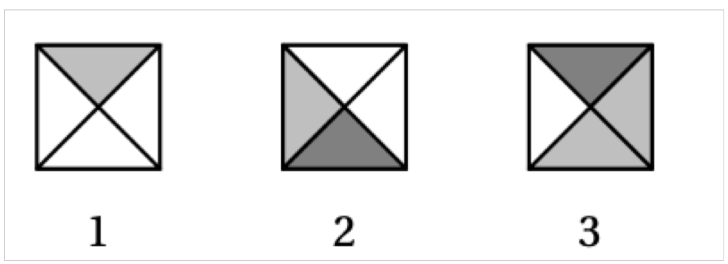
Wang Tilings continued

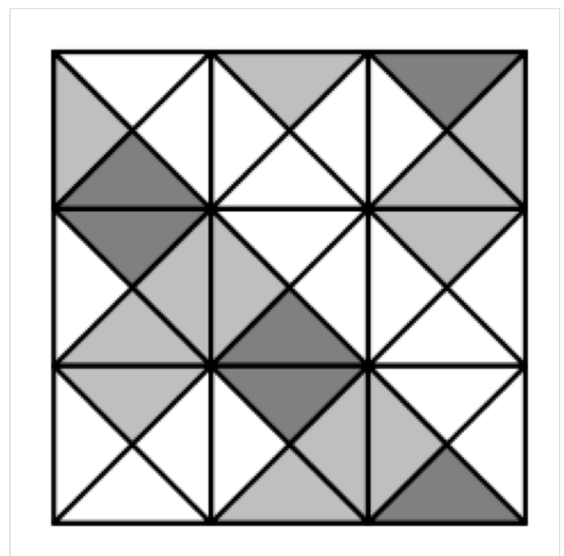
A fun problem to help review some background material: TM, reductions, decidability, undecidability, and connections with molecular programming

Wang tiling (1961)

- **Instance**: A finite set T of Wang tiles: square tiles, each with a fixed orientation, edges labeled with colours.

- **Problem**: Can the infinite plane $\mathbb{Z}^2$ be tiled with tiles from T? Colours on adjacent tile edges must match, and tiles must be in their fixed orientation (no rotations or reflections). A tile can be used repeatedly.





Illustrations from: The Nature of Computation, Chris Moore

Wang tiling: simulating a TM

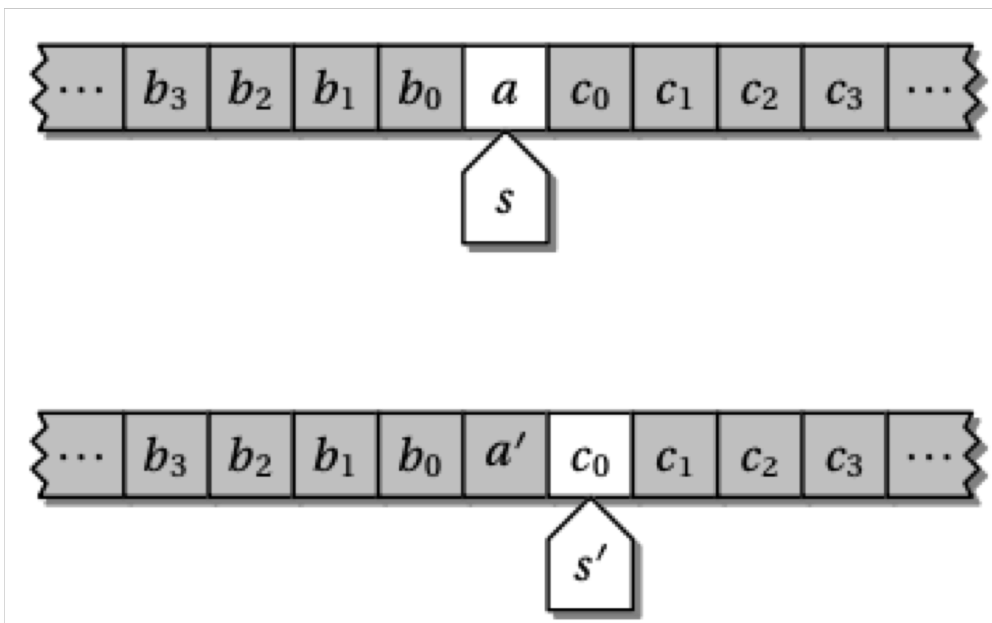
Robert Berger (1965) showed that the Wang Tiling problem is undecidable, thereby disproving Wang's conjecture. Let's construct ingredients of a reduction from the (undecidable) Blank Tape Halting problem to Wang Tiling.

