Self-assembly of Microsystems

Karl F. Böhringer, Ph.D.
Professor, Electrical & Computer Engineering and Bioengineering
Director, Institute for Nano-Engineered Systems (NanoES)
University of Washington
Seattle, WA, USA

Part IV: Computational Aspects of Self-assembly

Models for deterministic and stochastic assembly; analogy to chemical reaction kinetics; equivalence between computation and self-assembly; DNA computation, Turing machines and Wang tiles

Self-assembly and Chemical Reactions

- Hosokawa, Shimoyama and Miura observed in 1994 the analogy between self-assembling systems and chemical reactions.
  - They modeled self-assembly with equations for reaction kinetics.
  - This approach provides the means to generate the time evolution of a self-assembling system, and to determine its equilibrium state(s).

Self-assembly Reactions (1)

- Chemical reaction:

$$3\ SiH_4 + 6\ N_2O \rightarrow 3\ SiO_2 + 4\ NH_3 + 4\ N_2$$

- Industrial assembly:

ICs + resistors + capacitors + … $\rightarrow$ iPhone

Obviously, there are limitations to this approach.

Self-assembly Reactions (2)

- Consider a system with two types of components A and B that join one-on-one to form assembly C:

  $$A + B \rightarrow C$$

  with a forward reaction rate constant k_f

- If there is the possibility of disassembly, we write:

  $$A + B \leftrightarrow C$$

  with a reverse reaction rate constant k_r

Self-assembly Reactions (3)

- If we have very large numbers of components:

$$\frac{dA}{dt} = -k_f A \cdot B + k_r C$$

$$\frac{dB}{dt} = -k_f A \cdot B + k_r C$$

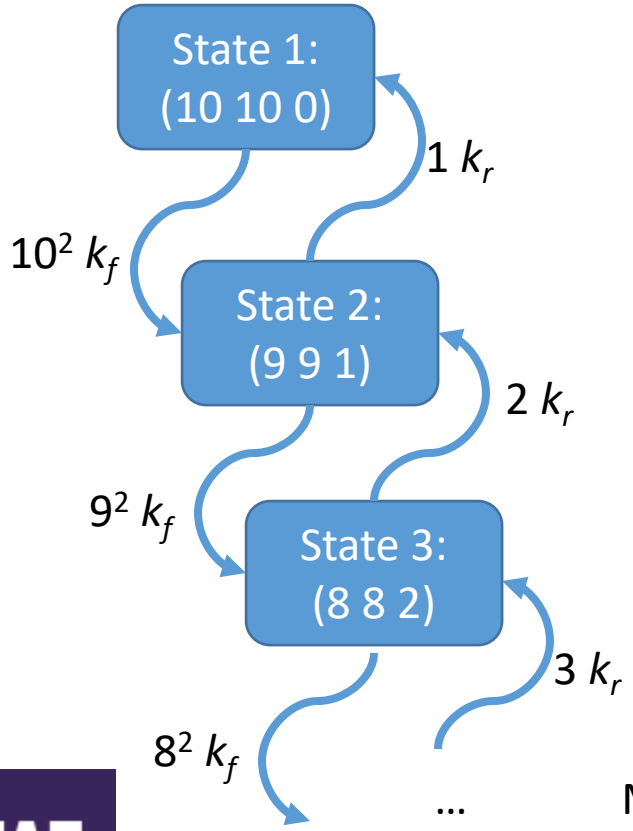
$$\frac{dC}{dt} = k_f A \cdot B - k_r C$$

- System of ordinary differential equations.

Steady state when $(A \cdot B)/C = k_r/k_f$

Self-assembly Reactions (4)

- If we have a smaller number of components:



Markov process

Matrix of transition rates:

$$\begin{pmatrix} . & k_r & . & \cdots \\ 10^2 k_f & . & 2kr & \\ . & 9^2 k_f & . & \\ \vdots & & & \ddots \end{pmatrix}$$

Steady states are eigenvectors of matrix.

