Railsconf

ATLANTA 2023

Railsconf
ATLANTA 2023

Off to the races

Kyle d'Oliveira

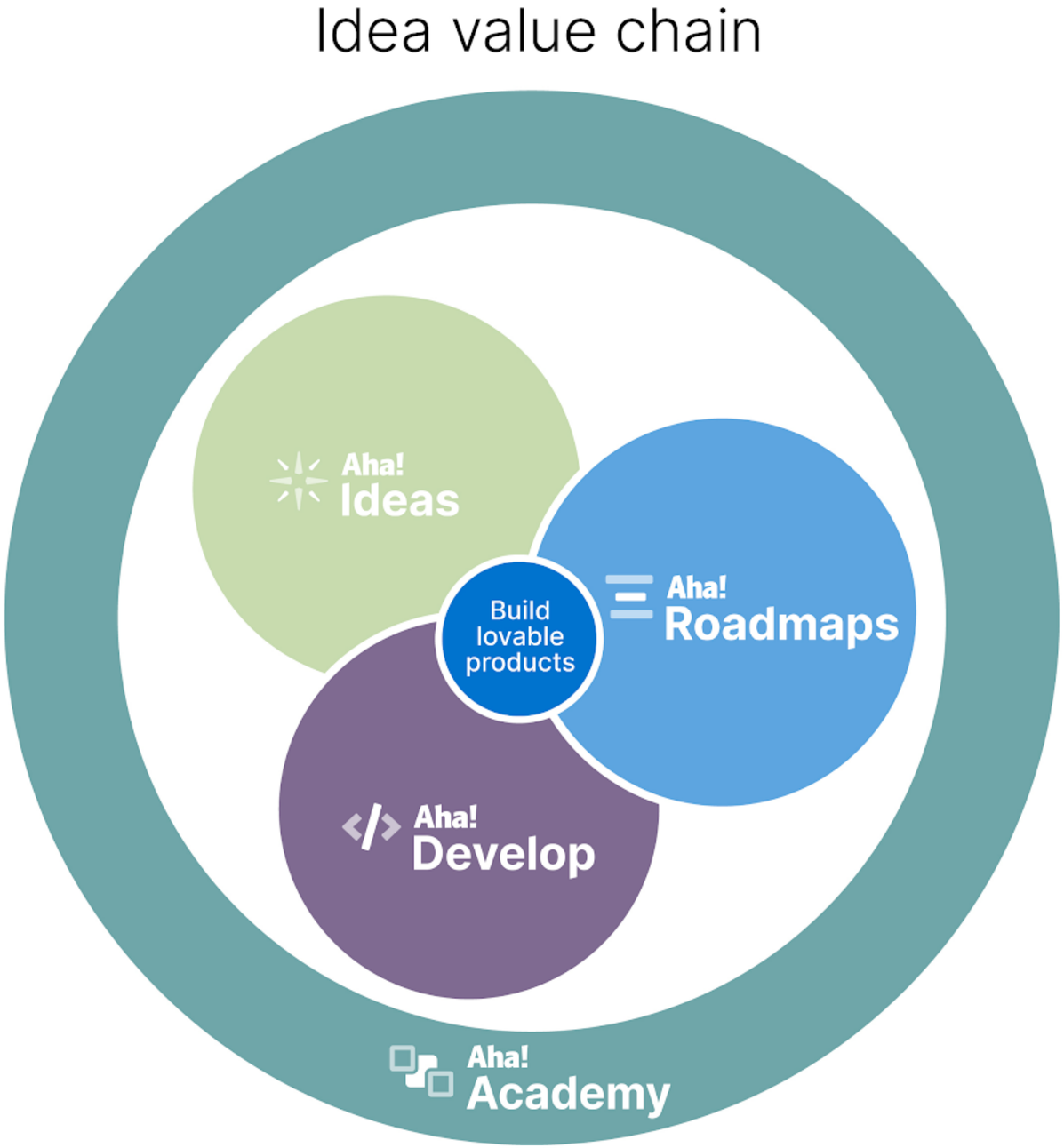
Kyle d’Oliveira
Principal software engineer

https://www.linkedin.com/in/doliveirakn
kyle@aha.io

Aha!

Our Software:

Tools to help companies build what matters



RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

The Responsive Method powers us

Philosophy

A radical new approach to personal and company growth

- Goal first
- Go boldly
- Perfect moments
- Wow, curious!
- Interrupt driven
- Yea or nay now
- Transparent
- Kind

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

RailsConf
ATLANTA 2023

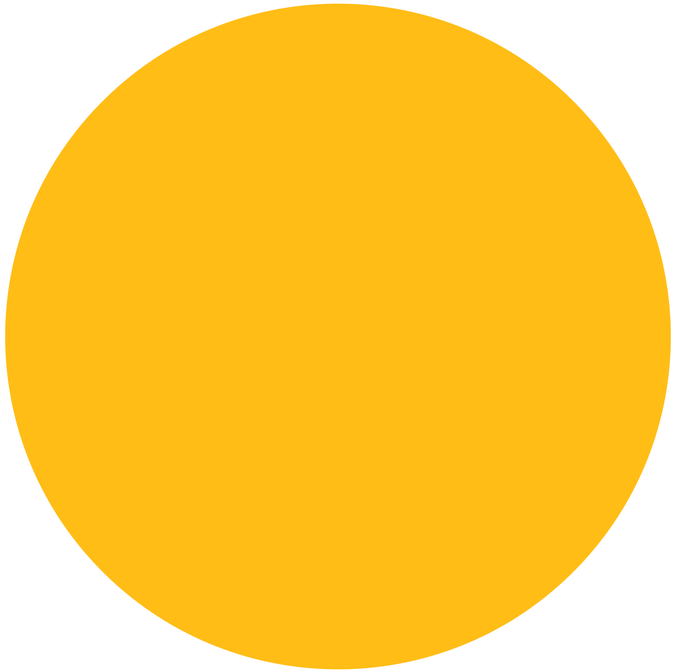
Railsconf
ATLANTA 2023

Railsconf
ATLANTA 2023

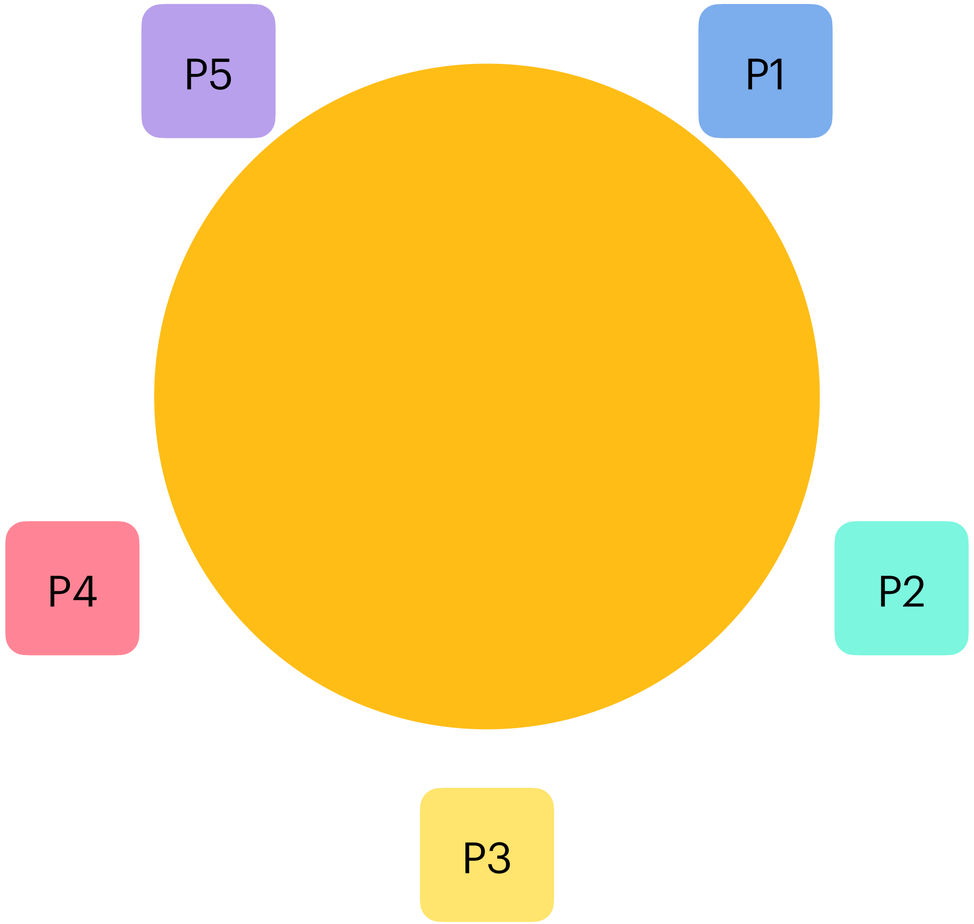
Unanticipated behaviour caused by multiple processes interacting with shared resources in a different order than expected

Dining Philosophers

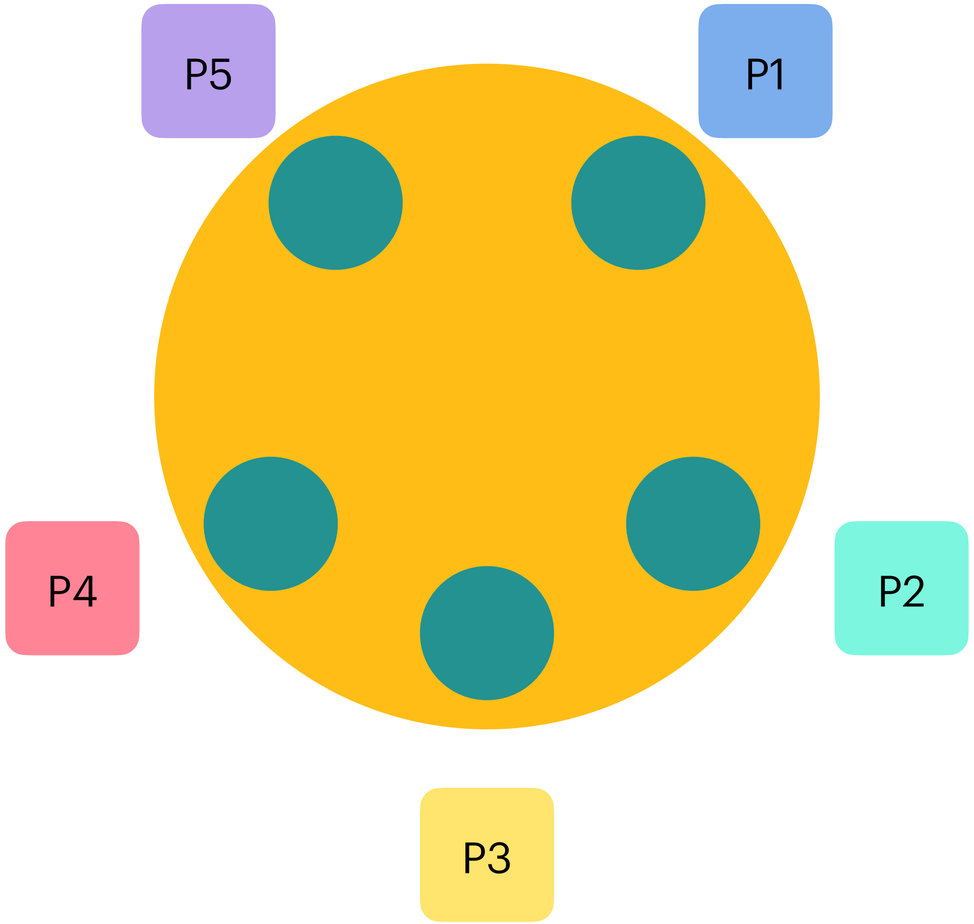
Dining Philosophers



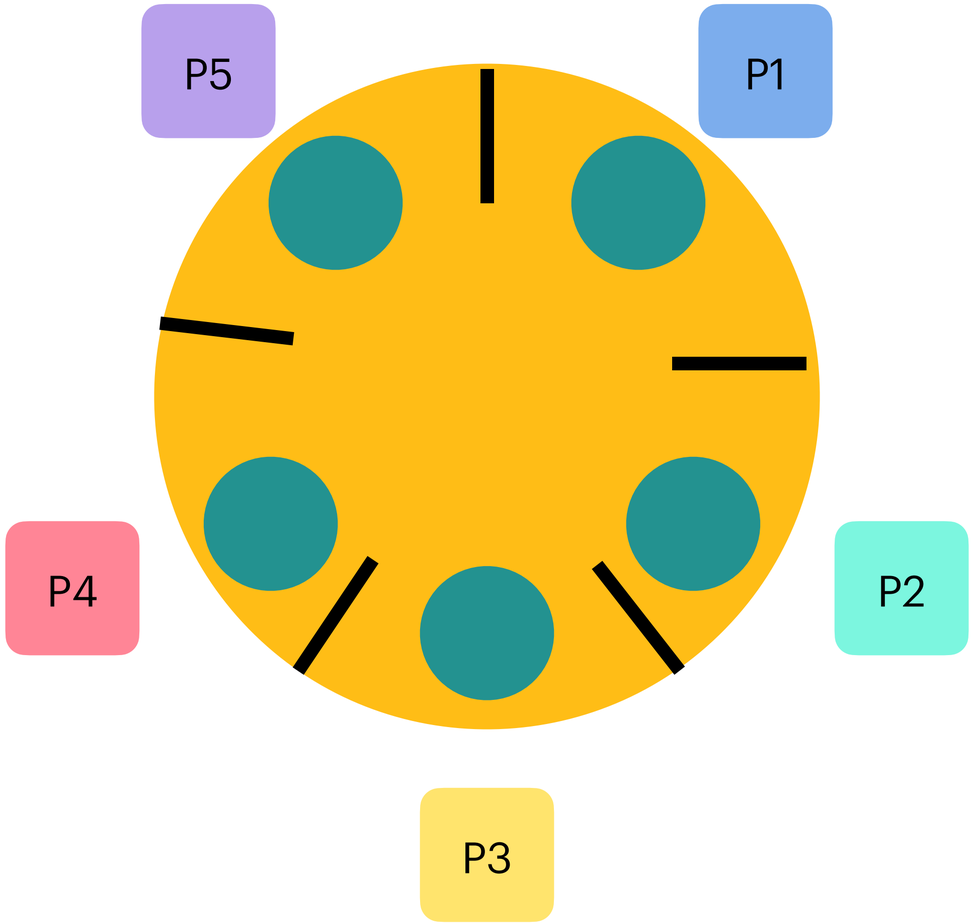
Dining Philosophers



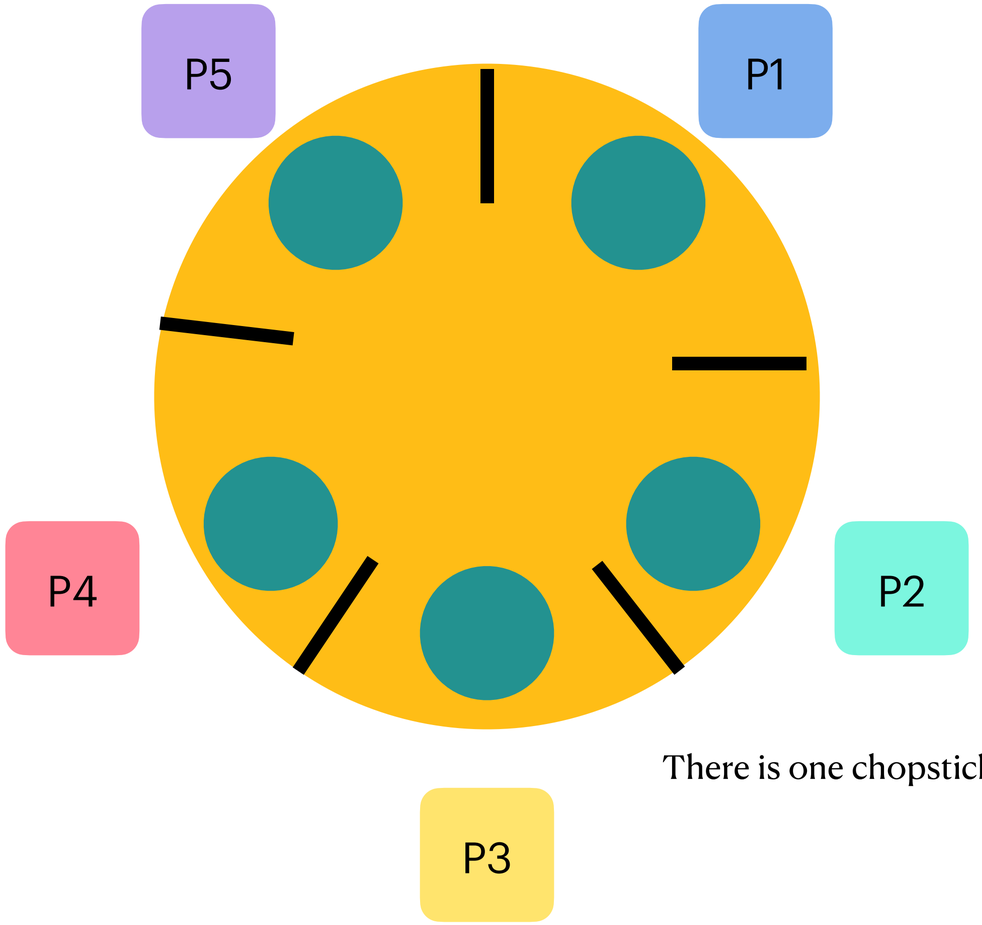
Dining Philosophers



Dining Philosophers

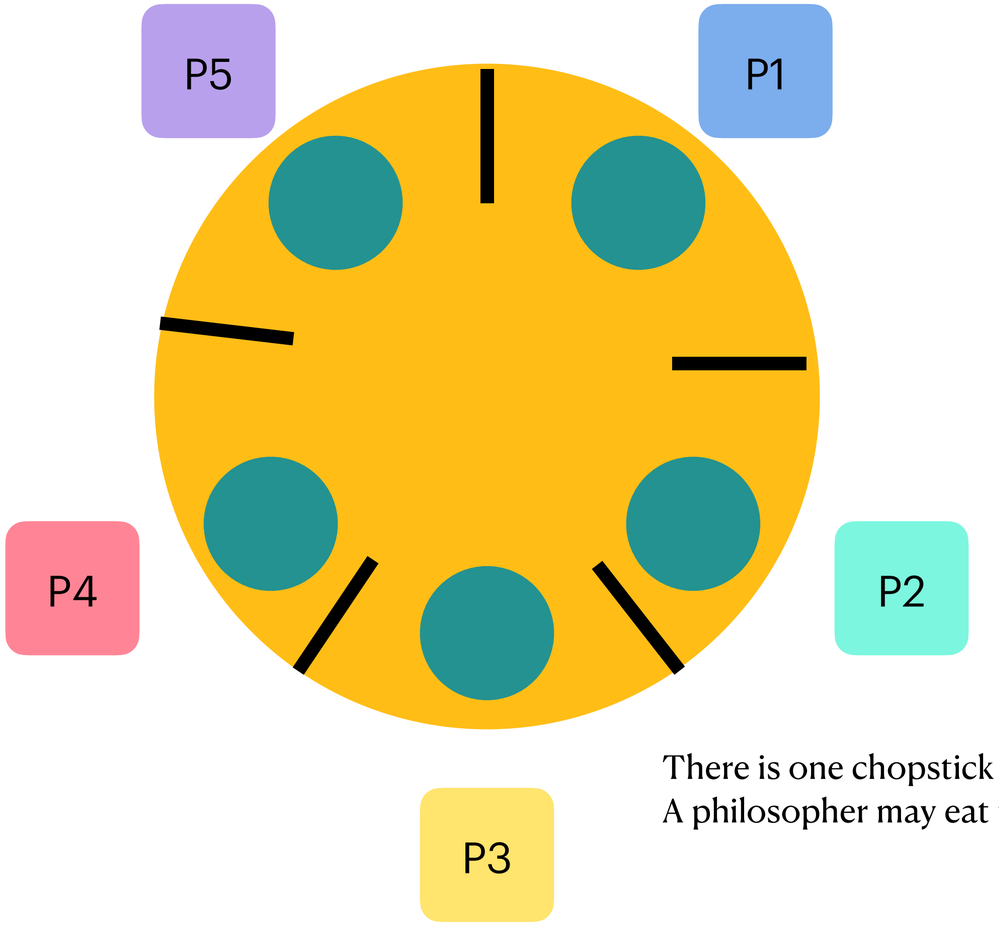


Dining Philosophers



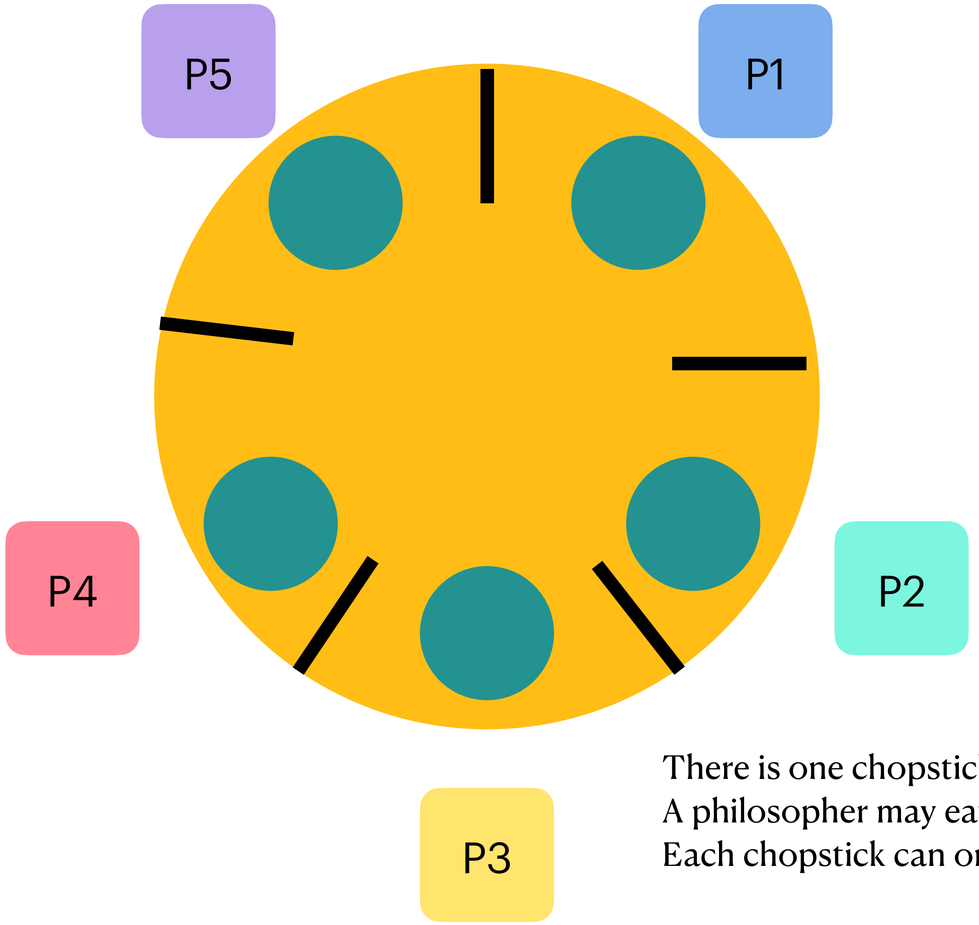
There is one chopstick between each philosopher

Dining Philosophers



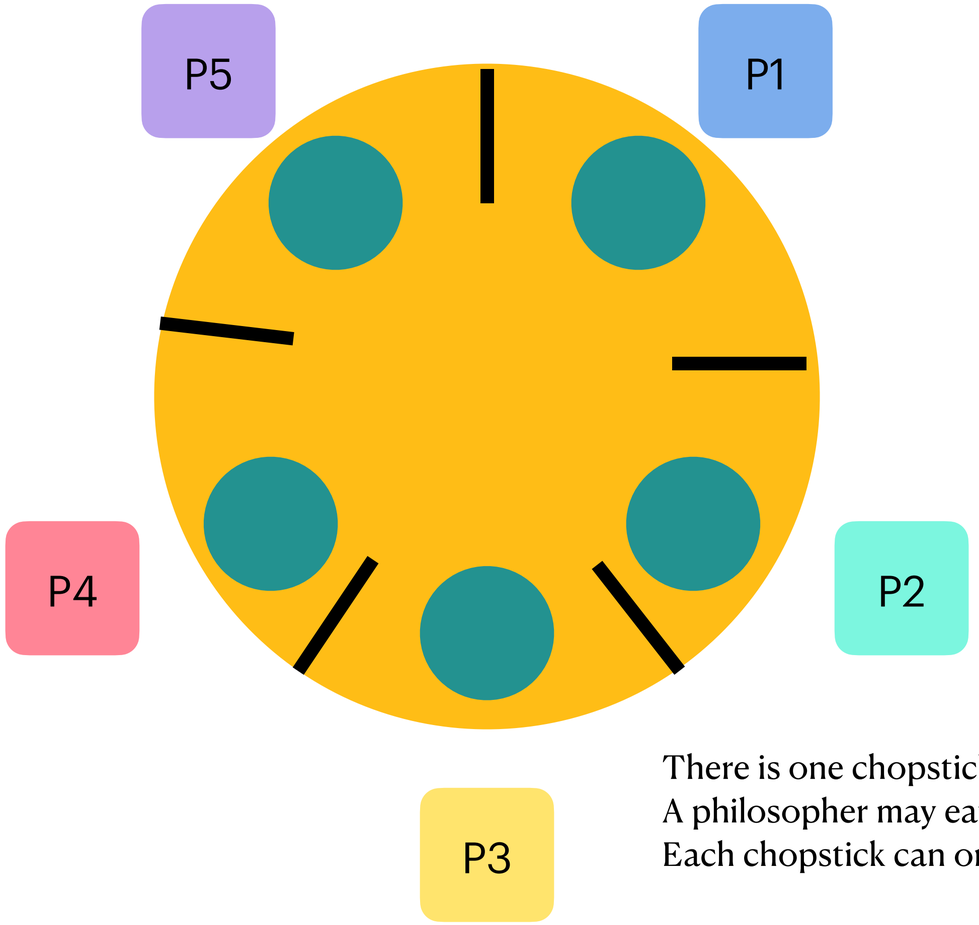
There is one chopstick between each philosopher
A philosopher may eat if they can pick up both adjacent chopsticks

