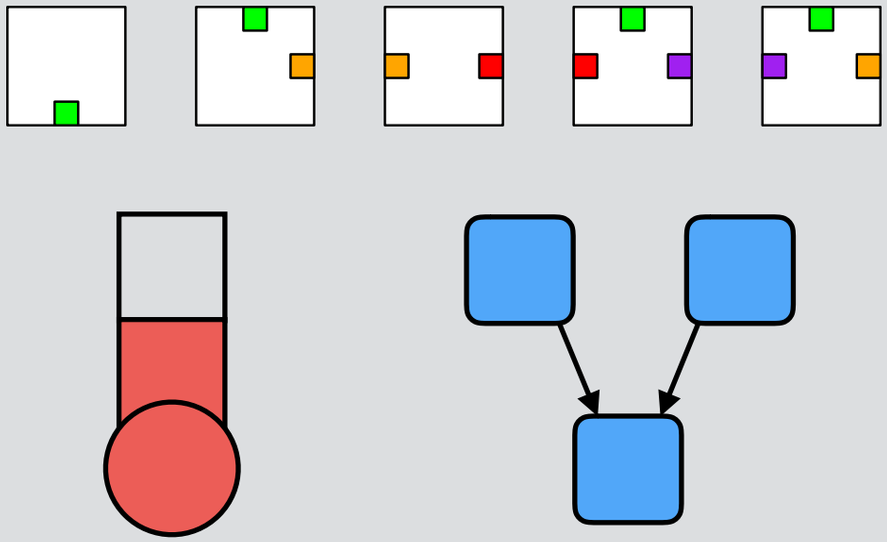
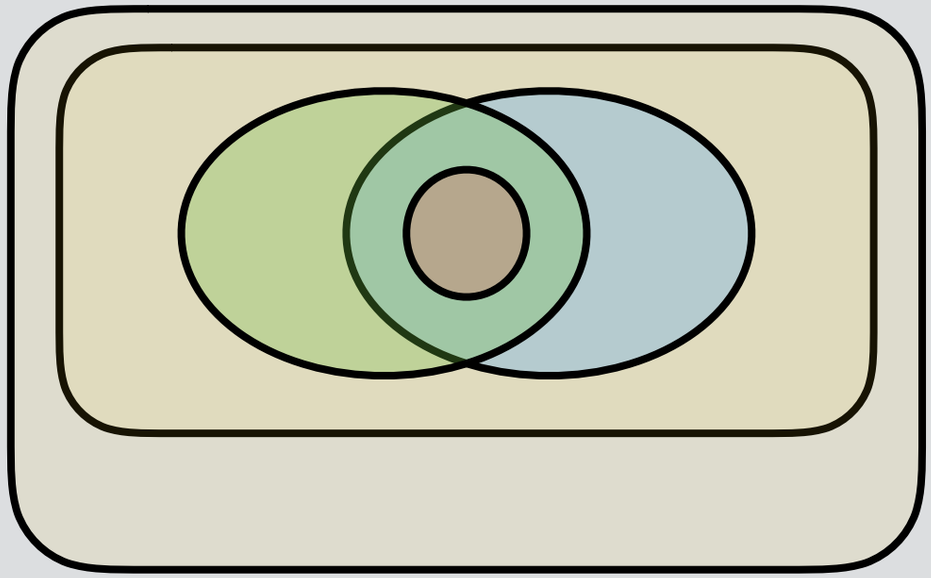
# Verification in Staged Tile Self-Assembly

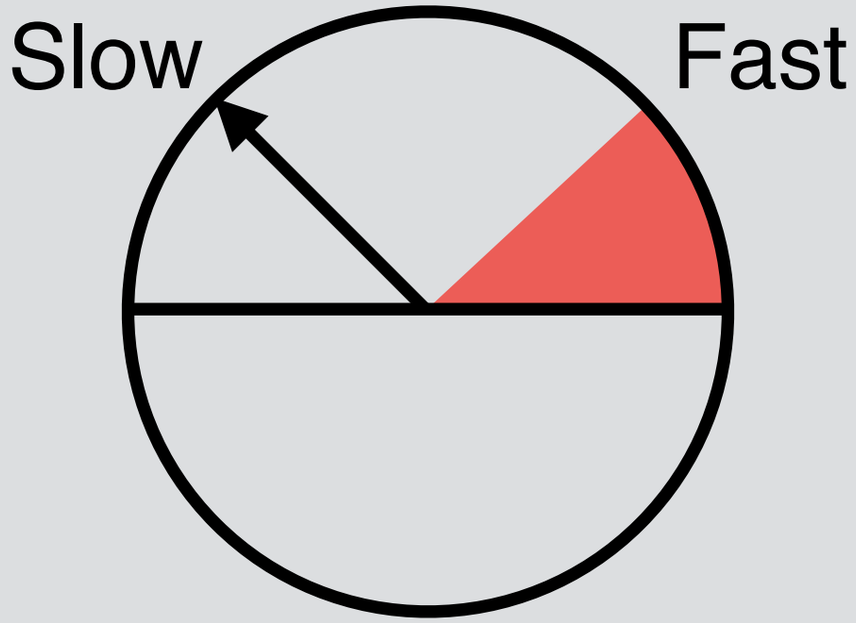




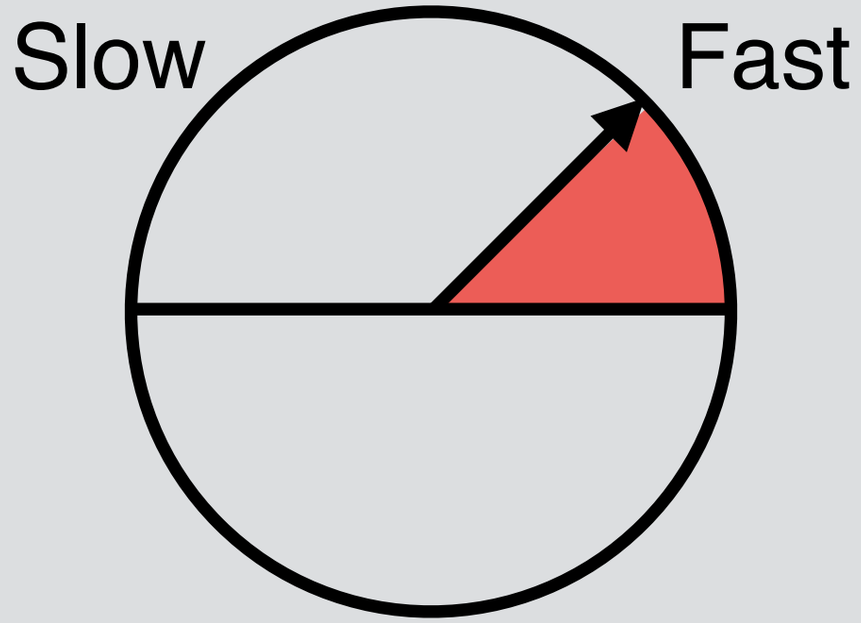
Robert Schweller, Andrew Winslow, Tim Wylie
University of Texas Rio Grande Valley

# DNA tile self-assembly

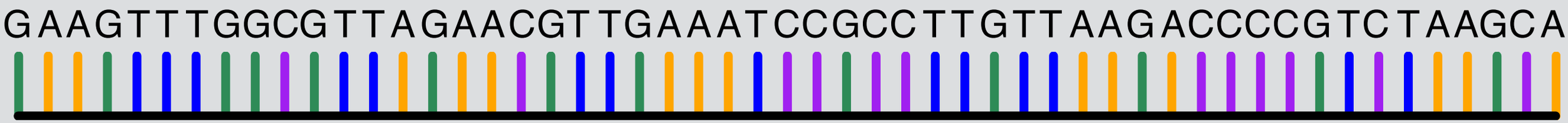
# DNA tile self-assembly



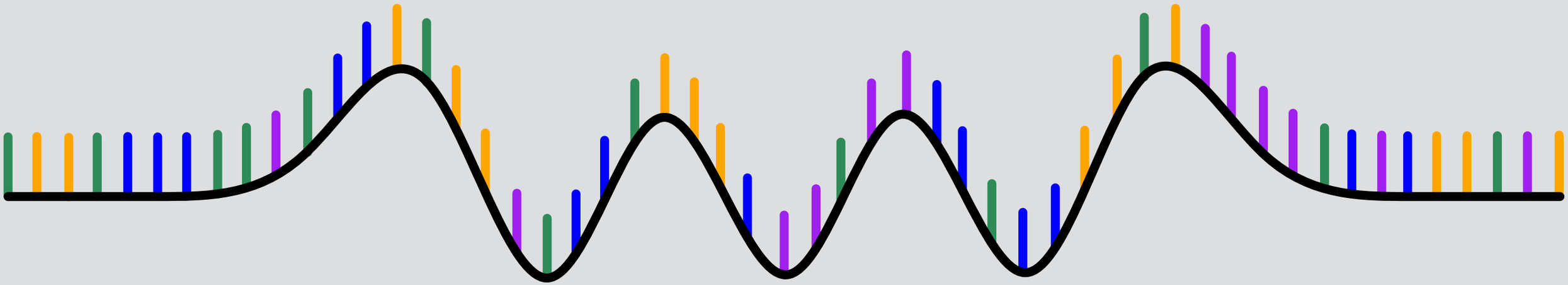
# DNA tile self-assembly

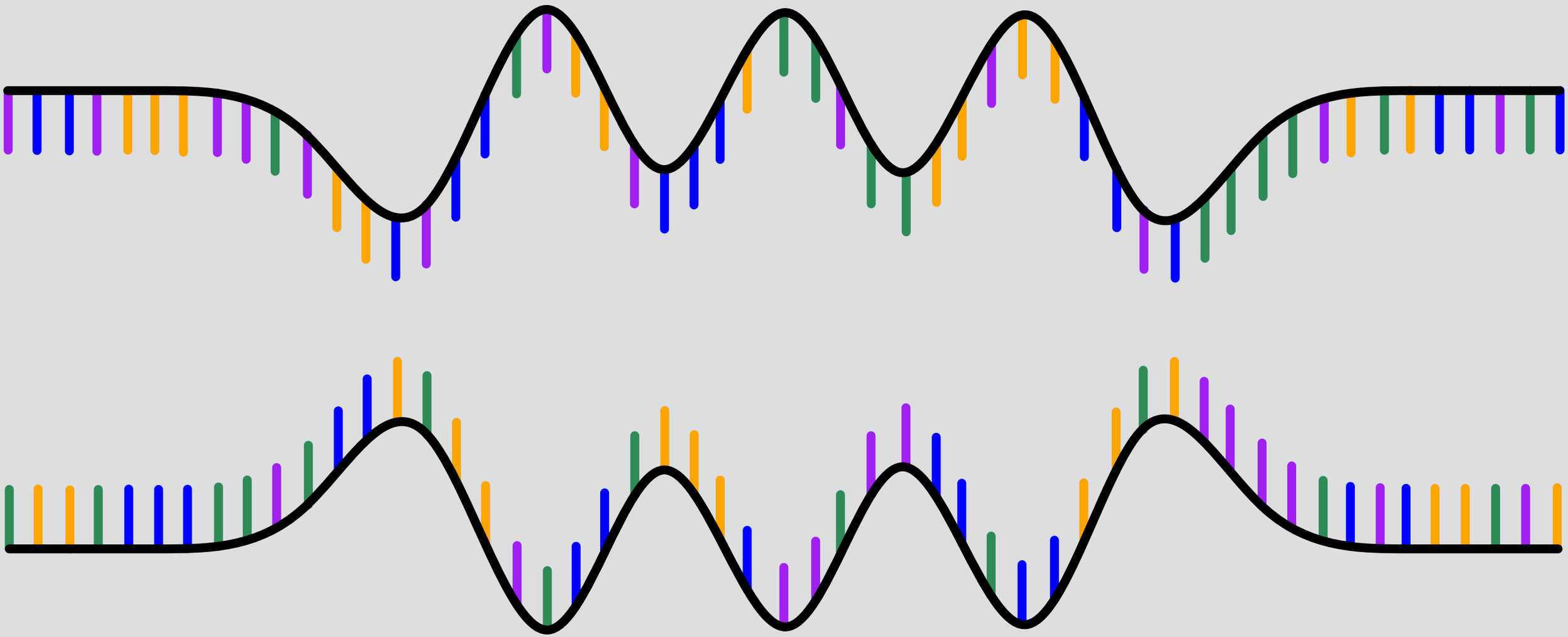


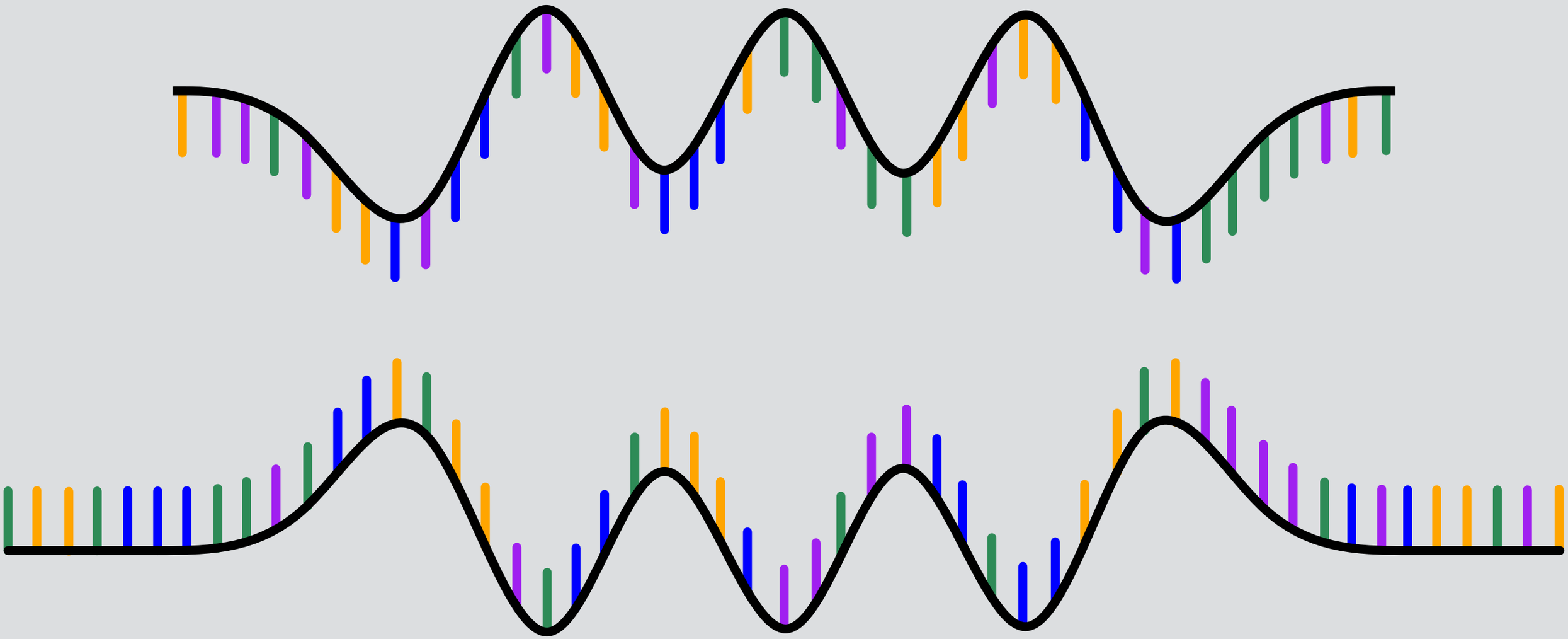
# DNA

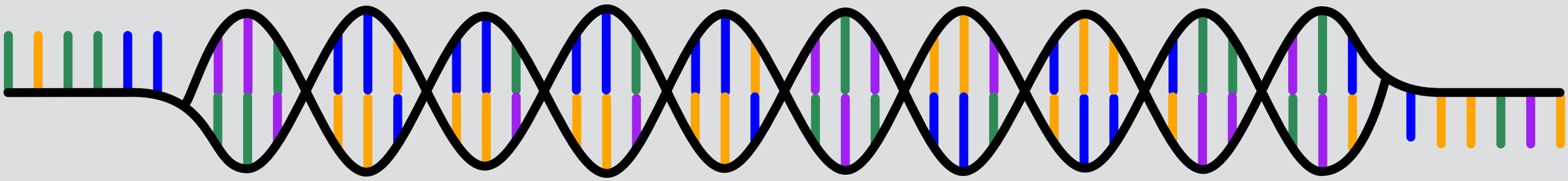



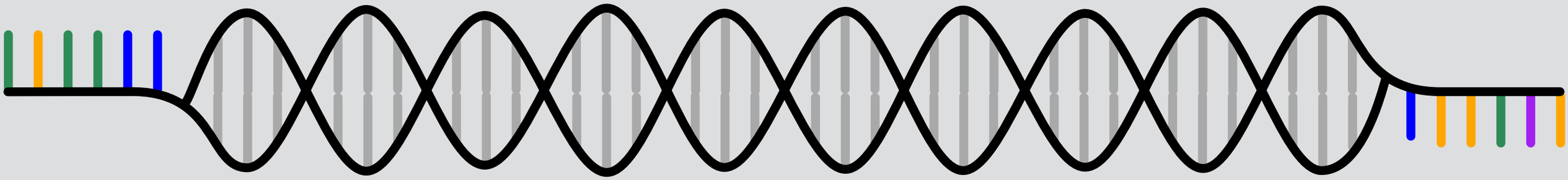

Single strand

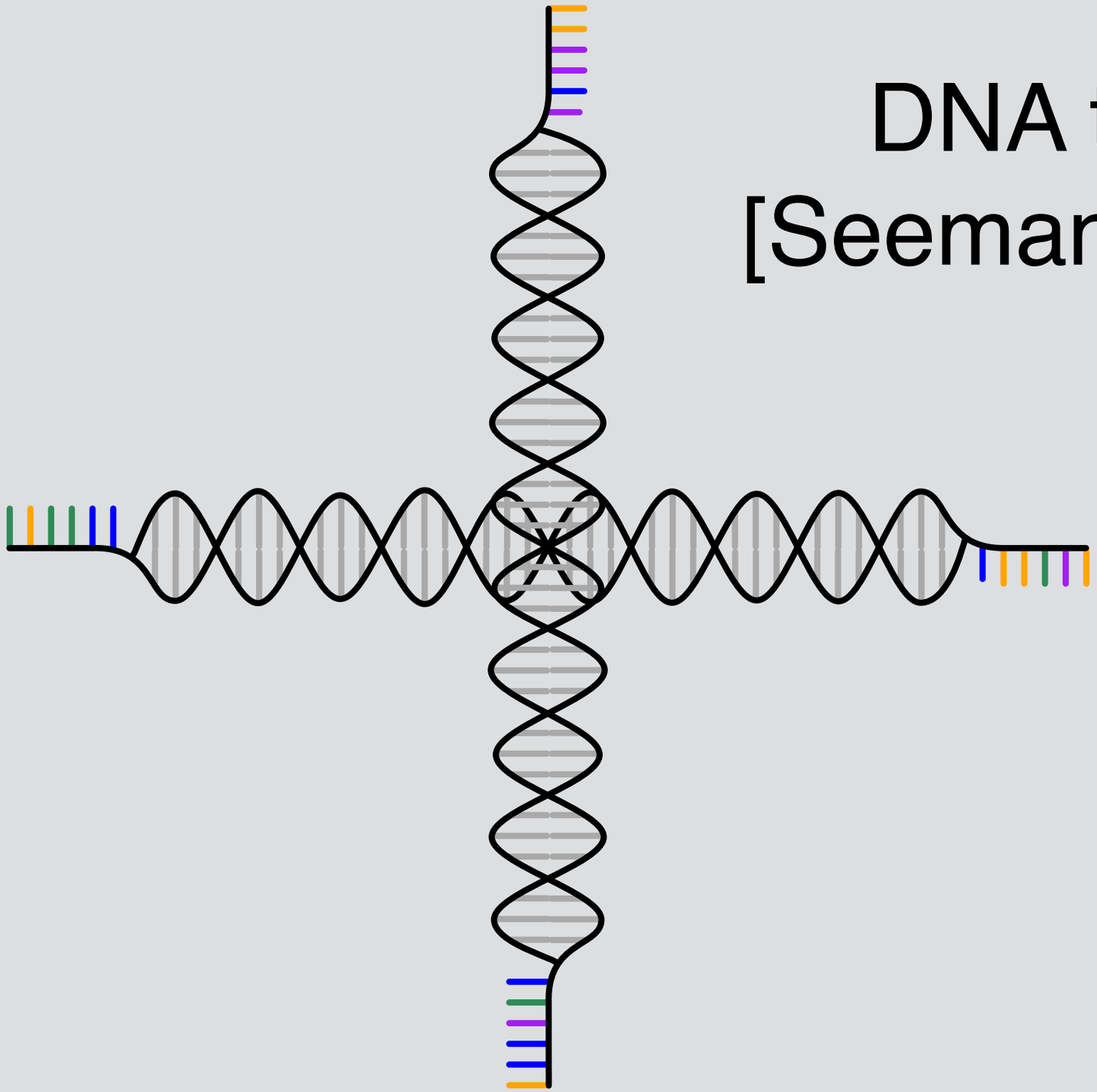






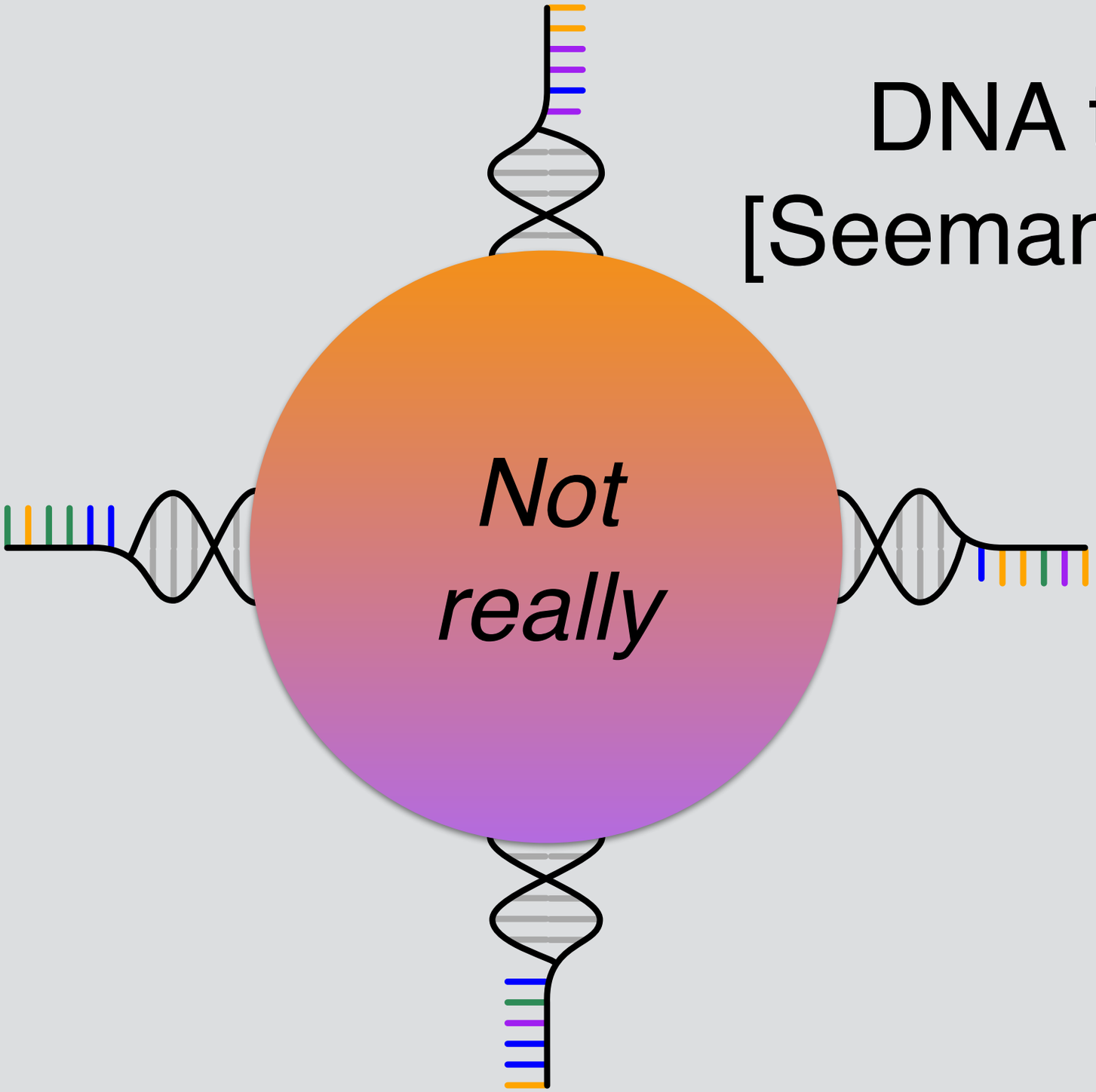




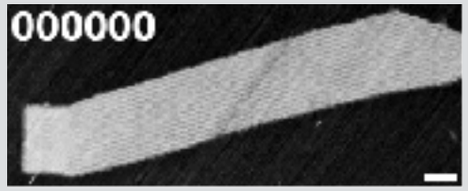


