Algorithmic self-assembly with DNA tiles

David Doty (University of California, Davis)

Algorithmic Foundations of Programmable Matter

Dagstuhl, August 2018

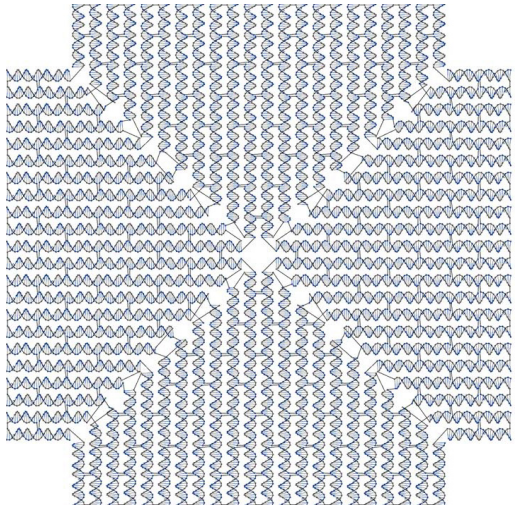
SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik

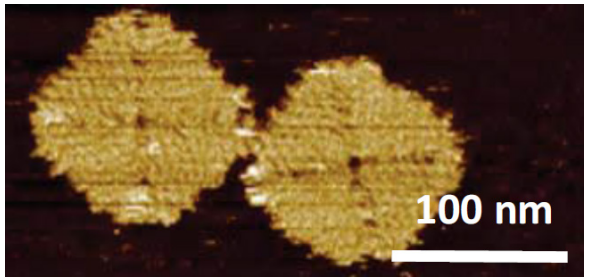
DNA tile self-assembly

DNA tile self-assembly

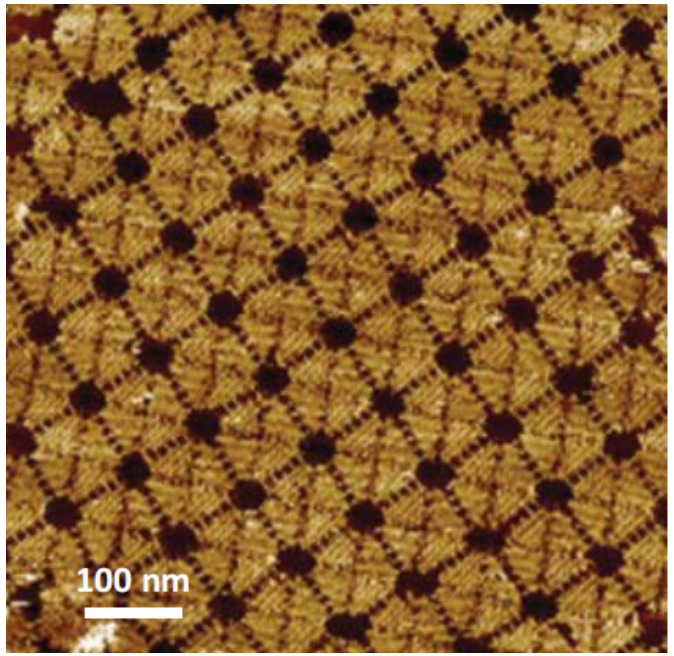
monomers (“tiles” made from DNA) bind into a crystal lattice

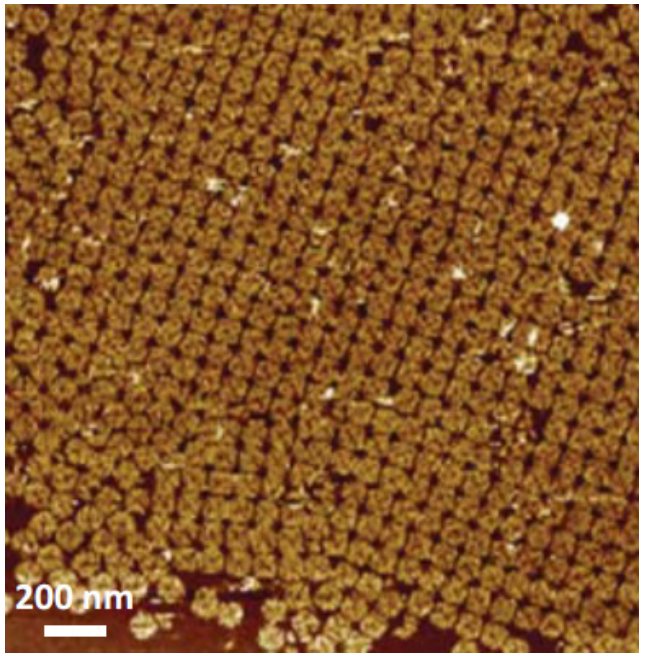
tile





lattice

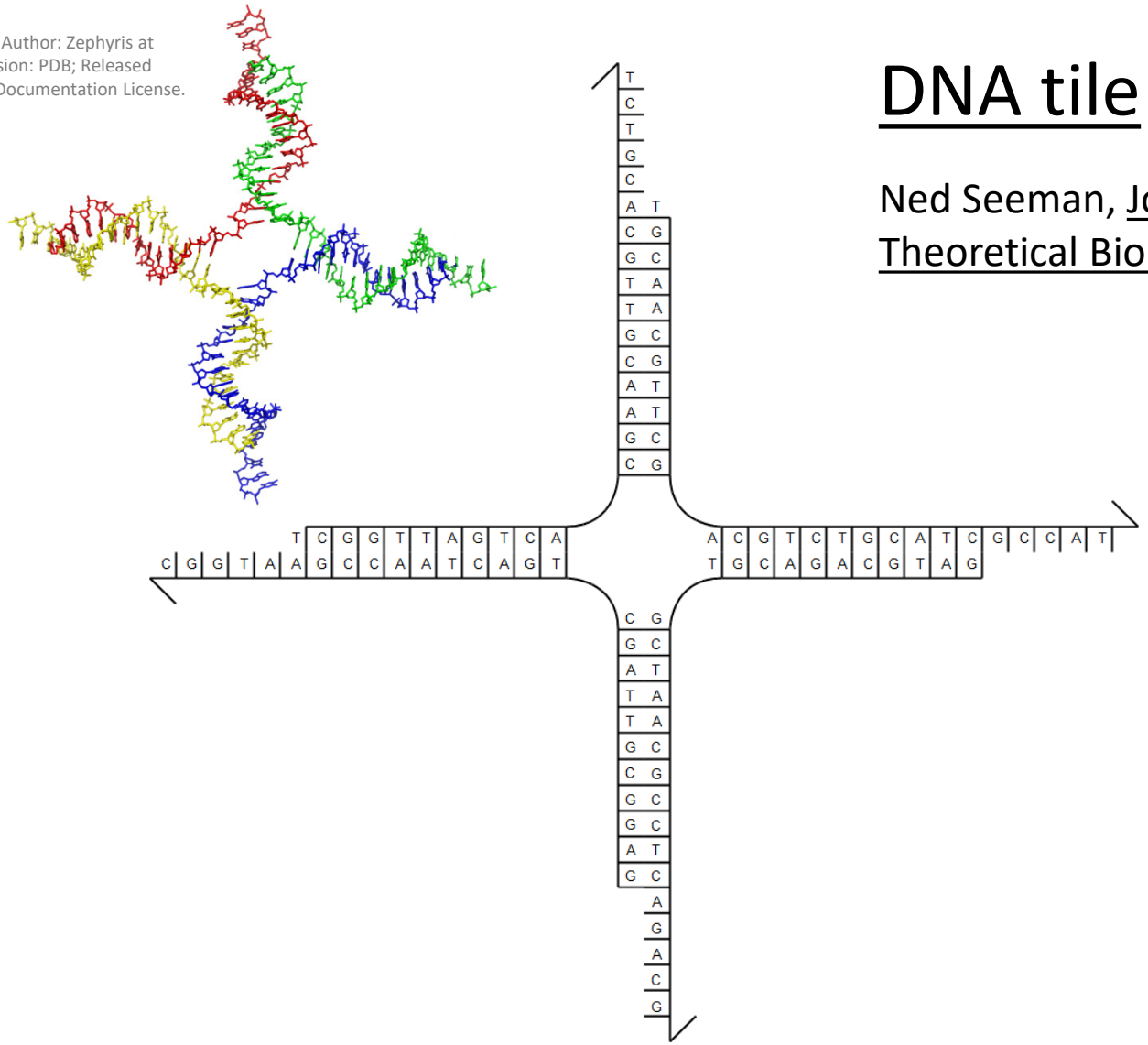




Source: *Programmable disorder in random DNA tilings*. Tikhomirov, Petersen, Qian, Nature Nanotechnology 2017

Practice of DNA tile self-assembly

Source:en.wikipedia; Author: Zephyris at en.wikipedia; Permission: PDB; Released under the GNU Free Documentation License.

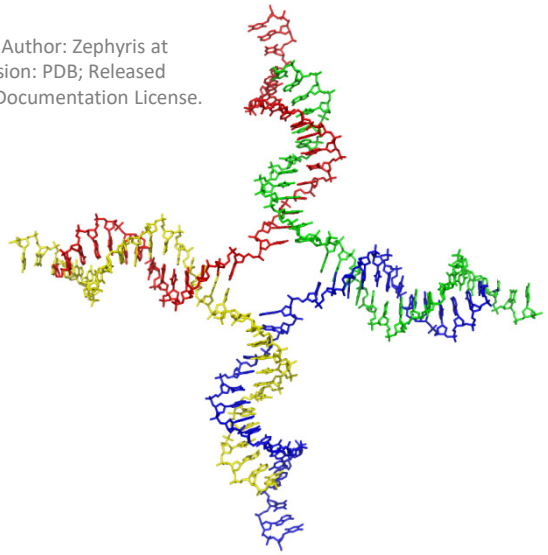


DNA tile

Ned Seeman, Journal of Theoretical Biology 1982

Practice of DNA tile self-assembly

Source:en.wikipedia; Author: Zephyris at en.wikipedia; Permission: PDB; Released under the GNU Free Documentation License.



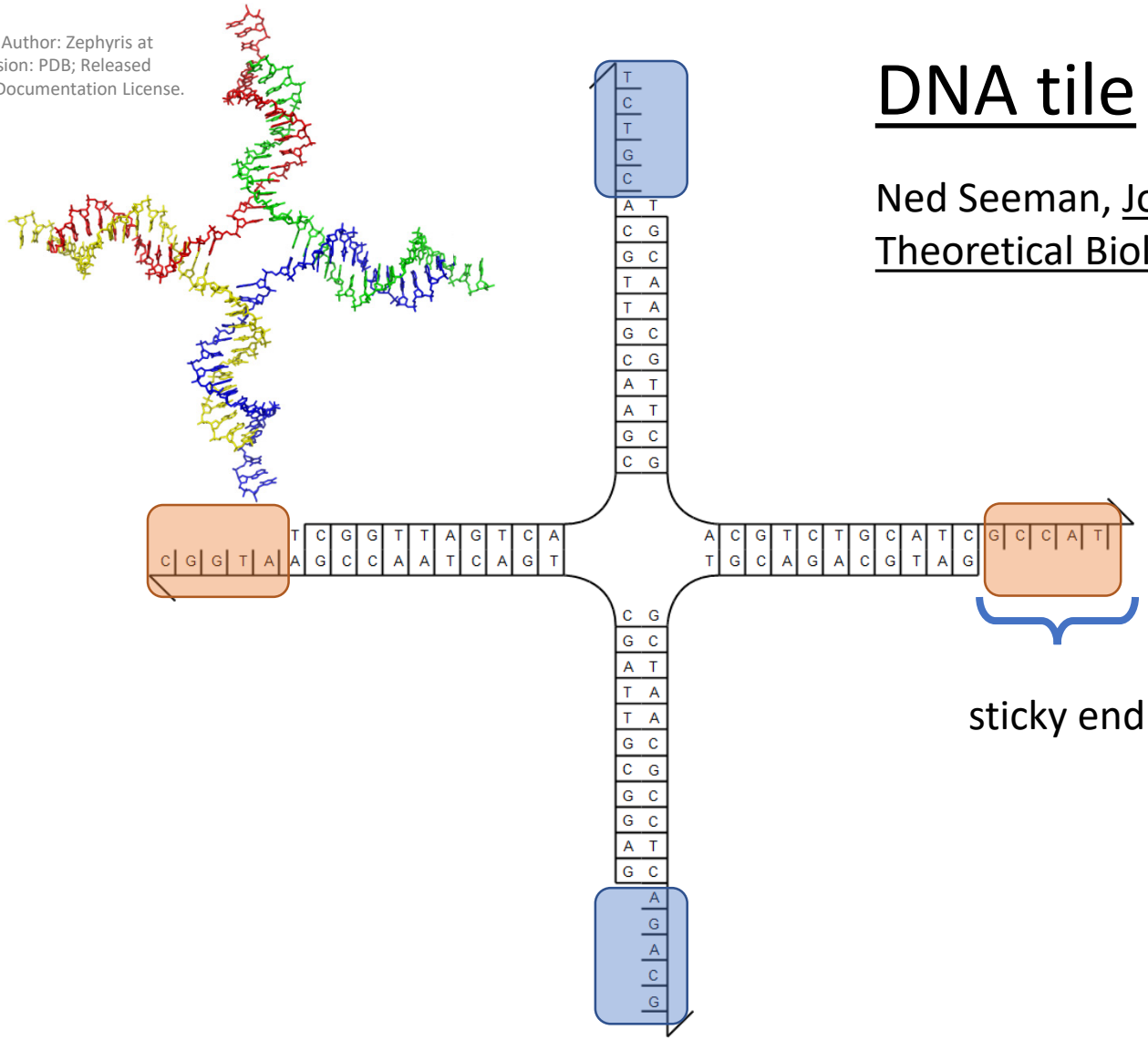
DNA tile

Ned Seeman, Journal of Theoretical Biology 1982

sticky end

Practice of DNA tile self-assembly

Source:en.wikipedia; Author: Zephyris at en.wikipedia; Permission: PDB; Released under the GNU Free Documentation License.



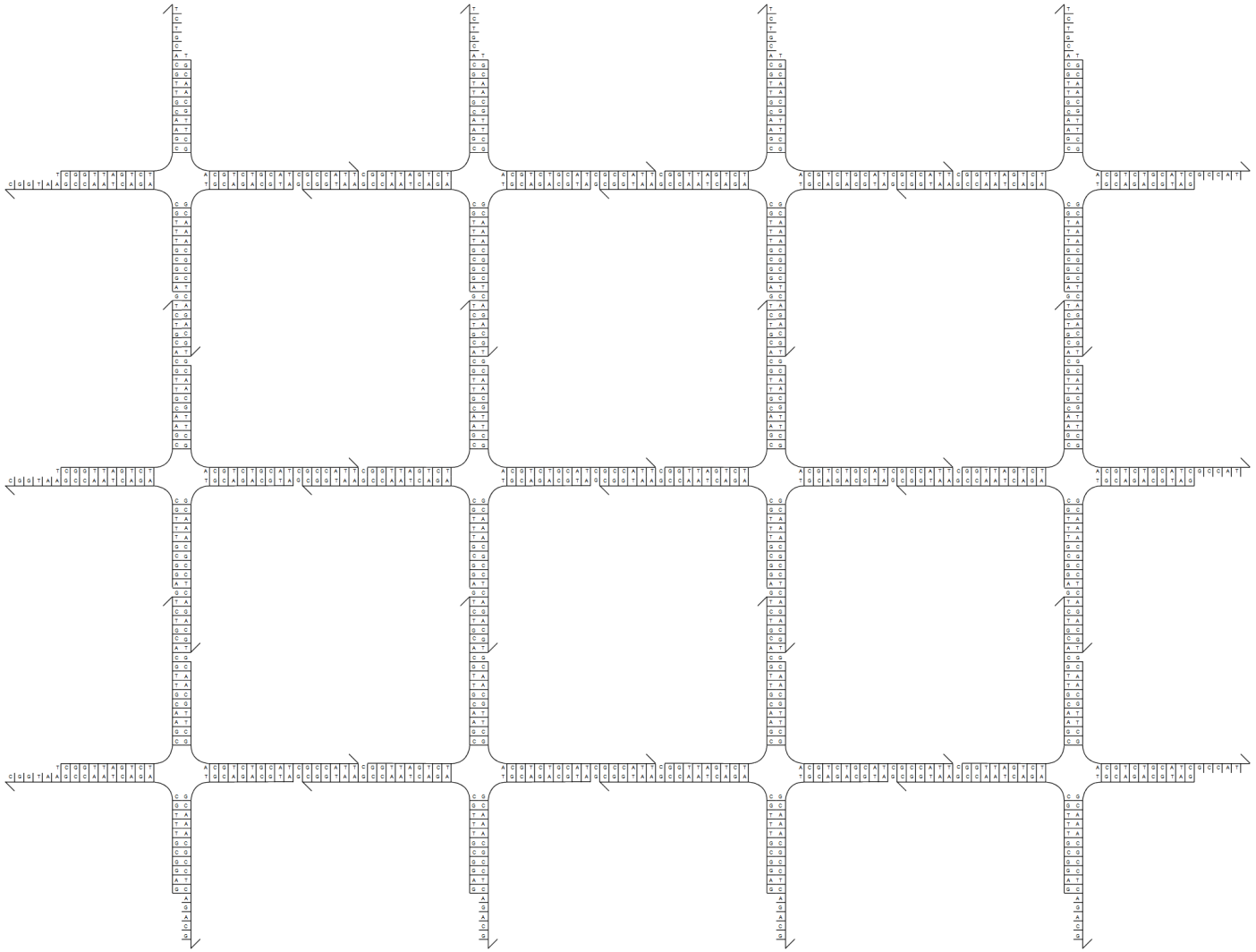
DNA tile

Ned Seeman, Journal of Theoretical Biology 1982

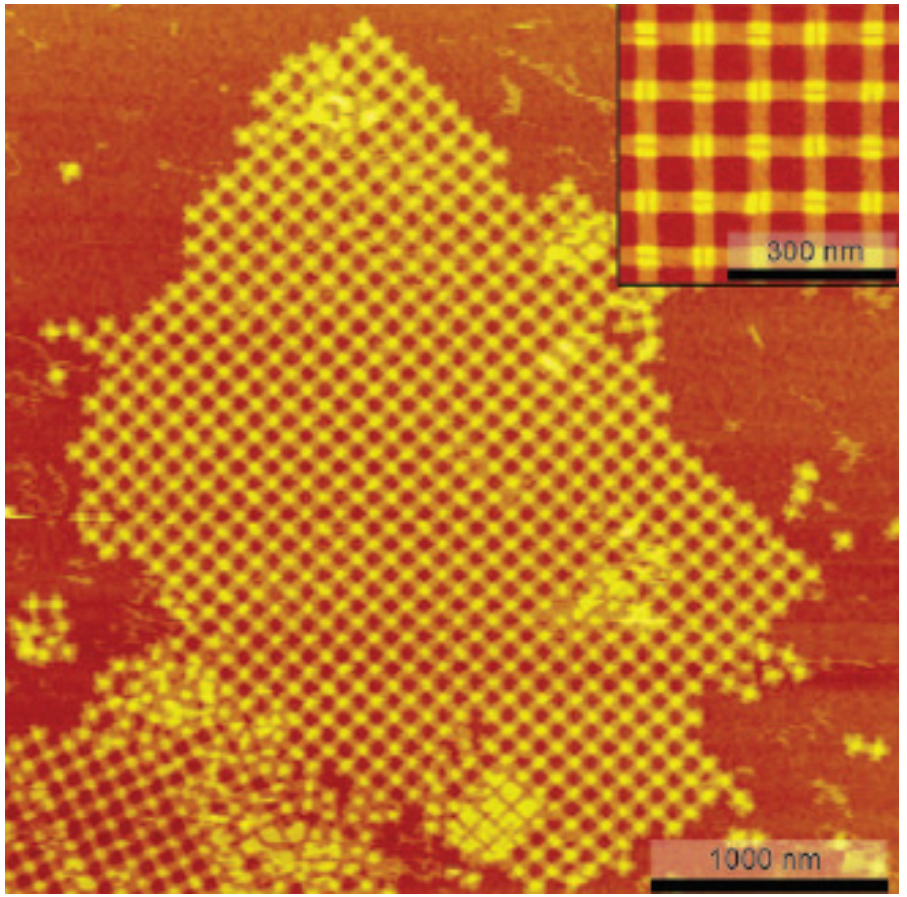
sticky end

Practice of DNA tile self-assembly

Place many copies of DNA tile in solution…



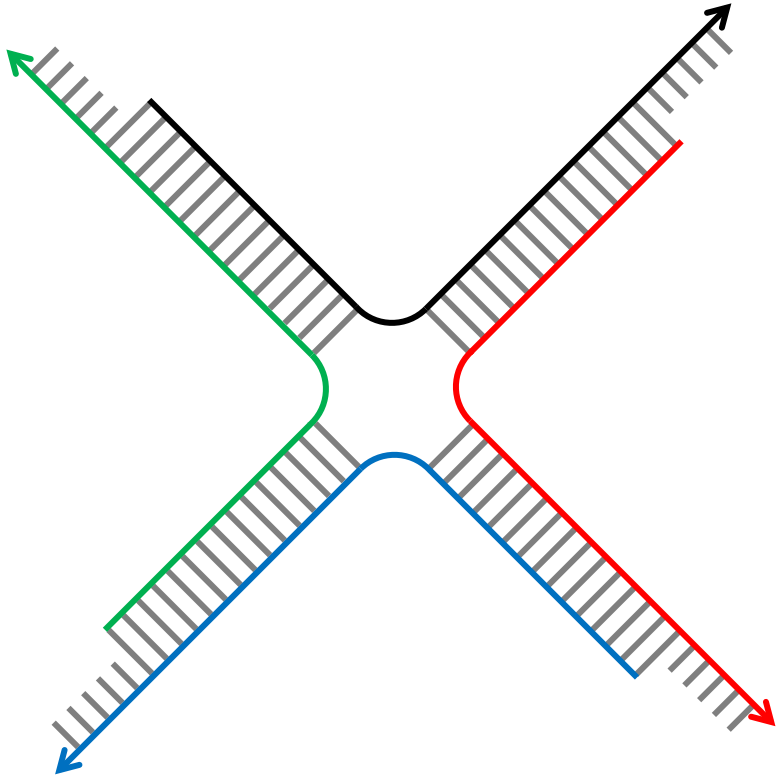
(not the same tile motif in this image)



Liu, Zhong, Wang, Seeman, Angewandte Chemie 2011

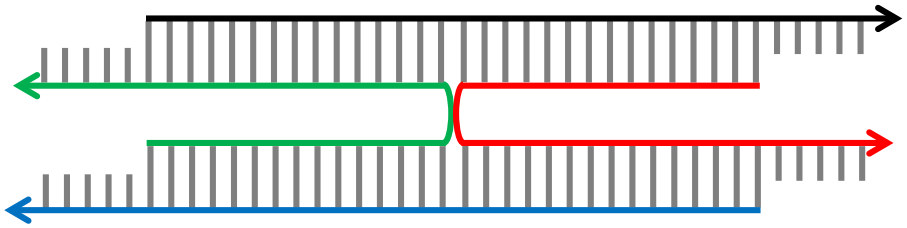
Practice of DNA tile self-assembly

What really happens in practice to Holliday junction (“base stacking”)



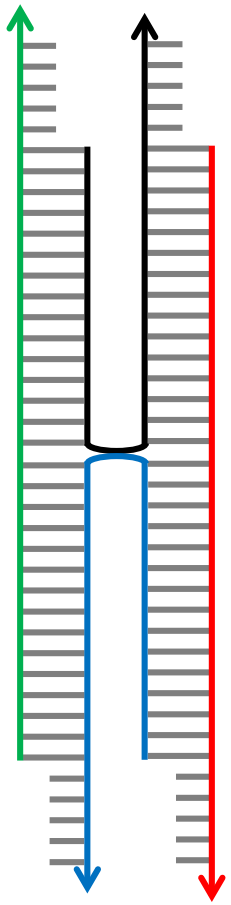
Practice of DNA tile self-assembly

What really happens in practice to Holliday junction (“base stacking”)



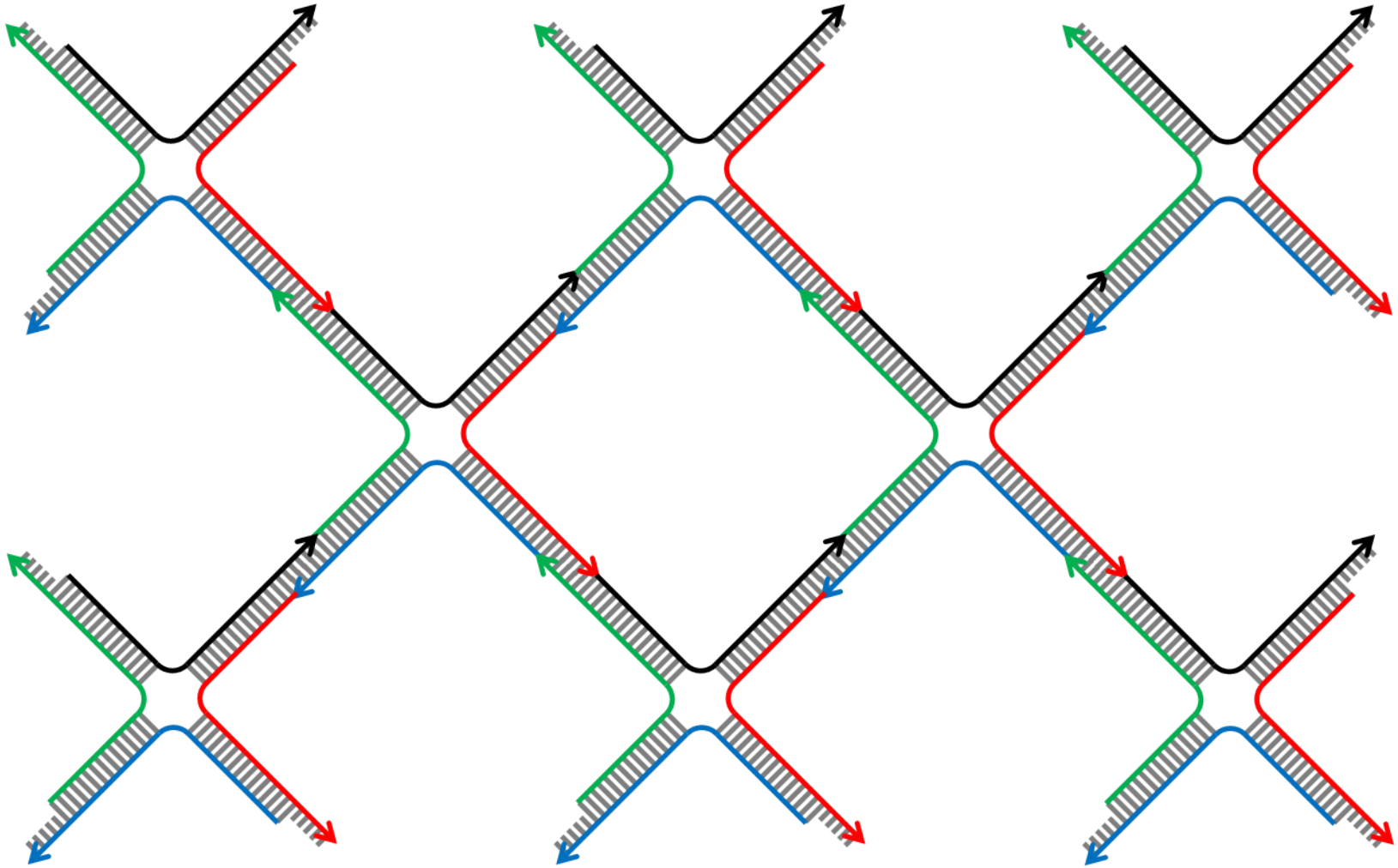
Practice of DNA tile self-assembly

What really happens in practice to Holliday junction (“base stacking”)



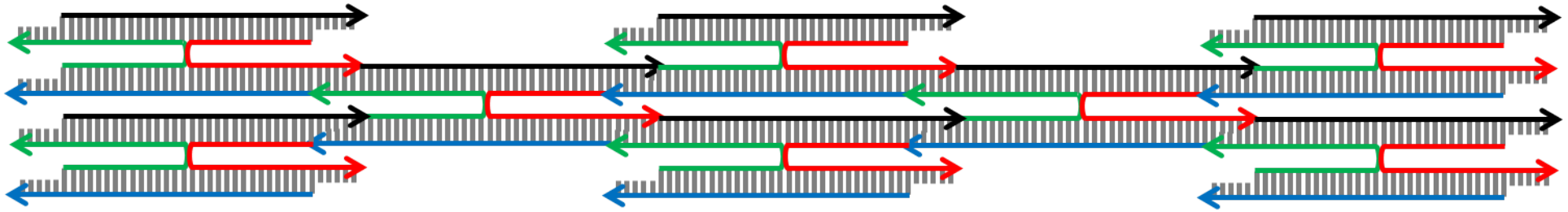
Practice of DNA tile self-assembly

What really happens in practice to Holliday junction (“base stacking”)

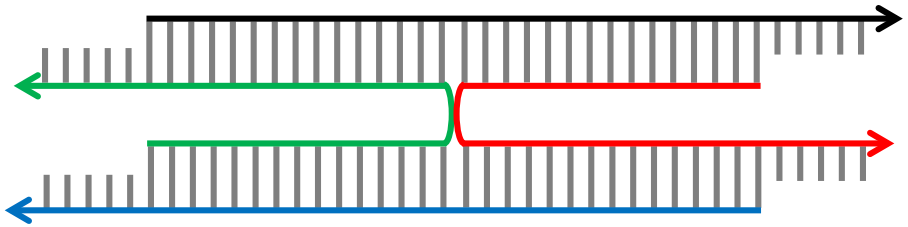


Practice of DNA tile self-assembly

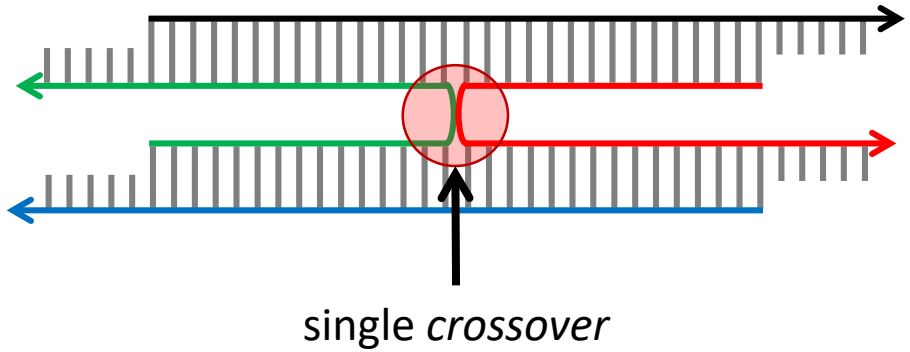
What really happens in practice to Holliday junction (“base stacking”)