Dining Philosophers



There is one chopstick between each philosopher
A philosopher may eat if they can pick up both adjacent chopsticks
Each chopstick can only be picked up by one philosopher

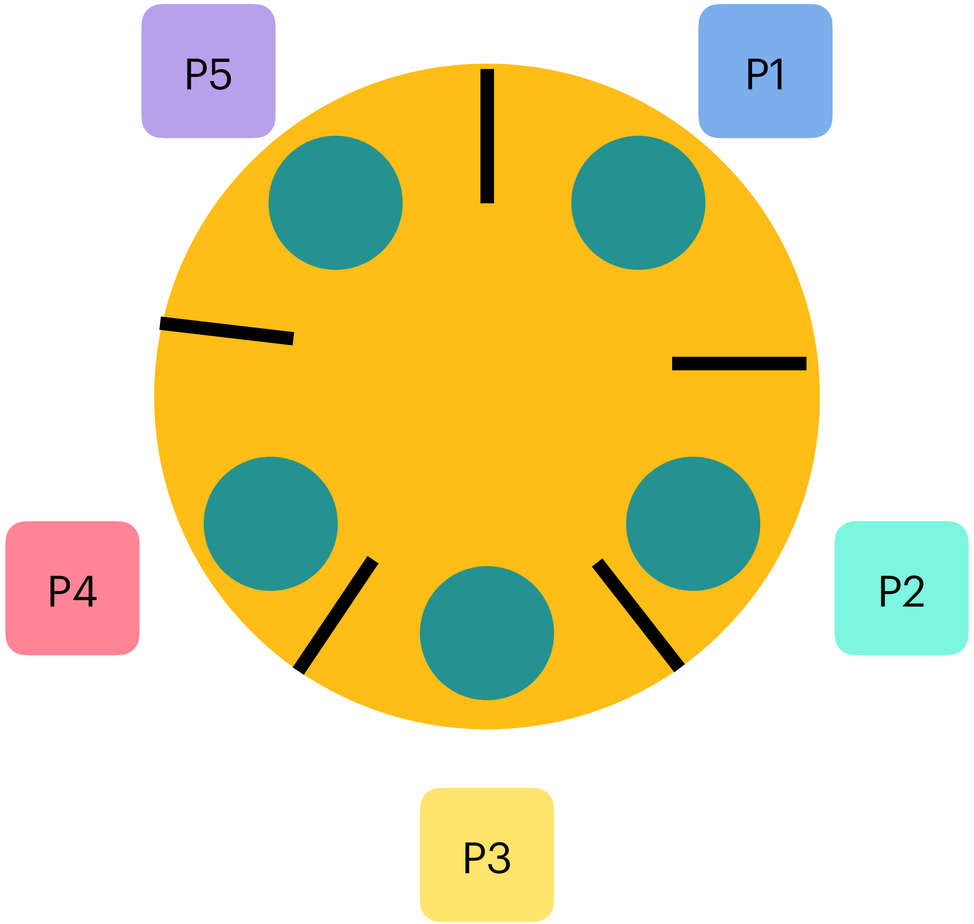
Dining Philosophers



There is one chopstick between each philosopher
A philosopher may eat if they can pick up both adjacent chopsticks
Each chopstick can only be picked up by one philosopher

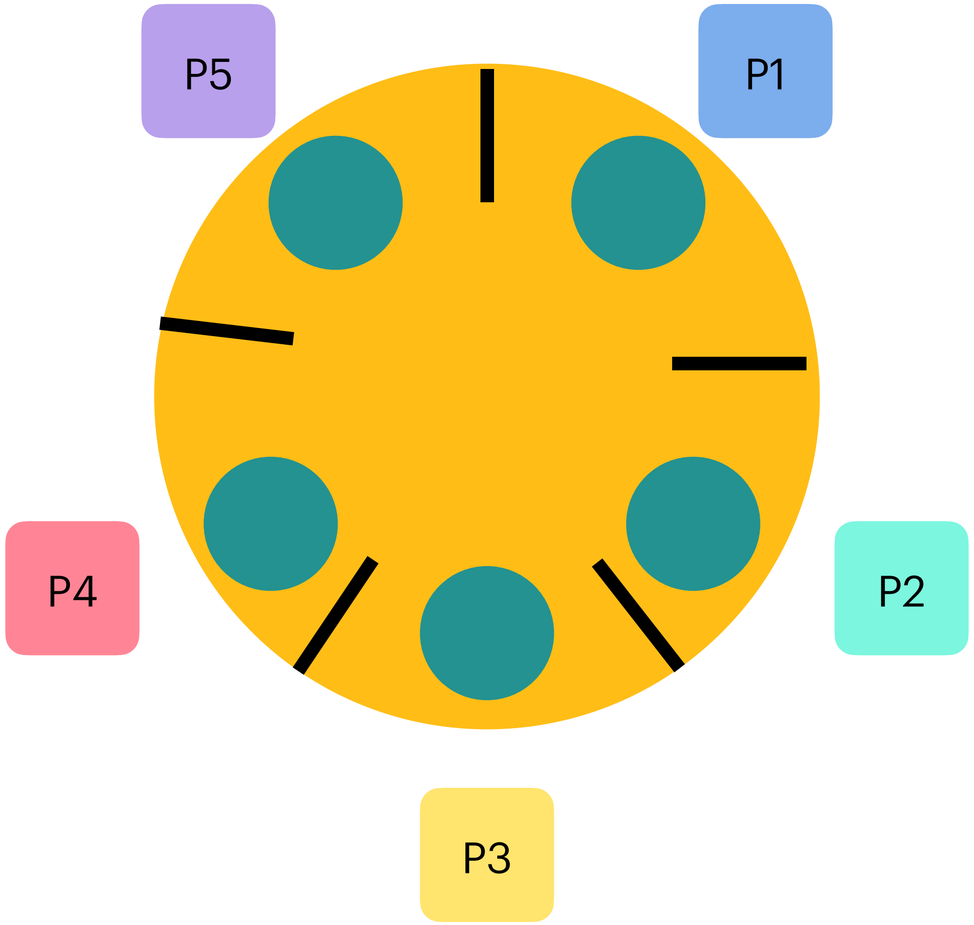
What is a process all philosophers can agree to such that no one will starve?

Dining Philosophers



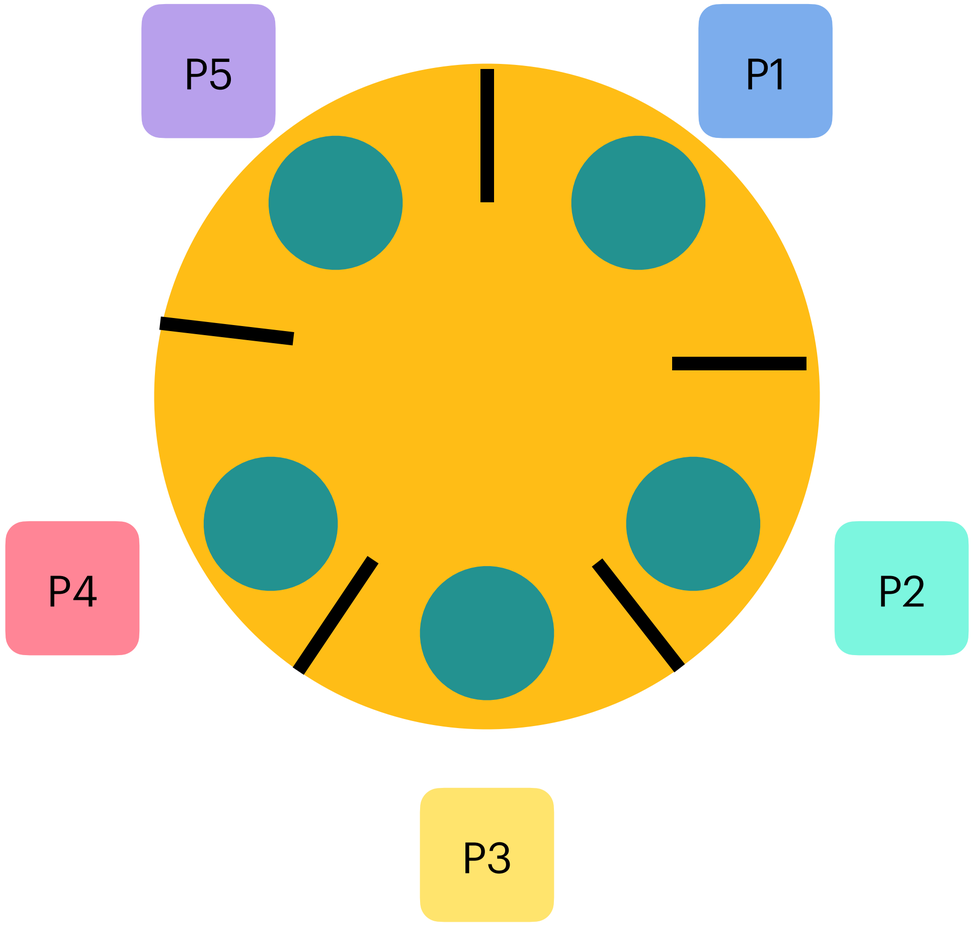
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



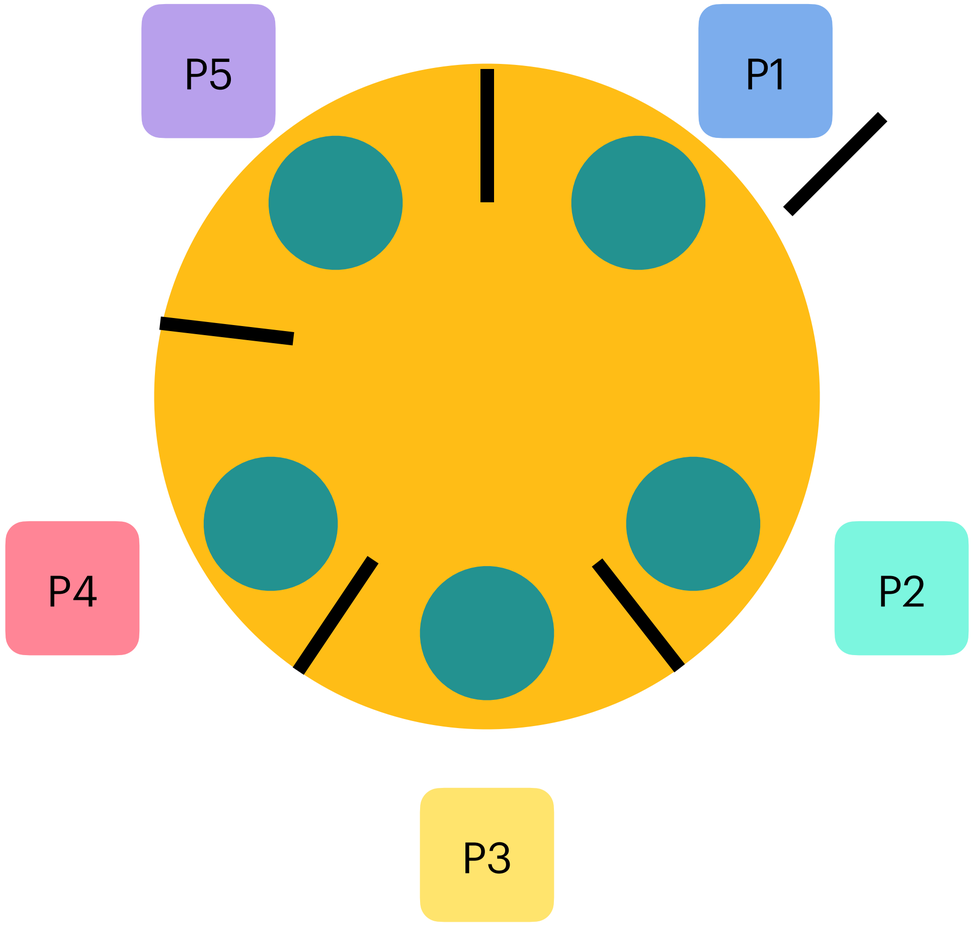
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



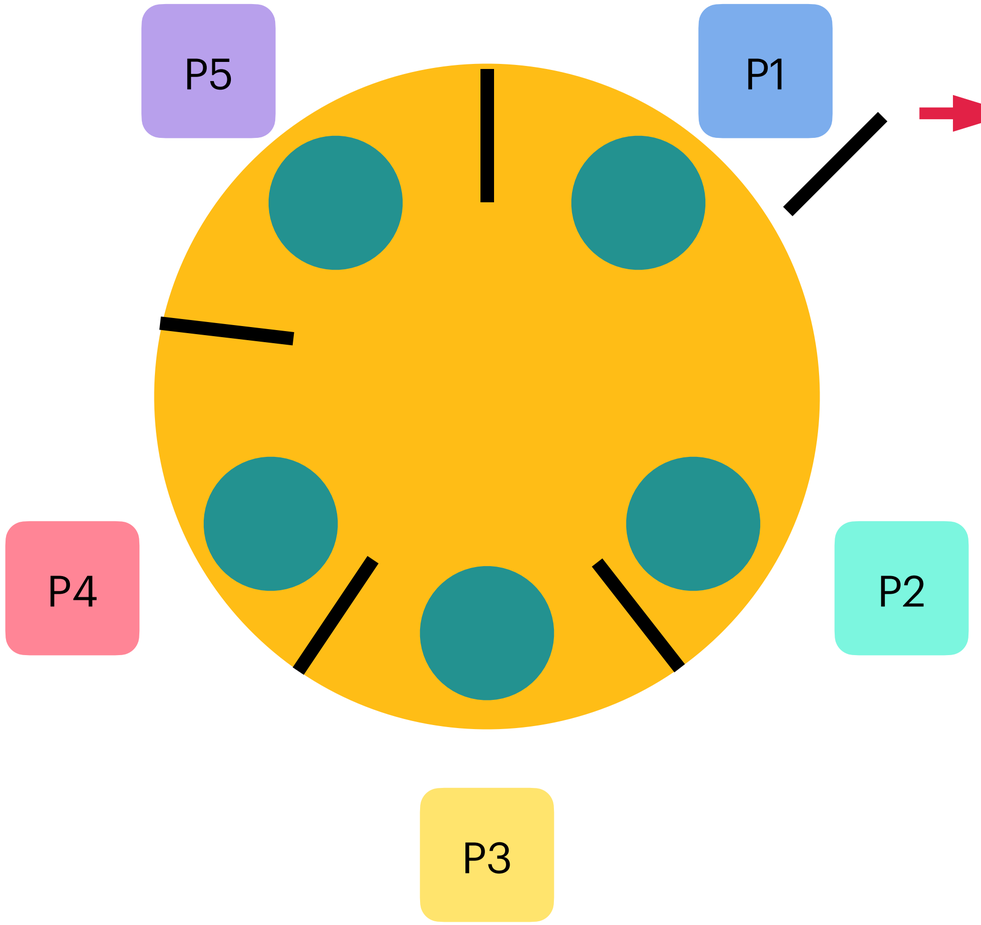
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



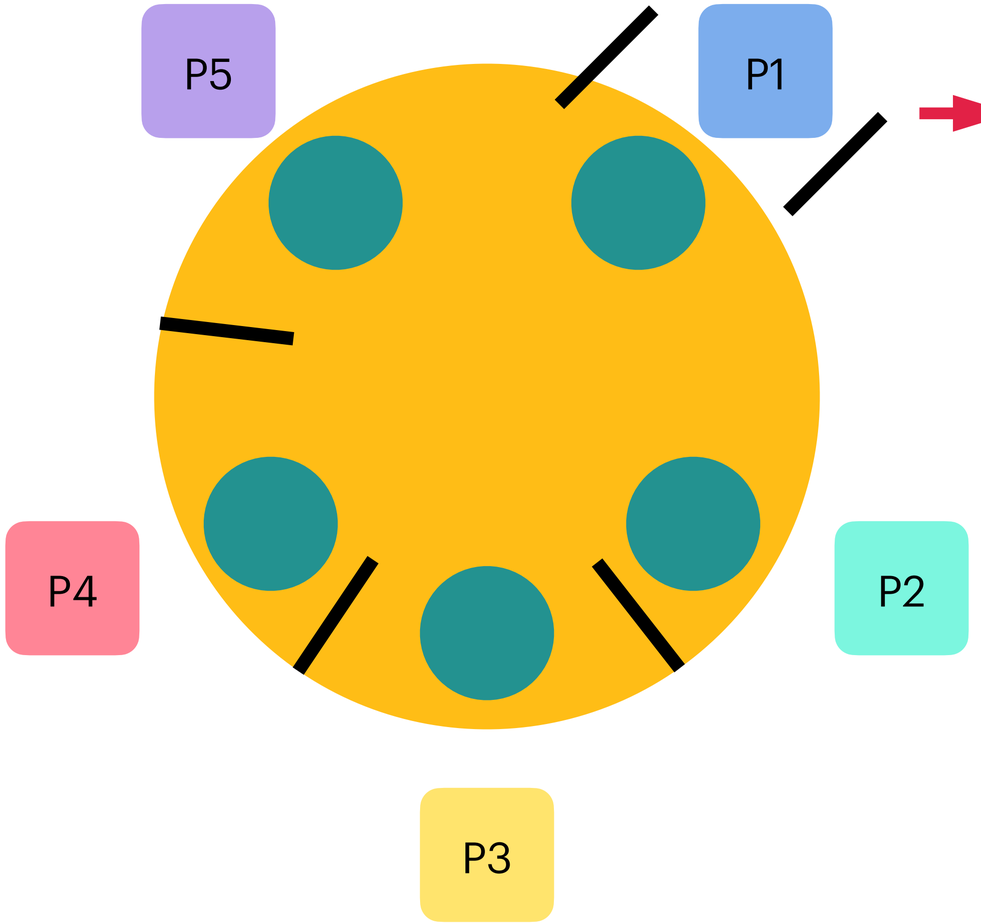
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



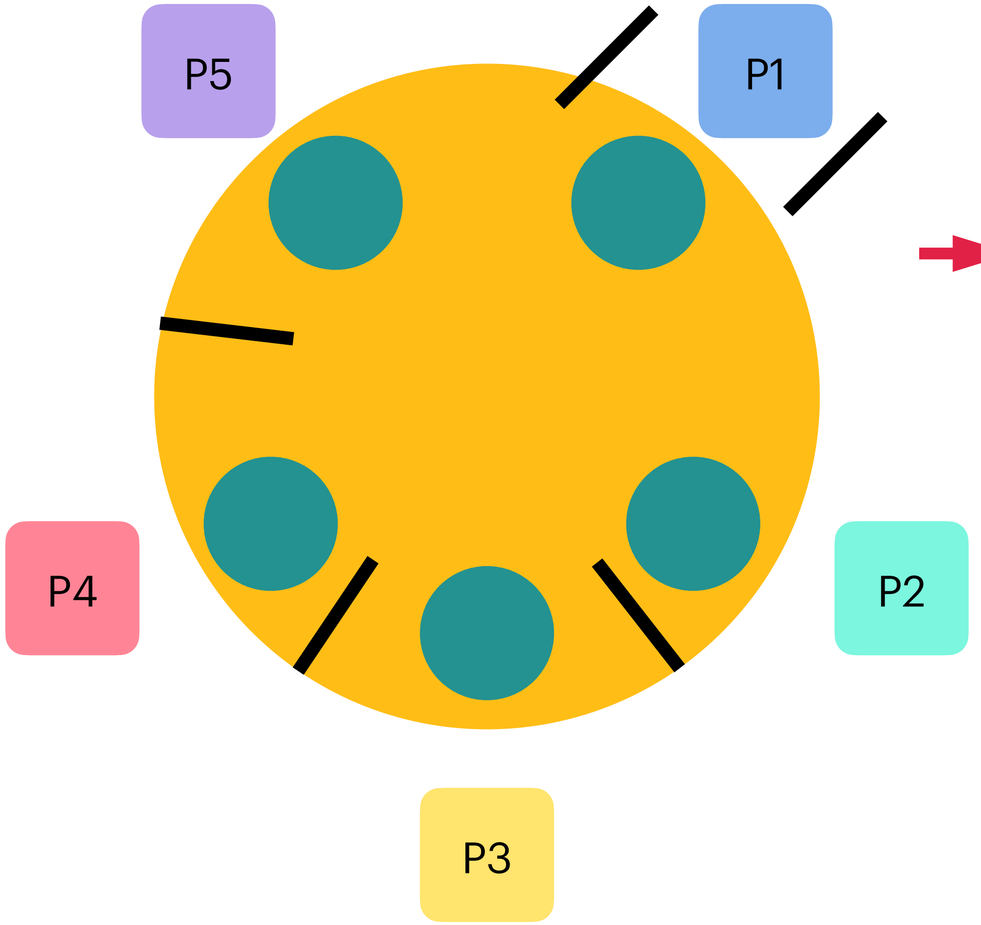
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



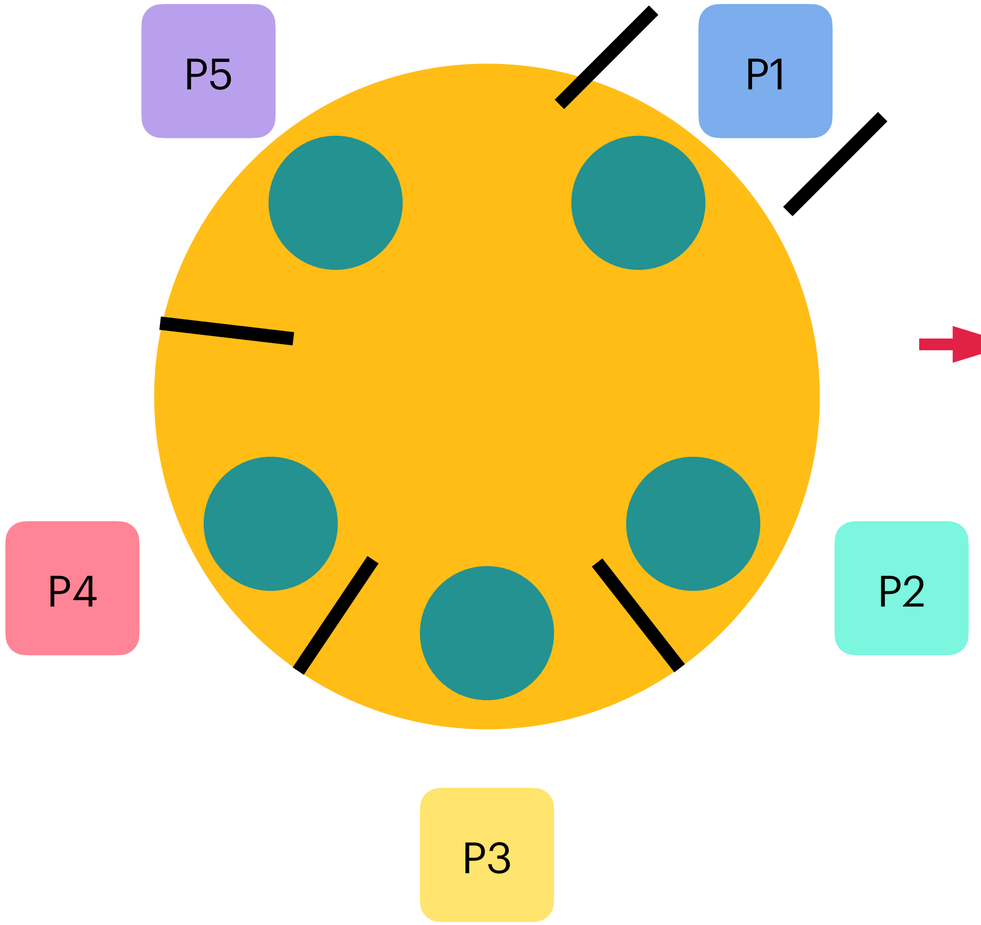
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