The Blank Tape Halting problem is as follows:

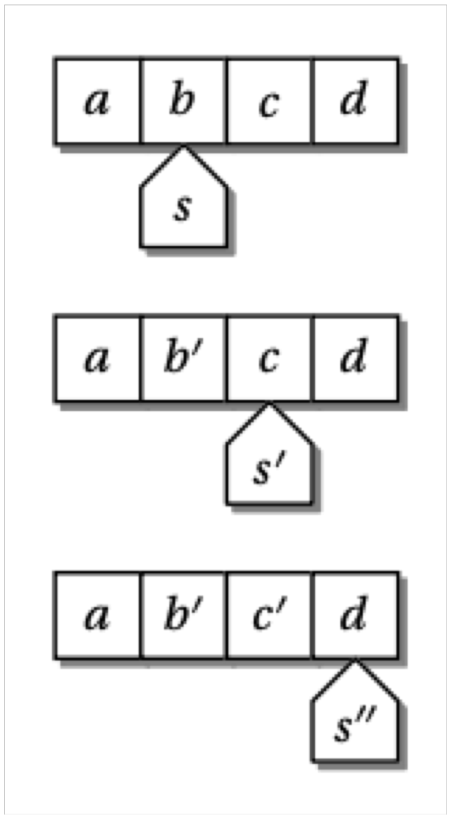
- **Instance**: A description of a Turing machine (TM), with a two-way infinite tape.
- **Problem**: does the TM halt when the tape is initially blank?

Wang tiling: simulating a TM

Consider the tape head position, plus the state, when the head moves to the *right*. Design tiles such that the next row of the Wang tiling is guaranteed to properly represent the configuration resulting from the TM transition. (A similar construction can handle transitions when the head moves to the left.)



Wang tiling: simulating a TM



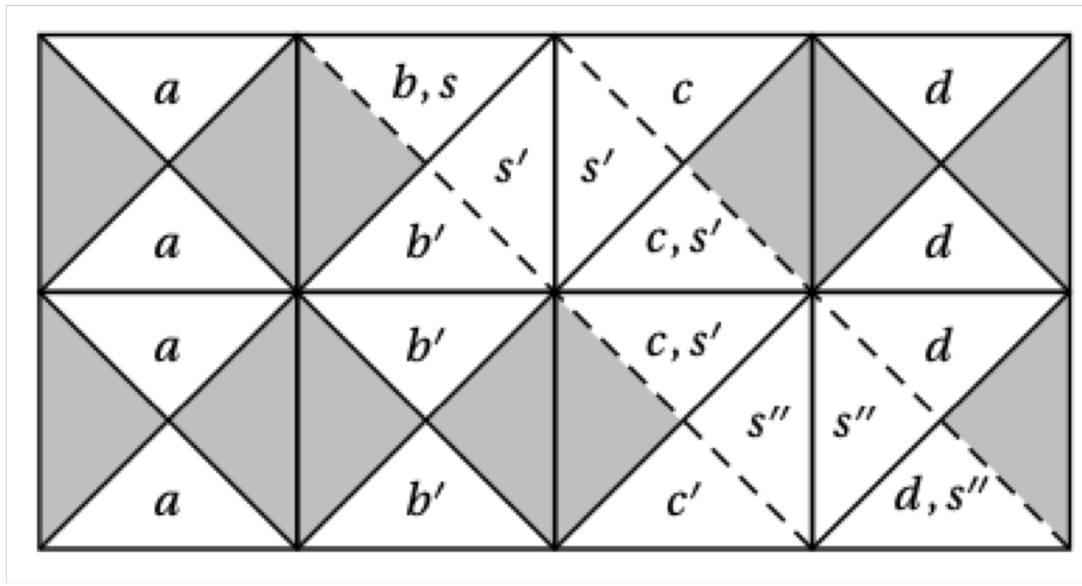


Illustration from: The Nature of Computation, Chris Moore

Wang tiling: simulating a TM

It is *not* the case that the set of tiles we've constructed is a yes-instance of Wang Tiling (i.e., can tile the infinite plane) if and only if the TM is a no-instance of Blank Tape Halting.

What can go wrong?

Wang tiling: simulating a TM

Can we salvage something from our work?

