kinetic Tile Assembly Model

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

$r_f \sim e^{-G_{mc}}$ forward rate

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

$$r_f \sim e^{-G_{mc}} \quad \text{forward rate}$$

$$r_r \sim e^{-b \cdot G_{se}} \quad \text{reverse rate}$$

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

$r_f \sim e^{-G_{mc}}$ forward rate

$r_r \sim e^{-b \cdot G_{se}}$ reverse rate

$e^{-G_{mc}}$ tile concentration

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

$r_f \sim e^{-G_{mc}}$	forward rate
$r_r \sim e^{-b \cdot G_{se}}$	reverse rate
$e^{-G_{mc}}$	tile concentration
b	# sticky ends bound

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

$r_f \sim e^{-G_{mc}}$	forward rate
$r_r \sim e^{-b \cdot G_{se}}$	reverse rate
$e^{-G_{mc}}$	tile concentration
b	# sticky ends bound
G_{se}	strength of 1 sticky end

kinetic Tile Assembly Model (kTAM)

differences with aTAM:

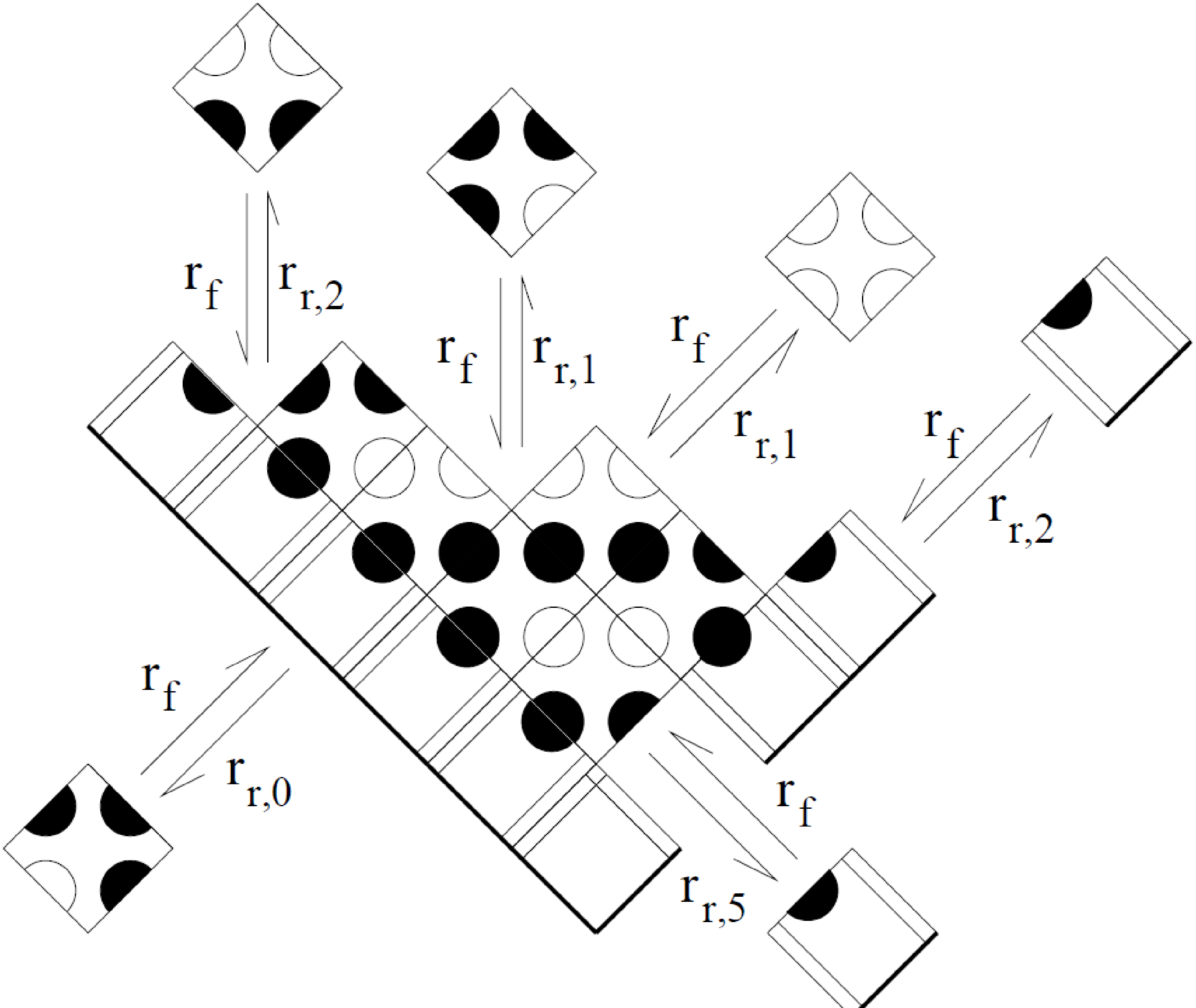
- tiles can attach by any positive strength glue
- tiles fall off, more quickly if bound weakly

$r_f \sim e^{-G_{mc}}$	forward rate
$r_r \sim e^{-b \cdot G_{se}}$	reverse rate
$e^{-G_{mc}}$	tile concentration
b	# sticky ends bound
G_{se}	strength of 1 sticky end

optimal growth when **forward rate just barely larger than reverse rate**, i.e., when

$$G_{mc} \approx 2 \cdot G_{se}$$

kTAM

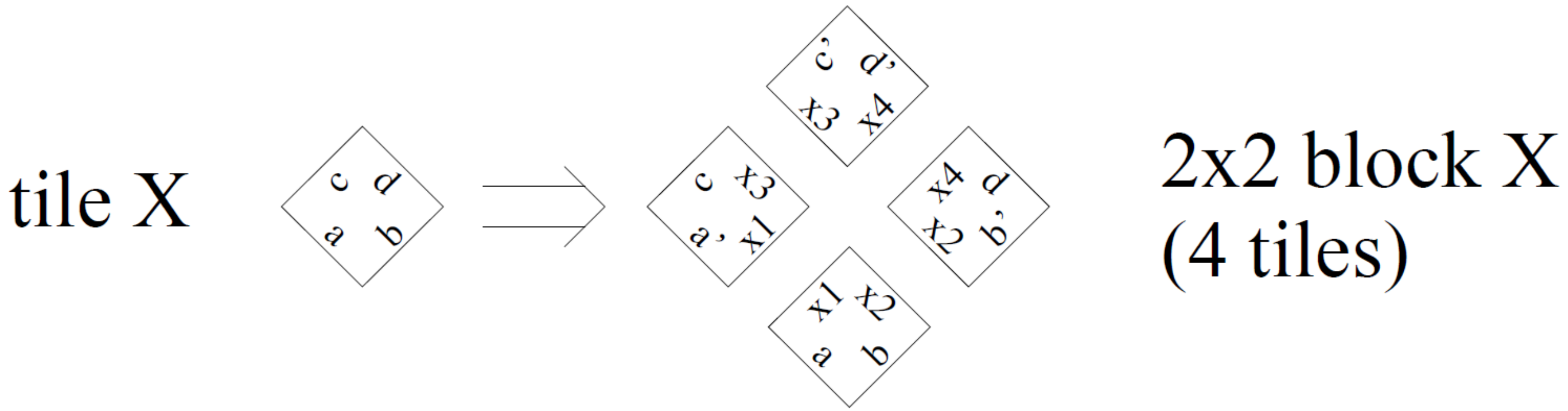


Proofreading: Error-correction in the kTAM

Definition: error = attachment by single strength 1 glue

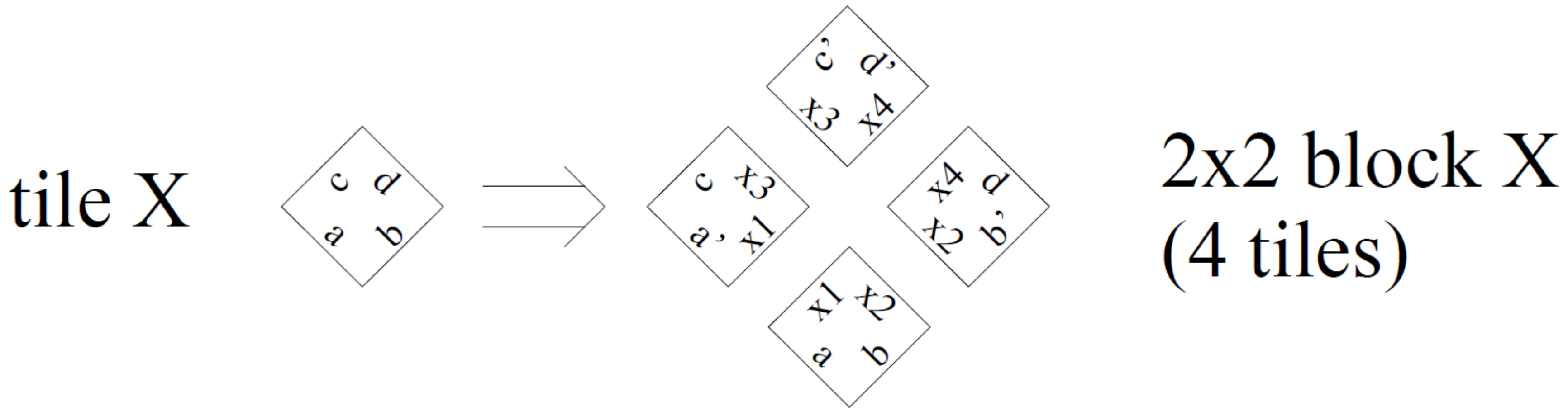
Proofreading: Error-correction in the kTAM

Definition: error = attachment by single strength 1 glue



Proofreading: Error-correction in the kTAM

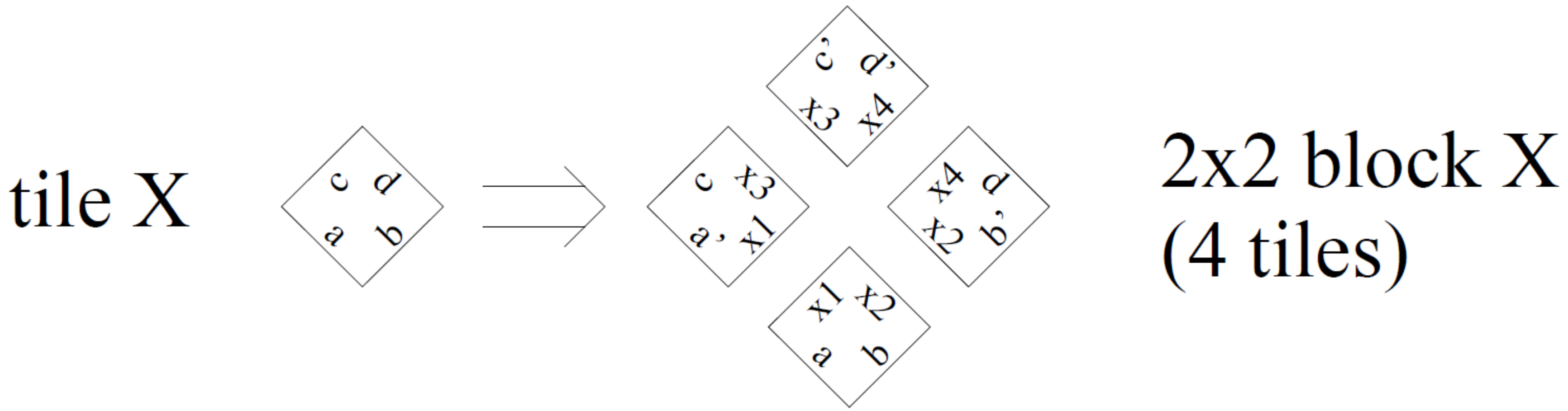
Definition: error = attachment by single strength 1 glue



glues internal to block are all unique

Proofreading: Error-correction in the kTAM

Definition: error = attachment by single strength 1 glue

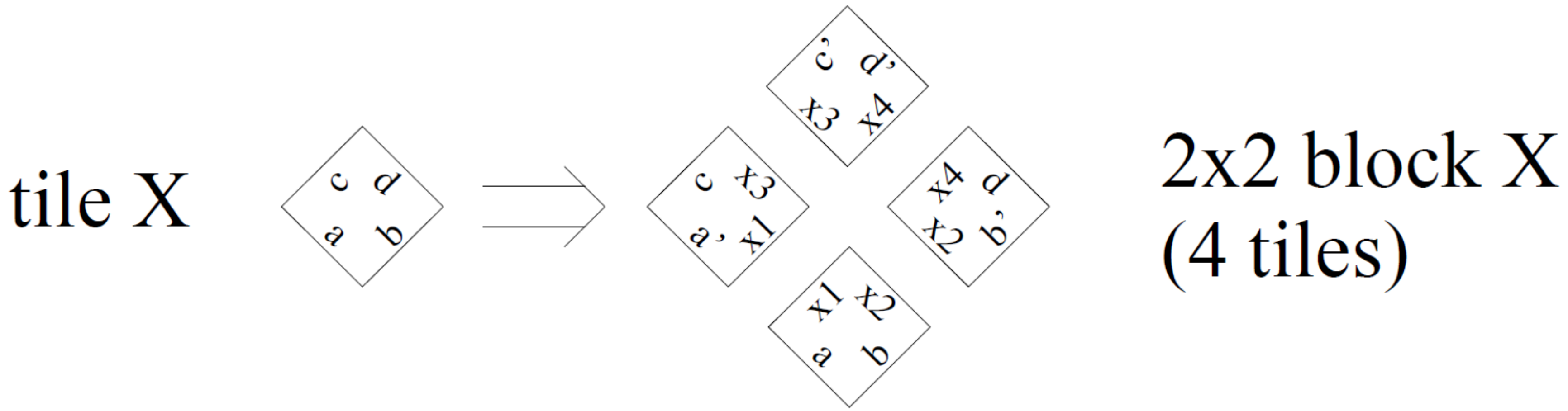


glues internal to block are all unique

errors must occur in multiples of 2

Proofreading: Error-correction in the kTAM

Definition: error = attachment by single strength 1 glue



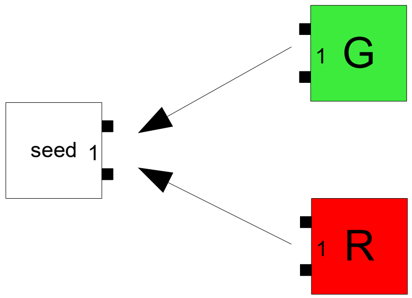
glues internal to block are all unique

errors must occur in multiples of 2

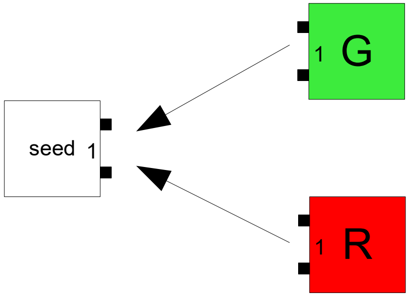
$k \times k$ proofreading roughly turns error rate of ε into ε^k

Concentration programming

Nondeterministic binding



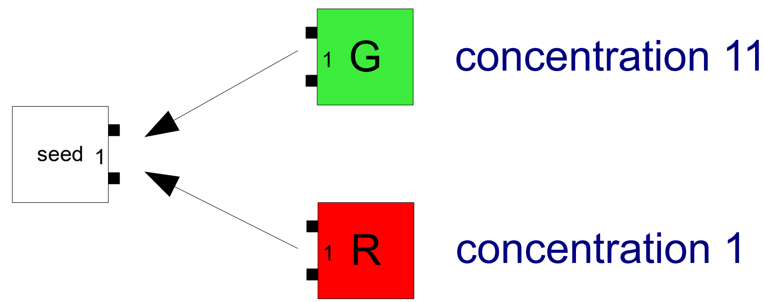
Nondeterministic binding

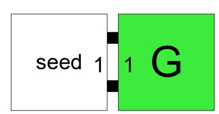


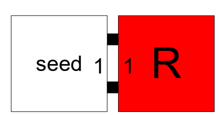
concentration 11

concentration 1

Nondeterministic binding



Pr[] = 11/12

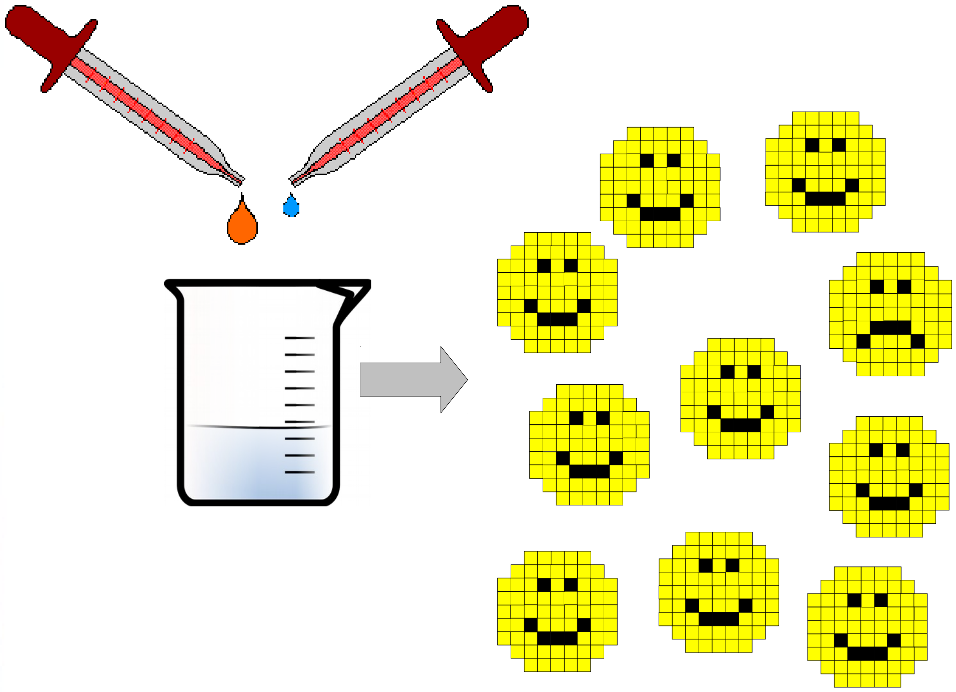
Pr[] = 1/12

Concentration programming of universal self-assembling molecules

A **singly-seeded** TAS can assemble *any* finite (scaled) shape (with high probability) by mixing them in the right concentrations.

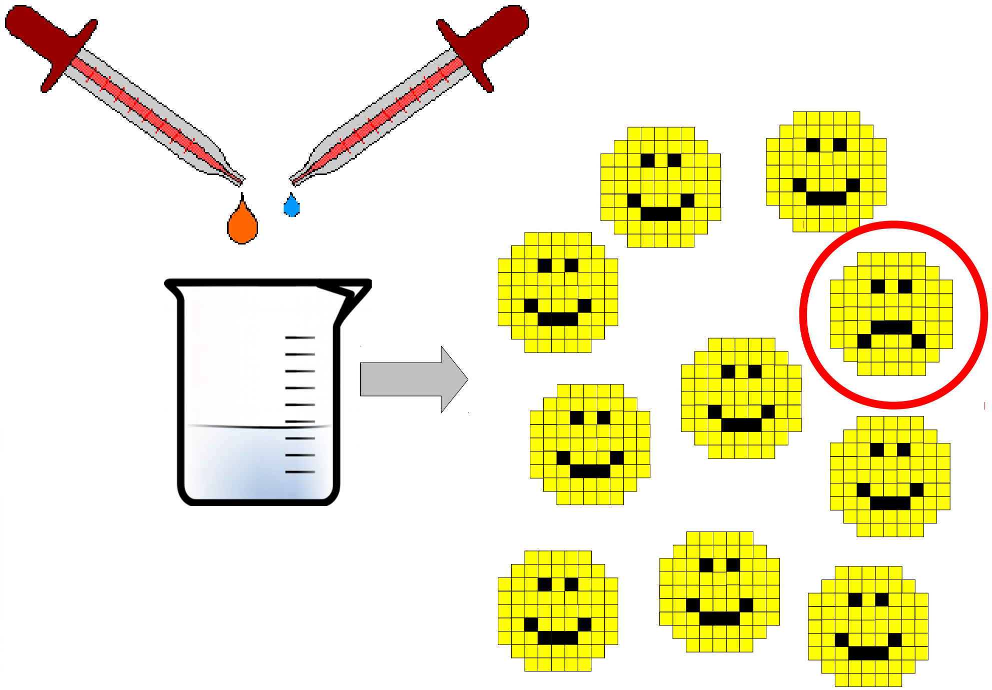
Concentration programming of universal self-assembling molecules

A **singly-seeded** TAS can assemble *any* finite (scaled) shape (with high probability) by mixing them in the right concentrations.



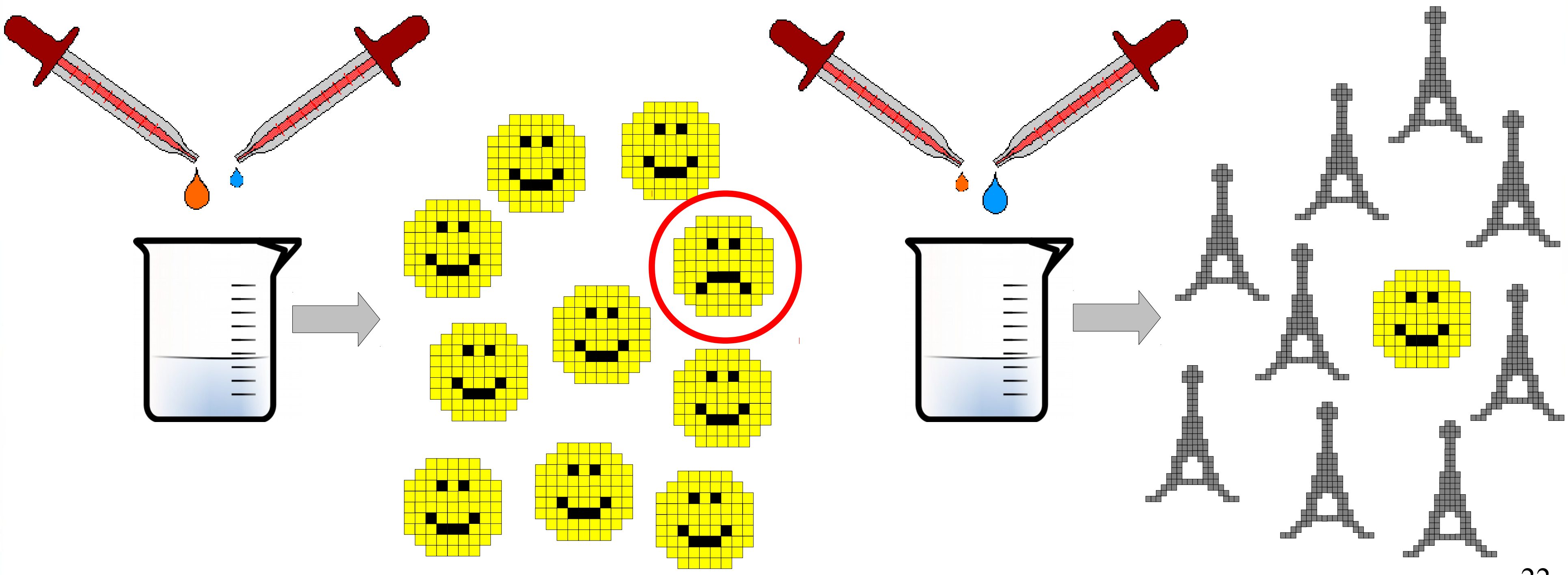
Concentration programming of universal self-assembling molecules

A **singly-seeded** TAS can assemble *any* finite (scaled) shape (with high probability) by mixing them in the right concentrations.



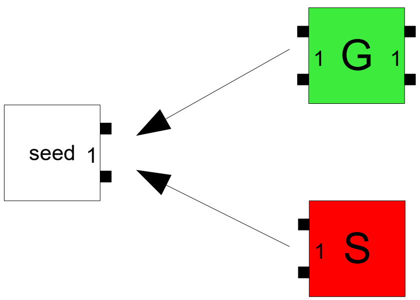
Concentration programming of universal self-assembling molecules

A **singly-seeded** TAS can assemble *any* finite (scaled) shape (with high probability) by mixing them in the right concentrations.