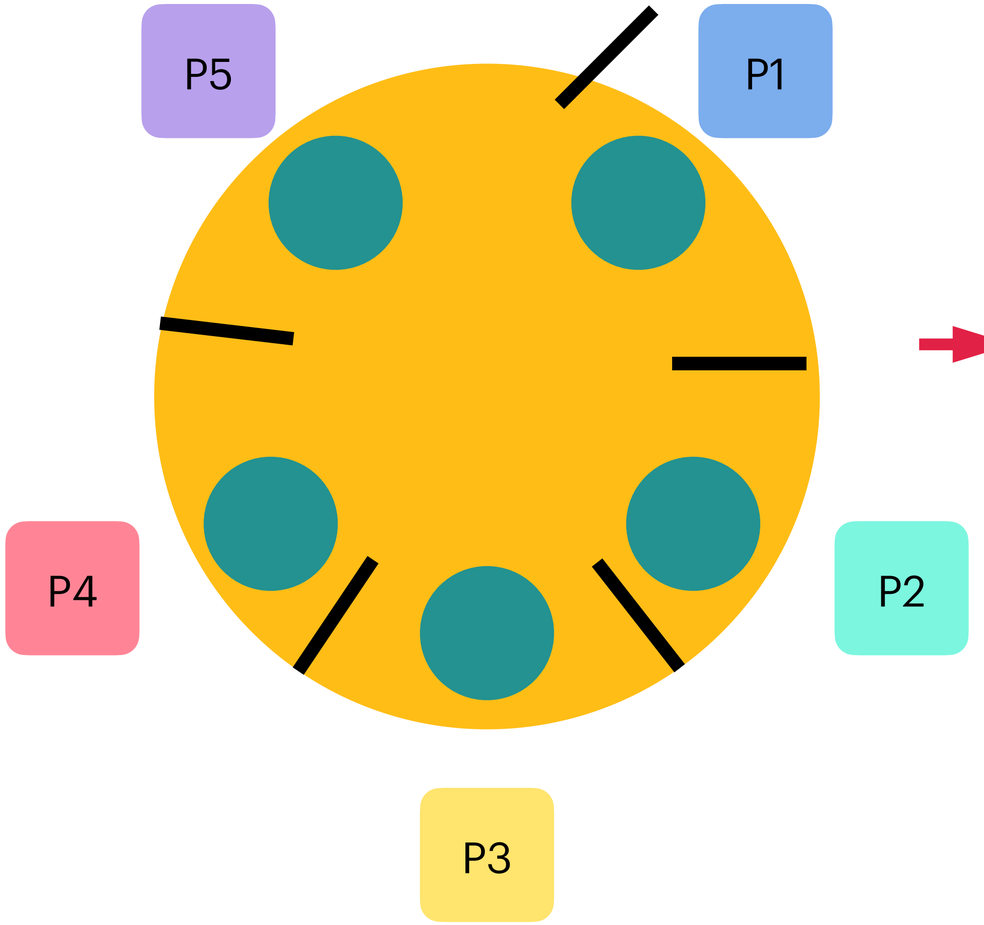
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



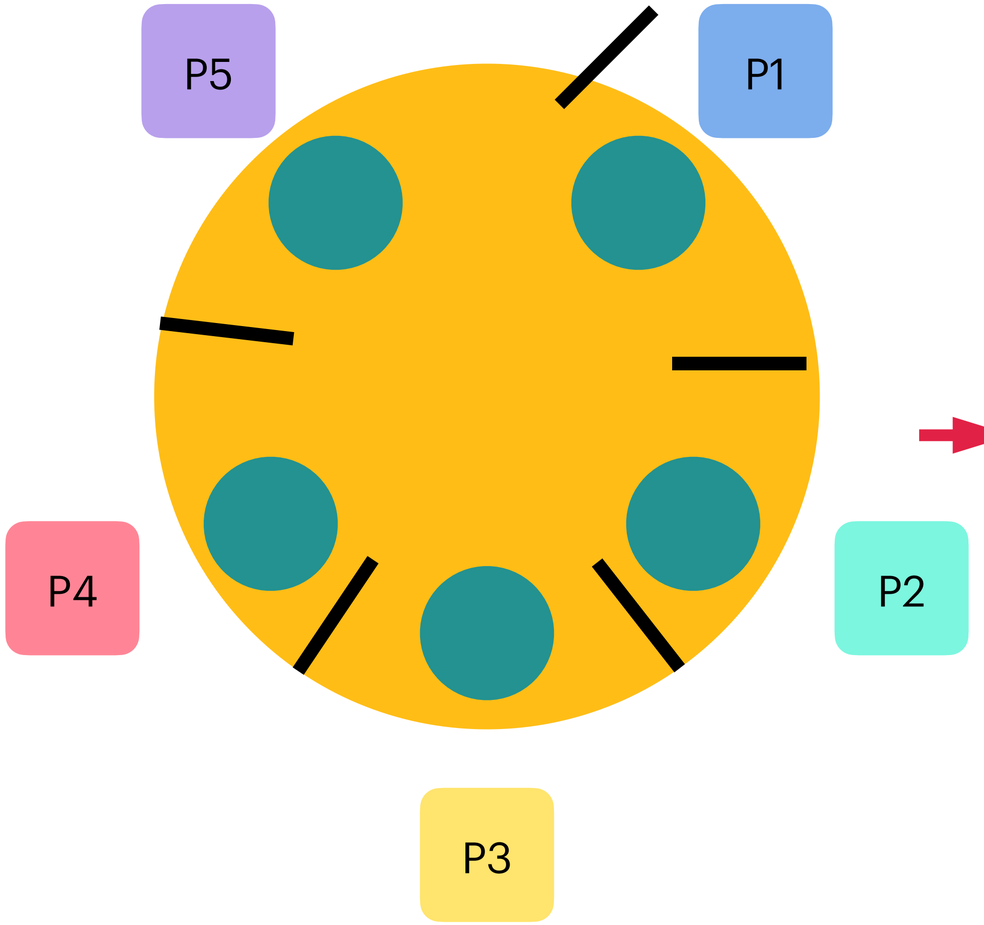
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



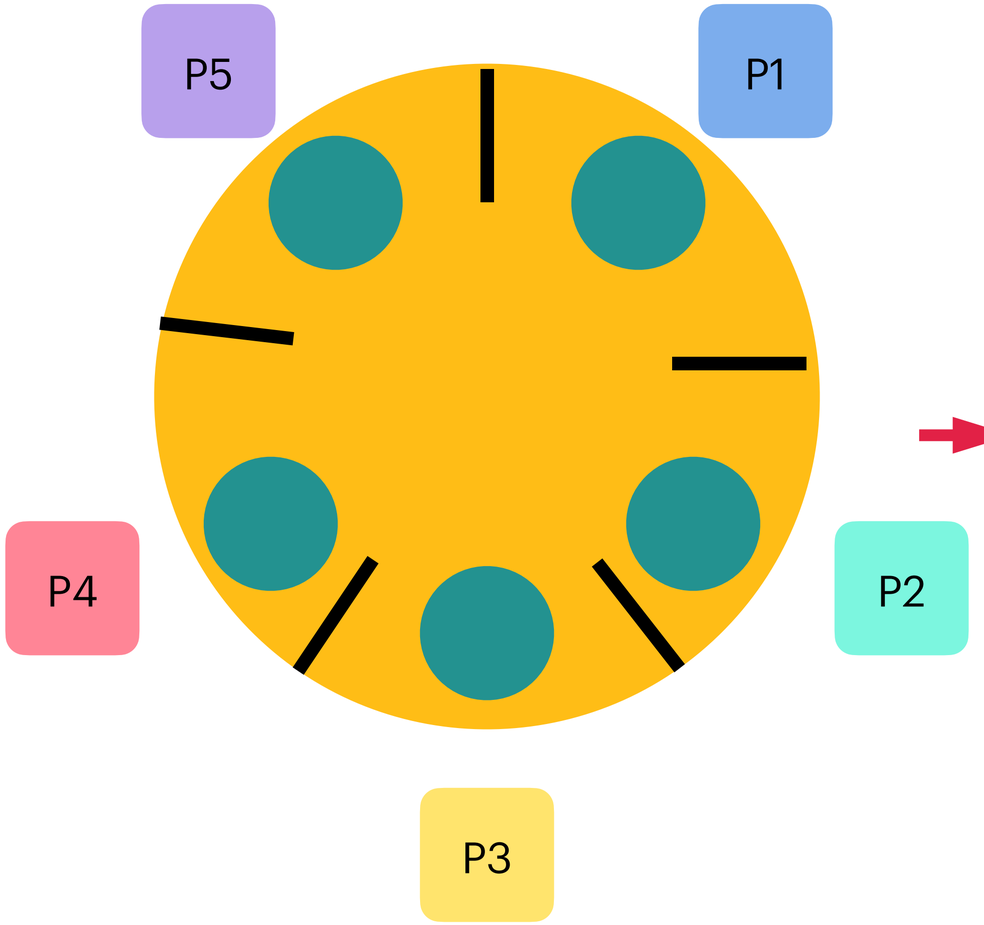
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



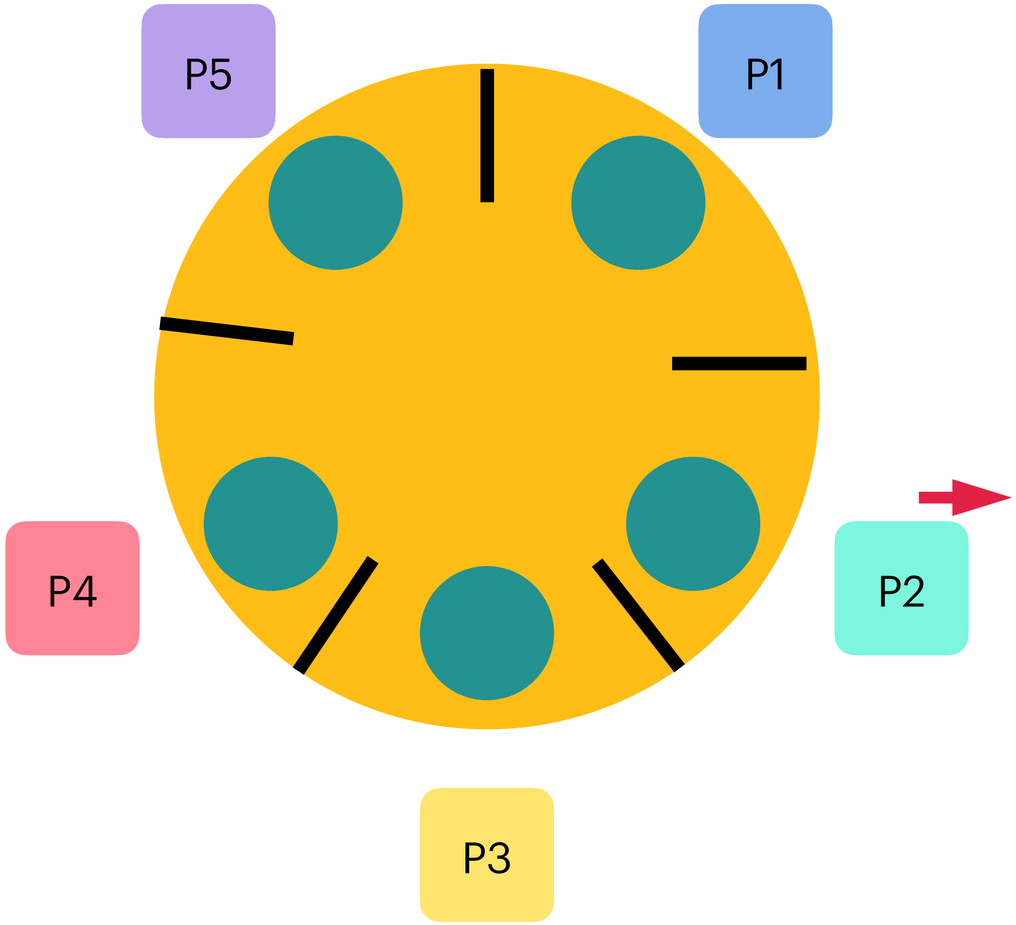
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



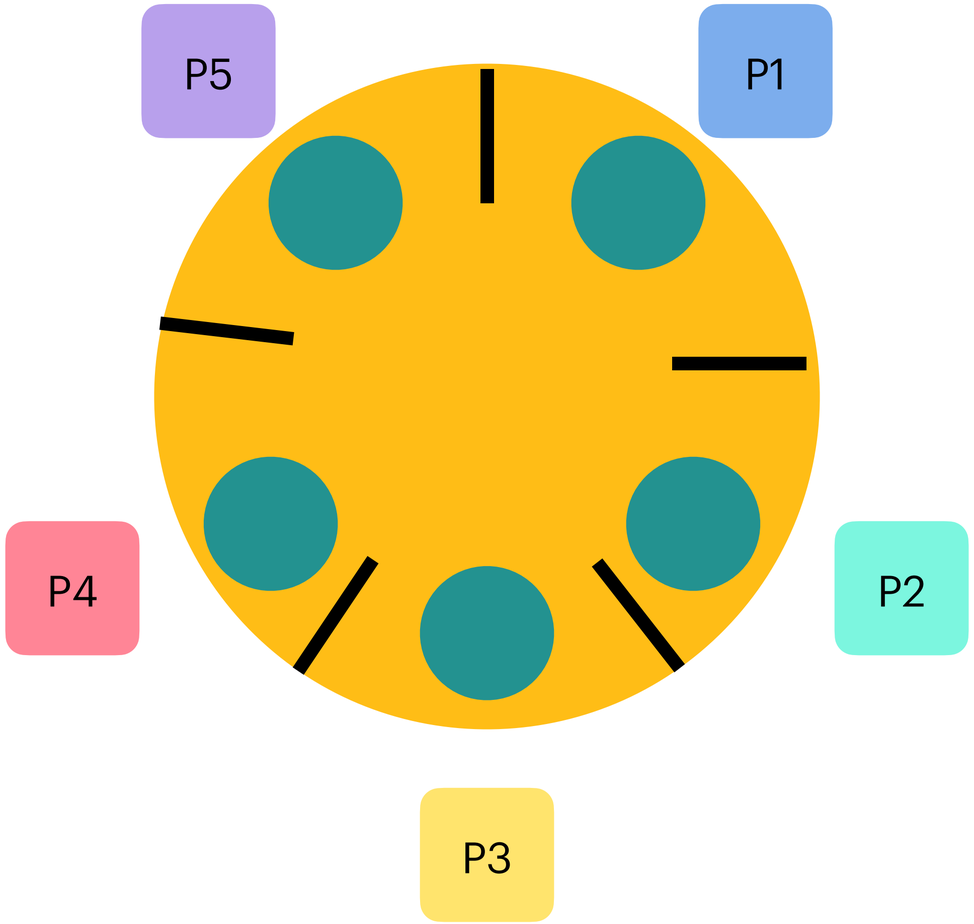
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



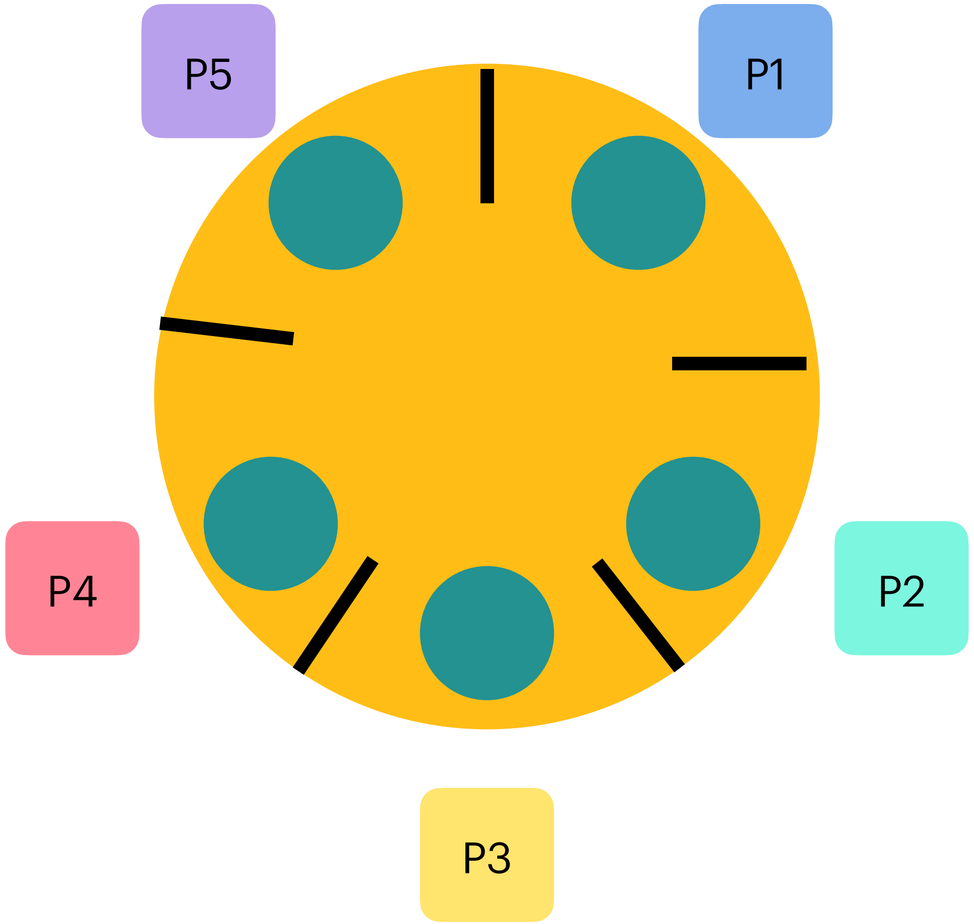
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



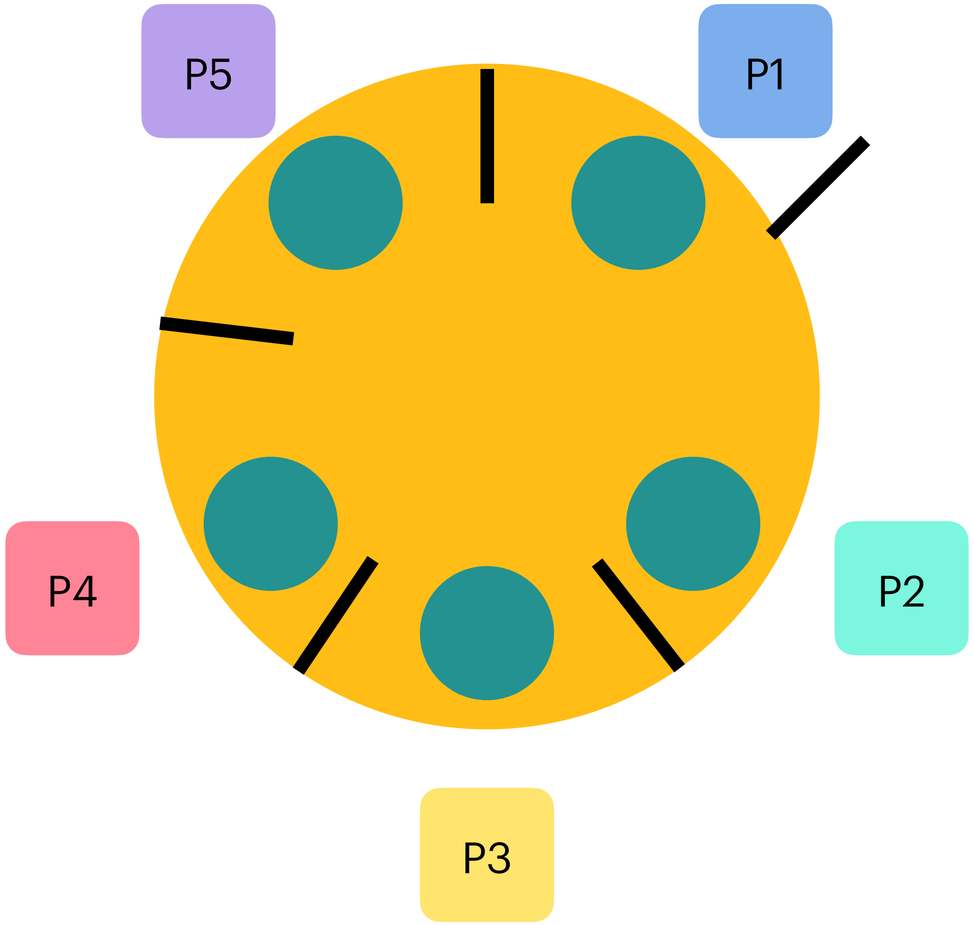
Dining Philosophers

1. Pick up the left chopstick when it is available.



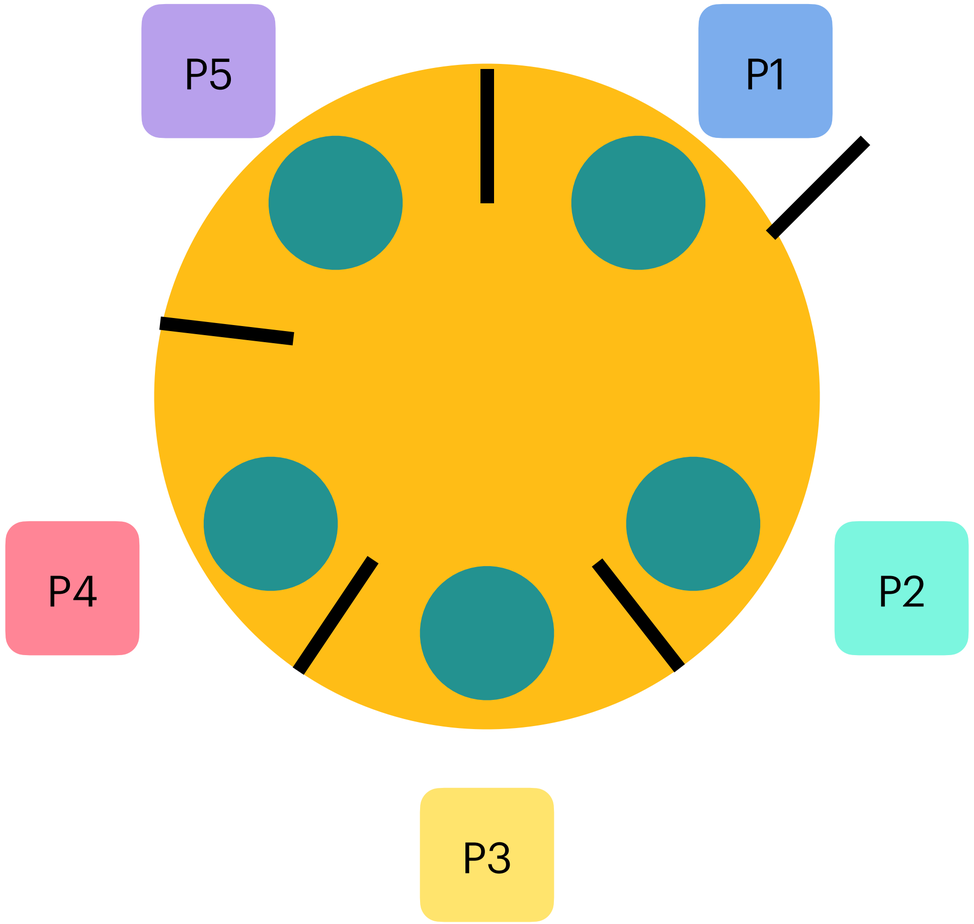
Dining Philosophers

1. Pick up the left chopstick when it is available.



Dining Philosophers

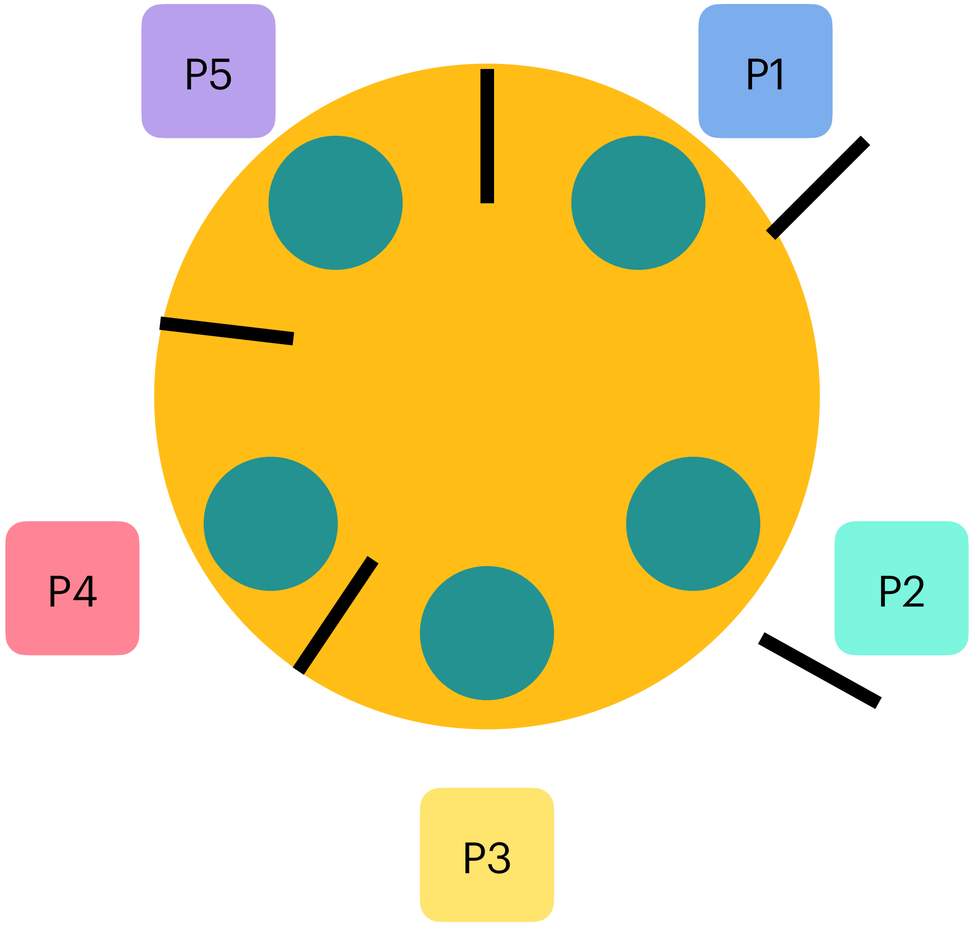
1. Pick up the left chopstick when it is available.



