Self-Stabilizing Card Games

Alex Klinkhamer

March 4, 2013
Two chopsticks are required for eating. Adjacent philosophers cannot eat simultaneously. Choose left, then right? Deadlock!

Descartes: I think, therefore I have a plan!

Alex Klinkhamer | Self-Stabilizing Card Games
Two chopsticks are required for eating.
Two chopsticks are required for eating.

Adjacent philosophers cannot eat simultaneously.
Dining Philosophers

Two chopsticks are required for eating. Adjacent philosophers cannot eat simultaneously. Choose left, then right?

Descartes: I think, therefore I have a plan!
Dining Philosophers

Two chopsticks are required for eating. Adjacent philosophers cannot eat simultaneously.

Choose left, then right?

Deadlock!

Philosophers

Chopsticks
Dining Philosophers

Two chopsticks are required for eating. Adjacent philosophers cannot eat simultaneously. Choose left, then right?

Deadlock!

Descartes: I think, therefore I have a plan!
Each philosopher gets two cards:

- ♠ means philosopher is hungry
- ♠ means philosopher has chopsticks

**Goal:** All philosophers eventually eat

**Protocol:**
- If hungry and not eating, eat if both chopsticks are free
- If eating, eventually stop being hungry
- When full, stop eating
Dining Philosophers, Atomic Solution

If hungry, and chopsticks available, start eating.

If eating, stop being hungry.

When full, stop eating.
All start as hungry and not eating.
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness. We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

Mill: We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I’m still $\frac{1}{16}$ hungry!

Mill: This method is inefficient and does not maximize happiness.

Mill: We can eat in two steps, three philosophers at a time!
Dining Philosophers, Atomic Solution

All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

Mill: We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

Mill: We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I’m still hungry!

Mill: This method is inefficient and does not maximize happiness.

Mill: We can eat in two steps, three philosophers at a time!
All start as hungry and not eating.

Mill: This method is inefficient and does not maximize happiness.
All start as hungry and not eating.

Mill: Hurry up...

Zeno: I'm still hungry!

Mill: This method is inefficient and does not maximize happiness.

We can eat in two steps, three philosophers at a time!
Each philosopher gets two cards, but only shows one!

- ♠ means philosopher will eat first
- ♥ means philosopher will eat second

**Goal:** No two adjacent colors are the same

**Protocol:**
- If your card has same color as your right neighbor, then switch cards
2-Coloring

If both red, then change to black.

If both black, then change to red.
Mill: We can cooperate to make a 2-coloring.

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...
Mill: We can cooperate to make a 2-coloring.

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
2-Coloring

Mill: We can cooperate to make a 2-coloring.

q

q

q

q

♣

q

q

q

q

♣

q

q

q

q

♣

q

q

q

q

♣

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

q

q

q

q

♣

q

q

q

q

♣

q

q

q

q

♣

q

q

q

q

♣

q

q

q

q

♣

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Black eats first, red eats second.

Popper: Perhaps some of us are not so agreeable...

Livelock!
Mill: We can cooperate to make a 2-coloring.

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!

Alex Klinkhamer  Self-Stabilizing Card Games
Mill: We can cooperate to make a 2-coloring.

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
2-Coloring

Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...
Mill: We can cooperate to make a 2-coloring.

q

q

♣

q

♣

q

♣

q

♣

q

♣

q

♣

q

♣

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

♣

q

♣

q

♣

q

♣

q

♣

q

♣

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

♣

q

♣

q

♣

q

♣

q

♣

q

♣

Livelock!
Mill: We can cooperate to make a 2-coloring.

q

♣

q

♣

♣

♣

Mill: Black eats first, red eats second.

Popper: Wait! This method has a counterexample!

♣

q

♣

♣

♣

q

♣

♣

♣

♣

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

♣

♣

♣

♣

Livelock!

♣

♣

♣

♣

♣

Alex Klinkhamer

Self-Stabilizing Card Games
2-Coloring

Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
Mill: We can cooperate to make a 2-coloring.

Popper: Wait! This method has a counterexample!

Mill: Not when we cooperate!

Popper: Perhaps some of us are not so agreeable...

Livelock!
Each philosopher gets two cards, but always shows one (or the back side)

- ♠ means philosopher will eat first
- ♣ means philosopher will eat second
- ♣ means philosopher will eat third

**Goal:** No two adjacent colors are the same

**Protocol:**

- If your card has the same color as your left or right neighbor, then switch it to be different
Mill: Try to find a livelock in this 3-coloring protocol, Popper.
Try to find a livelock in this 3-coloring protocol, Popper.

Popper: Touché, let's eat!
Try to find a livelock in this 3-coloring protocol, Popper.

Popper: Touché, let's eat!
After eating again, the philosophers do what they do best.
After eating again, the philosophers do what they do best
  - They argue with each other
After eating again, the philosophers do what they do best
  • They argue with each other
• It gets too noisy
After eating again, the philosophers do what they do best
  ● They argue with each other

It gets too noisy

Enter: Token ring protocol
  ● Only the philosopher with the token has the privilege to talk
Each player has one card which can be face-up or face-down

- or

Initially all cards are face-down

When a player can act, they have a token

**Goal:** Exactly one philosopher has a token

**Protocol:**

- For $P_0$, if your right neighbor has a card facing the same as yours, then flip your card
- For everyone else, if your right neighbor a card facing opposite yours, then flip your card to match
Let there be $N$ players

Each player has $N$ cards, show 1 to $N$ of them face-down

When a player can act, they have a *token*

**Goal:** Exactly one token exists

**Protocol:**
- For $P_0$, if your right neighbor has the same number of cards, then show one more
  - Or, if you don’t have any more cards, just show one
- For everyone else, if your right neighbor is showing a different number of cards, then add or remove your cards to match
Card Arrangement

5 columns.