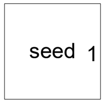
Programming polymer length with concentrations

[Becker, Rapaport, Rémila, *FSTTCS* 2006]



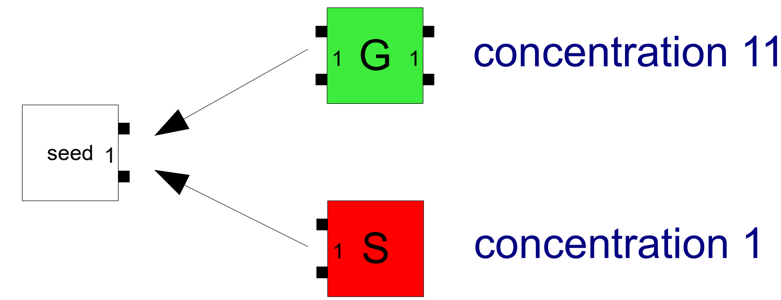
concentration 11

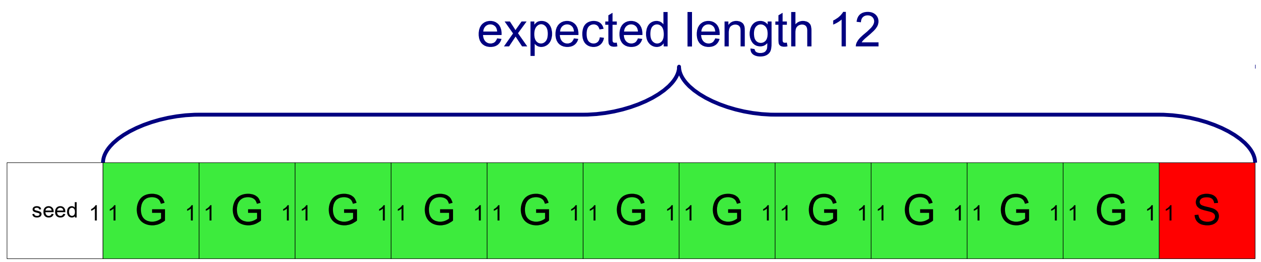
concentration 1



Programming polymer length with concentrations

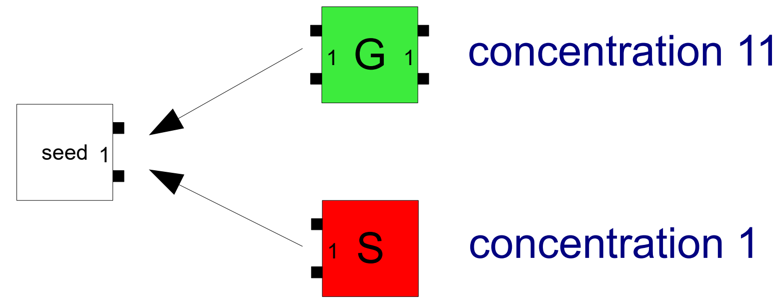
[Becker, Rapaport, Rémila, *FSTTCS* 2006]

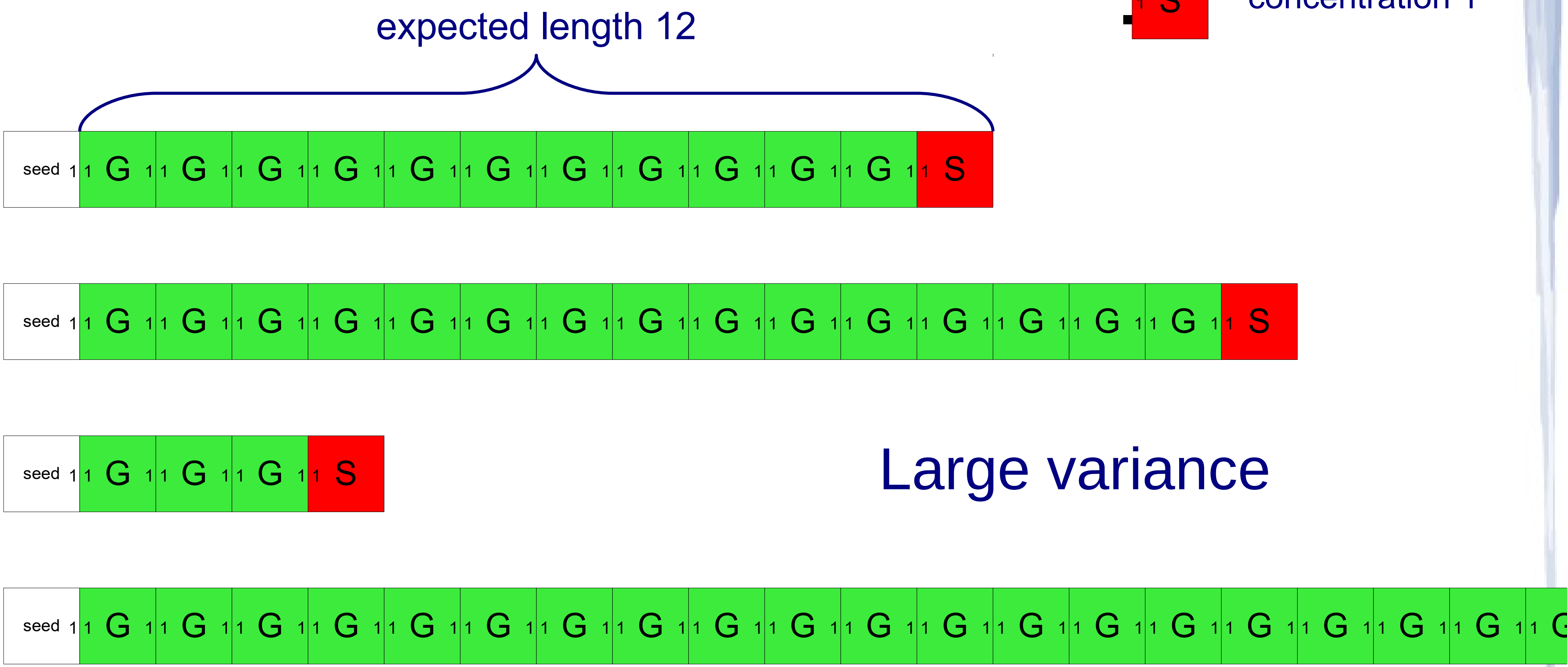




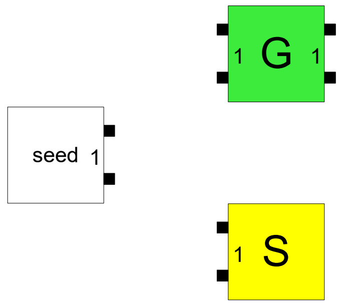
Programming polymer length with concentrations

[Becker, Rapaport, Rémila, *FSTTCS* 2006]





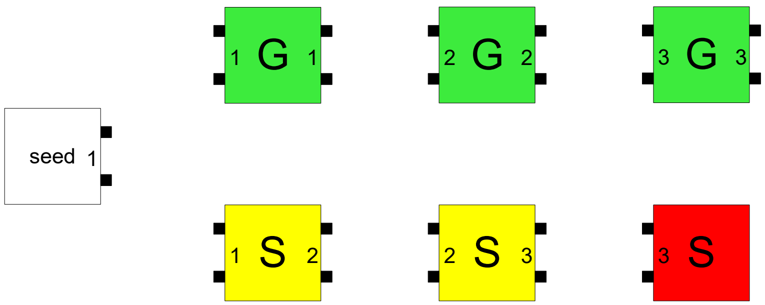
Programming polymer length (improved)



concentration 3

concentration 1

Programming polymer length (improved)

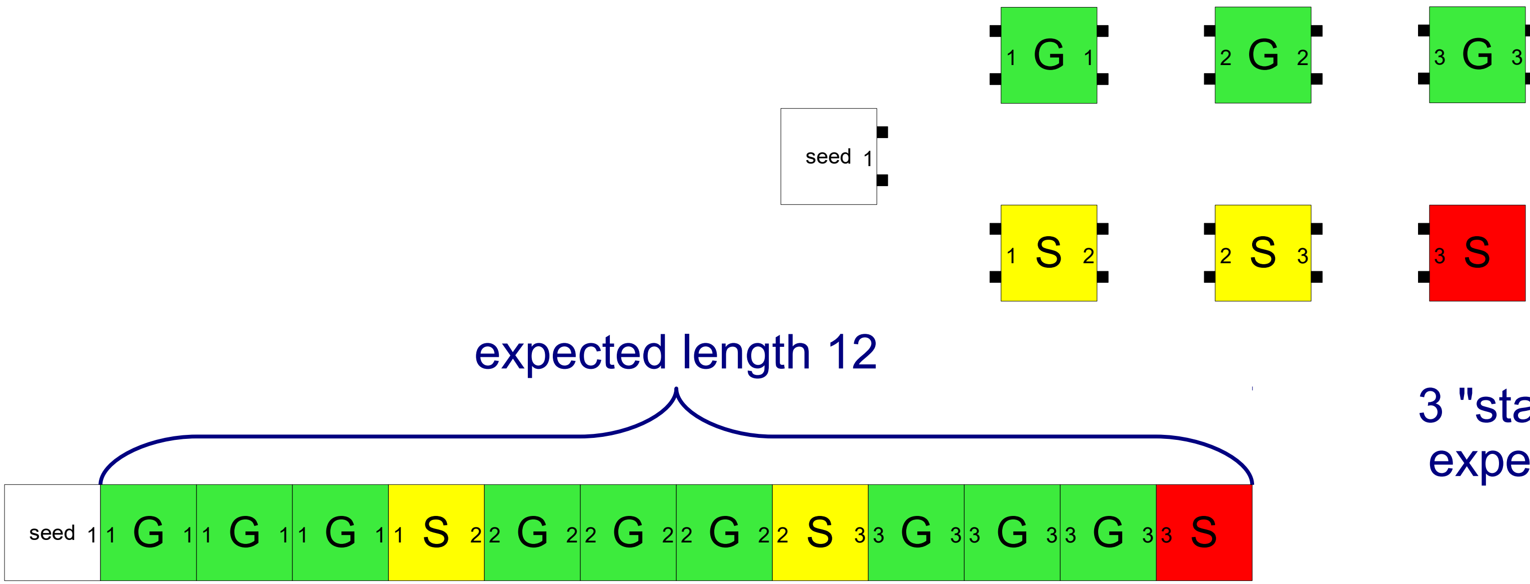


concentration 3

concentration 1

3 "stages", each of
expected length 4

Programming polymer length (improved)

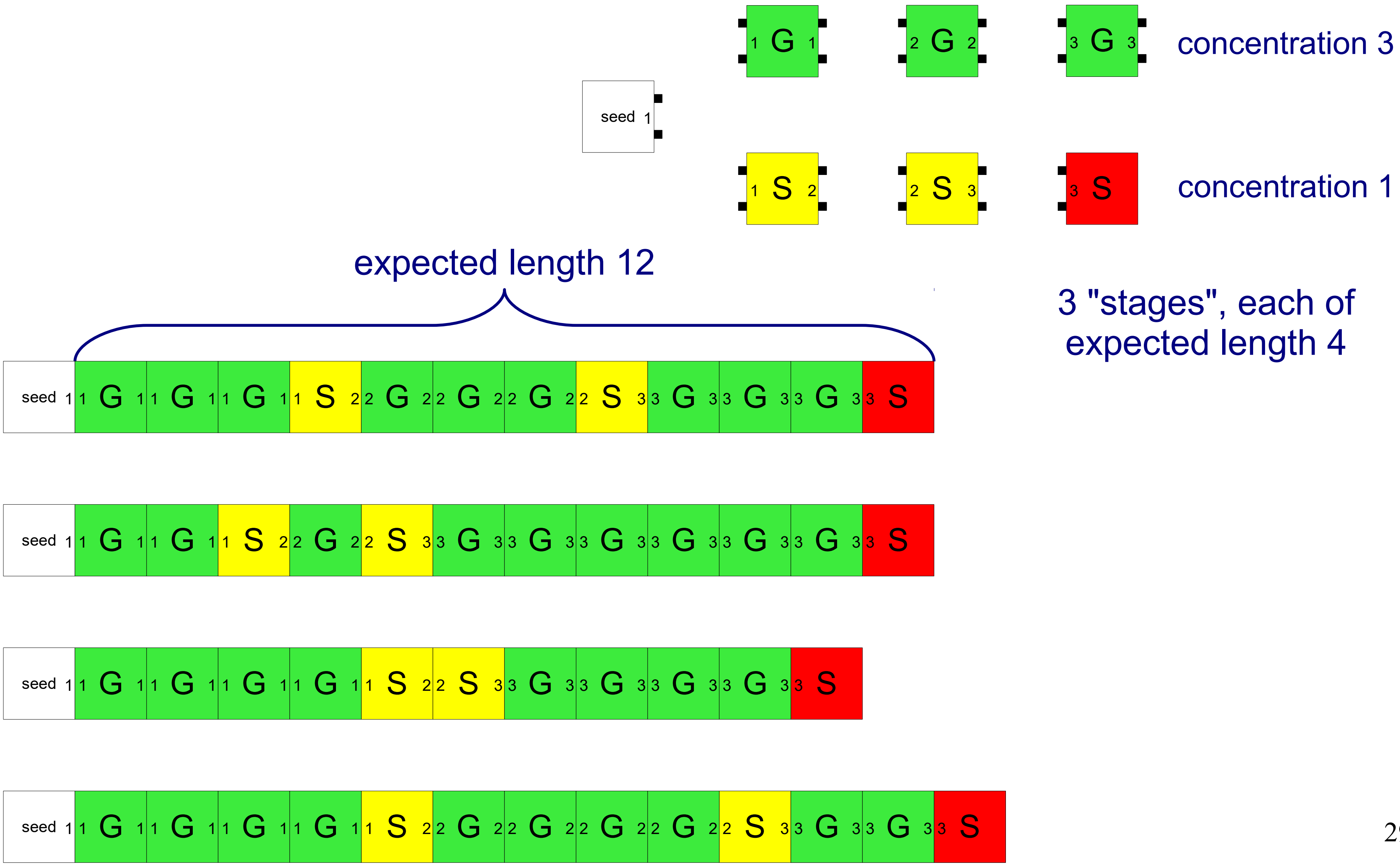


concentration 3

concentration 1

3 "stages", each of expected length 4

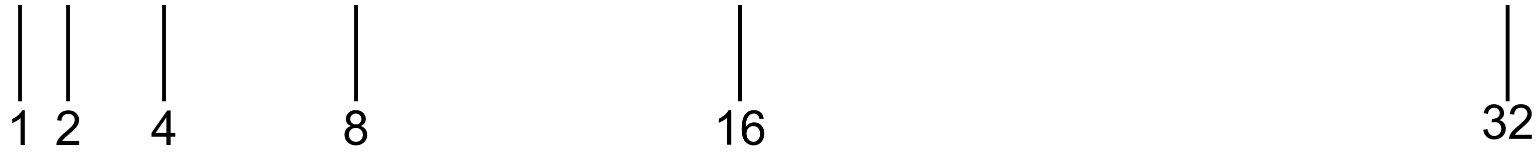
Programming polymer length (improved)



Programming polymer length (improved)

90 stages, expected length midway in $[2^{a-1}, 2^{a})$

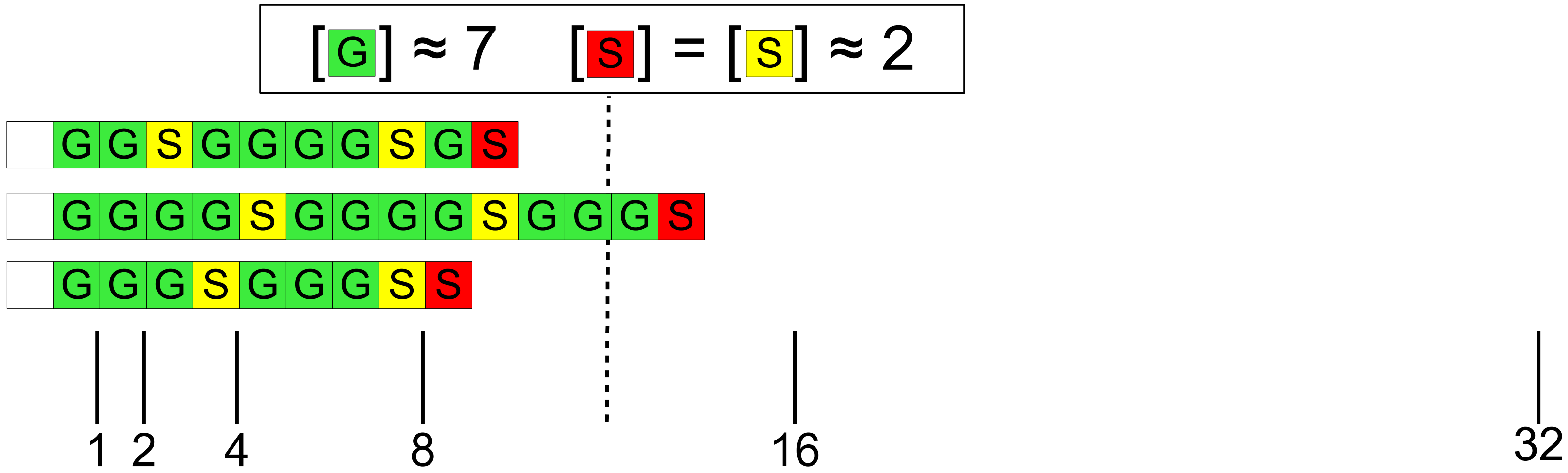
→ with probability > 99% **actual** length in $[2^{a-1}, 2^{a})$



Programming polymer length (improved)

90 stages, expected length midway in $[2^{a-1}, 2^{a})$

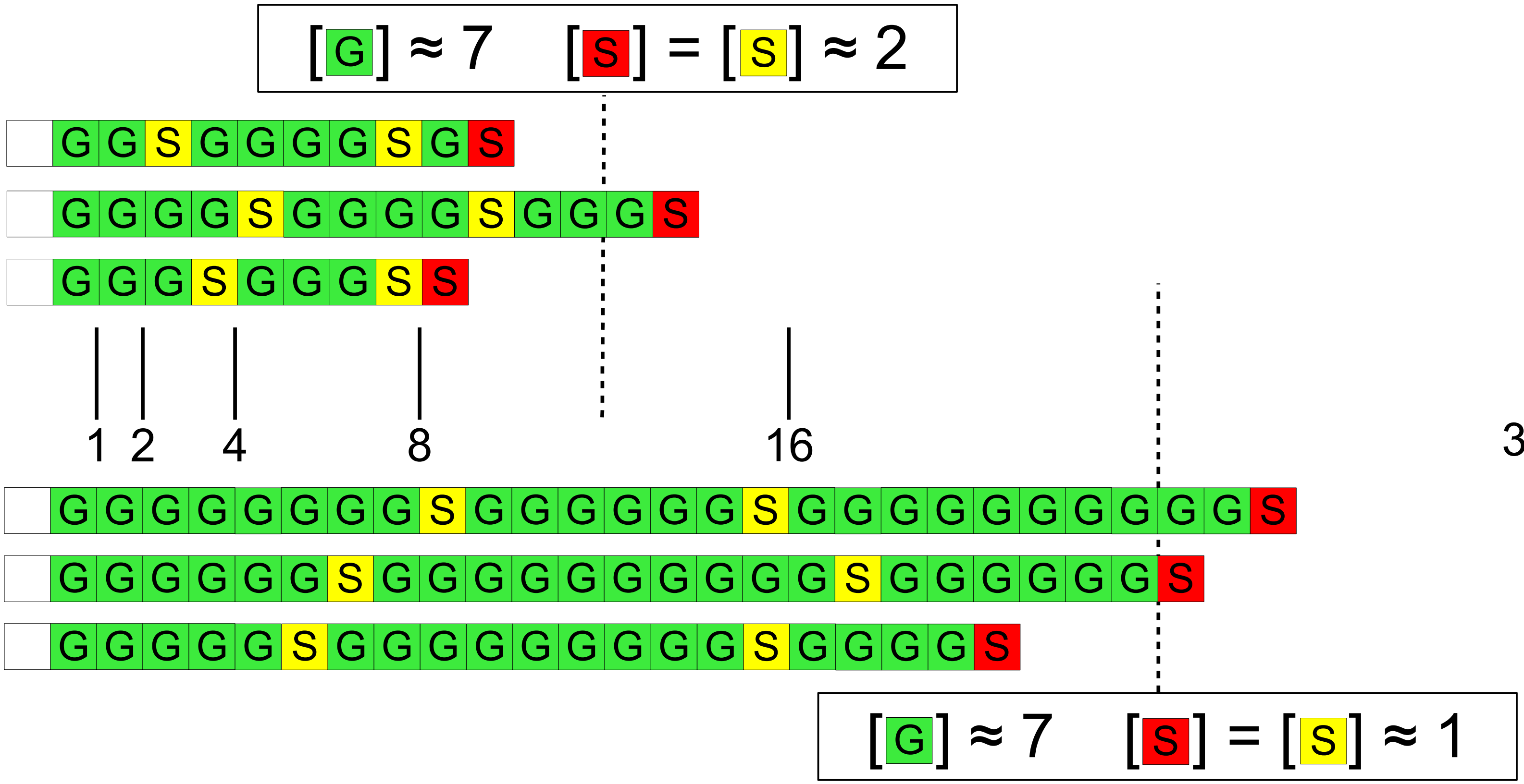
→ with probability > 99% **actual** length in $[2^{a-1}, 2^{a})$



Programming polymer length (improved)

90 stages, expected length midway in $[2^{a-1}, 2^a)$

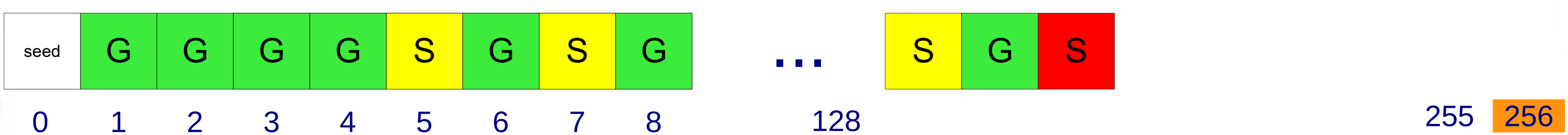
→ with probability > 99% **actual** length in $[2^{a-1}, 2^a)$



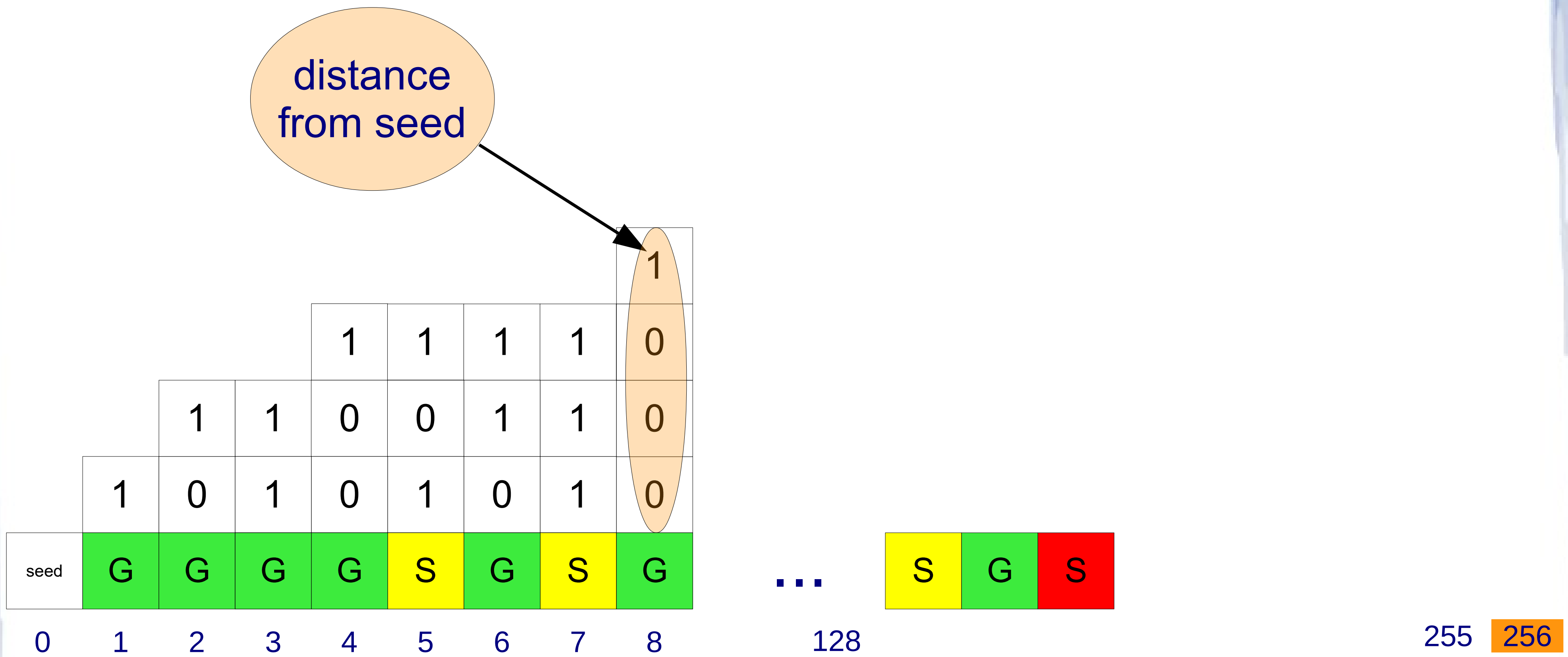
Programming polymer length 2^a precisely

256

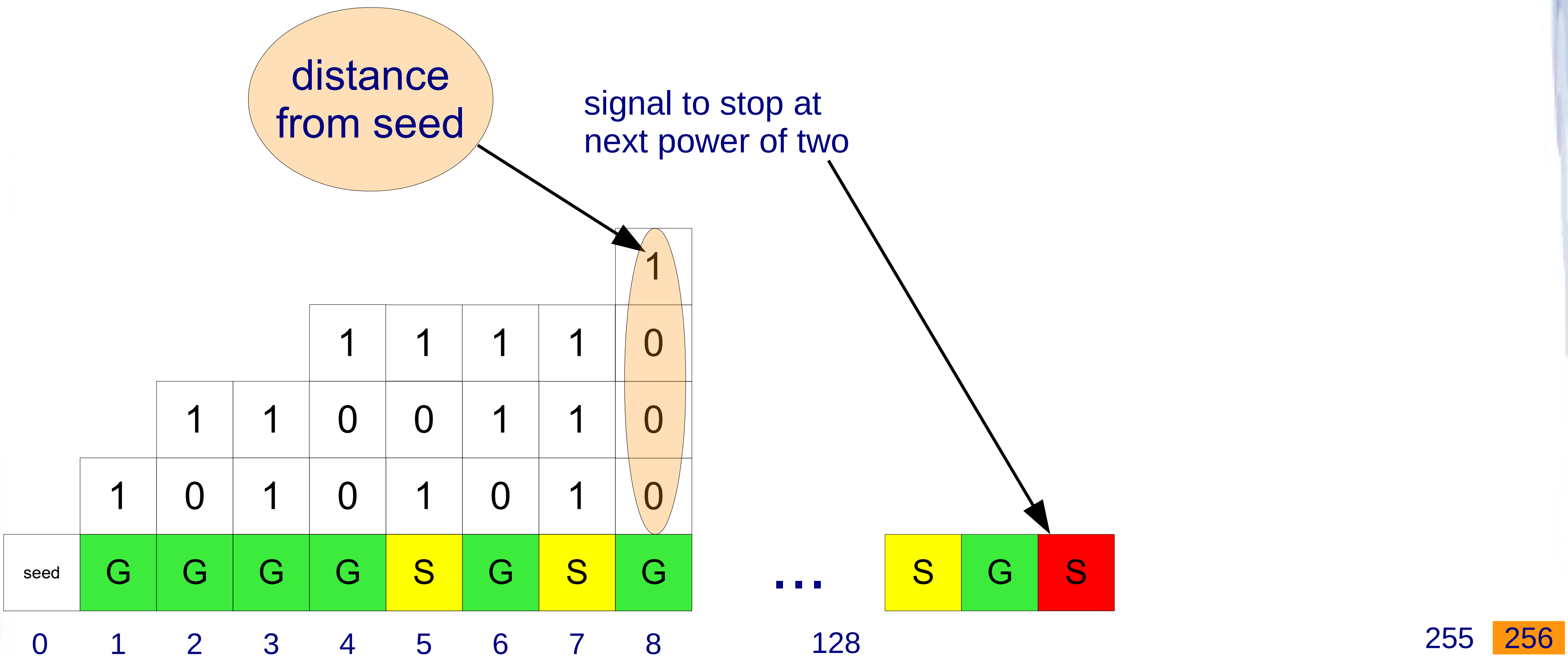
Programming polymer length 2^a precisely



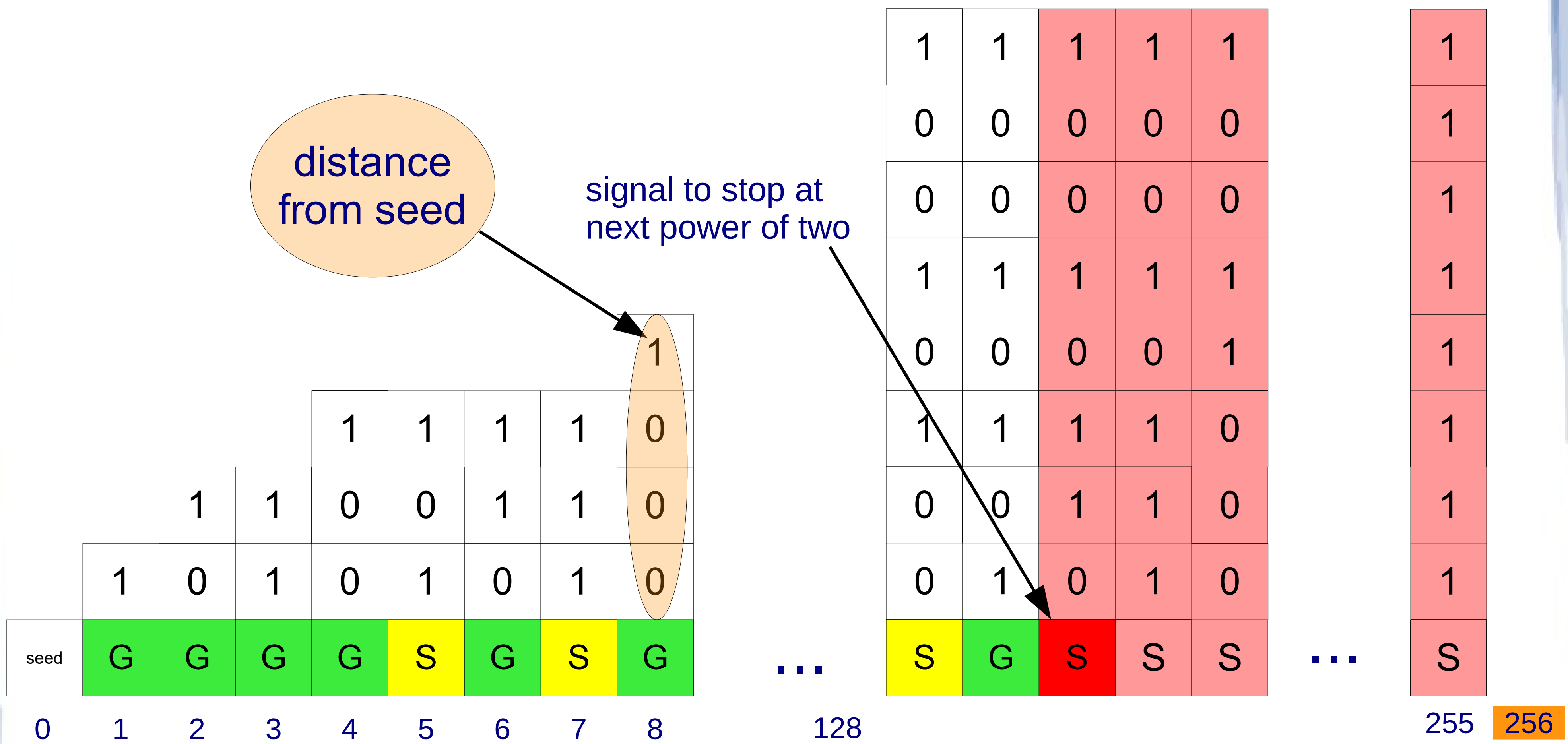
Programming polymer length 2^a precisely



