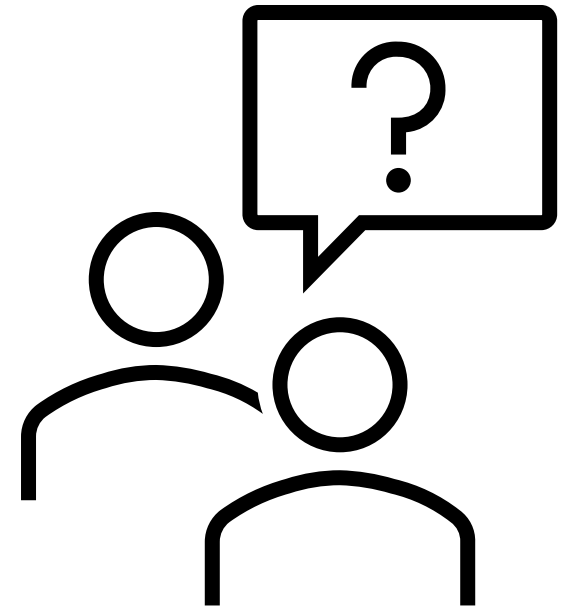
Hospital Assignments (Stable State?)

We say that an assignment M is **stable** if there exist no ___ p and ___ f such that

- p prefers f to their current ___ and
- f prefers p to their current ___.



Can we always find a stable assignment?



Can we always find a stable assignment?

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h 's list to whom h has not yet proposed.

 IF (s is unmatched)

 Add h – s to matching M .

 ELSE IF (s prefers h to current partner h')

 Replace h' – s with h – s in matching M .

 ELSE

s rejects h .

RETURN stable matching M .

Pet Assignments (Stable State?)

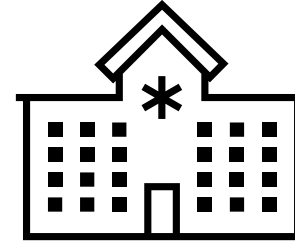
Dr. One



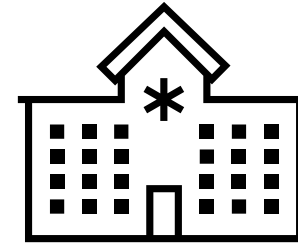
Dr. Two



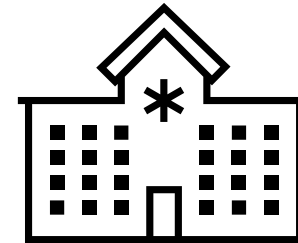
Dr. Three



Hospital A



Hospital B



Hospital C

Pet Assignments (Stable State?)

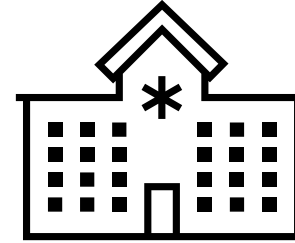
Dr. One
 $A > B > C$



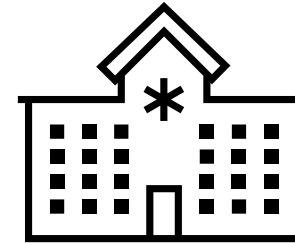
Dr. Two
 $A > C > B$



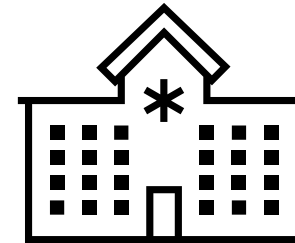
Dr. Three
 $C > A > B$



Hospital A
 $2 > 1 > 3$




Hospital B
 $1 > 2 > 3$




Hospital C
 $3 > 2 > 1$

Pet Assignments (Stable State?)

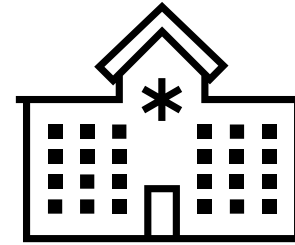
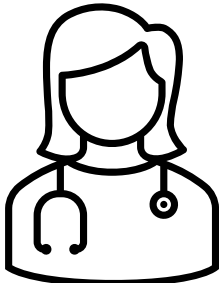
Dr. One
 $A > B > C$