Can you describe a variant of the Wang Tiling problem, and modify the above reduction to show that your variant is undecidable?

Back to undecidability of Wang Tiling

Key ideas of Johnson's proof:

- Find an *aperiodic* tiling, and "layer on" our TM simulation tiles

An aperiodic tiling (Robinson, 1971)

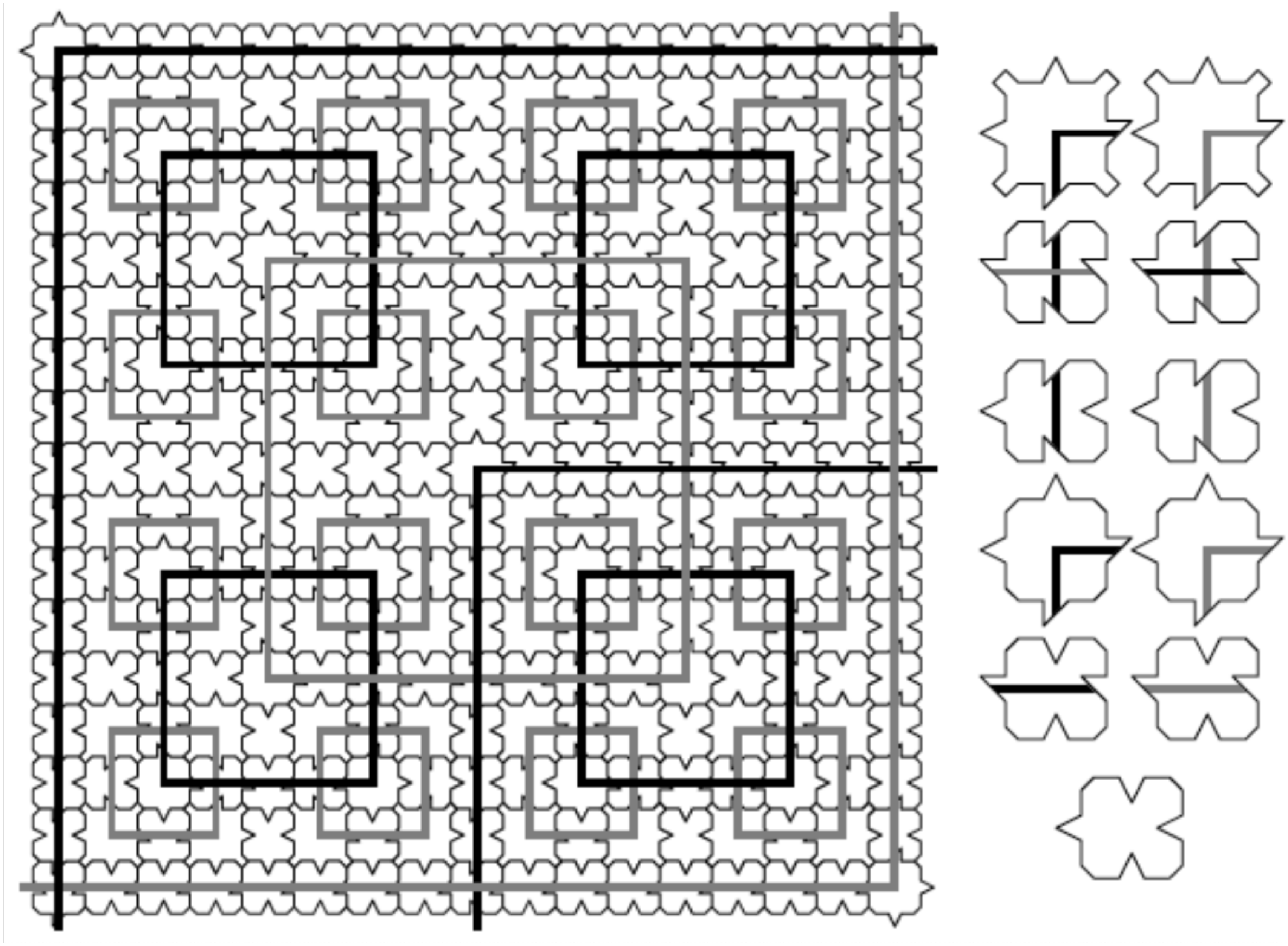


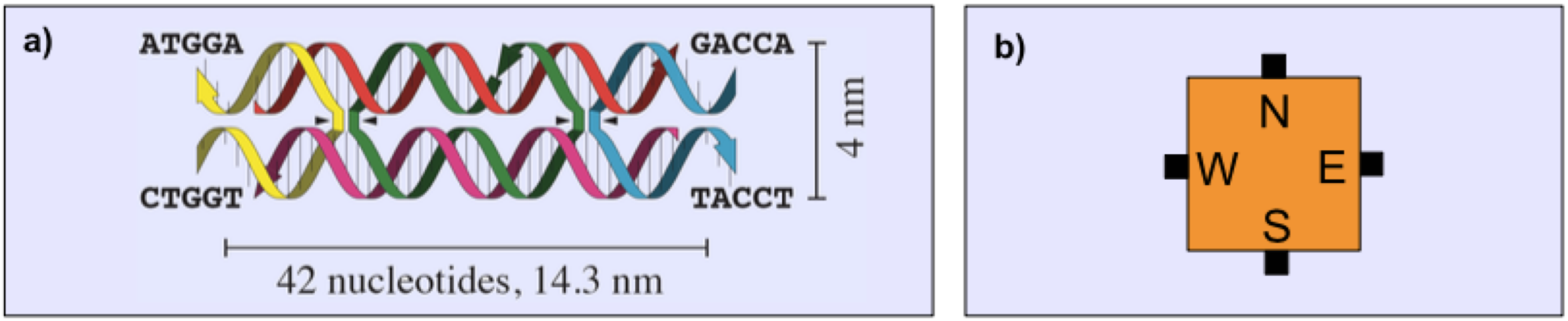
Illustration from: The Nature of Computation, Chris Moore