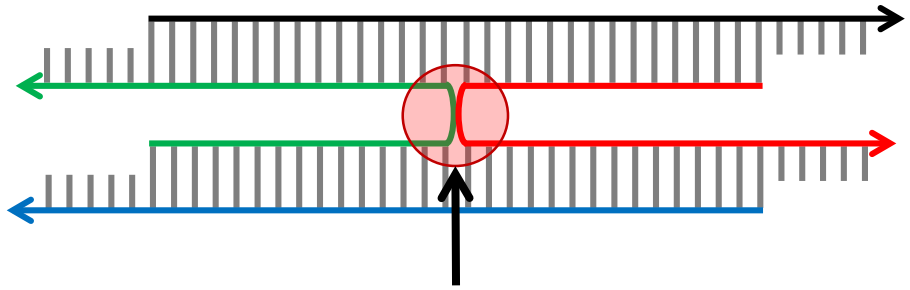
Practice of DNA tile self-assembly



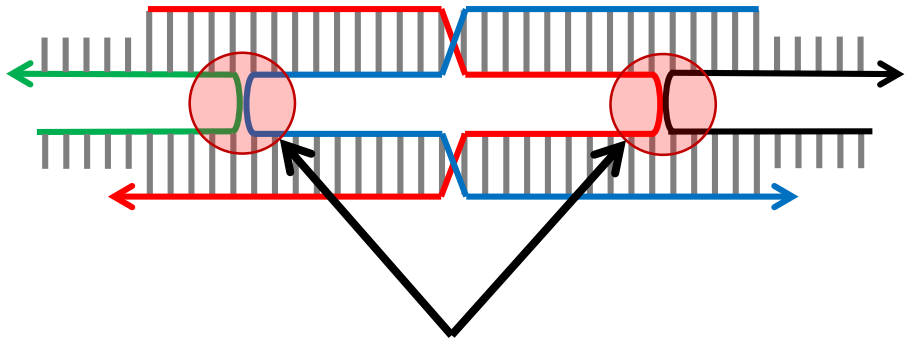
Practice of DNA tile self-assembly



Practice of DNA tile self-assembly



single *crossover*



double crossover

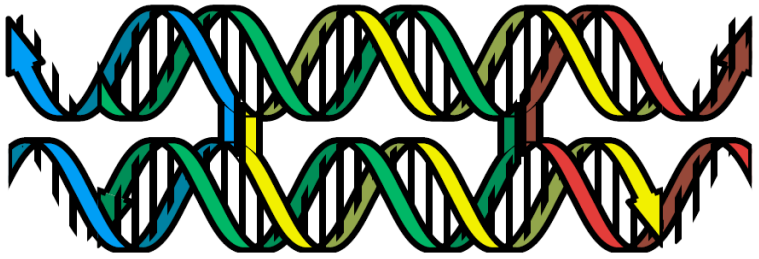
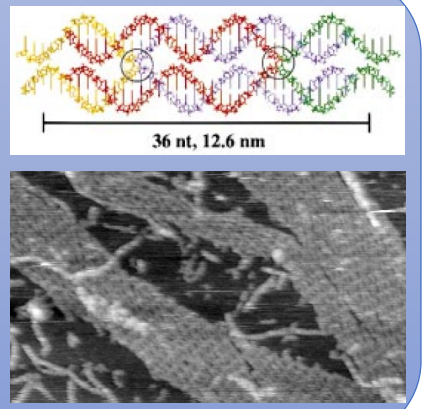


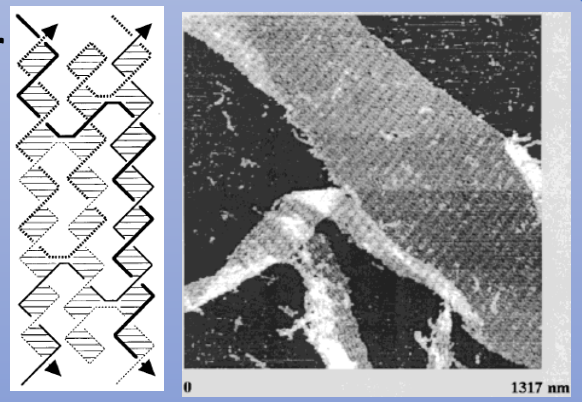
Figure from Schulman, Winfree, *PNAS* 2009

Practice of DNA tile self-assembly

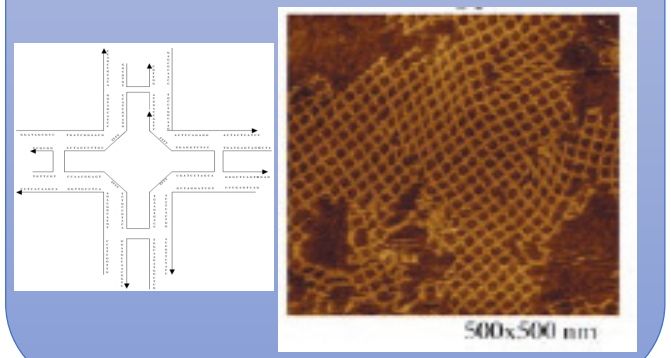
double-crossover tile (Winfree, Liu, Wenzler, Seeman, *Nature* 1998)



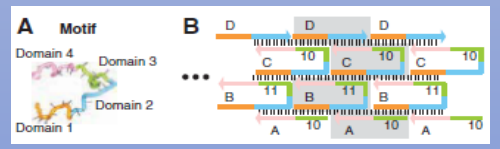
triple-crossover tile (LaBean, Yan, Kopatsch, Liu, Winfree, Reif, Seeman, *JACS* 2000)



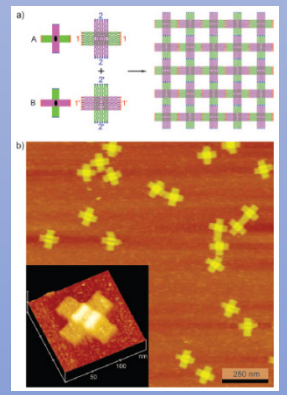
4x4 tile (Yan, Park, Finkelstein, Reif, LaBean, *Science* 2003)

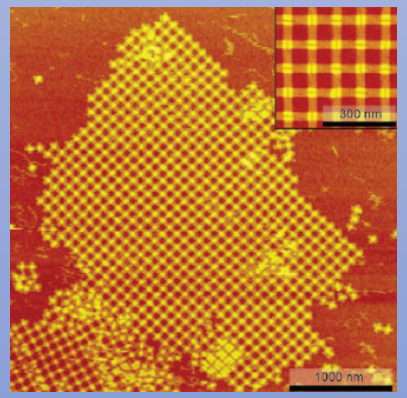


single-stranded tile (Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif, *Science* 2008)

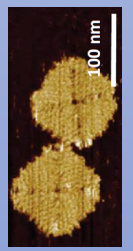


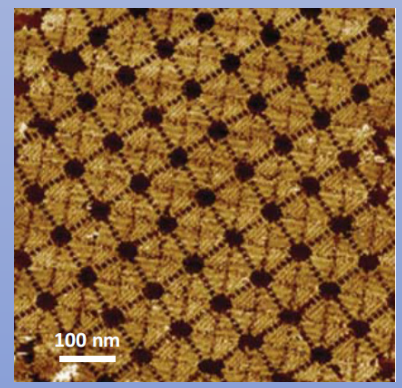
DNA origami tile (Liu, Zhong, Wang, Seeman, *Angewandte Chemie* 2011)





Tikhomirov, Petersen, Qian, Nature Nanotechnology 2017



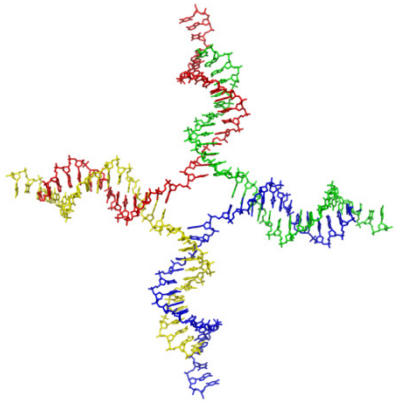


Theory of *algorithmic* self-assembly

What if…
… there is more than one tile type?
… some sticky ends are “weak”?

Erik Winfree

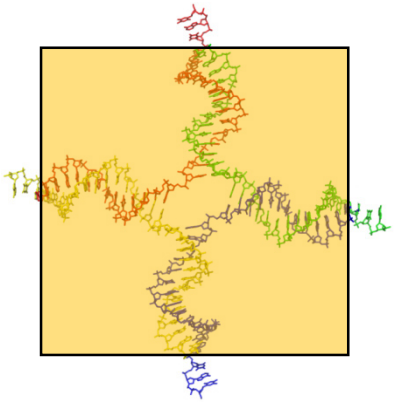
Abstract Tile Assembly Model



Erik Winfree, Ph.D. thesis,
Caltech 1998

Abstract Tile Assembly Model

- **tile type** = unit square

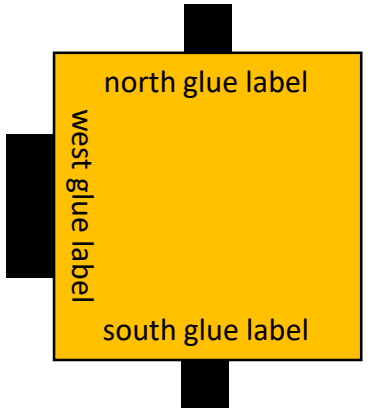


Erik Winfree, Ph.D. thesis,
Caltech 1998

Abstract Tile Assembly Model

- **tile type** = unit square

- each side has a **glue** with a **label** and **strength** (0, 1, or 2)



strength 0

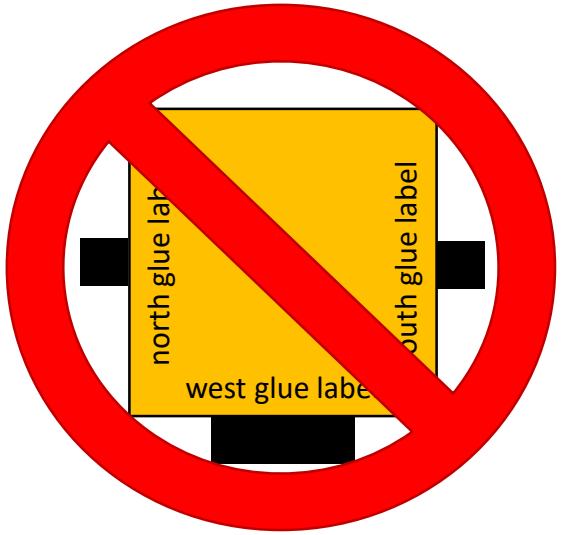
strength 1 (weak)

strength 2 (strong)

Erik Winfree, Ph.D. thesis, Caltech 1998

Abstract Tile Assembly Model

- **tile type** = unit square
- each side has a **glue** with a **label** and **strength** (0, 1, or 2)
- tiles cannot rotate



strength 0

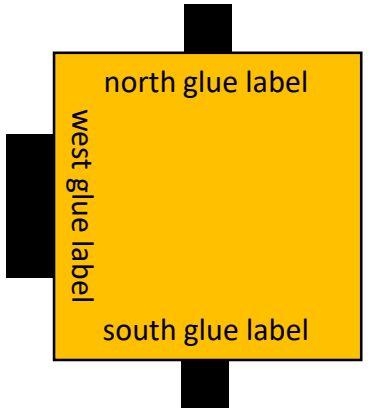
strength 1 (weak)

strength 2 (strong)

Erik Winfree, Ph.D. thesis, Caltech 1998

Abstract Tile Assembly Model

- **tile type** = unit square
- each side has a **glue** with a **label** and **strength** (0, 1, or 2)
- tiles cannot rotate



strength 0

strength 1 (weak)

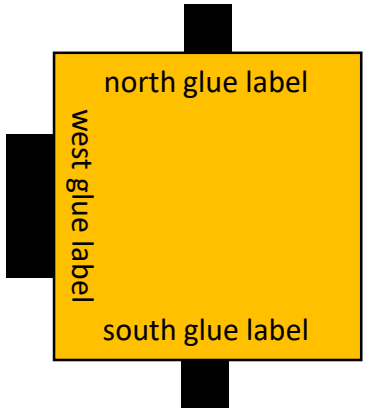
strength 2 (strong)

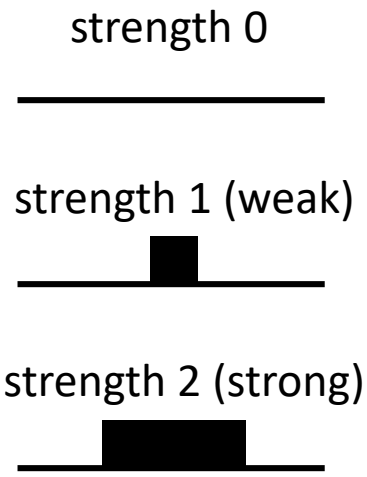
- finitely many tile **types**
- infinitely many **tiles**: copies of each type

Erik Winfree, Ph.D. thesis, Caltech 1998

Abstract Tile Assembly Model

- **tile type** = unit square
- each side has a **glue** with a **label** and **strength** (0, 1, or 2)
- tiles cannot rotate



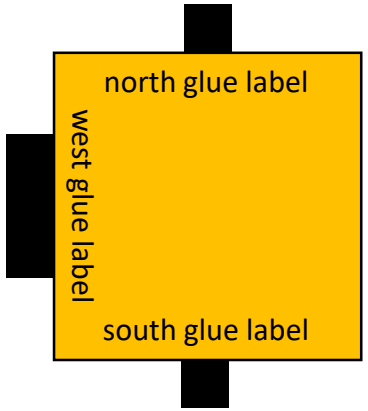


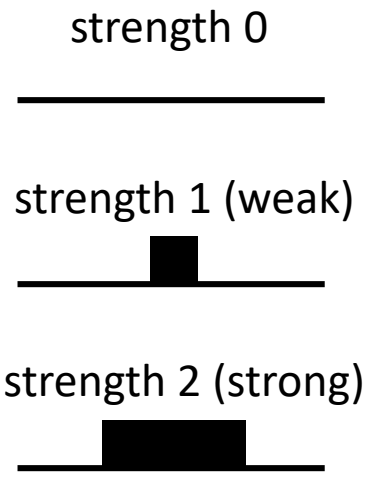
- finitely many tile **types**
- infinitely many **tiles**: copies of each type
- assembly starts as a single copy of a special **seed** tile

Erik Winfree, Ph.D. thesis, Caltech 1998

Abstract Tile Assembly Model

- **tile type** = unit square
- each side has a **glue** with a **label** and **strength** (0, 1, or 2)
- tiles cannot rotate

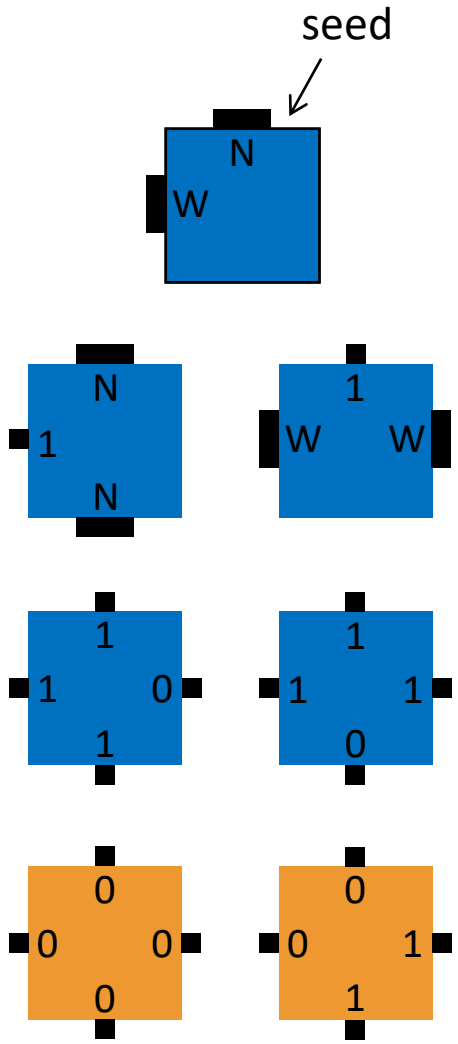




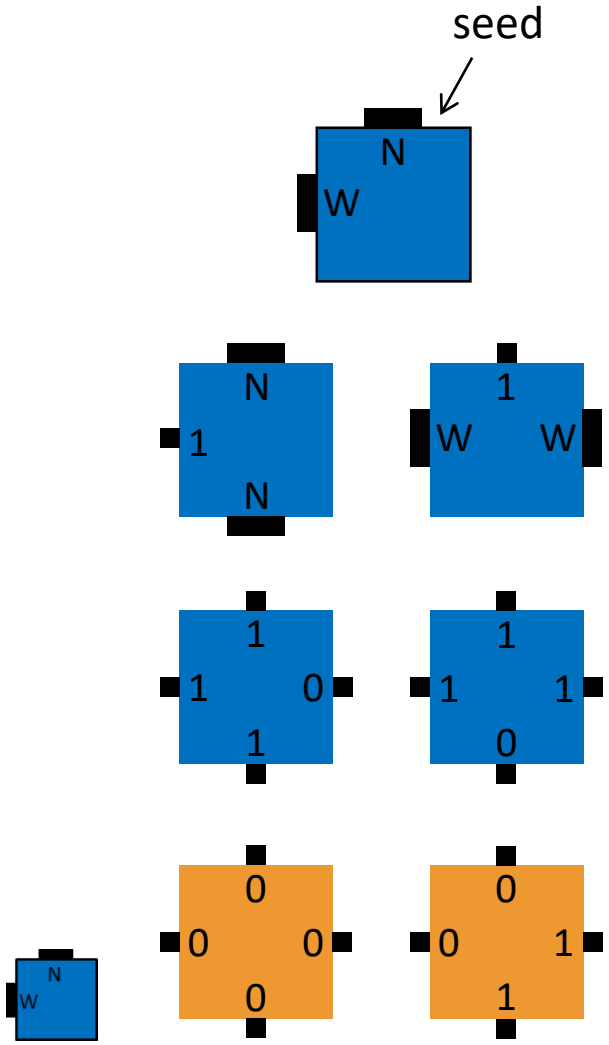
- finitely many tile **types**
- infinitely many **tiles**: copies of each type
- assembly starts as a single copy of a special **seed** tile
- tile can bind to the assembly if total binding strength ≥ 2 (two weak glues or one strong glue)