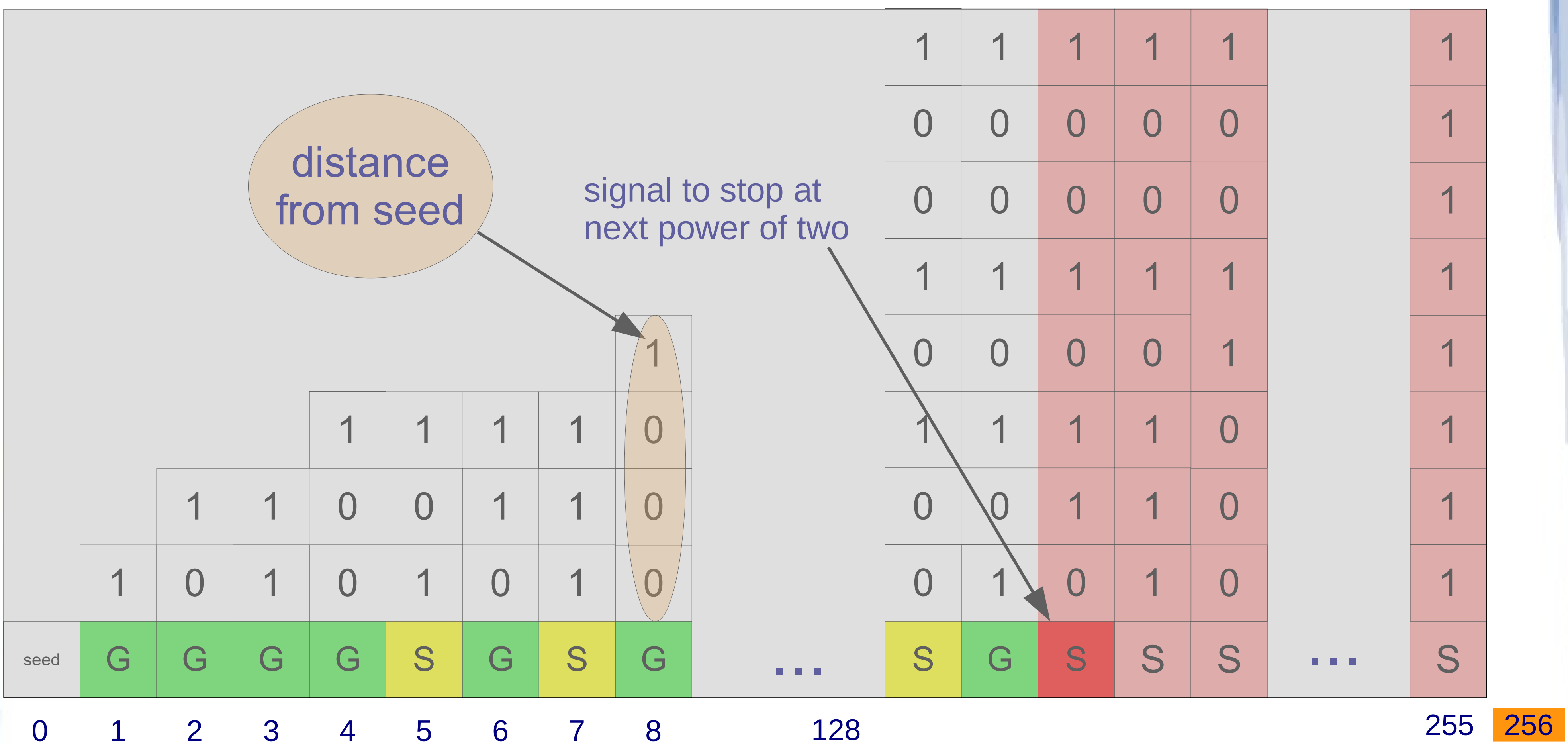
Programming polymer length 2^a precisely