1. Pick up the left chopstick when it is available.

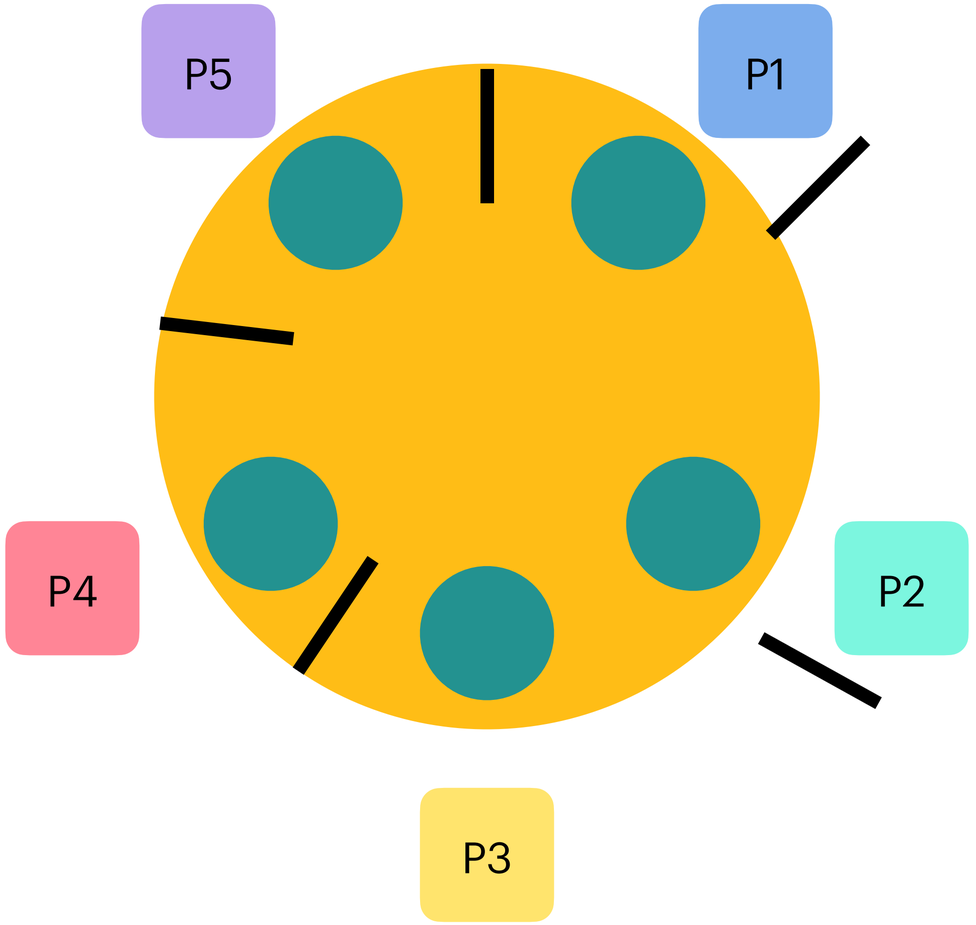
Dining Philosophers

1. Pick up the left chopstick when it is available.



1. Pick up the left chopstick when it is available.

Dining Philosophers



1. Pick up the left chopstick when it is available.

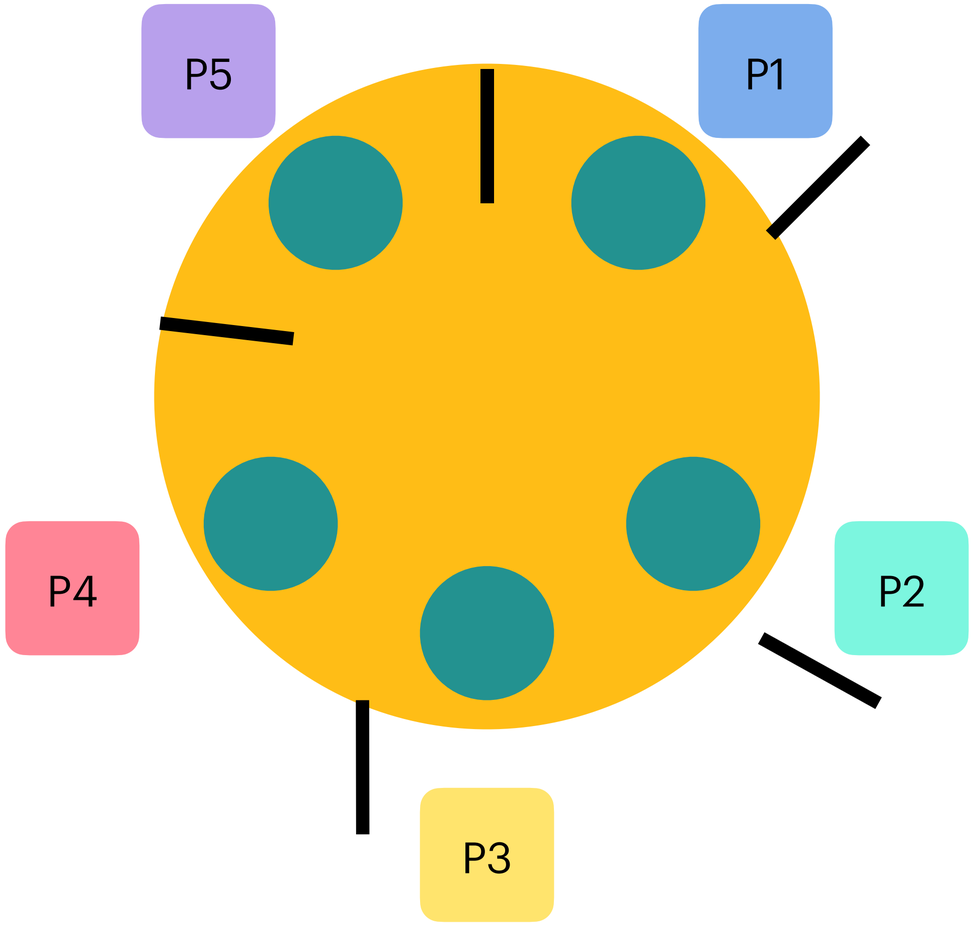
1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

Dining Philosophers



1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

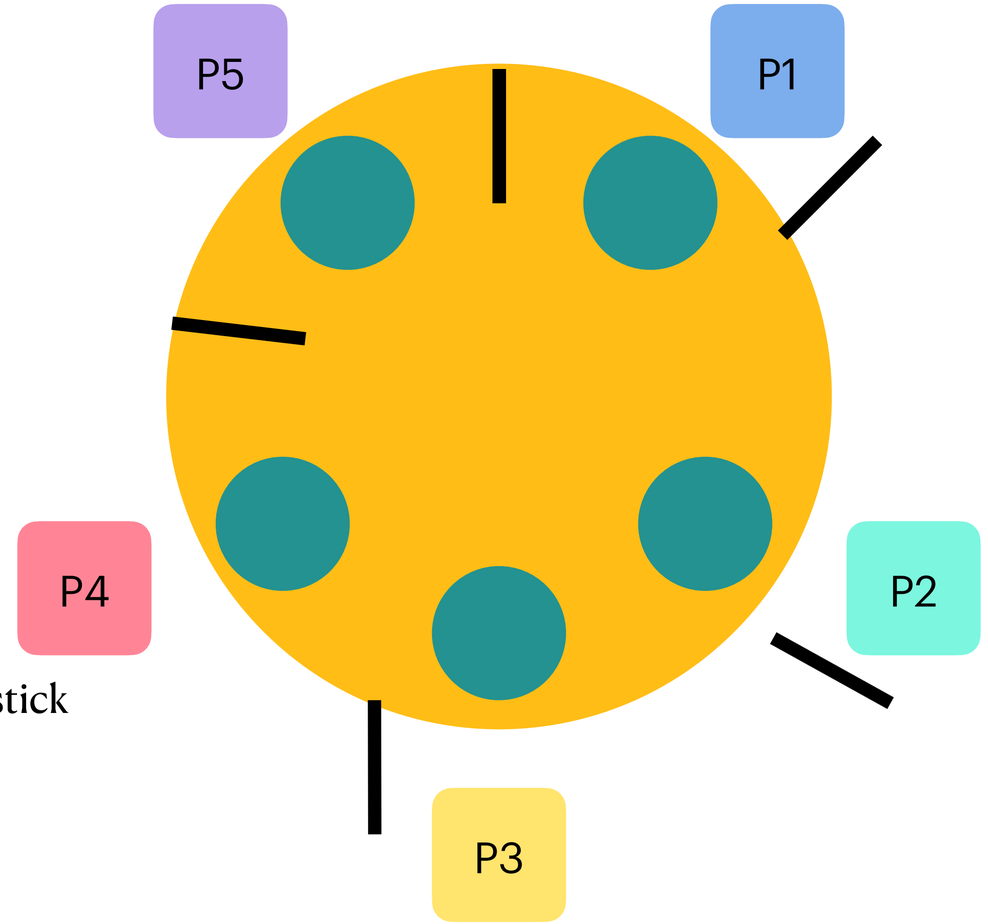
1. Pick up the left chopstick when it is available.

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

Dining Philosophers

1. Pick up the left chopstick when it is available.



1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

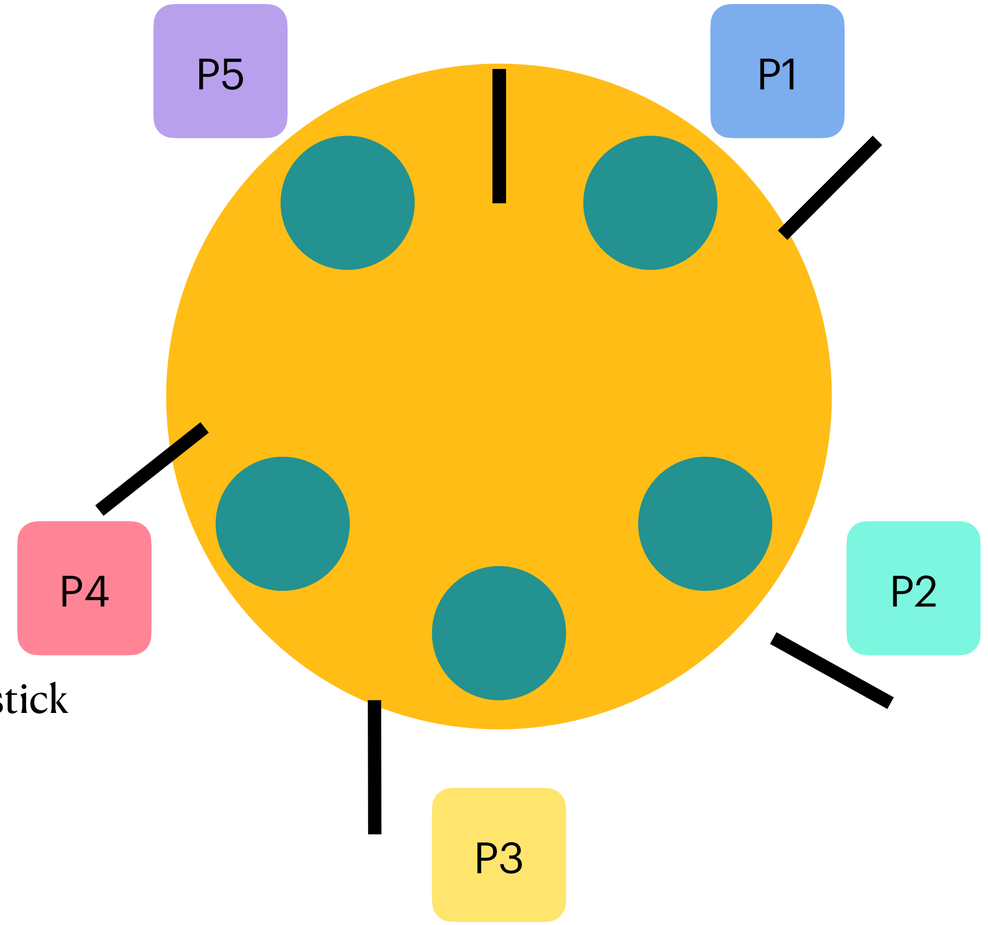
1. Pick up the left chopstick when it is available.

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

Dining Philosophers

1. Pick up the left chopstick when it is available.



1. Pick up the left chopstick when it is available.

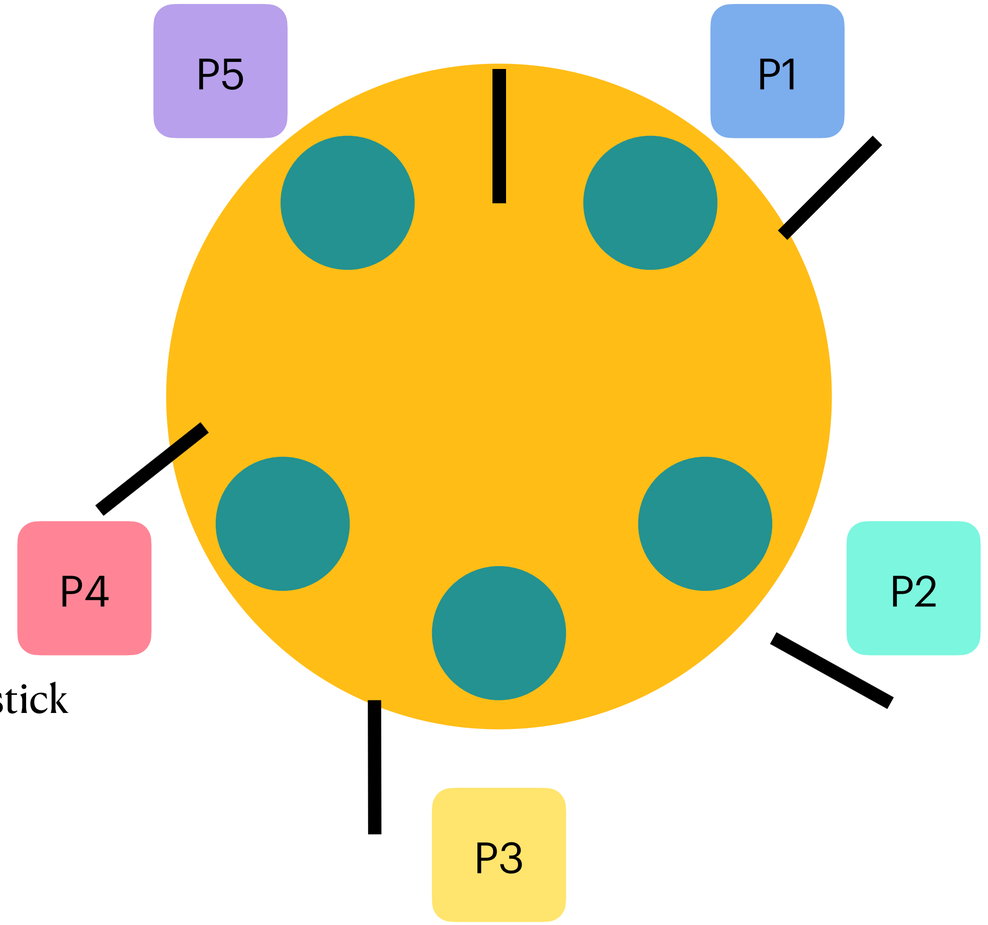
1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

Dining Philosophers

1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.



1. Pick up the left chopstick when it is available.

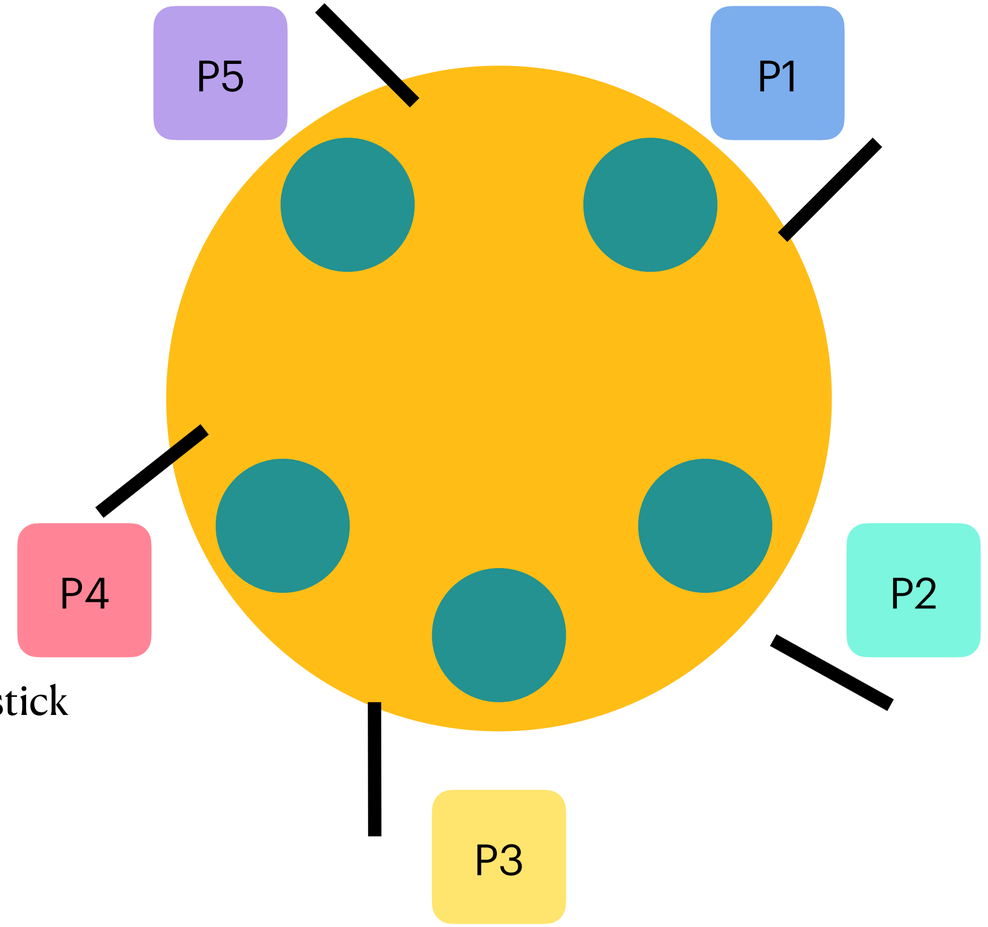
1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

Dining Philosophers

1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.



1. Pick up the left chopstick when it is available.

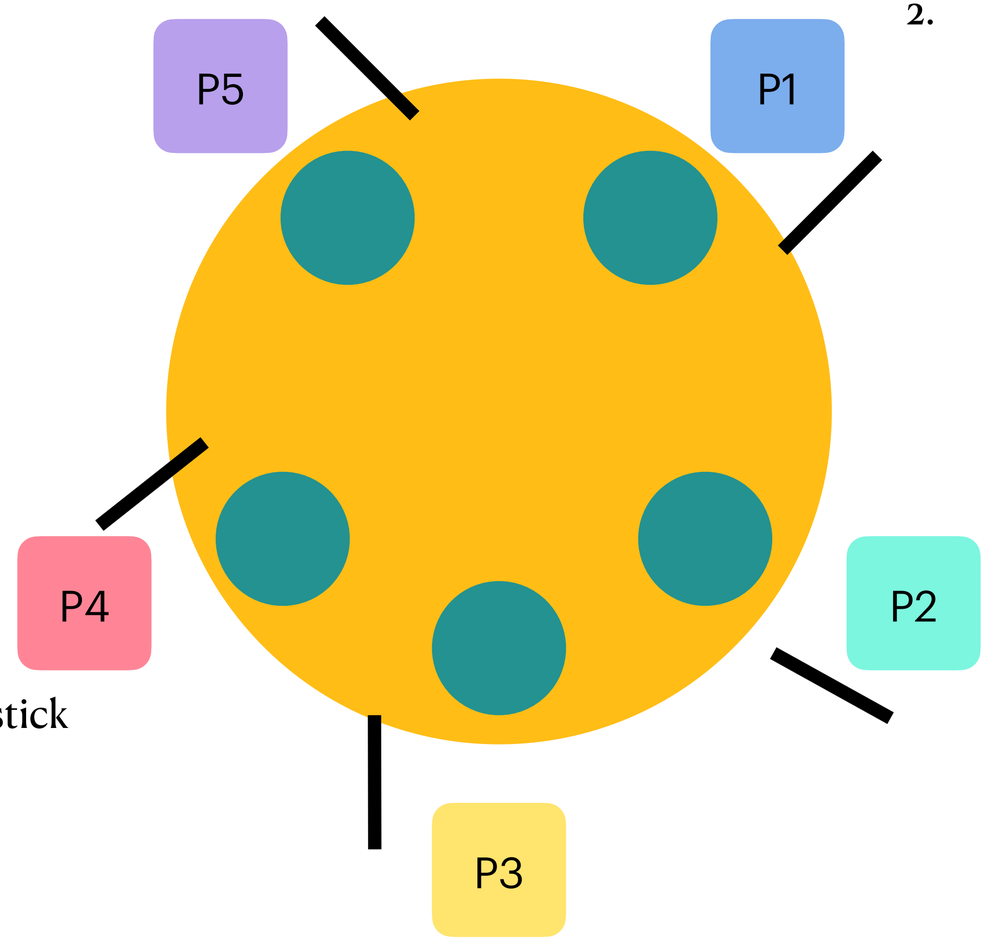
1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

Dining Philosophers

1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available



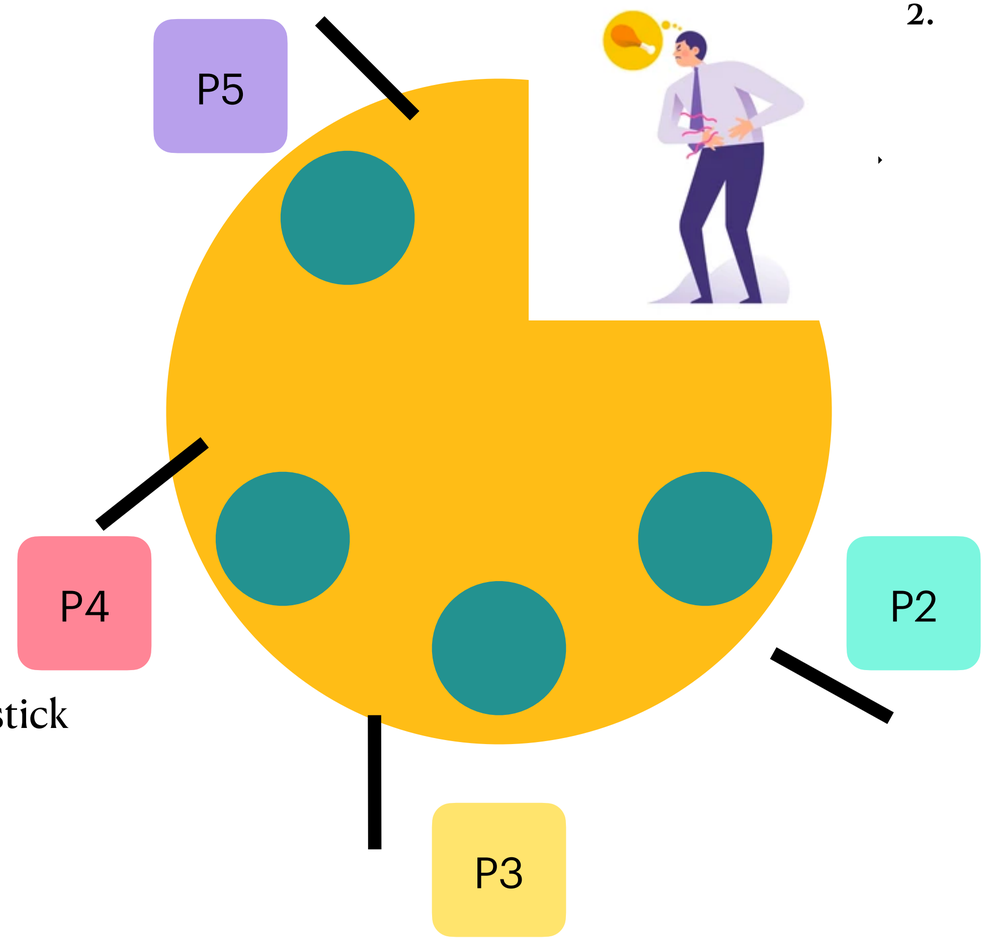
1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