Erik Winfree, Ph.D. thesis, Caltech 1998

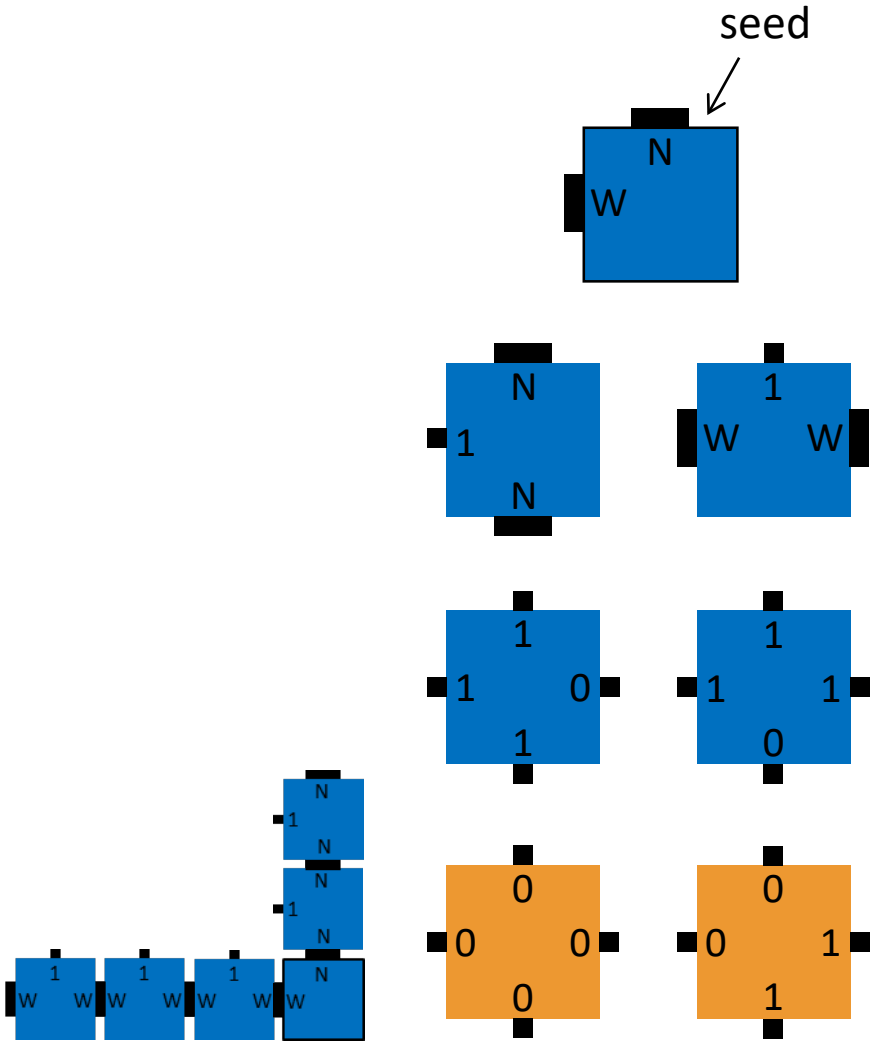
Example tile set



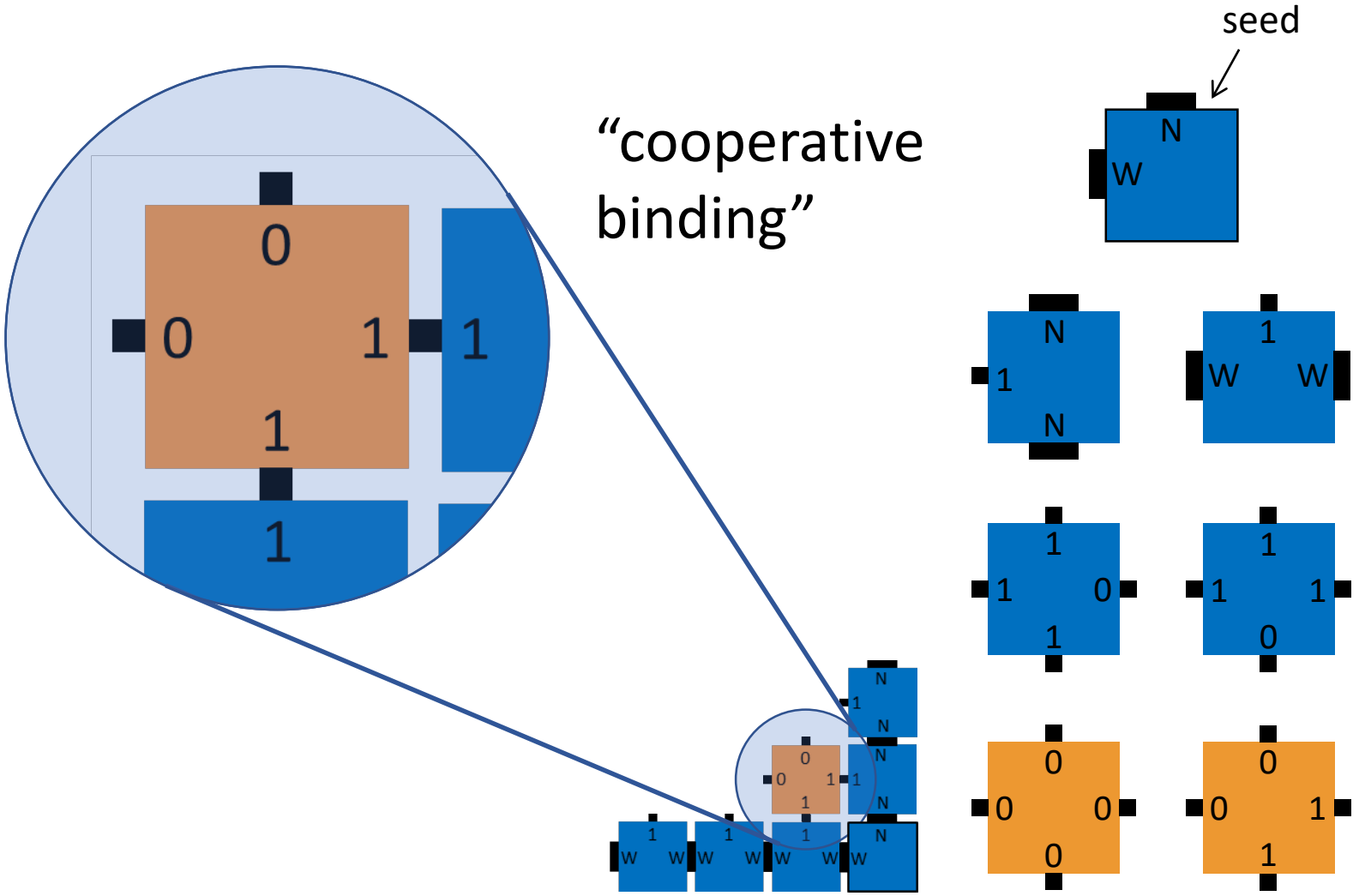
Example tile set



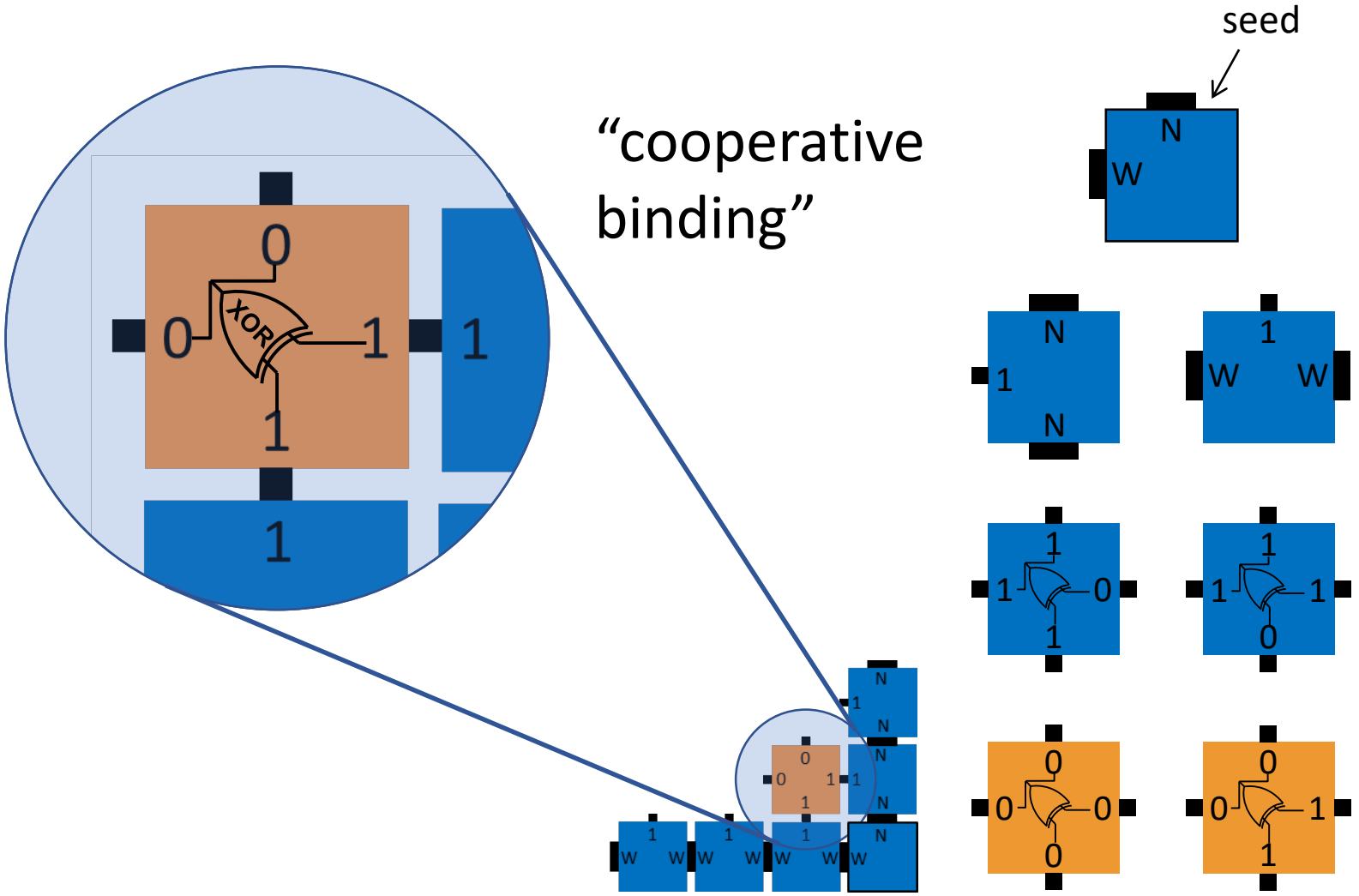
Example tile set



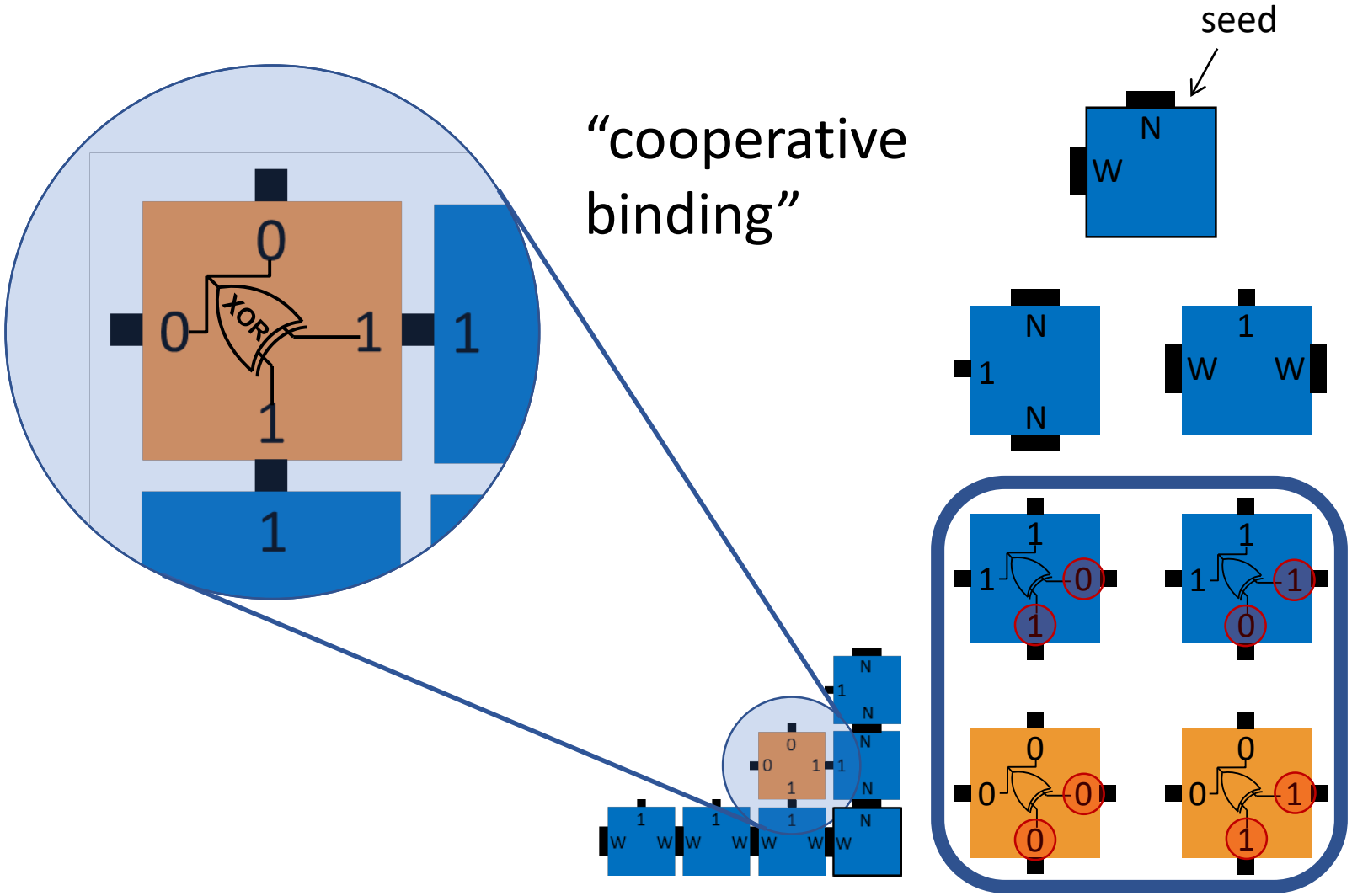
Example tile set



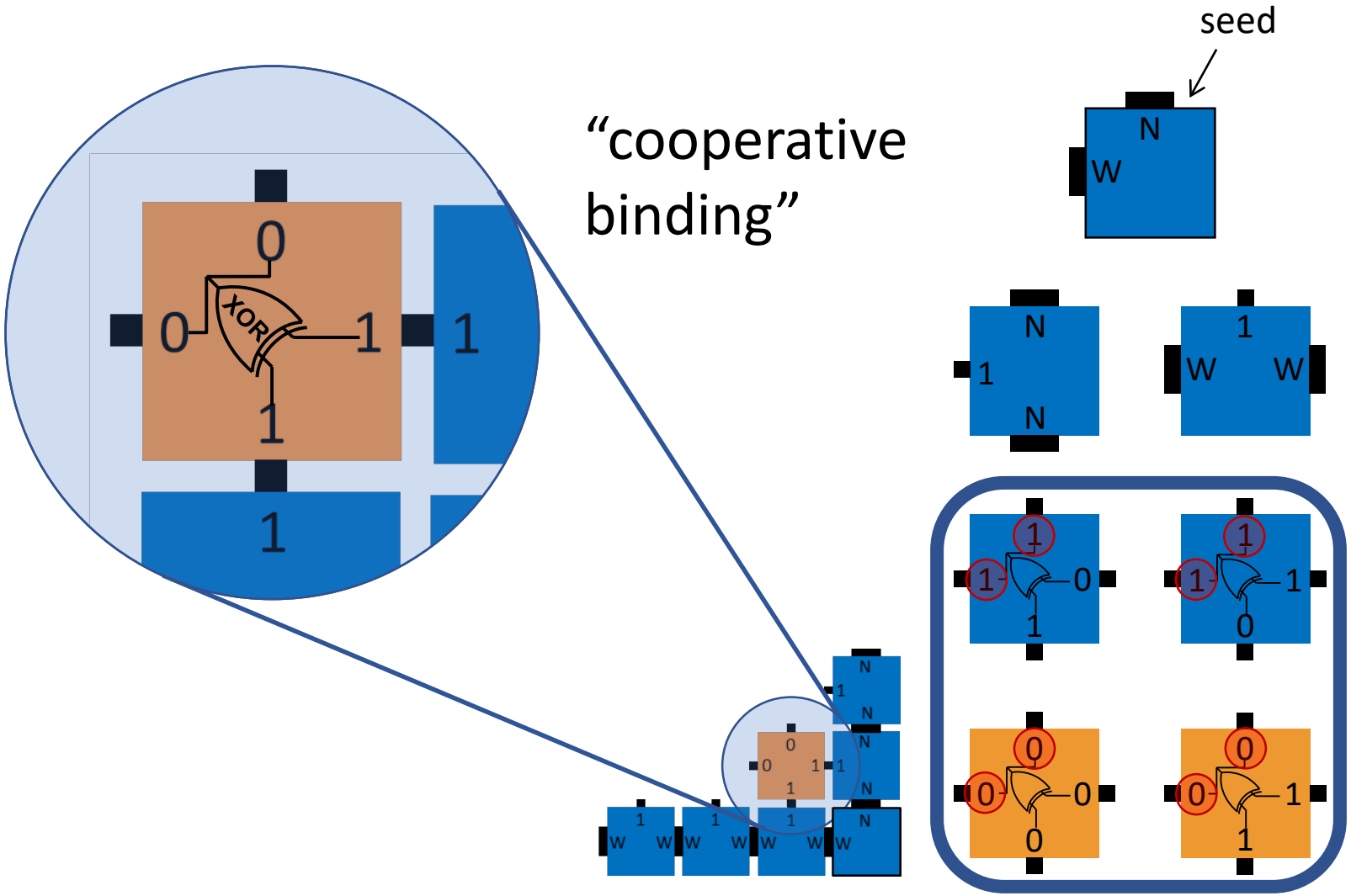
Example tile set



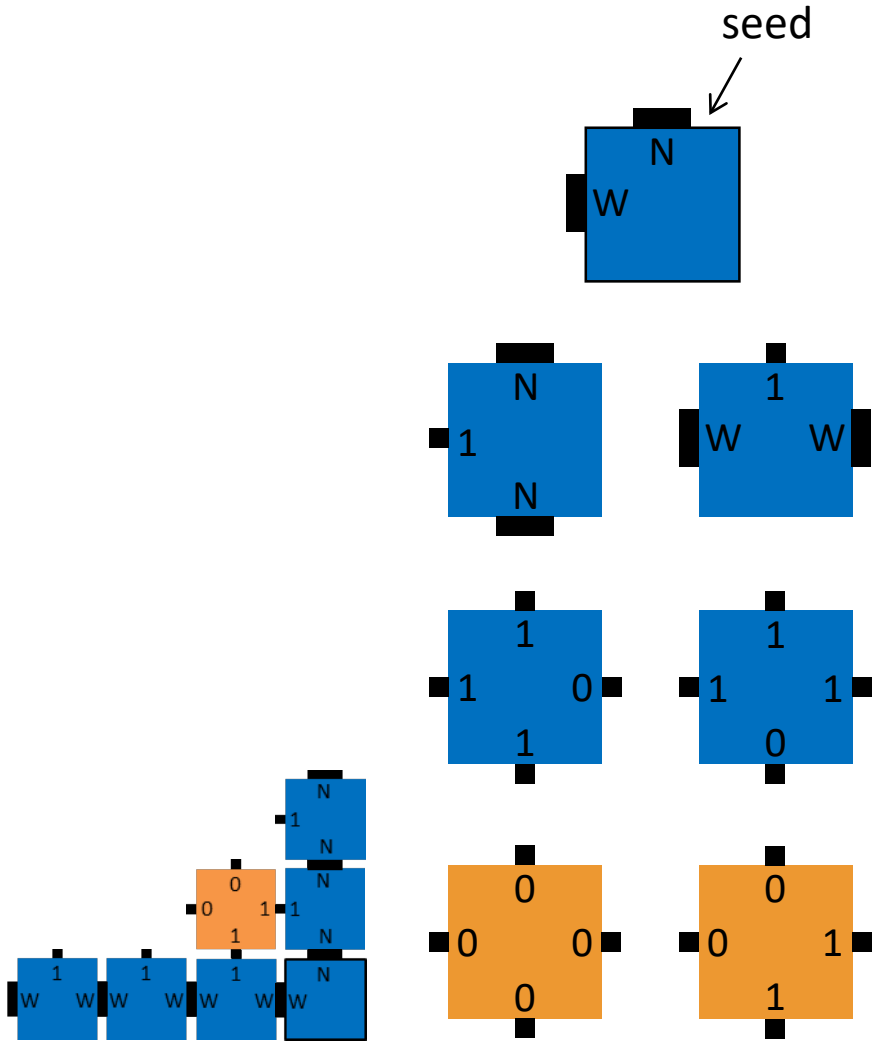
Example tile set

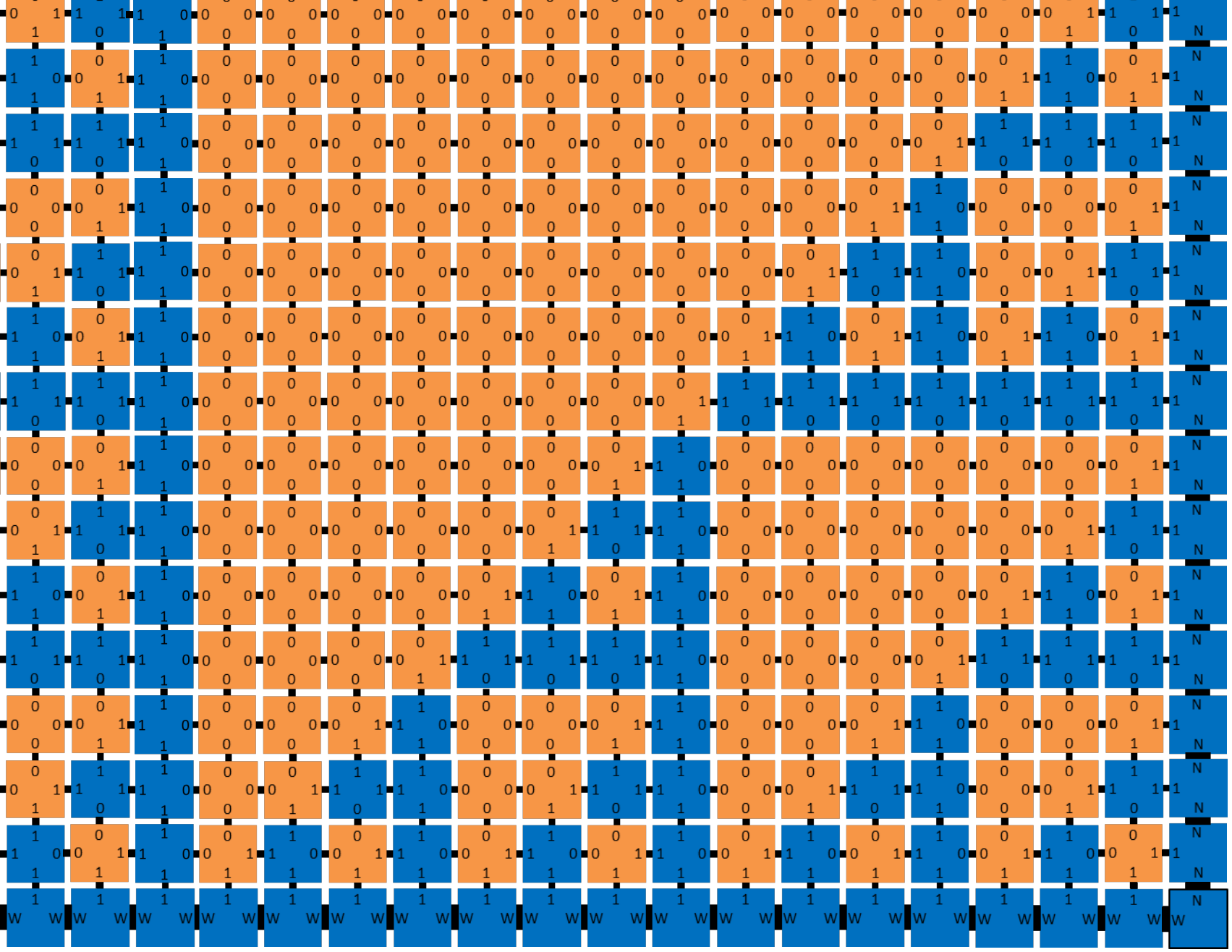


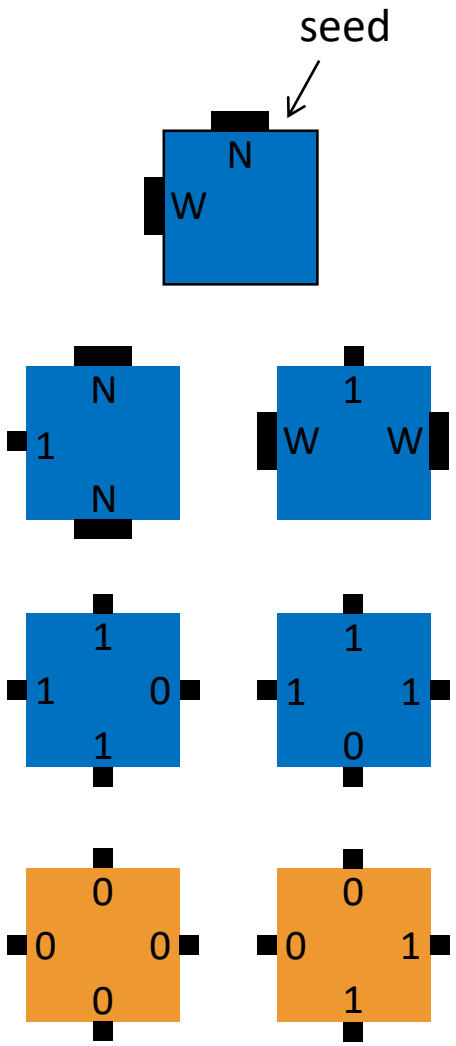
Example tile set

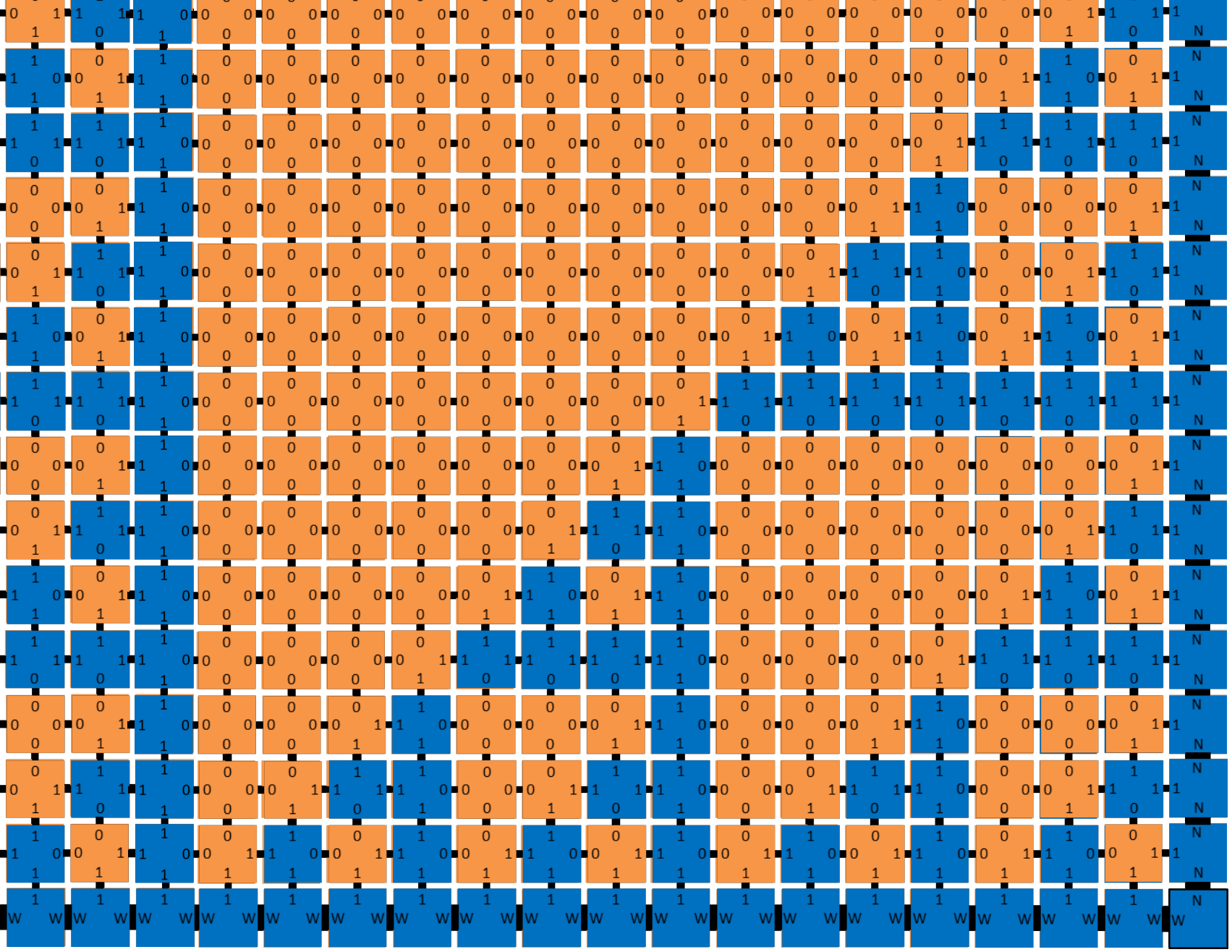


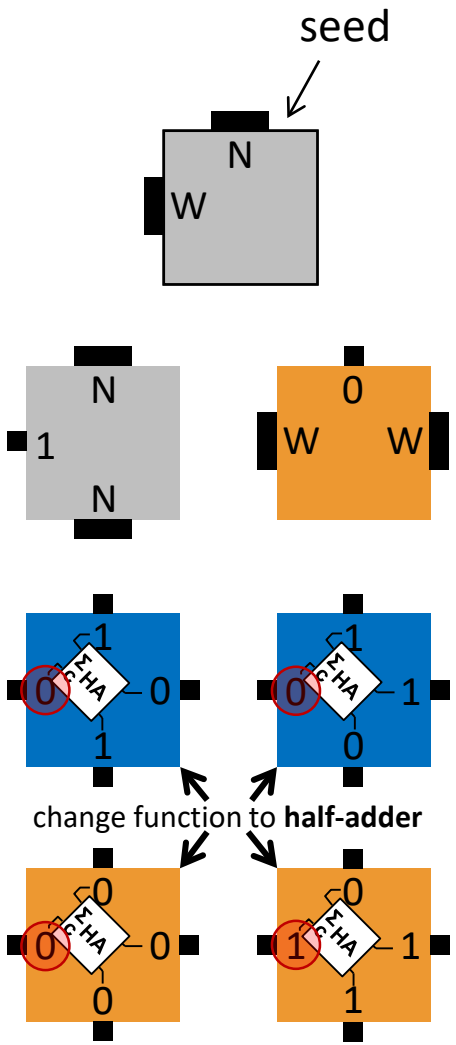
Example tile set

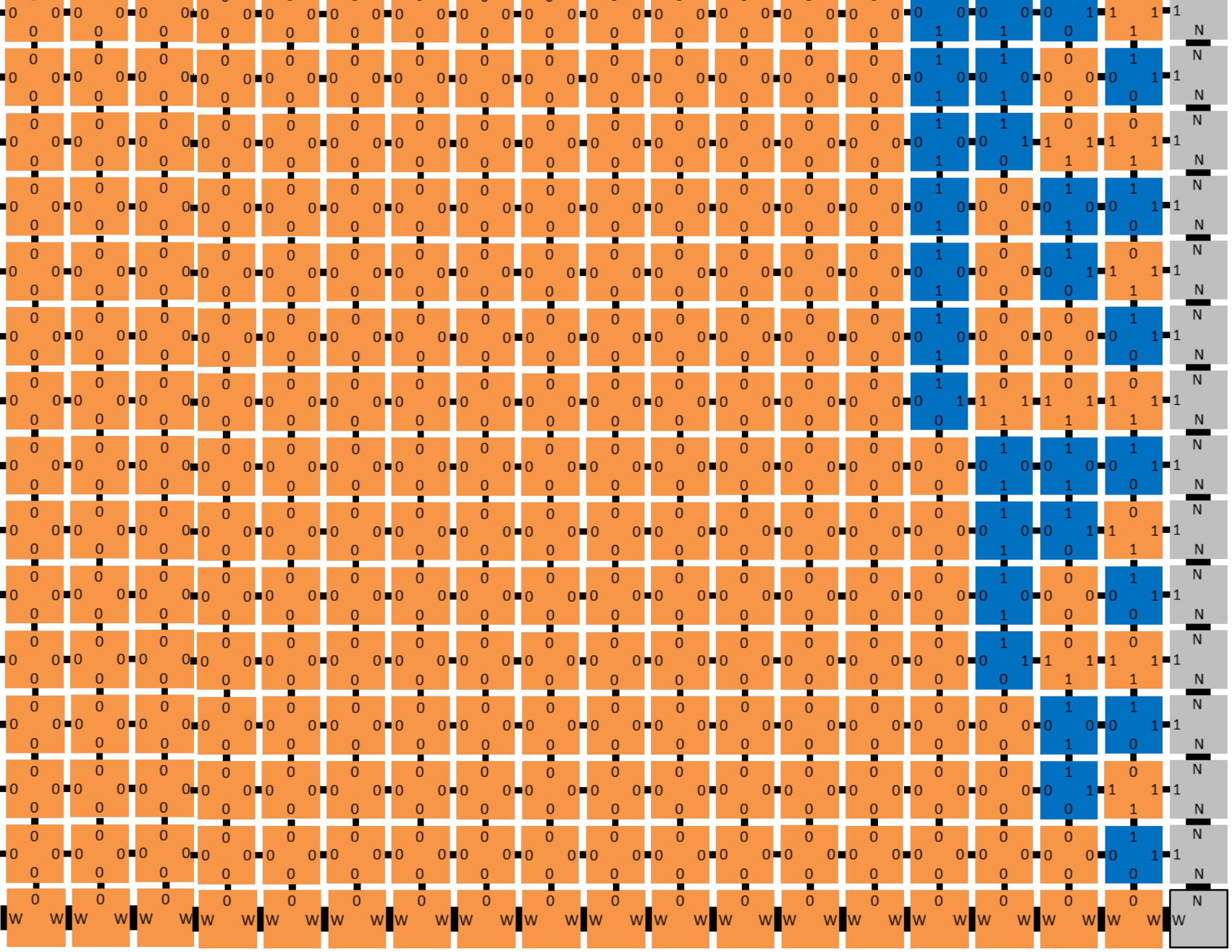


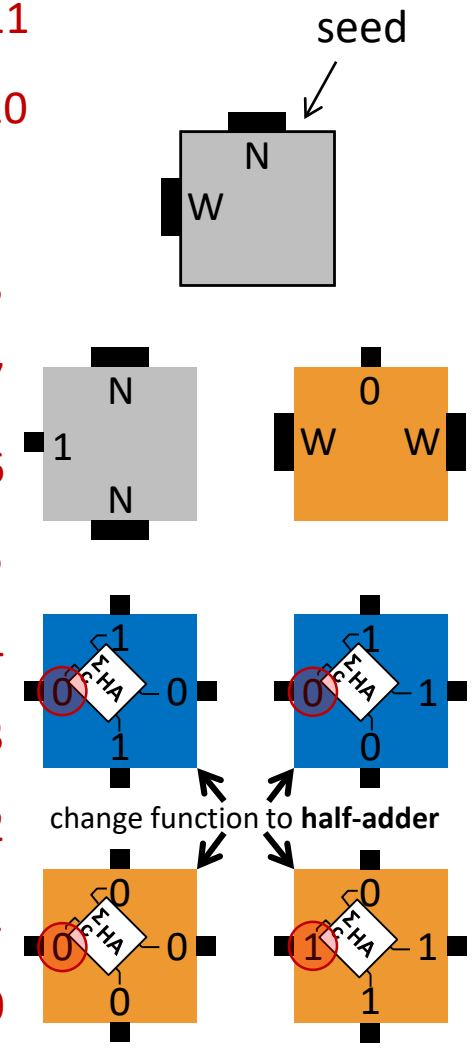



seed
N
W
N
1
N
1
W
W
1
1
0
1
1
1
0
0
0
0
0
1

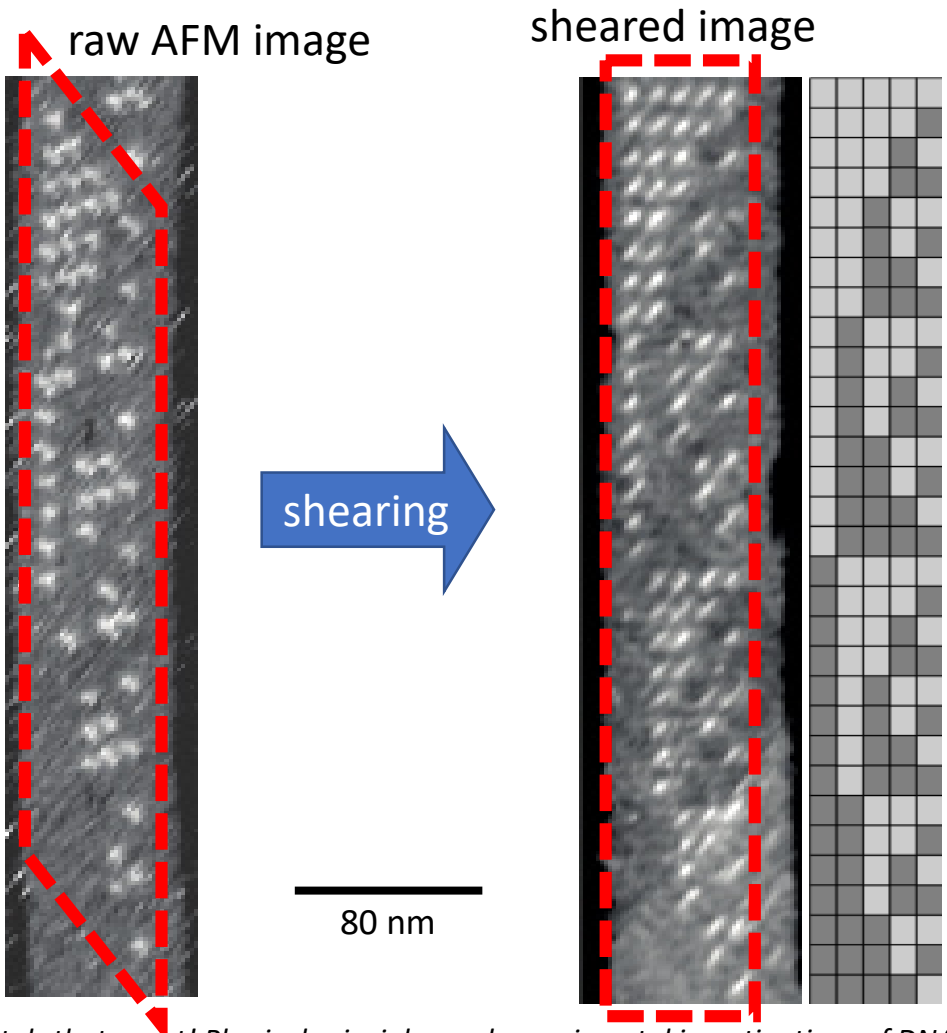



seed
N
W
N
1
N
0
W
W
1
0
0
1
change function to **half-adder**
0
0
0
0
1
1
0
1


14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

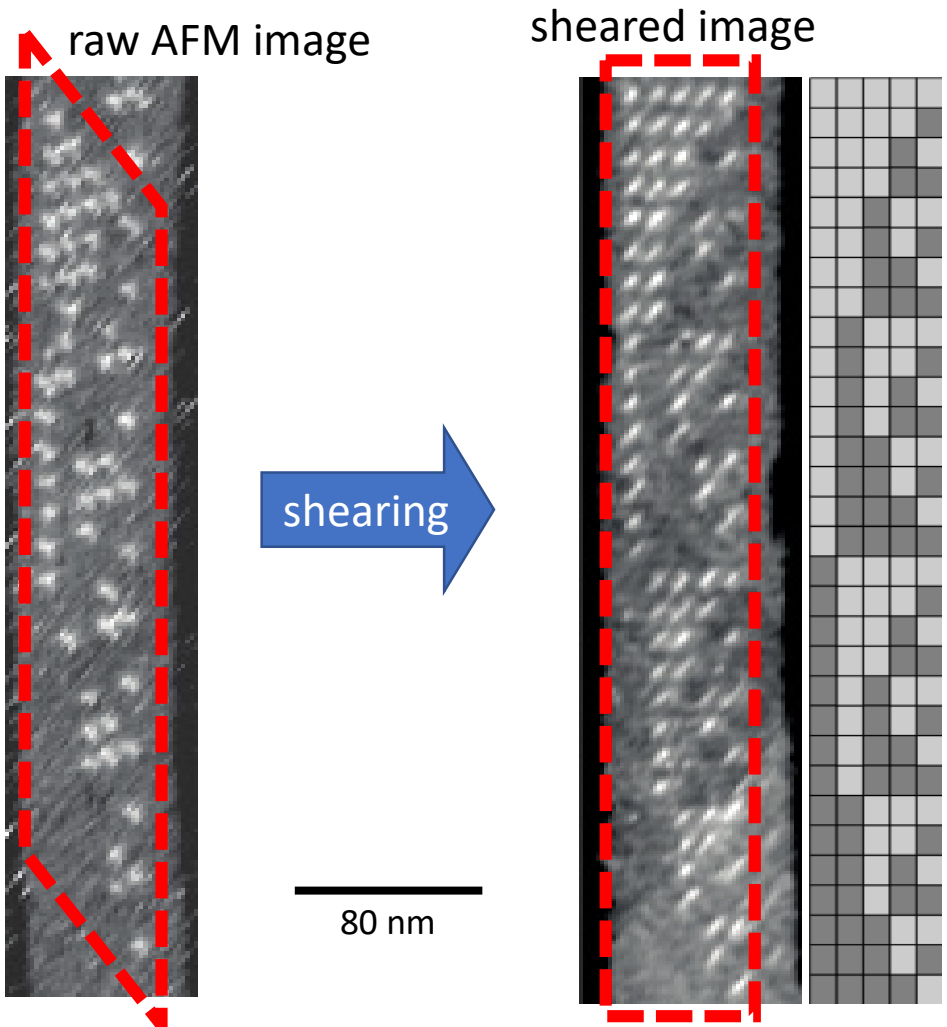

seed
N
W
N
1
N
0
W
W
1
0
0
1
1
0
change function to **half-adder**
0
0
0
0
1
1
1
0