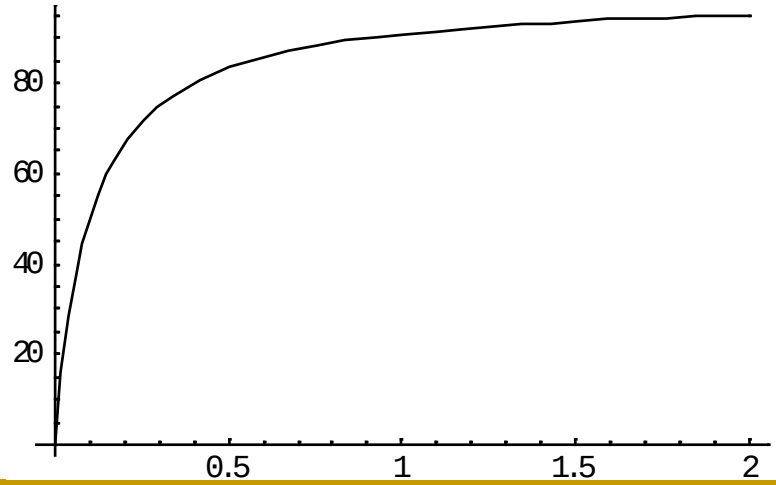
Chemical Reaction Kinetics

- There are two mathematical formalisms to describe behavior of a chemical system:
  - "Reaction rate equations" are coupled ordinary differential equations that provide a deterministic time evolution of the system.
  - The "master equation" is a single differential-difference equation that captures the stochastic behavior of chemical kinetics.
    - $P(X_1,X_2,\ldots,X_N;t)$ = probability that at time t, there will be X_1 molecules of species S_1, X_2 molecules of species S_2, …
- It can be shown that as the number of reactants goes towards infinity, the two formalisms converge.
- Daniel Gillespie developed an algorithm for exact stochastic simulation in 1977.

Deterministic Solution for Reaction Kinetics

Assume we have n components A and n sites B.

- Initially, the rate of reaction is $k\ n^2$.
- If there are x complete assemblies then the rate of reaction is $k\ (n - x)^2$.
- This leads to a differential equation for $x(t)$: $dx/dt = k\ (n - x)^2$
- Equilibrium: $dx/dt = 0 \rightarrow x = n$
- Solution of this differential equation: $x(t) = k\ n^2\ t\ /\ (1 + k\ n\ t)$

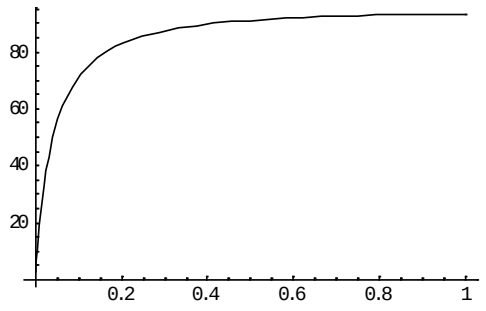


Kinetics with Reverse Reaction

Assume we have n components A and n sites B.

- Forward rate of reaction is $k_f\,[A]\,[B]$.
- Reverse rate of reaction is $k_r\,[A \cdot B]$.
- If there are x complete assemblies then the forward rate of reaction is $k_f\,(n-x)^2$ and the reverse rate of reaction is $k_r\,x$.
- This leads to a new differential equation for $x(t)$: $dx/dt = k_f\,(n-x)^2 - k_r\,x$.

$$x(t) = \frac{2k_f n^2}{k_r + 2k_f n + \sqrt{k_r}\sqrt{k_r + k_f n}\cosh\left(\frac{1}{2}\sqrt{k_r}\sqrt{k_r + k_f n}\,t\right)}$$



Limitations to the Analogy with Chemical Reaction Kinetics

- The number of components is finite, may not be assumed as infinite.
- Self-assembly components are geometrically and physically more complex than atoms and molecules.
- Deriving reaction rate constants from first principles is very difficult.
  - What is “temperature” in a self-assembly system?
- Describing a self-assembly system is a multi-physics problem.
  - Models may require techniques spanning from molecular dynamics to robotics to computational geometry.