Dining Philosophers

1. Pick up the left chopstick when it is available.



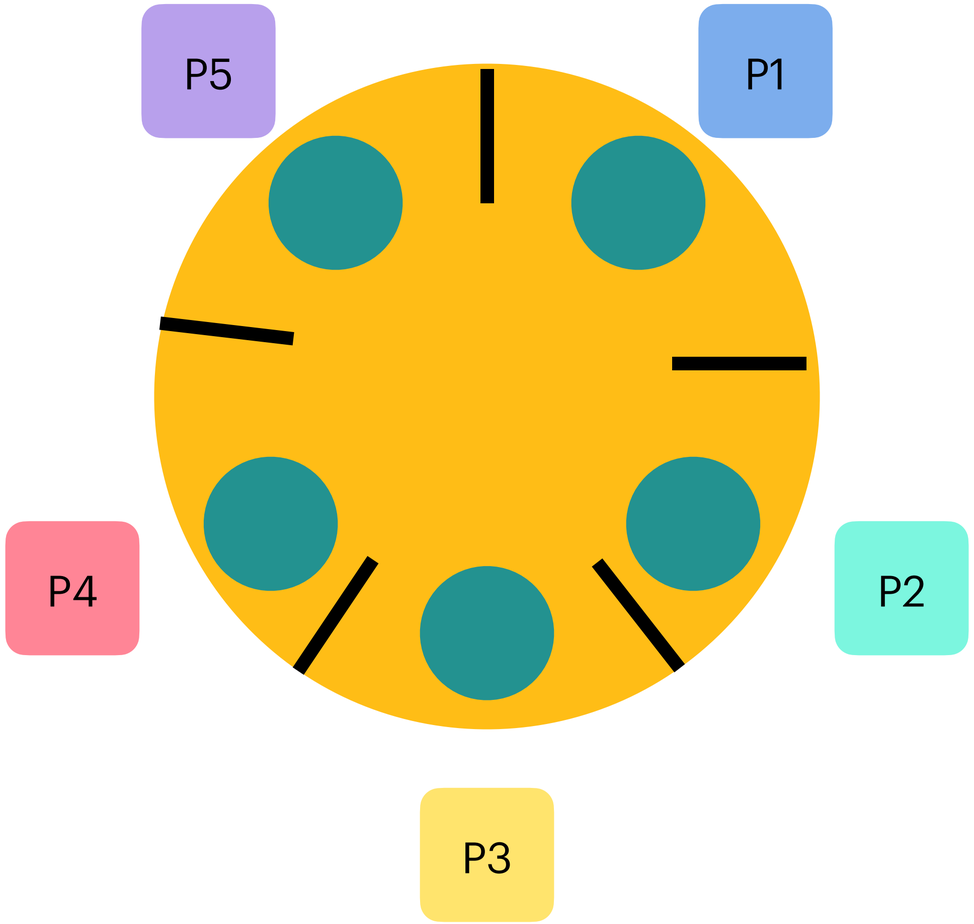
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available

1. Pick up the left chopstick when it is available.

1. Pick up the left chopstick when it is available.

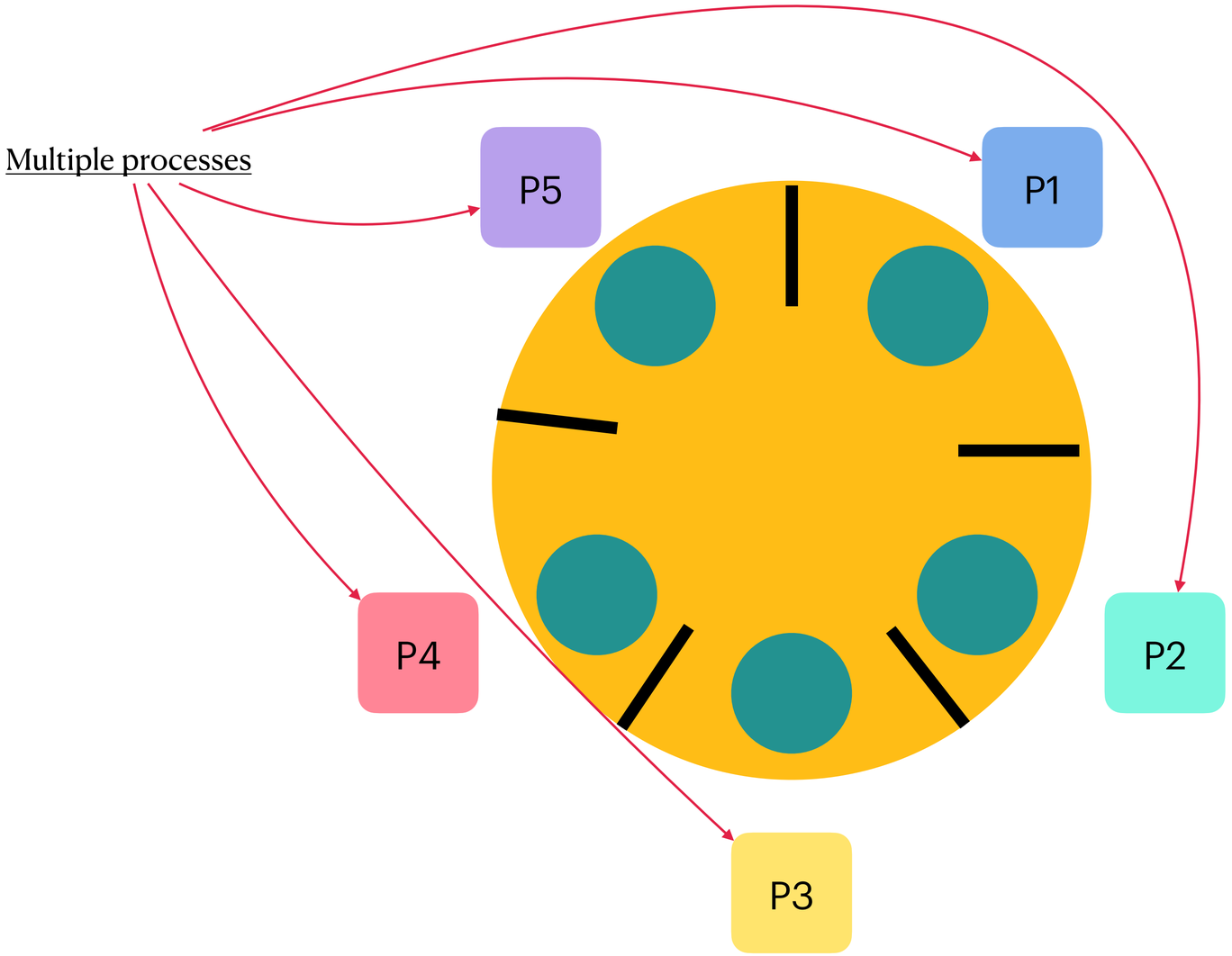
1. Pick up the left chopstick when it is available.

Dining Philosophers



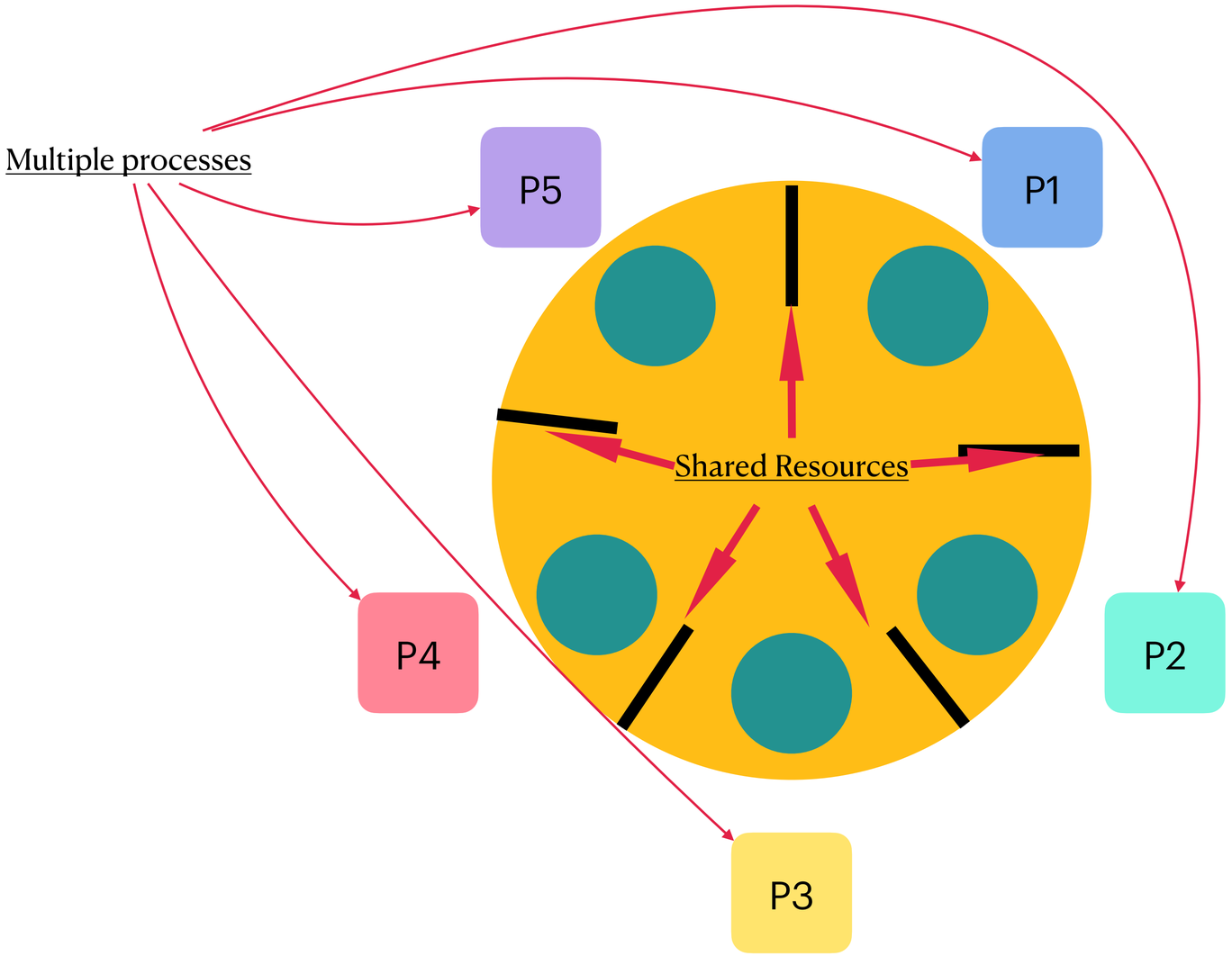
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



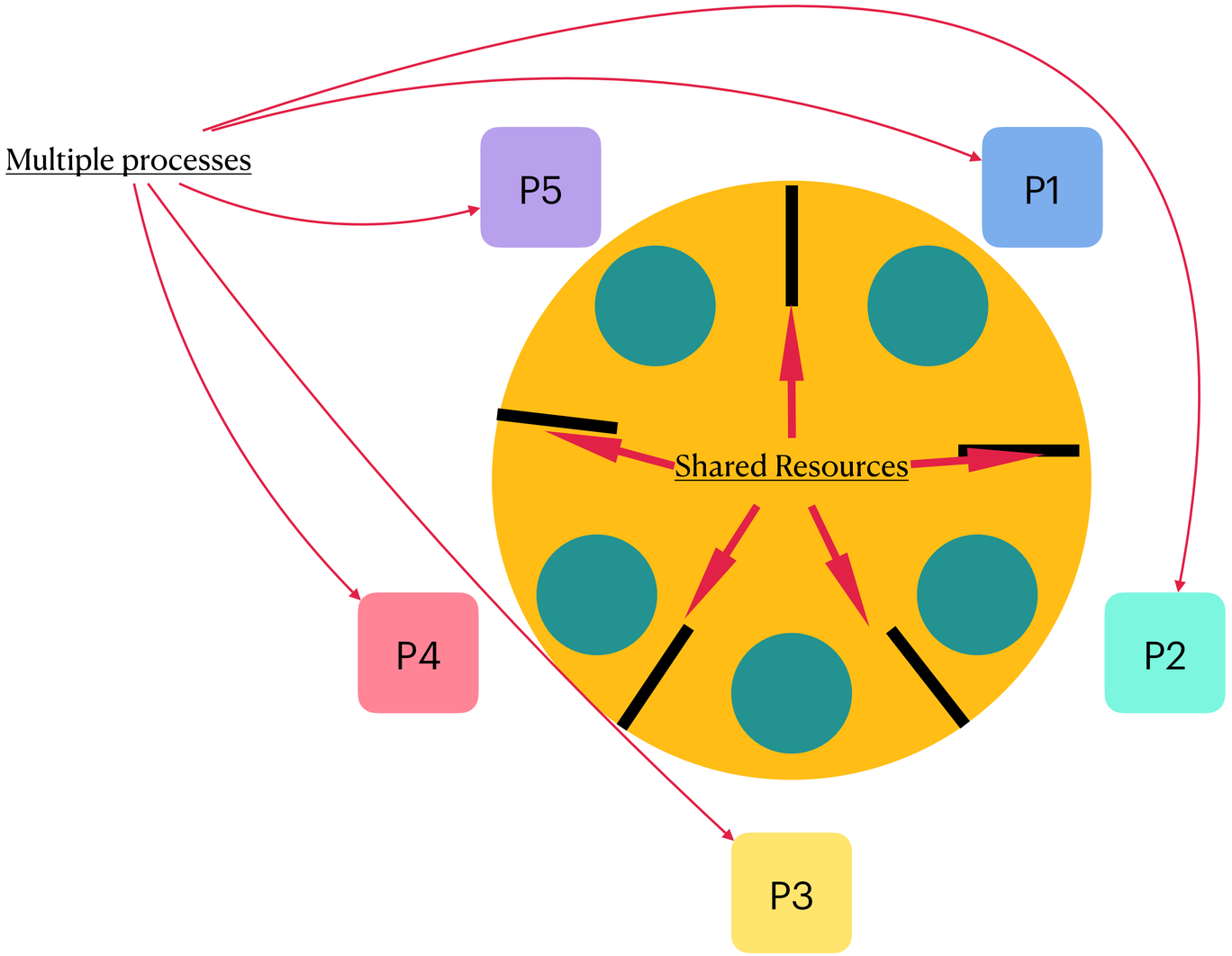
1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

Dining Philosophers



Critical section

1. Pick up the left chopstick when it is available.
2. Pick up right chopstick when it is available
3. Eat for some amount of time
4. Put left chopstick down
5. Put right chopstick down
6. Repeat

RailsConf
ATLANTA 2023

Railsconf
ATLANTA 2023

Railsconf
ATLANTA 2023

Multiple processes

Railsconf
ATLANTA 2023

Multiple processes

Requests

Multiple processes

Requests

Background jobs

Railsconf
ATLANTA 2023

Multiple processes

Requests

Background jobs

Rake tasks

Multiple processes

Shared Resources

Requests

Background jobs

Rake tasks

Multiple processes

Shared Resources

Requests

Background jobs

Rake tasks

Multiple processes

Requests

Background jobs

Rake tasks

Shared Resources

Multiple processes

Requests

Background jobs

Rake tasks

Shared Resources

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 2

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

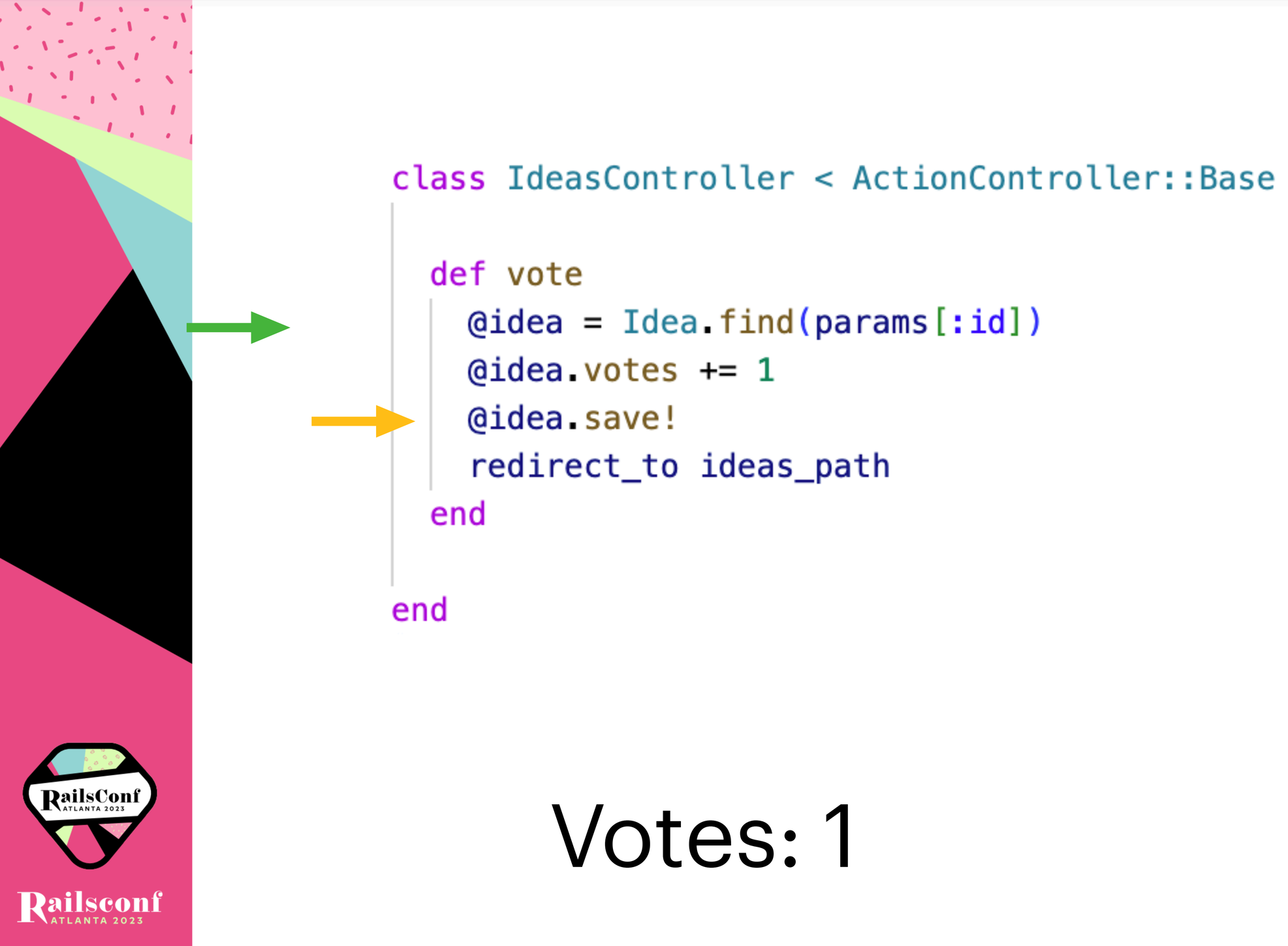
Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 0



Votes: 1

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

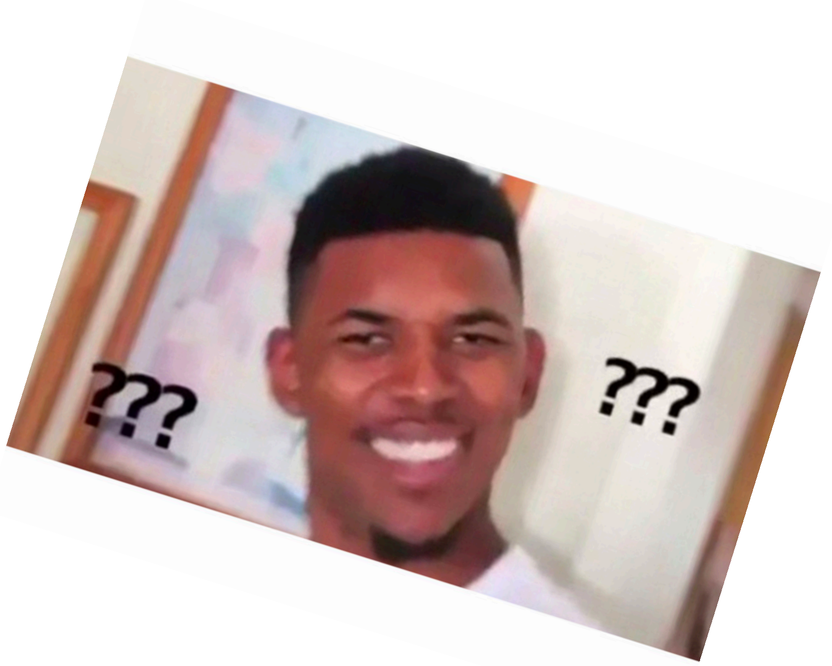
```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Votes: 1

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```



Votes: 1

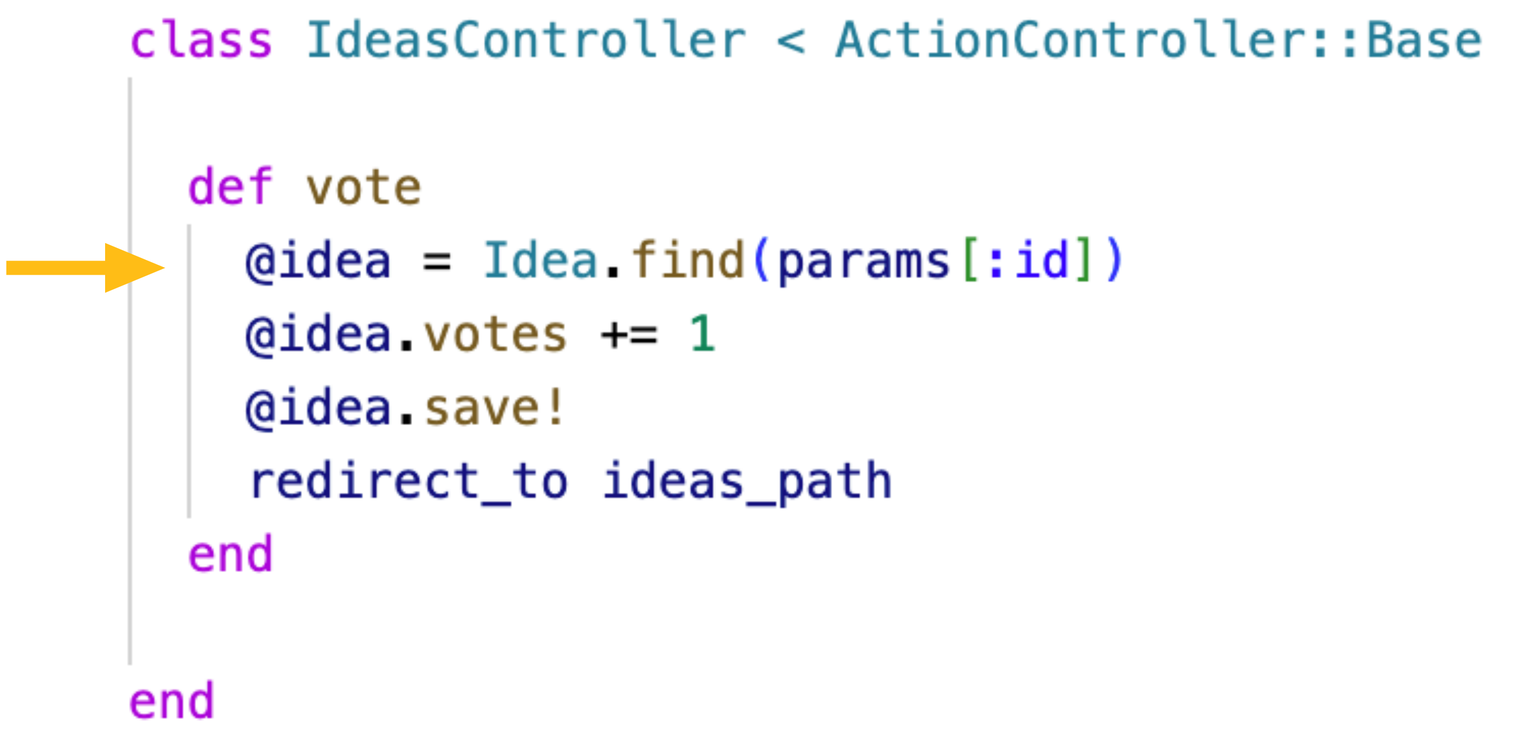
```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```sql
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```




```
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 0>
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 1>
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 1>
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

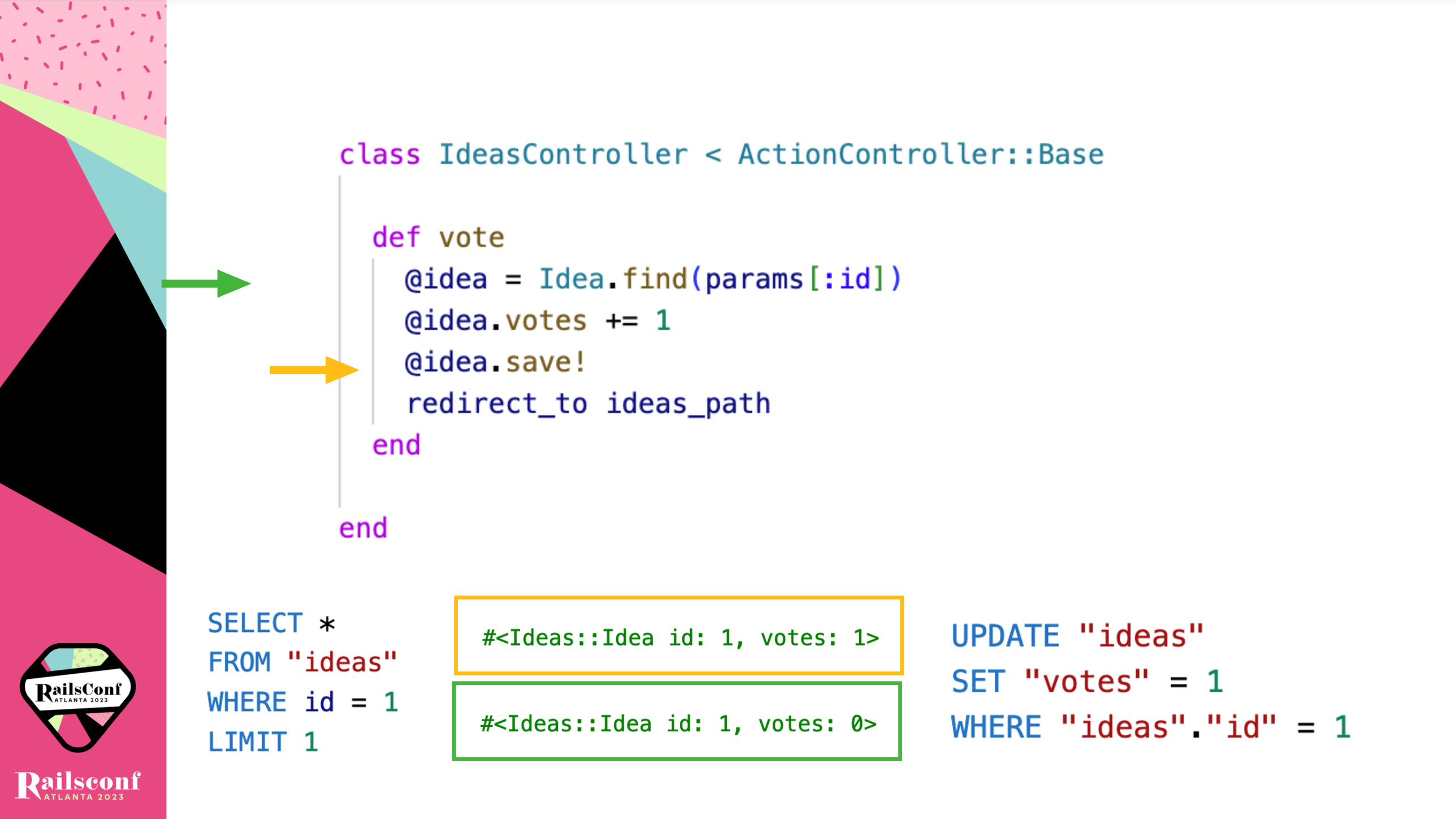
```
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```
#<Ideas::Idea id: 1, votes: 0>
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023