Back to undecidability of Wang Tiling

Key ideas of Johnson’s proof (simplified later):

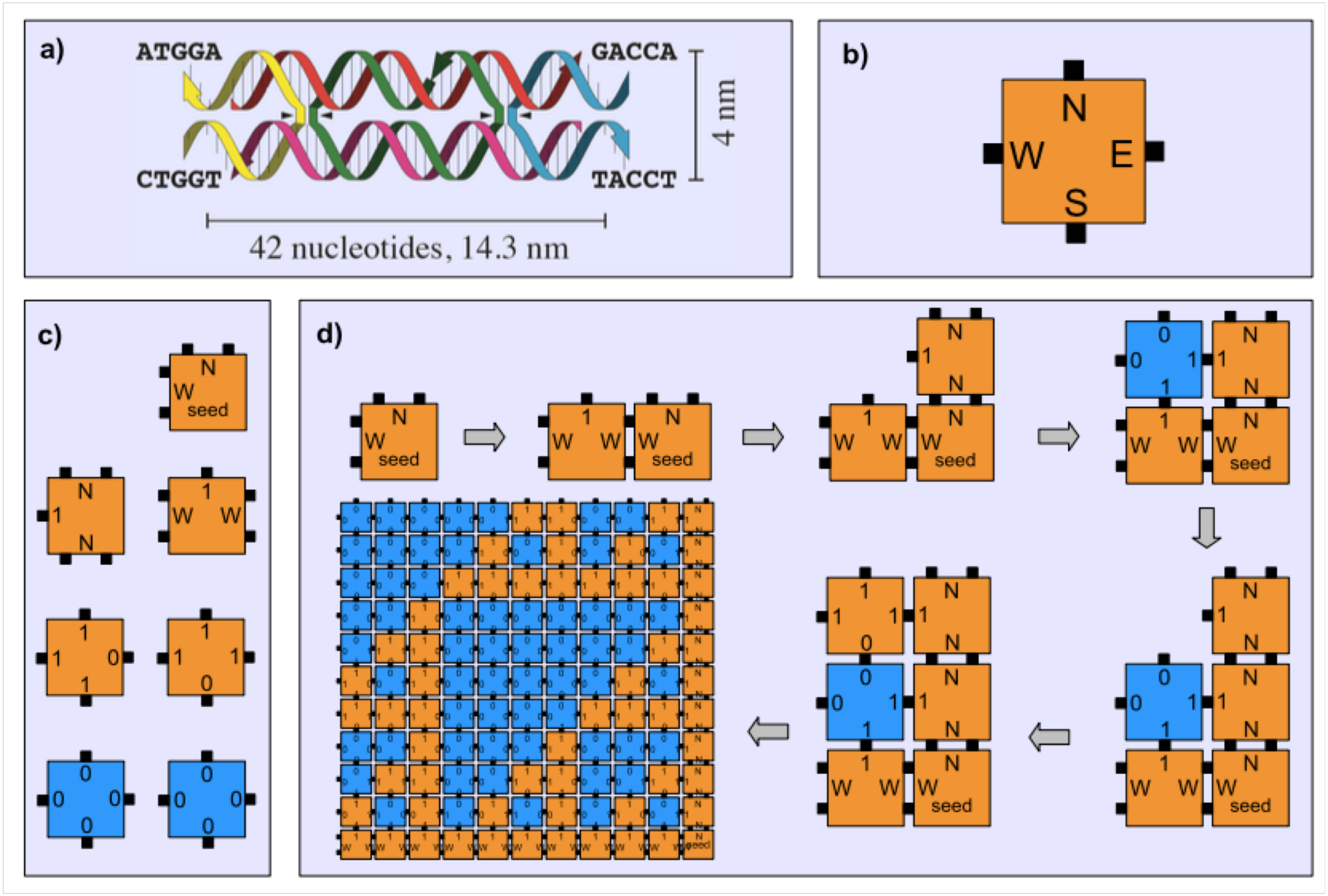
- Find an *aperiodic* tiling, and “layer on” our TM simulation tiles
- Using Robinson’s tiles, *only the initial state on blank symbo*l is layered on to the tile for the top left corner of a black square
- Ever-larger black squares model ever-larger TM computations
- There is an infinite tiling iff the TM does not halt