Programming polymer length 2^a precisely



Programming polymer length 2^a precisely

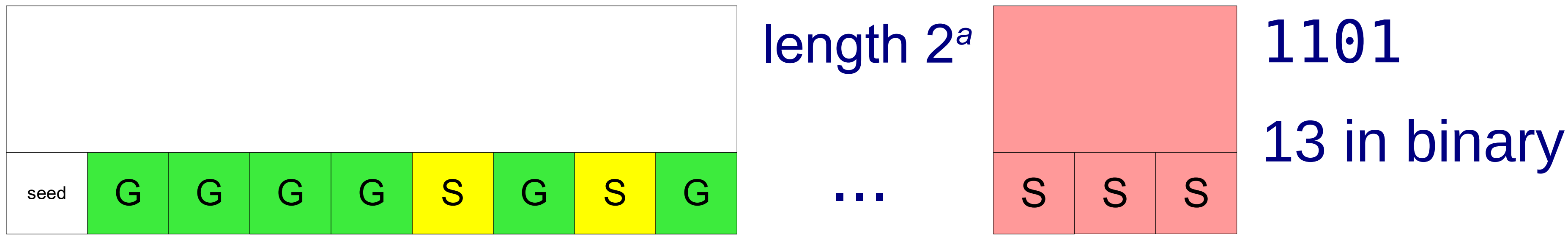


Programming a binary string

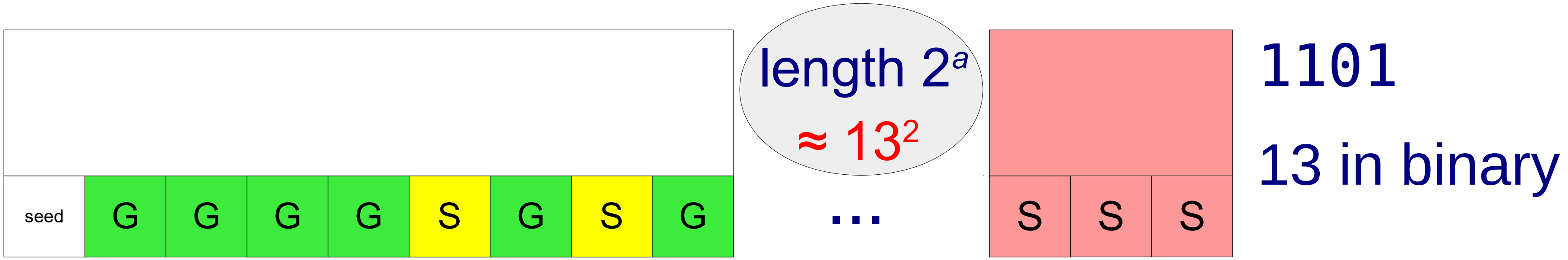
`1101`

13 in binary

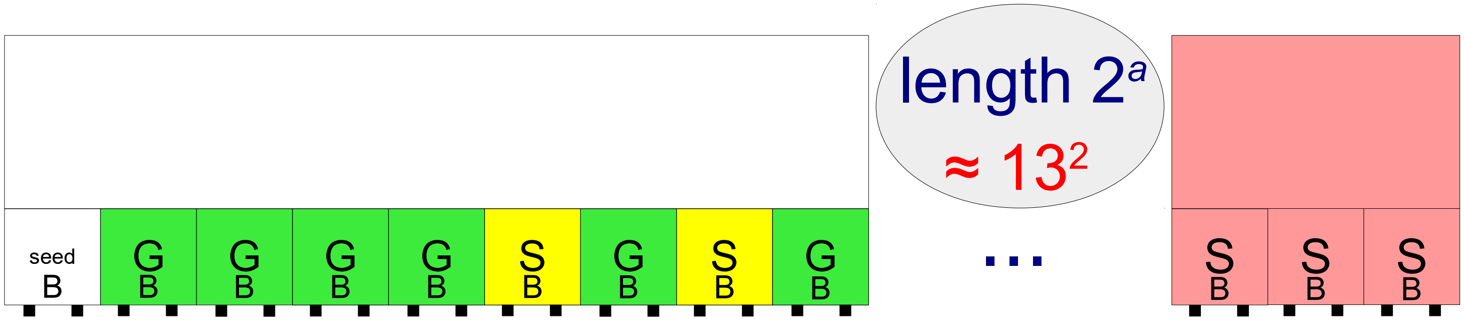
Programming a binary string



Programming a binary string

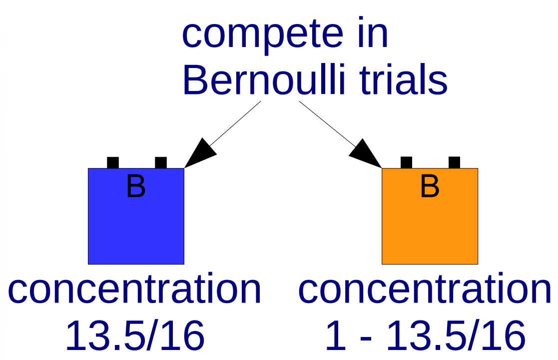


Programming a binary string

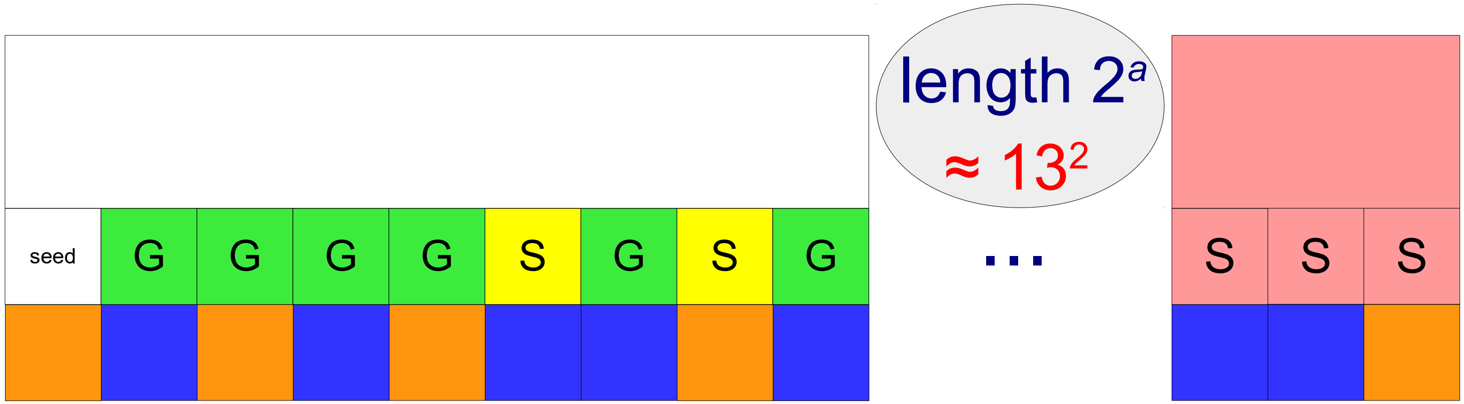


1101

13 in binary



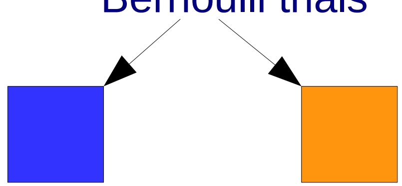
Programming a binary string



1101

13 in binary

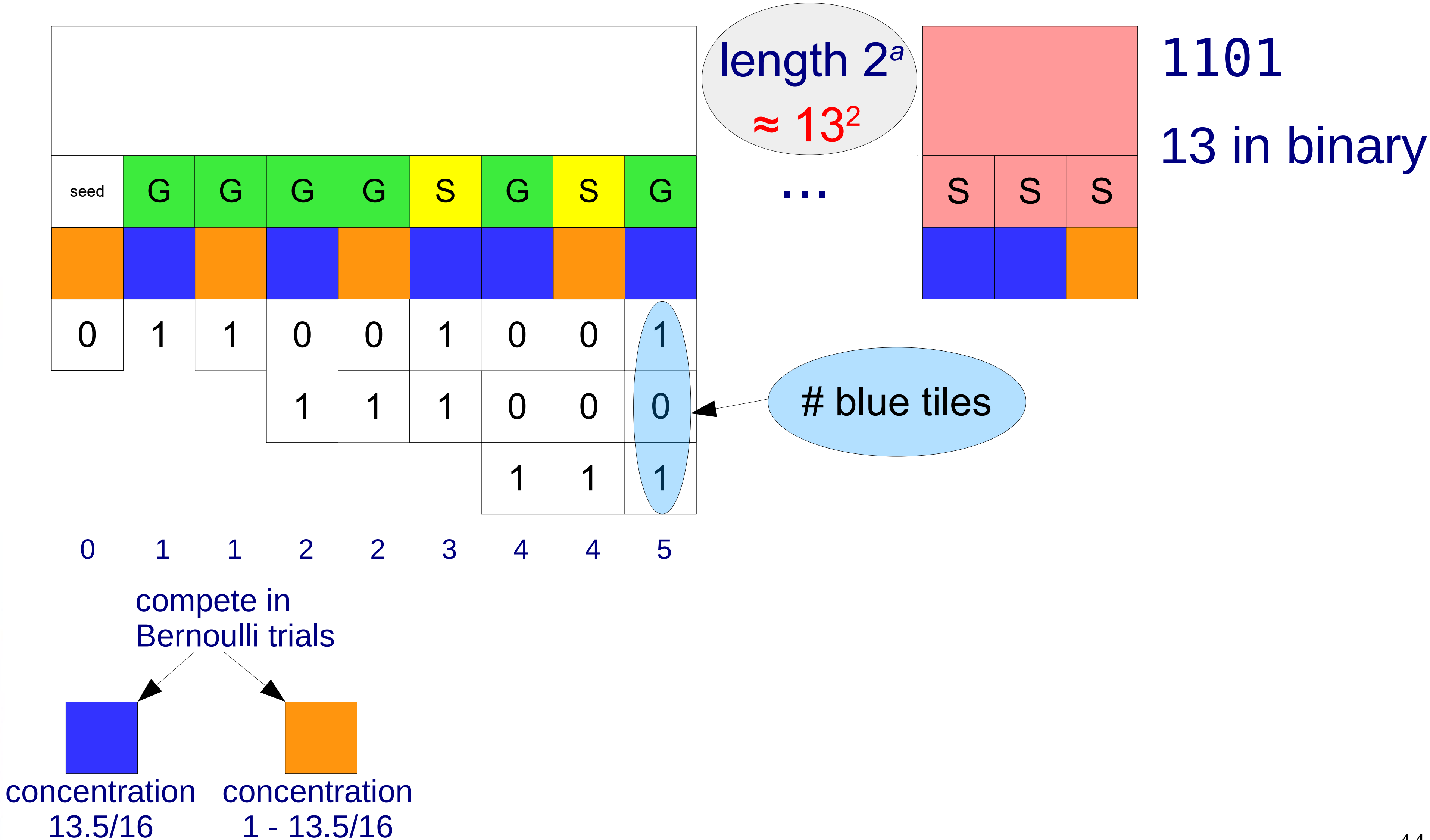
compete in
Bernoulli trials



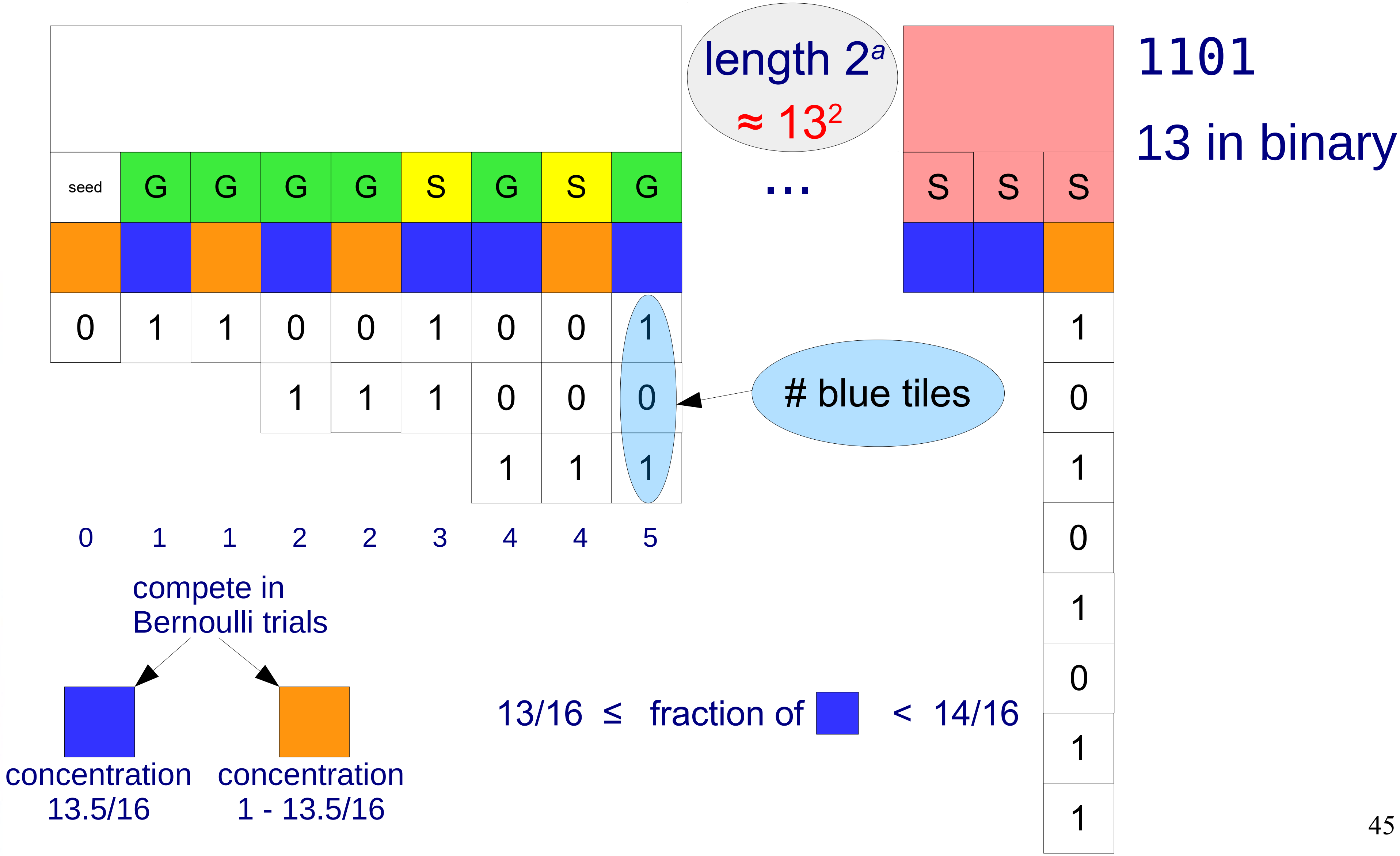
concentration
13.5/16

concentration
1 - 13.5/16

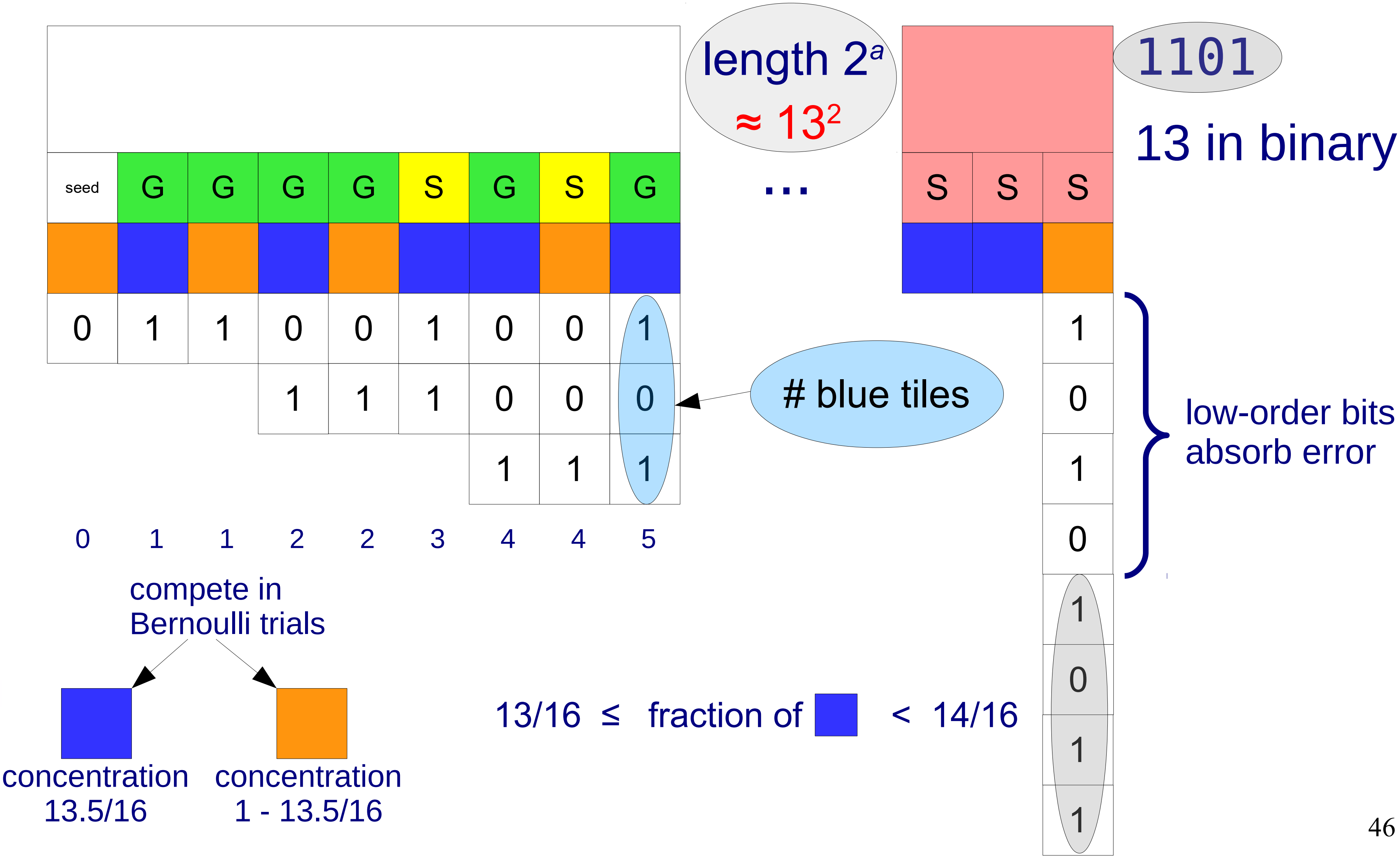
Programming a binary string



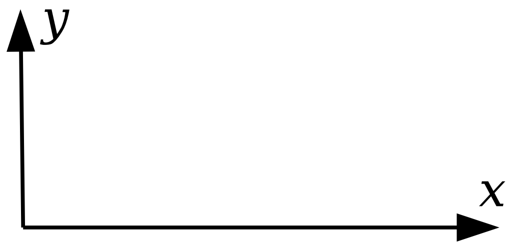
Programming a binary string



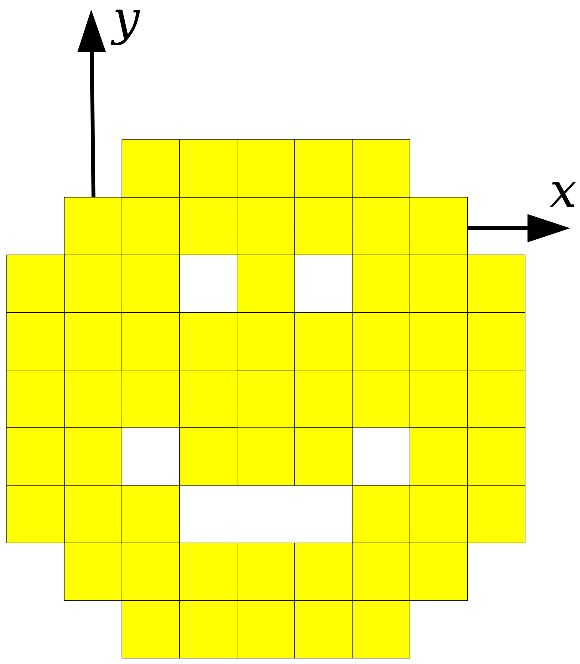
Programming a binary string



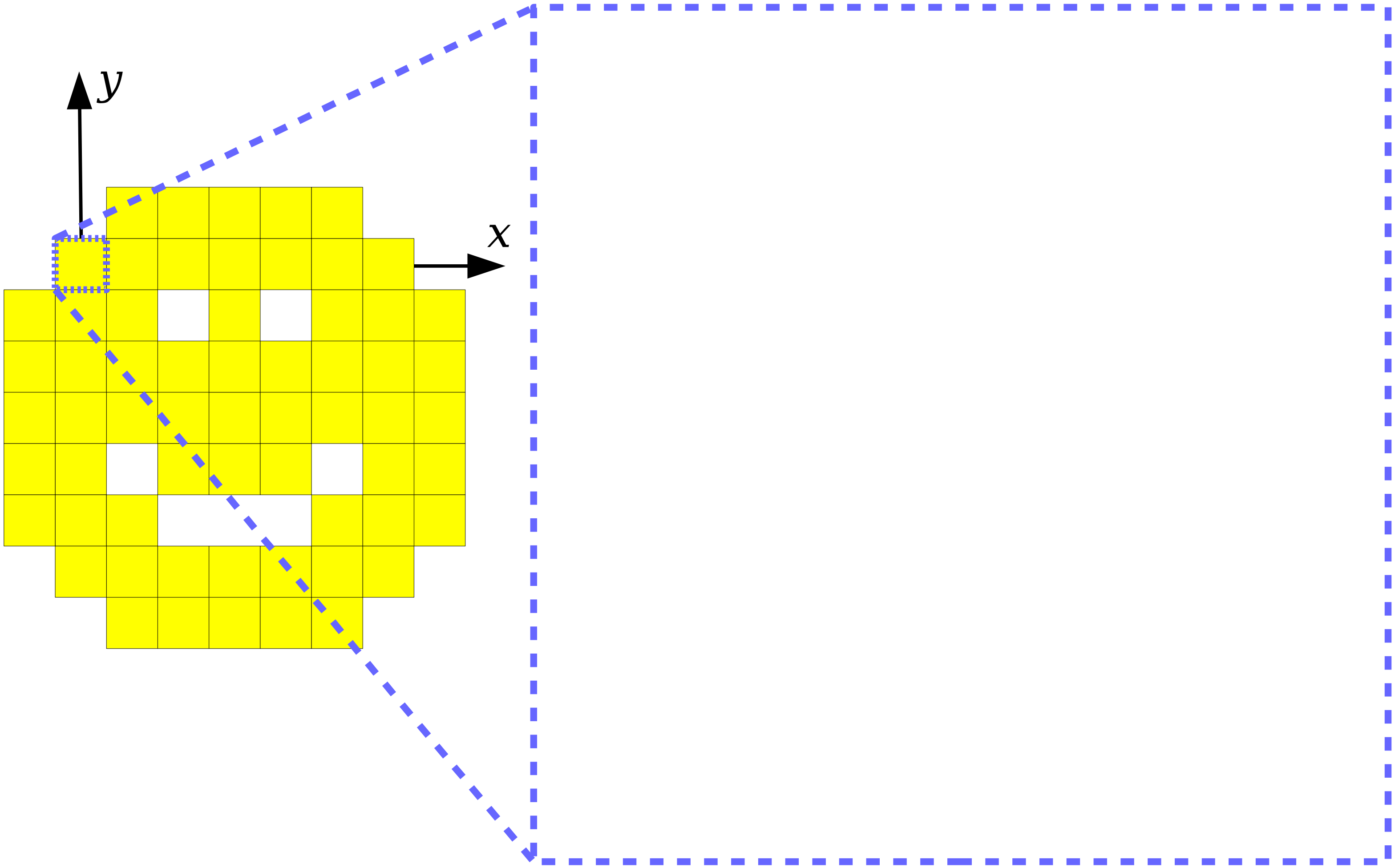
Programming a shape



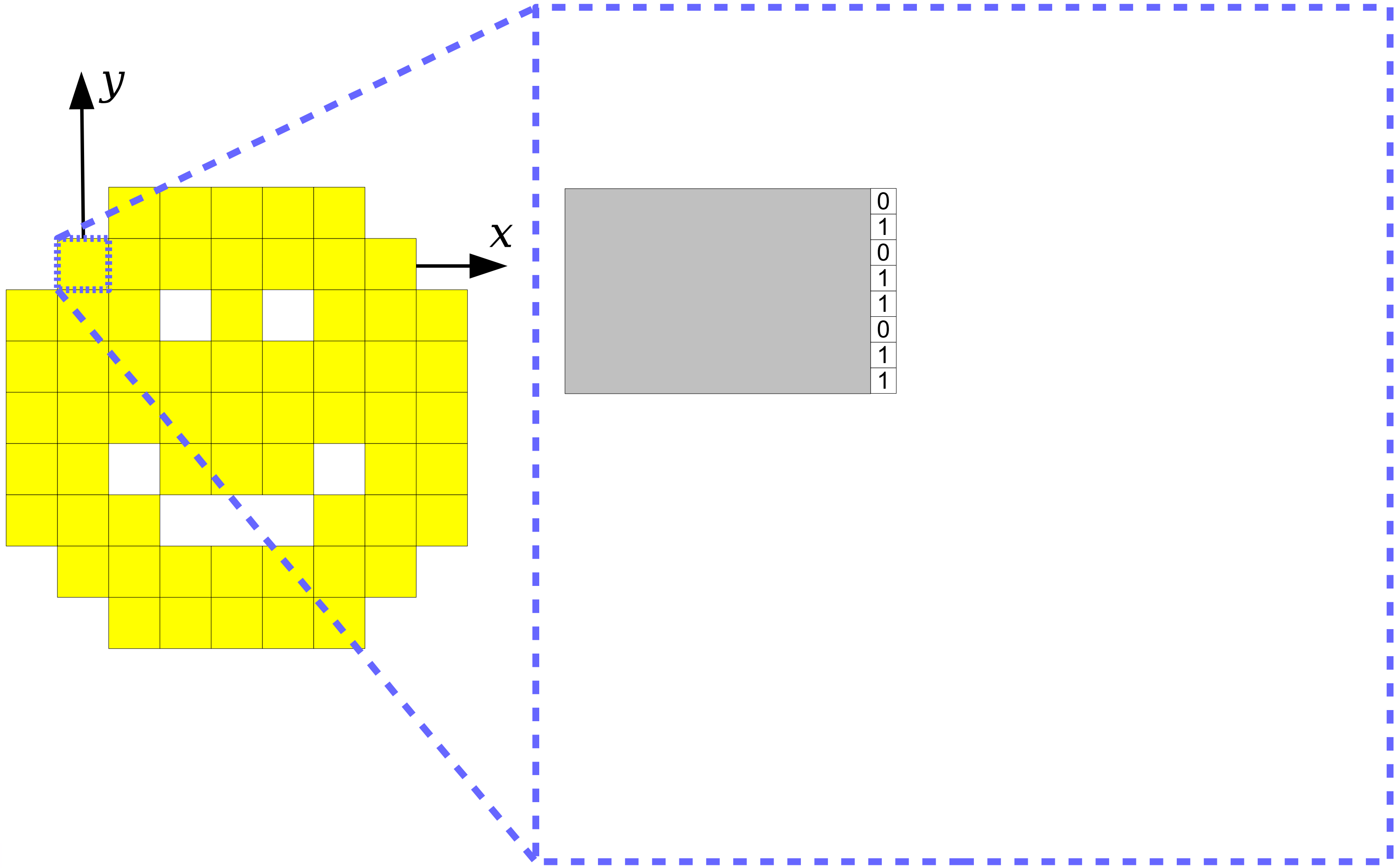
Programming a shape



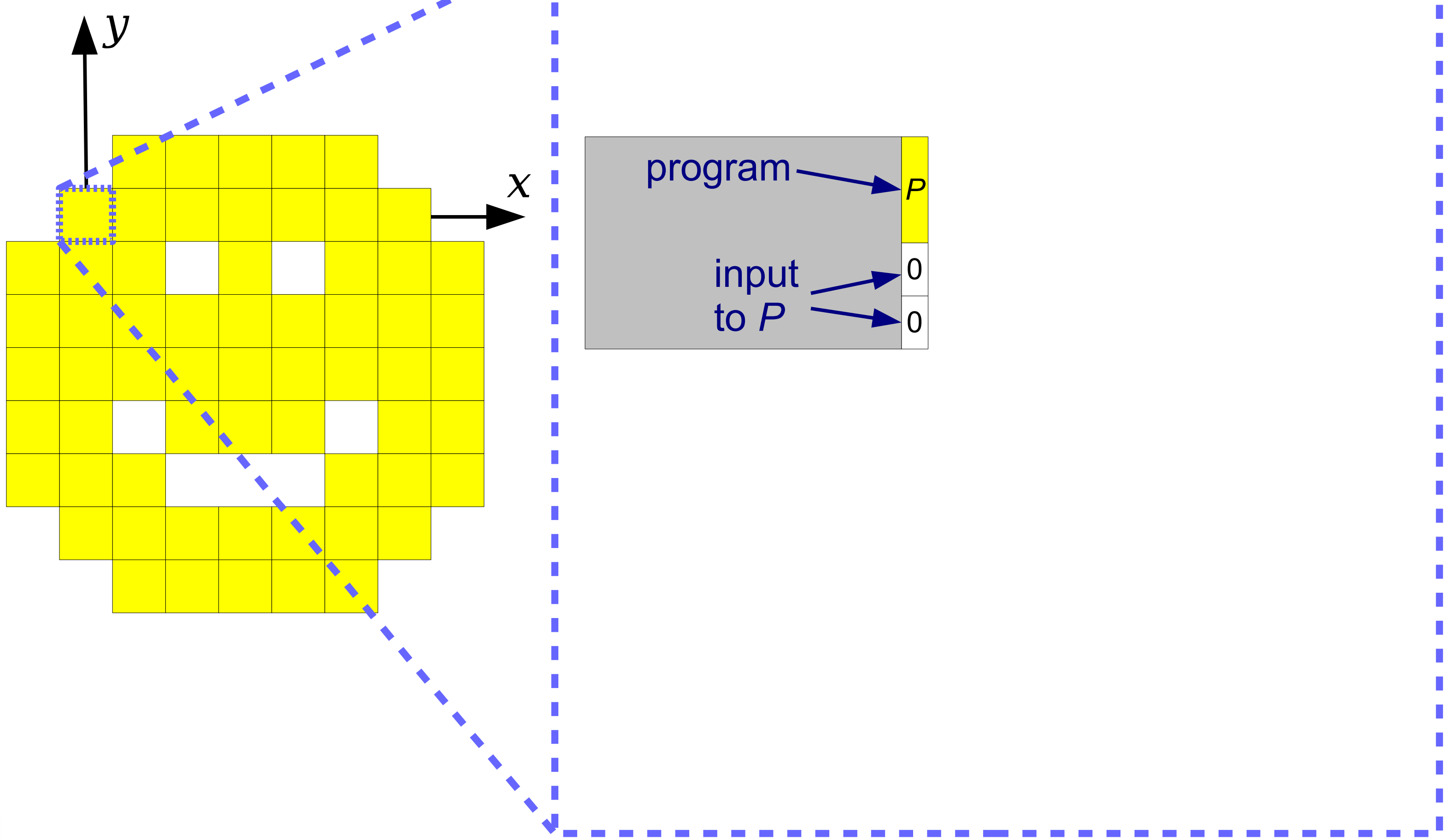
Programming a shape



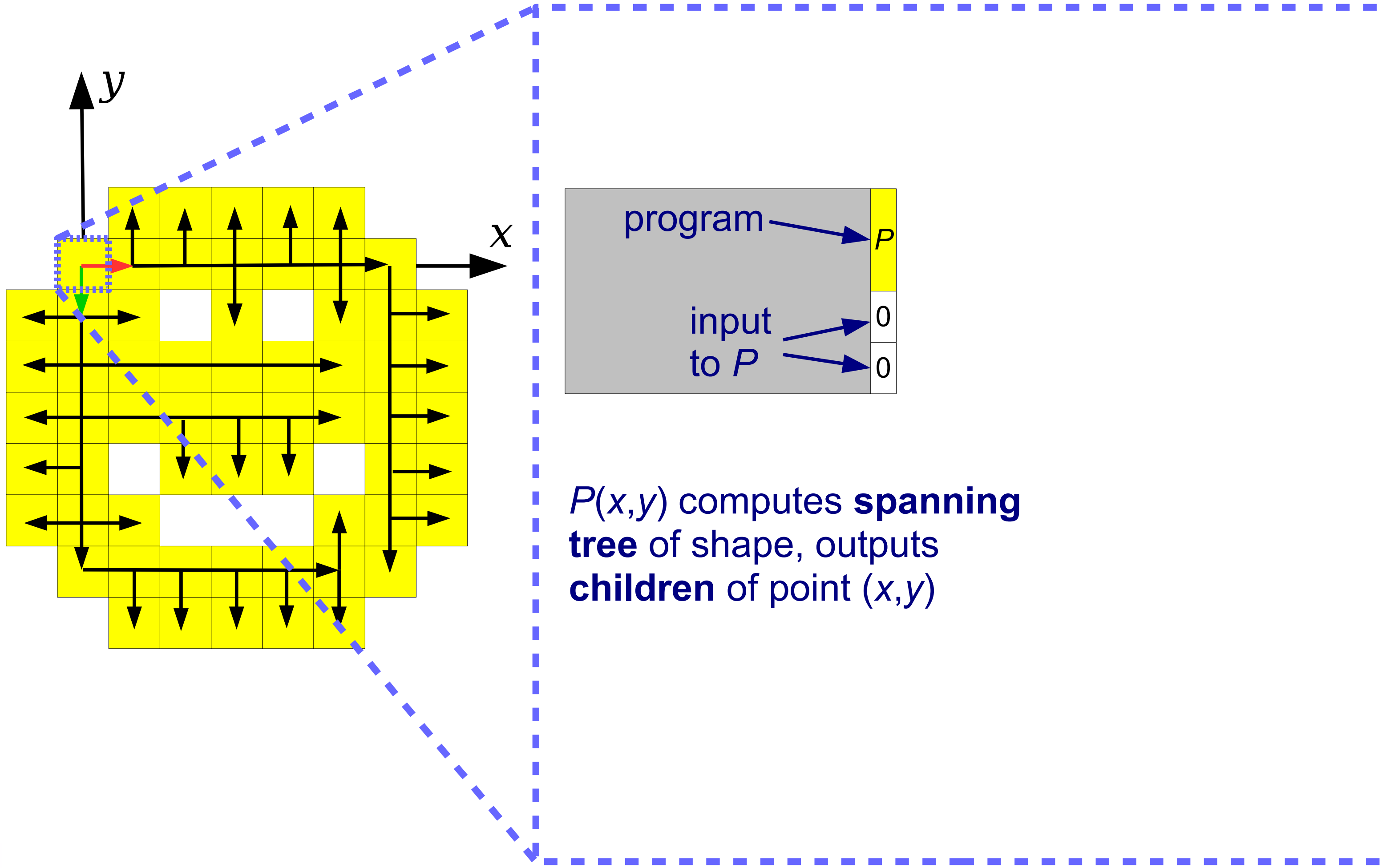
Programming a shape



Programming a shape

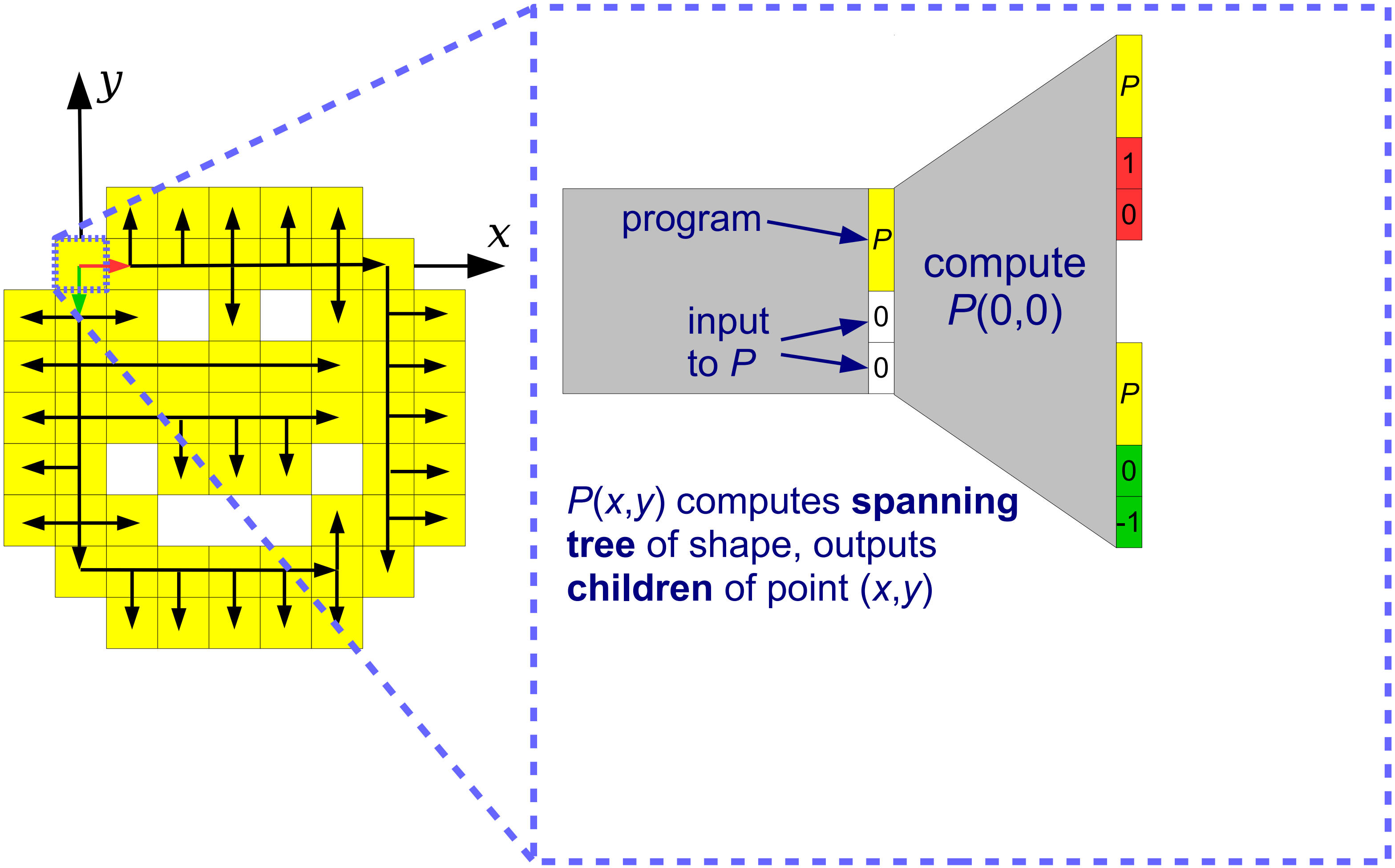


Programming a shape

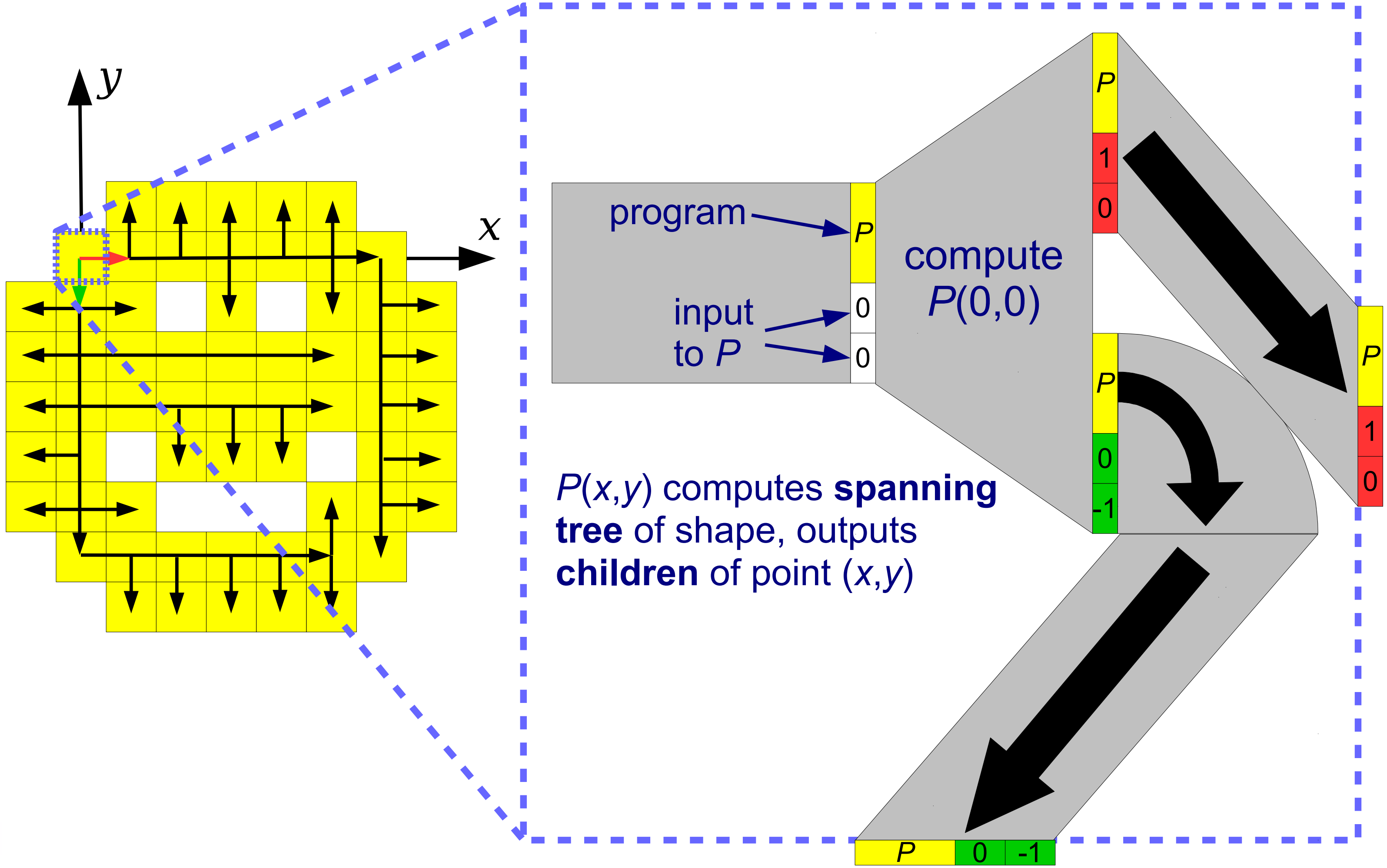


P(*x*,*y*) computes **spanning tree** of shape, outputs **children** of point (*x*,*y*)

Programming a shape

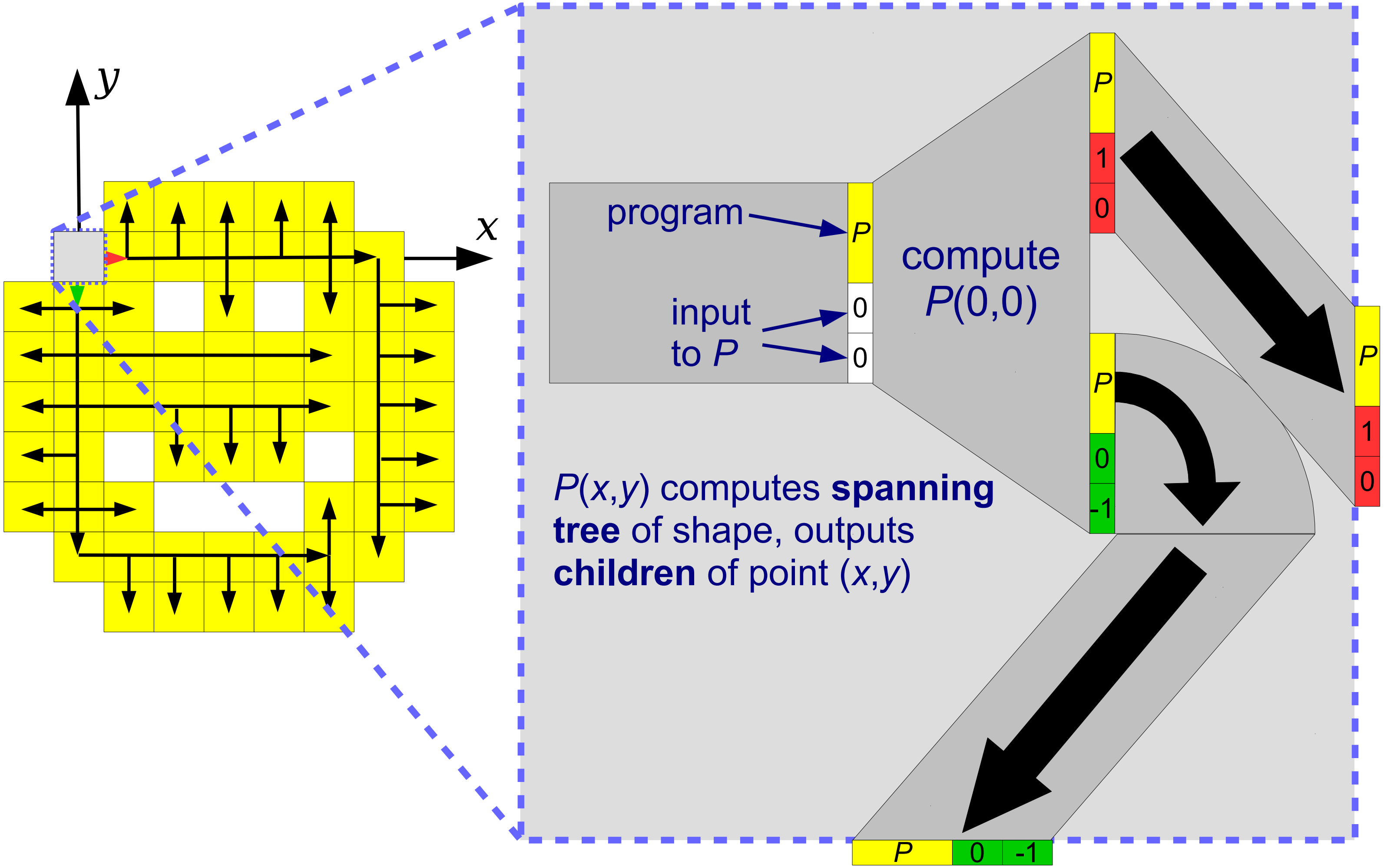


Programming a shape

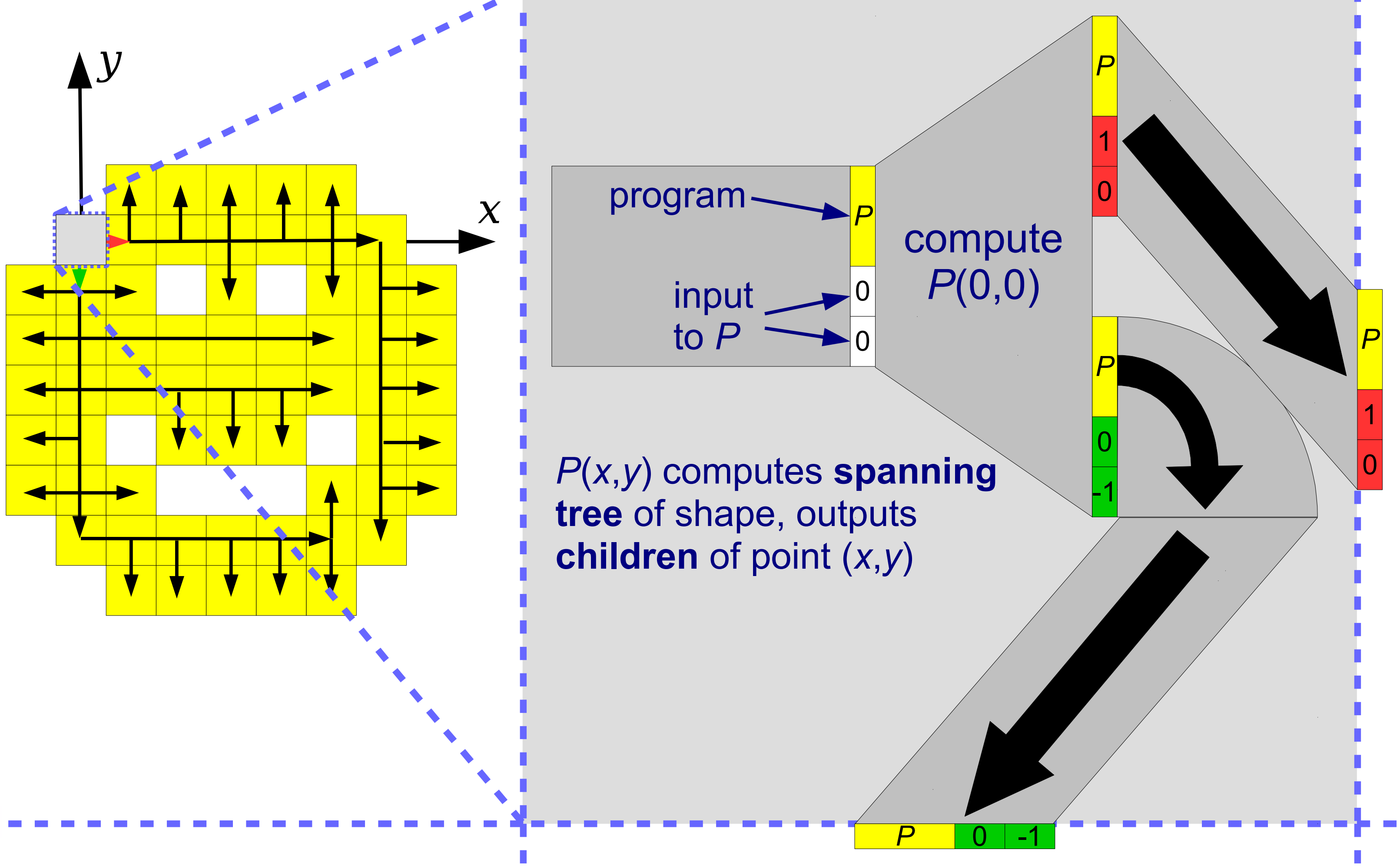


P(*x*,*y*) computes **spanning tree** of shape, outputs **children** of point (*x*,*y*)