Algorithmic self-assembly in action

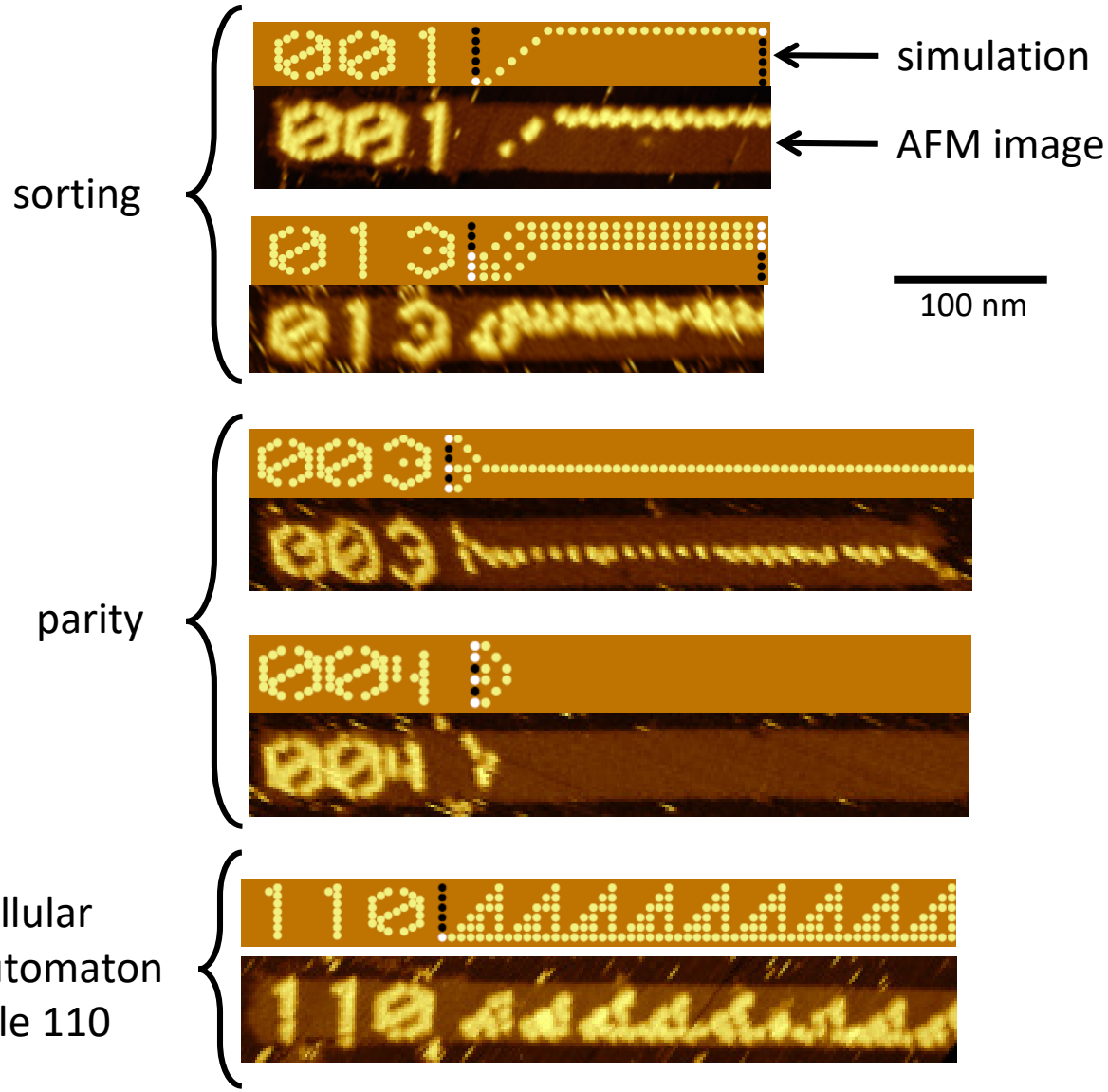


[*Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*, Constantine Evans, Ph.D. thesis, Caltech, 2014]

Algorithmic self-assembly in action



[*Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*, Constantine Evans, Ph.D. thesis, Caltech, 2014]

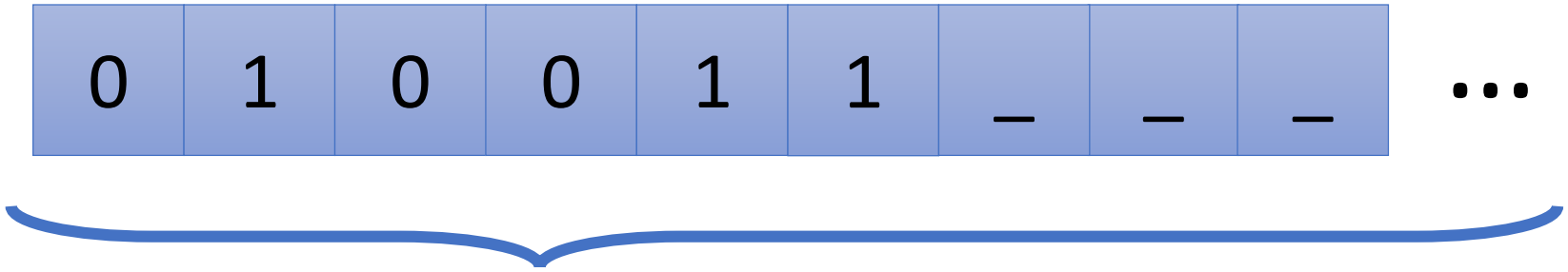


[*Diverse and robust molecular algorithms using reprogrammable DNA self-assembly*. Woods, Doty, Myhrvold, Hui, Wu, Yin, Winfree, submitted]

How computationally powerful are self-assembling tiles?

Turing machines

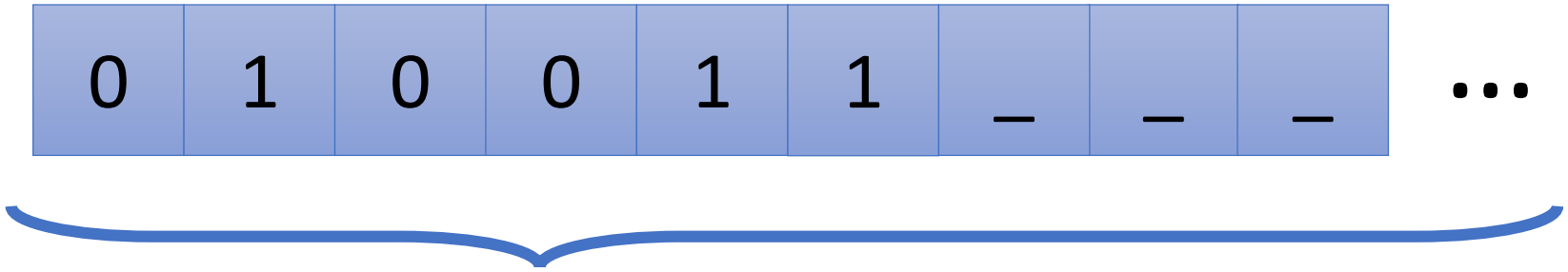
Turing machines



tape ≈ memory

Turing machines

state ≈ line of code

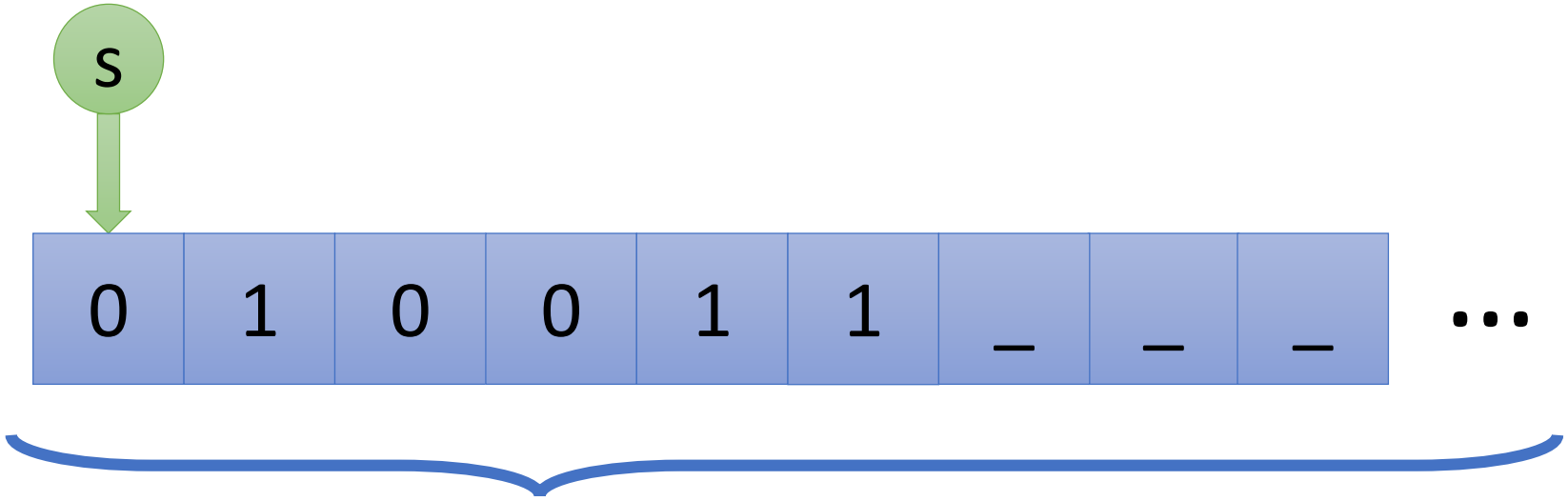


tape ≈ memory

Turing machines

state ≈ line of code

initial state = s



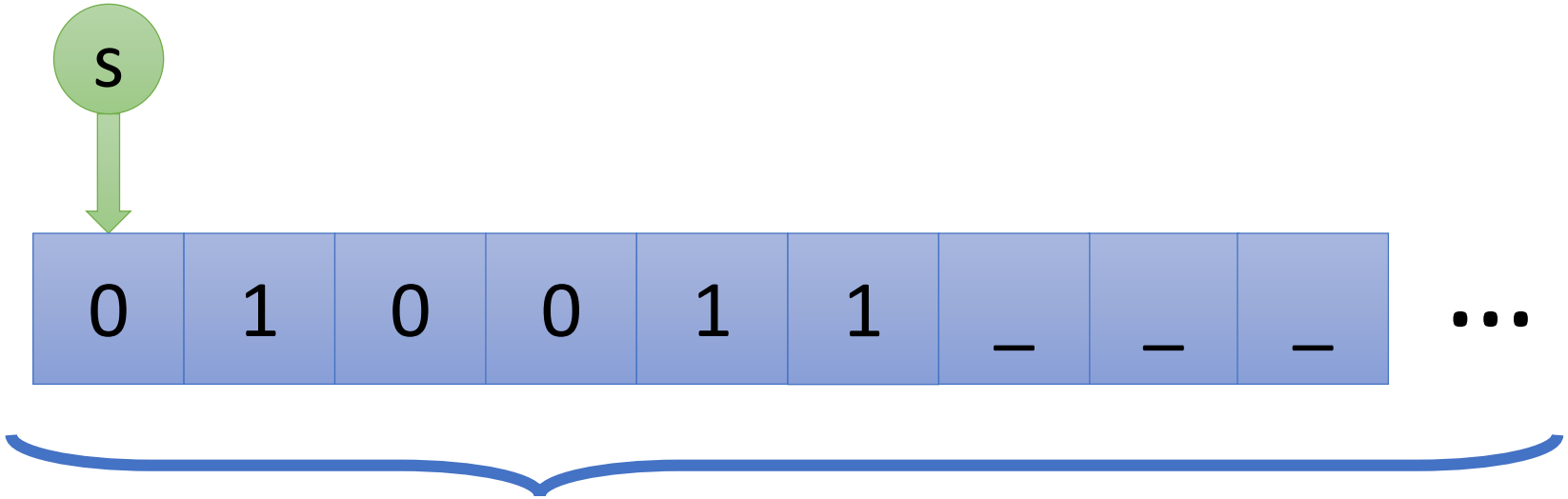
tape ≈ memory

Turing machines

state ≈ line of code

initial state = s

s,0: q,0,→



transitions
(instructions)

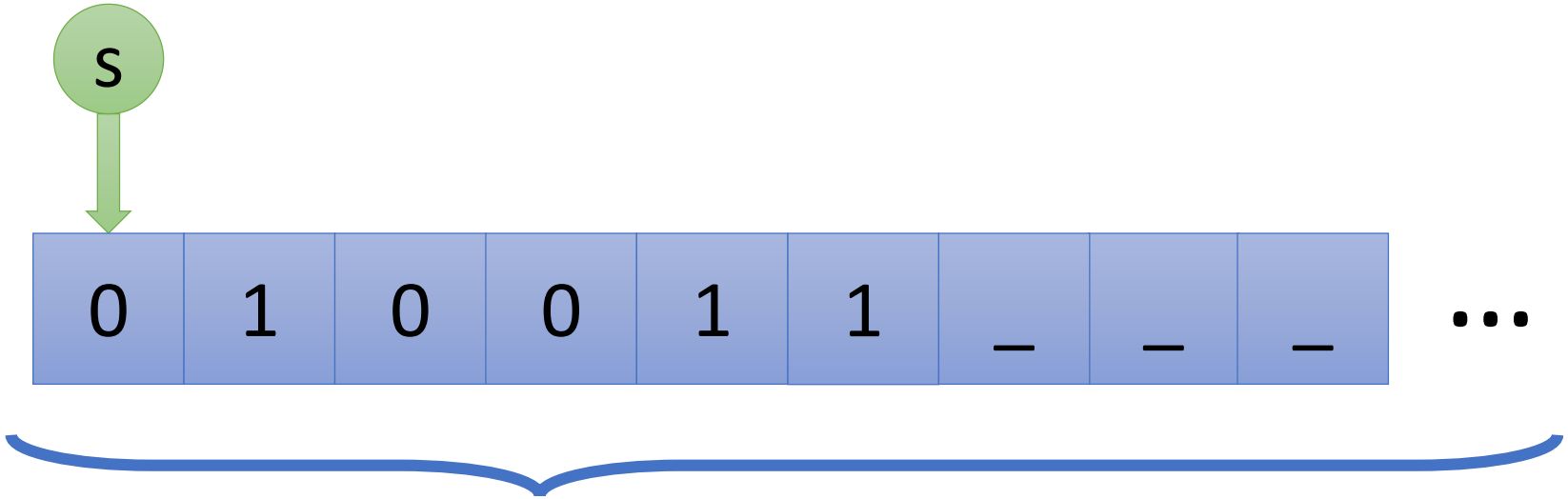
tape ≈ memory

Turing machines

state ≈ line of code

initial state = s

current state



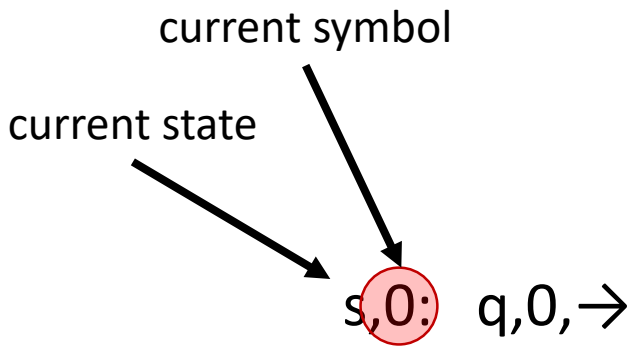
transitions
(instructions)

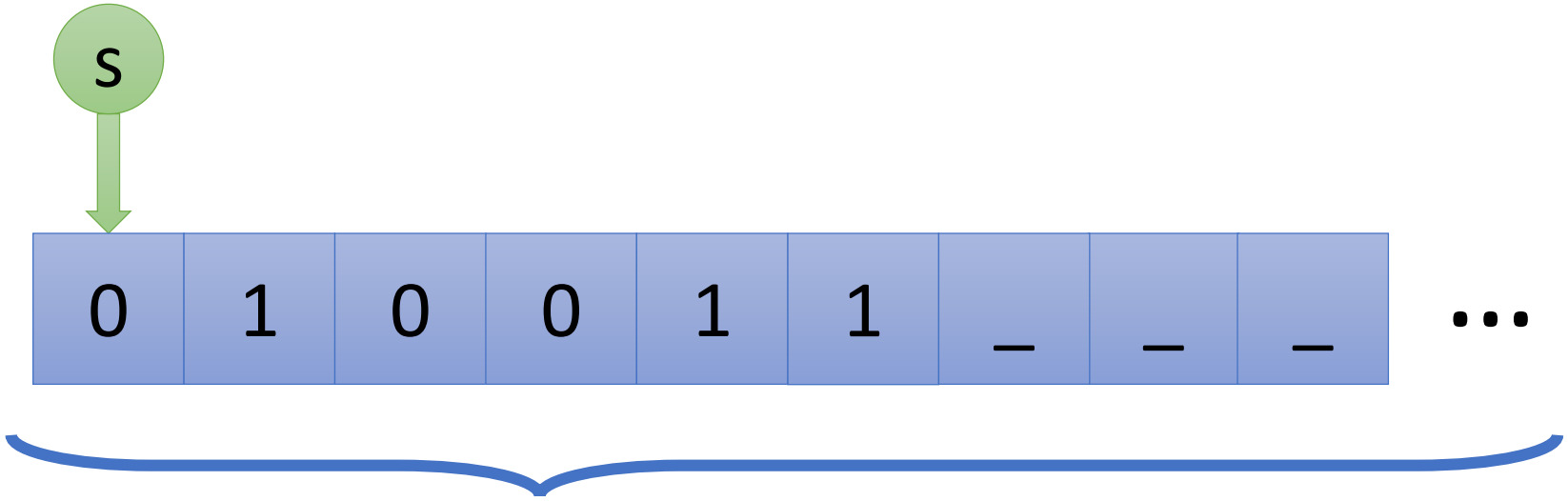
tape ≈ memory

Turing machines

state ≈ line of code

initial state = s





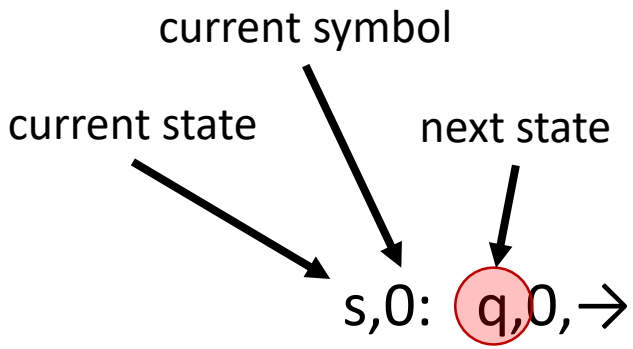
transitions
(instructions)

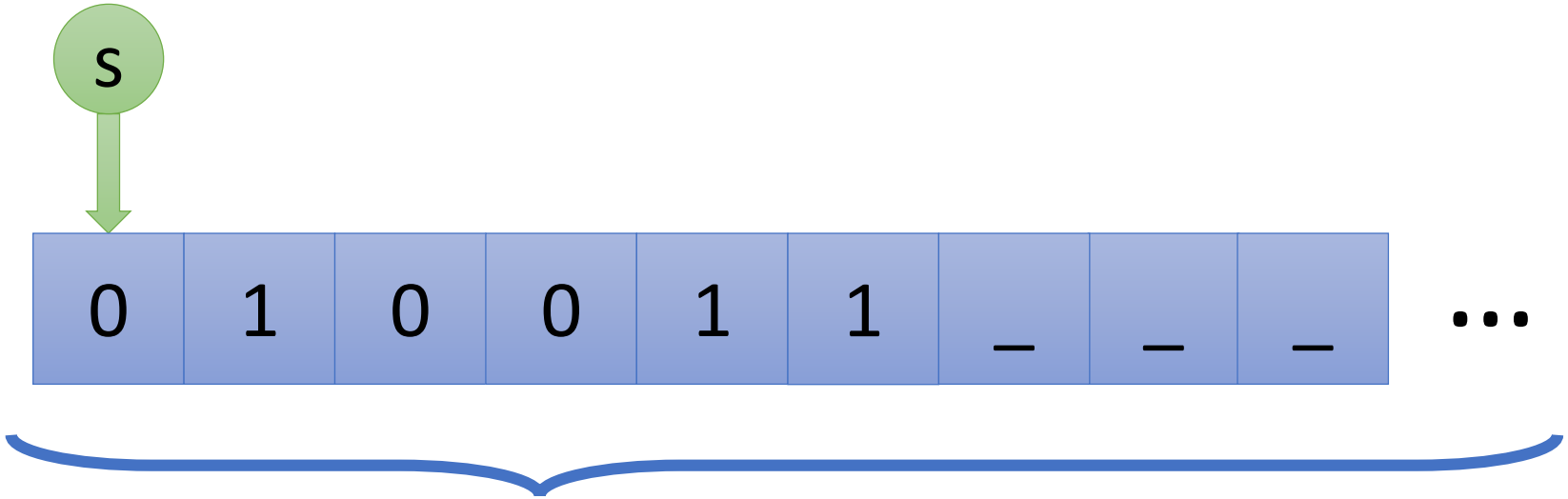
tape ≈ memory

Turing machines

state ≈ line of code

initial state = s





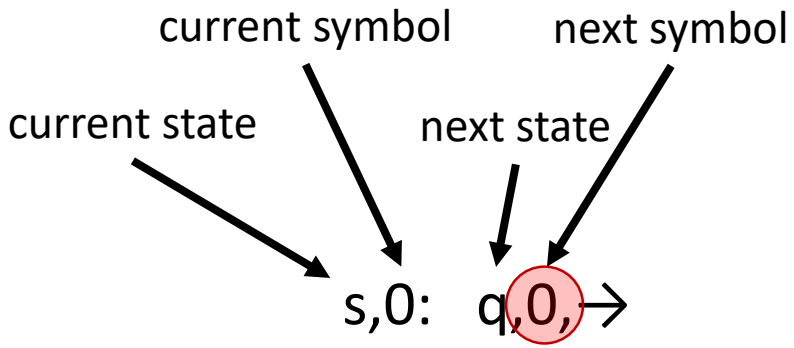
transitions
(instructions)

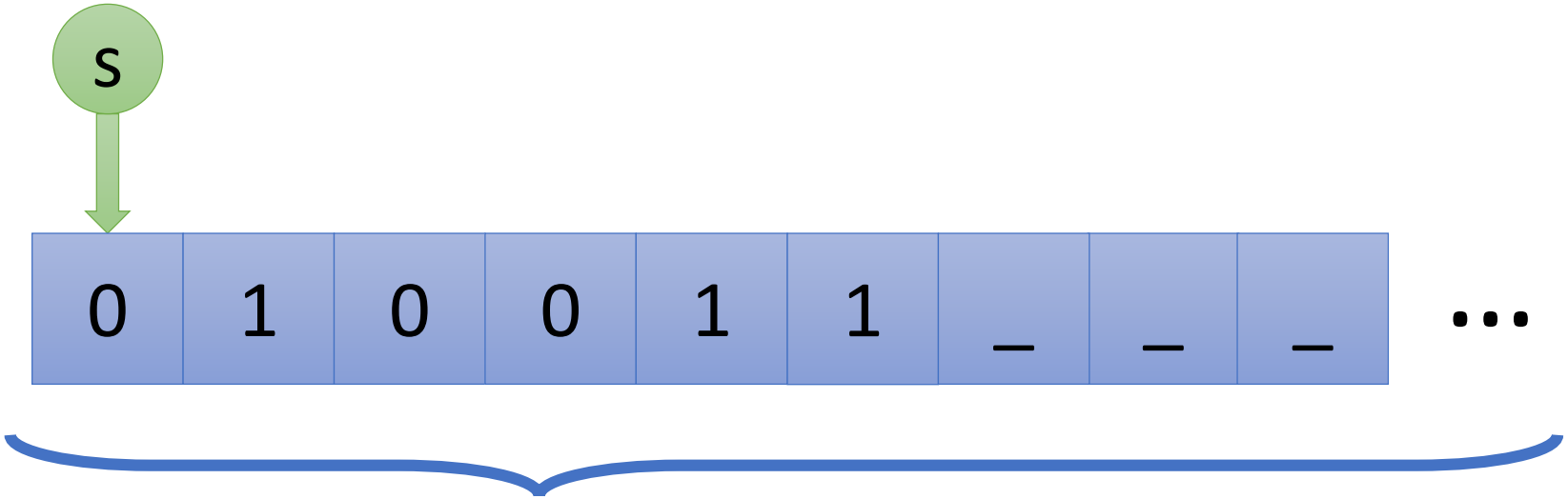
tape ≈ memory

Turing machines

state ≈ line of code

initial state = s





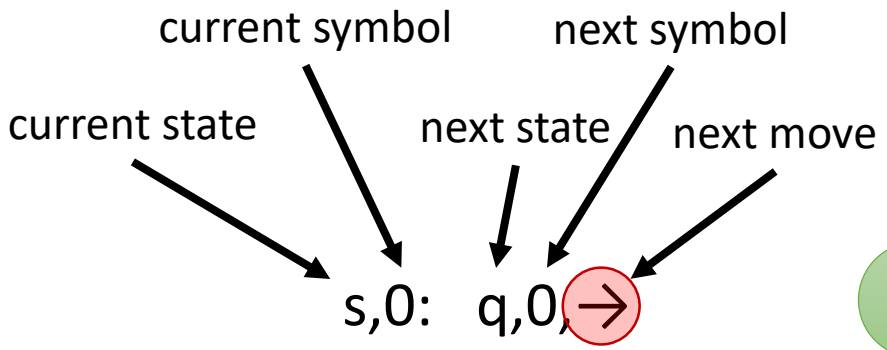
transitions
(instructions)

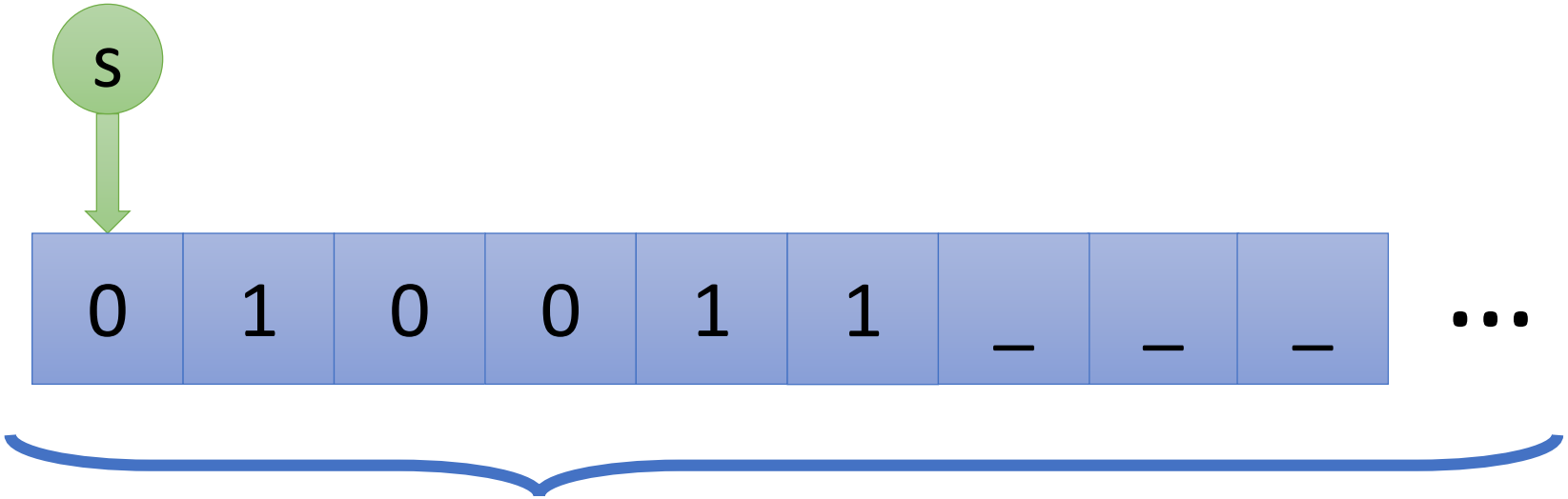
tape ≈ memory

Turing machines

state ≈ line of code

initial state = s





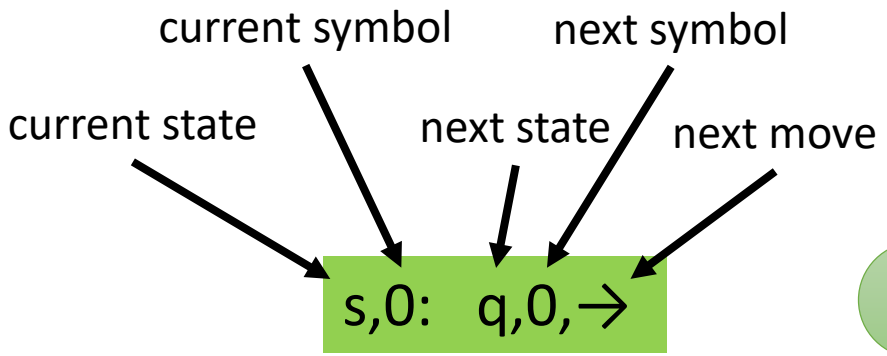
transitions
(instructions)

tape ≈ memory

Turing machines

state ≈ line of code

initial state = s



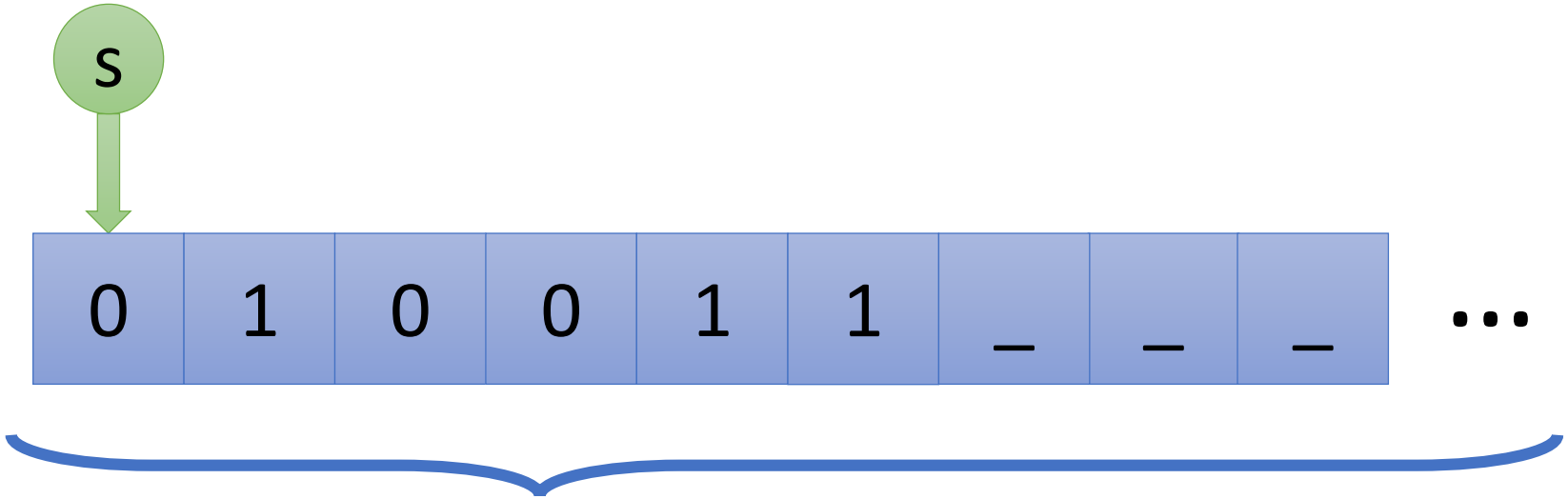
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

transitions
(instructions)