Wang Tiling application: DNA self-assembly



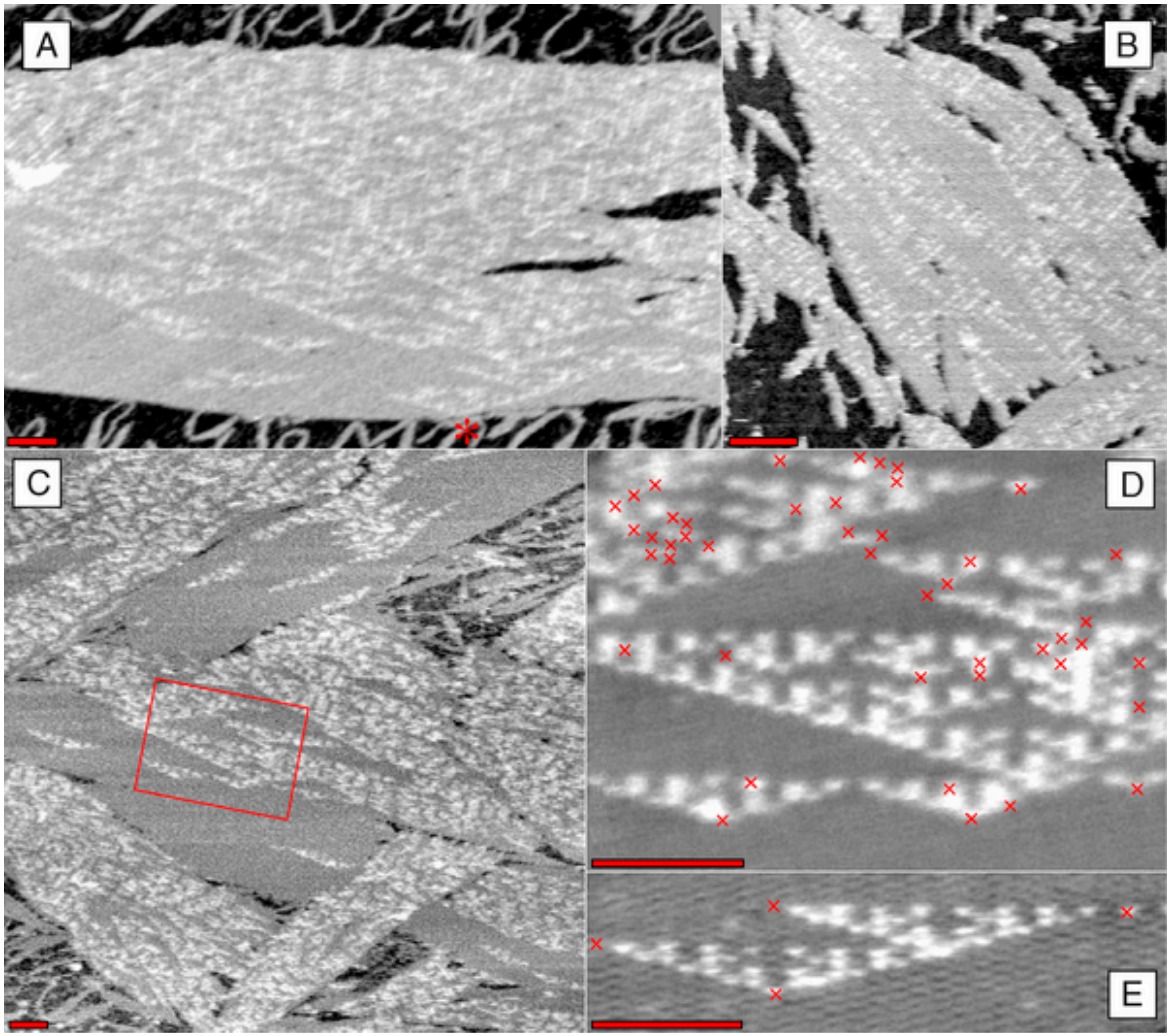
From: The Theory of Algorithmic Self-Assembly, Dave Doty