Programming a shape

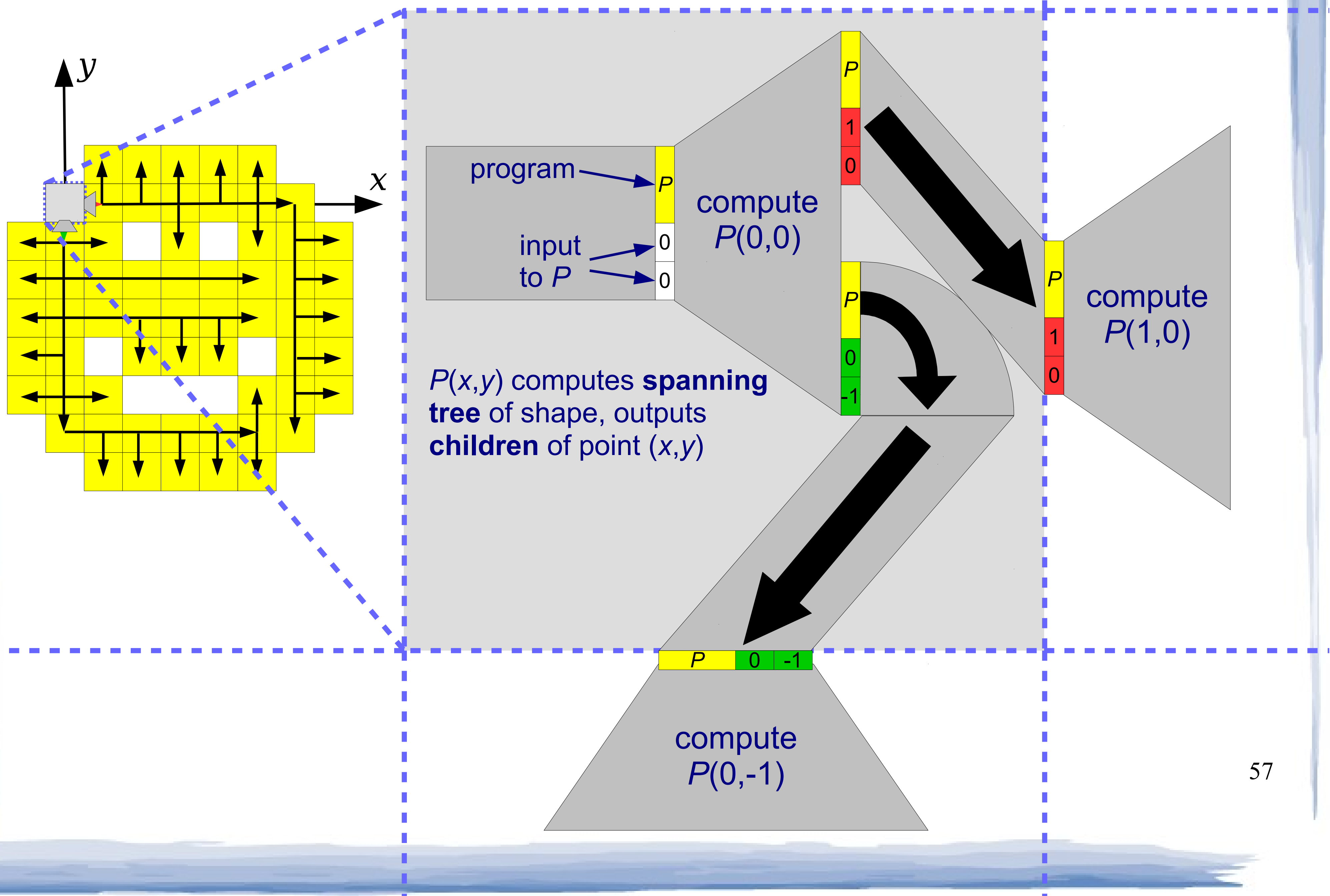


Programming a shape

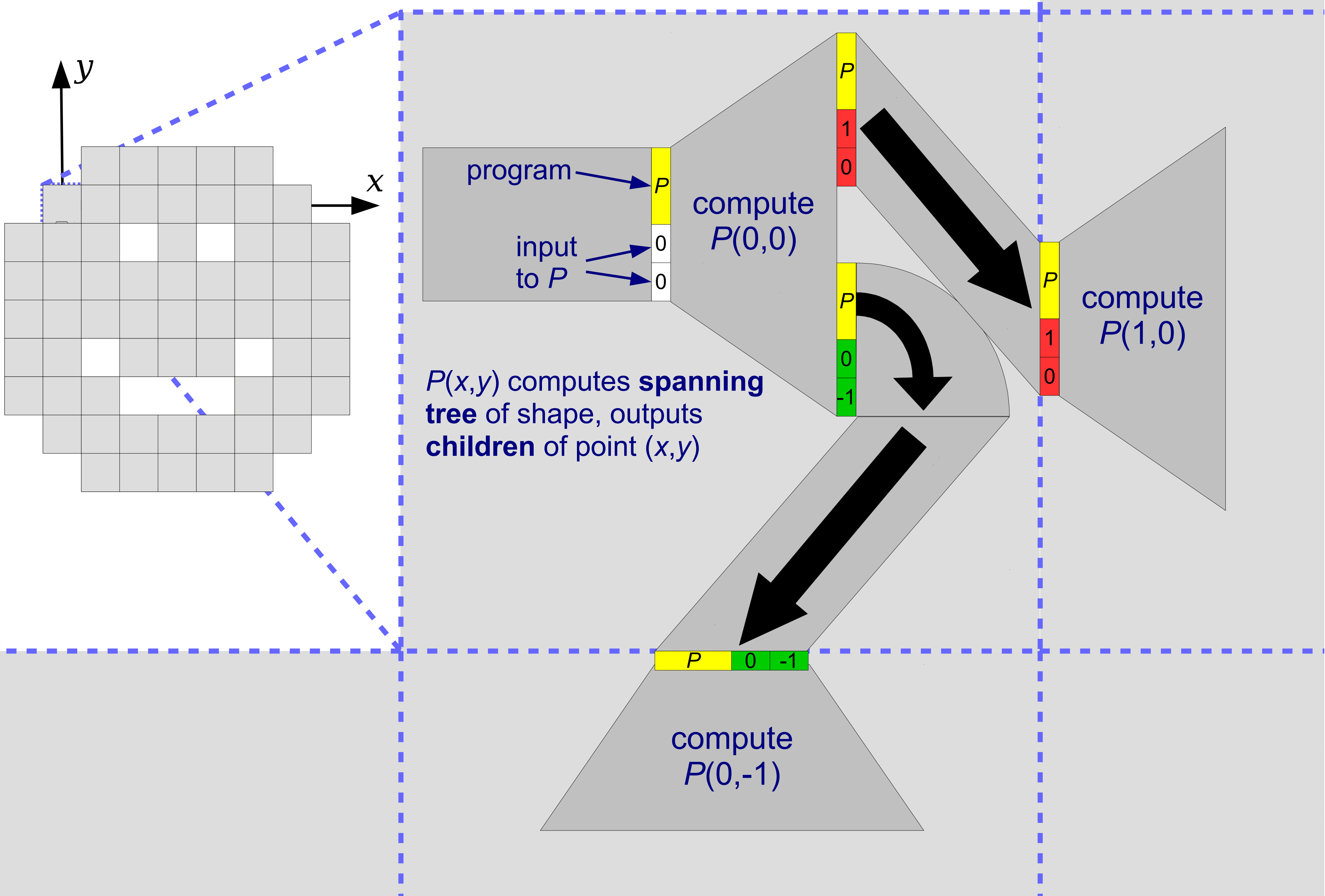


P(*x*,*y*) computes **spanning tree** of shape, outputs **children** of point (*x*,*y*)

Programming a shape



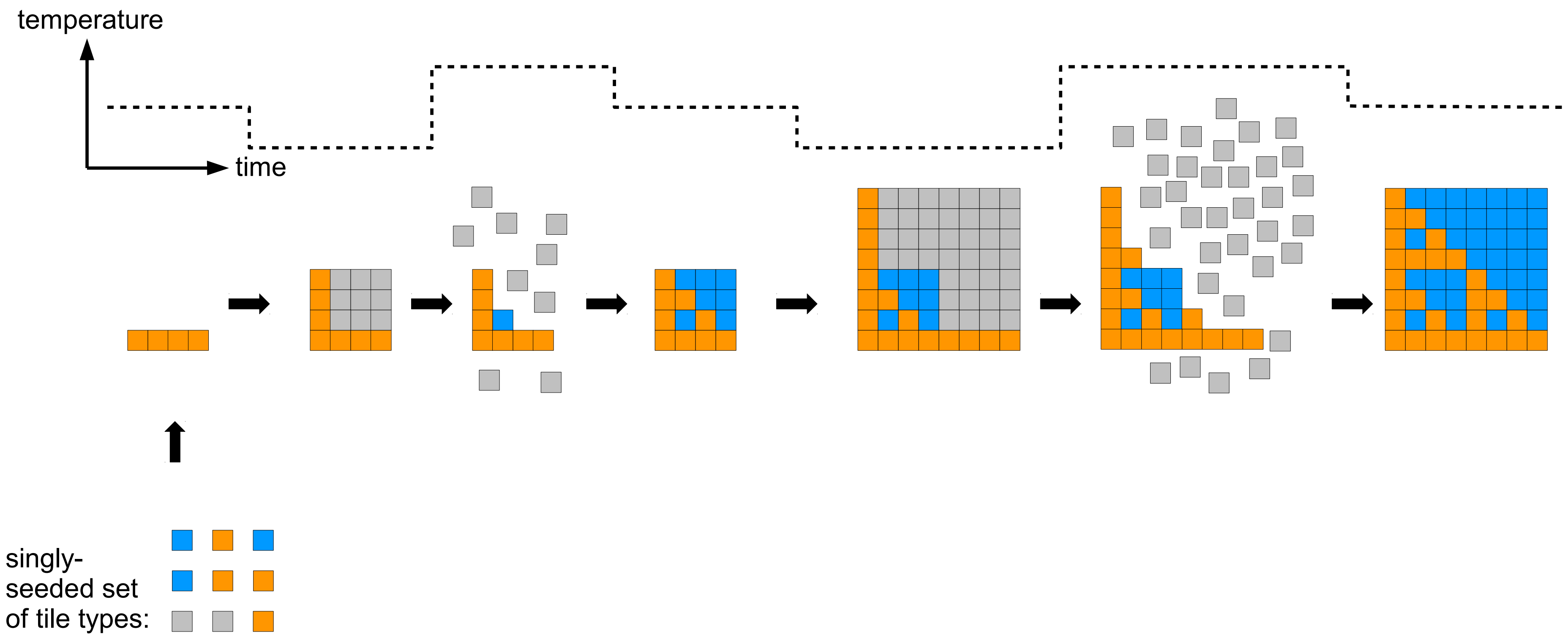
Programming a shape



Temperature programming

Temperature programming

(Kao, Schweller, SODA 2006): Vary temperature (binding strength threshold) throughout assembly to control what assembles.



Complexity of Temperature Programming

Scott Summers: A fixed set of (singly-seeded) tile types can assemble any finite scaled shape through temperature programming.

Number of tile types (a self-assembly "resource") is constant (maybe big), no matter the shape.

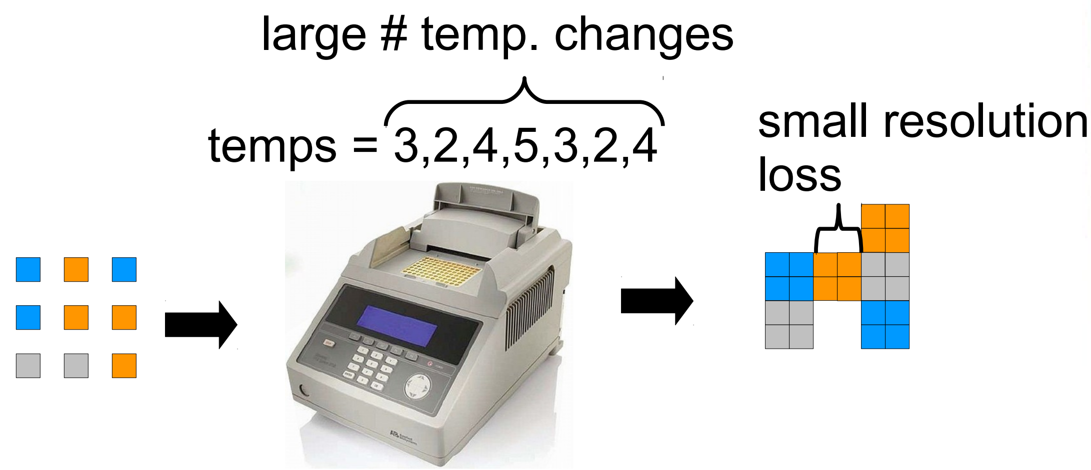
Scott wondered about two other self-assembly resources that might change for each shape:

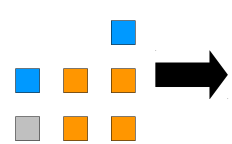
- What **resolution loss** is required?
- What **number of temperature changes** are required?

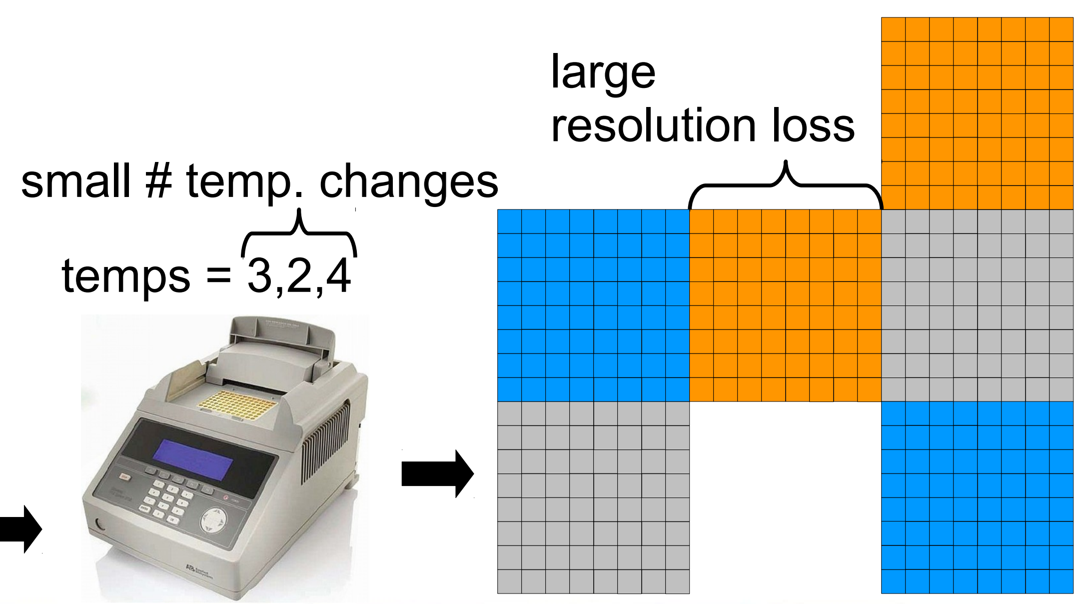
Complexity of Temperature Programming

For shape *S* with *n* points, trade-off between **resolution loss** and **number of temperature changes**:

- With *optimal resolution loss* = constant (22 in Scott's paper although shown smaller in the example), need ≈ *n* temperature changes.
- With *optimal number of temperature changes* = size of smallest program *p* that prints *S*, need resolution loss ≈ *t* = running time of *p*.







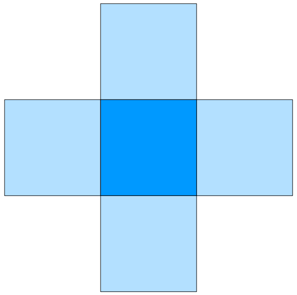
Hierarchical assembly

Parallelism in the Model

potential attachment location

attached tile

time step 0

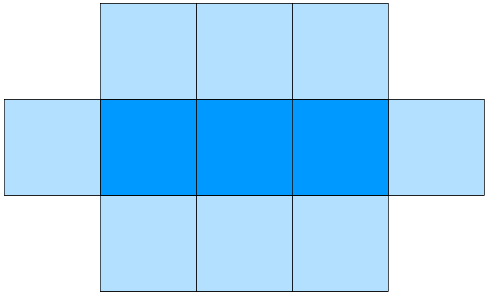


Parallelism in the Model

potential attachment location

attached tile

time step 1

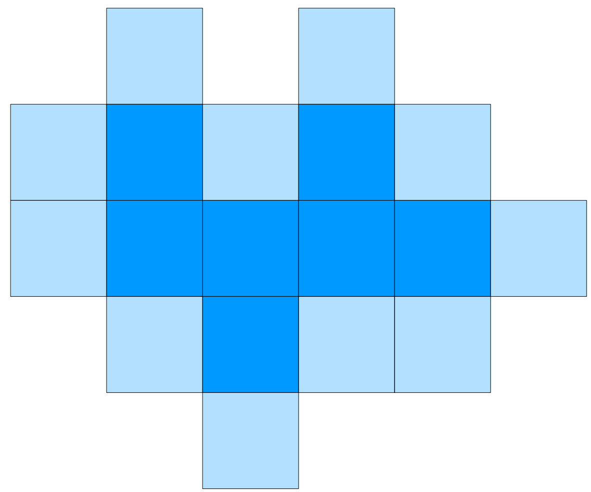


Parallelism in the Model

potential attachment location

attached tile

time step 2

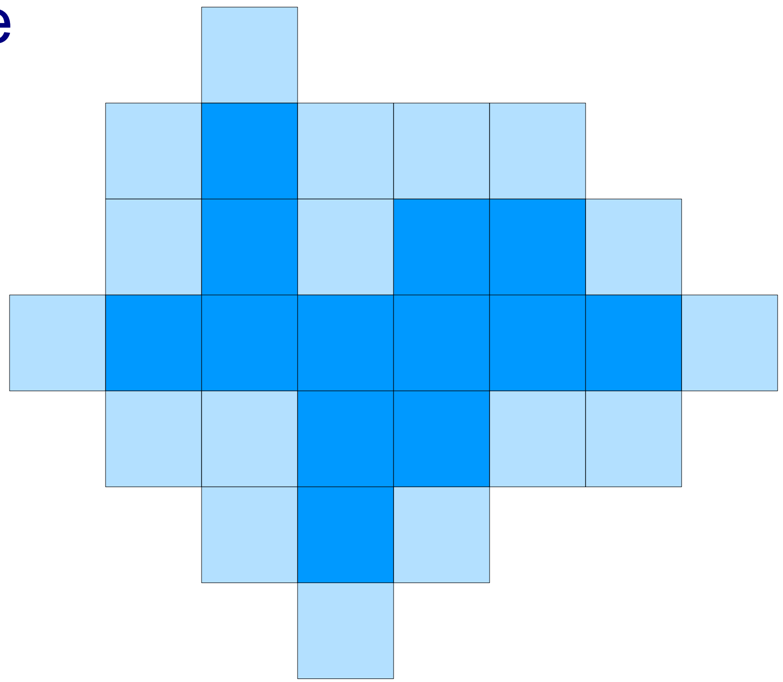


Parallelism in the Model

potential attachment location

attached tile

time step 3

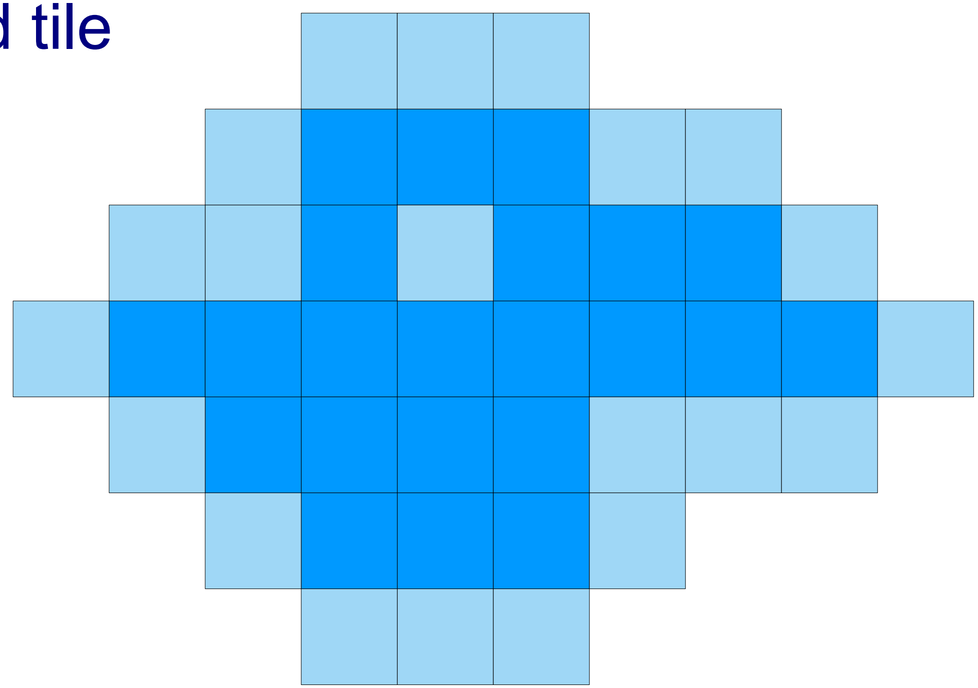


Parallelism in the Model

potential attachment location

attached tile

time step 4

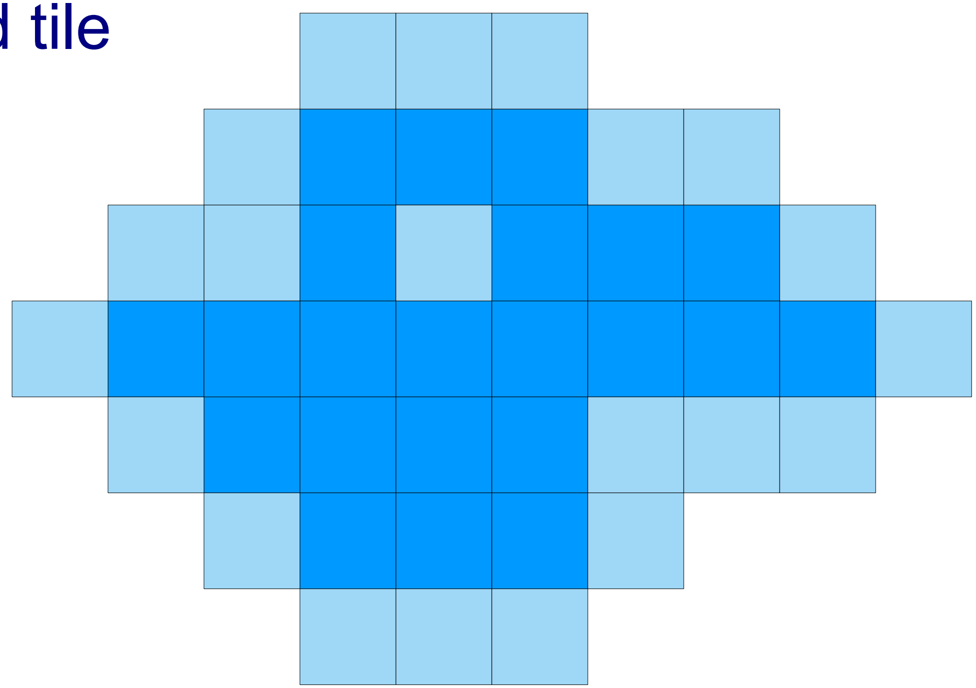


Parallelism in the Model

potential attachment location

attached tile

time step 4



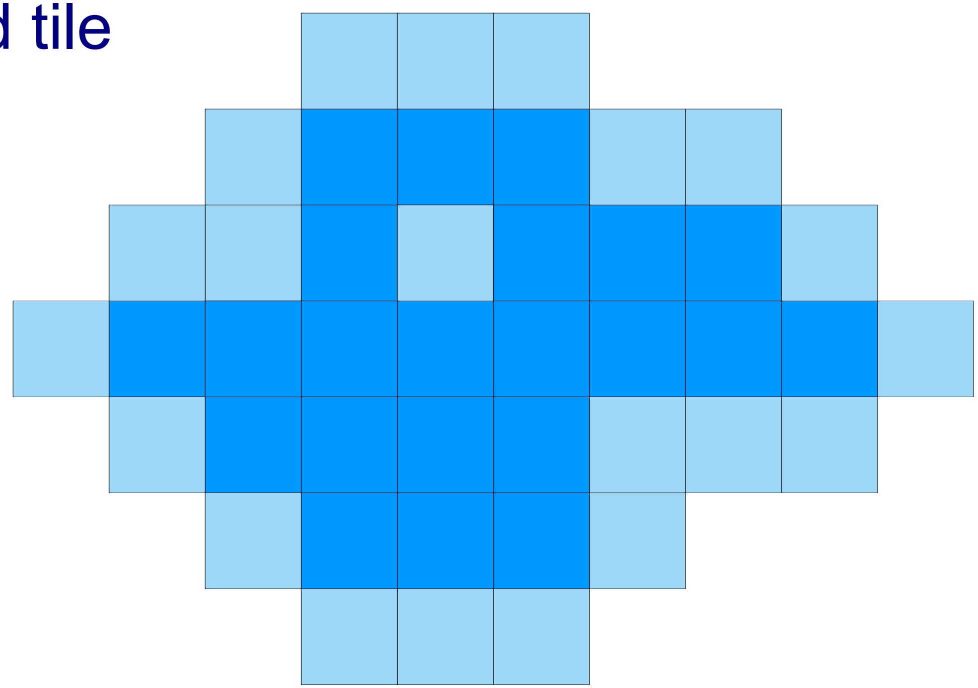
time t: perimeter ≤ $O(t)$ (with high probability)

Parallelism in the Model

potential attachment location

attached tile

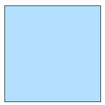
time step 4

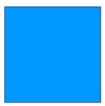


time t: perimeter $\leq O(t)$ (with high probability)

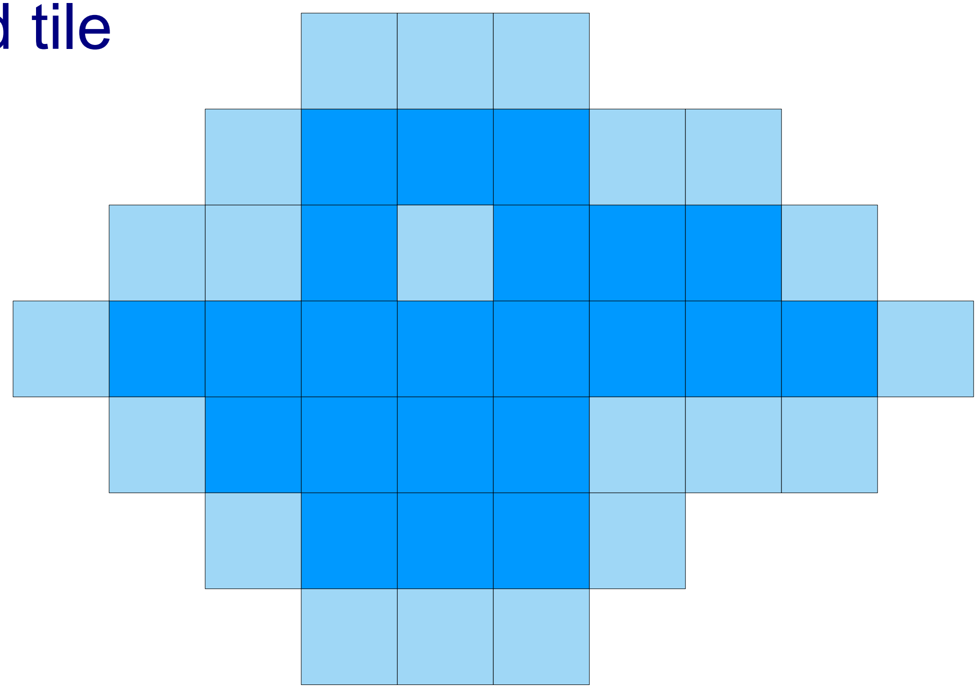
$\rightarrow$ max attachments per time step $\leq O(t)$

Parallelism in the Model

 potential attachment location

 attached tile

time step 4



time *t*: perimeter ≤ $O(t)$ (with high probability)

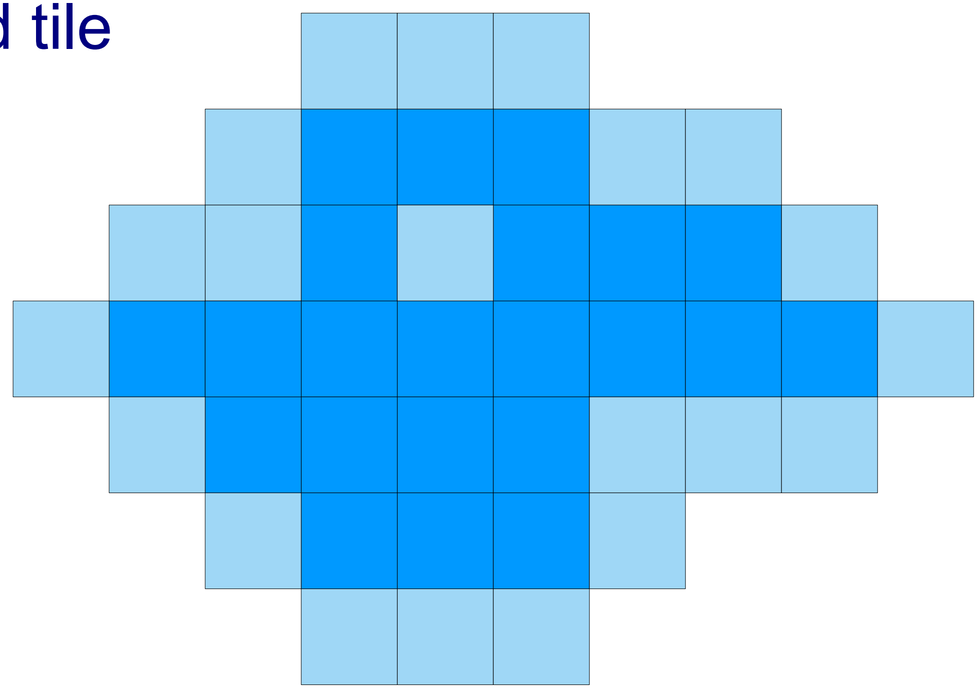
→ max attachments per time step ≤ $O(t)$

→ max total attachments after *t* steps ≤ $O(t^2)$

Parallelism in the Model

potential attachment location

attached tile



time step 4

time *t*: perimeter ≤ $O(t)$ (with high probability)

→ max attachments per time step ≤ $O(t)$

→ max total attachments after *t* steps ≤ $O(t^2)$

→ min time to assemble any shape of size $N \geq \Omega(\sqrt{N})$

Parallelism and Time

Can we speed up assembly by allowing large assemblies to form in parallel and then attach to each other in one step?