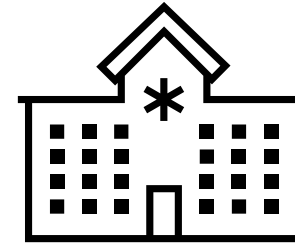
Dr. Two
 $A > C > B$



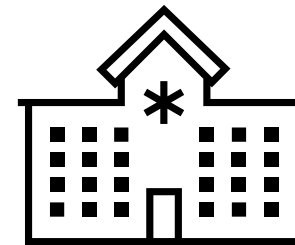
Dr. Three
 $C > A > B$



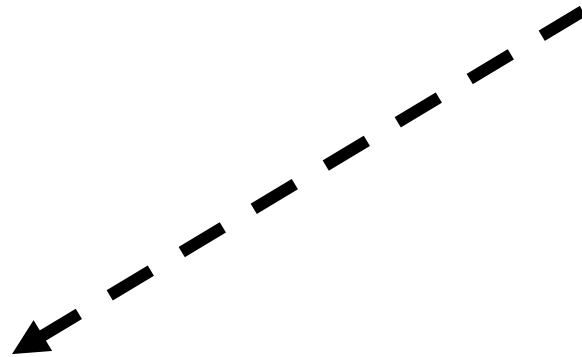
Hospital A
 $2 > 1 > 3$



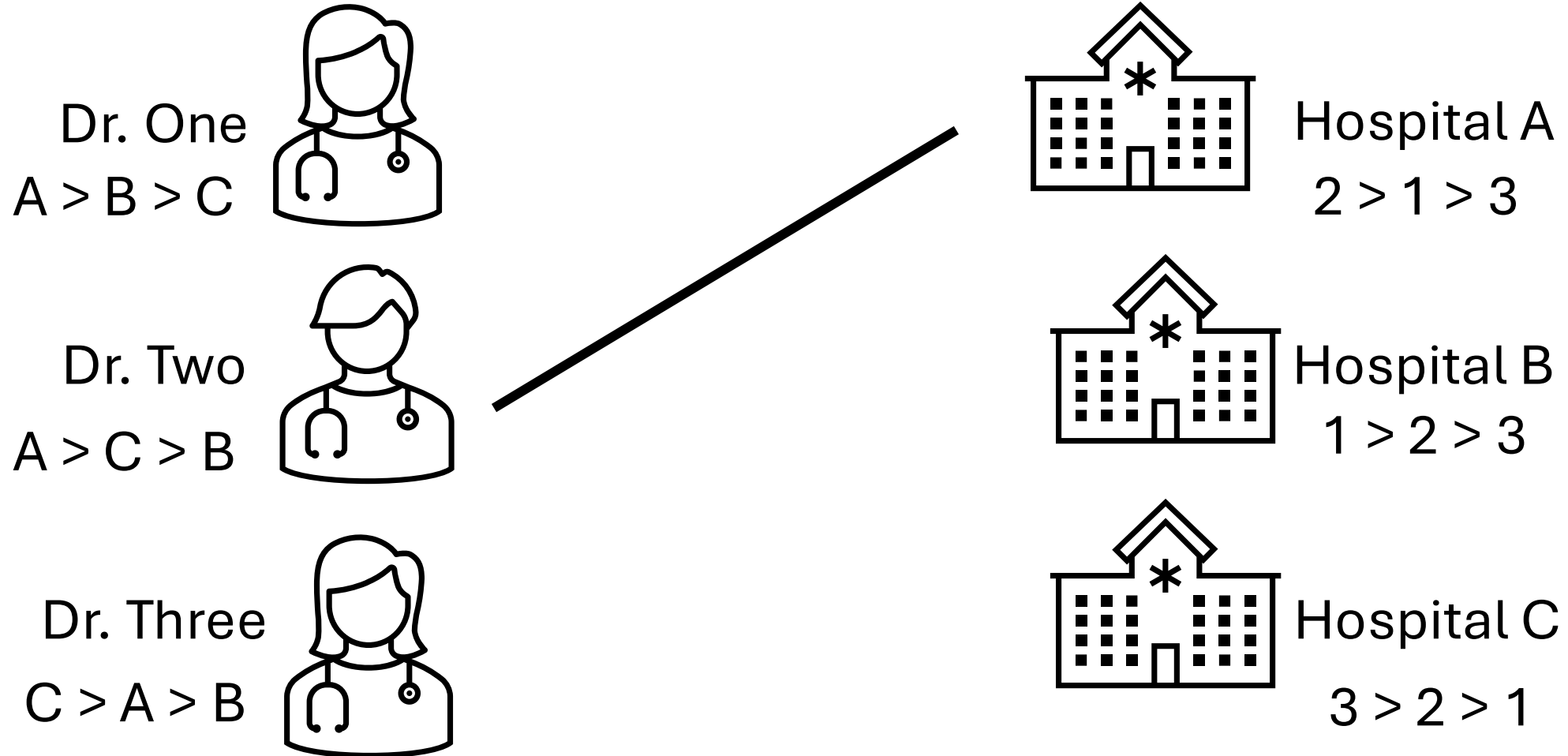
Hospital B
 $1 > 2 > 3$