Wang Tiling application: DNA self-assembly



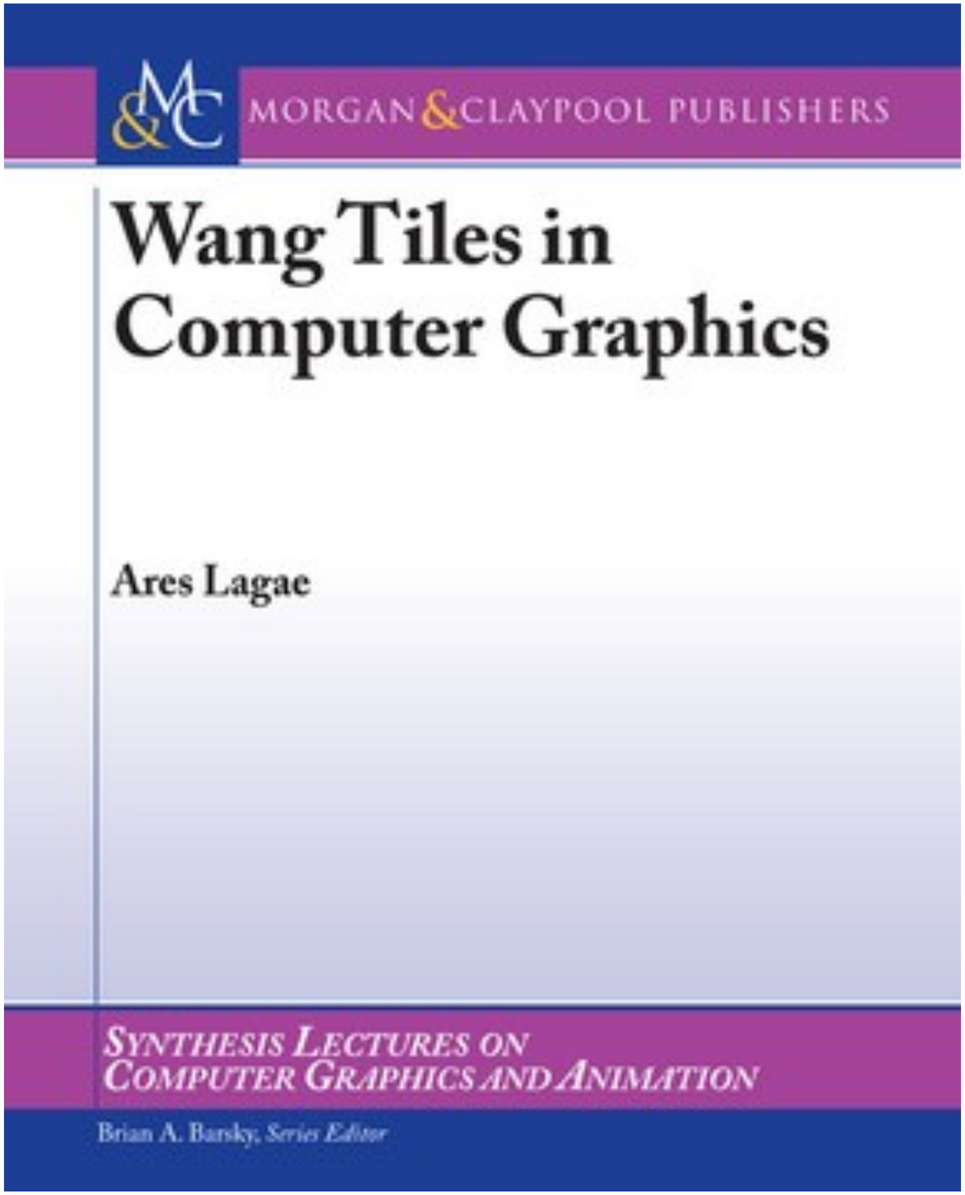
From: The Theory of Algorithmic Self-Assembly, Dave Doty