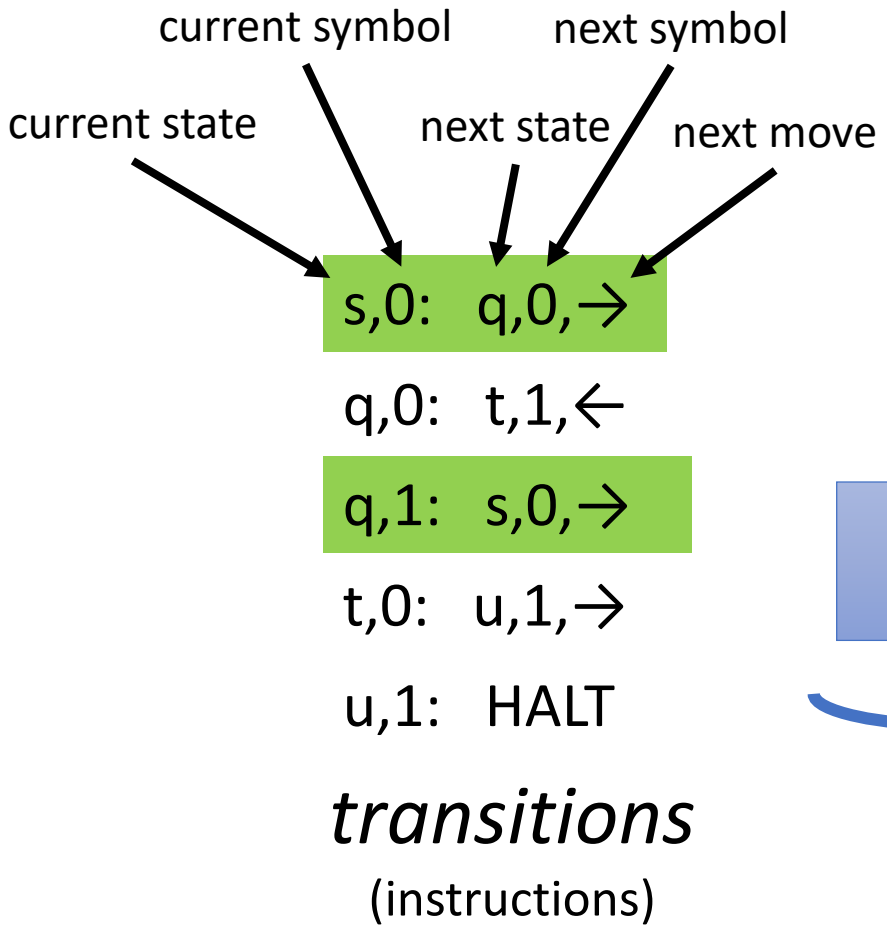


tape ≈ memory

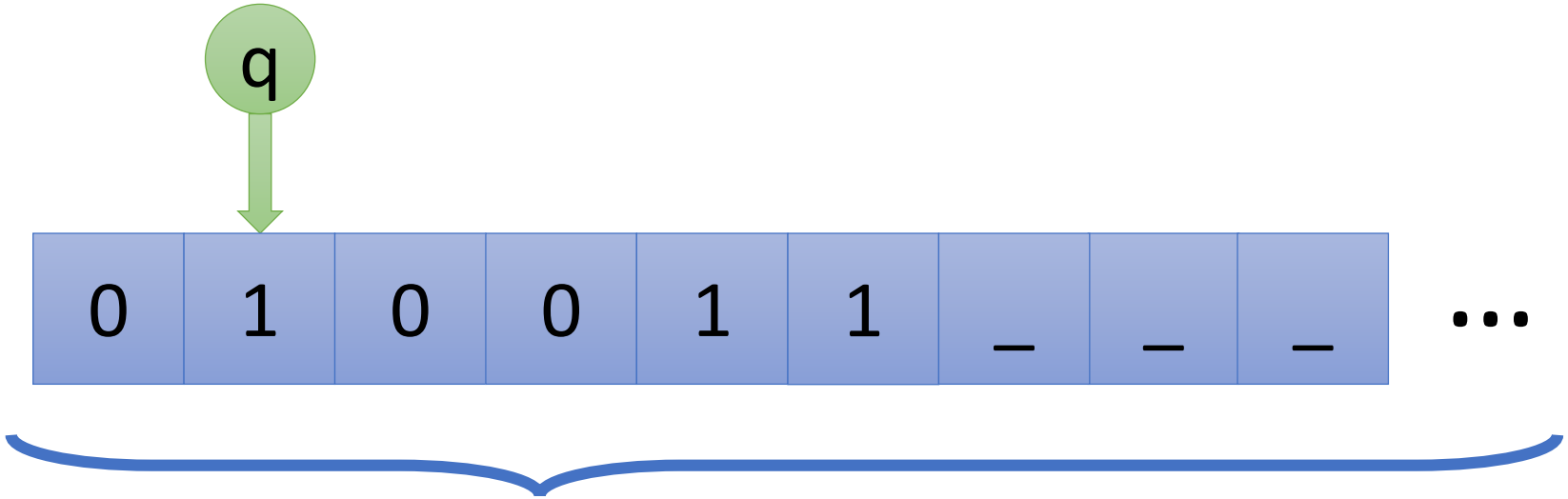
Turing machines

state ≈ line of code

initial state = s

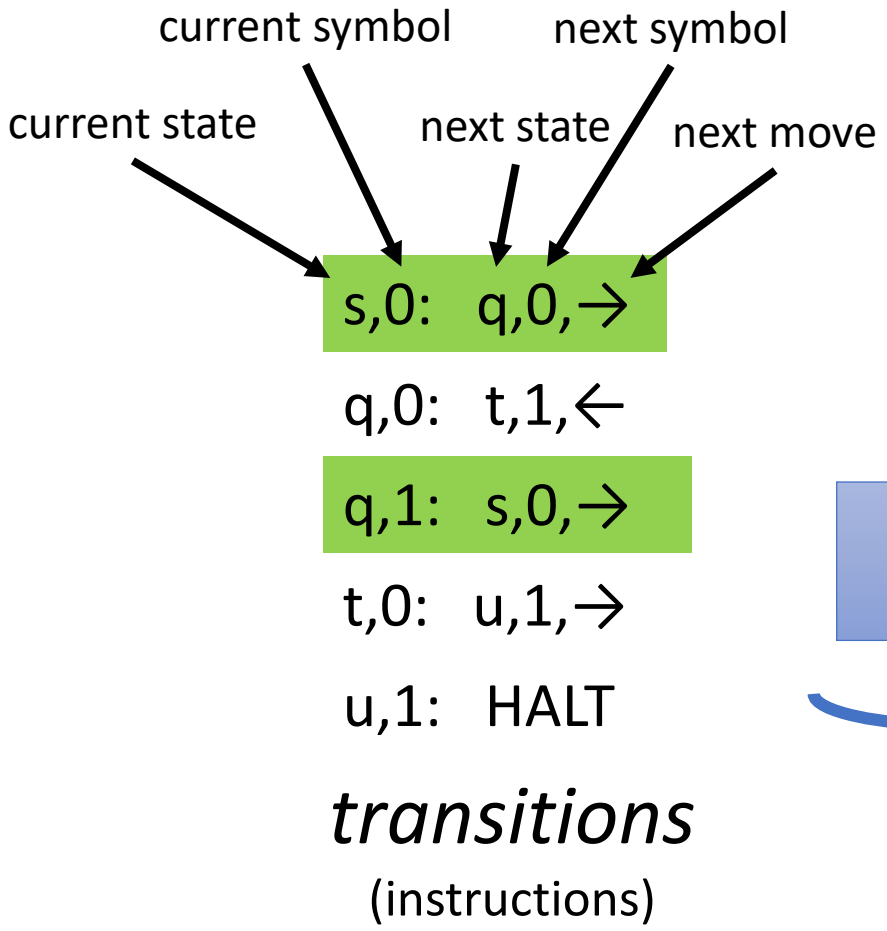


transitions
(instructions)



tape ≈ memory

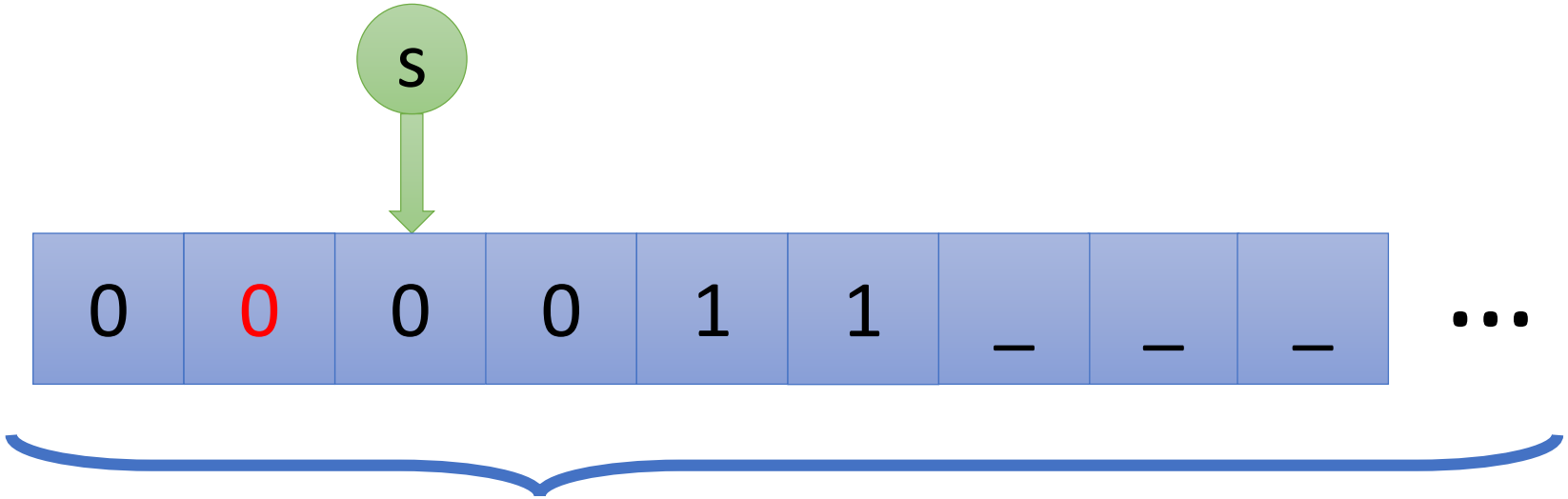
Turing machines



transitions
(instructions)

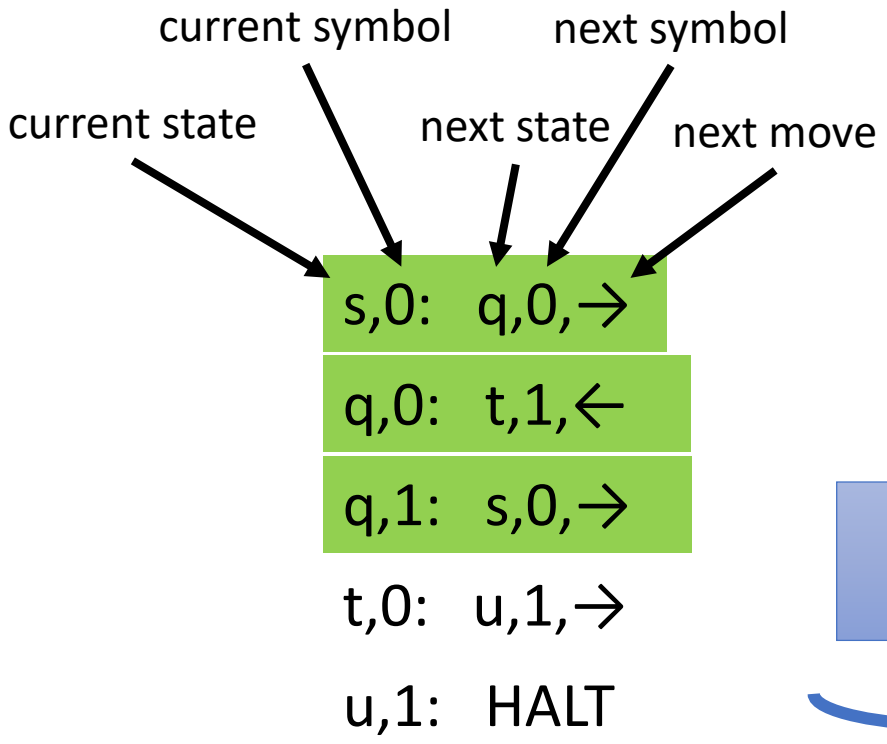
state ≈ line of code

initial state = s



tape ≈ memory

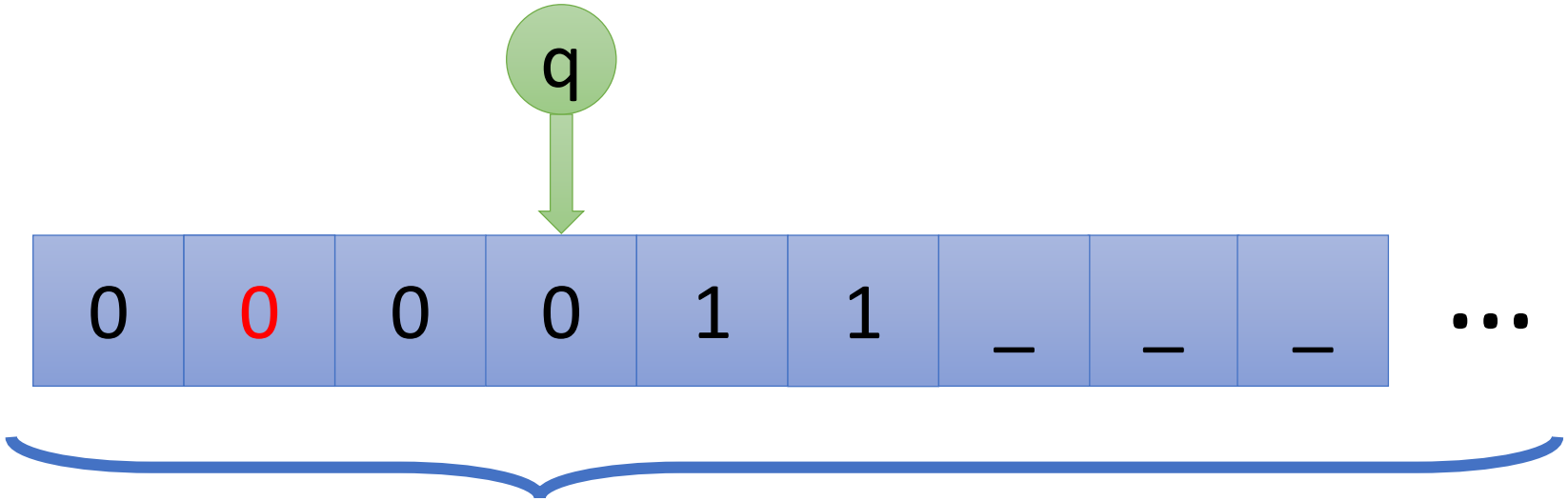
Turing machines



transitions
(instructions)

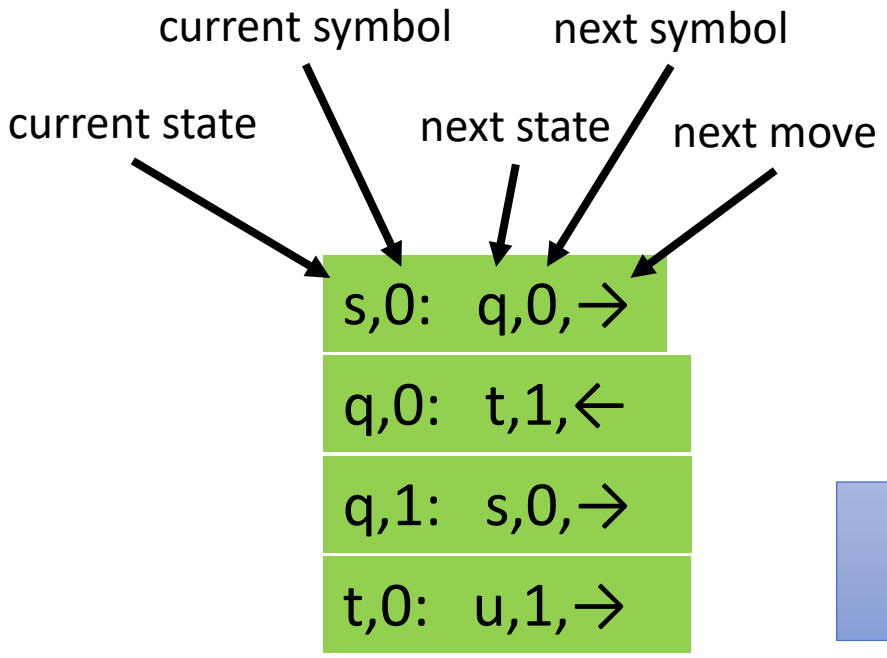
state ≈ line of code

initial state = s



tape ≈ memory

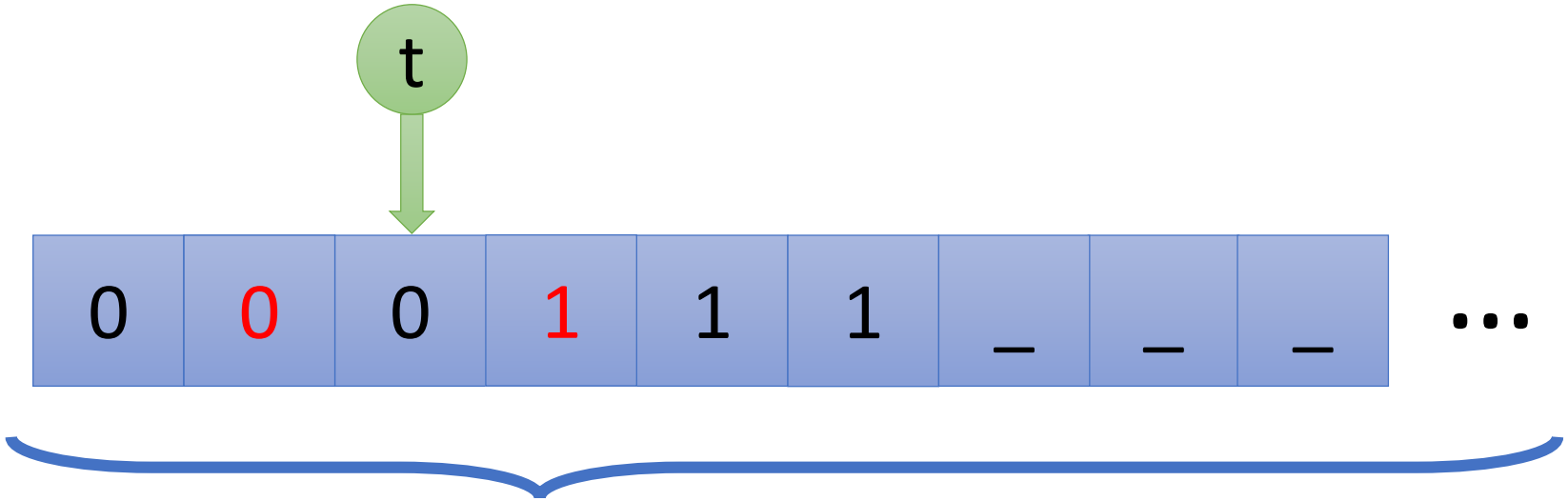
Turing machines



transitions
(instructions)

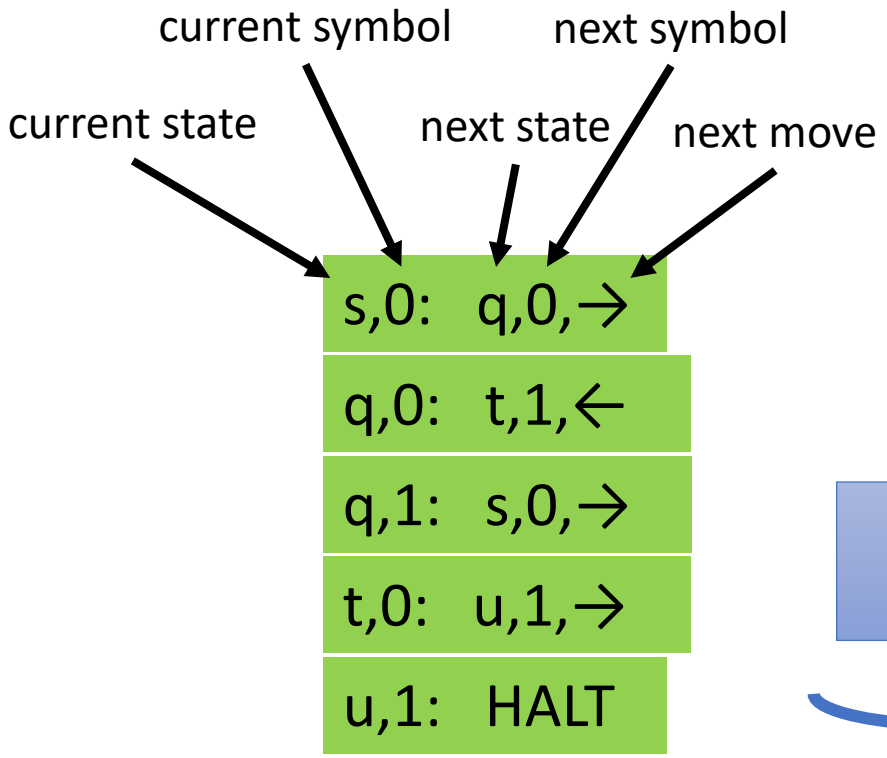
state ≈ line of code

initial state = s



tape ≈ memory

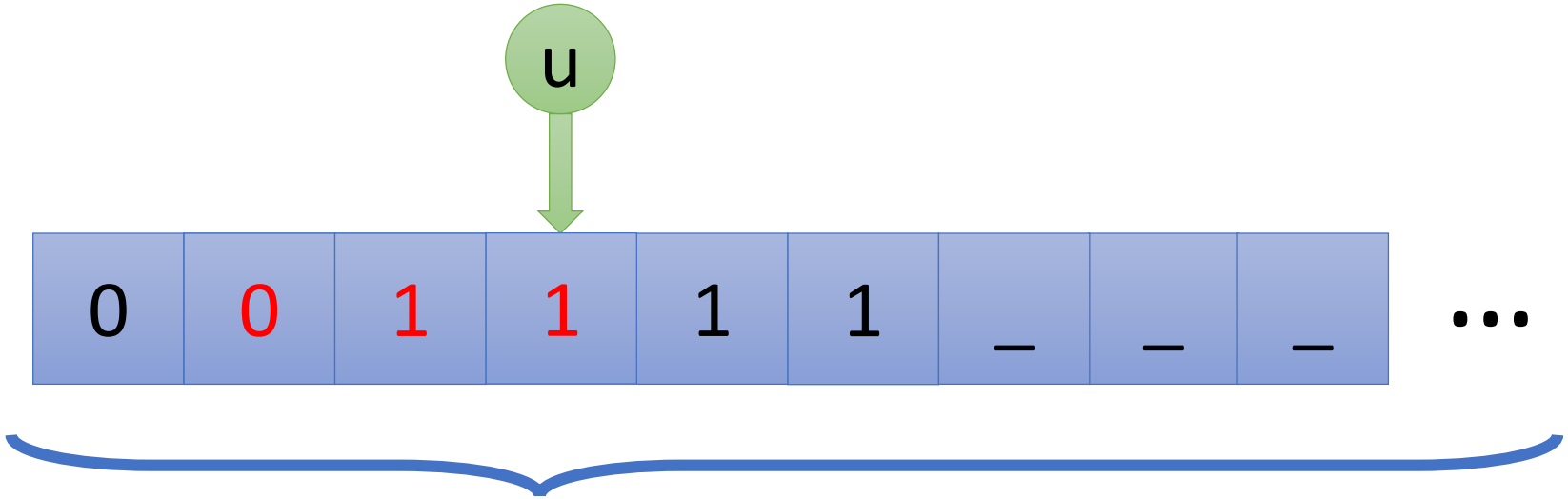
Turing machines



transitions
(instructions)

state ≈ line of code

initial state = s



tape ≈ memory

Tile assembly is Turing-universal

Tile assembly is Turing-universal

s,0: q,0,→

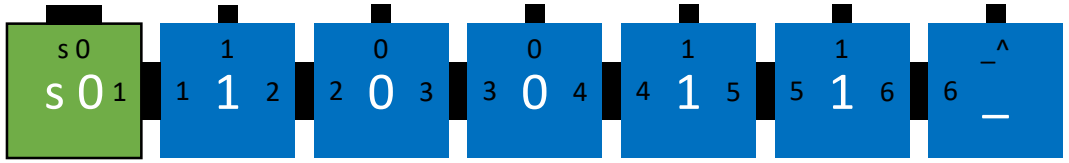
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

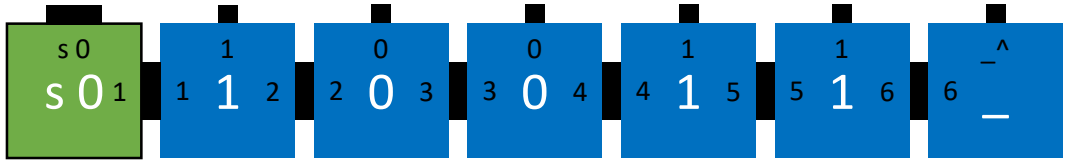
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

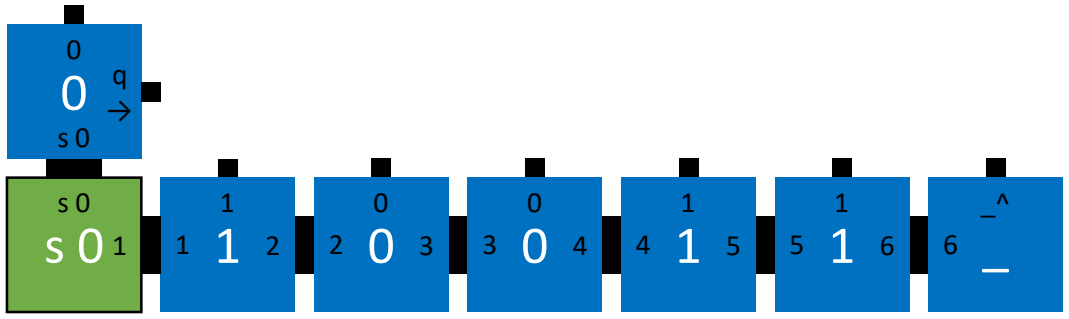
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

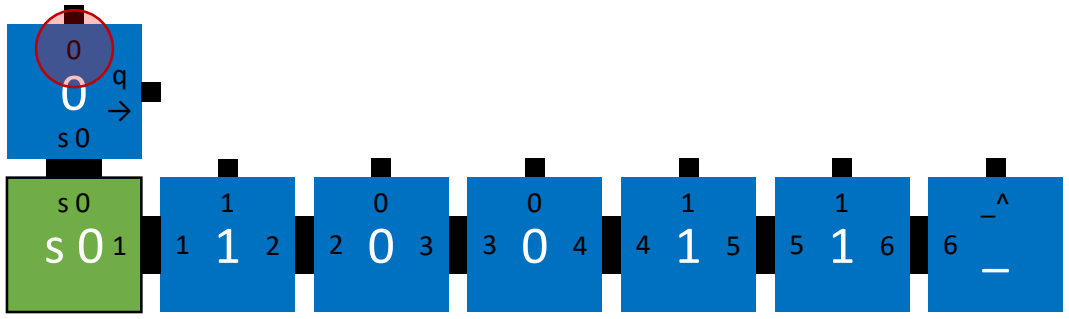
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

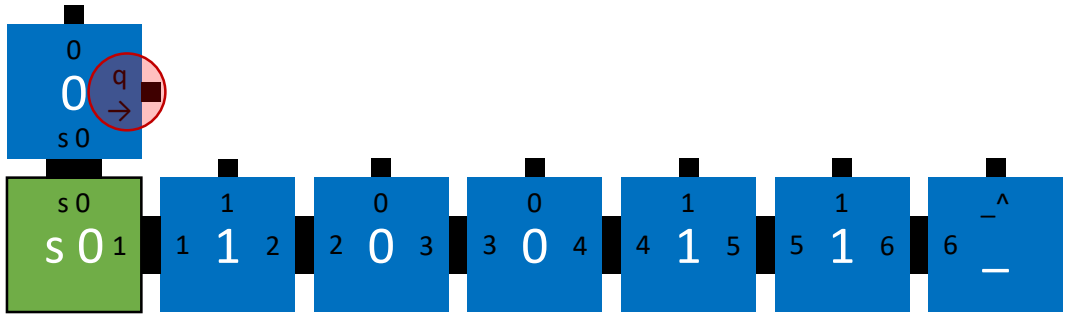
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

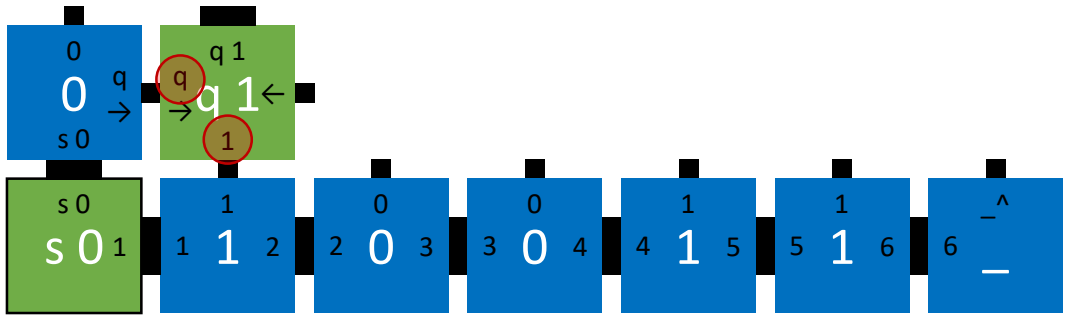
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

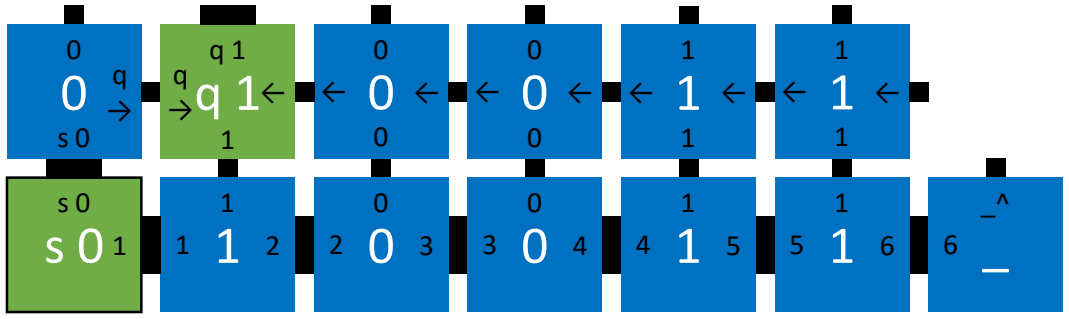
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