Self-assembly: the Big Picture

- > 100 years ago:

  "energy = mass"

  - Einstein 1905 (Nobel Prize Physics 1921)

- > 10 years ago:

  "assembly = computation"

  - Adleman 1994 (Turing Award 2002)
  - DNA computation of NP-hard combinatorial problems

- Proposition:
  - This equation not only applies to DNA, but in particular also to assemblies of micro and nano systems

Self-assembly: the Big Picture

"assembly = computation"

Direction of equations is important for practical engineering purposes:

- energy = mass:
  - "$\rightarrow$" esoteric physics
  - "$\leftarrow$" most powerful known source of energy
- assembly = computation:
  - "$\rightarrow$" esoteric computers
  - "$\leftarrow$" most powerful known method of manufacturing
    - from CAD to VLSI
    - from DNA to living organisms

Self-assembly: the Big Picture

Thus, “assembly = computation” means:

If I can create a (data) structure with some program
then I can create a corresponding (physical) structure by self-assembly

Assembly and Computation

- Background
  - NP-hard problems
  - Turing machines
  - Wang tilings

- Adleman: DNA strands compute solutions to combinatorial problems
- Winfree: DNA tiles perform arithmetic calculations

DNA Computing

- In 1994, Leonard M. Adleman showed that hard computational problems can be solved by self-assembly:
  - The problem is encoded in DNA.
  - The solution is found by processing the DNA.
- Why is this important?
  - A new way for nano-scale, massively parallel computing.
  - If self-assembly can simulate any algorithm, then any structure that can be described by an algorithm can also be realized with self-assembly.

Background: NP-hard Problems

- NP-hard problems are problems that are very difficult to solve with a computer (or without one).
- NP stands for nondeterministic polynomial-time.

- An NP-hard problem is a decision problem:
  - yes/no answer.
- If a possible solution is found, it is “easy” to check whether the solution is correct:
  - “easy” means “with a polynomial-time algorithm”.
- But it is “difficult” to find a solution:
  - “difficult” means “combinatorically many”.

Background: NP-hard Problems

- Example: “SUBSET-SUM”
  - Given a set *S* of *n* integers, does any non-empty subset of *S* add up to zero?
  - It is easy to verify a given solution S_0:
    - Check whether S_0 is a subset of *S*.
    - Check whether S_0 adds up to zero.
  - It is difficult to find a solution:
    - To find a solution, or to prove that there is none, we have to enumerate (more or less) all possible subsets.
    - This leads to a “combinatorial explosion”, i.e., an algorithm that is exponential in *n*.

Background: NP-hard Problems

- There exist many such problems.
  - Another example: “KNAPSACK”
    Given n items with cost c_i and value v_i, are there items of value at least V without exceeding cost C ?

- Interestingly, all these problems are similarly hard.
  - If I can find an efficient algorithm for KNAPSACK then I can also find an efficient algorithm for SUBSET-ZERO, and vice versa.
  - These problems are “NP-hard”, they are the most difficult ones that can be solved nondeterministically in polynomial time.

Hamiltonian Path

- Another NP-hard problem:
  - Traces back to 19th century Irish mathematician W. R. Hamilton, who invented a game: find a path along the edges of a dodecahedron that visits each vertex exactly once.
- In a general graph, a Hamiltonian path is a path that connects all vertices via edges without visiting any vertex twice.

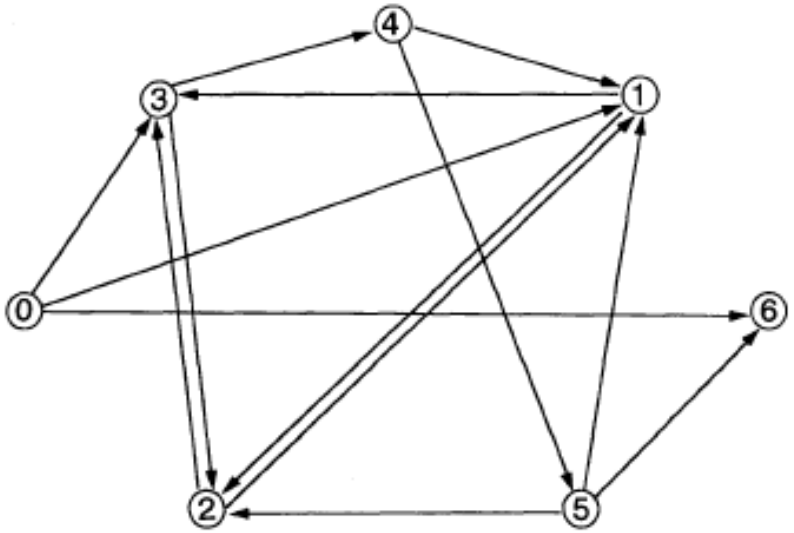


Fig. 1. Directed graph. When $v_{in} = 0$ and $v_{out} = 6$, a unique Hamiltonian path exists: 0→1, 1→2, 2→3, 3→4, 4→5, 5→6.