Wang Tiling application: DNA self-assembly



From: Algorithmic Self-Assembly of DNA Sierpinski Triangle, Rothemund et al., 2004

Another application of Wang Tilings



Another application of Wang Tilings

MORGAN & CLAYPOOL PUBLISHERS

Wang Tiles in Computer Graphics

Book Description

Many complex signals in computer graphics, such as point distributions and textures, cannot be efficiently synthesized and stored. This book presents tile-based methods based on Wang tiles and corner tiles to solve both these problems. Instead of synthesizing a complex signal when needed, the signal is synthesized beforehand over a small set of Wang tiles or corner tiles. Arbitrary large amounts of that signal can then efficiently be generated when needed by generating a stochastic tiling, and storing only a small set of tiles reduces storage requirements. A tile-based method for generating a complex signal consists of a method for synthesizing the

Wang Tilings completed

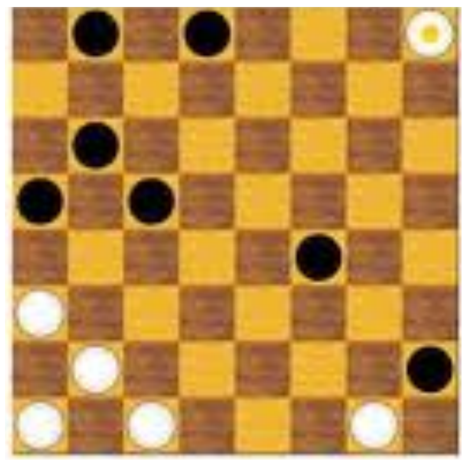
co-NP, EXP, NEXP

New problems that don’t fit neatly in P or NP, and complexity classes where they do fit

Generalized Checkers

Instance: A configuration of an n x n checkerboard

Problem: Can the Black player force a win, if Black moves first?



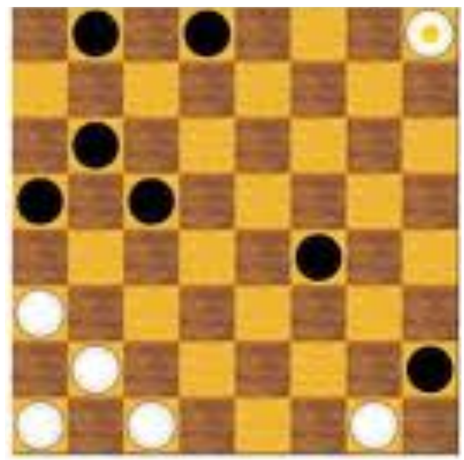
Players alternate moves of one of their pieces; a move may reduce the number of pieces

A player wins when there are no pieces on the board of the opposite colour

Generalized Checkers

Instance: A configuration of an n x n checkerboard

Problem: Can the Black player force a win, if Black moves first?



Generalized checkers is complete for EXP (Robson, 1984. See also work by Jonathan Schaeffer at U. Alberta)

Generalized Checkers algorithm

Construct the following directed graph:

- Nodes correspond to game configurations; there can be two copies of each configuration, one where White moves and where Black moves
- Edges represent next possible moves

Label winning Black nodes “Black Wins!”

Generalized Checkers algorithm

Repeat

For each unlabeled node *i*, label *i* if

- If *i* is Black and *some* child of *i* is labeled
- If *i* is White and *all* children of *i* are labeled

Until no new node is labeled

If the input configuration is labeled "Black wins", output "yes", otherwise output "no"

Succinct SAT

Let ϕ be a Boolean formula in conjunctive normal form with n variables and m clauses

A *succinct representation* of ϕ is a Boolean circuit C which

- on input $(0,i,k)$ returns 1 iff the literal $\neg x_i$ appears in clause k
- on input $(1,i,k)$ returns 1 iff the literal x_i appears in clause k

Succinct SAT

Instance: A *succinct representation* of a Boolean formula ϕ in conjunctive normal form

Problem: Is ϕ satisfiable?