class IdeasController < ActionController::Base
def vote
@idea = Idea.find(params[:id])
@idea.votes += 1
@idea.save!
redirect_to ideas_path
end
end
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
#<Ideas::Idea id: 1, votes: 1>
#<Ideas::Idea id: 1, votes: 0>
UPDATE "ideas"
SET "votes" = 1
WHERE "ideas"."id" = 1
RailsConf
ATLANTA 2023
Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```sql
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```sql
UPDATE "ideas"
SET "votes" = 1
WHERE "ideas"."id" = 1
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```sql
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```sql
UPDATE "ideas"
SET "votes" = 1
WHERE "ideas"."id" = 1
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```sql
SELECT *
FROM "ideas"
WHERE id = 1
LIMIT 1
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```
#<Ideas::Idea id: 1, votes: 1>
```

```sql
UPDATE "ideas"
SET "votes" = 1
WHERE "ideas"."id" = 1
```

Railsconf
ATLANTA 2023

Read-modify-write

Manual Testing

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])

    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    sleep(10)
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])

    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

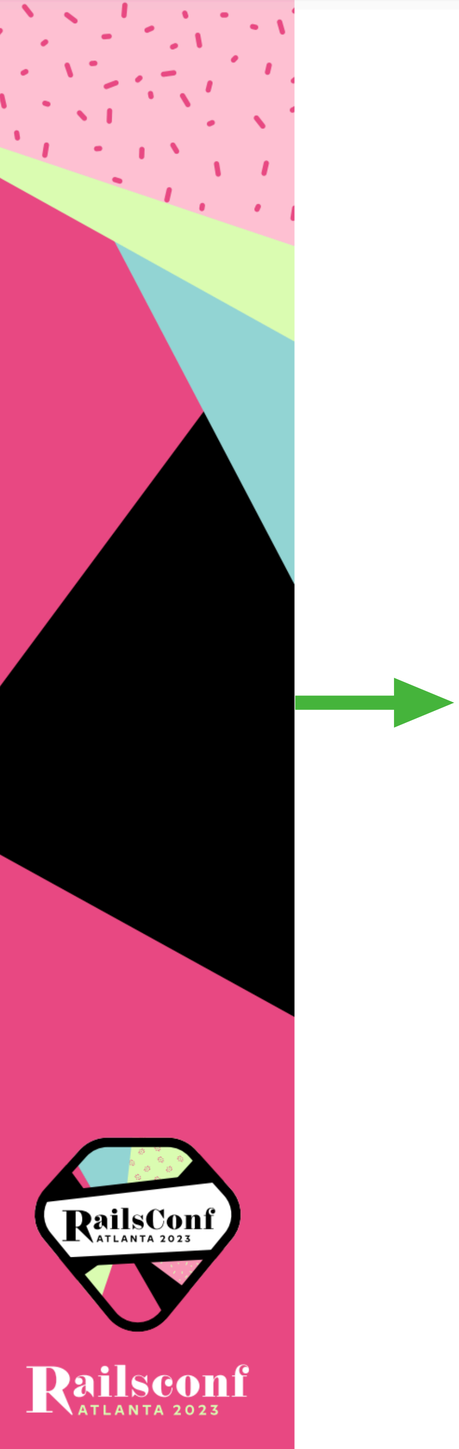
```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```


```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```



```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    if $redis.setnx("wait", 1)
      sleep(0.5) while $redis.get("wait")
    else
      $redis.del("wait")
    end
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

Automated Testing

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end

expect(Idea).to receive(:find).and_wrap_original do |method, *args|



end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end

expect(Idea).to receive(:find).and_wrap_original do |method, *args|
  method.call(*args).tap do |idea|

  end
end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  end

end

expect(Idea).to receive(:find).and_wrap_original do |method, *args|
  method.call(*args).tap do |idea|
    Idea.where(id: idea.id).update_all(votes: idea.votes + 1)
  end
end
```

```ruby
describe IdeasController do
  describe "POST vote" do
    it "handles race conditions when voting" do
      idea = Idea.create!(votes: 0)

      expect(Idea).to receive(:find).and_wrap_original do |method, *args|
        method.call(*args).tap do |idea|
          Idea.where(id: idea.id).update_all(votes: idea.votes + 1)
        end
      end

      post :vote, :id => idea.id

      expect(idea.votes).to eql 2
    end
  end
end
```

```ruby
describe IdeasController do
  describe "POST vote" do
    it "handles race conditions when voting" do
      idea = Idea.create!(votes: 0)

      expect(Idea).to receive(:find).and_wrap_original do |method, *args|
        method.call(*args).tap do |idea|
          Idea.where(id: idea.id).update_all(votes: idea.votes + 1)
        end
      end

      post :vote, :id => idea.id

      expect(idea.votes).to eql 2
    end
  end
end
```

```ruby
describe IdeasController do
  describe "POST vote" do
    it "handles race conditions when voting" do
      idea = Idea.create!(votes: 0)

      expect(Idea).to receive(:find).and_wrap_original do |method, *args|
        method.call(*args).tap do |idea|
          Idea.where(id: idea.id).update_all(votes: idea.votes + 1)
        end
      end

      post :vote, :id => idea.id

      expect(idea.votes).to eql 2
    end
  end
end
```

```ruby
describe IdeasController do
  describe "POST vote" do
    it "handles race conditions when voting" do
      idea = Idea.create!(votes: 0)

      expect(Idea).to receive(:find).and_wrap_original do |method, *args|
        method.call(*args).tap do |idea|
          Idea.where(id: idea.id).update_all(votes: idea.votes + 1)
        end
      end

      post :vote, :id => idea.id

      expect(idea.votes).to eql 2
    end
  end
end
```

```ruby
describe IdeasController do
  describe "POST vote" do
    it "handles race conditions when voting" do
      idea = Idea.create!(votes: 0)

      expect(Idea).to receive(:find).and_wrap_original do |method, *args|
        method.call(*args).tap do |idea|
          Idea.where(id: idea.id).update_all(votes: idea.votes + 1)
        end
      end

      post :vote, :id => idea.id

      expect(idea.votes).to eql 2
    end
  end
end
```

Railsconf
ATLANTA 2023

How to fix

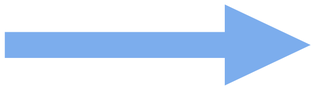
Railsconf
ATLANTA 2023

1. Remove the critical section

```ruby
@idea = Idea.find(id)
@idea.votes += 1
@idea.save!
```

Railsconf
ATLANTA 2023

```ruby
@idea = Idea.find(id)
@idea.votes += 1
@idea.save!
```

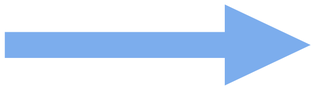


```ruby
Idea.where(id: id)
    .update_all(
      '"votes" = "votes" + 1'
    )
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
@idea = Idea.find(id)
@idea.votes += 1
@idea.save!
```



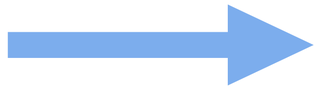
```ruby
Idea.where(id: id)
    .update_all(
      '"votes" = "votes" + 1'
    )
```

```
UPDATE "ideas"
SET "votes" = 1
WHERE "ideas"."id" = 1
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
@idea = Idea.find(id)
@idea.votes += 1
@idea.save!
```



```ruby
Idea.where(id: id)
    .update_all(
      '"votes" = "votes" + 1'
    )
```

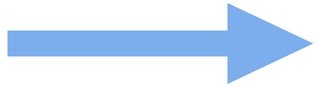
```
UPDATE "ideas"
SET "votes" = 1
WHERE "ideas"."id" = 1
```

```
UPDATE "ideas"
SET "votes" = "votes" + 1
WHERE "ideas"."id" = 1
```

```ruby
redis = Redis.new
count = redis.get("idea_count")
redis.set(
  "idea_count",
  count.to_i + 1
)
```

Railsconf
ATLANTA 2023

```ruby
redis = Redis.new
count = redis.get("idea_count")
redis.set(
  "idea_count",
  count.to_i + 1
)
```



```ruby
redis = Redis.new
count = redis.incr(
  "idea_count",
  1
)
```

Railsconf
ATLANTA 2023

2. Detect and recover

Ruby on Rails 7.0.4.2

Module

ActiveRecord::Locking::Optimistic

activerecord/lib/active_record/locking/optimistic.rb

Railsconf
ATLANTA 2023

```ruby
change_table :ideas do |t|
  t.integer :lock_version, default: 0
end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path


  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path


  end

end
```

```
UPDATE "ideas"
SET "votes" = 1,
    "lock_version" = 1
WHERE "ideas"."id" = 1
  AND "ideas"."lock_version" = 0
```

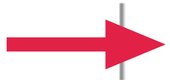
```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path


  end

end
```

```sql
UPDATE "ideas"
SET "votes" = 1,
  "lock_version" = 1
WHERE "ideas"."id" = 1
  AND "ideas"."lock_version" = 0
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path


  end

end
```

```
UPDATE "ideas"
SET "votes" = 1,
    "lock_version" = 1
WHERE "ideas"."id" = 1
 AND "ideas"."lock_version" = 0
```



```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path


  end

end
```

```
UPDATE "ideas"
SET "votes" = 1,
    "lock_version" = 1
WHERE "ideas"."id" = 1
  AND "ideas"."lock_version" = 0
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  rescue ActiveRecord::StaleObjectError
    retry
  end

end
```

```
UPDATE "ideas"
SET "votes" = 1,
    "lock_version" = 1
WHERE "ideas"."id" = 1
  AND "ideas"."lock_version" = 0
```

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  rescue ActiveRecord::StaleObjectError
    retry
  end

end
```

```sql
UPDATE "ideas"
SET "votes" = 1,
    "lock_version" = 1
WHERE "ideas"."id" = 1
  AND "ideas"."lock_version" = 0
```

1. How many times will retry?

```ruby
class IdeasController < ActionController::Base

  def vote
    @idea = Idea.find(params[:id])
    @idea.votes += 1
    @idea.save!
    redirect_to ideas_path
  rescue ActiveRecord::StaleObjectError
    retry
  end

end
```

```sql
UPDATE "ideas"
SET "votes" = 1,
    "lock_version" = 1
WHERE "ideas"."id" = 1
  AND "ideas"."lock_version" = 0
```

1. How many times will retry?
2. What will you do if the retries aren't successful?

Important note

All changes need to be undone on failure

Railsconf
ATLANTA 2023

3. Protect the critical section

RailsConf
ATLANTA 2023

Railsconf
ATLANTA 2023

Ruby on Rails 7.0.4.2

Module

ActiveRecord::Locking::Pessimistic

activerecord/lib/active_record/locking/pessimistic.rb

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```sql
SELECT *
FROM "ideas"
WHERE "ideas"."id" = 1
LIMIT 1
FOR UPDATE
```

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```sql
SELECT *
FROM "ideas"
WHERE "ideas"."id" = 1
LIMIT 1
FOR UPDATE
```

RailsConf
ATLANTA 2023

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023


```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```


```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```


```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

1. How long will you wait?

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.transaction do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

1. How long will you wait?
2. What do you do when there is a timeout?

leandromoreira/ **redlock-rb**

Redlock is a redis-based distributed lock implementation in Ruby. More than 15M downloads.

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    lock_manager = Redlock::Client.new

    lock_manager.lock!("idea_vote_#{params[:id]}", 1_000) do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    lock_manager = Redlock::Client.new

    lock_manager.lock!("idea_vote_#{params[:id]}", 1_000) do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    lock_manager = Redlock::Client.new

    lock_manager.lock!("idea_vote_#{params[:id]}", 1_000) do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023


```ruby
class IdeasController < ActionController::Base

  def vote
    lock_manager = Redlock::Client.new

    lock_manager.lock!("idea_vote_#{params[:id]}", 1_000) do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    lock_manager = Redlock::Client.new

    lock_manager.lock!("idea_vote_#{params[:id]}", 1_000) do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

ClosureTree/
with_advisory_lock

Advisory locking for ActiveRecord

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.with_advisory_lock("idea_vote_#{params[:id]}") do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.with_advisory_lock("idea_vote_#{params[:id]}") do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```


```ruby
class IdeasController < ActionController::Base

  def vote
    Idea.with_advisory_lock("idea_vote_#{params[:id]}") do
      @idea = Idea.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    redirect_to ideas_path
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    success = Idea.with_advisory_lock("idea_vote_#{params[:id]}", timeout_seconds: 1) do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    if success
      redirect_to ideas_path
    else
      flash[:error] = "Something went wrong"
      redirect_to ideas_path
    end
  end

end
```

Railsconf
ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    success = Idea.with_advisory_lock("idea_vote_#{params[:id]}", timeout_seconds: 1) do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    if success
      redirect_to ideas_path
    else
      flash[:error] = "Something went wrong"
      redirect_to ideas_path
    end
  end

end
```

```ruby
class IdeasController < ActionController::Base

  def vote
    success = Idea.with_advisory_lock("idea_vote_#{params[:id]}", timeout_seconds: 1) do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    if success
      redirect_to ideas_path
    else
      flash[:error] = "Something went wrong"
      redirect_to ideas_path
    end
  end

end
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
class IdeasController < ActionController::Base

  def vote
    success = Idea.with_advisory_lock("idea_vote_#{params[:id]}", timeout_seconds: 1) do
      @idea = Idea.lock.find(params[:id])
      @idea.votes += 1
      @idea.save!
    end
    if success
      redirect_to ideas_path
    else
      flash[:error] = "Something went wrong"
      redirect_to ideas_path
    end
  end

end
```

Important note

Be really mindful of how long you will wait

Railsconf
ATLANTA 2023

Railsconf
ATLANTA 2023

Check-then-act

validates_uniqueness_of(*attr_names) Link

Validates whether the value of the specified attributes are unique across the system. Useful for making sure that only one user can be named “davidhh”.

```
class Person < ActiveRecord::Base
  validates_uniqueness_of :user_name
end
```

It can also validate whether the value of the specified attributes are unique based on a `:scope` parameter:

```
class Person < ActiveRecord::Base
  validates_uniqueness_of :user_name, scope: :account_id
end
```

Concurrency and integrity

Using this validation method in conjunction with ActiveRecord::Base#save does not guarantee the absence of duplicate record insertions, because uniqueness checks on the application level are inherently prone to race conditions. For example, suppose that two users try to post a Comment at the same time, and a Comment's title must be unique. At the database-level, the actions performed by these users could be interleaved in the following manner:

Concurrency and integrity

Using this validation method in conjunction with ActiveRecord::Base#save does not guarantee the absence of duplicate record insertions, because uniqueness checks on the application level are inherently prone to race conditions. For example, suppose that two users try to post a Comment at the same time, and a Comment's title must be unique. At the database-level, the actions performed by these users could be interleaved in the following manner:

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

Railsconf
ATLANTA 2023

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end


Idea.create(name: "My Idea")
```

Railsconf
ATLANTA 2023

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

```ruby
Idea.create(name: "My Idea")
```

```ruby
def create(attributes = nil, &block)
  if attributes.is_a?(Array)
    attributes.collect { |attr| create(attr, &block) }
  else
    object = new(attributes, &block)
    object.save
    object
  end
end
```

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

```ruby
Idea.create(name: "My Idea")
```

```ruby
def create(attributes = nil, &block)
  if attributes.is_a?(Array)
    attributes.collect { |attr| create(attr, &block) }
  else
    object = new(attributes, &block)
    object.save
    object
  end
end
```

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

```ruby
Idea.create(name: "My Idea")
```

```ruby
def create(attributes = nil, &block)
  if attributes.is_a?(Array)
    attributes.collect { |attr| create(attr, &block) }
  else
    object = new(attributes, &block)
    object.save
    object
  end
end
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

Railsconf
ATLANTA 2023

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

```ruby
Idea.create(name: "My Idea")
```

```ruby
def create(attributes = nil, &block)
  if attributes.is_a?(Array)
    attributes.collect { |attr| create(attr, &block) }
  else
    object = new(attributes, &block)
    object.save
    object
  end
end
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```ruby
class Idea < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

```ruby
Idea.create(name: "My Idea")
```

```ruby
def create(attributes = nil, &block)
  if attributes.is_a?(Array)
    attributes.collect { |attr| create(attr, &block) }
  else
    object = new(attributes, &block)
    object.save
    object
  end
end
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

Railsconf
ATLANTA 2023

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```sql
SELECT 1 AS one
FROM "ideas"
WHERE "ideas"."name" = 'My Idea'
LIMIT 1
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```sql
SELECT 1 AS one
FROM "ideas"
WHERE "ideas"."name" = 'My Idea'
LIMIT 1
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```sql
SELECT 1 AS one
FROM "ideas"
WHERE "ideas"."name" = 'My Idea'
LIMIT 1
```

```sql
INSERT INTO "ideas" ("name")
VALUES ('My Idea')
RETURNING "id"
```

```ruby
def save(**options)
  perform_validations(options) ? super : false
end
```

```
INSERT INTO "ideas" ("name")
VALUES ('My Idea')
RETURNING "id"
```

```
SELECT 1 AS one
FROM "ideas"
WHERE "ideas"."name" = 'My Idea'
LIMIT 1
```

```
INSERT INTO "ideas" ("name")
VALUES ('My Idea')
RETURNING "id"
```

Railsconf
ATLANTA 2023

How to fix

1. Removing the critical section

```
Idea.create(name: "My Idea")
```

Railsconf
ATLANTA 2023

1. Removing the critical section

```ruby
Idea.create(name: "My Idea")
```

```ruby
Idea.where(name: "My idea")
  .upsert({}, unique_by: :name)
```

1. Removing the critical section

```ruby
Idea.create(name: "My Idea")
```

```ruby
Idea.where(name: "My idea")
  .upsert({}, unique_by: :name)
```

1. Removing the critical section

```ruby
Idea.create(name: "My Idea")
```

```ruby
Idea.where(name: "My idea")
  .upsert({}, unique_by: :name)
```

```
INSERT INTO "ideas" ("name")
VALUES ('My idea')
ON CONFLICT ("name") DO NOTHING
RETURNING "id"
```

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

1. Removing the critical section

```ruby
Idea.create(name: "My Idea")
```

```ruby
Idea.where(name: "My idea")
  .upsert({}, unique_by: :name)
```

```
INSERT INTO "ideas" ("name")
VALUES ('My idea')
ON CONFLICT ("name") DO NOTHING
RETURNING "id"
```

2. Detect and Recover

```
Idea.create(name: "My Idea")
```

Railsconf
ATLANTA 2023

2. Detect and Recover

```ruby
Idea.create(name: "My Idea")
```

```ruby
begin
  idea = Idea.new(name: "My Idea")
  idea.save
rescue ActiveRecord::RecordNotUnique
  idea = Idea.find_by(name: "My Idea")
end
```

2. Detect and Recover

```ruby
Idea.create(name: "My Idea")
```

```ruby
begin
  idea = Idea.new(name: "My Idea")
  idea.save
rescue ActiveRecord::RecordNotUnique
  idea = Idea.find_by(name: "My Idea")
end
```

2. Detect and Recover

```ruby
Idea.create(name: "My Idea")
```

```ruby
begin
  idea = Idea.new(name: "My Idea")
  idea.save
rescue ActiveRecord::RecordNotUnique
  idea = Idea.find_by(name: "My Idea")
end
```

3. Protect critical section

```
Idea.create(name: "My Idea")
```

Railsconf
ATLANTA 2023

3. Protect critical section

```ruby
Idea.create(name: "My Idea")
```

```ruby
name = "My Idea"
Idea.with_advisory_lock("idea_uniqueness_#{name}") do
  idea = Idea.find_or_create_by(name: name)
end
```

3. Protect critical section

```ruby
Idea.create(name: "My Idea")
```

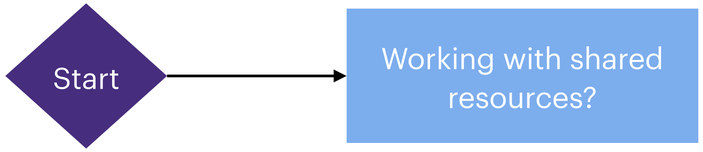
```ruby
name = "My Idea"
Idea.with_advisory_lock("idea_uniqueness_#{name}") do
  idea = Idea.find_or_create_by(name: name)
end
```

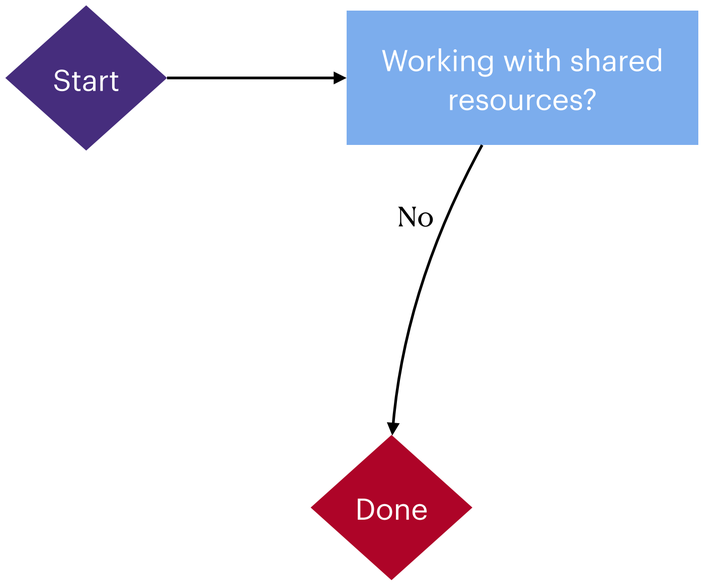
3. Protect critical section

```ruby
Idea.create(name: "My Idea")
```