Finding a Hamiltonian Path

A nondeterministic algorithm for a Hamiltonian path of a graph with *n* vertices from v_{in} to v_{out} :

1. Generate random paths through graph (i.e., random sequences of vertices).
2. Delete all paths that do not start with v_{in} or do not end with v_{out} .
3. Delete all paths of length not equal to *n.*
4. Delete all paths that visit a vertex more than once.

If there is a path left over, then it is a Hamiltonian path.

Adleman realized all these steps with DNA processing for the graph with 7 vertices on the previous slide.

1. Encoding a Graph in DNA

- Each vertex v_i is represented by a 20-base sequence (20-mer) of DNA.
- Each directed edge (v_i,v_j) is represented by the last 10-mer of v_i and the first 10-mer of v_j .
  - Exception 1: all edges (v_{in},v_j) use the entire 20-mer of v_{in} and the first 10-mer of v_j .
  - Exception 2: all edges (v_i,v_{out}) use the last 10-mer of v_j and the entire 20-mer of v_{out} .
- Experiment:
  - 50pmol of complementary DNA for each vertex.
  - 50pmol of DNA for each edge.
  - The complementary strands bind and thus represent random paths in the graph.

1. Encoding a Graph in DNA

- There were about 3 10^{13} oligonucleotides for each vertex and edge in the solution.
- How many different paths exist?
  - Infinitely many, but very long paths (i.e., very long DNA sequences) are unlikely to be created since v_{in} and v_{out} terminate the DNA sequence.
  - Count all possible sequences up to length 14: $7^{14} = 6.8\ 10^{11}$.
  - It is very likely that the solution is among the self-assembled DNA sequences.

2. Select Paths from v_{in} to v_{out}

- DNA amplification with polymerase chain reaction (PCR):
  - PCR splits DNA between two markers and duplicates it in each cycle.
  - Here, we use markers for v_{in} and v_{out} such that only DNA representing paths from v_{in} to v_{out} are amplified.

3. Select Paths of Length *n*

- Run DNA through agarose gel:
  - Separation of DNA molecules by length: the mobility of the DNA molecule is directly proportional to its size.
  - Extract the DNA sequences with 20*n* = 140 base pairs, representing paths with exactly 7 vertices.
  - PCR amplification to improve purity.

4. Delete Missing-Vertex Paths

- Create single stranded DNA.
- For i = 0 to n
  - Add complementary DNA sequence representing v_i with biotin-avidin bound magnetic beads.
  - Hybridization.
  - Separate labeled DNA from unlabeled DNA.
  - Repeat
- Here, we are left only with paths that include all vertices.
- … and we have solved the Hamiltonian Path.

Pro's and Con's

- Massively parallel computing.
- High density (theoretically, 1 molecule = 1 data structure).
- Energy efficiency (near thermodynamic optimum).
- Linear in number of vertices (as opposed to exponential).

- Demo for a trivial problem size. Ultimately, combinatorial explosion is unavoidable.
- Lengthy lab procedure.
- …