Tautology

- **Instance**: Boolean formula ϕ
- **Problem:** Is ϕ a tautology, i.e., satisfied by every truth assignment to its variables?

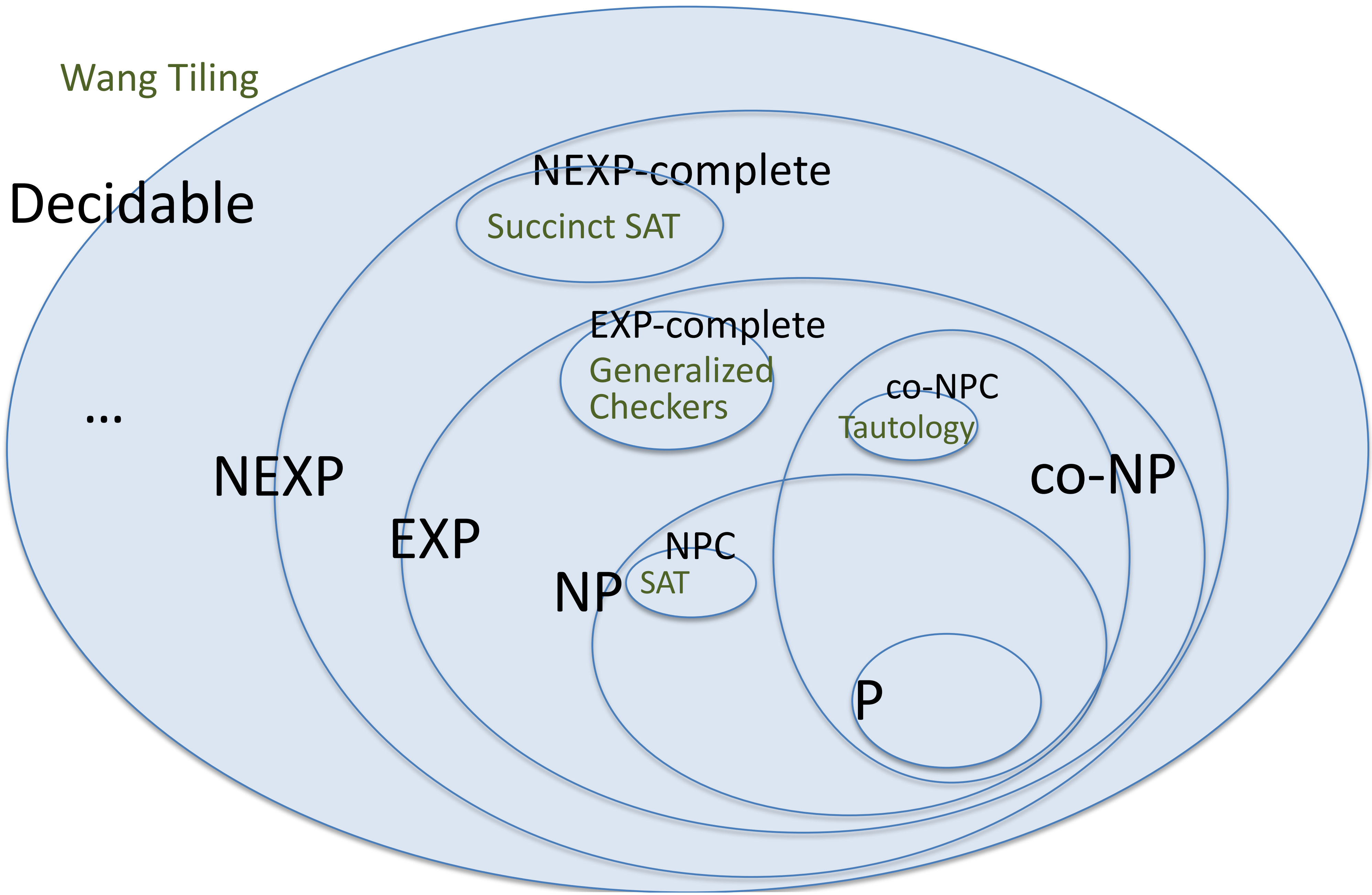
Tautology

- **Instance**: Boolean formula ϕ
- **Problem:** Is ϕ a tautology, i.e., satisfied by every truth assignment to its variables?

- Tautology is complete for

  co-NP = $\{ L \mid \overline{L} \text{ is in NP}\}$

Summary: New problems and complexity classes



Next class

- Time hierarchy theorem
- Reading: Arora-Barak, 3.1