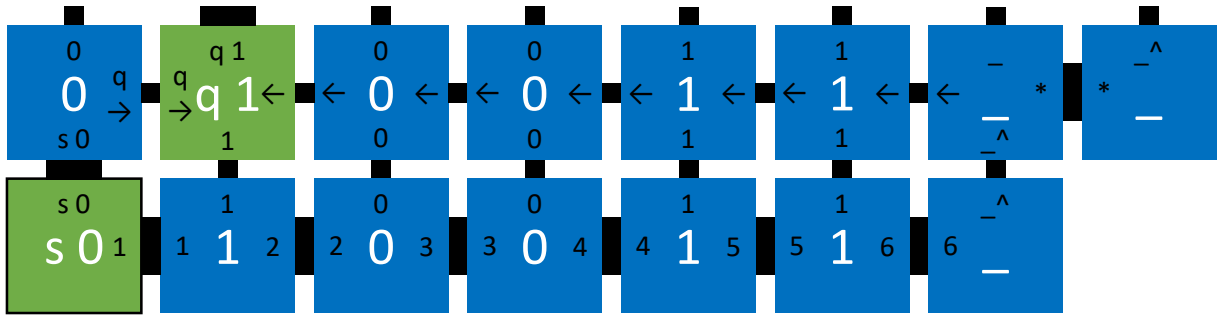
q,0: t,1,←

q,1: s,0,→

t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



s,0: q,0,→

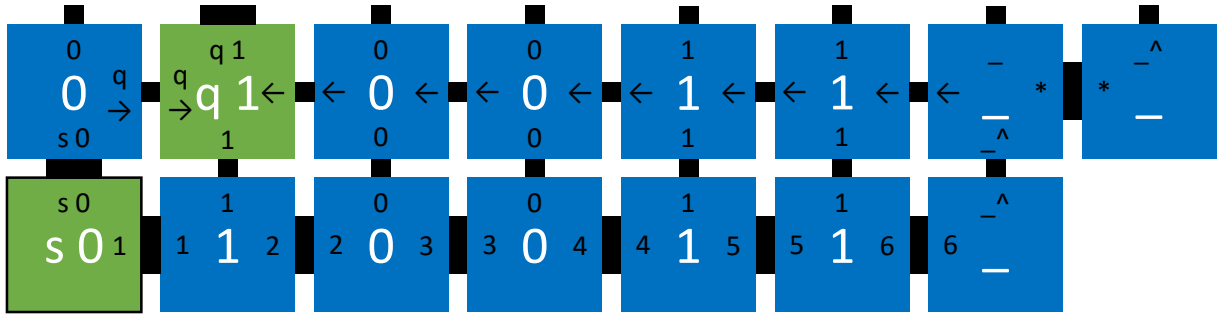
q,0: t,1,←

q,1: s,0,→

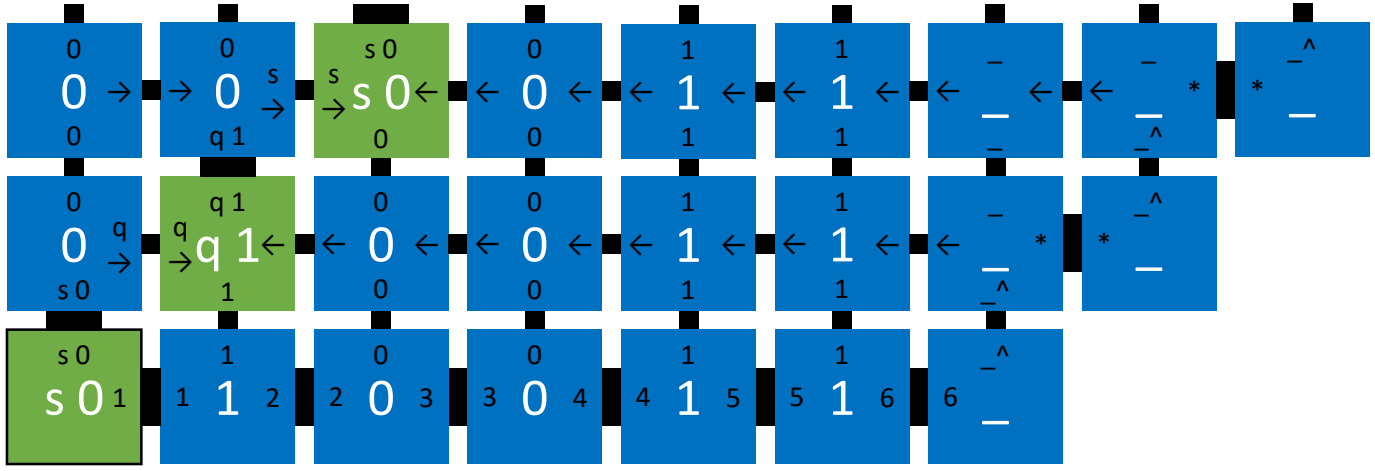
t,0: u,1,→

u,1: HALT

Tile assembly is Turing-universal



Tile assembly is Turing-universal



s,0: q,0,→

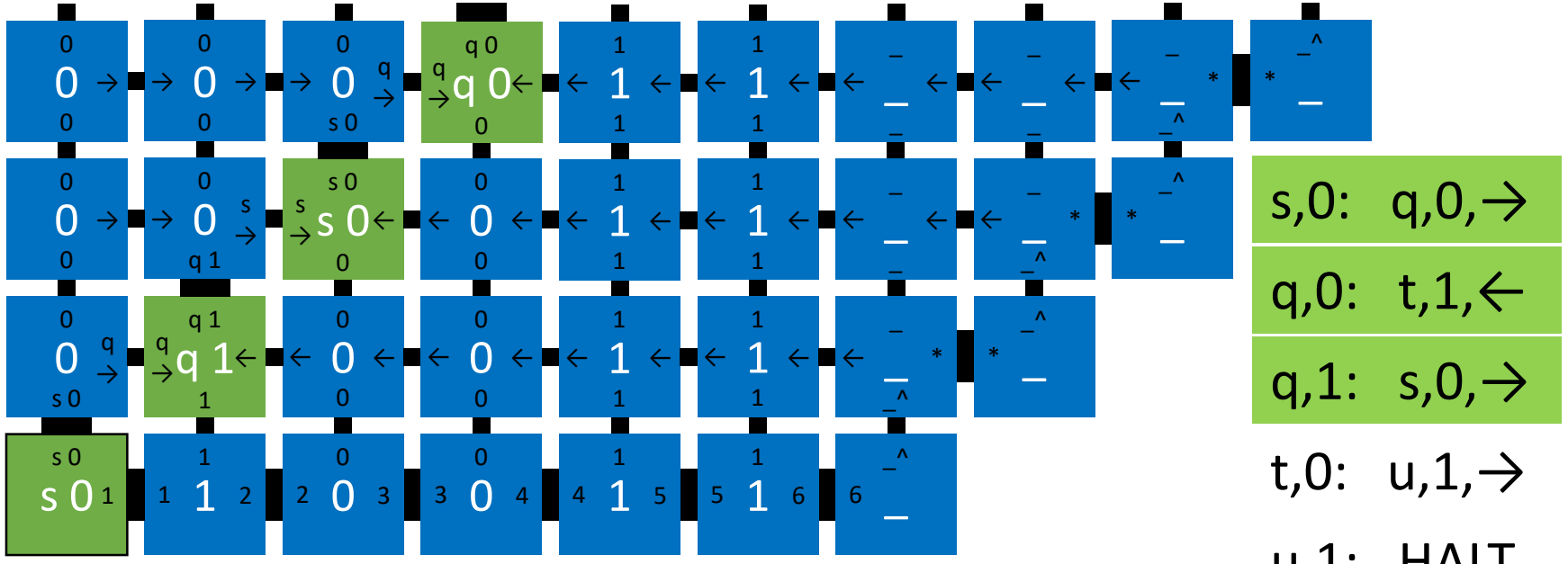
q,0: t,1,←

q,1: s,0,→

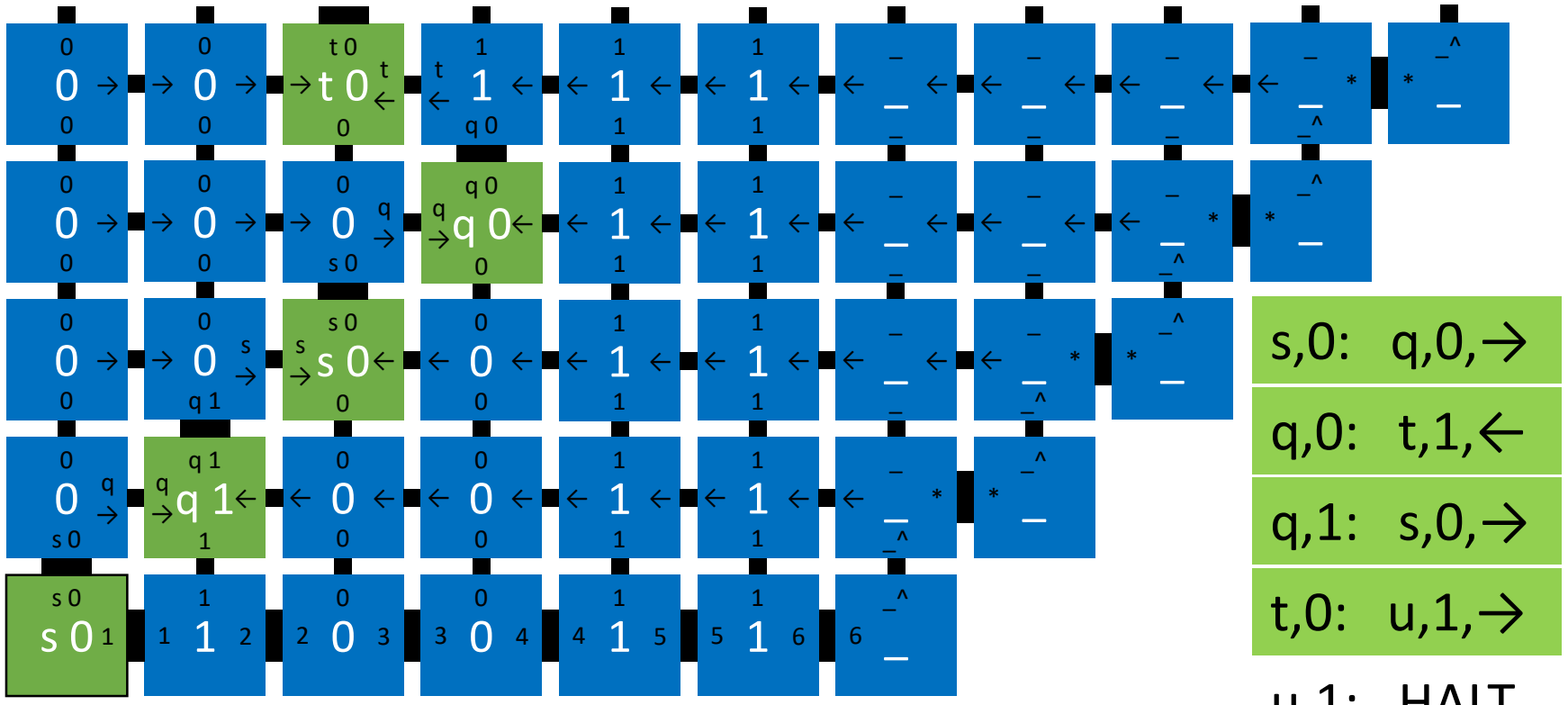
t,0: u,1,→

u,1: HALT

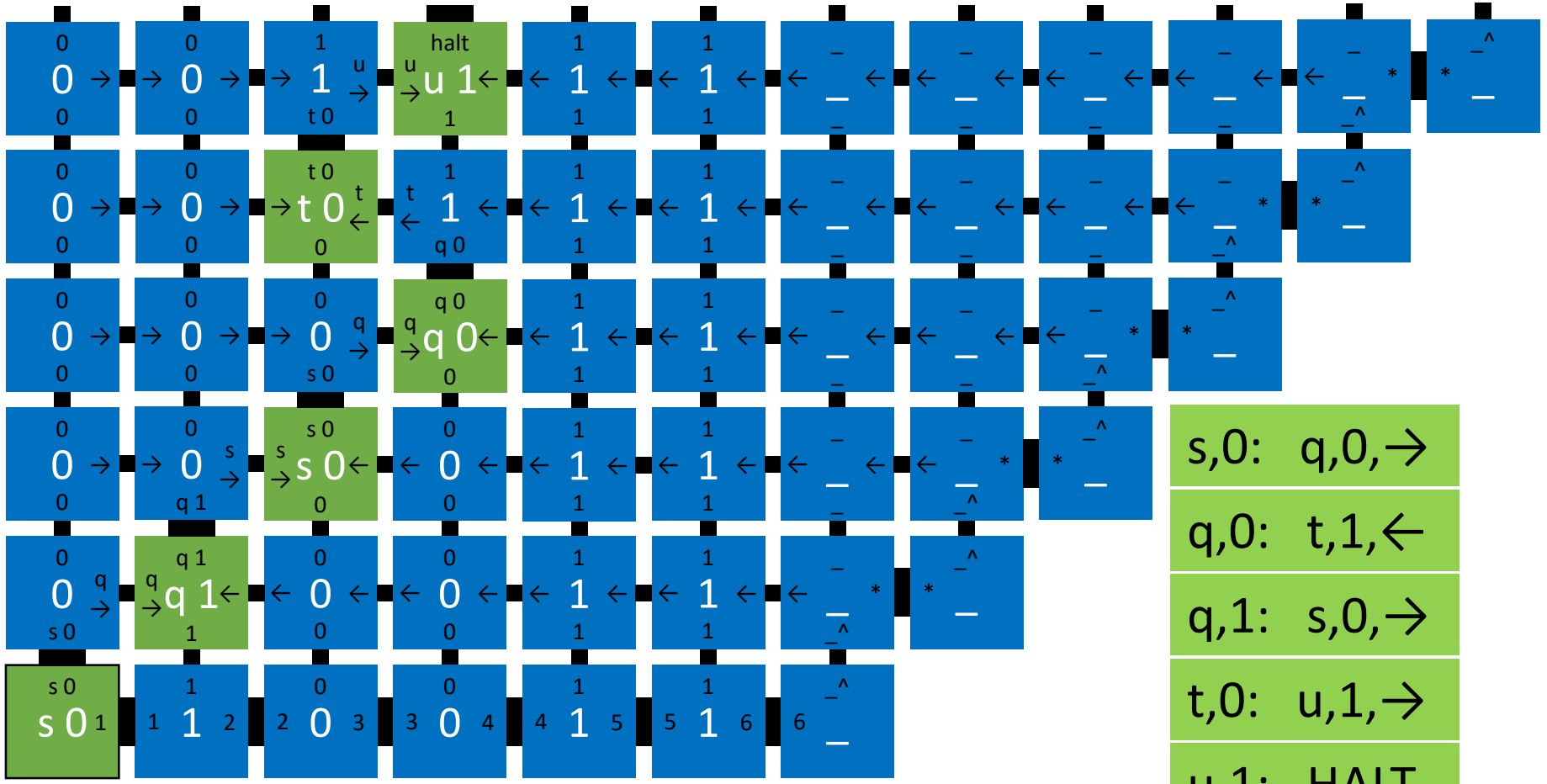
Tile assembly is Turing-universal



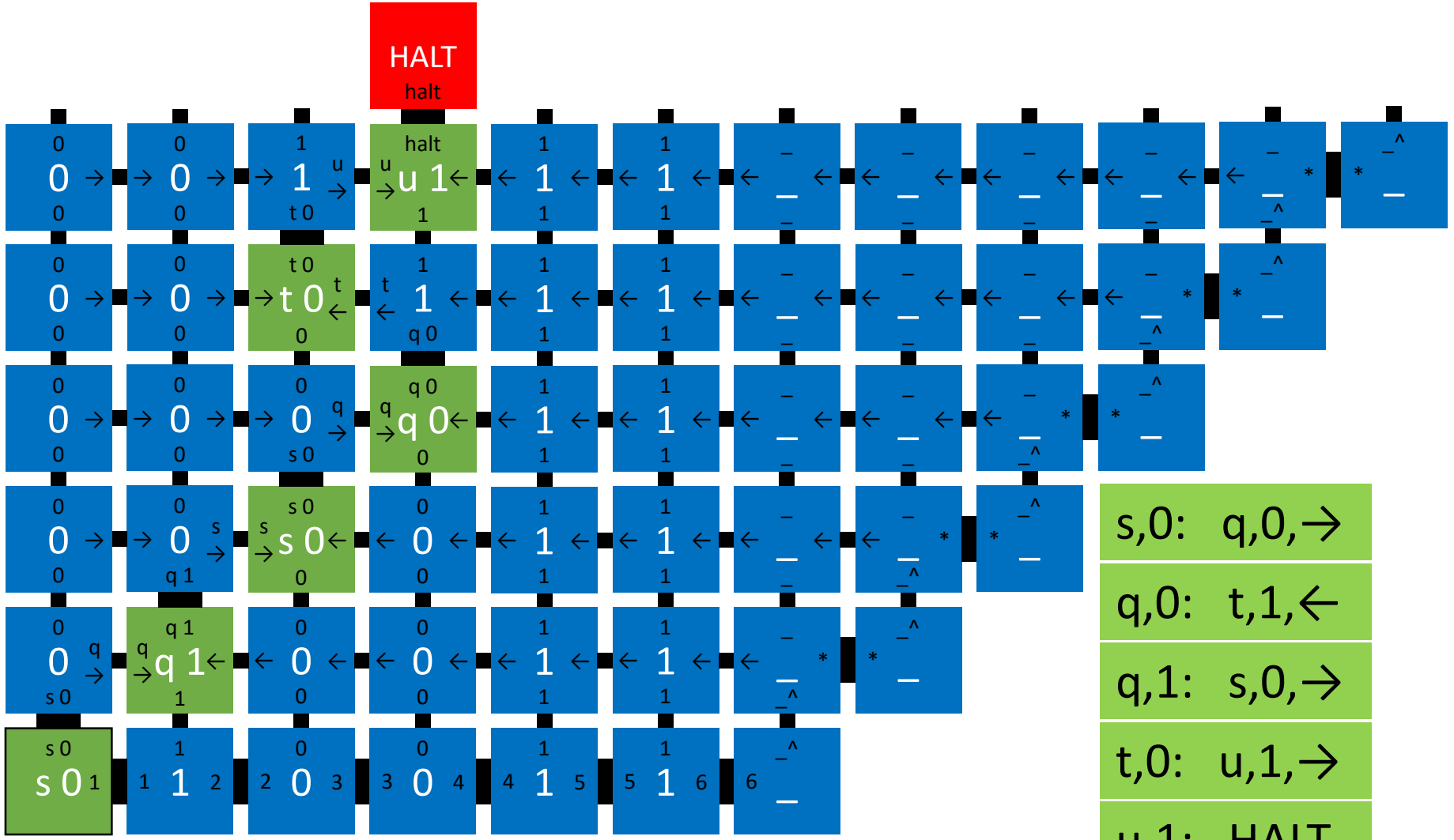
Tile assembly is Turing-universal



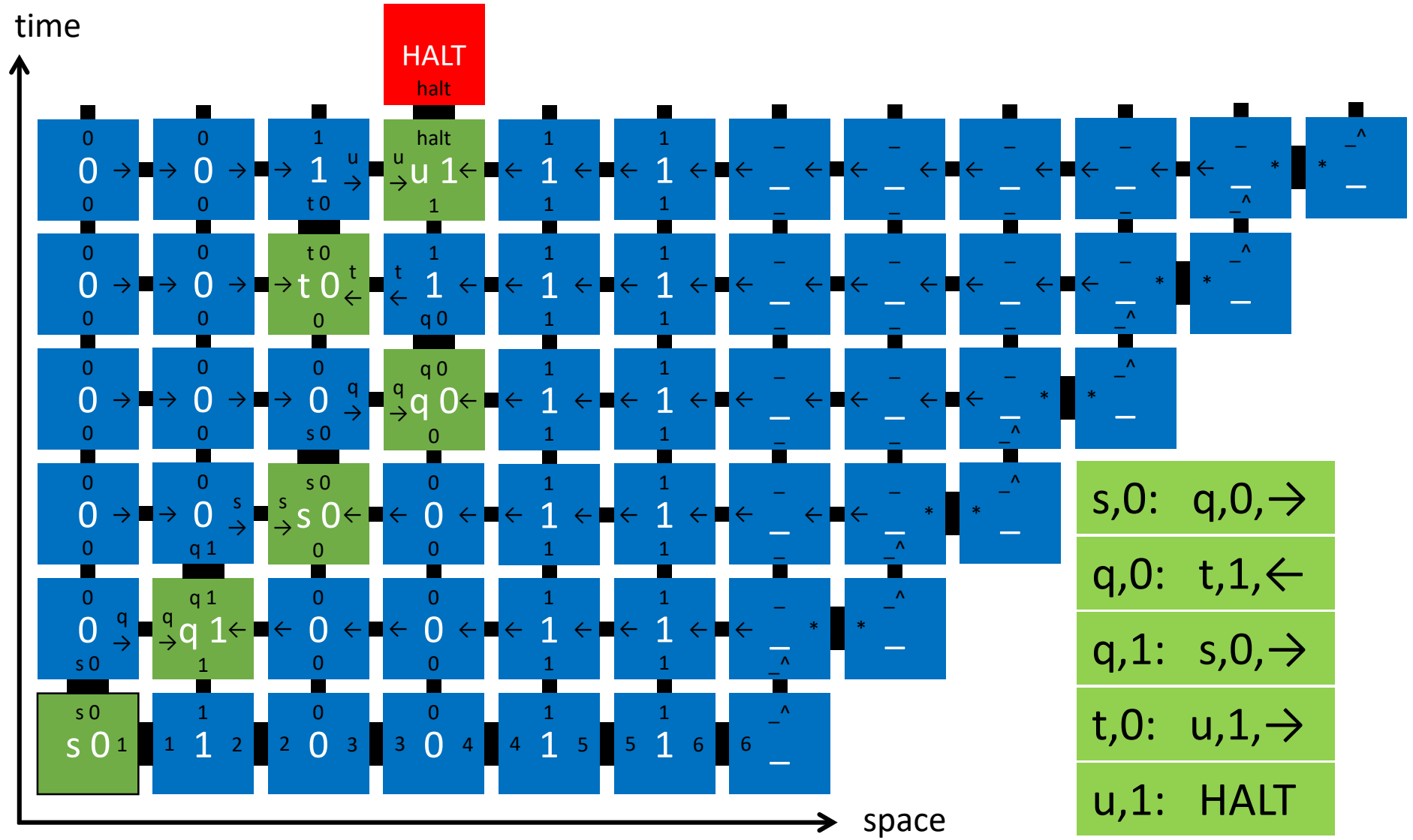
Tile assembly is Turing-universal



Tile assembly is Turing-universal

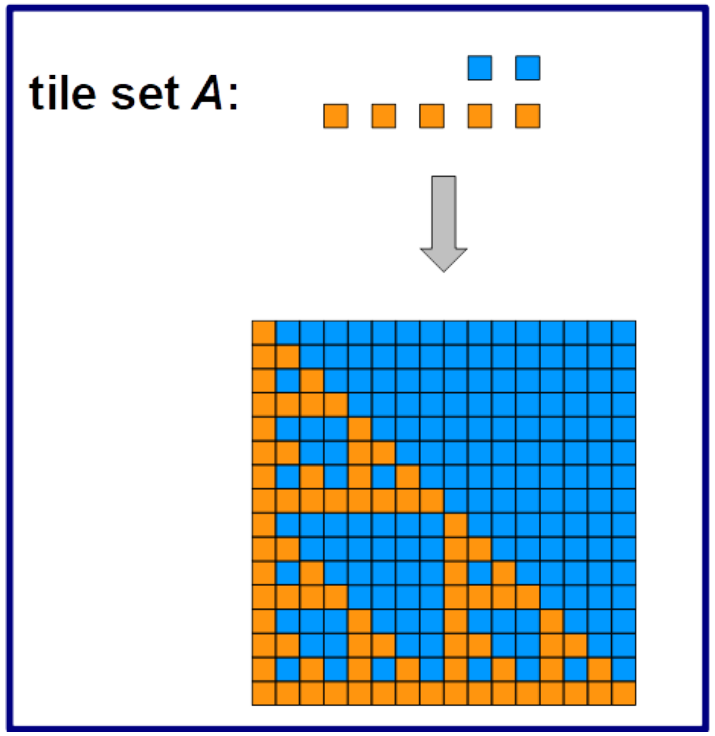


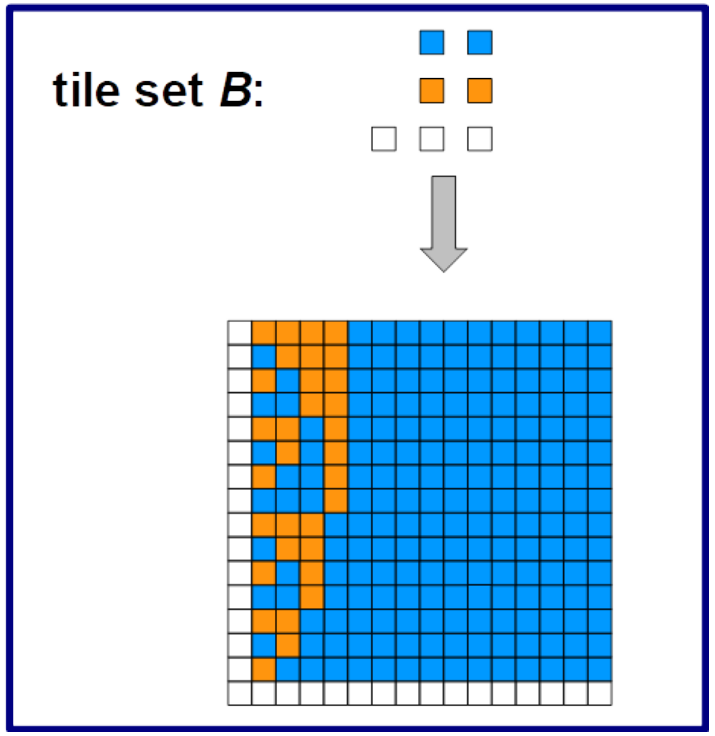
Tile assembly is Turing-universal



Putting the *algorithm* in *algorithmic* self-assembly

- set of tile types is like a program
- shape it creates, or pattern it paints, is like the output of the program





Putting the *algorithm* in *algorithmic* self-assembly

How is a set of tile types **not** like a program?

Putting the *algorithm* in *algorithmic* self-assembly

How is a set of tile types **not** like a program?

- Where's the input to the program?

Putting the *algorithm* in *algorithmic* self-assembly

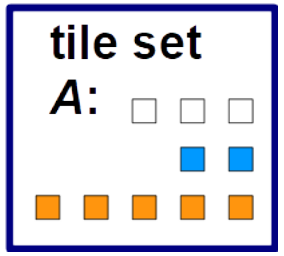
How is a set of tile types **not** like a program?

- Where's the input to the program?
- One perspective: pre-assembled seed encodes the input

Putting the *algorithm* in *algorithmic* self-assembly

How is a set of tile types **not** like a program?

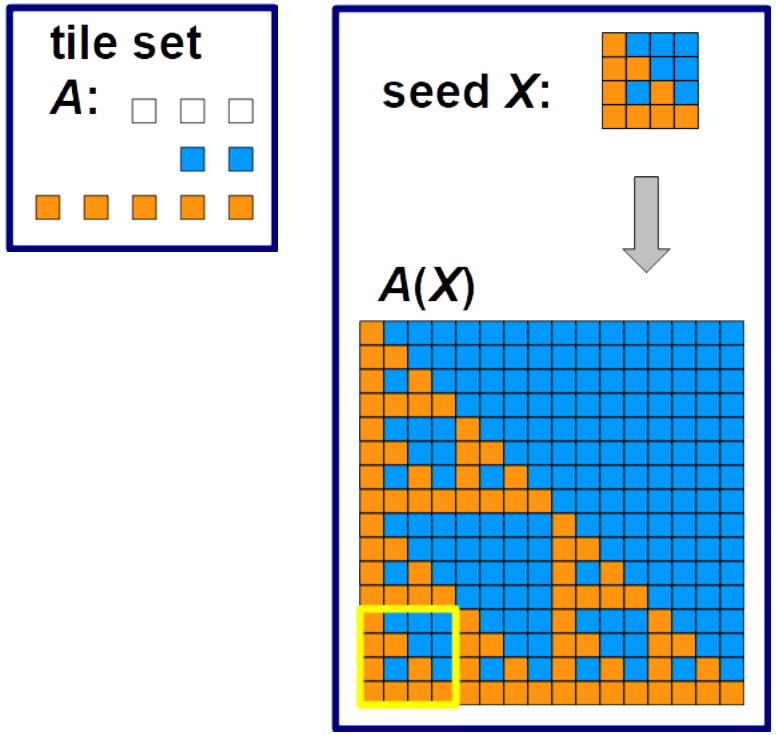
- Where's the input to the program?
- One perspective: pre-assembled seed encodes the input



Putting the *algorithm* in *algorithmic* self-assembly

How is a set of tile types **not** like a program?