Turing Machine

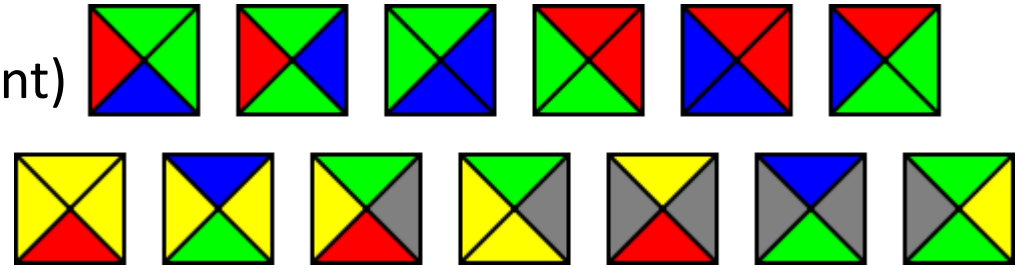
- Proposed 1936 by English mathematician Alan Turing.
- An extremely simple machine that can perform computation.
- Various versions of Turing Machines exist; a typical configuration consists of
  - an endless tape,
  - a head that can read and write symbols on the tape,
  - a look-up table that decides
    - whether to move the tape left of right,
    - whether to read or what to write on the tape,
    - what “state” to transition to.

Turing Machines

- It has been shown that computations with modern computers / programming languages can also be performed by Turing Machines.
- It is generally agreed that *any* computation can be performed by a Turing Machine.
- In fact, one way to define what is meant by "computation" is a program executed by a Turing Machine.

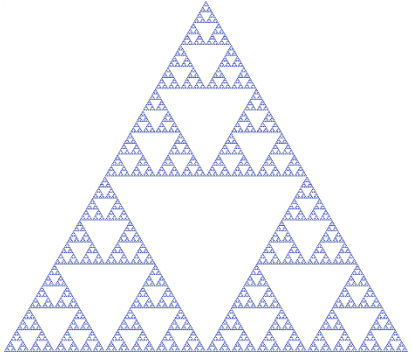
Wang Tiles

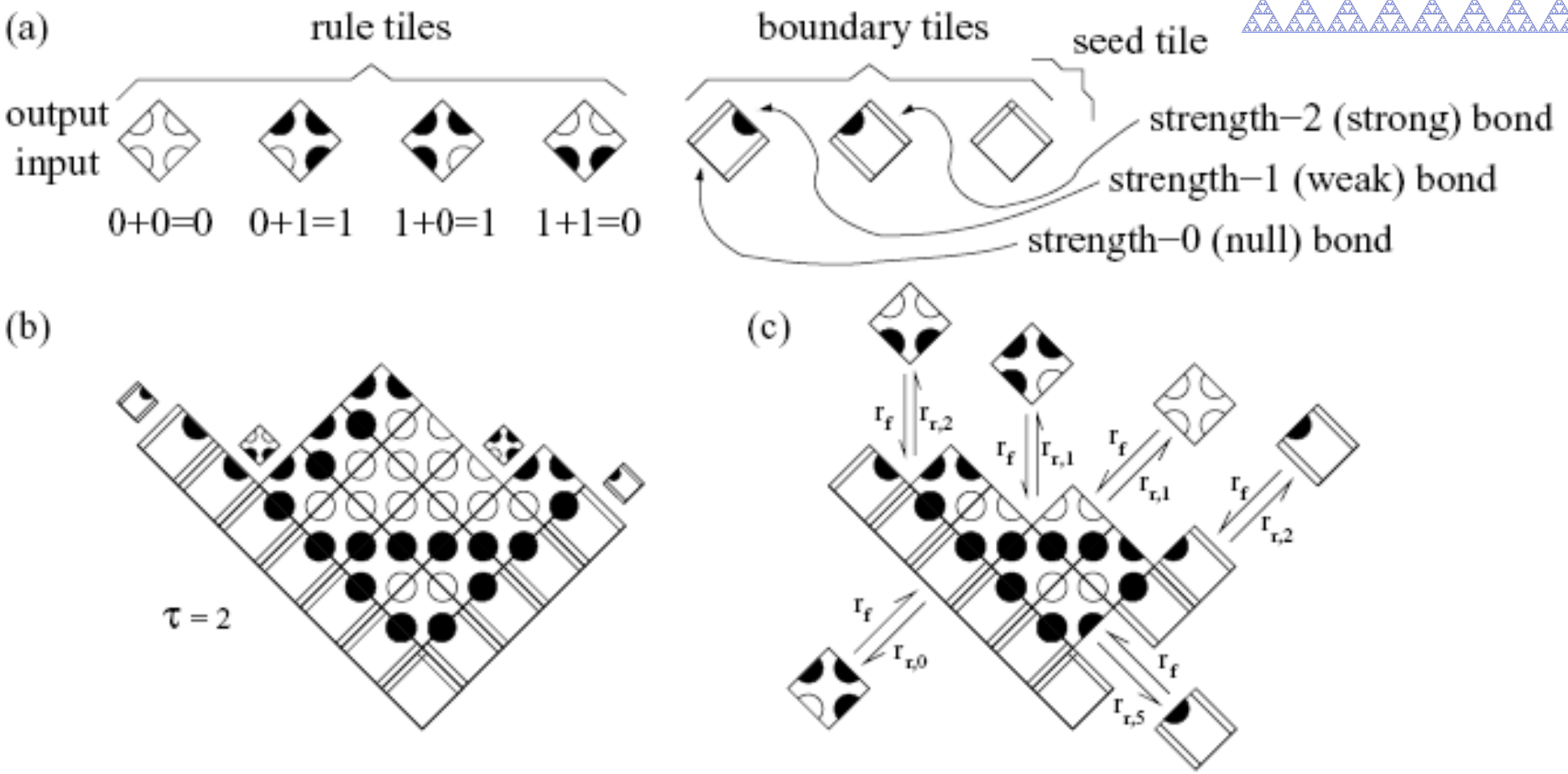
- Introduced by Chinese-American mathematician and philosopher Hao Wang in 1961.
  - Wang gave an algorithm to decide whether a given tile set can cover the plane (adjacent colors must match, and tiles cannot be rotated).
  - Robert Berger (his student) proved in 1966 that this algorithm was wrong. In fact, no such algorithm can exist.