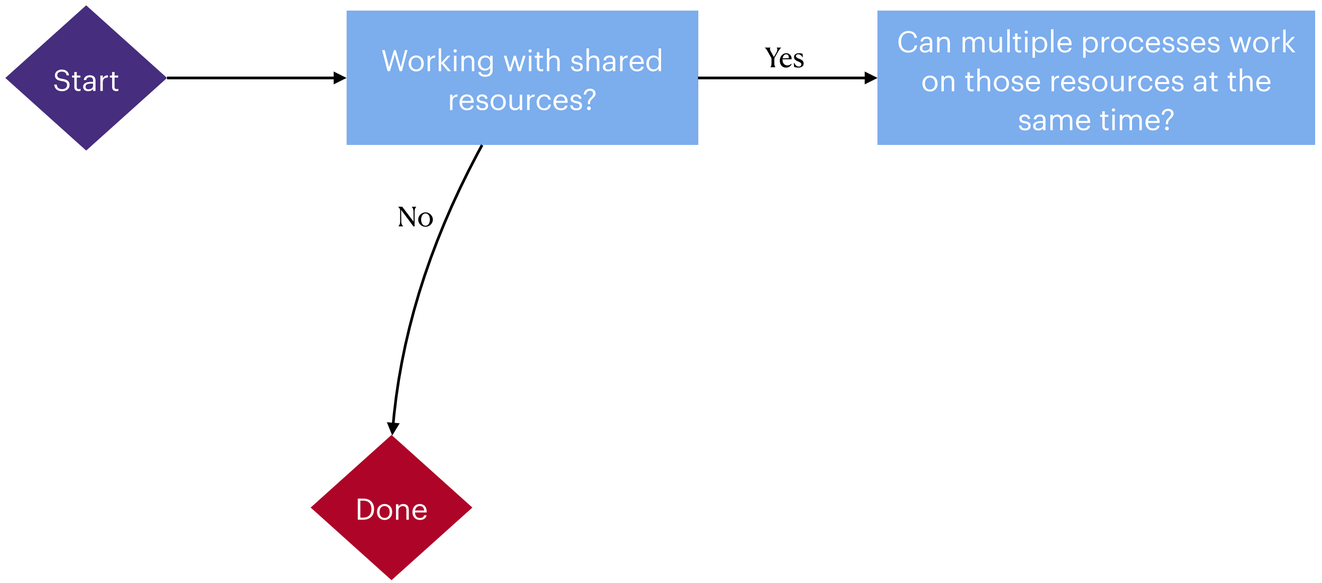
```ruby
name = "My Idea"
Idea.with_advisory_lock("idea_uniqueness_#{name}") do
  idea = Idea.find_or_create_by(name: name)
end
```

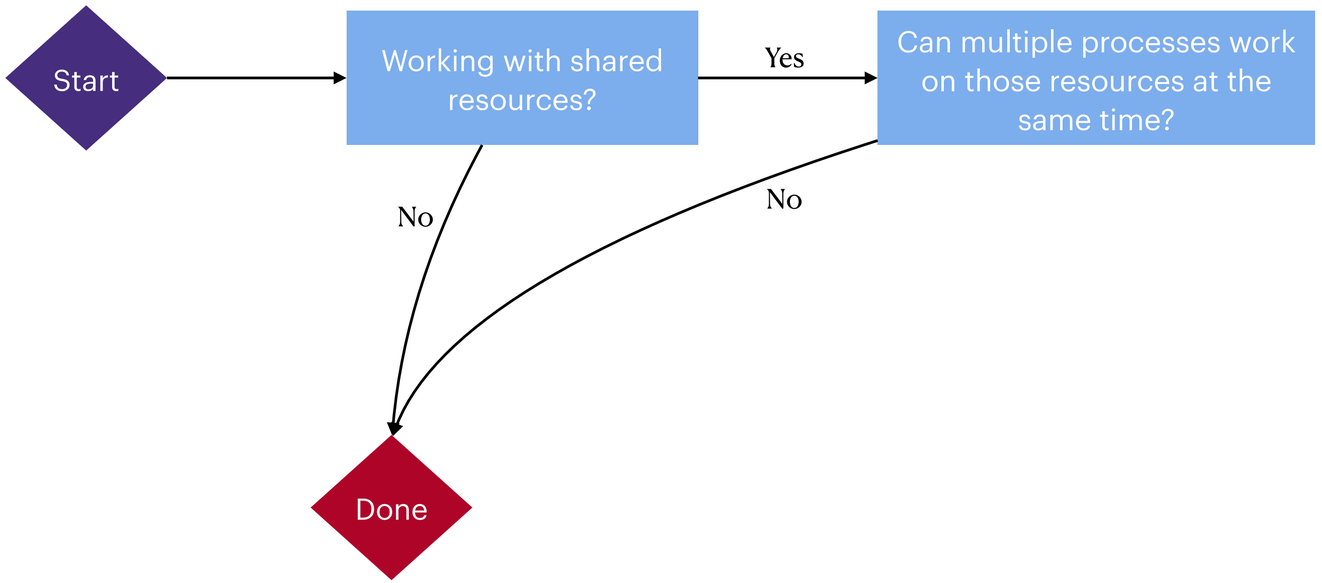
Railsconf
ATLANTA 2023

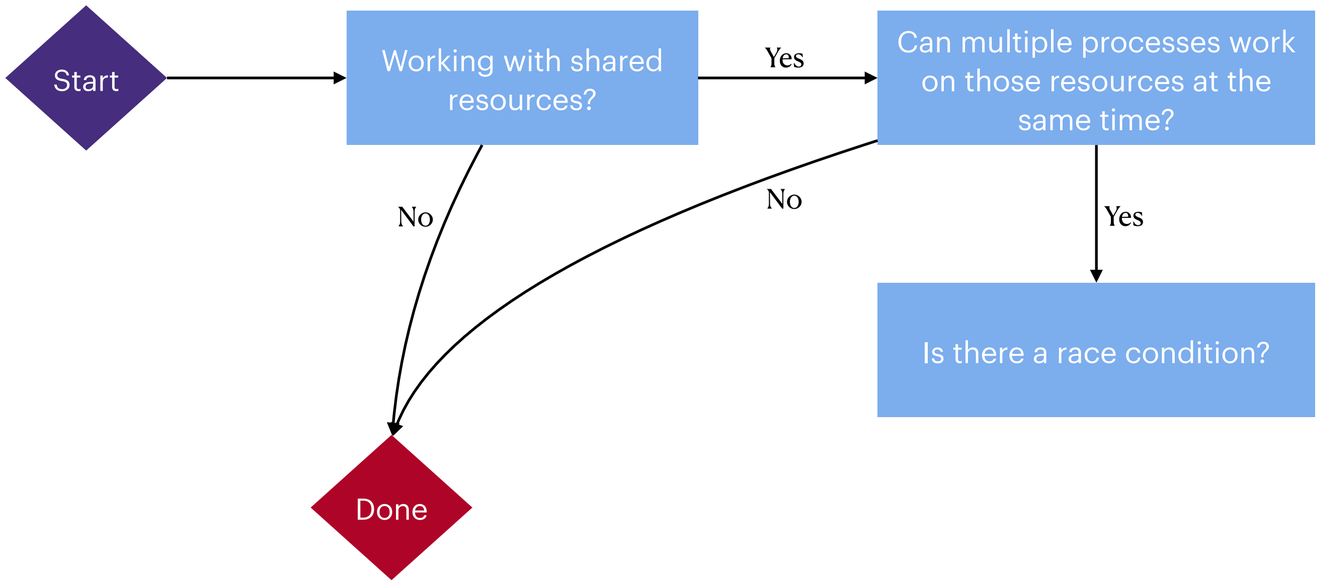

Start
Working with shared resources?

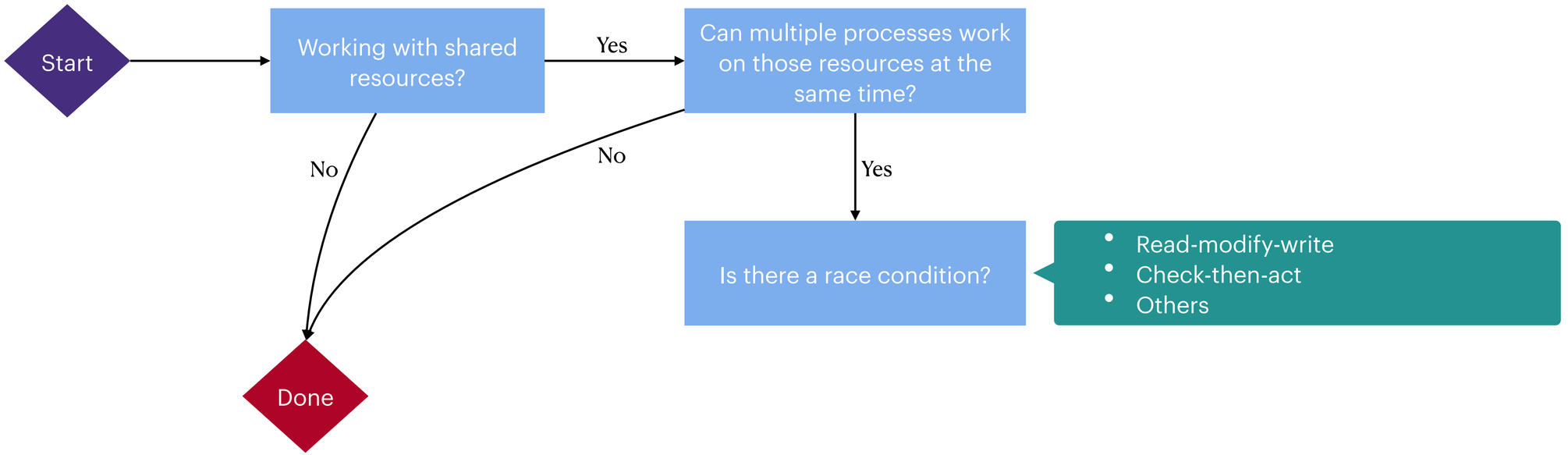


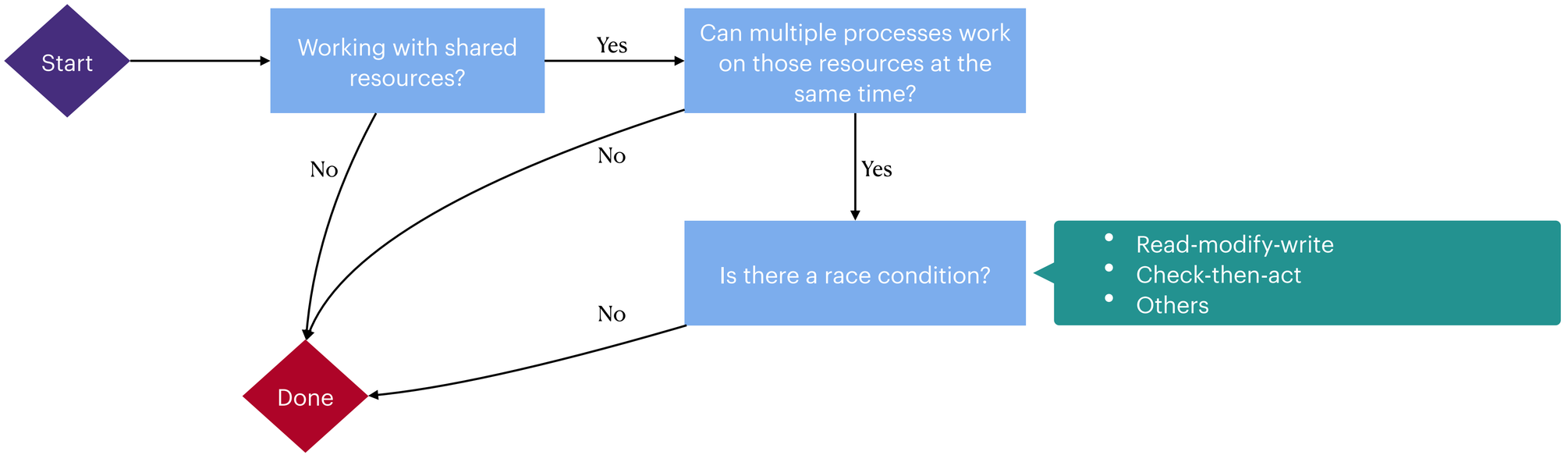
Railsconf
ATLANTA 2023

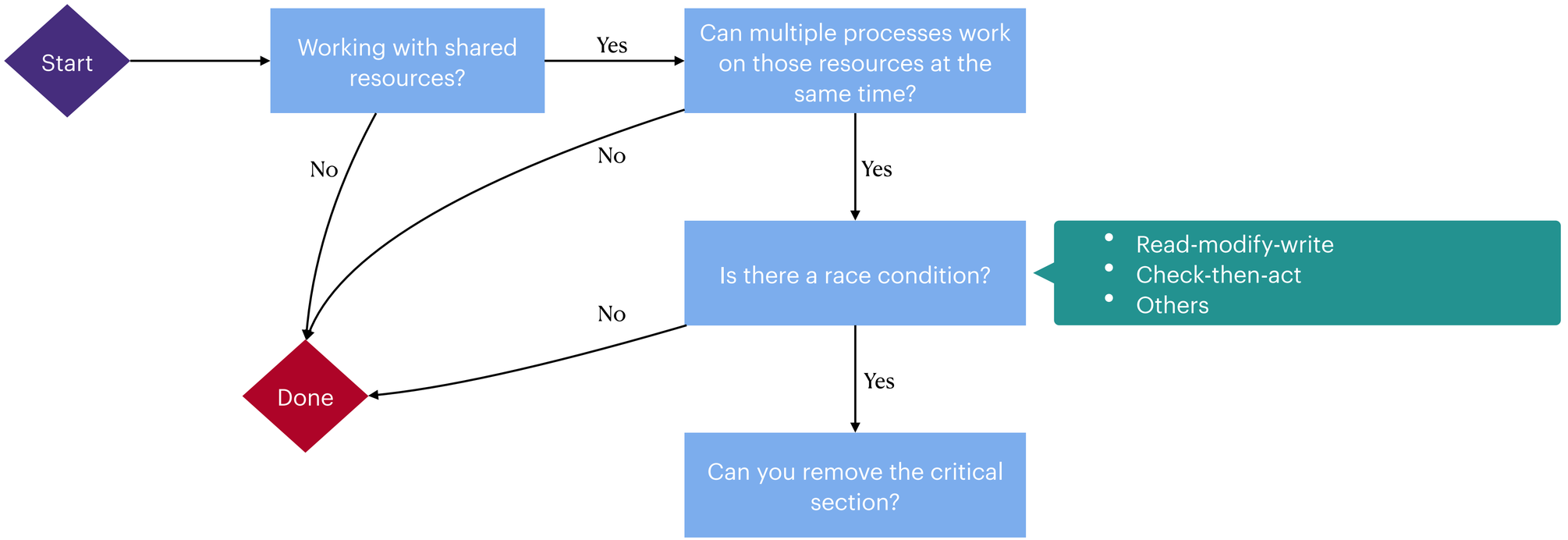


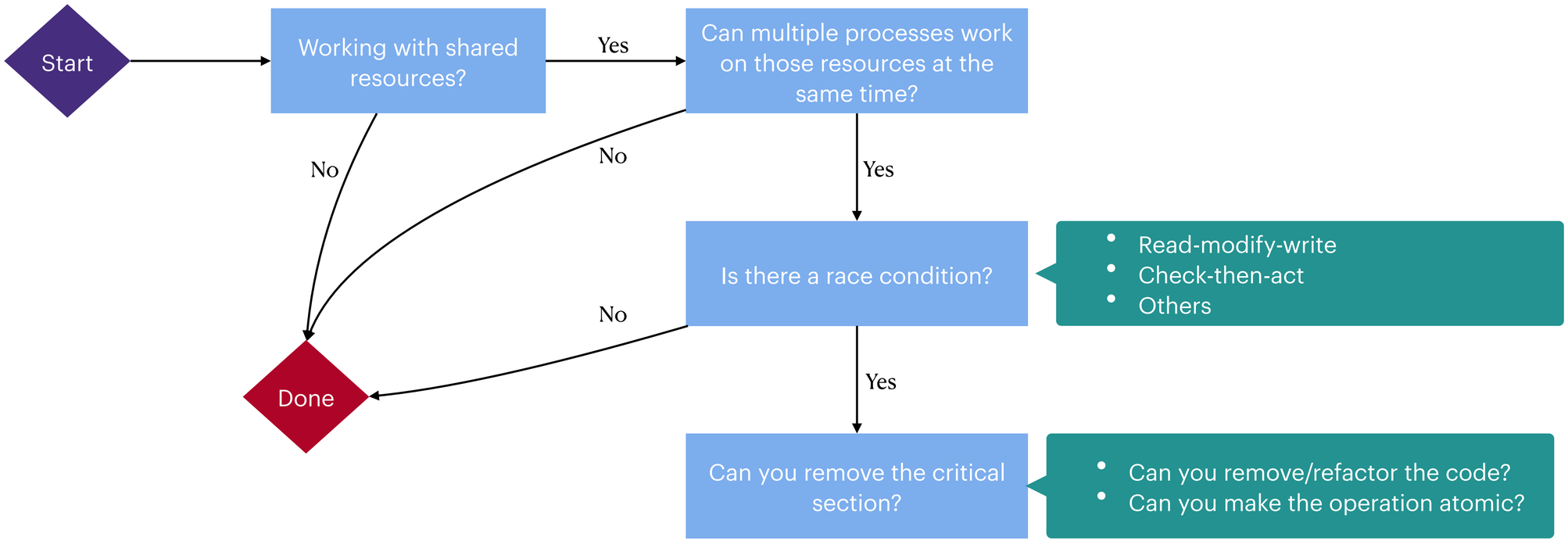


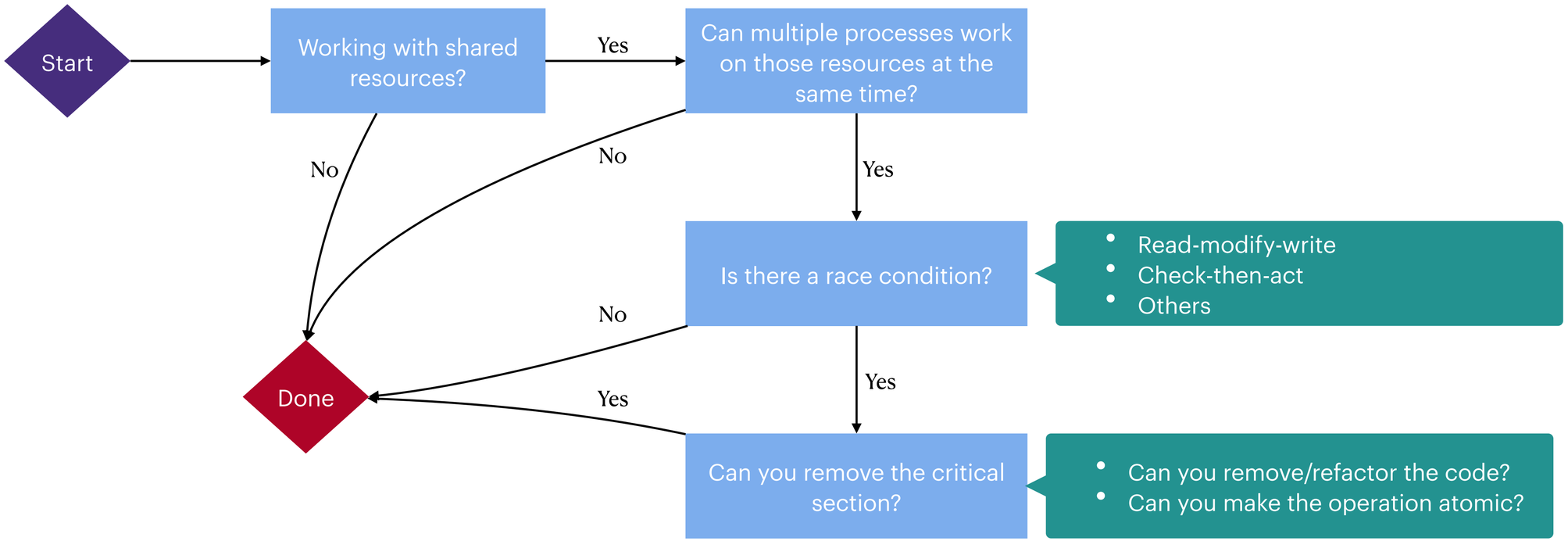

Start
Working with shared resources?
Yes
Can multiple processes work on those resources at the same time?
No
No
Yes
Is there a race condition?
Done

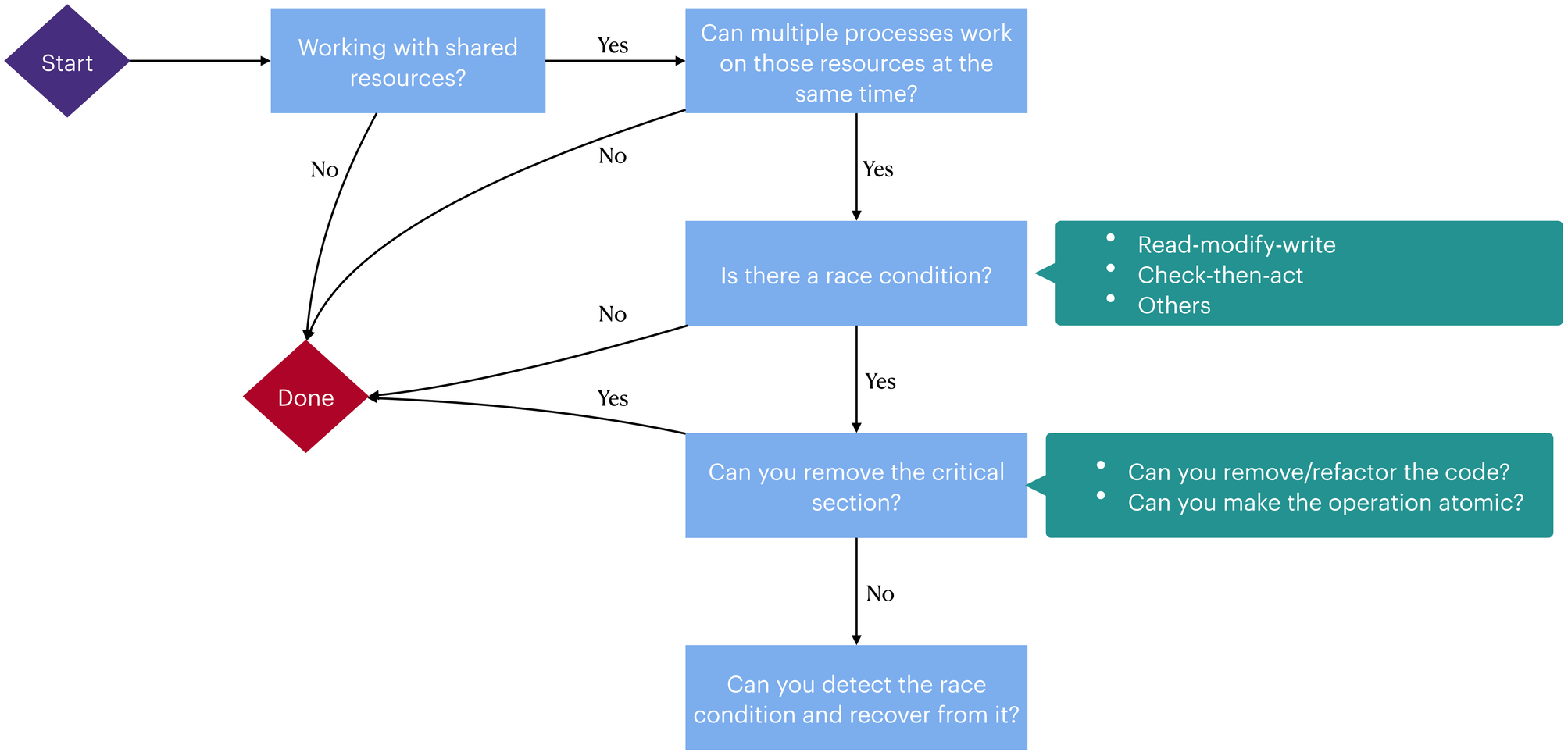

Start
Working with shared resources?
Yes
Can multiple processes work on those resources at the same time?
No
No
Yes
Is there a race condition?
Read-modify-write
Check-then-act
Others
Done