15

5 columns.
Card Arrangement

Remove from bottom row.
Card Arrangement

5 columns.
Remove from bottom row.
Add to bottom row.

15?

Add to bottom row.
Card Arrangement

Add to bottom row.
Card Arrangement

18

Any column!
Let there be \( N \) players, where \( N \) is prime
- \( N \) does not need to be prime for agreement [1]

Pick a number
- Any number from 0 to \( N - 1 \)

Compute the number of cards to put between you and your left neighbor
- Your pick minus your left neighbor’s pick
- If the difference is not positive, then add \( N \) to it

Goal:
- Number of cards between everyone is the same
- Initial pick + number of moves (mod \( N \)) is different for everyone

Protocol:
- If the stack to your right contains more cards than the stack to your left, then move a card from right to left
- If both stacks have \( N \) cards, then take one card from right and take all but one from left
  - Only needed when everyone’s initial choice is the same
Cleanup - Phase 1

- Pass black suits to the left
- Pass red suits to the right
- If spade and diamond meet, then put the pair in front of you
  - With spade suit on top
- If heart and club meet, then put the pair in front of you
  - With heart suit on top
Cleanup - Phase 1

Self-Stabilizing Card Games
Cleanup - Phase 1

Alex Klinkhamer
Self-Stabilizing Card Games
Cleanup - Phase 1

Alex Klinkhamer  Self-Stabilizing Card Games
Cleanup - Phase 1

A³

A²

A²

A²

A³

A³

A²

A³

A³

A³

A³

A³

A³

A³

A³

A³

A³
Cleanup - Phase 1

Alex Klinkhamer
Self-Stabilizing Card Games
Cleanup - Phase 1

Alex Klinkhamer  
Self-Stabilizing Card Games
Cleanup - Phase 2

- Pass spade-diamond pairs to the left
- Pass heart-club pairs to the right
- If two pairs have the same rank, and different color, then put the 4-of-a-kind in front of you
Cleanup - Phase 2

![Diagram showing cards and players]

- $P_0$: 2 spades
- $P_1$: 2 spades
- $P_2$: A spades
- $P_3$: A hearts
- $P_4$: 3 spades

Alex Klinkhamer  |  Self-Stabilizing Card Games
Everyone pass 4-of-a-kinds from right to left

Anyone with an ace, collect one full deck in front of you
  
  If a 4-of-a-kind is in your deck, continue passing to the left
Cleanup - Phase 3

Alex Klinkhamer
Self-Stabilizing Card Games
Cleanup - Phase 3

Alex Klinkhamer  Self-Stabilizing Card Games
Goal: Make self-stabilization more accessible
Goal: Make self-stabilization more accessible
  Why? Very robust systems!
Goal: Make self-stabilization more accessible
  Why? Very robust systems!

Automatically construct self-stabilizing protocols
  Given invariant, topology, and non-stabilizing protocol
Goal: Make self-stabilization more accessible
  Why? Very robust systems!

Automatically construct self-stabilizing protocols
  Given invariant, topology, and non-stabilizing protocol
  Have shown this is NP-complete in size of state space (yikes!)
Goal: Make self-stabilization more accessible
  - Why? Very robust systems!

Automatically construct self-stabilizing protocols
  - Given invariant, topology, and non-stabilizing protocol
  - Have shown this is NP-complete in size of state space (yikes!)

Construct and reason about parameterized self-stabilizing protocols
Goal: Make self-stabilization more accessible
  - Why? Very robust systems!

Automatically construct self-stabilizing protocols
  - Given invariant, topology, and non-stabilizing protocol
  - Have shown this is NP-complete in size of state space (yikes!)

Construct and reason about parameterized self-stabilizing protocols
  - That is, protocols that work for any number of processes
Goal: Make self-stabilization more accessible
  Why? Very robust systems!

Automatically construct self-stabilizing protocols
  Given invariant, topology, and non-stabilizing protocol
  Have shown this is NP-complete in size of state space (yikes!)

Construct and reason about parameterized self-stabilizing protocols
  That is, protocols that work for any number of processes
  Examples: 3-coloring, Dijkstra’s token ring, agreement
Goal: Make self-stabilization more accessible
  Why? Very robust systems!
Automatically construct self-stabilizing protocols
  Given invariant, topology, and non-stabilizing protocol
  Have shown this is NP-complete in size of state space (yikes!)
Construct and reason about parameterized self-stabilizing protocols
  That is, protocols that work for any number of processes
  Examples: 3-coloring, Dijkstra’s token ring, agreement
  Working on some undecidability results (more bad news, but expected)
A simplified proof for a self-stabilizing protocol: A game of cards.  

Self-stabilizing systems in spite of distributed control.  

Leader election in uniform rings.  