- It has been shown that Wang tiles are equivalent to Turing machines.
  - This means that any computational problem can be represented with Wang tiles.

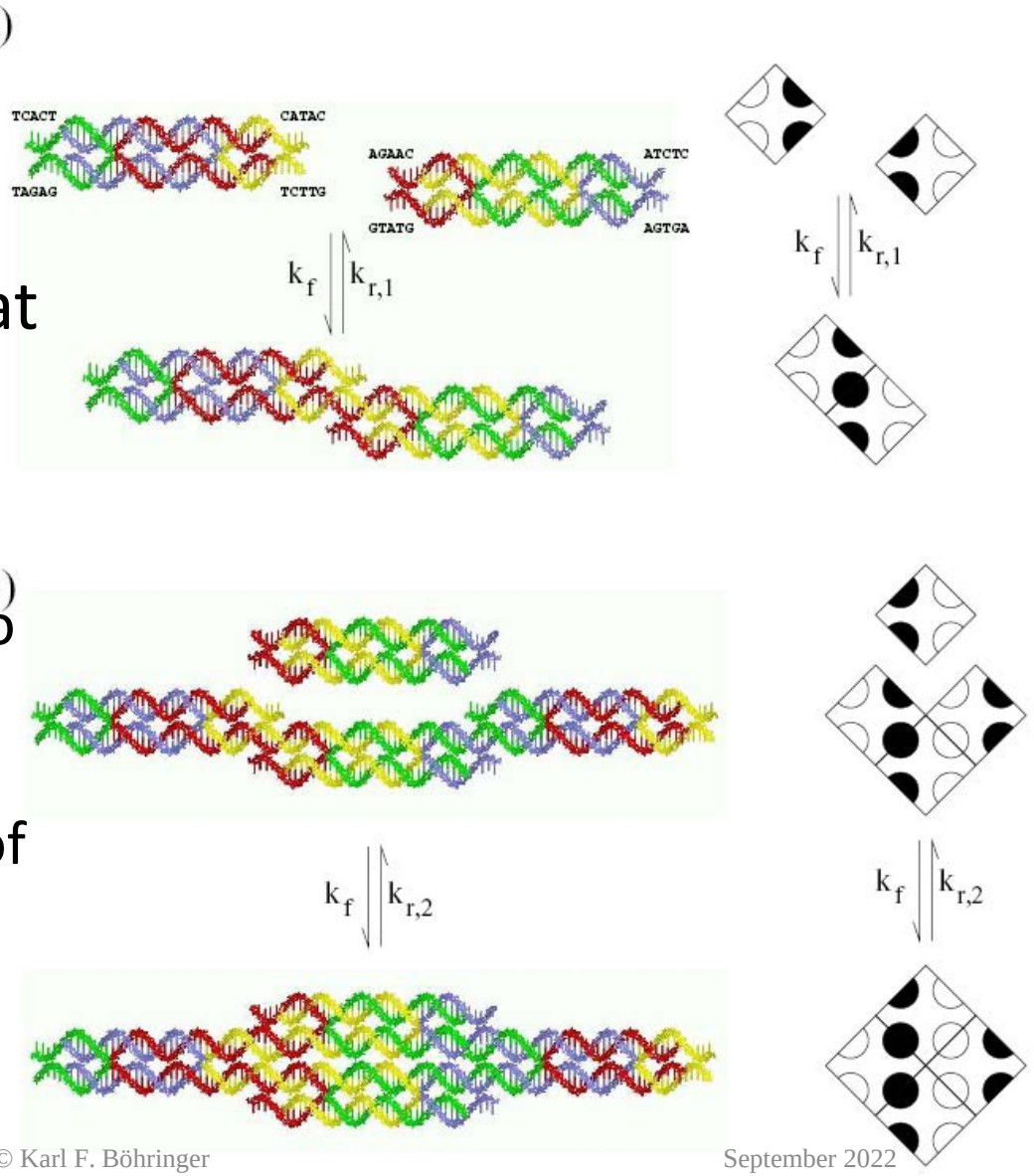
Example: Sierpinski Triangle





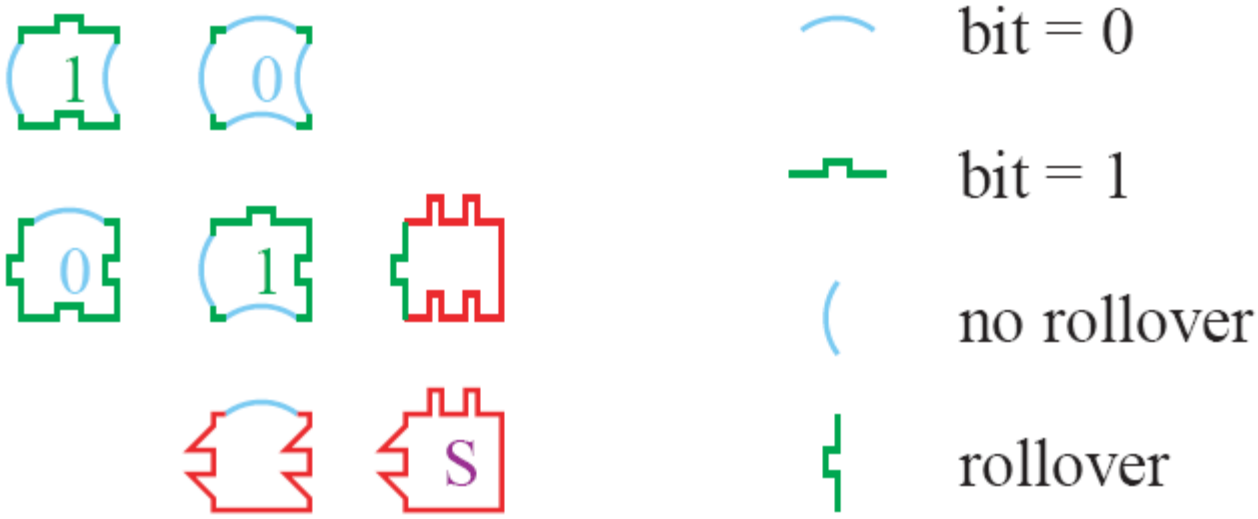
DNA Tiles

- Double cross-over (DX) molecules are ideal to implement Wang tiles at the molecular level.
  - DX provide stiffness for 2D assembly.
  - 4 sticky ends connect to neighbor tiles.
  - ssDNA on sticky ends provide a high degree of programmability.