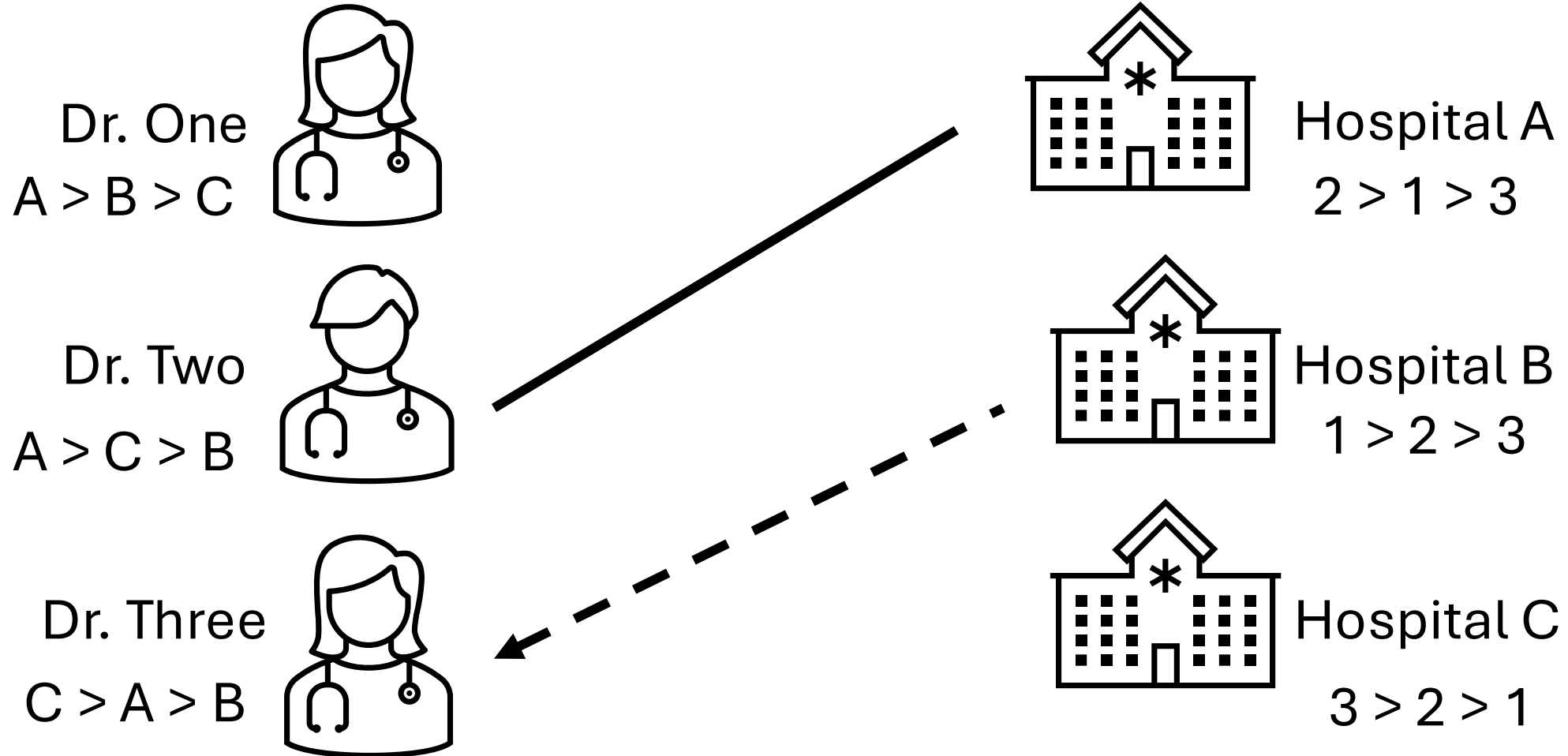
Hospital C
 $3 > 2 > 1$



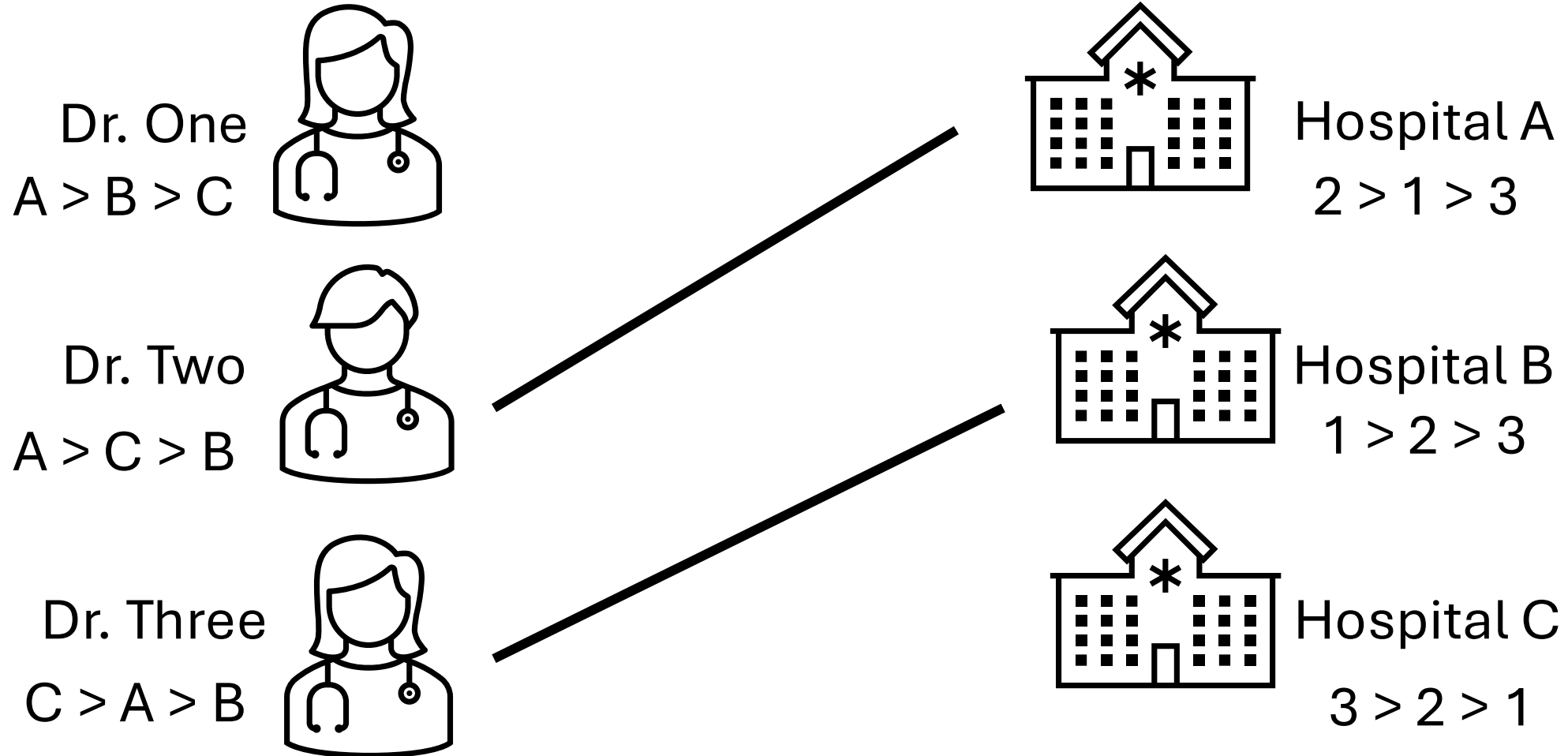
Pet Assignments (Stable State?)