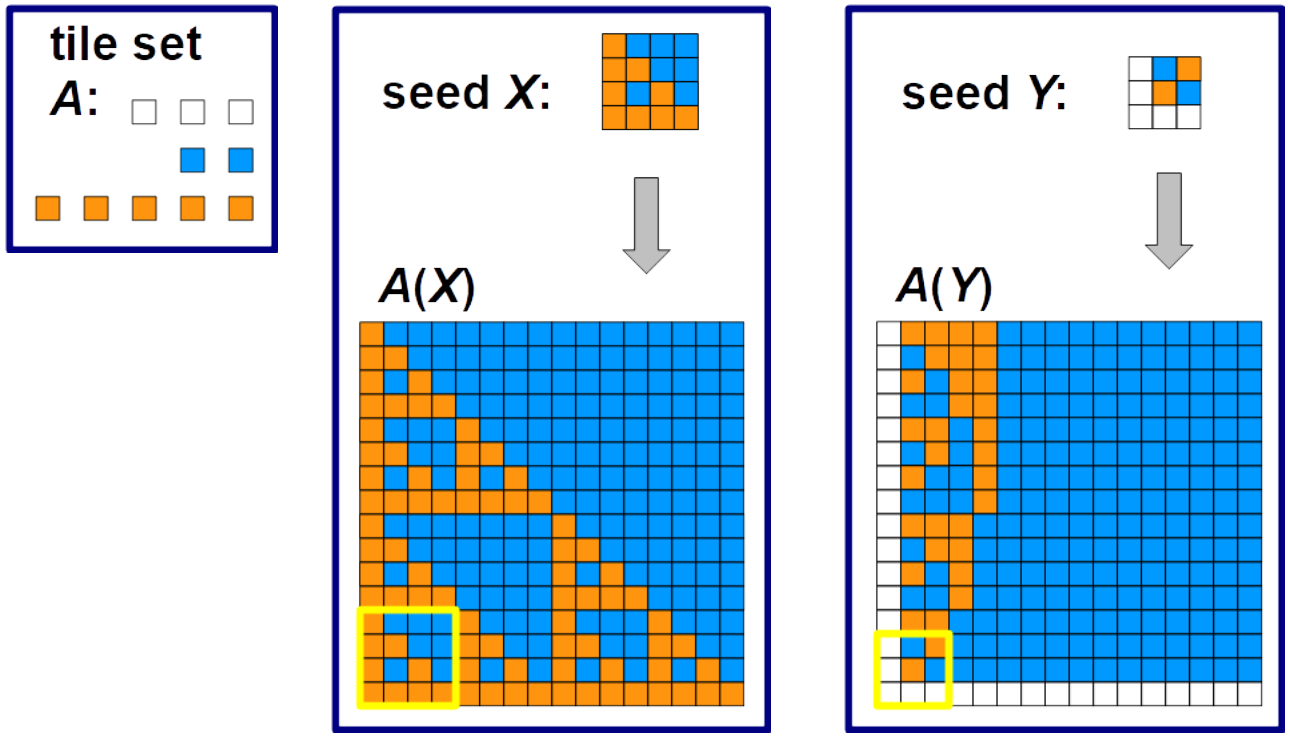
- Where's the input to the program?
- One perspective: pre-assembled seed encodes the input



Putting the *algorithm* in *algorithmic* self-assembly

How is a set of tile types **not** like a program?

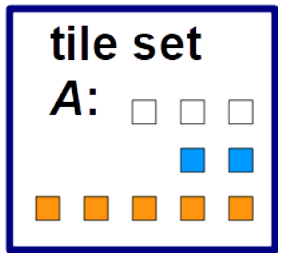
- Where's the input to the program?
- One perspective: pre-assembled seed encodes the input

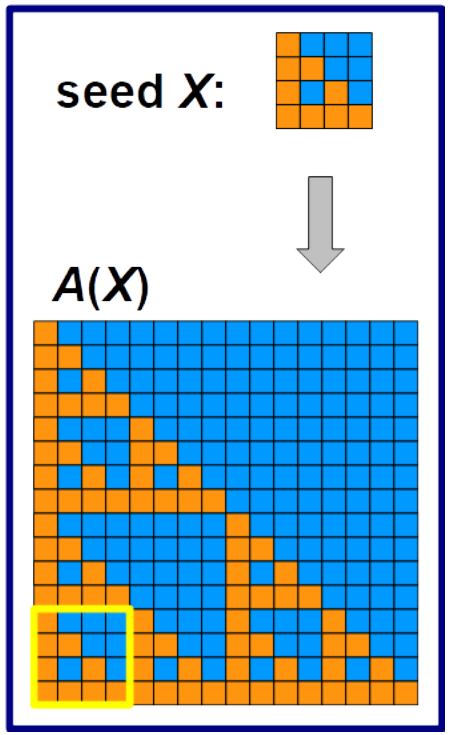


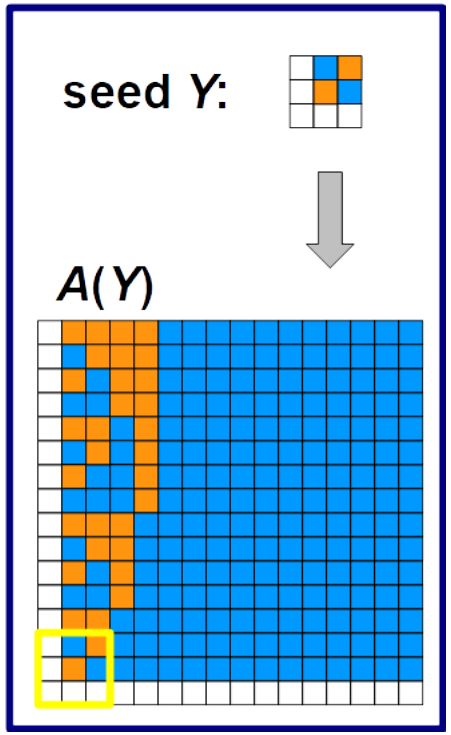
Putting the *algorithm* in *algorithmic* self-assembly

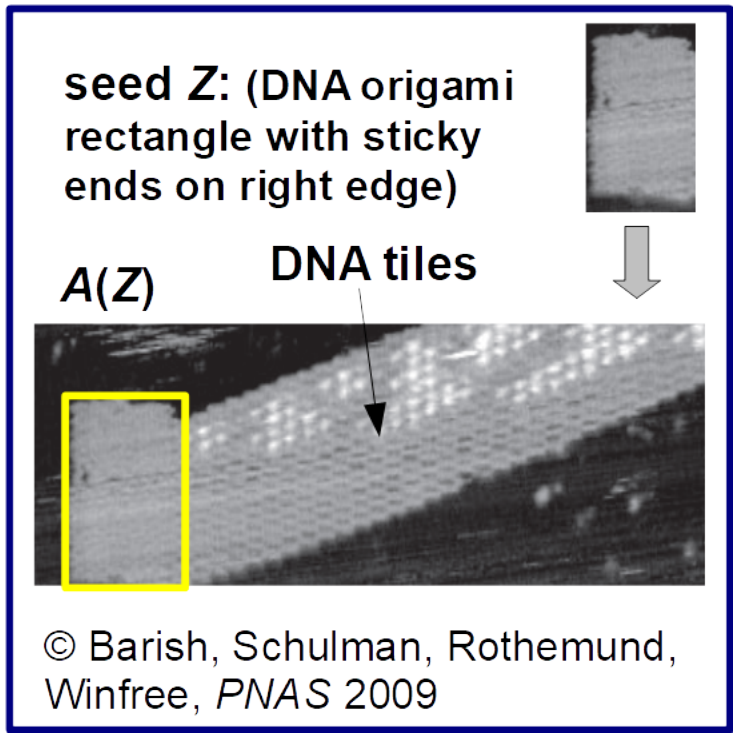
How is a set of tile types **not** like a program?

- Where's the input to the program?
- One perspective: pre-assembled seed encodes the input









Calculating parity of 6-bit string: 1 algorithm, 2^6 inputs

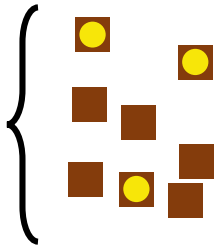
seed encoding **1**0**0**1**0**1

seed encoding **1**1**0**1**0**1

Calculating parity of 6-bit string: 1 algorithm, 2^6 inputs

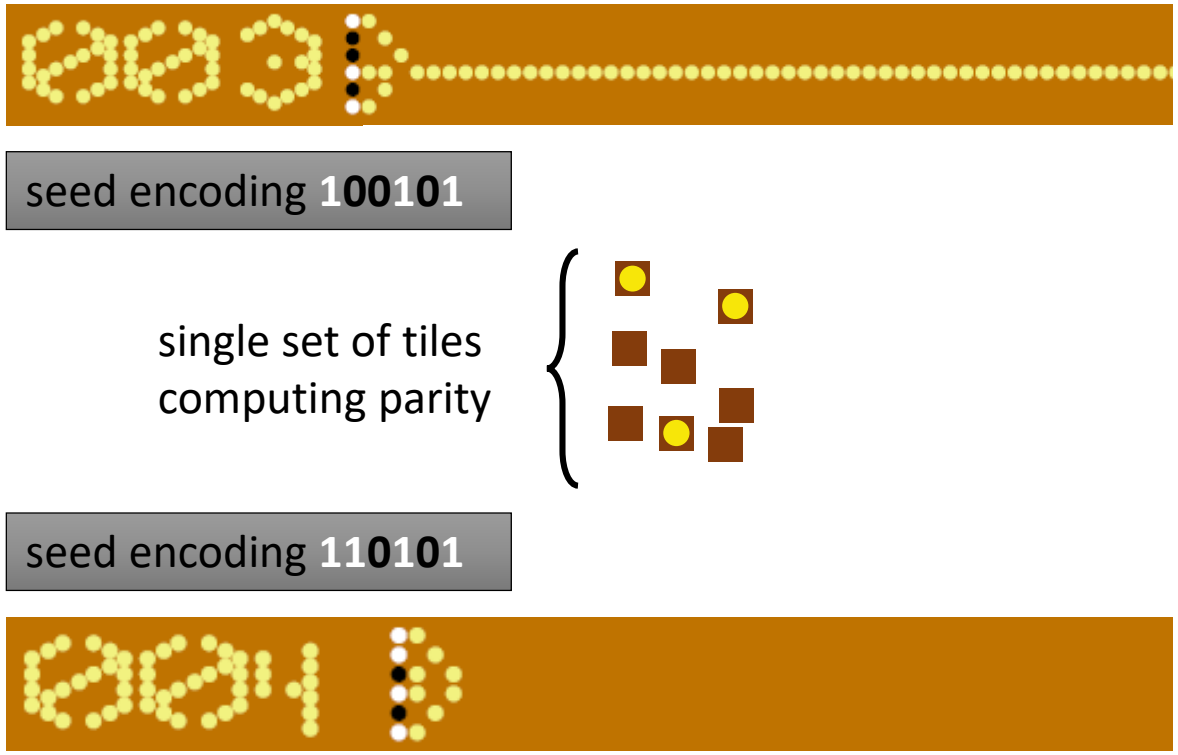
seed encoding **1**00**1**0**1**

single set of tiles computing parity

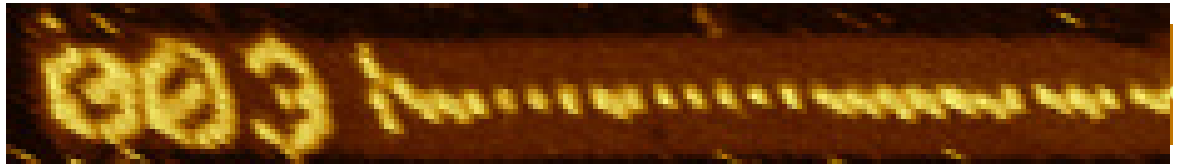


seed encoding **11**0**1**0**1**

Calculating parity of 6-bit string: 1 algorithm, 2^6 inputs

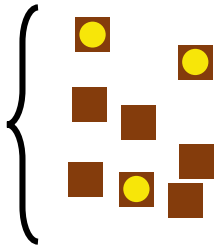


Calculating parity of 6-bit string: 1 algorithm, 2^6 inputs



seed encoding **1**00**1**0**1**

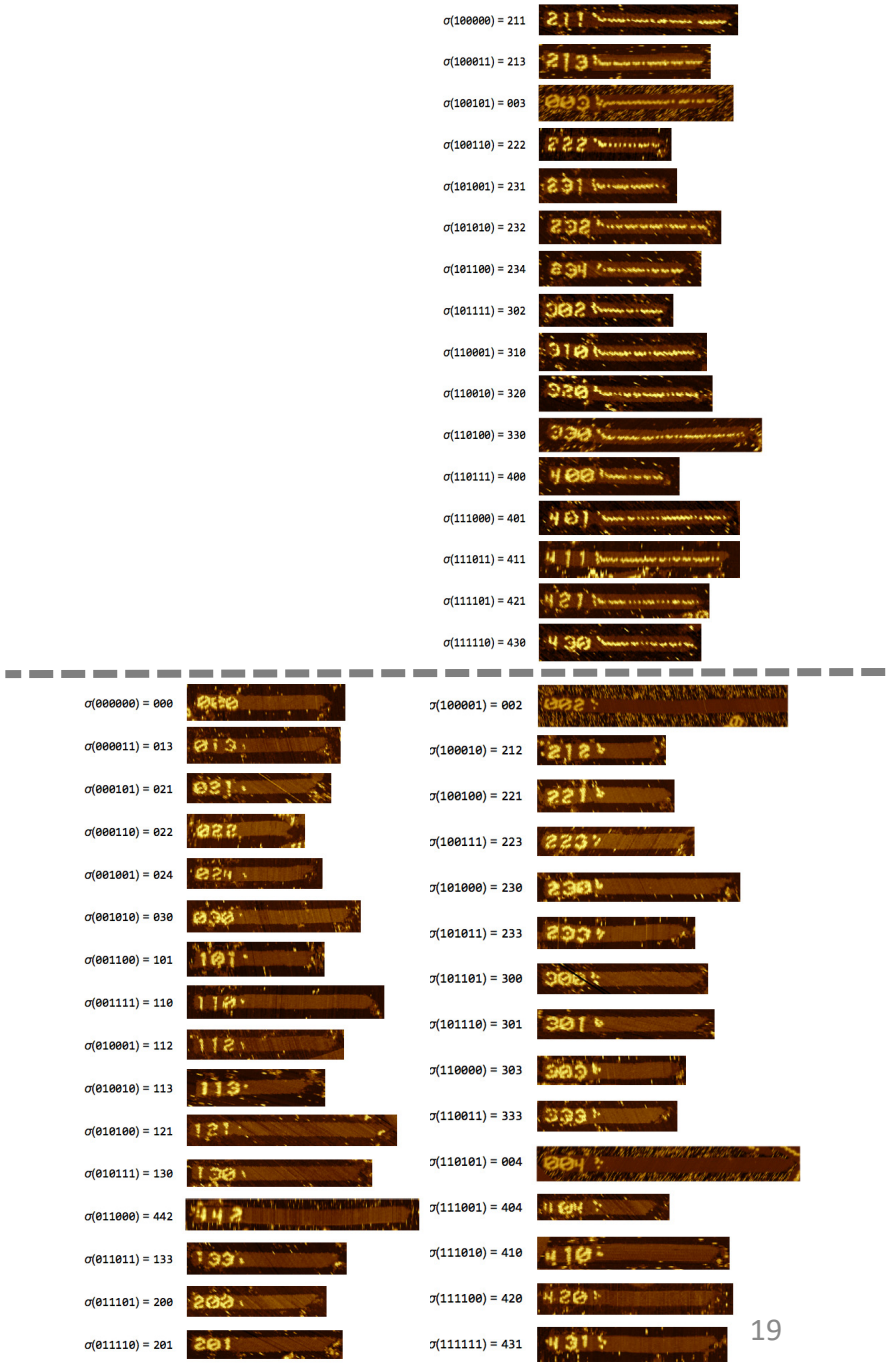
single set of tiles computing parity



seed encoding **1**10**1**0**1**

[*Iterated Boolean circuit computation via a programmable DNA tile array*. Woods, Doty, Myhrvold, Hui, Wu, Yin, Winfree, submitted]

2^6 seeds:



So tiles can compute… what’s that good for?

So tiles can compute… what's that good for?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

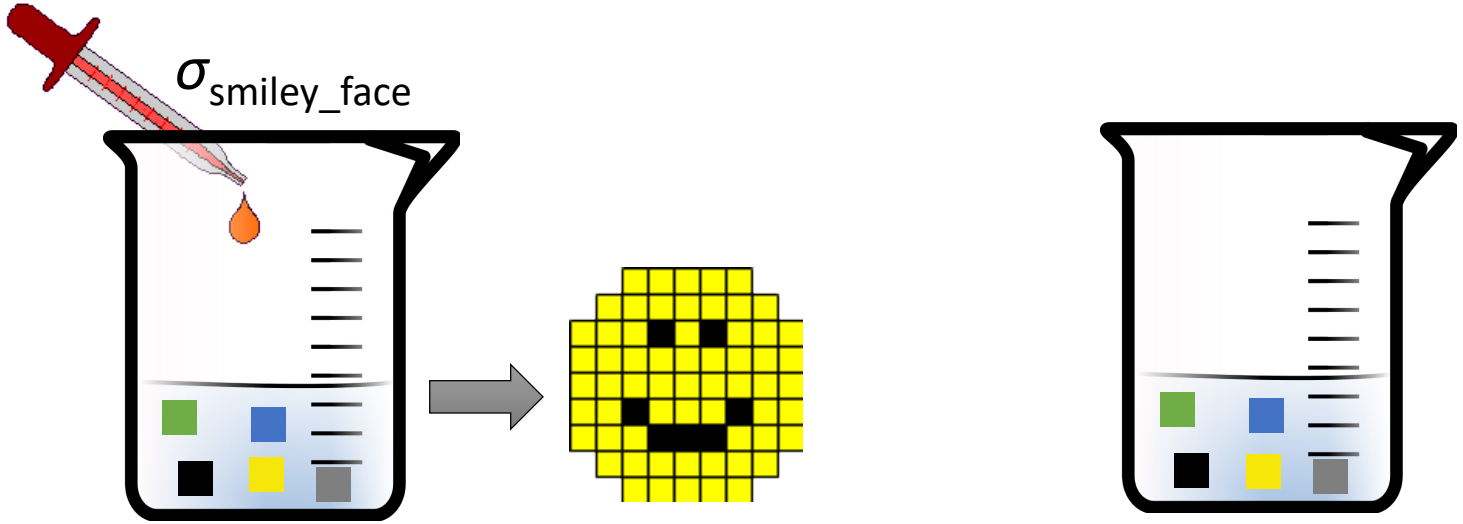
So tiles can compute… what's that good for?

Theorem: There is a single set *T* of tile types, so that, for any finite shape *S*, from an appropriately chosen seed σ_S "encoding" *S*, *T* self-assembles *S*.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

So tiles can compute… what's that good for?

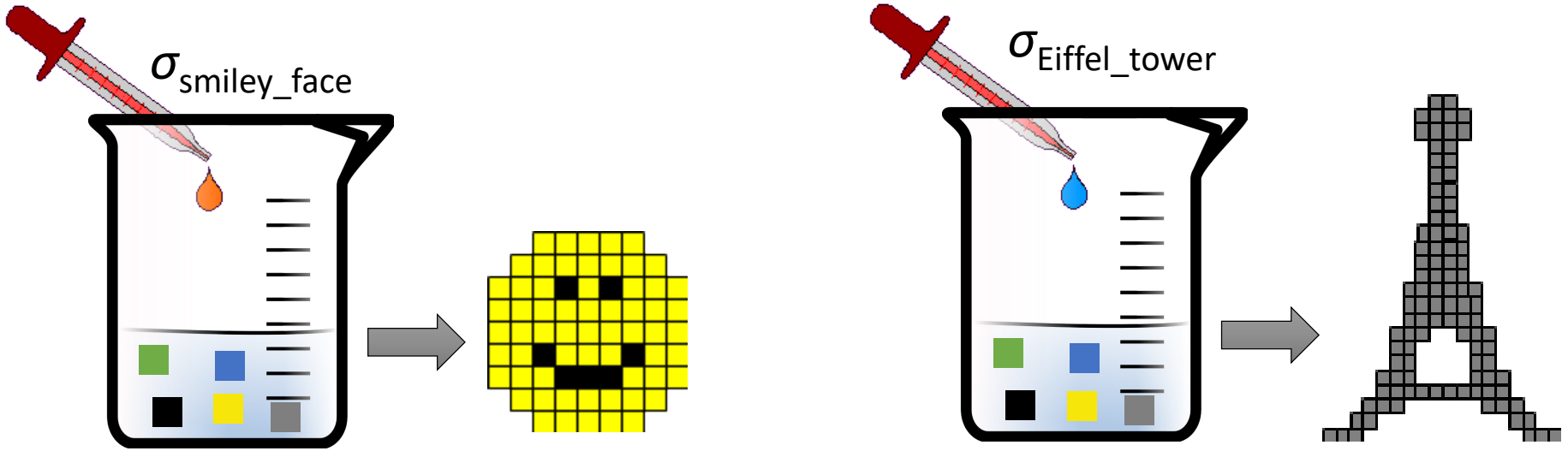
Theorem: There is a single set *T* of tile types, so that, for any finite shape *S*, from an appropriately chosen seed σ_S "encoding" *S*, *T* self-assembles *S*.



[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

So tiles can compute… what's that good for?

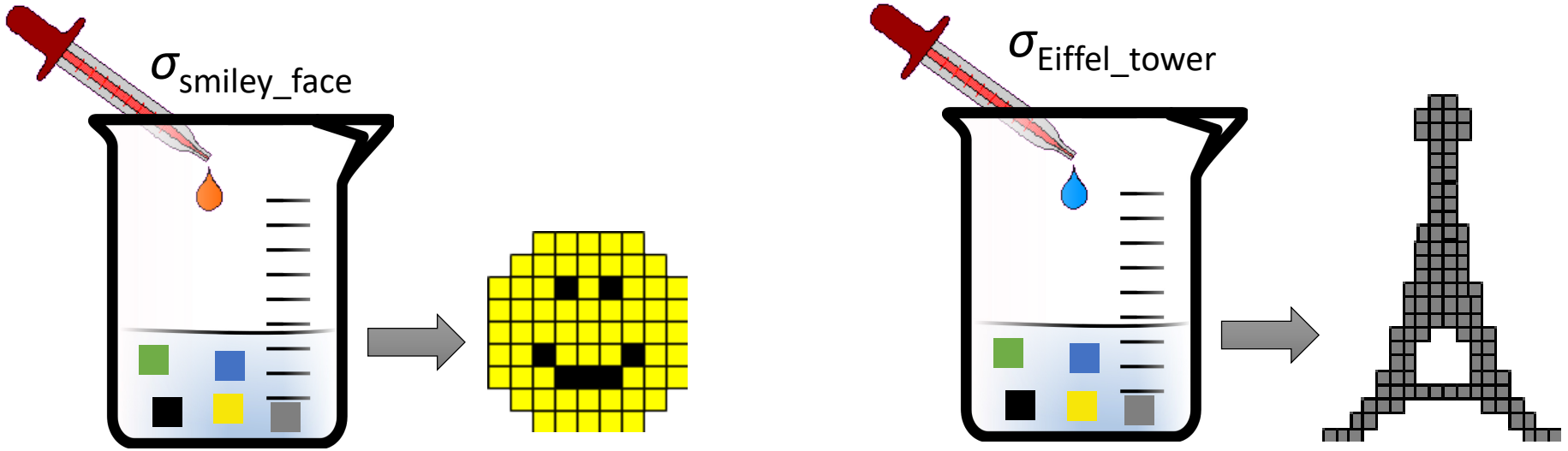
Theorem: There is a single set *T* of tile types, so that, for any finite shape *S*, from an appropriately chosen seed σ_S "encoding" *S*, *T* self-assembles *S*.



[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

So tiles can compute… what's that good for?

Theorem: There is a single set *T* of tile types, so that, for any finite shape *S*, from an appropriately chosen seed σ_S "encoding" *S*, *T* self-assembles *S*.



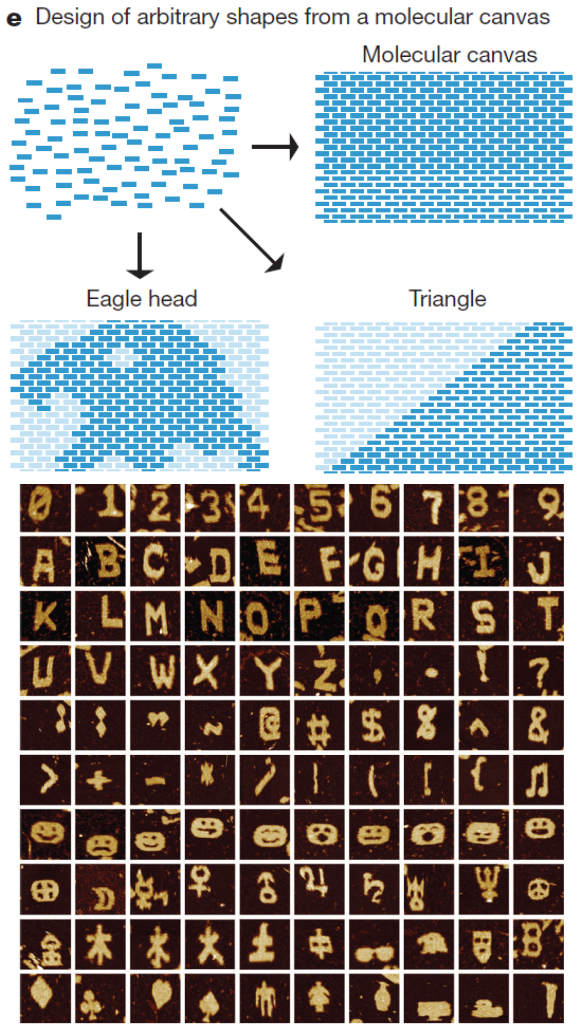
These tiles are universally programmable for building any shape.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Open problems

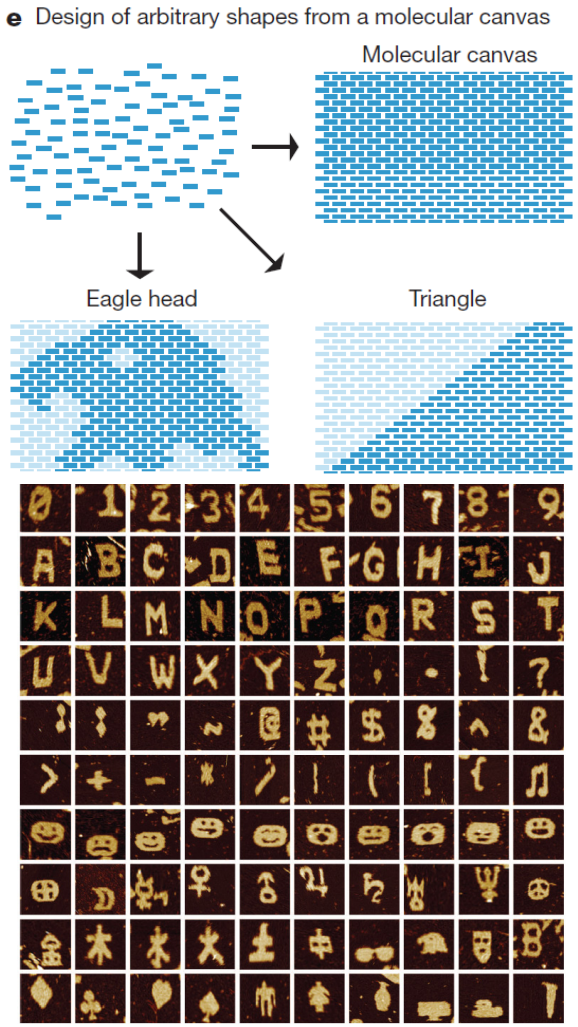
Theory of programmable barriers to nucleation in tile self-assembly