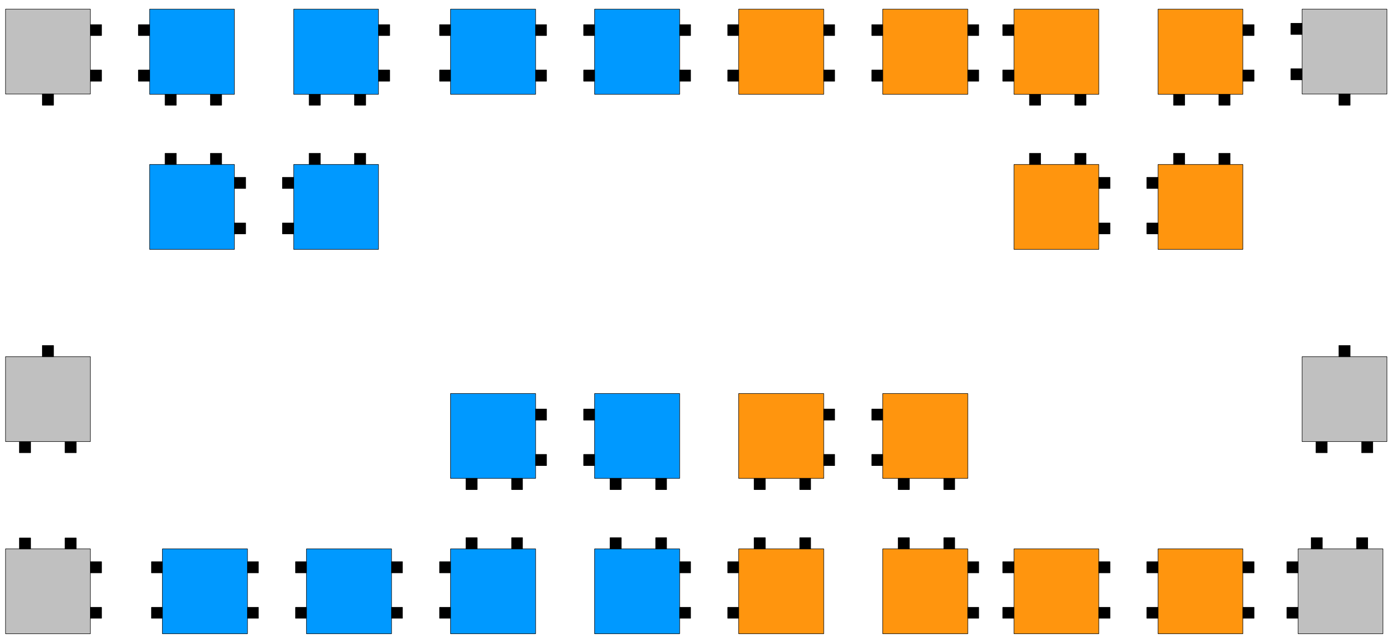
Hierarchical Tile Assembly Model

- **seeded** model
  - growth nucleates from a single seed tile
  - tiles attach one at a time

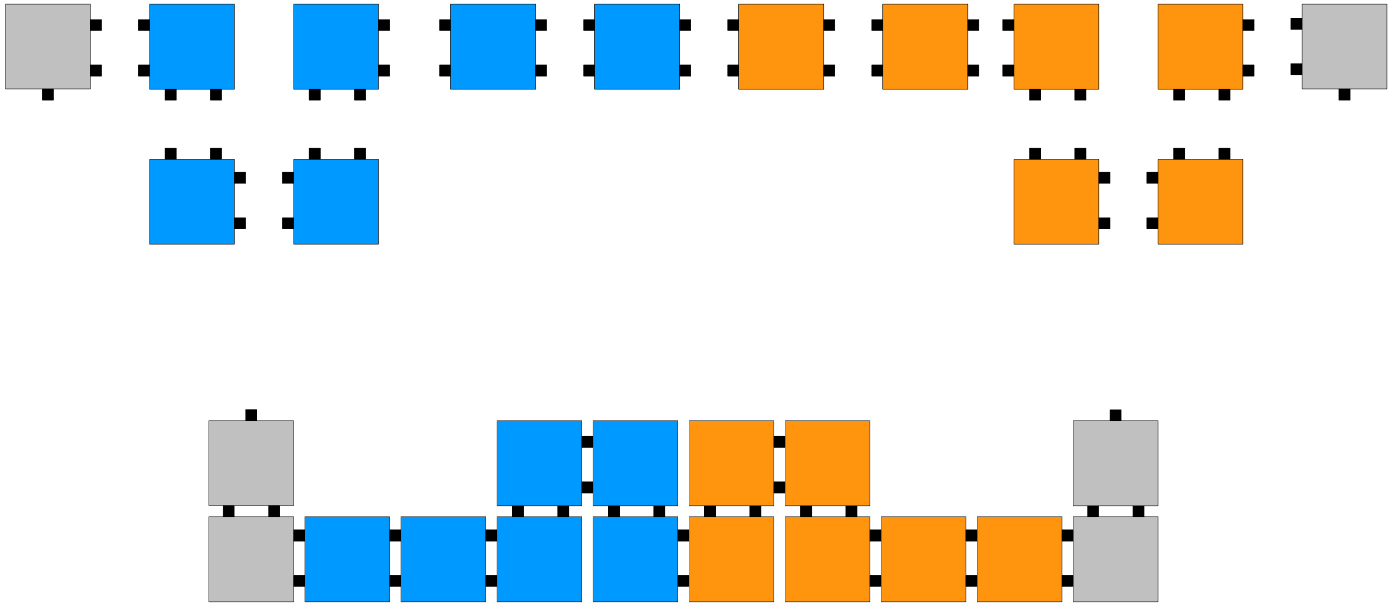
Hierarchical Tile Assembly Model

- **seeded** model
  - growth nucleates from a single seed tile
  - tiles attach one at a time
- **hierarchical** model: assembly is producible if
  - base case: it is a single tile, or
  - recursive case: it results from translating two producible assemblies so they *stably attach* without overlap

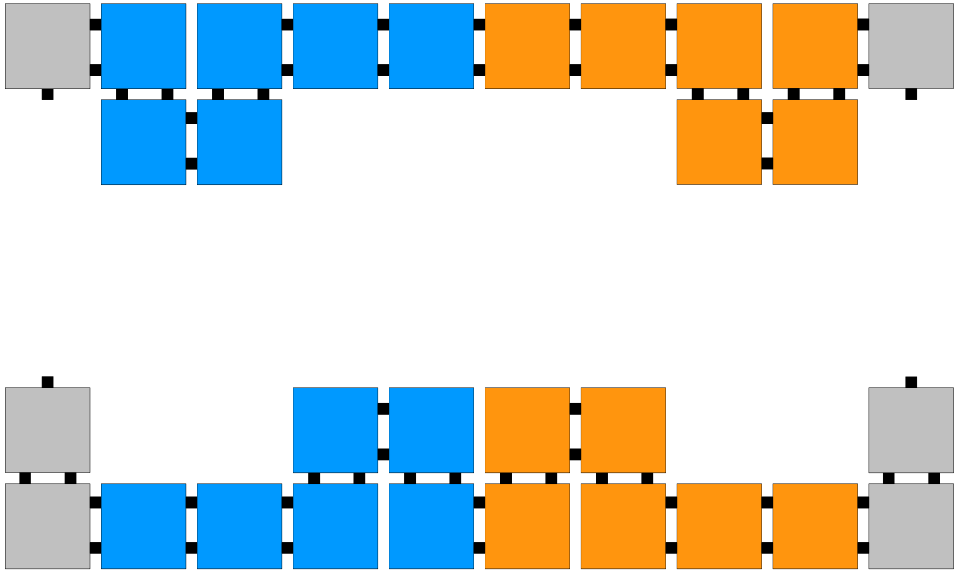
Hierarchical Tile Assembly Model



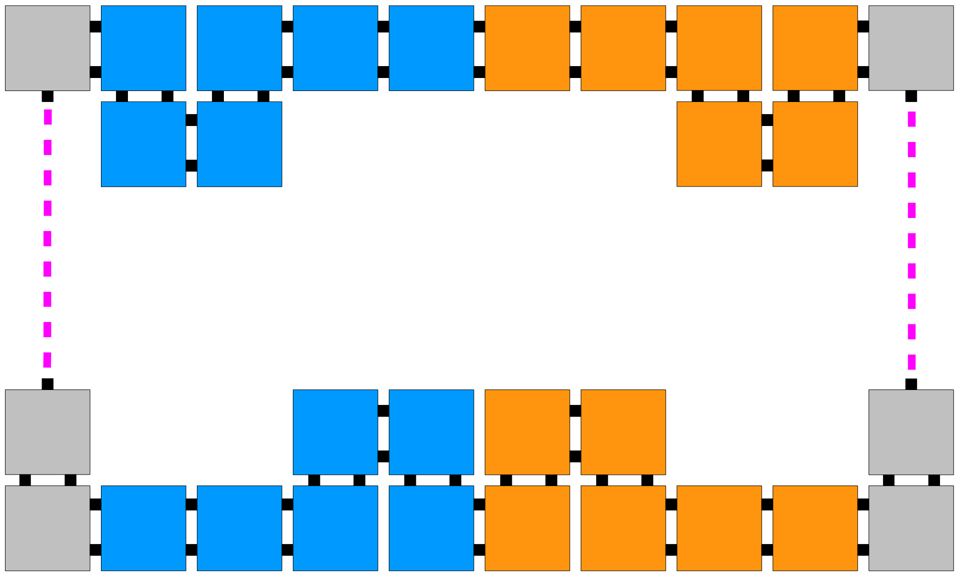
Hierarchical Tile Assembly Model



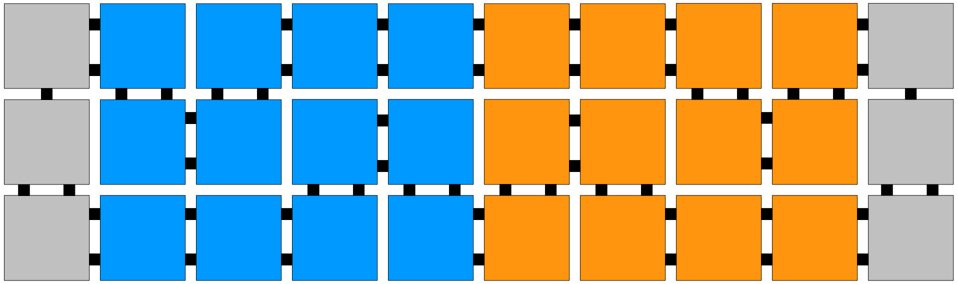
Hierarchical Tile Assembly Model



Hierarchical Tile Assembly Model

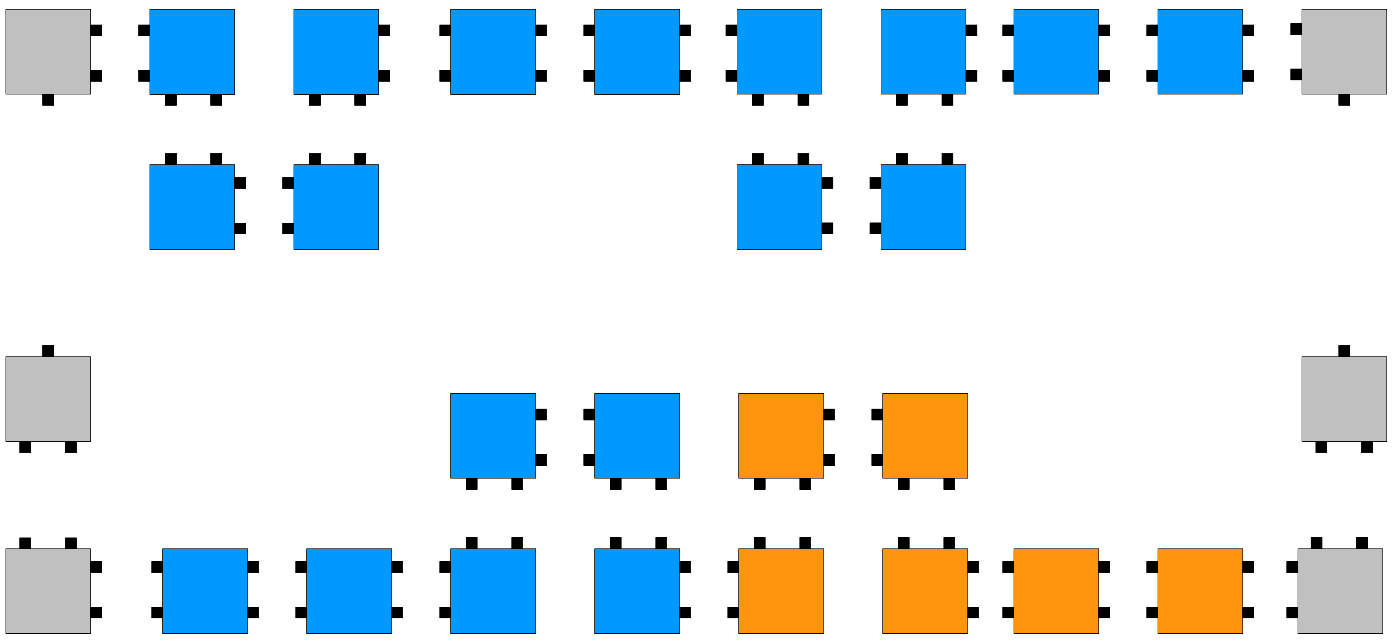


Hierarchical Tile Assembly Model



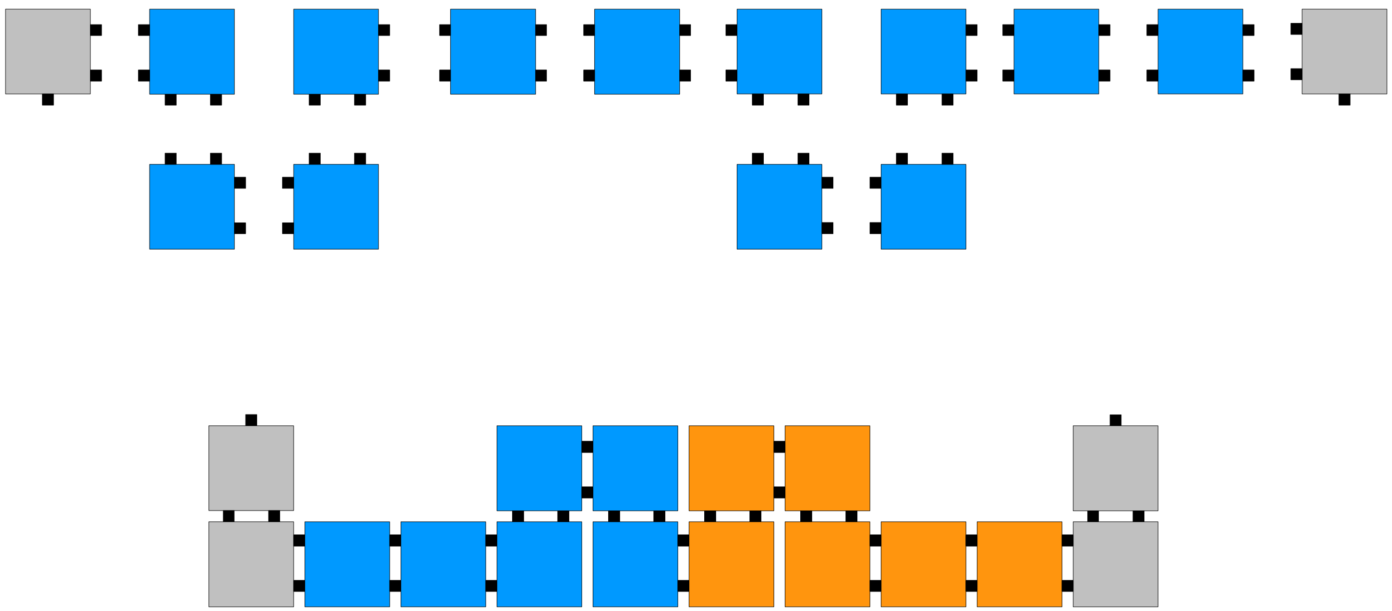
Hierarchical Tile Assembly Model

Overlap disallowed in attachment events ("steric protection")



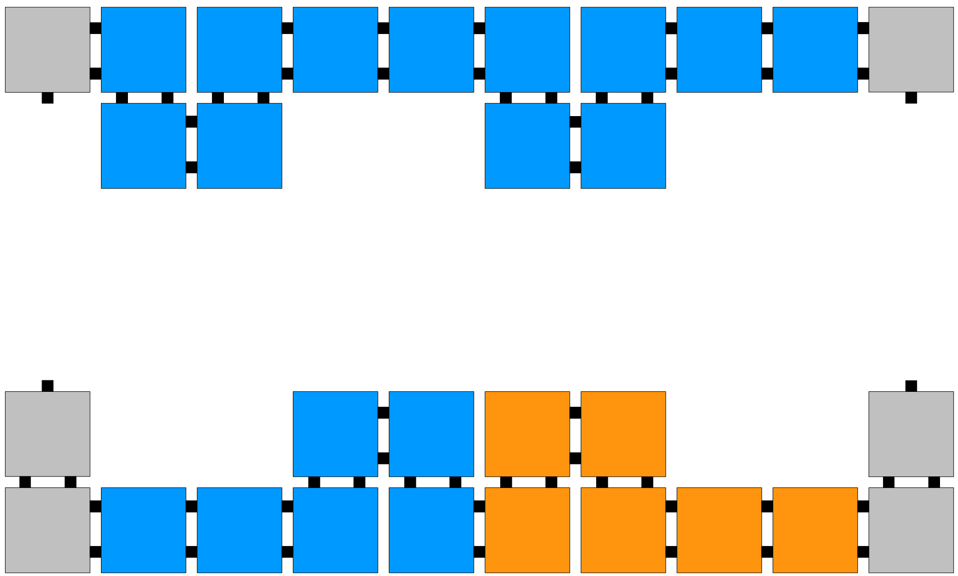
Hierarchical Tile Assembly Model

Overlap disallowed in attachment events ("steric protection")



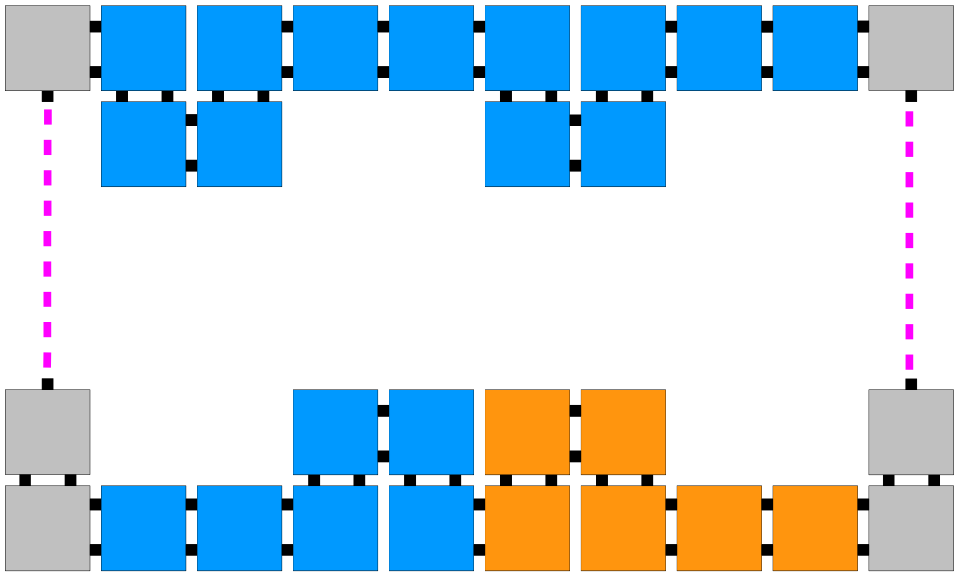
Hierarchical Tile Assembly Model

Overlap disallowed in attachment events ("steric protection")



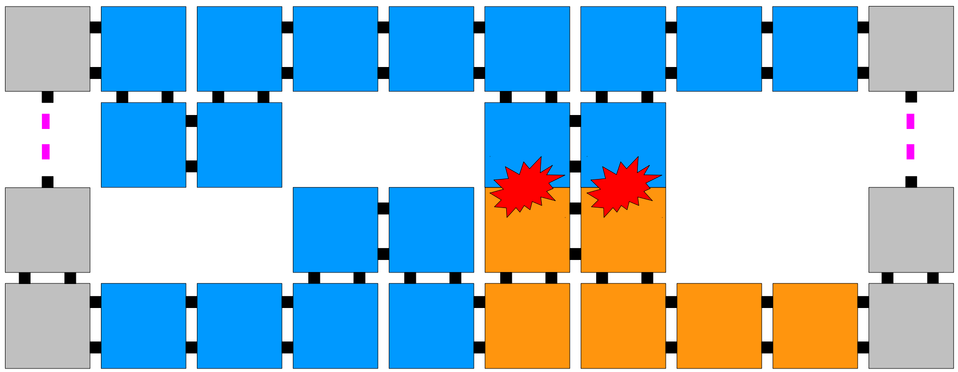
Hierarchical Tile Assembly Model

Overlap disallowed in attachment events (“steric protection”)



Hierarchical Tile Assembly Model

Overlap disallowed in attachment events ("steric protection")



Potentially Unrealistic Aspects of the Hierarchical Assembly Model

- Overlap restriction:
  - DNA is floppy; won't stay in the plane
  - Engineering problem; not fundamental

Potentially Unrealistic Aspects of the Hierarchical Assembly Model

- Overlap restriction:
  - DNA is floppy; won't stay in the plane
  - Engineering problem; not fundamental

More fundamental problems:

- Large assemblies assumed to diffuse as fast as individual tiles
- Uniform binding strength threshold; should be higher for larger assemblies

Potentially Unrealistic Aspects of the Hierarchical Assembly Model

- Overlap restriction:
  - DNA is floppy; won't stay in the plane
  - Engineering problem; not fundamental

More fundamental problems:

- Large assemblies assumed to diffuse as fast as individual tiles
- Uniform binding strength threshold; should be higher for larger assemblies

} artificially boost assembly speed

Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log \log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)

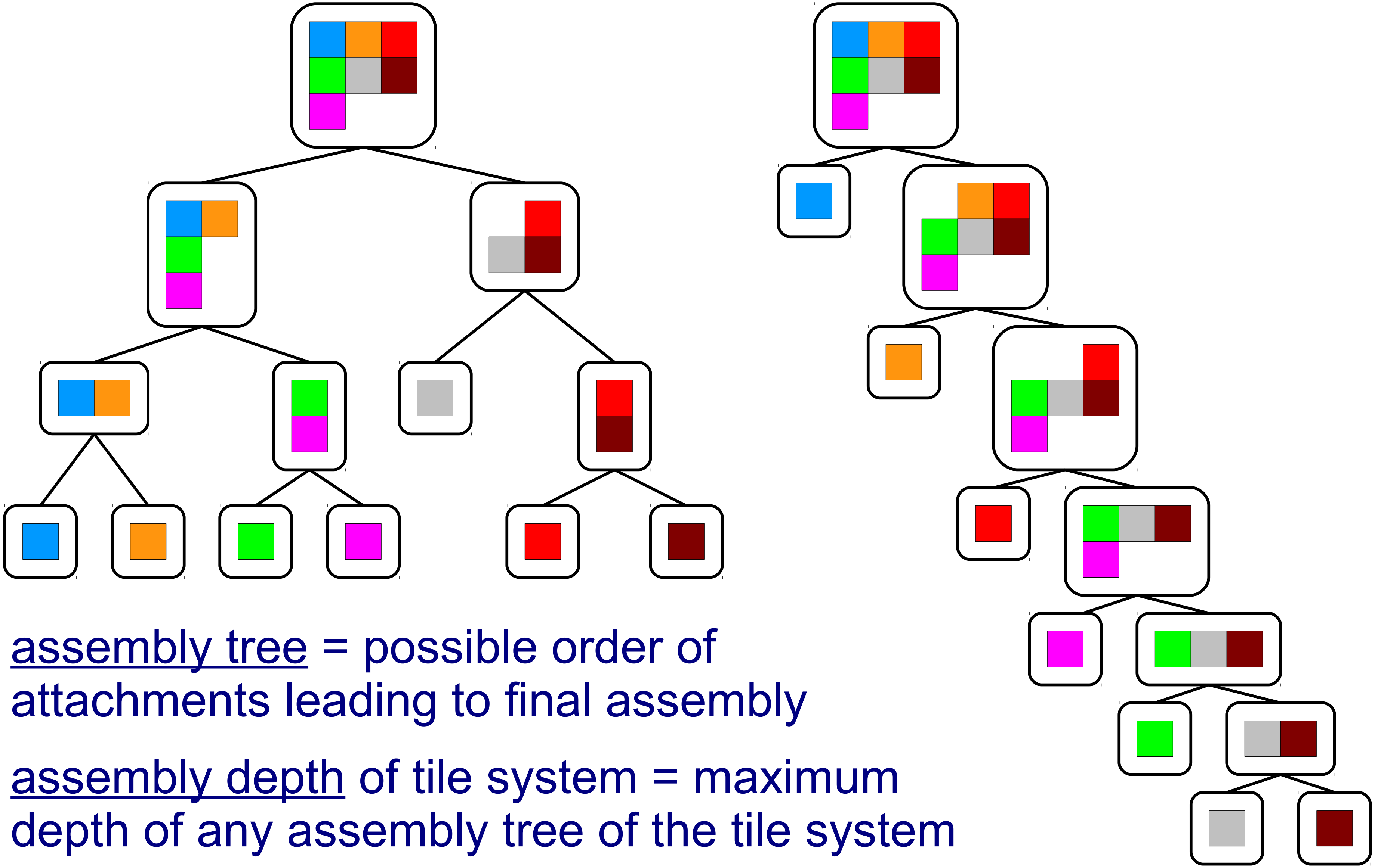
Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log\log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)
- They asked: Can the extra parallelism in the hierarchical model break the $\Omega(n)$ lower bound?

Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log \log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)
- They asked: Can the extra parallelism in the hierarchical model break the $\Omega(n)$ lower bound?
- <u>We show</u>:
  - $O(\log n / \log \log n)$ tile types can assemble an n x n square using "nearly maximal" parallelism.

Definition of Hierarchical Parallelism



<u>assembly tree</u> = possible order of attachments leading to final assembly

<u>assembly depth</u> of tile system = maximum depth of any assembly tree of the tile system

Highly Parallel Square Assembly

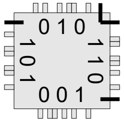
- Best possible assembly depth for any shape with *N* points is log *N*.

Highly Parallel Square Assembly

- Best possible assembly depth for any shape with N points is $\log N$.
- **Theorem:** For every positive integer n, there is a tile system with $O(\log n / \log \log n)$ tile types and assembly depth $O(\log^2 n)$ that assembles an n x n square.

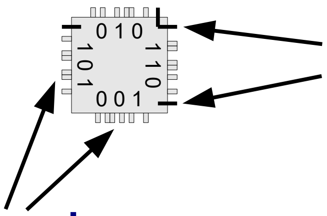
Idea: Blocks of size $O(\log n) \times O(\log n)$, assembled "nonparallelly", randomly guess their (x,y) position in square and bind only to carefully selected neighboring blocks.

$O(\log n)$ x $O(\log n)$
block of tiles



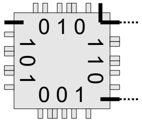
Idea: Blocks of size *O*(log *n*) x *O*(log *n*), assembled “nonparallelly”, randomly guess their (*x*,*y*) position in square and bind only to carefully selected neighboring blocks.

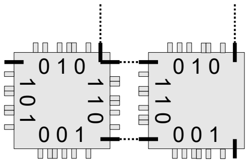
O(log *n*) x *O*(log *n*)
block of tiles

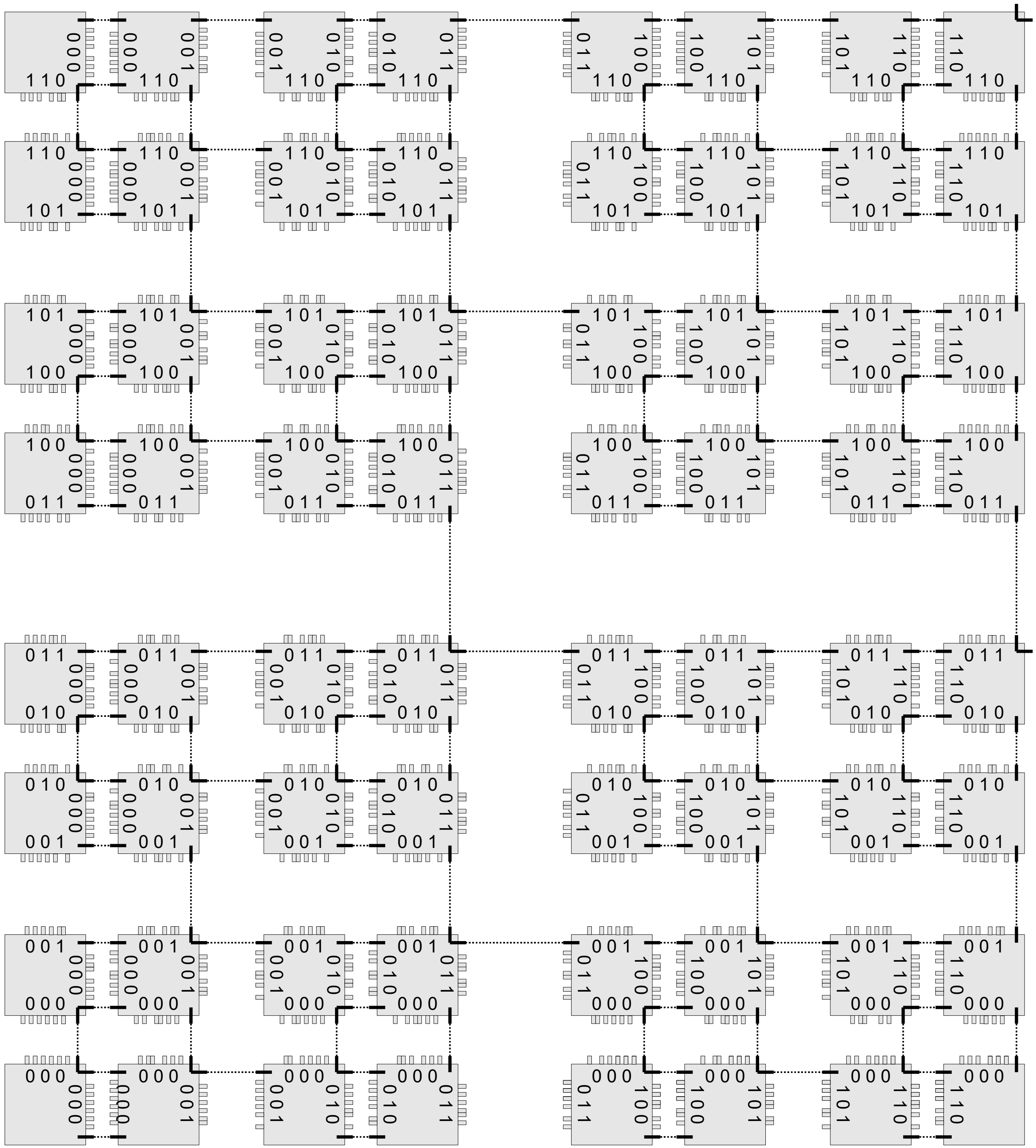


strength 1
glues

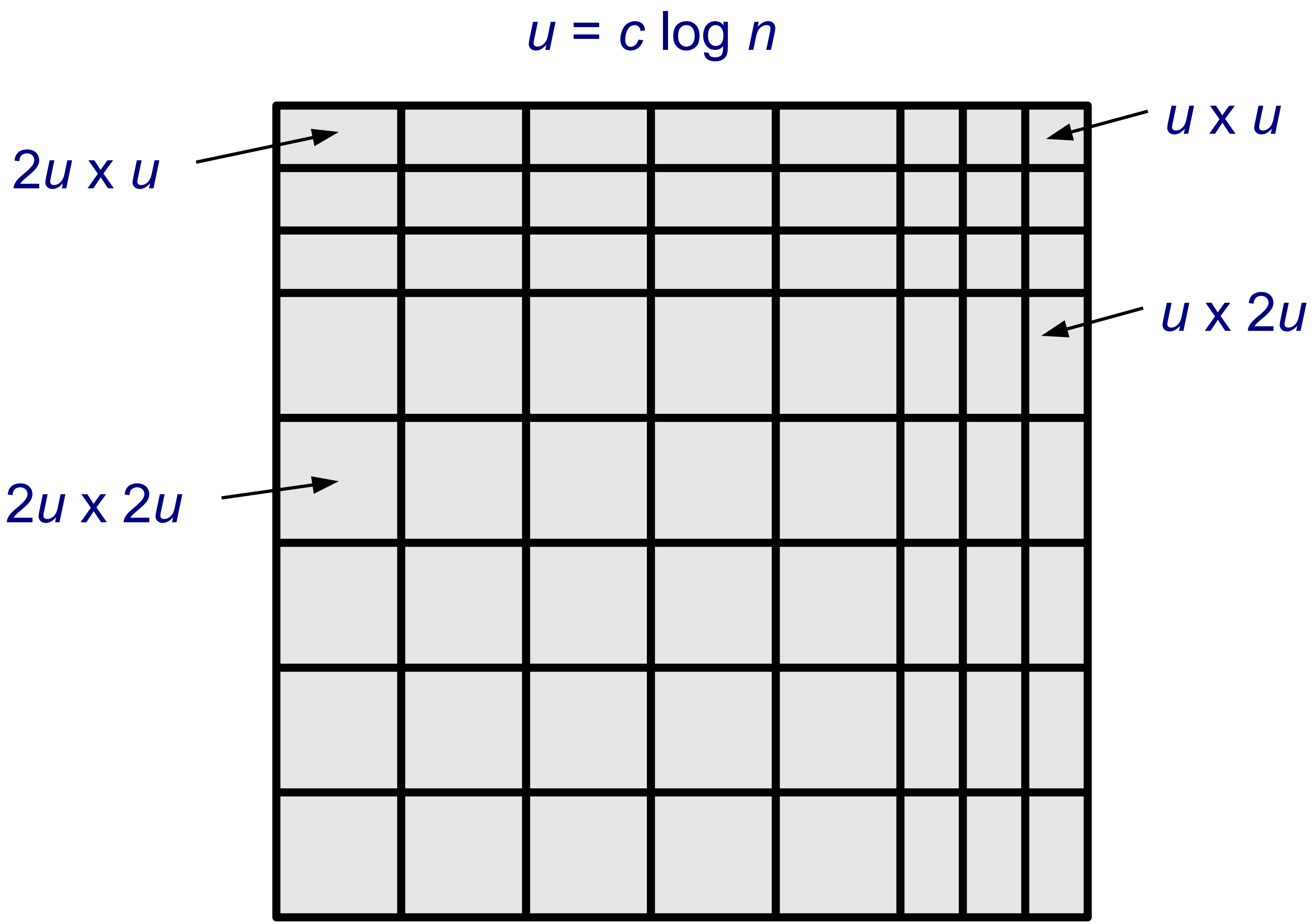
bumps encode
(*x*,*y*) address




010
101
110
001
010
110
001

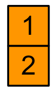


Handling Non-Powers-of-2

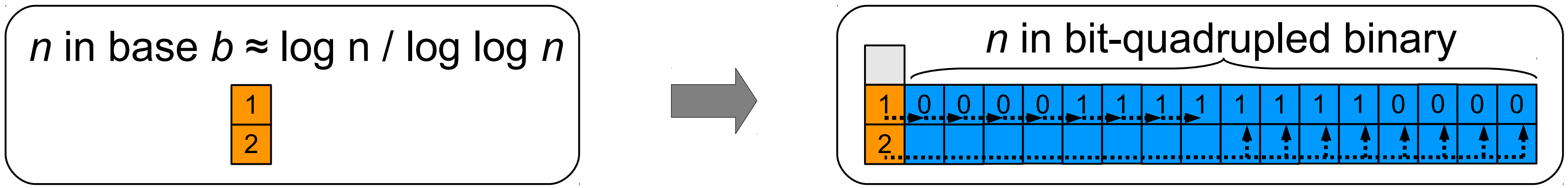


Assembly of Each Block

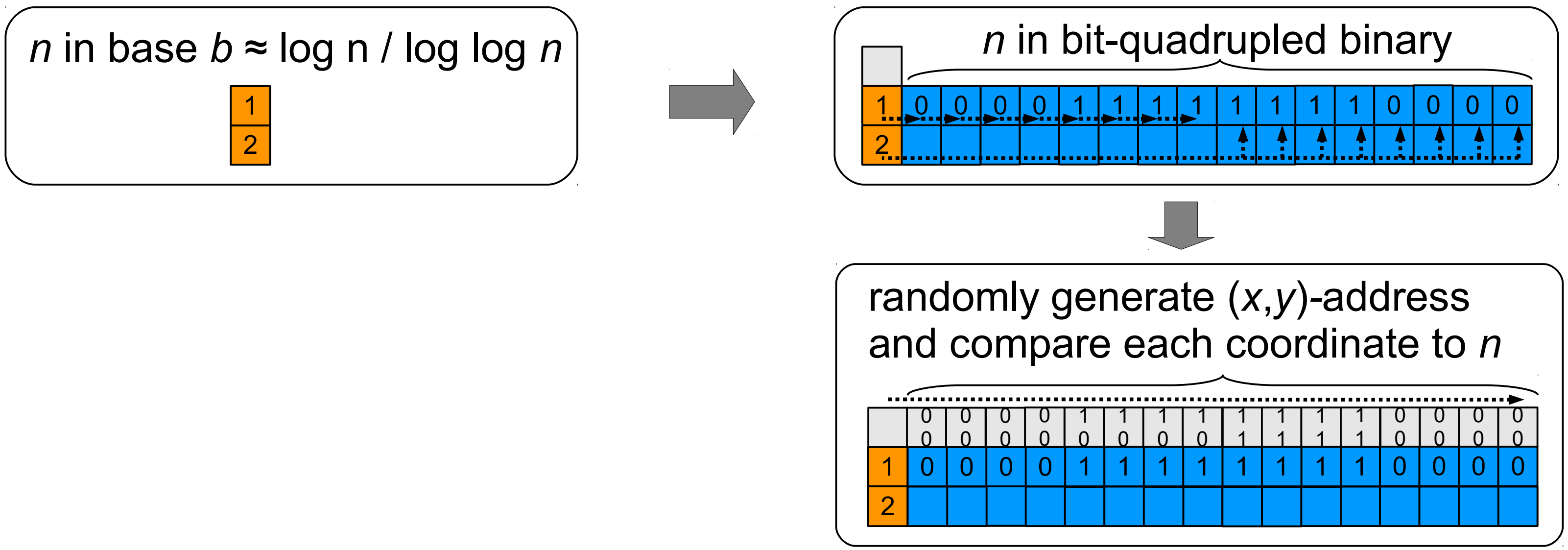
n in base *b* ≈ log n / log log *n*



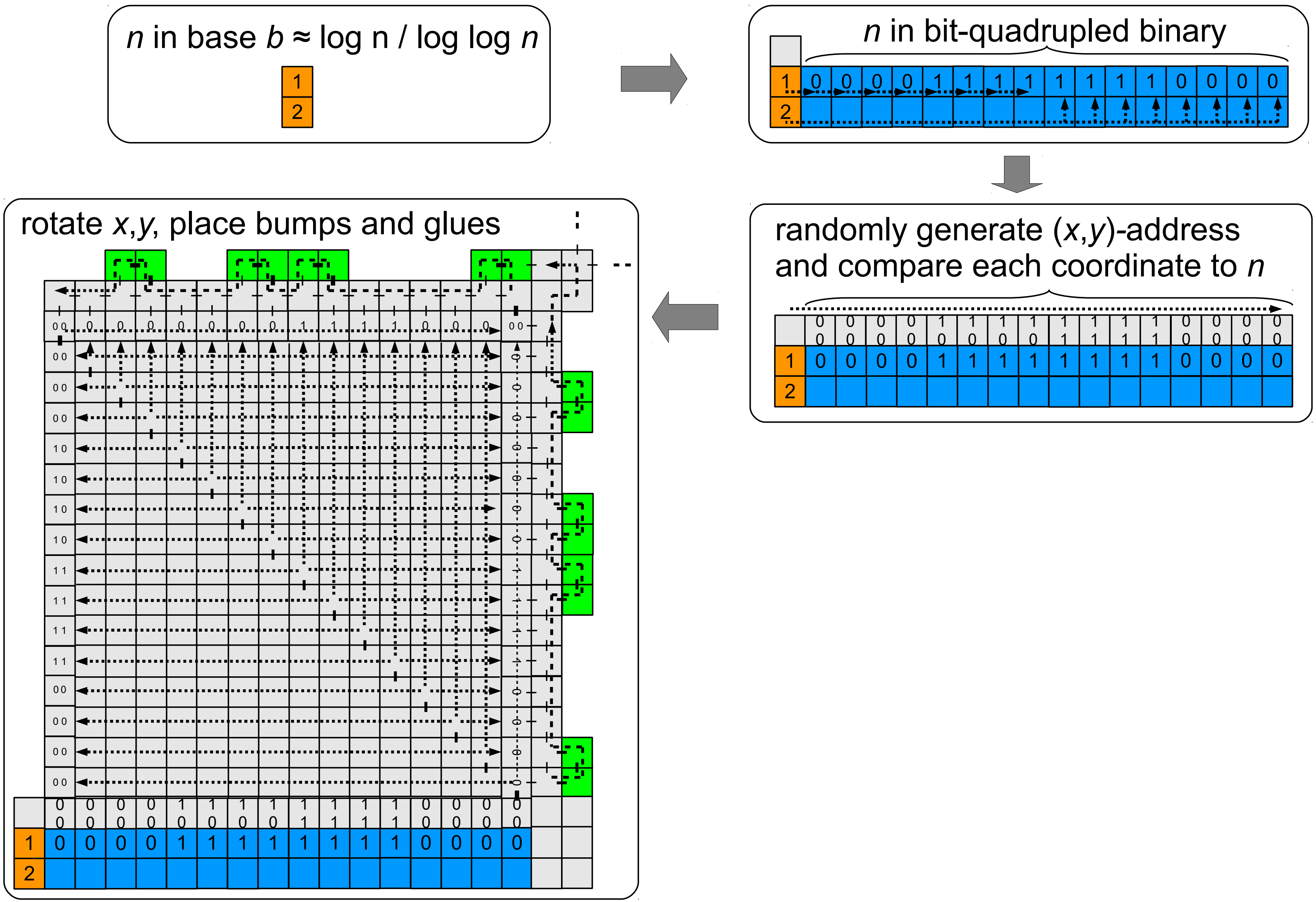
Assembly of Each Block



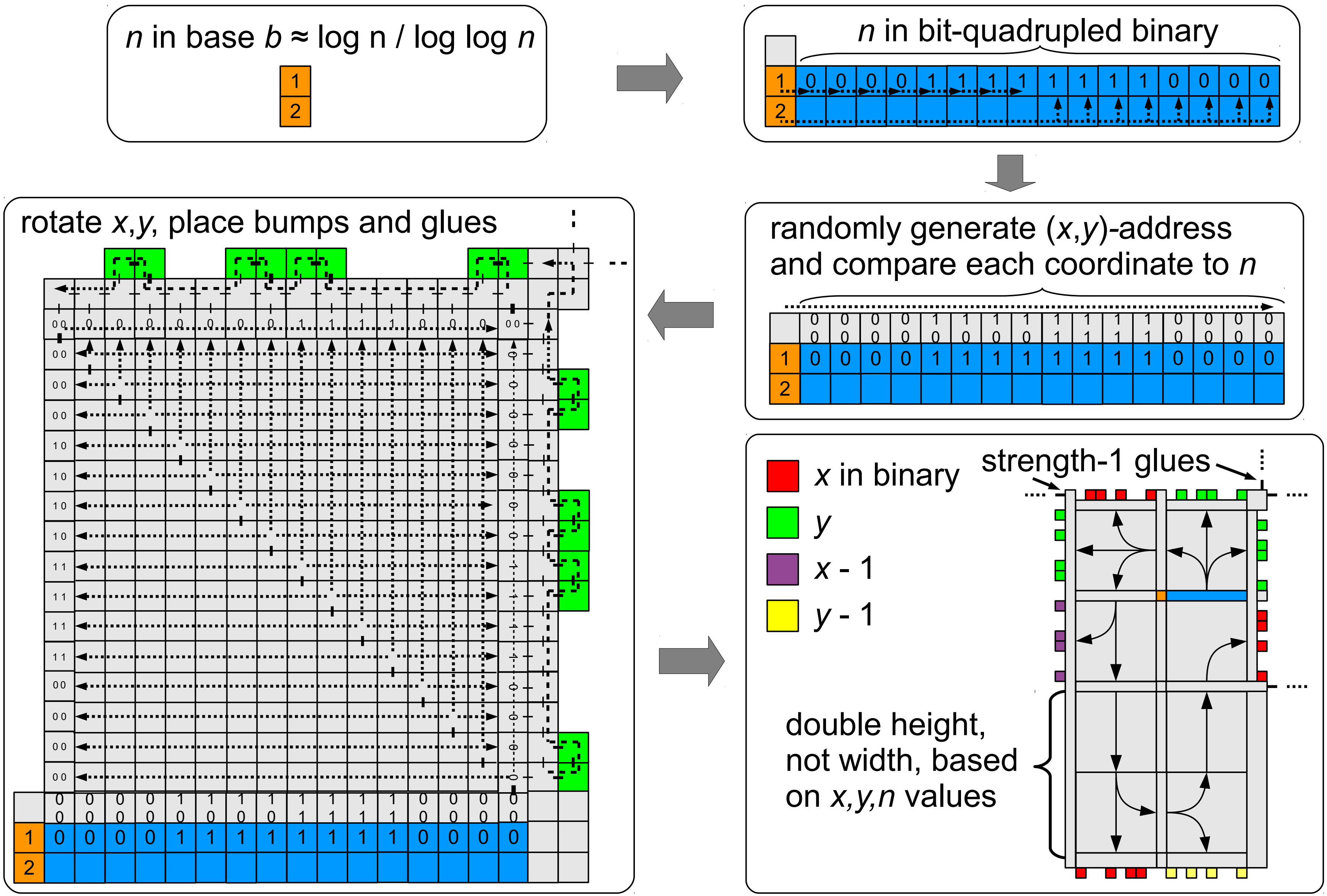
Assembly of Each Block



Assembly of Each Block



Assembly of Each Block



Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log \log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)
- They asked: Can the extra parallelism in the hierarchical model break the $\Omega(n)$ lower bound?
- We show:
  - $O(\log n / \log \log n)$ tile types can assemble an n x n square using "nearly maximal" parallelism.

Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log \log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)
- They asked: Can the extra parallelism in the hierarchical model break the $\Omega(n)$ lower bound?
- <u>We show</u>:
  - $O(\log n / \log \log n)$ tile types can assemble an n x n square using "nearly maximal" parallelism.
  - This construction takes **superlinear** time.

Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log \log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)
- They asked: Can the extra parallelism in the hierarchical model break the $\Omega(n)$ lower bound?
- We show:
  - $O(\log n / \log \log n)$ tile types can assemble an n x n square using "nearly maximal" parallelism.
  - This construction takes **superlinear** time.
  - **Every** "partial order system" requires time $\Omega(N)$ to assemble any shape of diameter N.

Our Results

- **Previous result:** Assembling an n x n square requires $\Omega(n)$ steps in the **seeded** model; achievable with optimal $O(\log n / \log \log n)$ tile types (Adleman, Cheng, Goel, Huang, *STOC* 2001)
- They asked: Can the extra parallelism in the hierarchical model break the $\Omega(n)$ lower bound?
- We show:
  - $O(\log n / \log \log n)$ tile types can assemble an n x n square using "nearly maximal" parallelism.
  - This construction takes **superlinear** time.
  - **Every** "partial order system" requires time $\Omega(N)$ to assemble any shape of diameter N.

The extra parallelism of the hierarchical model is **useless** for speeding up partial order systems.

Assembly Time Complexity Model

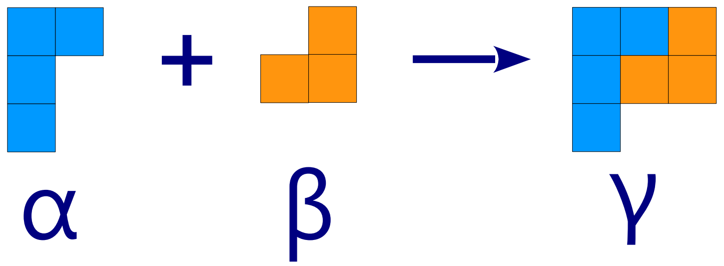
- Assign each tile type s an initial concentration $C(s)$ so that $\sum_s C(s) = 1$ (finite density constraint).

Assembly Time Complexity Model

- Assign each tile type s an initial concentration $C(s)$ so that $\sum_s C(s) = 1$ (finite density constraint).
- At time $t = 0$, each assembly α with only a single tile s has initial concentration $[\alpha](t) = C(s)$. All larger assemblies α have $[\alpha](t) = 0$ at time $t = 0$.

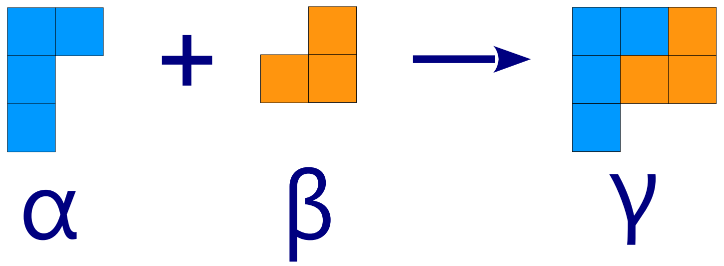
Assembly Time Complexity Model

- Assign each tile type s an initial concentration $C(s)$ so that $\sum_s C(s) = 1$ (finite density constraint).
- At time $t = 0$, each assembly α with only a single tile s has initial concentration $[\alpha](t) = C(s)$. All larger assemblies α have $[\alpha](t) = 0$ at time $t = 0$.
- Each attachment $\alpha + \beta \rightarrow \gamma$ is a chemical reaction with rate $[\alpha](t)\cdot[\beta](t)$ at time t.



Assembly Time Complexity Model

- Assign each tile type s an initial concentration $C(s)$ so that $\sum_s C(s) = 1$ (finite density constraint).
- At time $t = 0$, each assembly α with only a single tile s has initial concentration $[\alpha](t) = C(s)$. All larger assemblies α have $[\alpha](t) = 0$ at time $t = 0$.
- Each attachment $\alpha + \beta \to \gamma$ is a chemical reaction with rate $[\alpha](t)\cdot[\beta](t)$ at time t.