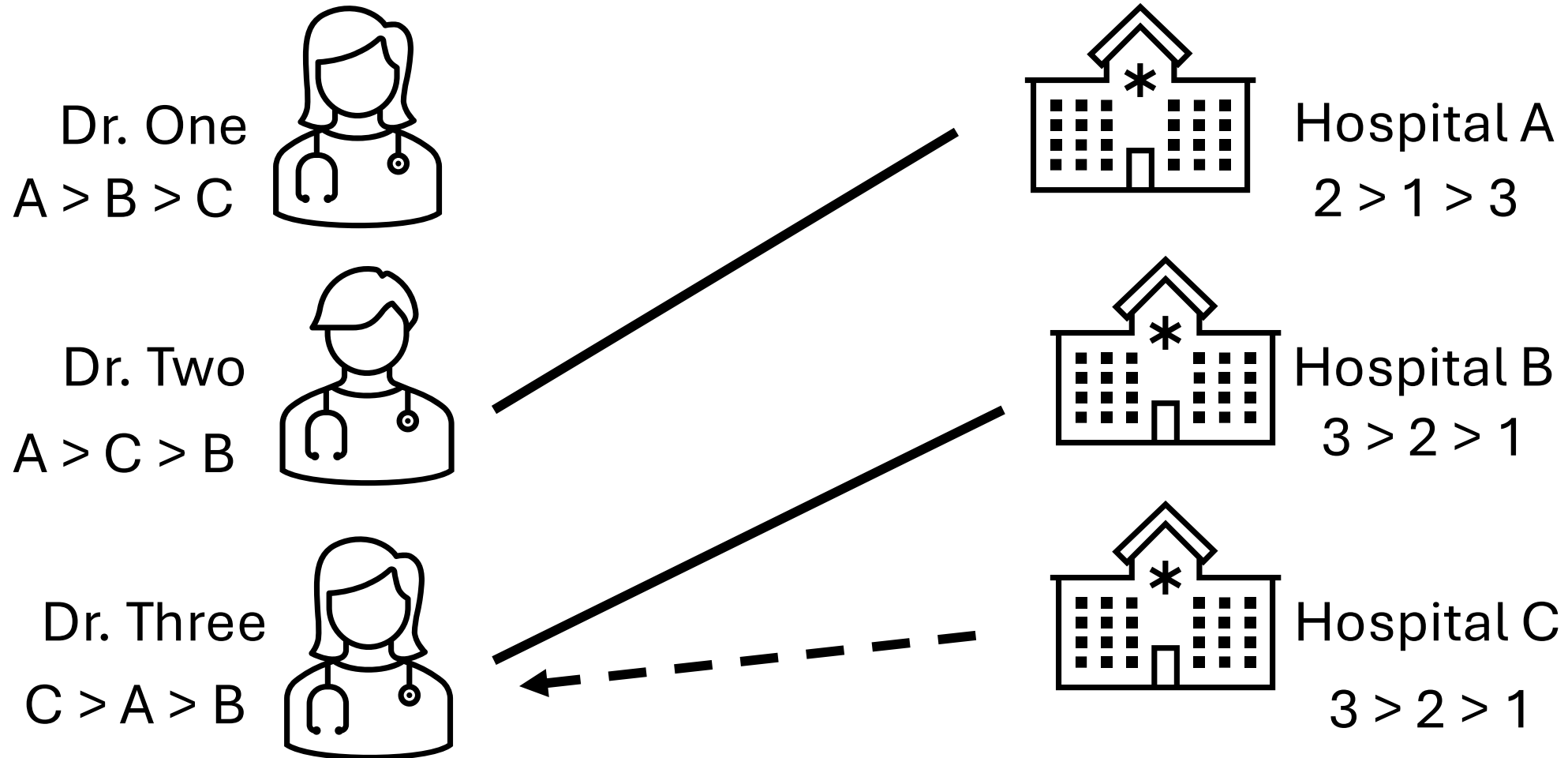
Pet Assignments (Stable State?)