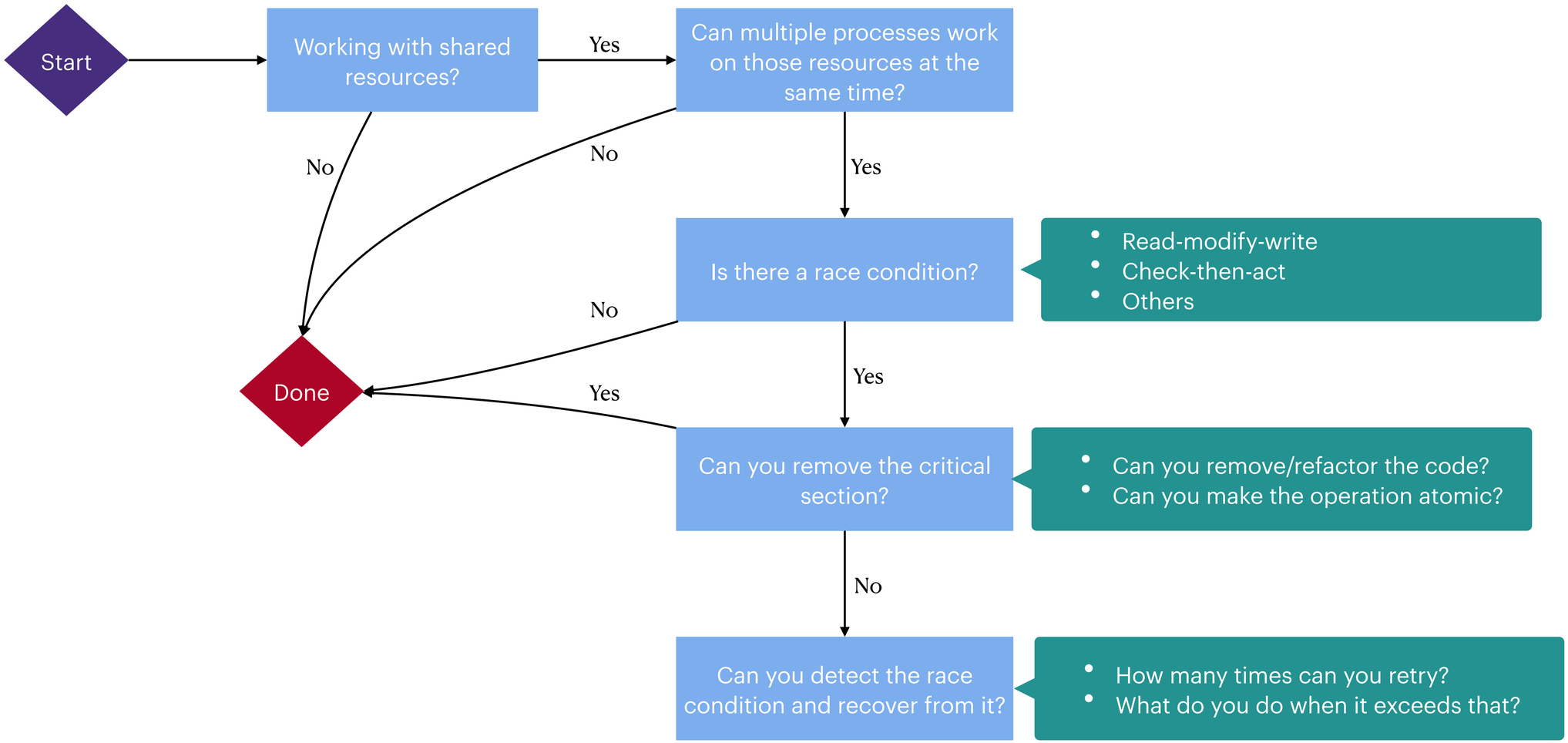

Start
Working with shared resources?
Yes
Can multiple processes work on those resources at the same time?
No
No
Yes
Is there a race condition?
Read-modify-write
Check-then-act
Others
No
Done

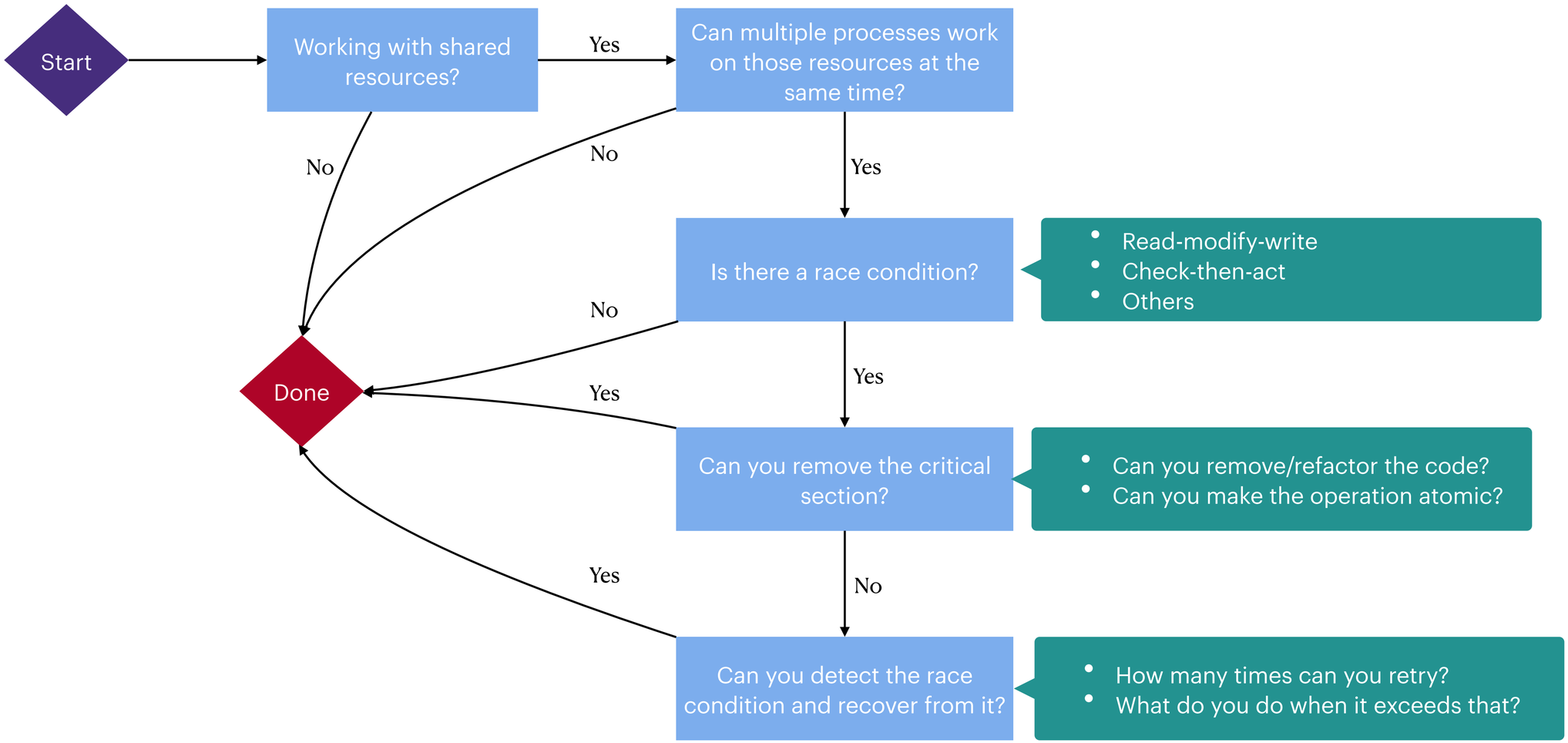


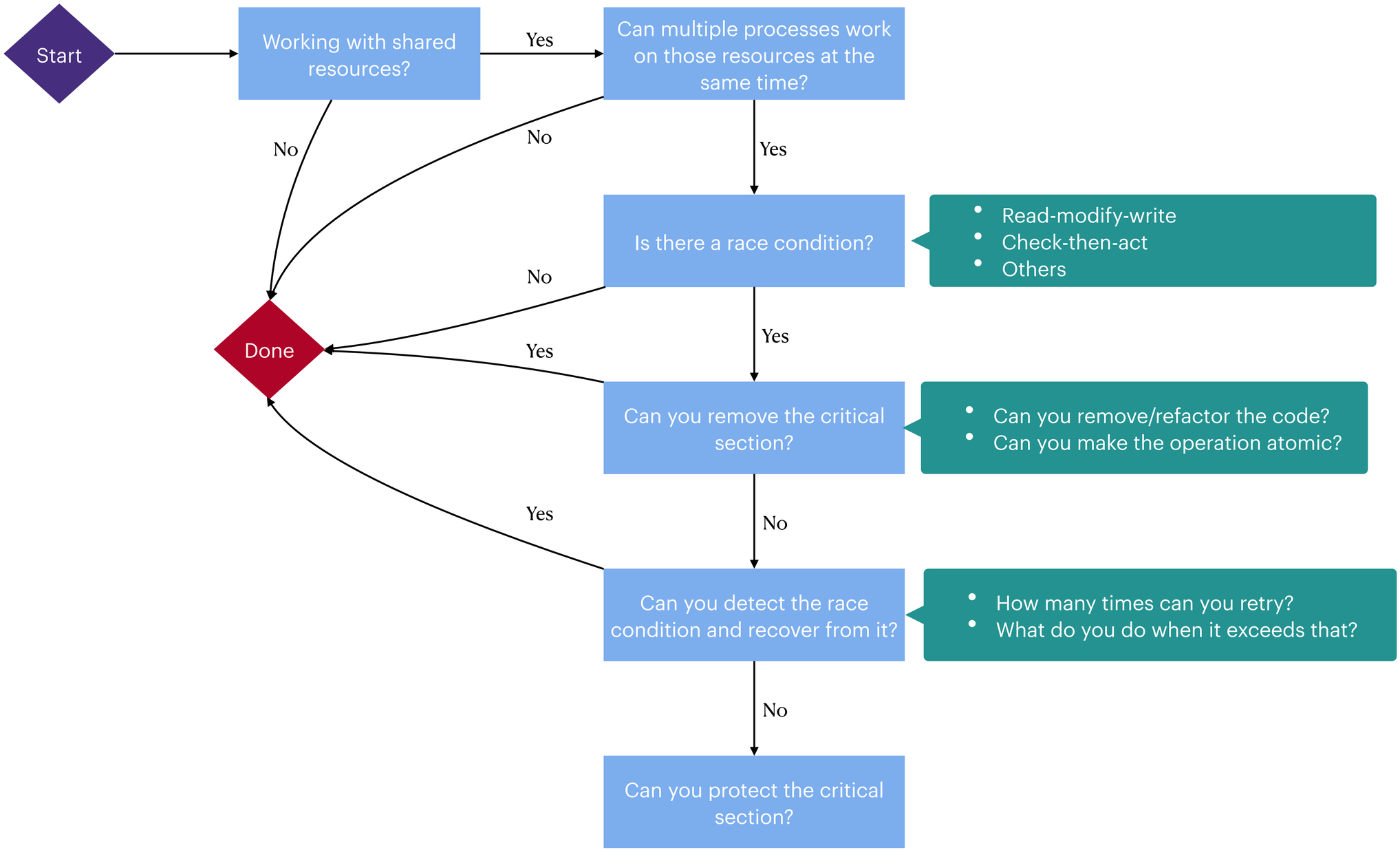

Start
Working with shared resources?
Yes
Can multiple processes work on those resources at the same time?
No
No
Yes
Is there a race condition?
Read-modify-write
Check-then-act
Others
No
Done
Yes
Can you remove the critical section?
Can you remove/refactor the code?
Can you make the operation atomic?





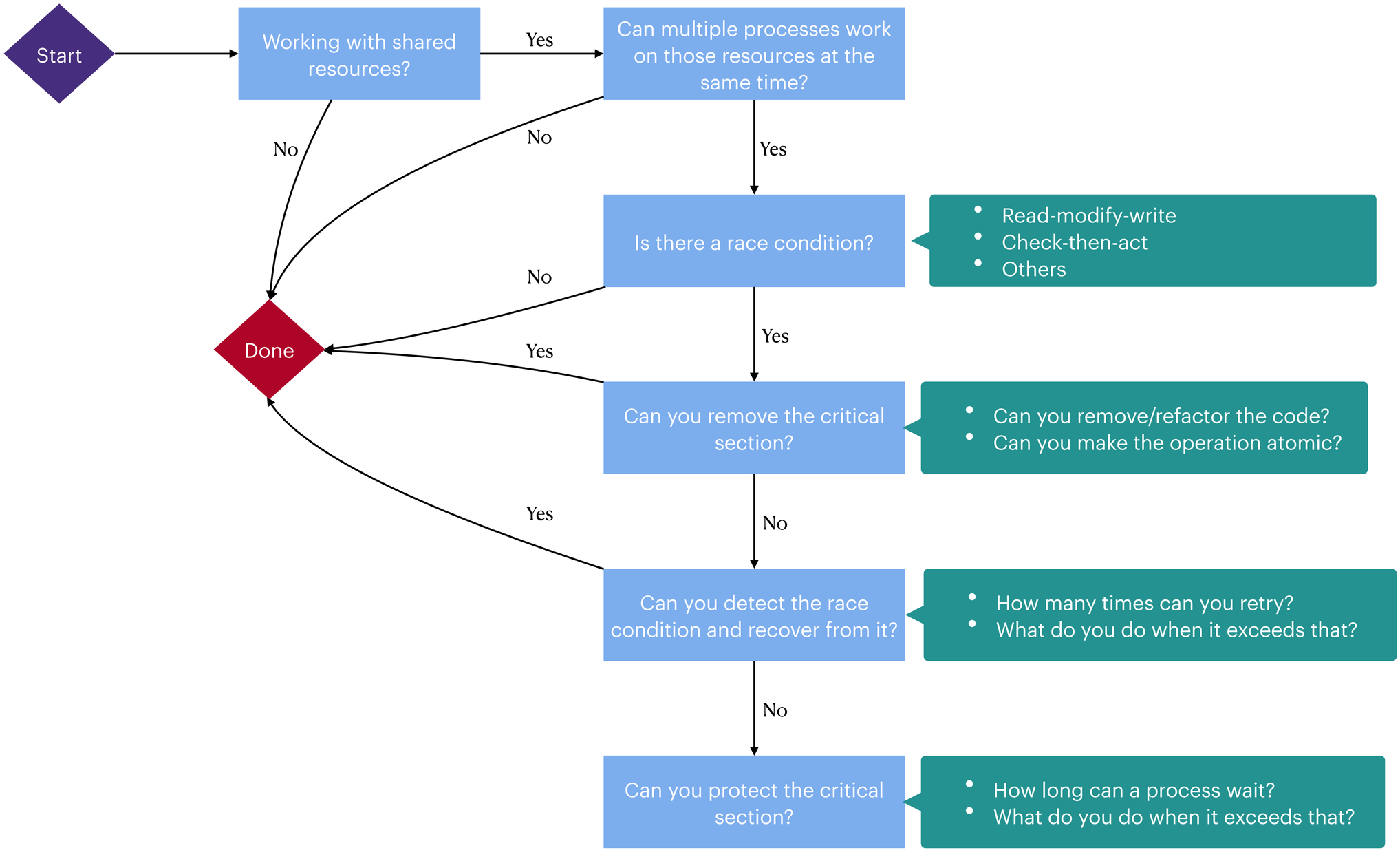


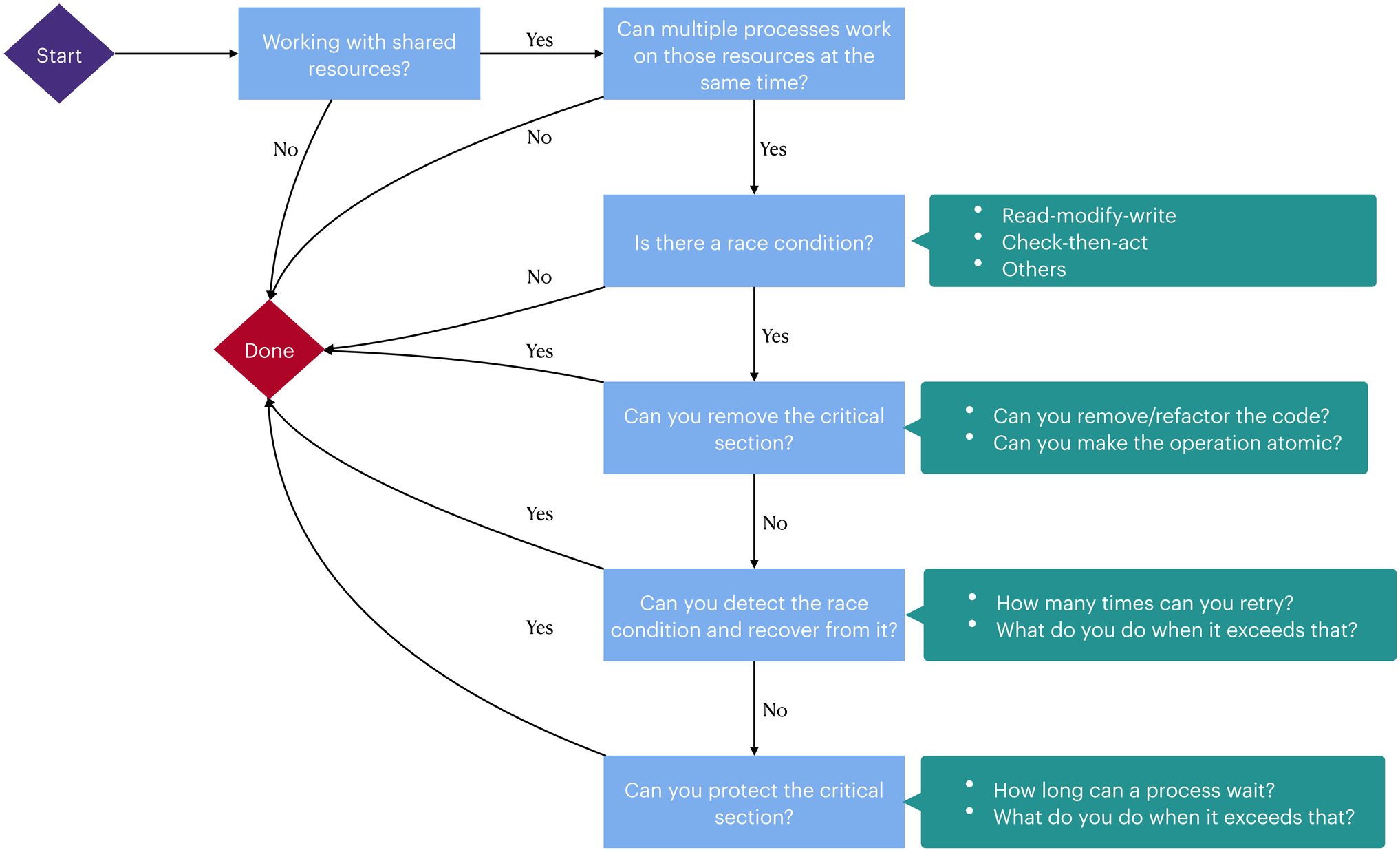


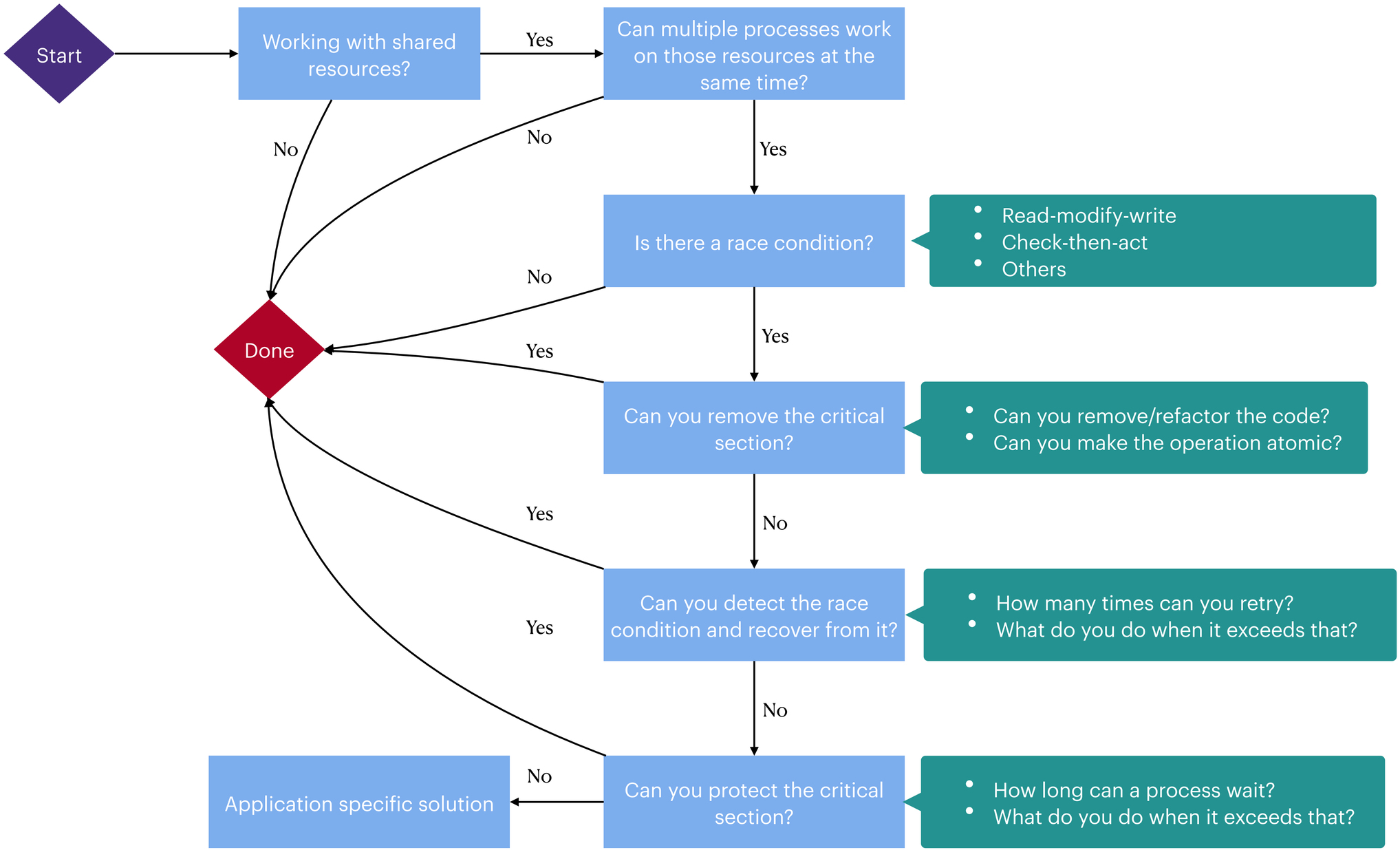


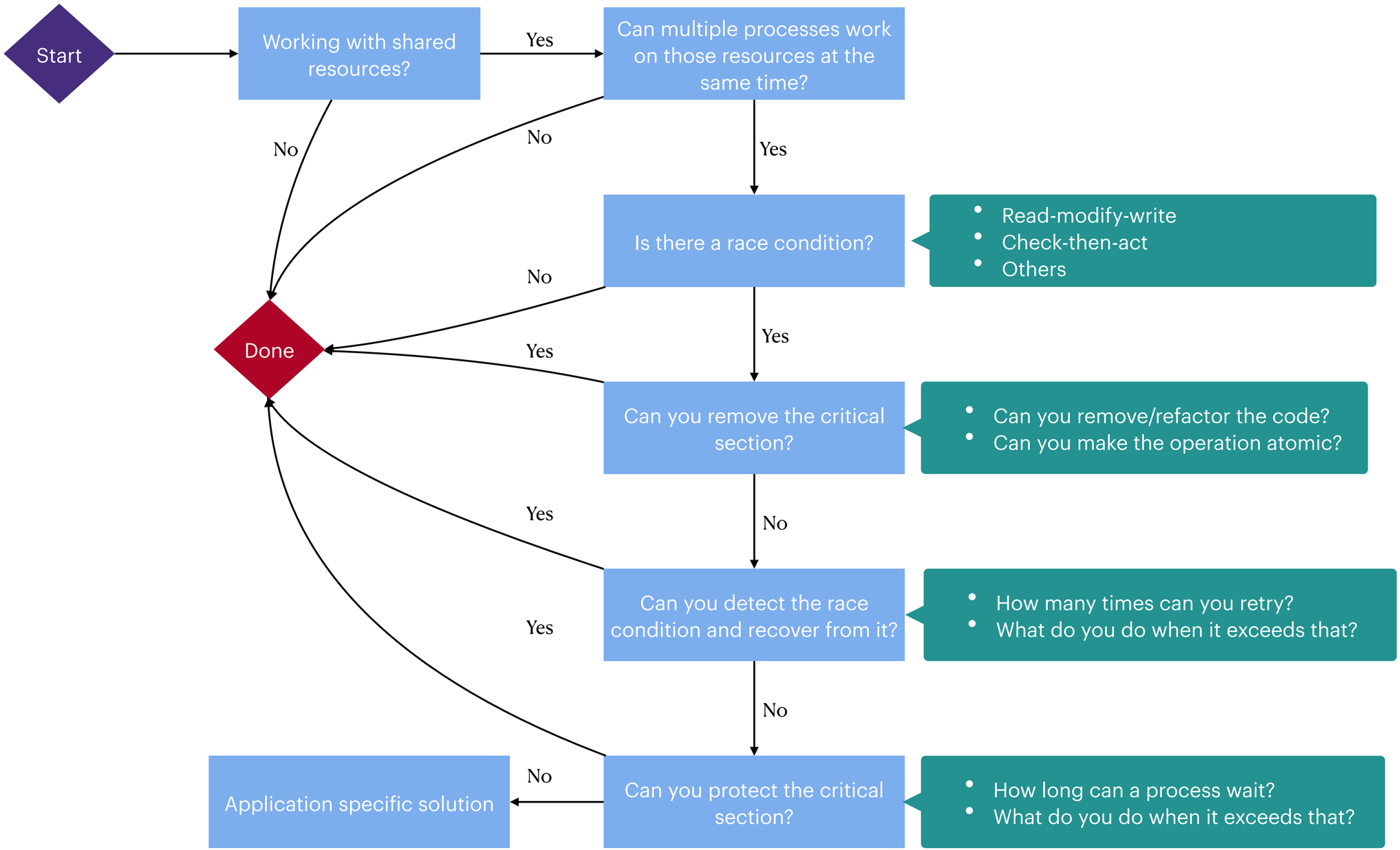
RailsConf ATLANTA 2023

Railsconf ATLANTA 2023









Railsconf
ATLANTA 2023

Thank you