Experimental tile self-assembly

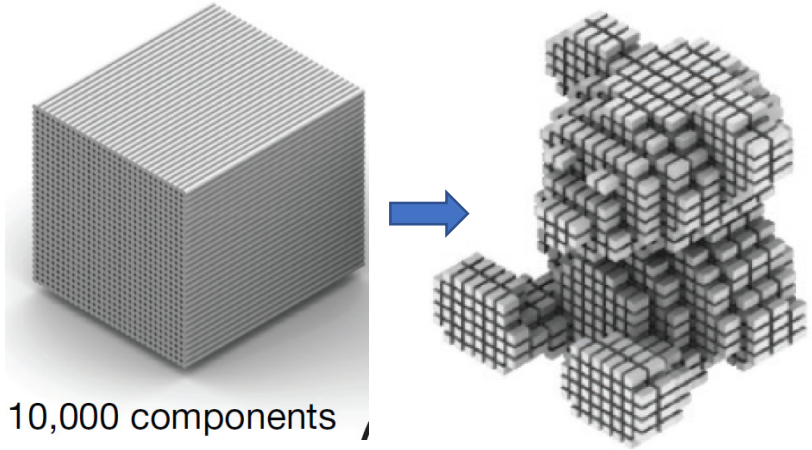


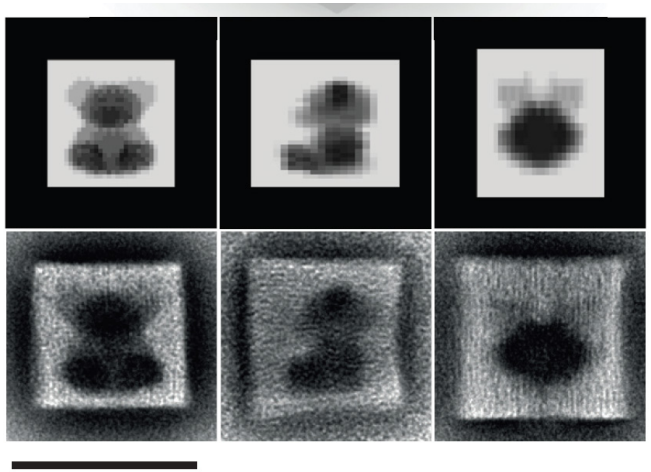
Wei, Dai, Yin, *Nature* 2012

Experimental tile self-assembly



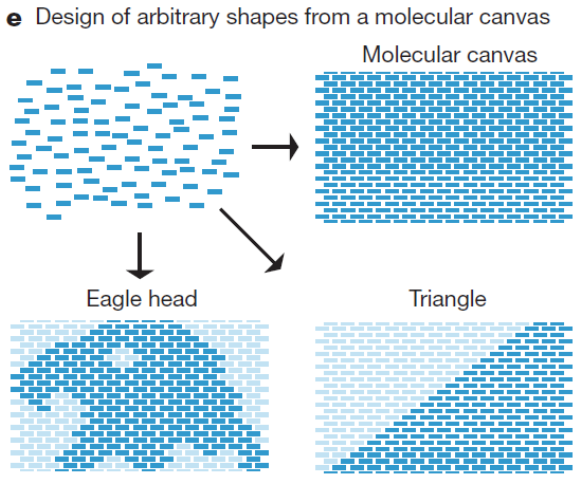
Wei, Dai, Yin, *Nature* 2012



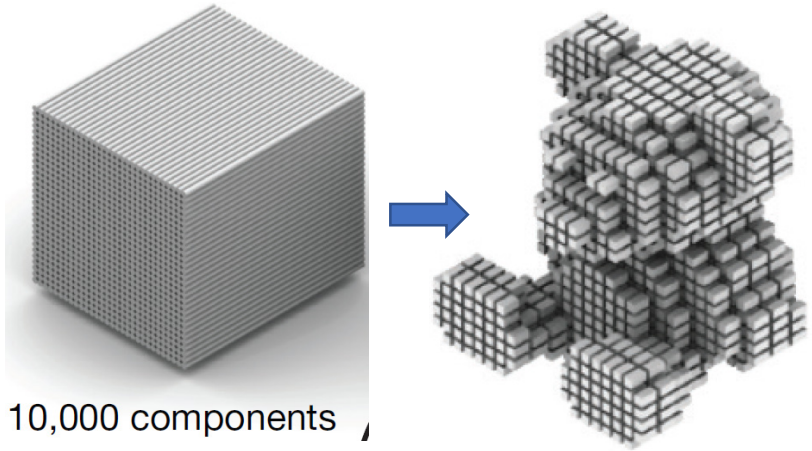


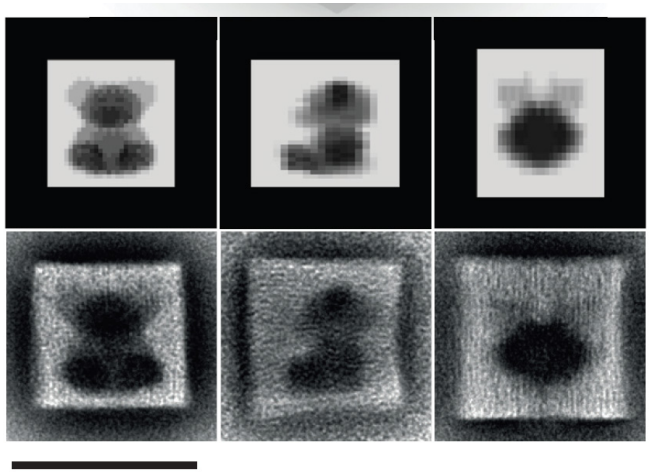
Ong *et al*, *Nature* 2017

Experimental tile self-assembly

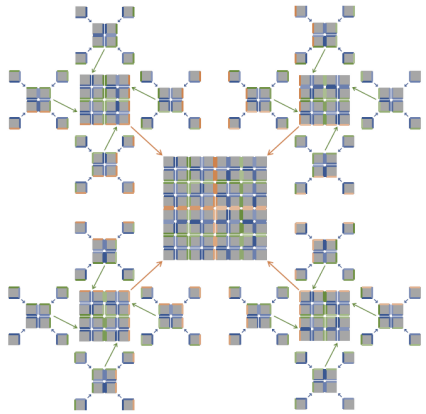


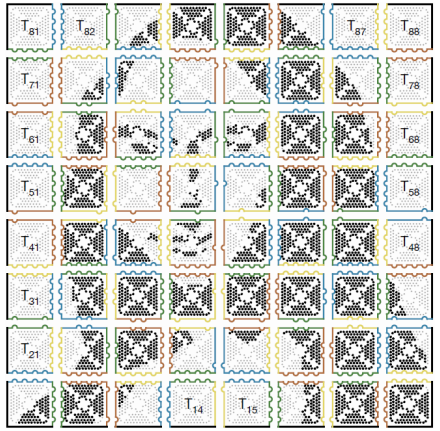
Wei, Dai, Yin, *Nature* 2012

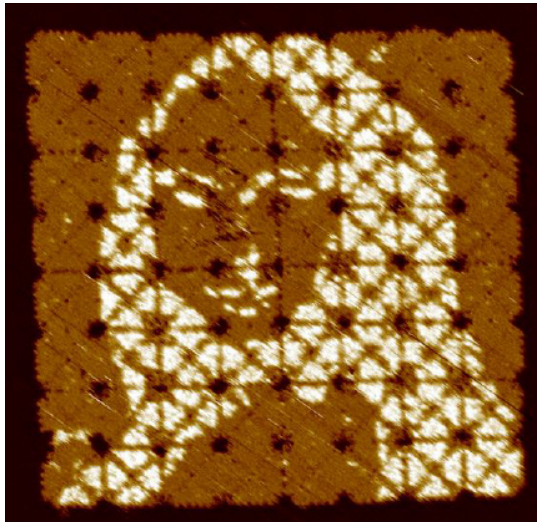




Ong *et al*, *Nature* 2017

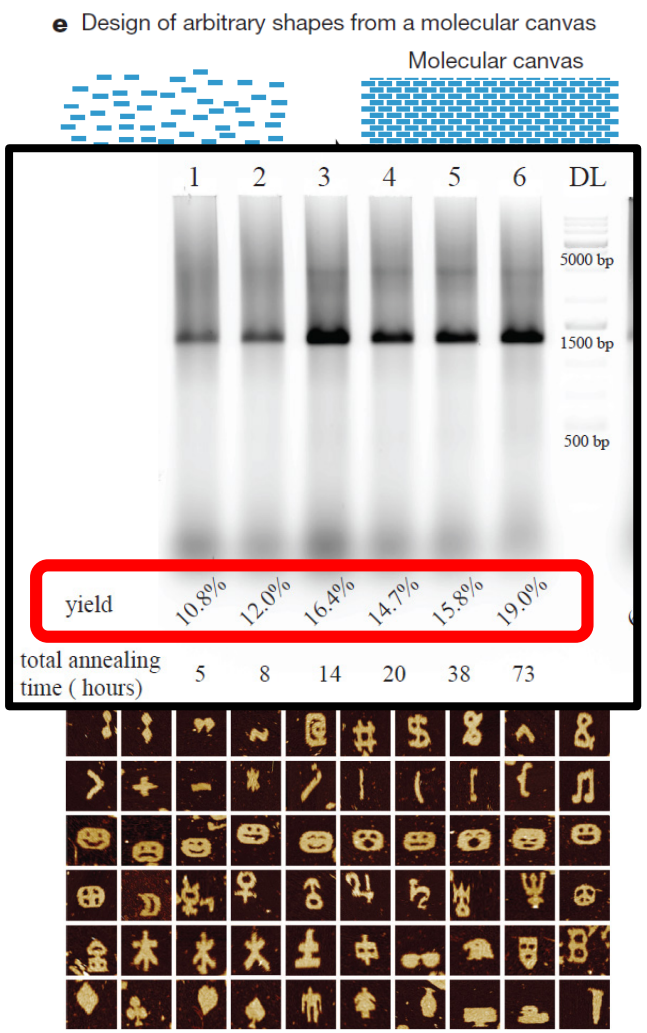




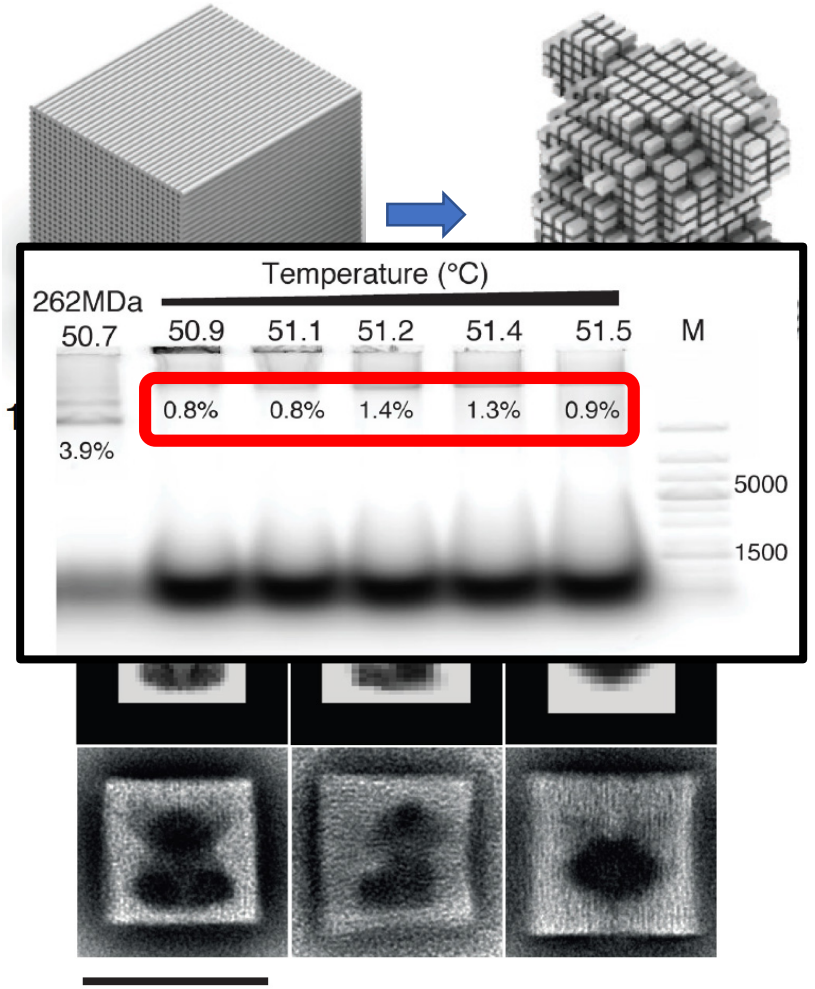


Tikhomirov, Peterson, Qlan, *Nature* 2017

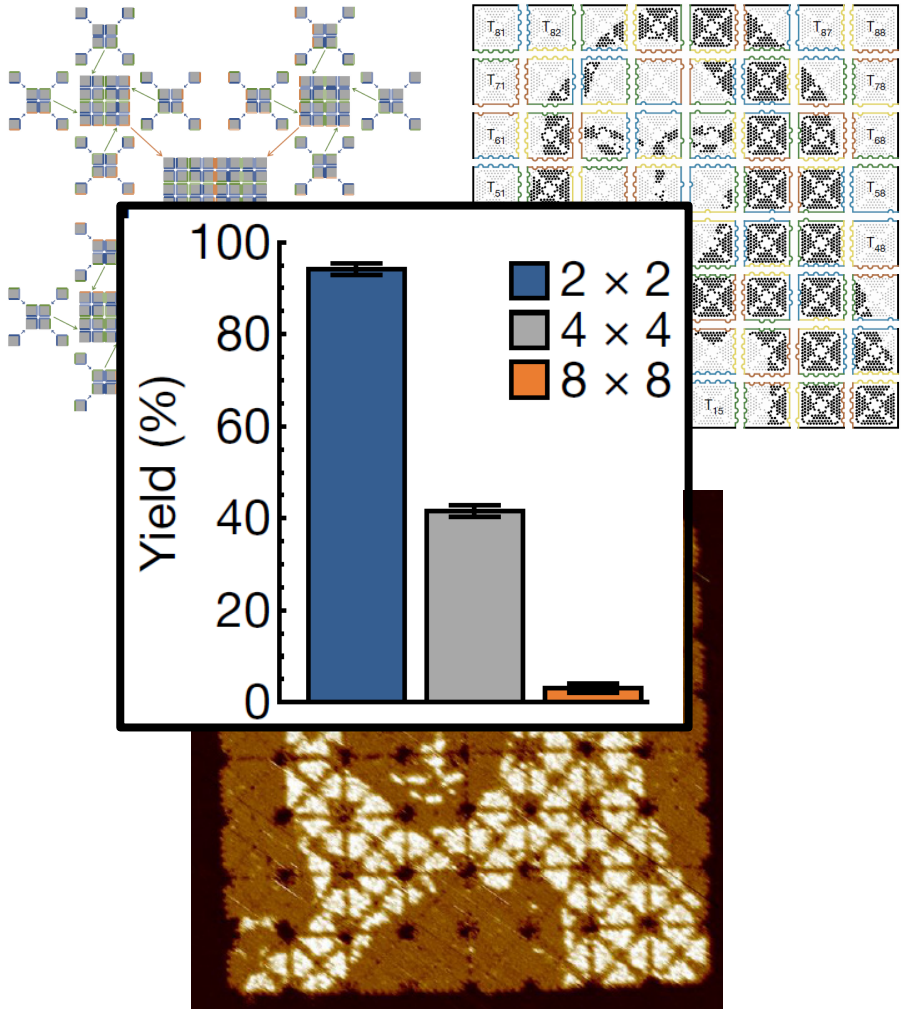
Experimental tile self-assembly



Wei, Dai, Yin, *Nature* 2012

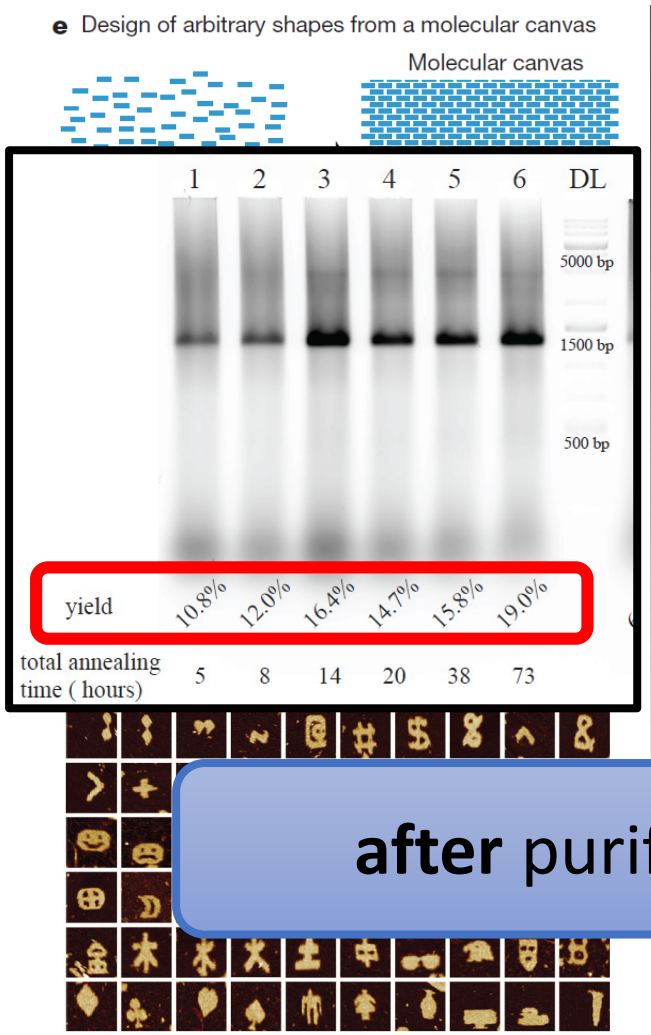


Ong *et al*, *Nature* 2017

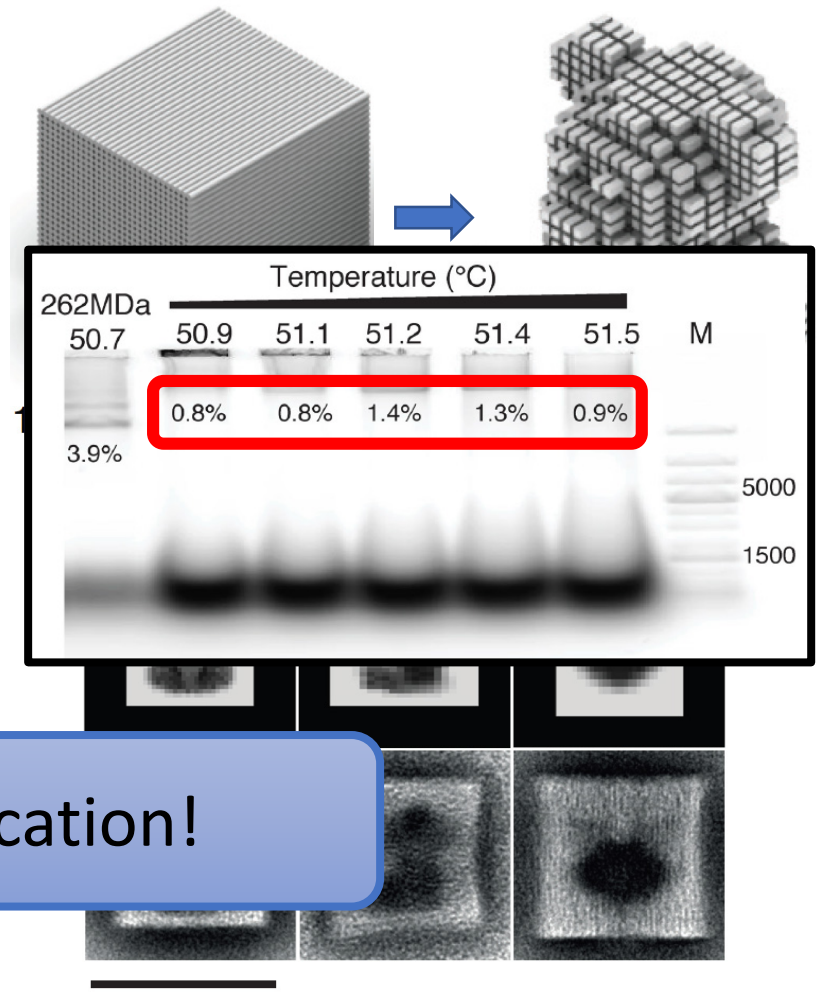


Tikhomirov, Peterson, Qlan, *Nature* 2017

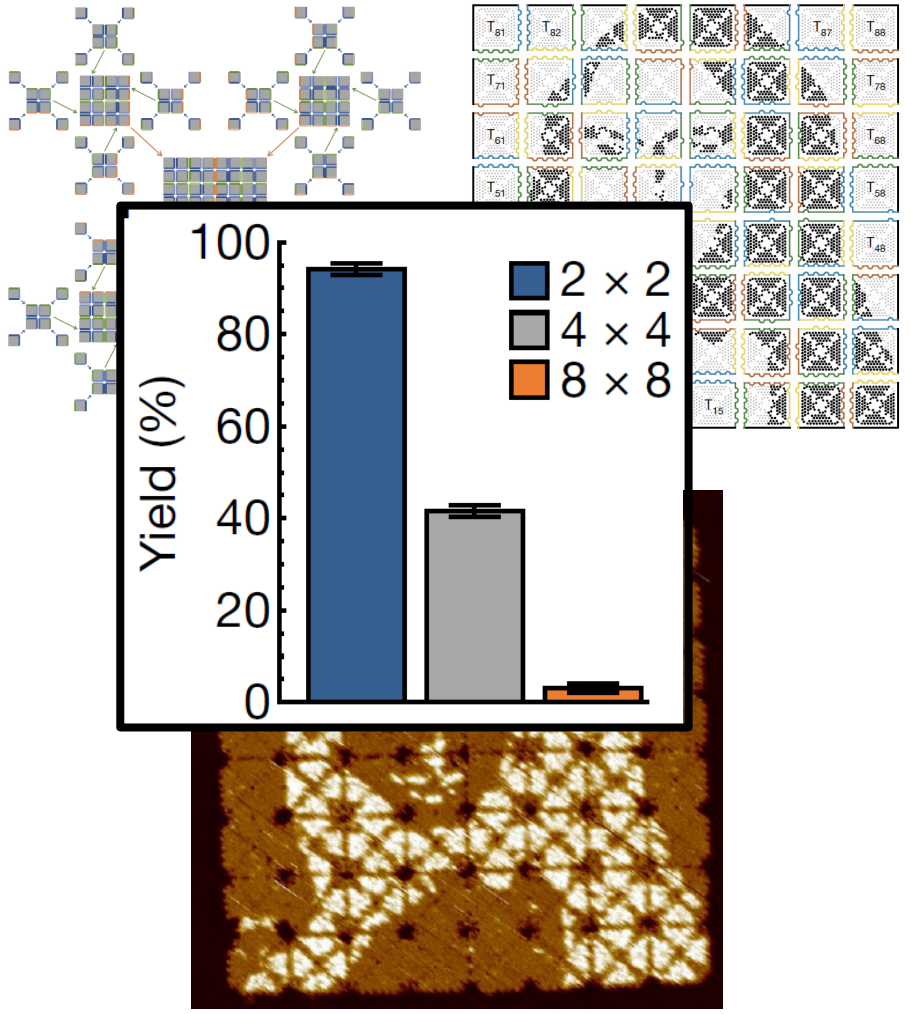
Experimental tile self-assembly



Wei, Dai, Yin, *Nature* 2012



Ong *et al*, *Nature* 2017



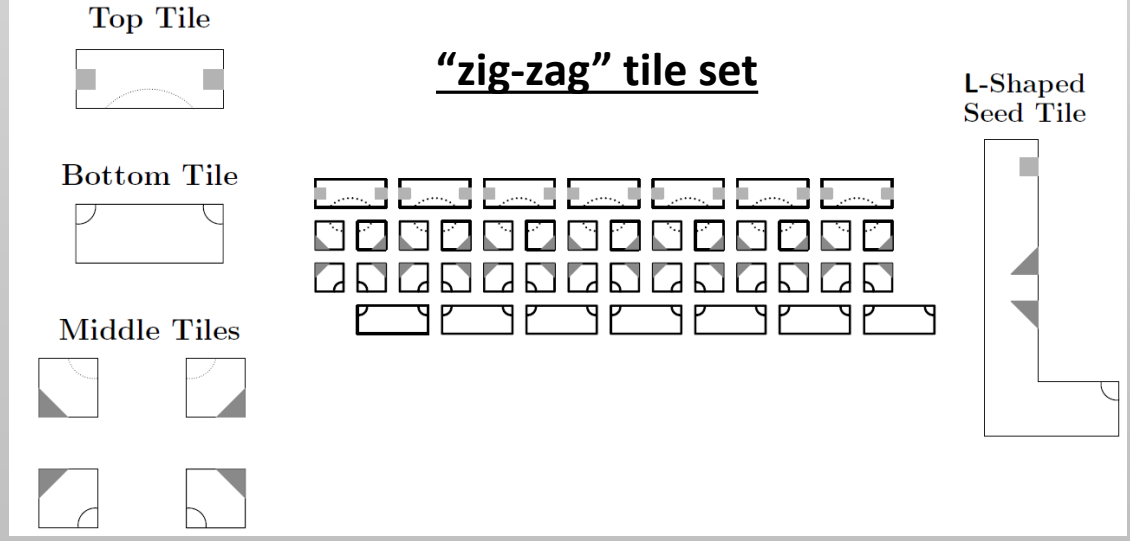
Tikhomirov, Peterson, Qlan, *Nature* 2017

after purification!

Secret to higher yields: Control of nucleation

Schulman, Winfree, *SICOMP* 2009

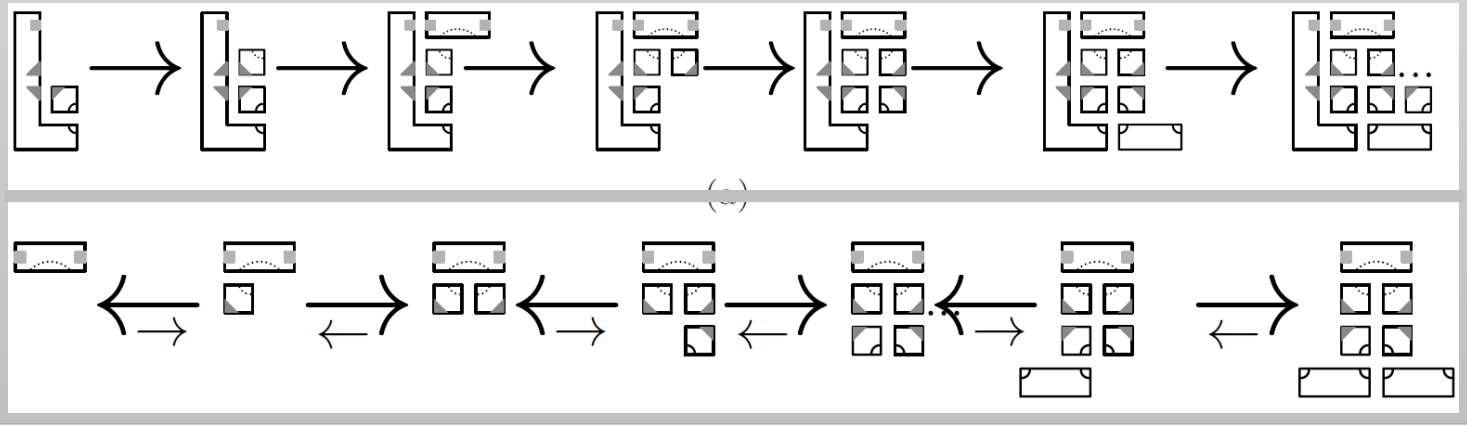
Secret to higher yields: Control of nucleation



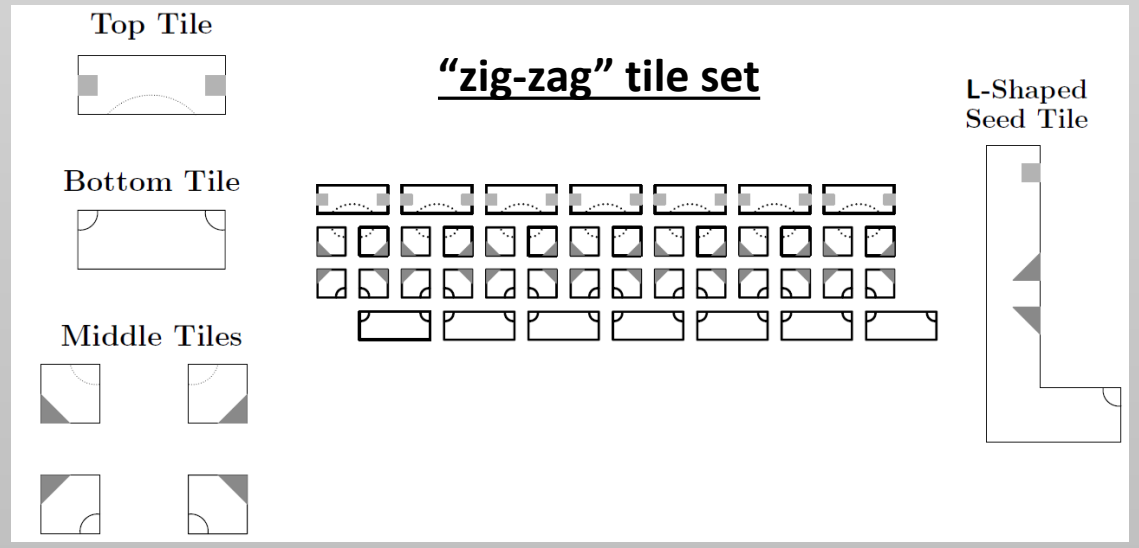
Schulman, Winfree, *SICOMP* 2009

intended growth from seed:

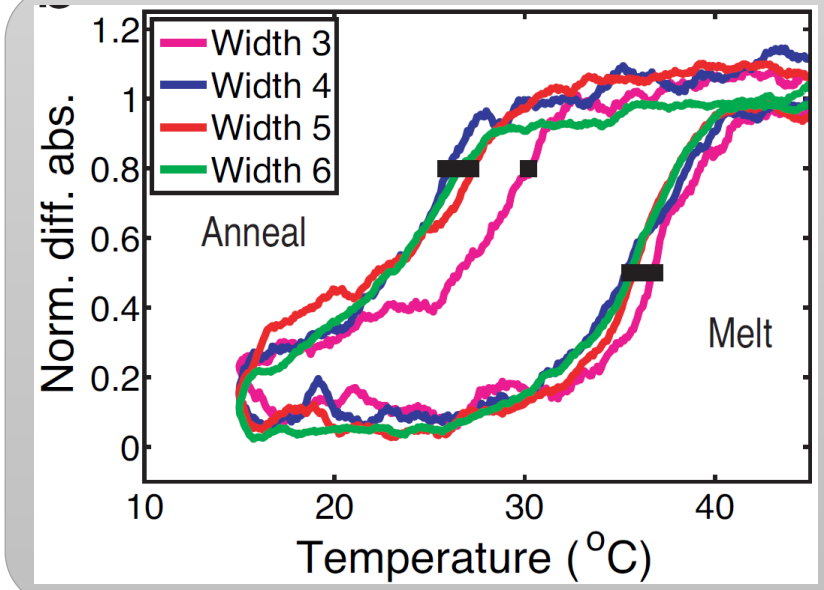
growth pathways without seed:



Secret to higher yields: Control of nucleation



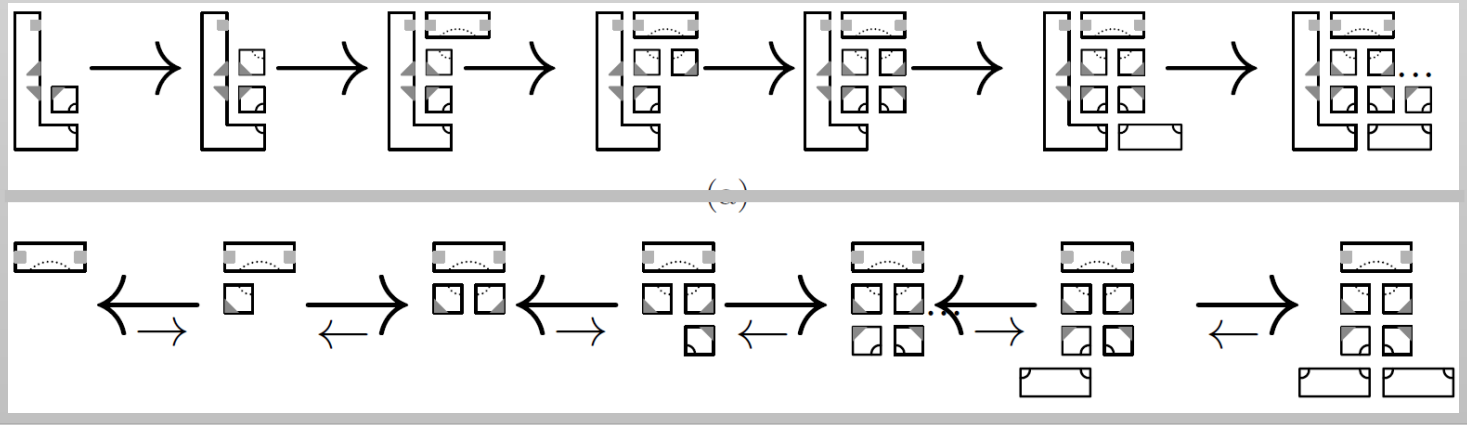
Schulman, Winfree, *SICOMP* 2009



Schulman, Winfree, *PNAS* 2009

intended growth from seed:

growth pathways without seed:



Open problems

- **Goal:** Define *kinetic barrier to nucleation*: something like “assembling any structure of size b requires $\Omega(b)$ weak attachments”.

Open problems

- **Goal:** Define *kinetic barrier to nucleation*: something like “assembling any structure of size b requires $\Omega(b)$ weak attachments”.
- **Conjecture:** If tiles self-assemble with seed σ, but have kinetic barrier b to nucleation without σ, then σ must be “size” at least b.

Open problems

- **Goal:** Define *kinetic barrier to nucleation*: something like “assembling any structure of size b requires $\Omega(b)$ weak attachments”.
- **Conjecture:** If tiles self-assemble with seed σ, but have kinetic barrier b to nucleation without σ, then σ must be “size” at least b.
- **Conjecture:** If there is a “combinatorial” barrier to nucleation (at least b weak attachments must occur to grow a structure α), then there is a “classical physics” barrier to nucleation (growth rate of α is “low” under mass-action kinetics)

Open problems

- **Goal:** Define *kinetic barrier to nucleation*: something like “assembling any structure of size b requires $\Omega(b)$ weak attachments”.
- **Conjecture:** If tiles self-assemble with seed σ, but have kinetic barrier b to nucleation without σ, then σ must be “size” at least b.
- **Conjecture:** If there is a “combinatorial” barrier to nucleation (at least b weak attachments must occur to grow a structure α), then there is a “classical physics” barrier to nucleation (growth rate of α is “low” under mass-action kinetics)
- **Goal:** Develop general scheme for self-assembling shapes with programmable kinetic barriers to nucleation. (even “hard-coded” would be interesting given low yields of experimental results)

Thank you!

Questions?