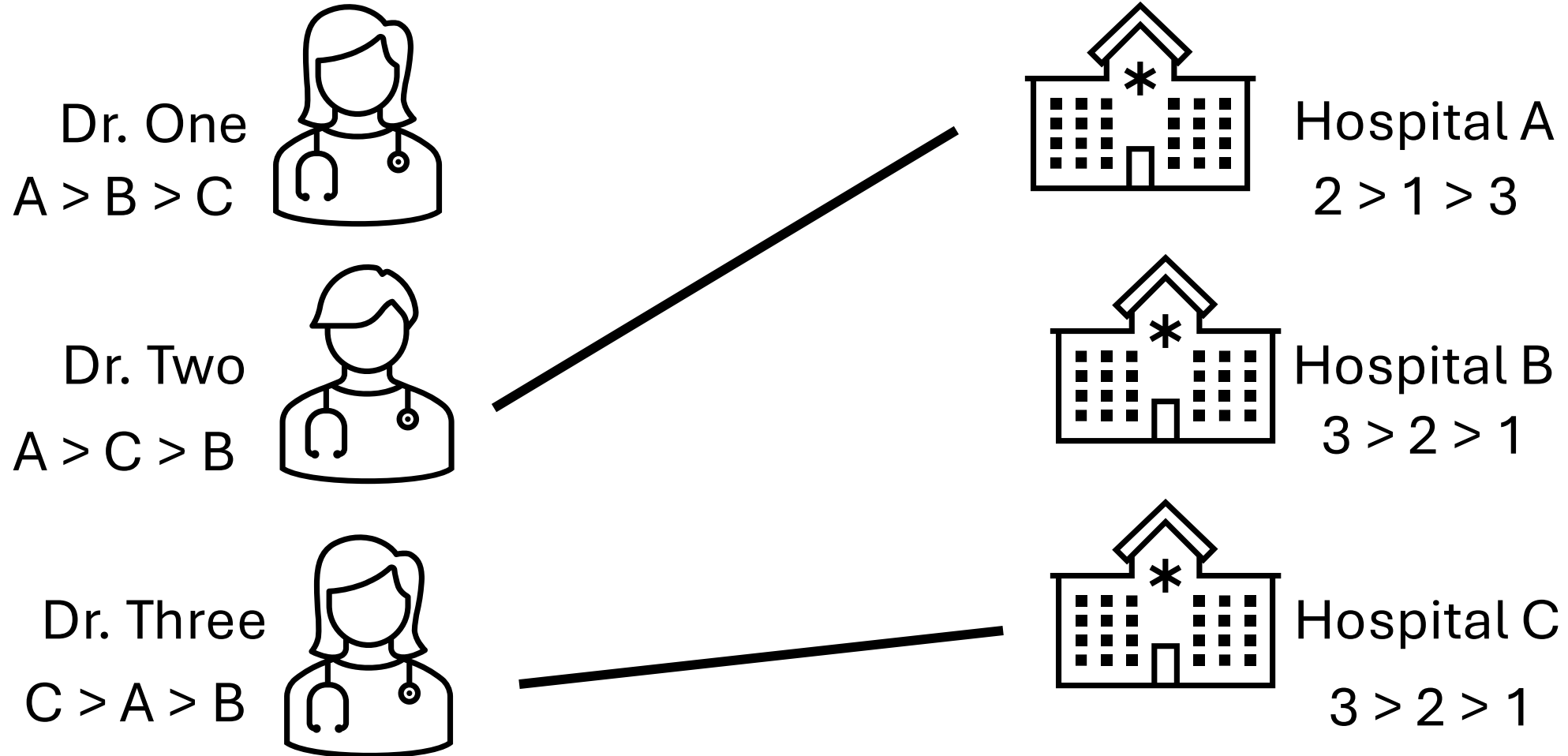
Pet Assignments (Stable State?)



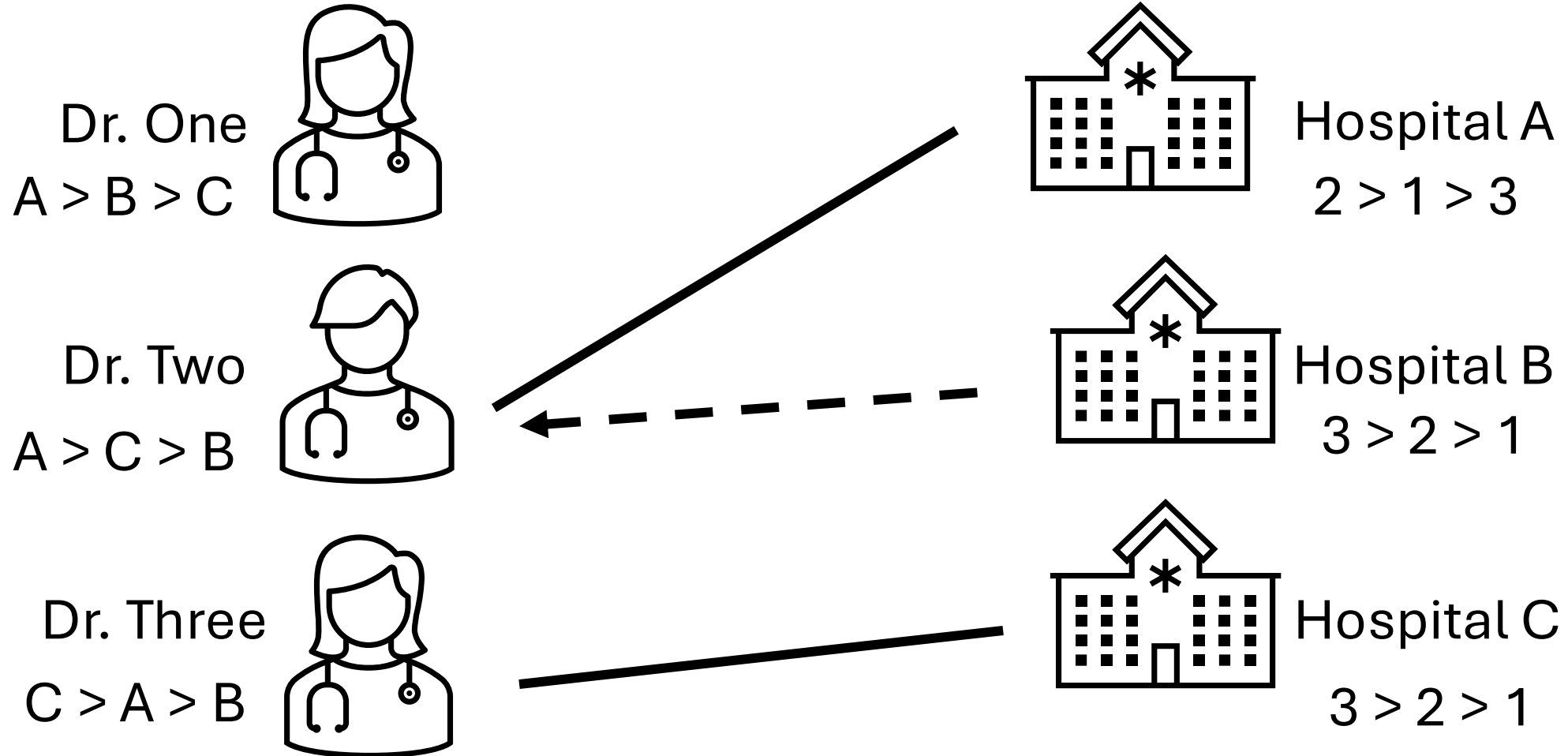
Pet Assignments (Stable State?)



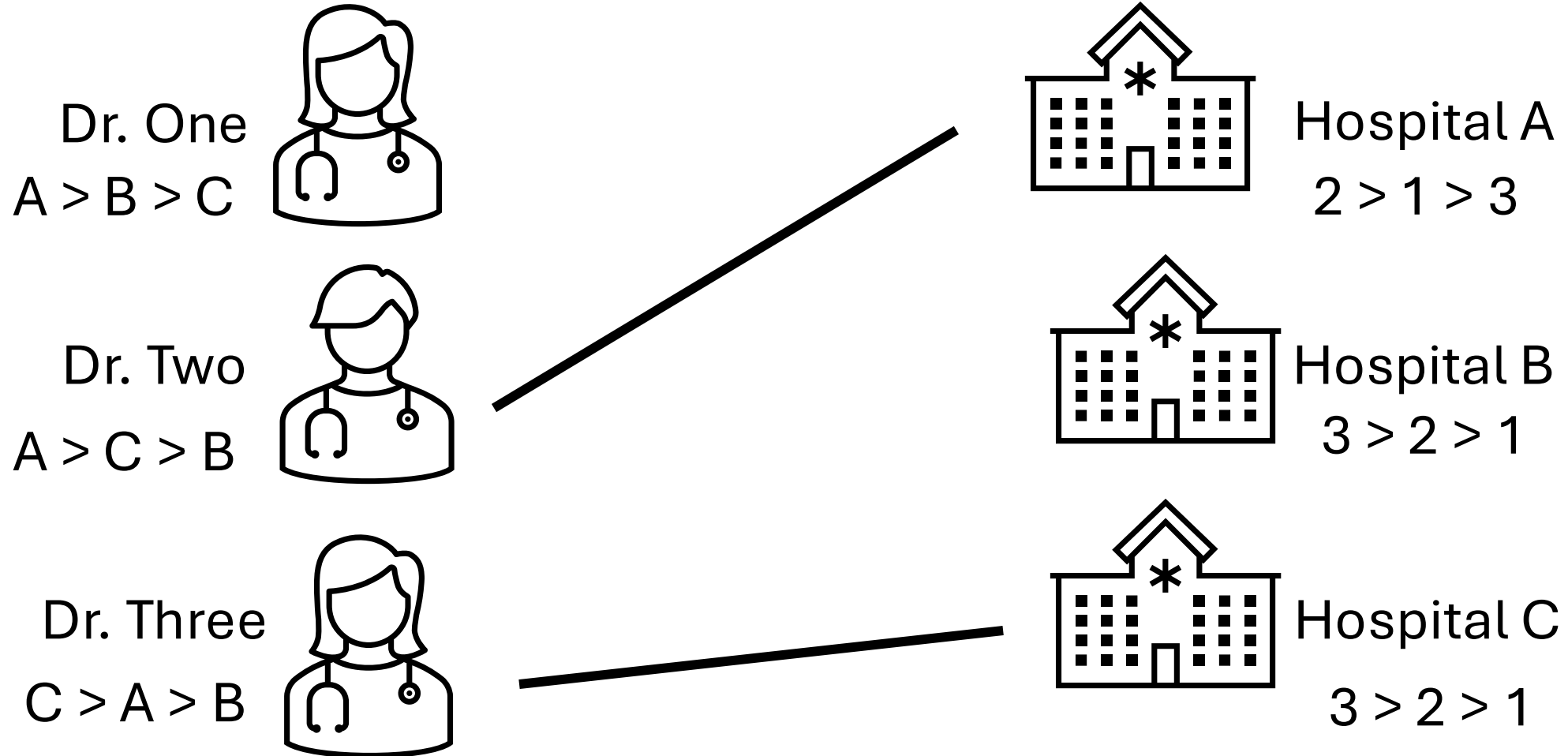
Pet Assignments (Stable State?)



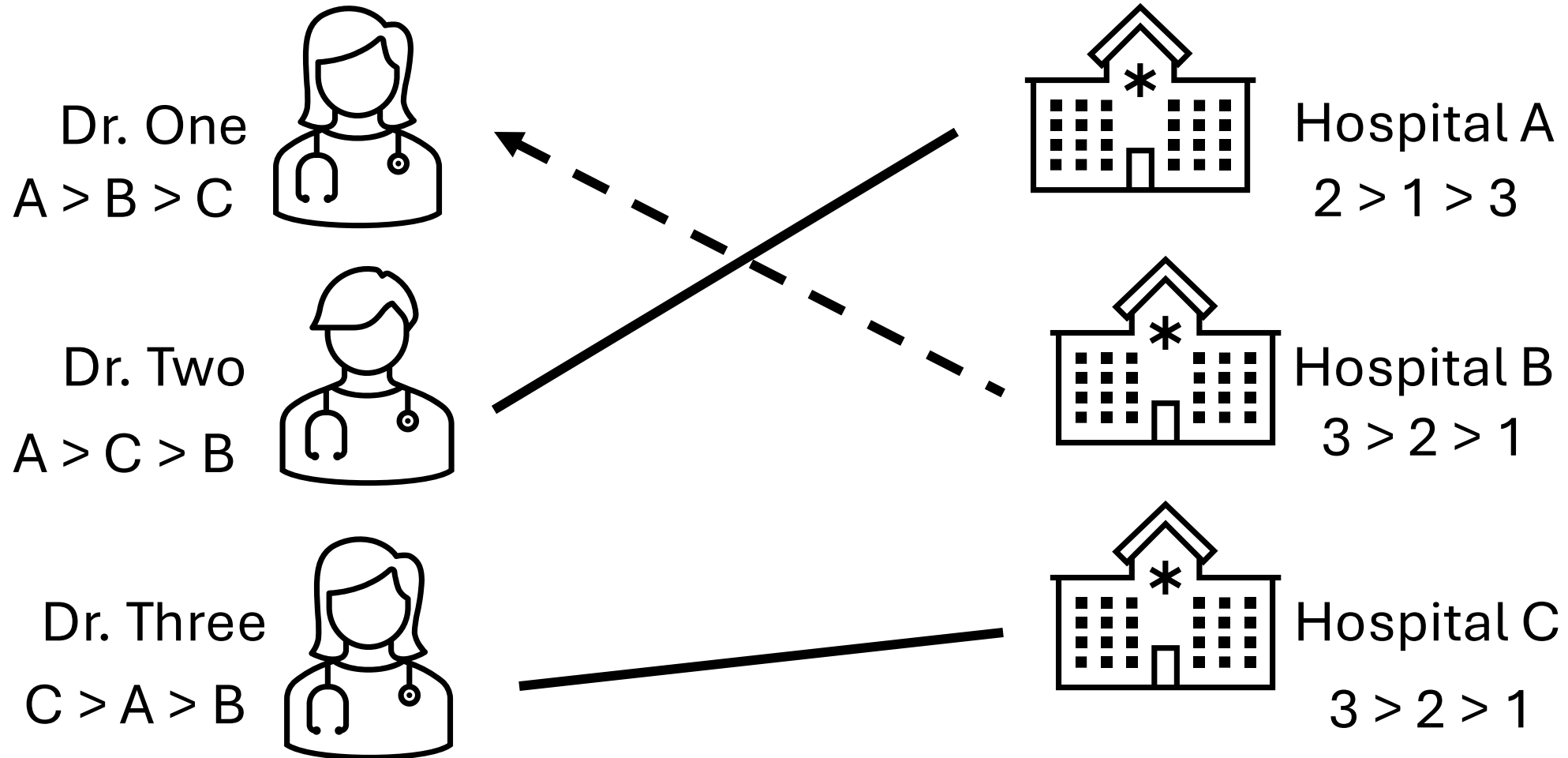
Pet Assignments (Stable State?)



Pet Assignments (Stable State?)



Pet Assignments (Stable State?)



Pet Assignments (Stable State?)