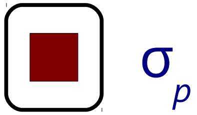


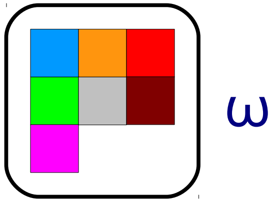
- Concentrations evolve by mass-action kinetics:

$$d[\alpha] / dt \;=\; \sum_{\gamma + \beta \to \alpha}[\gamma](t)\cdot[\beta](t) \;-\; \sum_{\alpha + \beta \to \gamma}[\alpha](t)\cdot[\beta](t)$$

Assembly Time Complexity Model

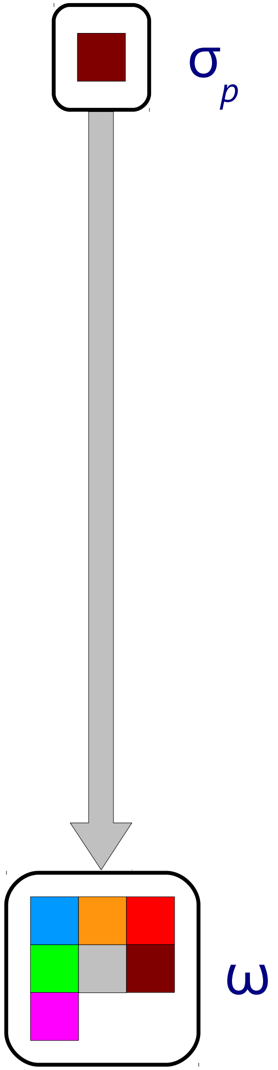
- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*





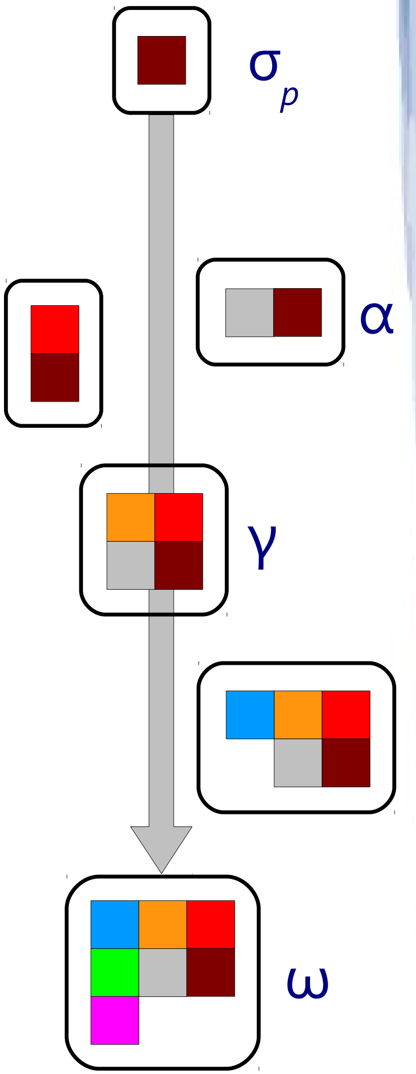
Assembly Time Complexity Model

- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*
- σ_p changes into ω by a continuous-time Markov chain



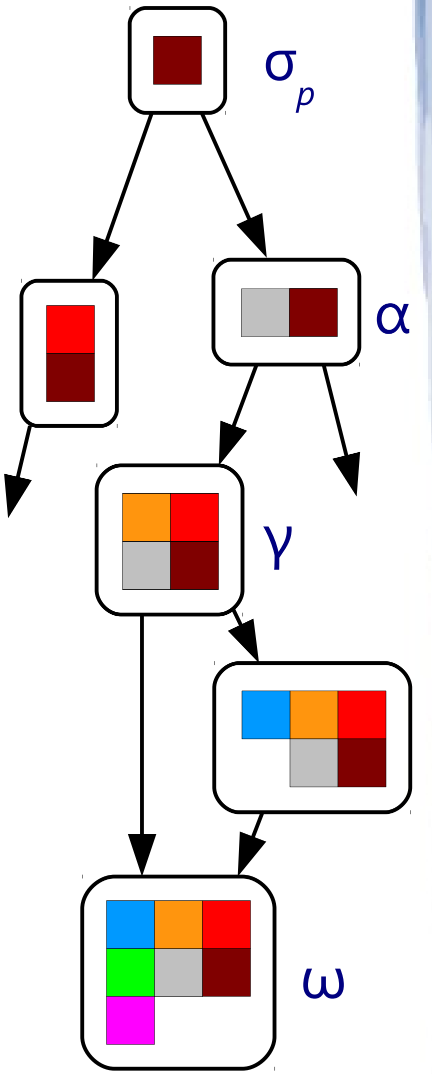
Assembly Time Complexity Model

- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*
- σ_p changes into ω by a continuous-time Markov chain
- States = assemblies σ_p, ω, and all possible intermediates



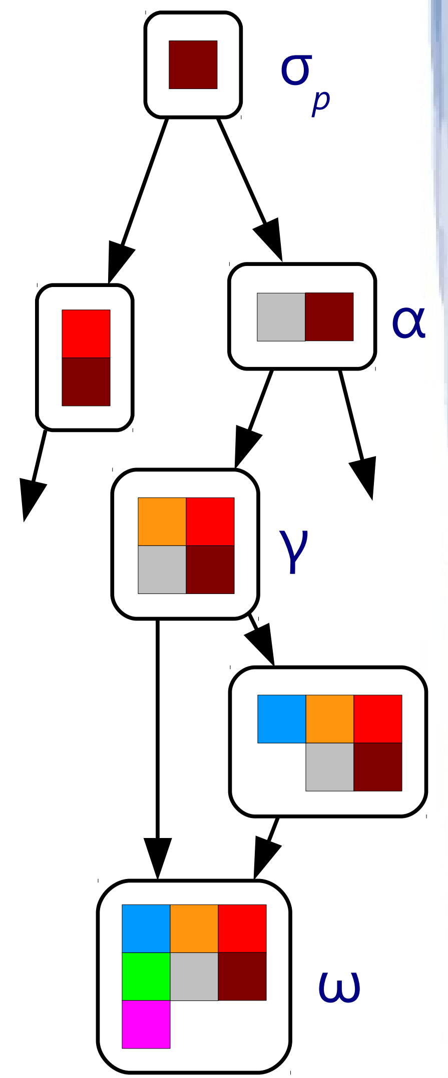
Assembly Time Complexity Model

- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*
- σ_p changes into ω by a continuous-time Markov chain
- States = assemblies σ_p, ω, and all possible intermediates
- Transition from α to γ if there is a producible assembly β such that $\alpha + \beta \rightarrow \gamma$, with time-dependent rate $[\beta](t)$



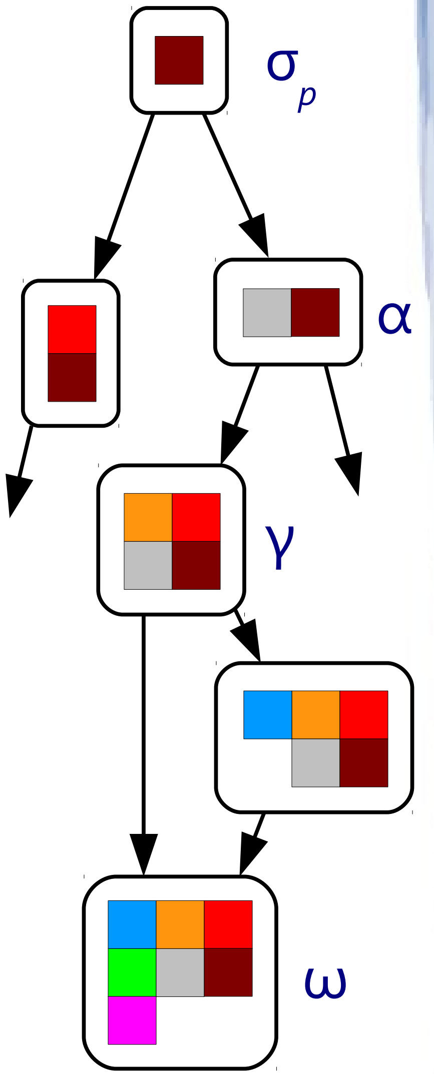
Assembly Time Complexity Model

- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*
- σ_p changes into ω by a continuous-time Markov chain
- States = assemblies σ_p, ω, and all possible intermediates
- Transition from α to γ if there is a producible assembly β such that $\alpha + \beta \rightarrow \gamma$, with time-dependent rate $[\beta](t)$
- Unique sink state of the Markov chain is ω



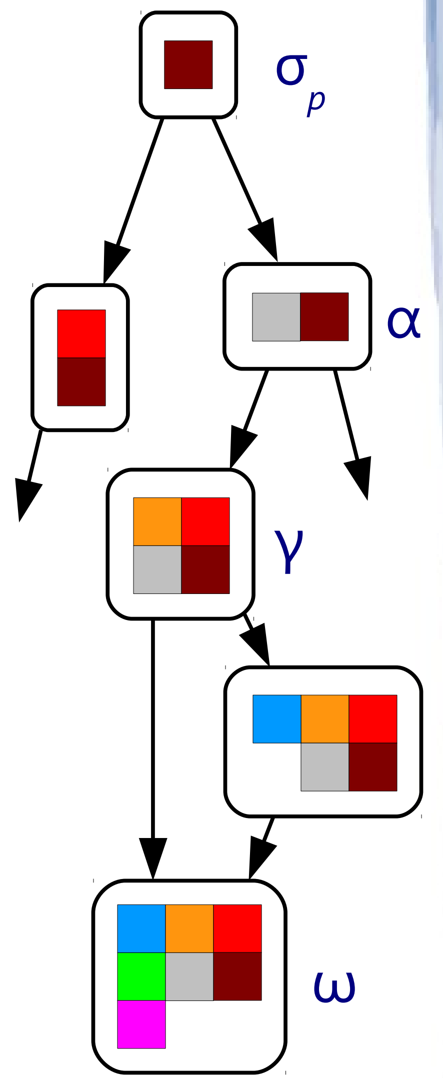
Assembly Time Complexity Model

- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*
- σ_p changes into ω by a continuous-time Markov chain
- States = assemblies σ_p, ω, and all possible intermediates
- Transition from α to γ if there is a producible assembly β such that α + β → γ, with time-dependent rate [β](*t*)
- Unique sink state of the Markov chain is ω
- time relative to *p* = expected time to reach ω from σ_p



Assembly Time Complexity Model

- Fix a position *p* in the unique final assembly ω, with *initial assembly* σ_p with just the tile at position *p*
- σ_p changes into ω by a continuous-time Markov chain
- States = assemblies σ_p, ω, and all possible intermediates
- Transition from α to γ if there is a producible assembly β such that α + β → γ, with time-dependent rate [β](*t*)
- Unique sink state of the Markov chain is ω
- time relative to *p* = expected time to reach ω from σ_p
- time = $\max_p$ time relative to *p*



Assembly Time Lower Bound

- partial order system: in the terminal assembly, each pair of adjacent binding tiles have an *assembly order precedence relationship* (one always binds first, or at the same time)

Assembly Time Lower Bound

- partial order system: in the terminal assembly, each pair of adjacent binding tiles have an *assembly order precedence relationship* (one always binds first, or at the same time)
- **Theorem**: Any partial order system whose terminal assembly has diameter N requires time $\Omega(N)$.

Main Proof Idea

conservation of mass: assemblies of size n and k attach to create assembly of size $n + k$

Main Proof Idea

conservation of mass: assemblies of size n and k attach to create assembly of size $n + k$

→ $(\forall t \geq 0) \quad \sum_{\alpha} [\alpha](t) \cdot |\alpha| = 1$

Main Proof Idea

conservation of mass: assemblies of size n and k attach to create assembly of size $n + k$

→ $(\forall t \geq 0)\quad \sum_\alpha [\alpha](t) \cdot |\alpha| = 1$

→ assembly of size k has concentration $\leq 1/k$

Main Proof Idea

conservation of mass: assemblies of size n and k attach to create assembly of size $n + k$

→ ($t \geq 0$) $\sum_{\alpha} [\alpha](t) \cdot |\alpha| = 1$

→ assembly of size k has concentration $\leq 1/k$

→ growing by size k in a single step takes expected time $\geq k$

Main Proof Idea

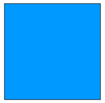
conservation of mass: assemblies of size n and k attach to create assembly of size $n + k$

$\rightarrow$ ($t \geq 0$) $\sum_{\alpha} [\alpha](t) \cdot |\alpha| = 1$

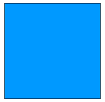
$\rightarrow$ assembly of size k has concentration $\leq 1/k$

$\rightarrow$ growing by size k in a single step takes expected time $\geq k$

seeded



hierarchical



Main Proof Idea

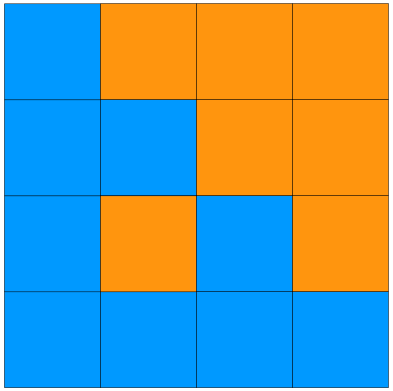
conservation of mass: assemblies of size *n* and *k* attach to create assembly of size *n* + *k*

→ ($t \geq 0$) $\sum_\alpha [\alpha](t) \cdot |\alpha| = 1$

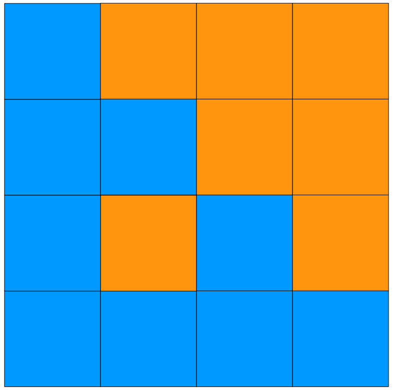
→ assembly of size *k* has concentration ≤ 1/*k*

→ growing by size *k* in a single step takes expected time ≥ *k*

seeded



hierarchical

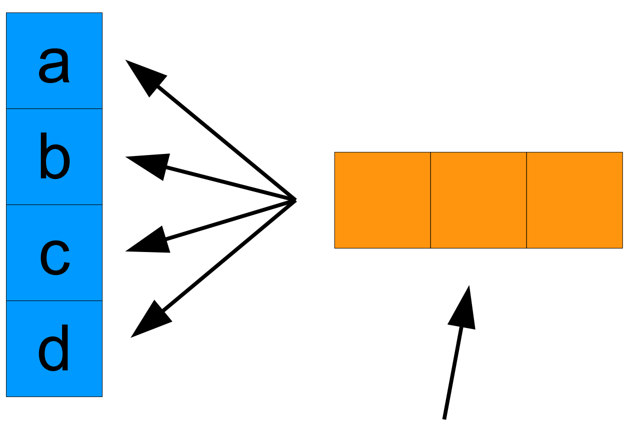


Why Partial Order Systems?

Argument breaks if a single assembly of size *k* could attach to many positions, any one of which suffices to proceed to terminal assembly.

Why Partial Order Systems?

Argument breaks if a single assembly of size *k* could attach to many positions, any one of which suffices to proceed to terminal assembly.



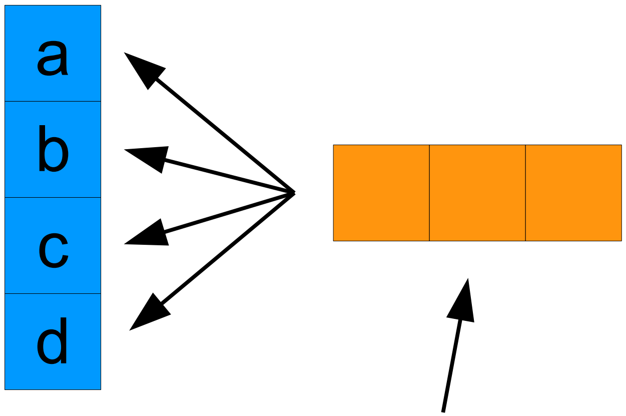
concentration ≤ 1/3

E[time to attach to a] ≥ 3

E[time to attach to any of a,b,c,d] ≥ 3 / 4

Why Partial Order Systems?

Argument breaks if a single assembly of size *k* could attach to many positions, any one of which suffices to proceed to terminal assembly.

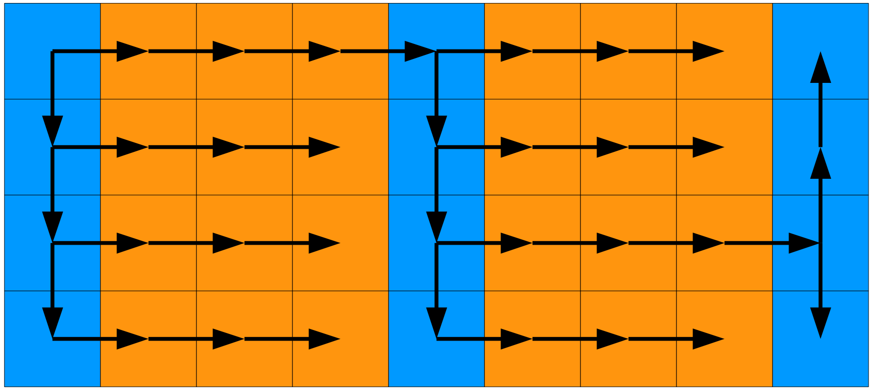


concentration ≤ 1/3

E[time to attach to a] ≥ 3

E[time to attach to any of a,b,c,d] ≥ 3 / 4

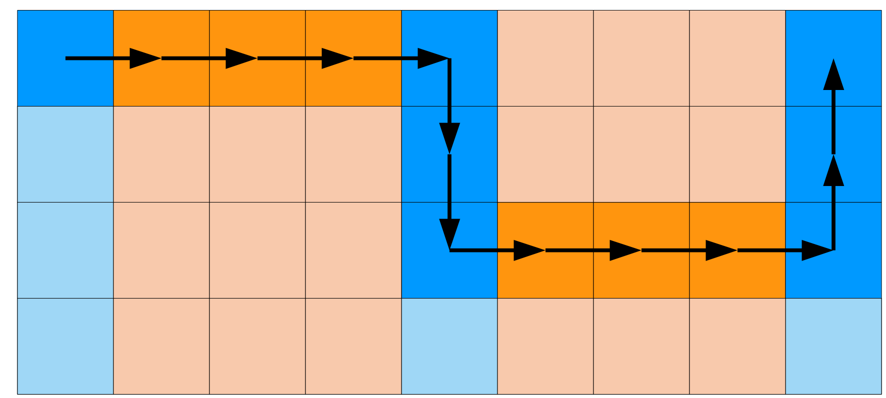
Any path in partial order DAG must assemble in order

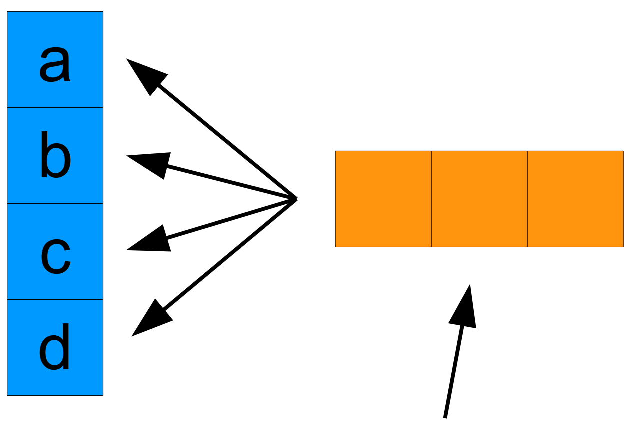


Why Partial Order Systems?

Argument breaks if a single assembly of size *k* could attach to many positions, any one of which suffices to proceed to terminal assembly.

Any path in partial order DAG must assemble in order





longest path has length ≥ diameter of shape

by concentration argument, path takes time *k* to grow by *k* tiles

concentration ≤ 1/3

E[time to attach to a] ≥ 3

E[time to attach to any of a,b,c,d] ≥ 3 / 4

Removing tiles (RNase model)

Removing Tiles

- aTAM is *monotone*: stably attached tiles do not detach
  - "Computation of a shape" with tiles may take a lot of space
  - Need large resolution loss to compute **within** the shape
  - kinetic model allows detachment but not controllable
- RNase model (Abel, Benbernou, Damian, Demaine, Demaine, Flatland, Kominers, Schweller)
  - make some tile types from RNA and some from DNA
  - after some time, add RNase enzyme to dissolve RNA tiles
  - only subassemblies made of DNA tiles remain

Shape-Building with Small Resolution Loss and Optimal Tile Complexity

Demaine, Patitz, Schweller, Summers (STACS 2011)*:*

given: finite shape *S*, |*S*|=*n*

there is a TAS T, |T| ≈ K(*S*), that assembles S at scale factor ≈ log *n*, with one step of dissolving RNA tiles

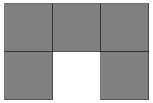
Shape-Building with Small Resolution Loss and Optimal Tile Complexity

Demaine, Patitz, Schweller, Summers (STACS 2011)*:*

given: finite shape *S*, |*S*|=*n*

there is a TAS T, |T| ≈ K(*S*), that assembles S at scale factor ≈ log *n*, with one step of dissolving RNA tiles

program *p* that prints *S*

S = 

RNA tiles:

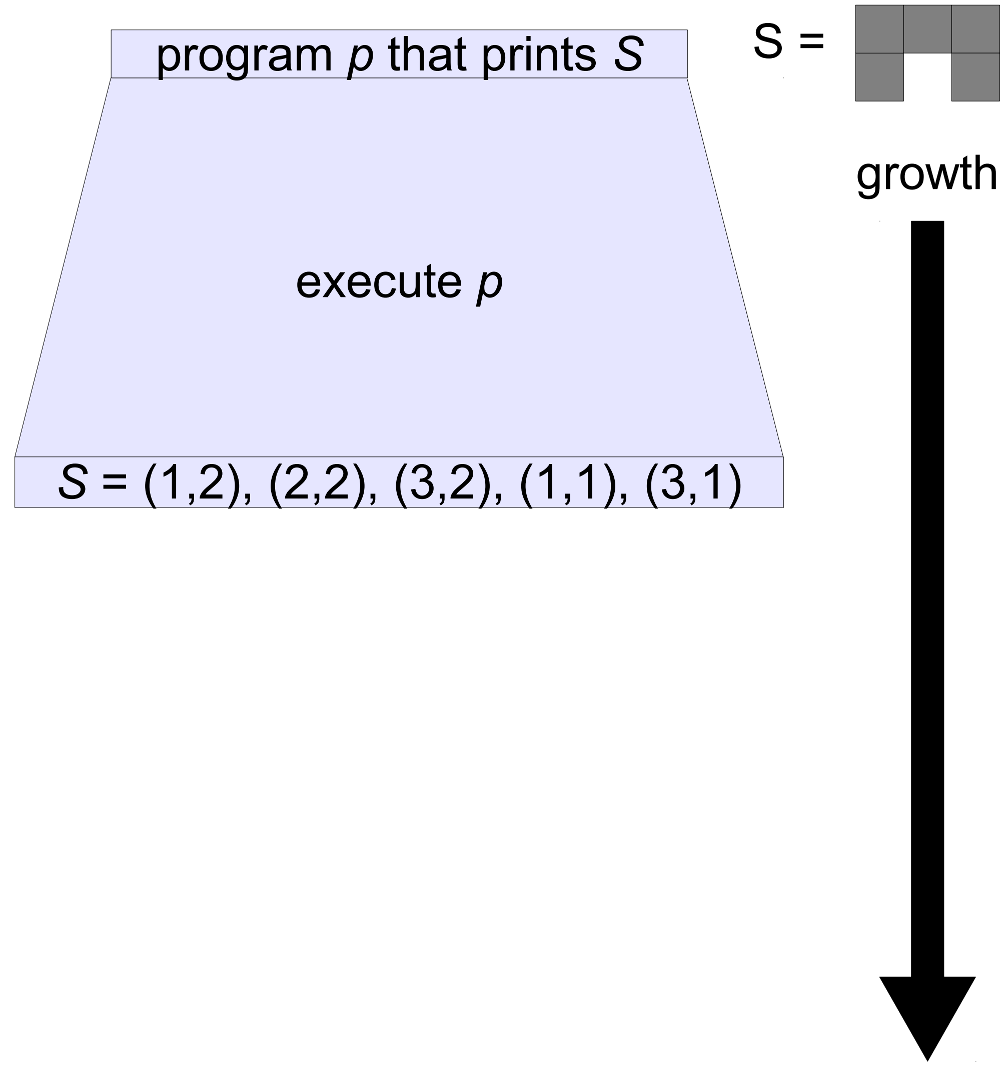
DNA tiles:

Shape-Building with Small Resolution Loss and Optimal Tile Complexity

Demaine, Patitz, Schweller, Summers (STACS 2011)*:*

given: finite shape *S*, |*S*|=*n*

there is a TAS T, |T| ≈ K(*S*), that assembles S at scale factor ≈ log *n*, with one step of dissolving RNA tiles



RNA tiles:

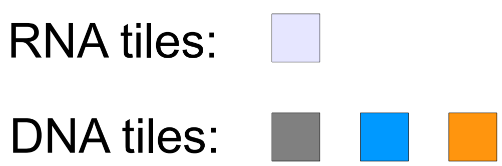
DNA tiles:

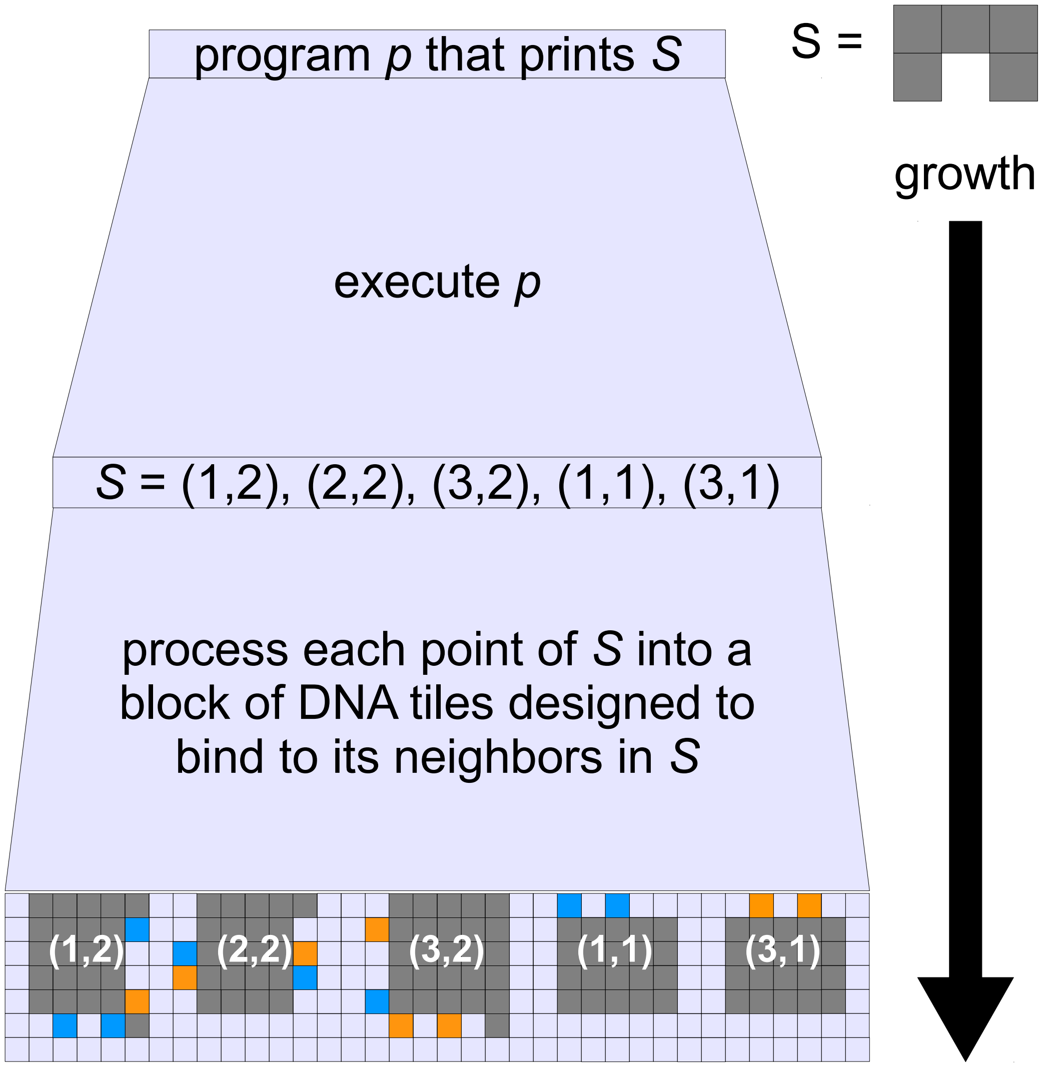
Shape-Building with Small Resolution Loss and Optimal Tile Complexity

Demaine, Patitz, Schweller, Summers (STACS 2011)*:*

given: finite shape *S*, |*S*|=*n*

there is a TAS T, |T| ≈ K(*S*), that assembles S at scale factor ≈ log *n*, with one step of dissolving RNA tiles



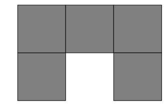


Shape-Building with Small Resolution Loss and Optimal Tile Complexity

Demaine, Patitz, Schweller, Summers (STACS 2011)*:*

given: finite shape *S*, $|S|=n$

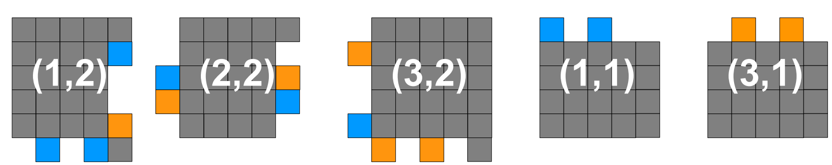
there is a TAS T, $|T| \approx K(S)$, that assembles S at scale factor $\approx \log n$, with one step of dissolving RNA tiles

S = 

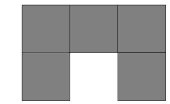
dissolve RNA

RNA tiles:

DNA tiles:



Shape-Building with Small Resolution Loss and Optimal Tile Complexity

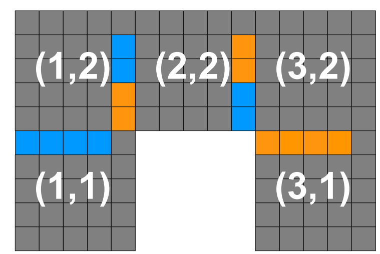
S = 

Demaine, Patitz, Schweller, Summers (STACS 2011)*:*

given: finite shape *S*, |*S*|=*n*

there is a TAS T, |T| ≈ K(*S*), that assembles S at scale factor ≈ log *n*, with one step of dissolving RNA tiles

dissolve RNA



RNA tiles:

DNA tiles: