Computational Modeling of Social Behavior

Day 3

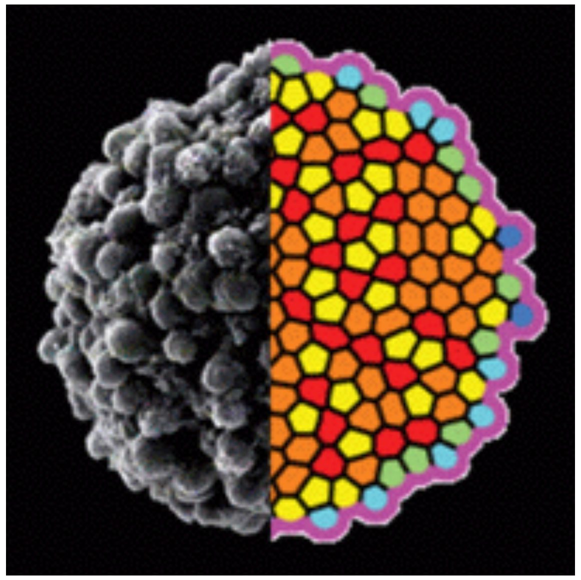
The Evolution of Cooperation

Paul Smaldino

Outline of the day

- **Morning**
  - Cooperation and the Prisoner's Dilemma
  - Paths to cooperation: Positive assortment
- **Afternoon**
  - Modeling agent interactions and exploring assumptions

The Problem of Cooperation



Individual organisms helping others at a cost

How does **cooperation** emerge in a population?

Once present, how do **cooperators** maintain an advantage over **non-cooperators**?

What factors facilitate more or less cooperation?

The Prisoner's Dilemma

The archetypal model for altruistic cooperation

Player 1	Player 2: Cooperate	Player 2: Defect
Cooperate	$R = 3$	$S = 0$
Defect	$T = 5$	$P = 1$

$$T > R > P > S$$
$$R > (T + S)/2$$

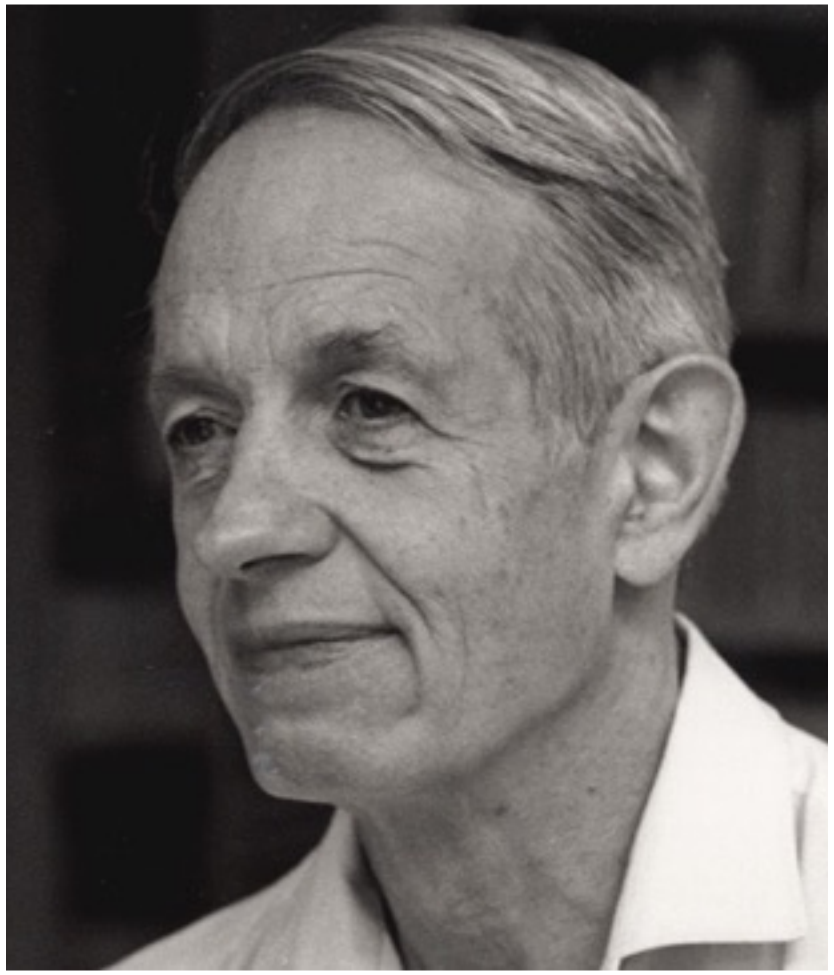
(Note: payoff is to Player 1)

Stag Hunt Game

Snowdrift Game

Nash Equilibrium

- Game solution in which no player can benefit by unilaterally changing his or her strategy
- Some games have no Nash Equilibrium, some have several
- In one-shot PD game, Nash equilibrium is **Defect**


John Nash
1928–2015

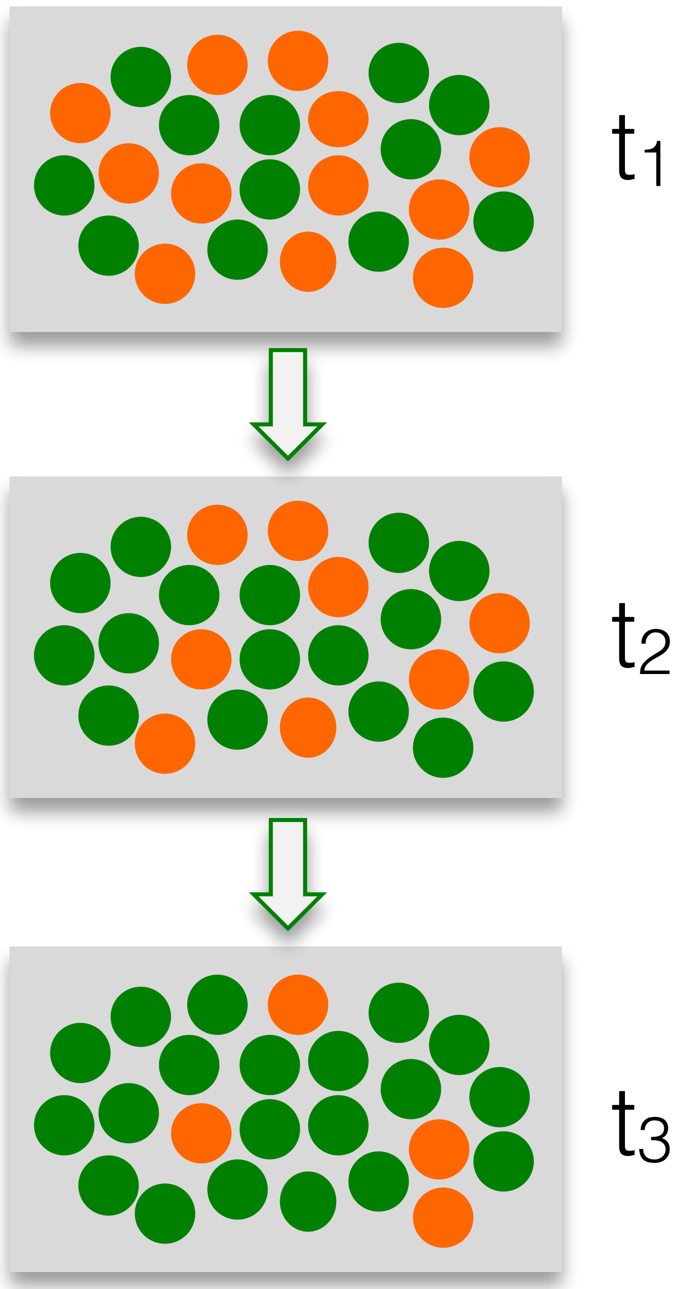
Evolutionary game theory

- Instead of considering optimal strategic decision makers, assume that strategies acts as traits that can be transmitted (either genetically or through social learning)
- Added realism: Not just what strategy is best, but how can we get from one population structure to another

Natural Selection

Traits: 🟠 🟢

Fitness of **green** > fitness of **orange**



How could cooperative strategies evolve?

- **Inclusive fitness (“kin selection”)**
  - If I help close relatives, I’m helping my genes to survive, even if they’re not **my** genes
  - Altruism can evolve if its directed toward relatives
- **Reciprocal altruism**
  - If we’re likely to meet often, we can build up expectations of reciprocal cooperation

Evolutionary Stable Strategy (ESS)

- In an evolutionary game, Strategy A is an ESS against Strategy B if a rare mutant of type B cannot invade a population where A is common.

Iterated games

ALLD is an ESS against ALLC

- Assume all games continue for another round with probability *w.*
- Payoff to ALLD vs. ALLD = $P + wP + w^2P + \ldots$
- Payoff to ALLC vs. ALLD = $S + wS + w^2S + \ldots$

$$\boxed{P > S\text{, so ALLC cannot invade}}$$

ALLC can be invaded by ALLD

- Assume all games continue for another round with probability *w.*
- Payoff to ALLC vs. ALLC = $R + wR + w^2R + \dots$
- Payoff to ALLD vs. ALLC = $T + wT + w^2T + \dots$

$\boxed{\text{T > R, so ALLD can always invade}}$

Axelrod & Hamilton's Tournament

- Iterated prisoner's dilemma: game played for multiple rounds
- Contestants submitted strategies that could learn and adapt
- All strategies paired in round robin tournament with game length of 200 moves
- Tournament 1: 14 entries + random
- Tournament 2: 62 entries
- The winner both times: **TIT FOR TAT**
  - Plays Cooperate on first move and then copies opponent's previous move
  - TFT is nice, retaliatory, forgiving, and clear

Axelrod & Hamilton (1981) *Science*

TFT can resist invasion by ALLD

- Assume all games continue for another round with probability *w.*
- Payoff to TFT vs. TFT = $R + wR + w^2R + \ldots$
- Payoff to ALLD vs. TFT = $T + wP + w^2P + \ldots$

T > R, but R > P. So, if games go on long enough, maybe TFT can be an ESS against ALLD…

When can TFT resist invasion by ALLD?

$$\begin{aligned} V(TFT|TFT) &= R + wR + w^2R + w^3R + \dots \\ &= R(1 + w + w^2 + w^3 + \dots) \end{aligned}$$

$$\begin{aligned} x &= 1 + w + w^2 + w^3 + \dots \\ &= 1 + w(1 + w + w^2 + w^3 + \dots) \\ &= 1 + wx \end{aligned}$$

$$x = 1 + wx$$

$$x - wx = 1$$

$$x(1 - w) = 1$$

$$x = \frac{1}{1 - w}$$

$$\boxed{V(TFT|TFT) = \frac{R}{1 - w}}$$

When can TFT resist invasion by ALLD?

$$\begin{aligned} V(ALLD|TFT) &= T + wP + w^2P + w^3P + \dots \\ &= T + P(w + w^2 + w^3 + \dots) \end{aligned}$$

$$\begin{aligned} z &= w + w^2 + w^3 + \dots \\ &= w(1 + w + w^2 + \dots) \\ &= wx \\ &= \frac{w}{1-w} \end{aligned}$$

$$\boxed{V(ALLD|TFT) = T + \frac{wP}{1-w}}$$

When is $V(TFT|TFT) > V(ALLD|TFT)$?

When can TFT resist invasion by ALLD?

$$\frac{R}{1-w} > T + \frac{wP}{1-w}$$

$$\frac{R-wP}{1-w} > T$$

$$R - wP > T - wT$$

$$R - T > wP - wT$$

$$R - T > w\underbrace{(P-T)}_{\text{neg}}$$

$$w > \frac{R-T^{\text{ neg}}}{P-T_{\text{ neg}}}$$

$$\boxed{w > \frac{T-R}{T-P}}$$

Hamiltonian payoffs:

	C	D
C	*b* – *c*	–*c*
D	*b*	*0*

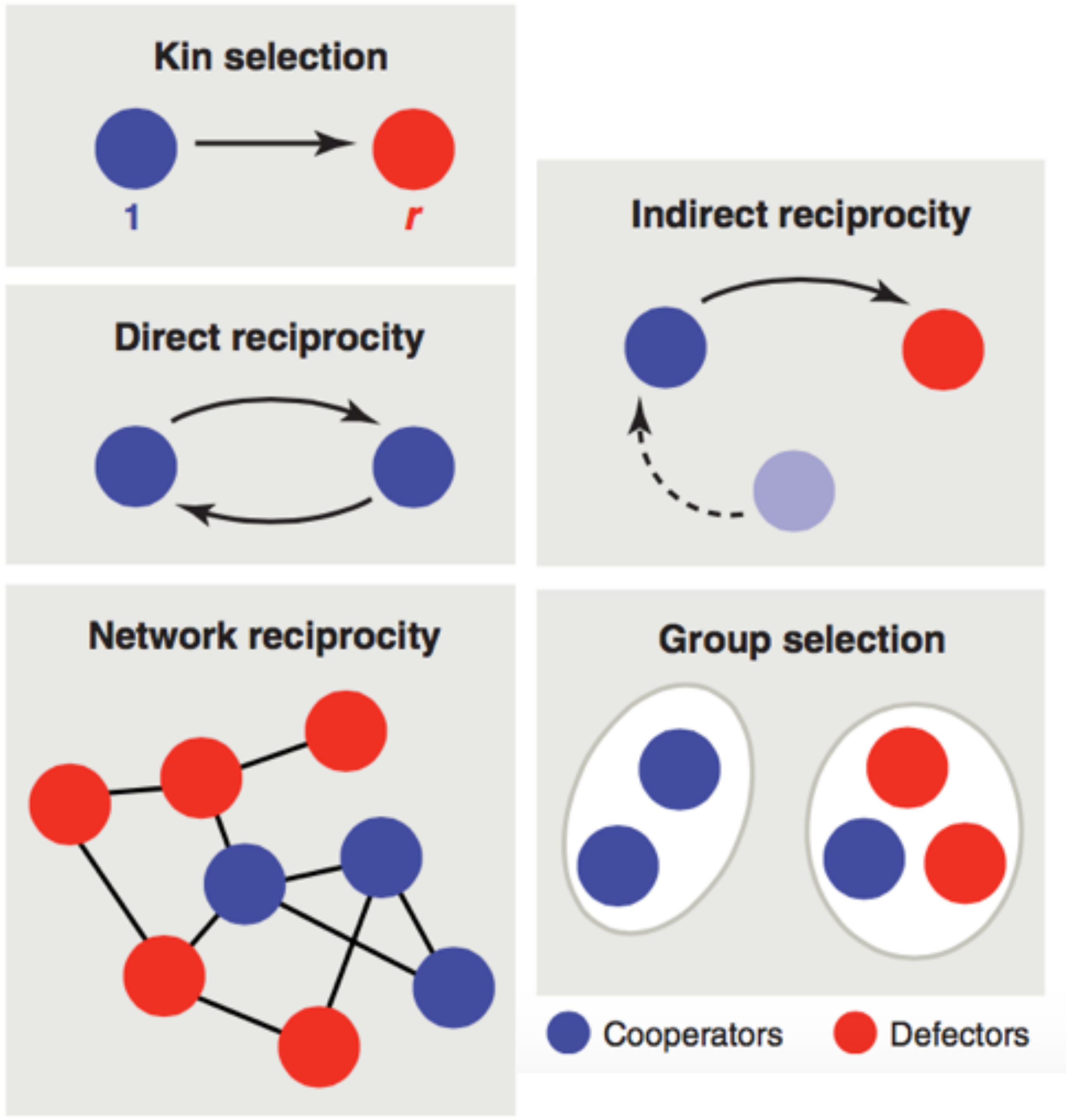
$$w > \frac{b-(b-c)}{b-0}$$

$$w > \frac{c}{b}$$

$$\boxed{wb > c}$$

Roads to Cooperation

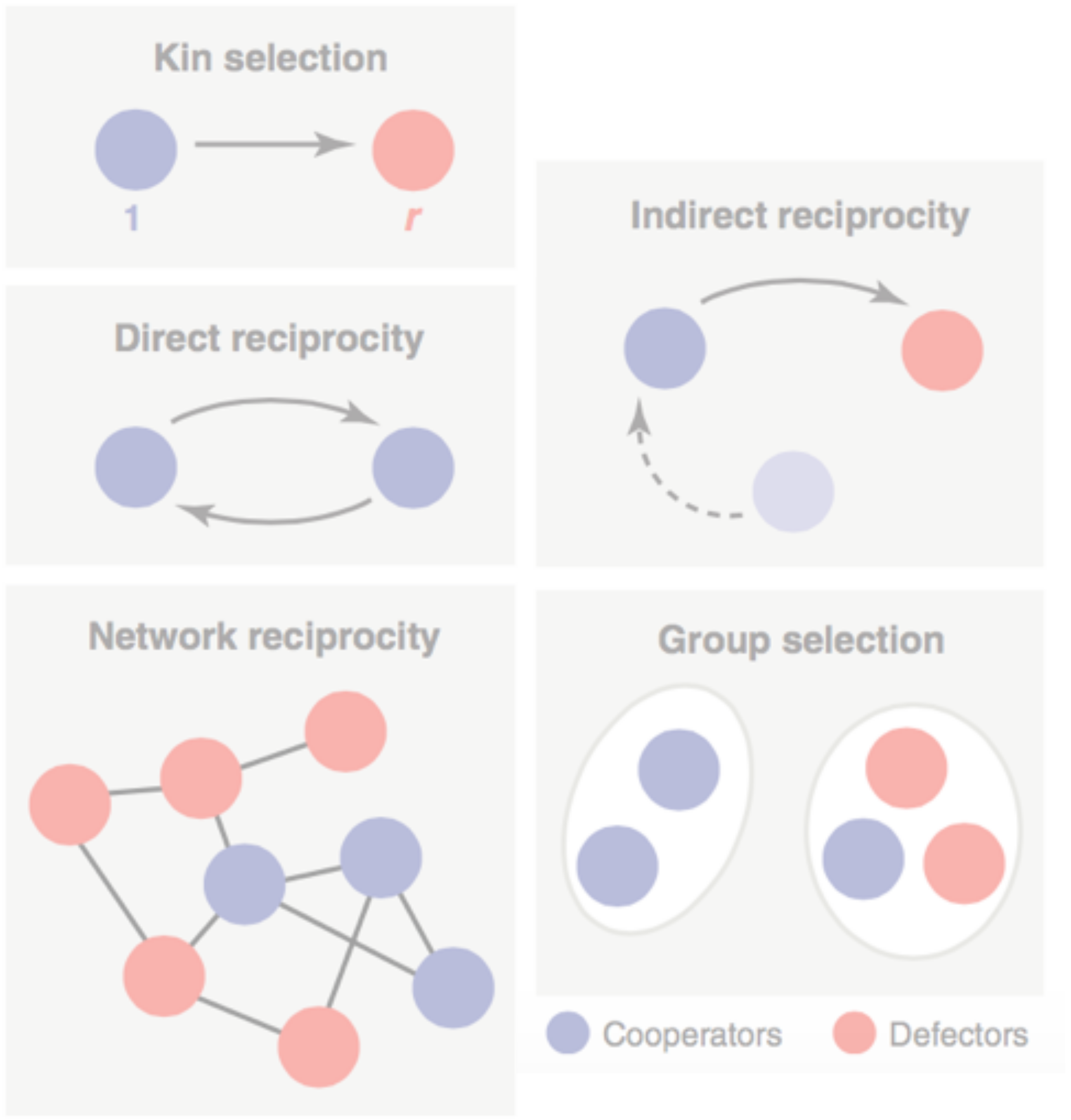
“Five rules for the evolution of cooperation”



Nowak MA (2006) *Science*

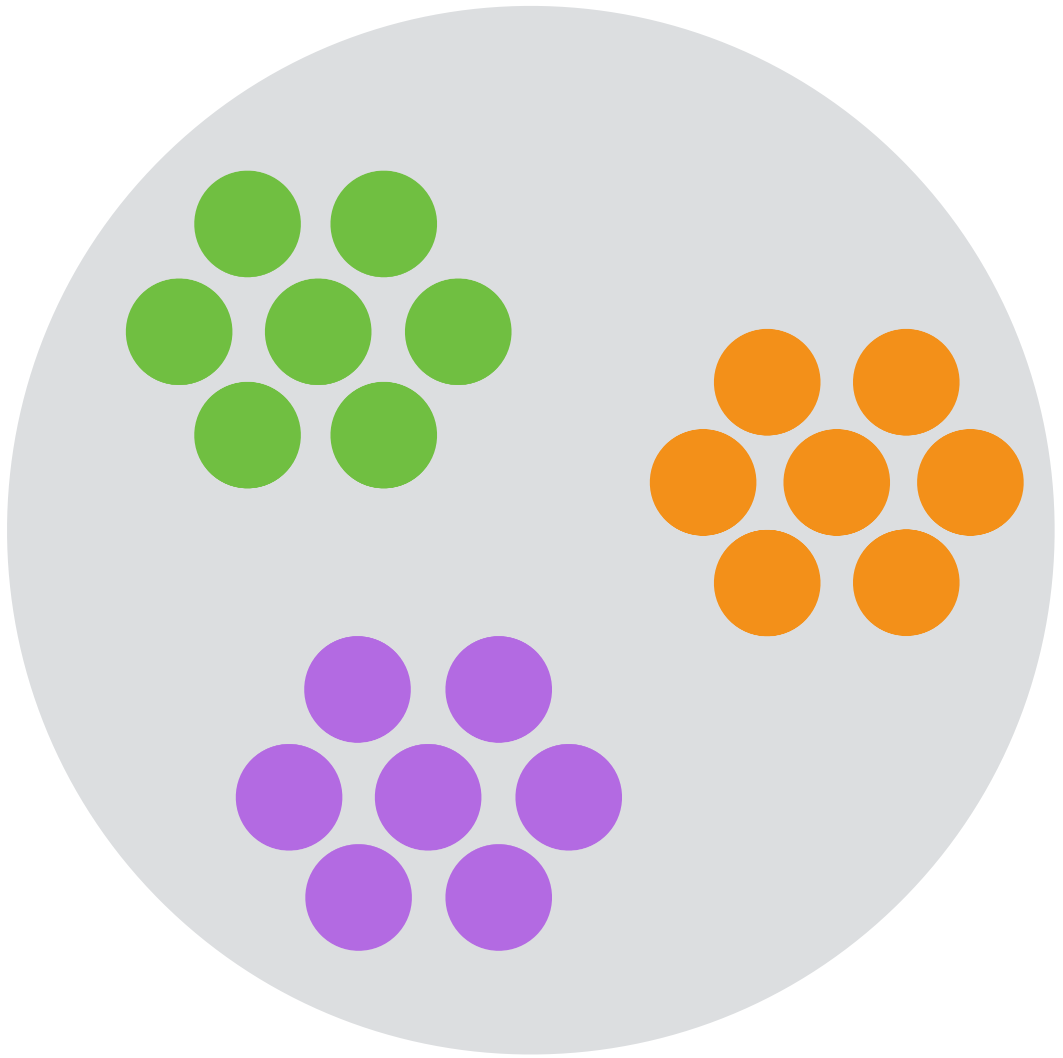
Roads to Cooperation

"Five rules for the evolution of cooperation"

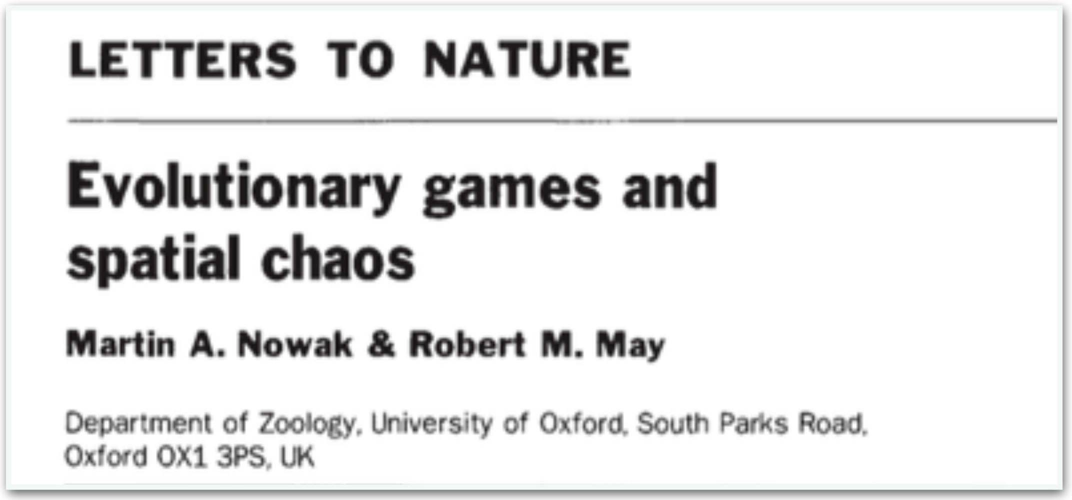


One rule for the evolution of cooperation:

Positive assortment



Structure Instead of Strategy


LETTERS TO NATURE

Evolutionary games and spatial chaos

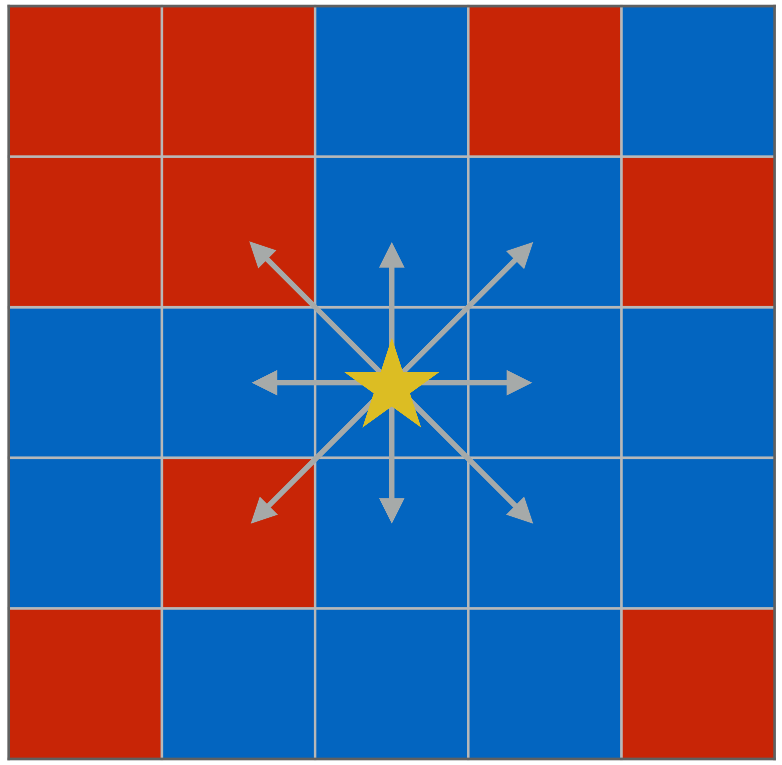
Martin A. Nowak & Robert M. May

Department of Zoology, University of Oxford, South Parks Road, Oxford OX1 3PS, UK

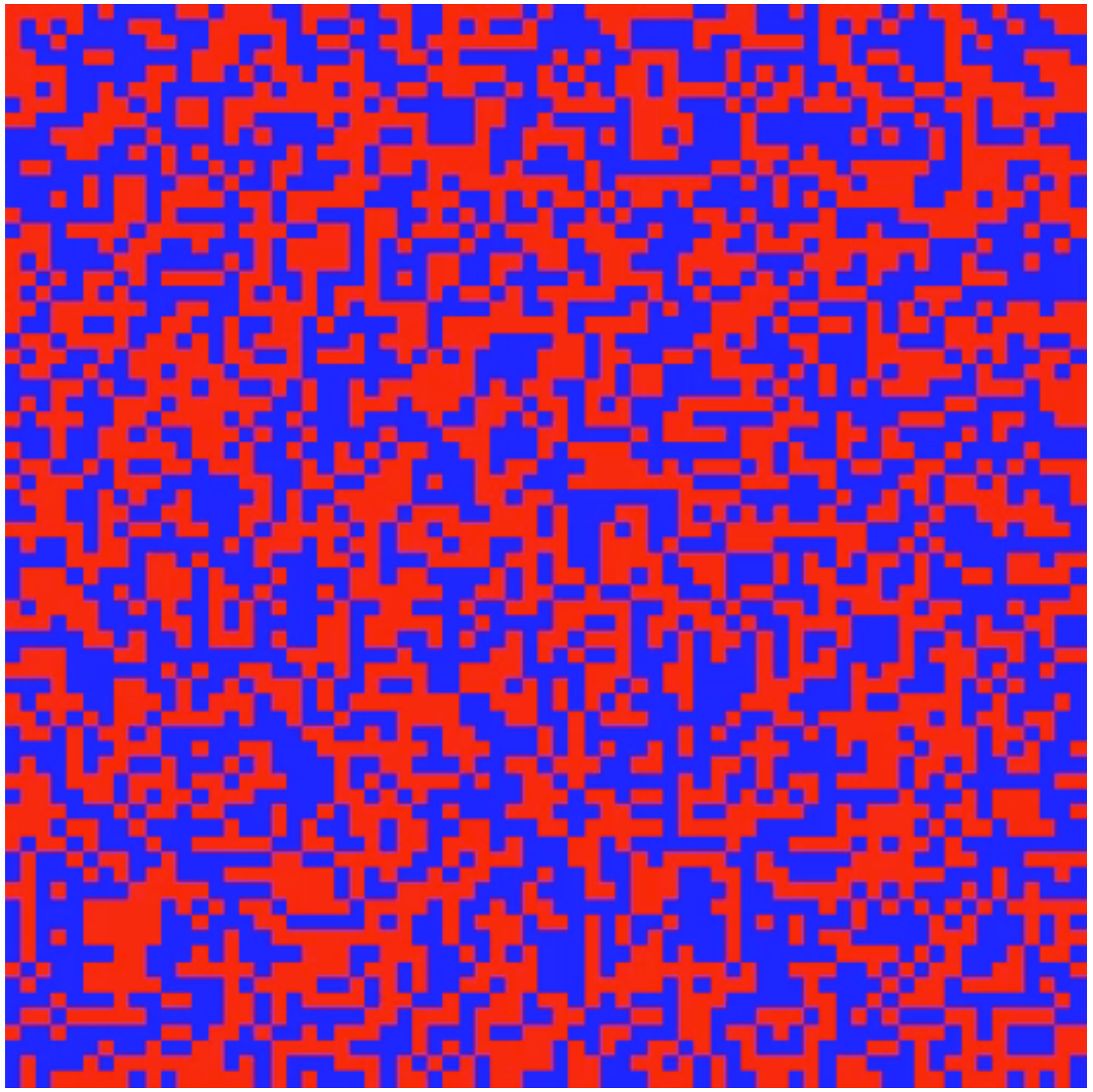
- Pure strategies only (**C** or **D**)
- Rigid spatial structure: 2D square lattice with Moore neighborhoods
- **Game play:** All agents spatial neighbors
- **Evolution:** After each round, each agent copies the strategy of its neighbor with the highest total payoff (itself included)

$$\begin{array}{c|c|c}
 & \text{Player 2: Cooperate} & \text{Player 2: Defect} \\
\hline
\text{Player 1: Cooperate} & R = 1 & S = 0 \\
\text{Player 1: Defect} & T = b & P = 0
\end{array}$$

$b > 1$



Only one free game parameter: *b*



t $\quad t-1$

🟦 C from C

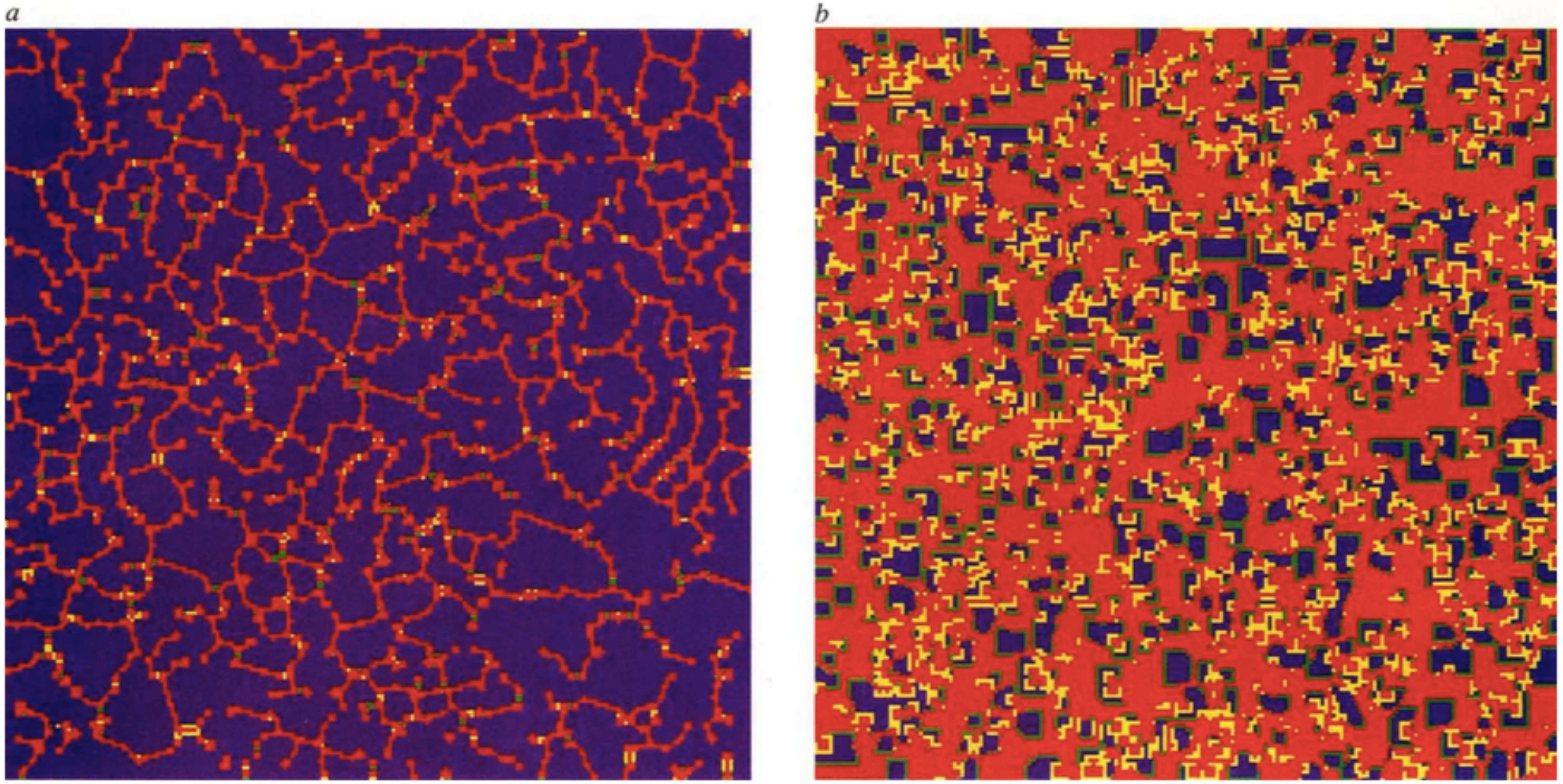
🟥 D from D

🟩 C from D

🟨 D from C

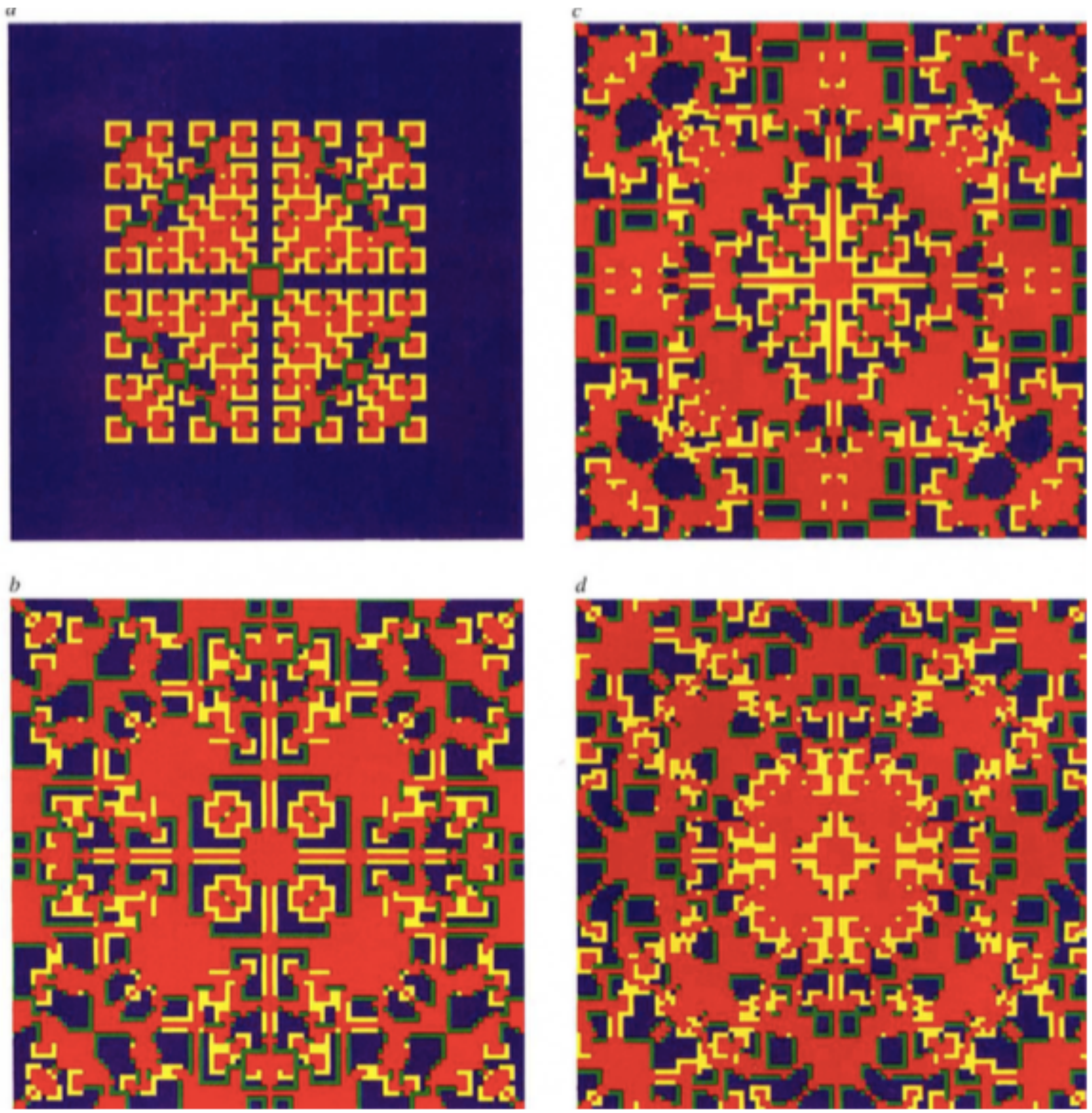
Evolutionary games and spatial chaos

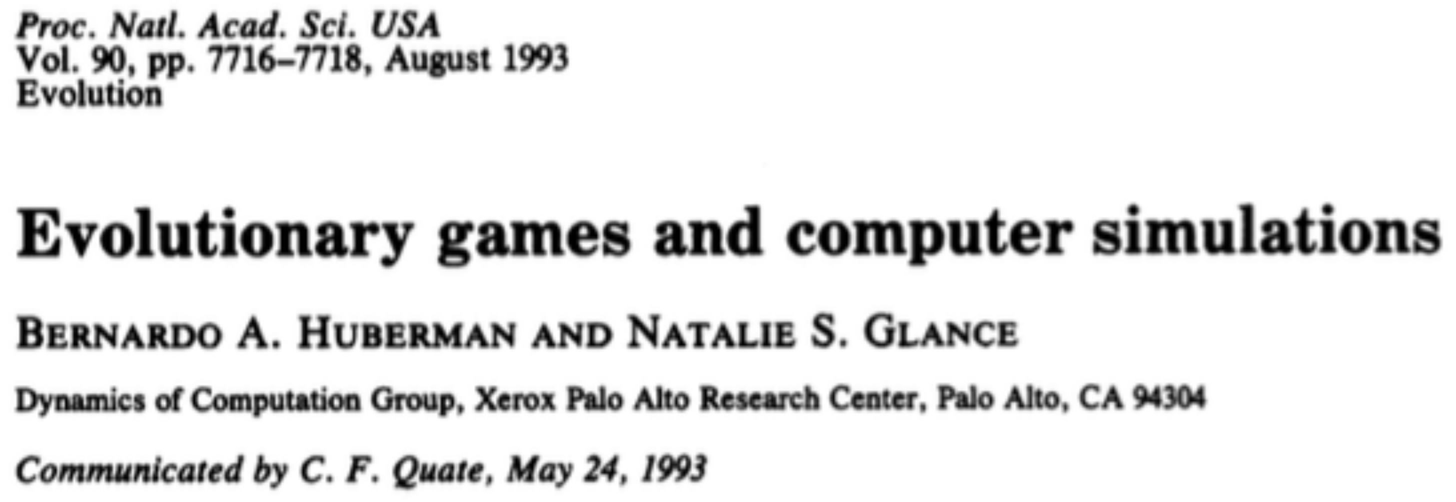
Martin A. Nowak & Robert M. May



Evolutionary games and spatial chaos

Martin A. Nowak & Robert M. May





Proc. Natl. Acad. Sci. USA
Vol. 90, pp. 7716–7718, August 1993
Evolution

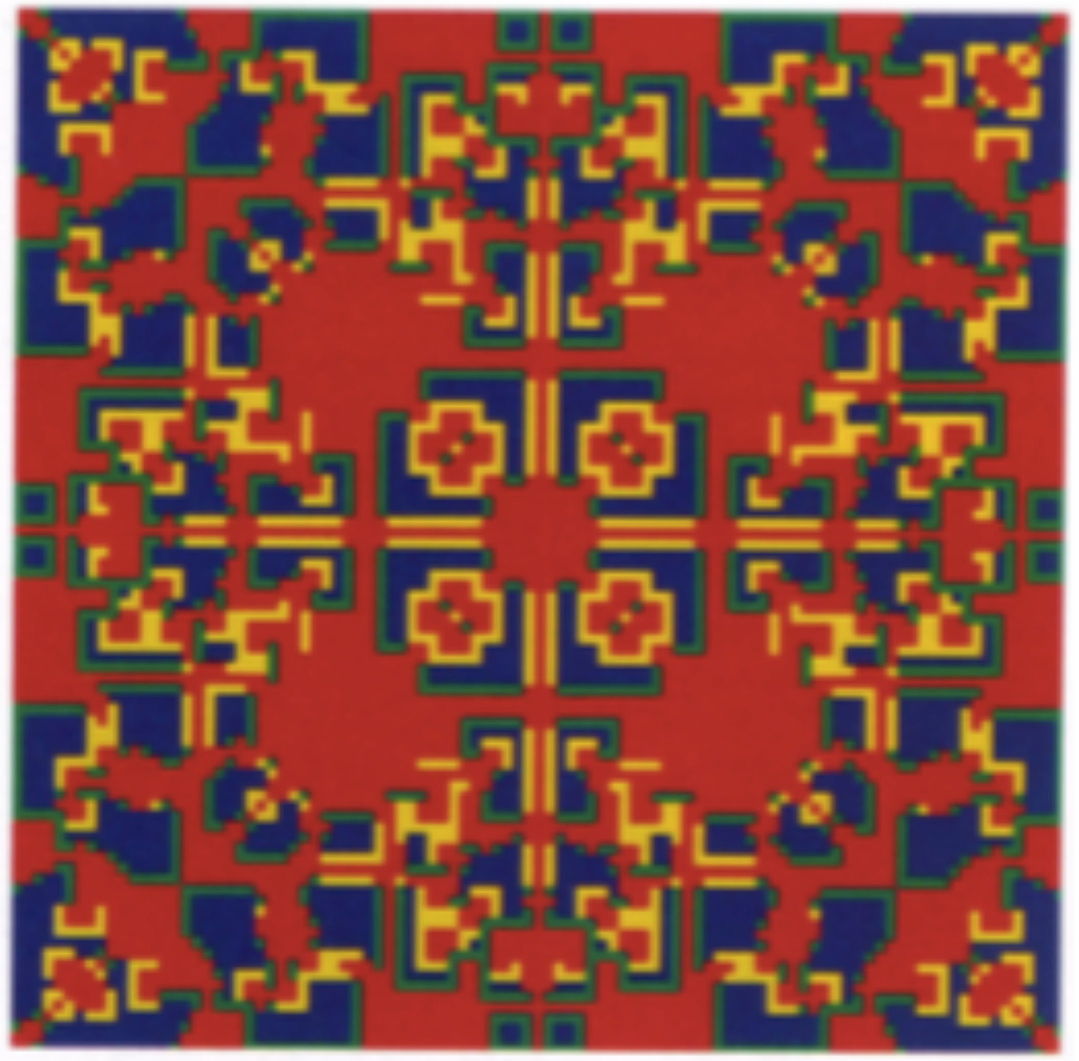
Evolutionary games and computer simulations

Bernardo A. Huberman and Natalie S. Glance

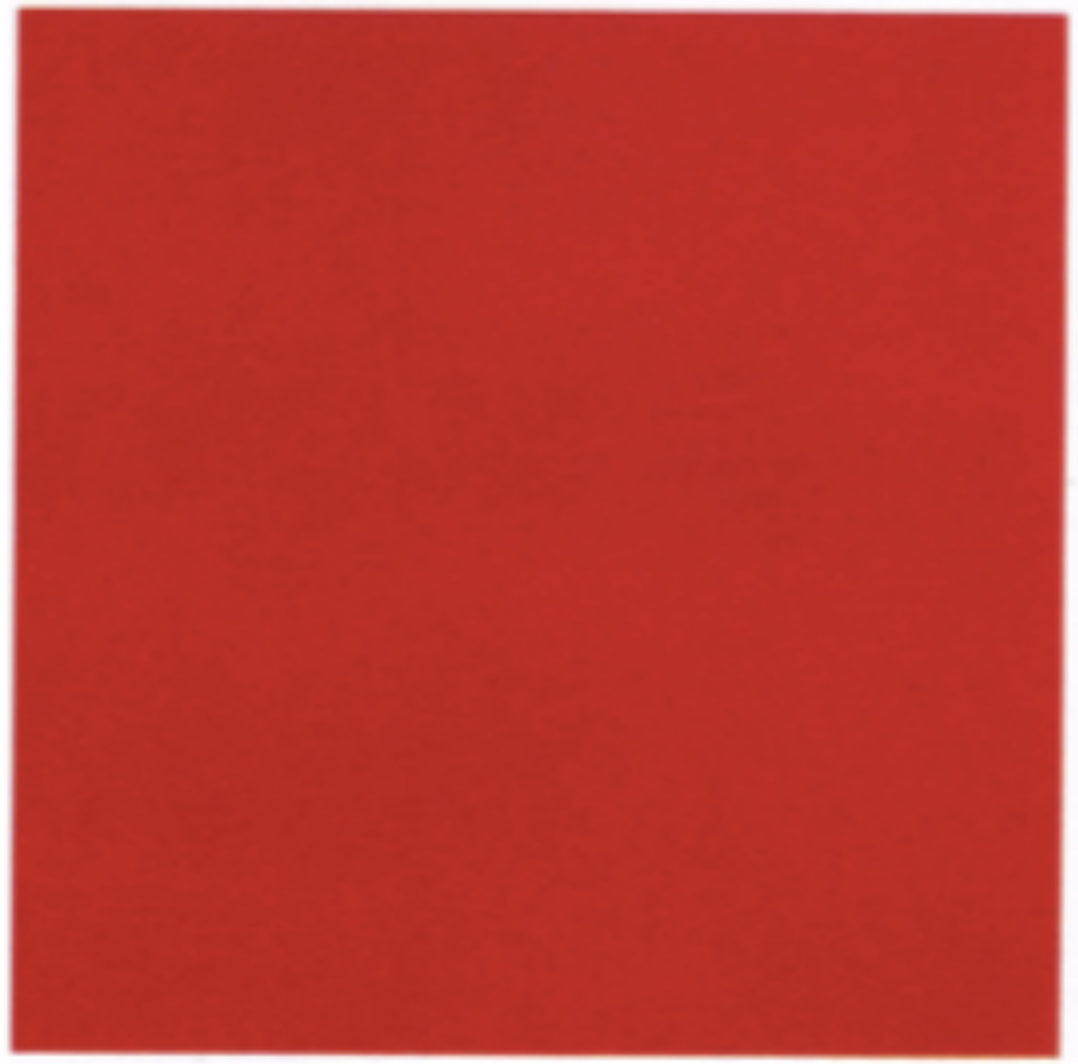
Dynamics of Computation Group, Xerox Palo Alto Research Center, Palo Alto, CA 94304

Communicated by C. F. Quate, May 24, 1993

synchronous updating



asynchronous updating



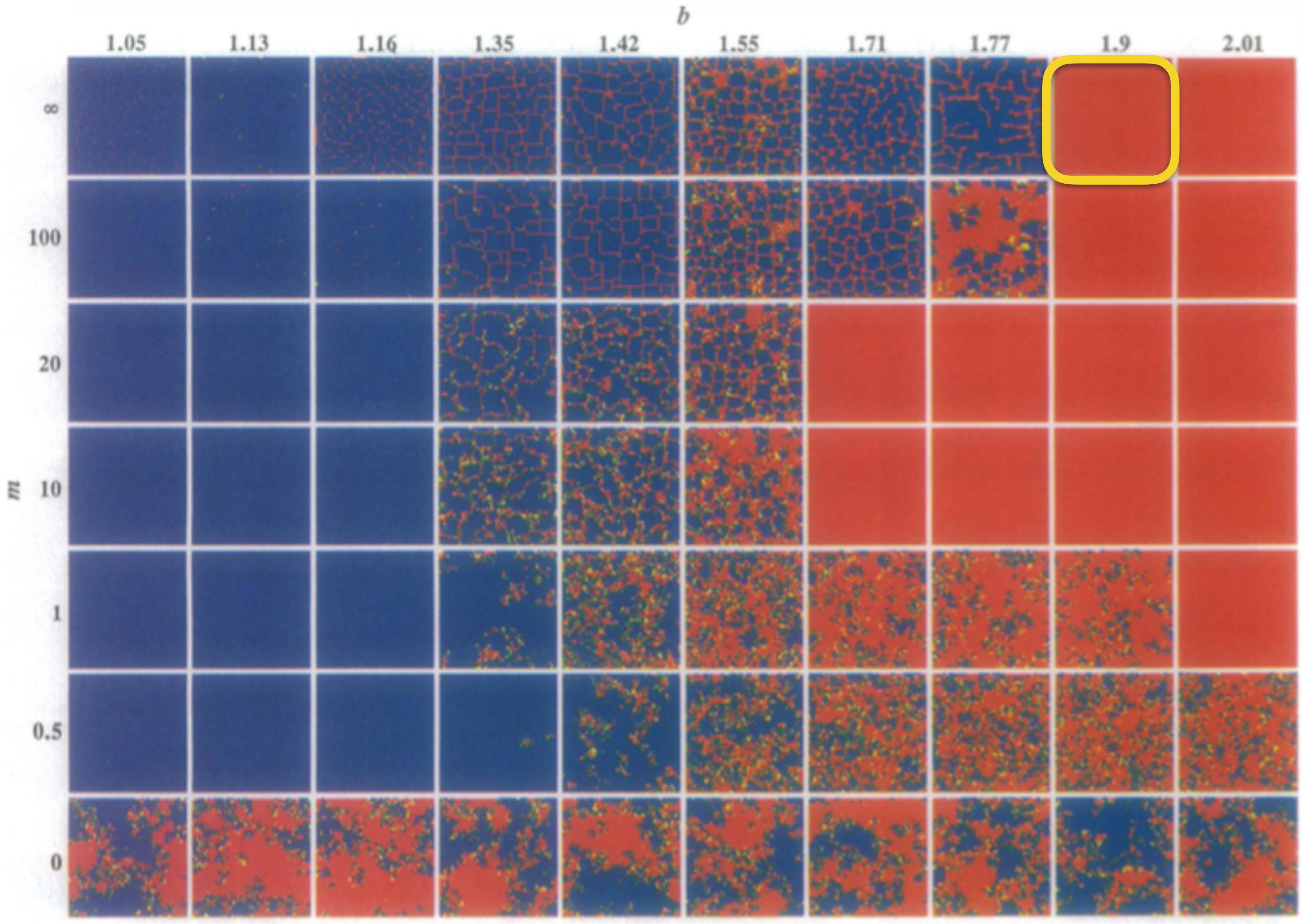
asynchronous updating

Proc. Natl. Acad. Sci. USA
Vol. 91, pp. 4877–4881, May 1994
Evolution

Spatial games and the maintenance of cooperation

Martin A. Nowak, Sebastian Bonhoeffer, and Robert M. May
Department of Zoology, University of Oxford, South Parks Road, OX1 3PS, Oxford, United Kingdom

Contributed by Robert M. May, January 28, 1994



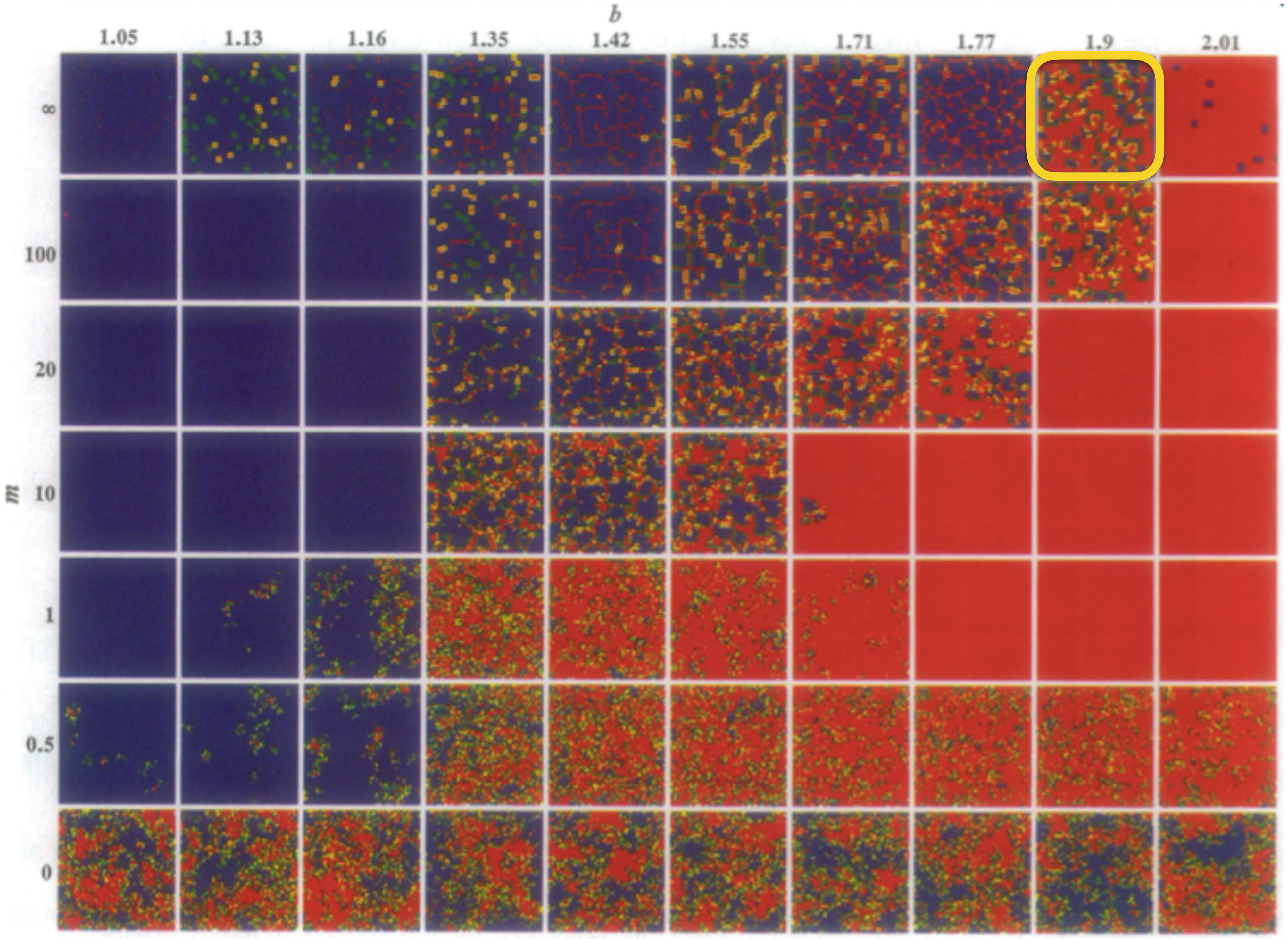
Proc. Natl. Acad. Sci. USA
Vol. 91, pp. 4877–4881, May 1994
Evolution

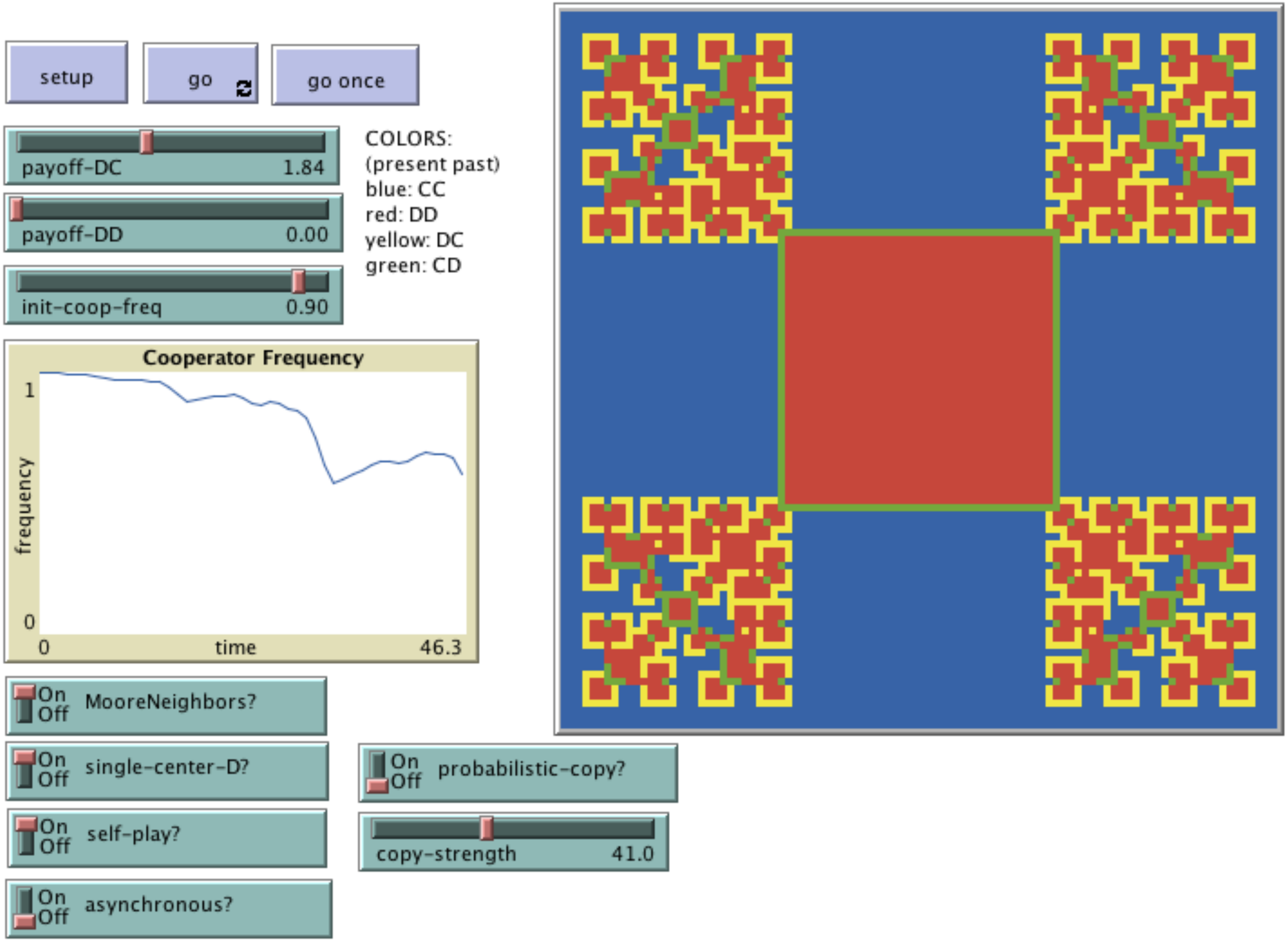
synchronous updating

Spatial games and the maintenance of cooperation

MARTIN A. NOWAK, SEBASTIAN BONHOEFFER, AND ROBERT M. MAY
Department of Zoology, University of Oxford, South Parks Road, OX1 3PS, Oxford, United Kingdom

Contributed by Robert M. May, January 28, 1994





payoff-DC 1.84

payoff-DD 0.00

init-coop-freq 0.90

COLORS:
(present past)
blue: CC
red: DD
yellow: DC
green: CD

MooreNeighbors? [✓ On] Off

single-center-D? [✓ On] Off

self-play? [✓ On] Off

asynchronous? On [✓ Off]

probabilistic-copy? On [✓ Off]

copy-strength 41.0

```
patches-own[
  current-strategy ;;let 0 = defect, 1 = cooperate
  previous-strategy
  payoff
  co-players
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; SETUP PROCEDURE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup
  clear-all
  ;;assign strategies to turtles
  ifelse single-center-D?
  [setup-singleD]
  [
    ask patches [
      ifelse random-float 1 < init-coop-freq
      [
        set pcolor blue
        set current-strategy 1
        set previous-strategy 1
      ]
      [
        set pcolor red
        set current-strategy 0
        set previous-strategy 0
      ]
      set payoff 0
    ]
  ]
  ;;assign neighbors to 4 or 8 nearest neighbors
  ask patches [
    ifelse MooreNeighbors?
      [ set co-players neighbors]
      [ set co-players neighbors4 ]
  ]
  reset-ticks
end
```

```
to setup-singleD
  ask patches[
    set pcolor blue
    set current-strategy 1
    set previous-strategy 1
    set payoff 0
  ]
  ask patch 0 0 [
    set pcolor red
    set current-strategy 0
    set previous-strategy 0
  ]
end
```

```
to go
  ask patches [
    set payoff 0 ;;reset payoffs
    set-prev-strategy ;;record the strategy at the beginning of the round
  ]
  ;;calculate payoffs
  ifelse asynchronous?[
    ask patches [
      set-prev-strategy
      play-game ;;play the game and get payoffs
      ask co-players [
        play-game]
      imitate
      recolor
    ]
  ]
  [
    ;imitate more successful strategies synchronously
    ask patches [
      play-game ;;play the game and get payoffs
    ]
    ask patches [
      imitate ;;copy more successful strategies.
    ]
    ask patches [ recolor ] ;; recolor to match strategies
  ]
  tick
end
```

```
;;agents note their strategy before play.
to set-prev-strategy
  ifelse (pcolor = blue or pcolor = green)
  [set previous-strategy 1]
  [set previous-strategy 0]
end


;;the agent plays the game and calculates its payoff.
to play-game
  ; count neighbor strategies
  let neighbors-C count co-players with [pcolor = blue or pcolor = green]
  let neighbors-D count co-players with [pcolor = red or pcolor = yellow]

  ;if I'm a cooperator
  if (pcolor = blue or pcolor = green)
  [ set payoff (neighbors-C * 1) + (neighbors-D * 0) ]

  ;if I'm a defector
  if (pcolor = red or pcolor = yellow)
  [ set payoff (neighbors-C * payoff-DC) + (neighbors-D * payoff-DD) ]

  if self-play? ;;if I also play with myself
  [
    ifelse (pcolor = blue or pcolor = green)
    [set payoff payoff + 1]
    [set payoff payoff + payoff-DD]
  ]
end
```

```
;;copy your most successful neighbor
to imitate
  ifelse not probabilistic-copy?
  [
    let best-neighbor max-one-of co-players [payoff] ;;who's my neighbor with the highest payoff
    if ([payoff] of best-neighbor) > payoff[ ;;if their payoff is better than mine, imitate them
      copy-strategy best-neighbor
    ]
  ]
  [
    prob-imitate
  ]
end

to copy-strategy [bn]
  let pcolor-best [pcolor] of bn
  ifelse (pcolor-best = blue or pcolor-best = green)
  [set current-strategy 1]
  [set current-strategy 0]
end


to prob-imitate
  ;;probabilistically based on neighbor payoffs.
  let num ((payoff ^ copy-strength) * current-strategy)
  let denom (payoff ^ copy-strength)
  ask co-players [
    set num (num + ((payoff ^ copy-strength) * current-strategy))
    set denom (denom +  (payoff ^ copy-strength))
  ]

  ifelse (num = 0 and denom = 0) ;;avoid divide by zero error
  [set current-strategy 0]
  [
    ifelse random-float 1 < (num / denom)
    [ set current-strategy 1]
    [ set current-strategy 0]
  ]
end
```

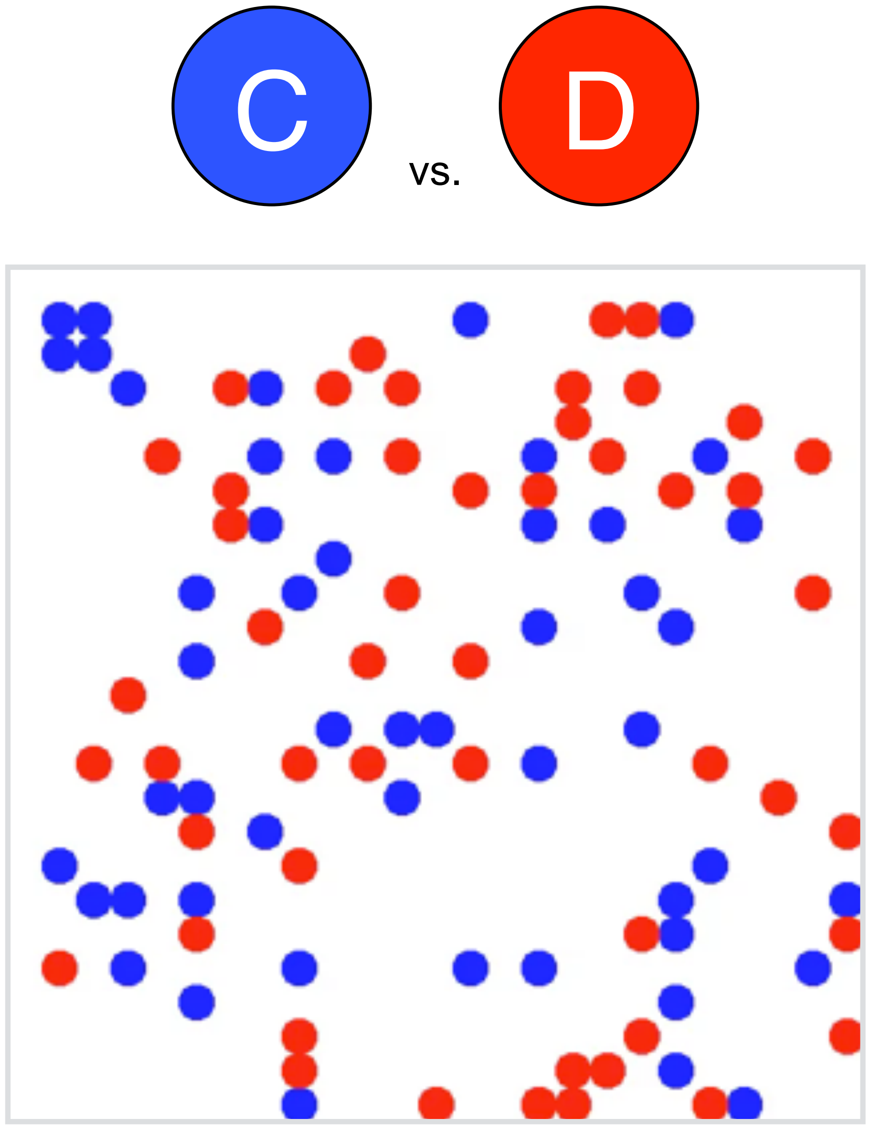
```
to recolor
    if current-strategy = 0
    [
      ifelse previous-strategy = 0
      [set pcolor red]
      [set pcolor yellow]
    ]
    if current-strategy = 1
    [
      ifelse previous-strategy = 1
      [set pcolor blue]
      [set pcolor green]
    ]
end
```

Models of cooperation used to study dynamics related to

- Reciprocity
- Reputation
- Punishment
- Monitoring
- Group competition
- Social markers
- Mobility
- Network structure
- Etc.

Moving toward greater biologically realism

- Cellular automata models are elegant, but highly constrained
- Some minimal properties of biologically realistic agents
  - Embodiment
  - Mobility
  - Costly reproduction
  - Long lifespan relative to interaction time



Questions about implementation

- Some minimal properties of biologically realistic agents
  - **Embodiment**
    - How does embodiment affect interactions?
  - **Mobility**
    - How do they move?
  - **Costly reproduction**
    - Do offspring disperse?
  - **Long lifespan relative to interaction time**
    - What affects their survival?

```
turtles-own[ ;;turtle variables
 energy ;turtles accumulated energy
 played? ;has the turtle played this round?
]

globals[ ;;global parameters
  reproduce-threshold
  reproduce-cost
  max-energy
]
```

```
to setup
  clear-all
  ask patches [set pcolor white]
  initialize-variables
  create-agents num-agents init-coop-freq
  reset-ticks
end

to initialize-variables
  ;;initialize all variables
  set reproduce-threshold 100
  set reproduce-cost 50
  set max-energy 150
end

to create-agents [n coop-freq]
  create-turtles n[
    ifelse random-float 1 < coop-freq
    [set color blue]
    [set color red]
    set shape "circle"
    set size 1.0
    move-to one-of patches with [not any? other turtles-here]
    set energy (random 50 + 1)
  ]
end
```

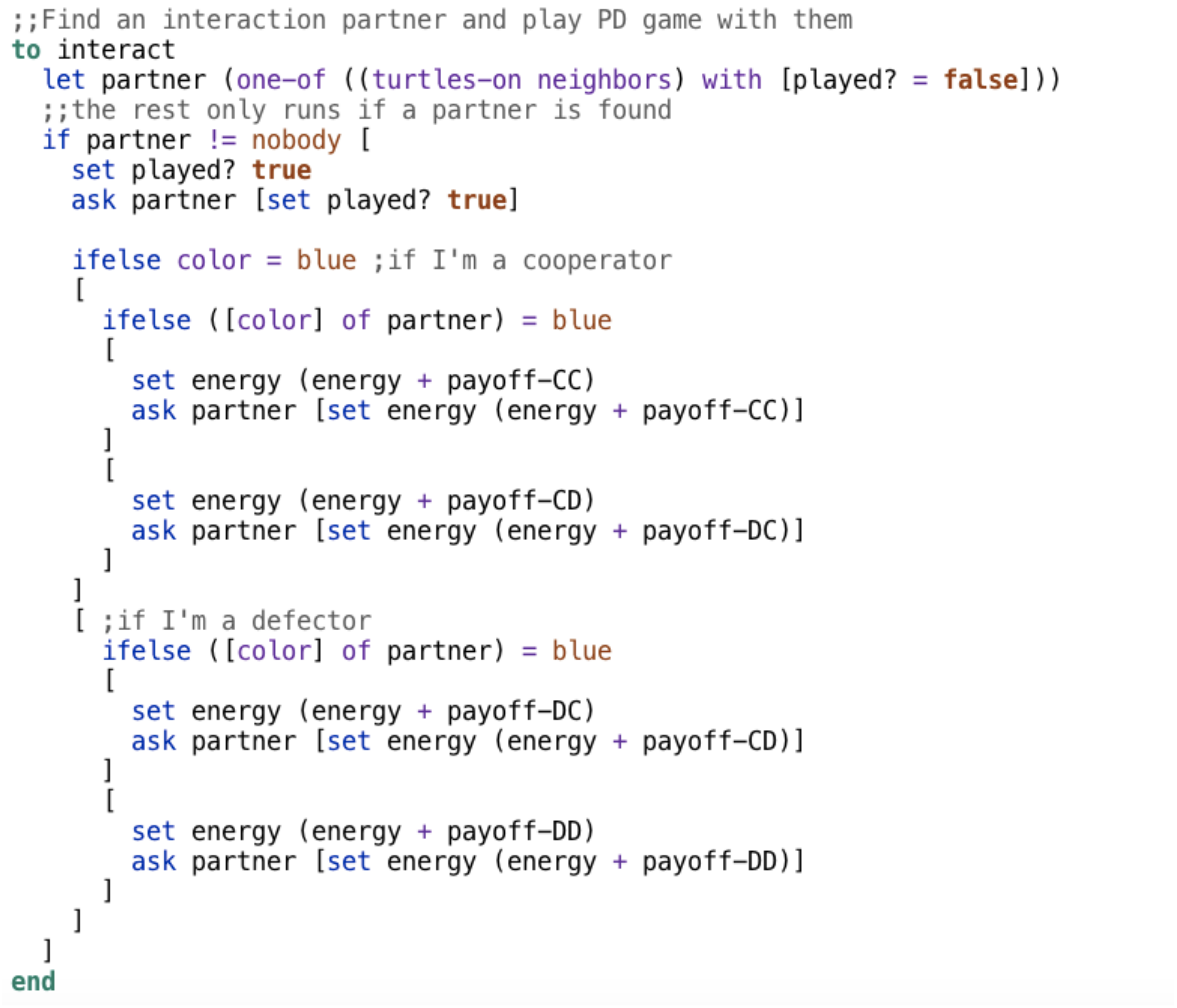
```
to go
  if not any? turtles [ stop ] ;stop the simulation if everyone is dead
  if all? turtles [color = blue] or all? turtles [color = red] [stop]
  ask turtles [set played? false] ;;reset "have I played this tick?"
  ask turtles[
    if energy <= 0 [die] ;;die if not enough energy

    ;;play PD game and collect payoffs
    if (not played?)[ interact ] ;;find coplayer

    ;;reproduce
    if (energy >= reproduce-threshold)
      [reproduce]

    cull-herd ;;if population over carrying capacity, have agents lose energy

    ;;movement
    if (not played?) [ move ];;if didn't find coplayer, move
  ]
  tick
end
```



```
;;Find an interaction partner and play PD game with them
to interact
  let partner (one-of ((turtles-on neighbors) with [played? = false]))
  ;;the rest only runs if a partner is found
  if partner != nobody [
    set played? true
    ask partner [set played? true]

    ifelse color = blue ;if I'm a cooperator
    [
      ifelse ([color] of partner) = blue
      [
        set energy (energy + payoff-CC)
        ask partner [set energy (energy + payoff-CC)]
      ]
      [
        set energy (energy + payoff-CD)
        ask partner [set energy (energy + payoff-DC)]
      ]
    ]
    [ ;if I'm a defector
      ifelse ([color] of partner) = blue
      [
        set energy (energy + payoff-DC)
        ask partner [set energy (energy + payoff-CD)]
      ]
      [
        set energy (energy + payoff-DD)
        ask partner [set energy (energy + payoff-DD)]
      ]
    ]
  ]
end
```

```
;;attempt to place a duplicate agent in a neighboring cell
to reproduce
  let new-patch (one-of neighbors with [ not any? turtles-here ])
  if (new-patch != nobody)
  [
    set energy (energy - reproduce-cost)
    hatch 1 [
      set energy reproduce-cost
      move-to new-patch
      set played? false
  ]
  ]
end


;;random agents lose energy if population exceeds carrying capacity
to cull-herd
  if (count turtles > carrying-capacity)
  [
    ask one-of turtles [
      set energy ( energy - 10 )
    ]
  ]
end


;;The agent chooses a neighboring patch to move to, does so if that patch is unoccupied.
to move
  let new-patch (one-of neighbors)
  if (not any? turtles-on new-patch)
  [
    move-to new-patch
  ]
end
```

- Remainder of today:
  - Attempt today’s Modeling Challenge
- Tomorrow:
  - Model validity and empirical data
  - Networks
  - Coda