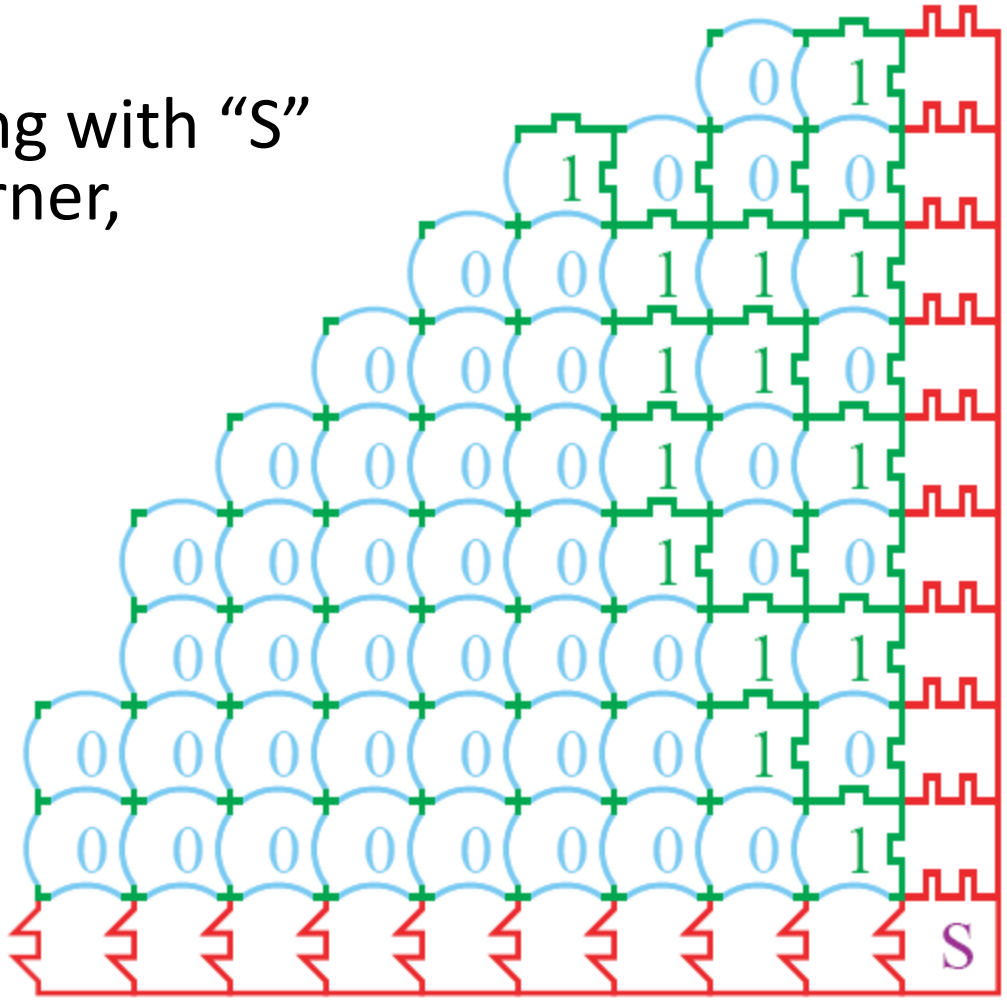
DNA Arithmetic

- Winfree (among others) showed that tiles can perform arithmetic. The following 7 tiles create a binary counter:



DNA Arithmetic

- Self-assembly, starting with “S” in the lower right corner, produces rows of increasing binary numbers.
- This works well only if the new tiles assemble prefer-entially with 2 already assembled tiles.



DNA Computation: Conclusions

- Molecular computation by DNA self-assembly is attractive because of
  - massively parallel processing,
  - high data density,
  - abstraction that separates computation from chemistry.
- Adleman (Science 1994):
  - “One can imagine the eventual emergence of a general purpose computer consisting of nothing more than a single macromolecule conjugated to a ribosome-like collection of enzymes that act on it.”
- Currently, DNA computation is not widely expected to replace electronic computers for the solution of hard computational problems.

DNA Computation: Conclusions

- Consider an analogy: $E = mc^2$
  - Converting energy into mass realizes (sort of) the alchemists' dream of converting one element into another.
  - But it is not practical (based on today's physics and engineering knowledge).
  - However, the inverse process (generating nuclear energy) is possible and – arguably – useful.
- Similarly:
  - Using DNA self-assembly for computation may not be practical.
  - But applying theory of computation to create DNA structures via programmable self-assembly is practical.