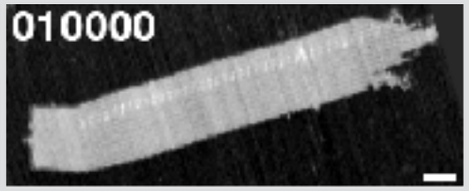
# DNA tiles
# [Seeman 1982]

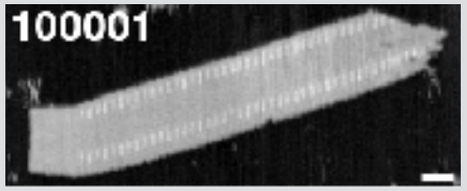
DNA tiles
[Seeman 1982]

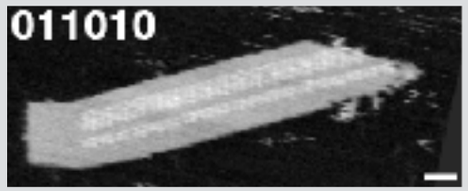


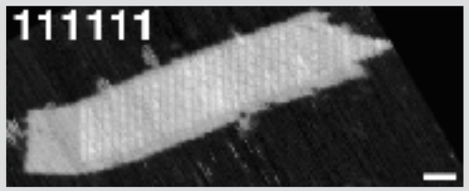
# DNA tile self-assembly

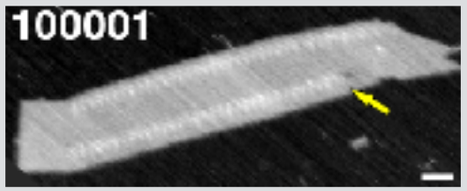


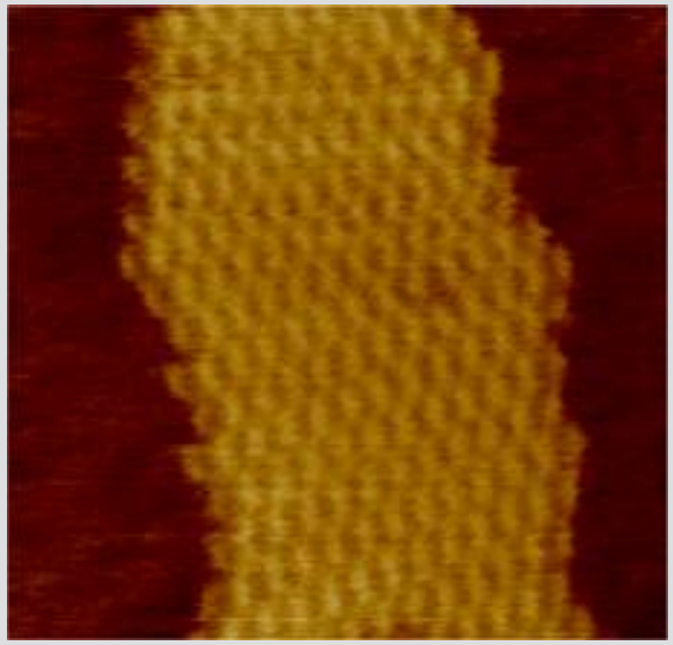



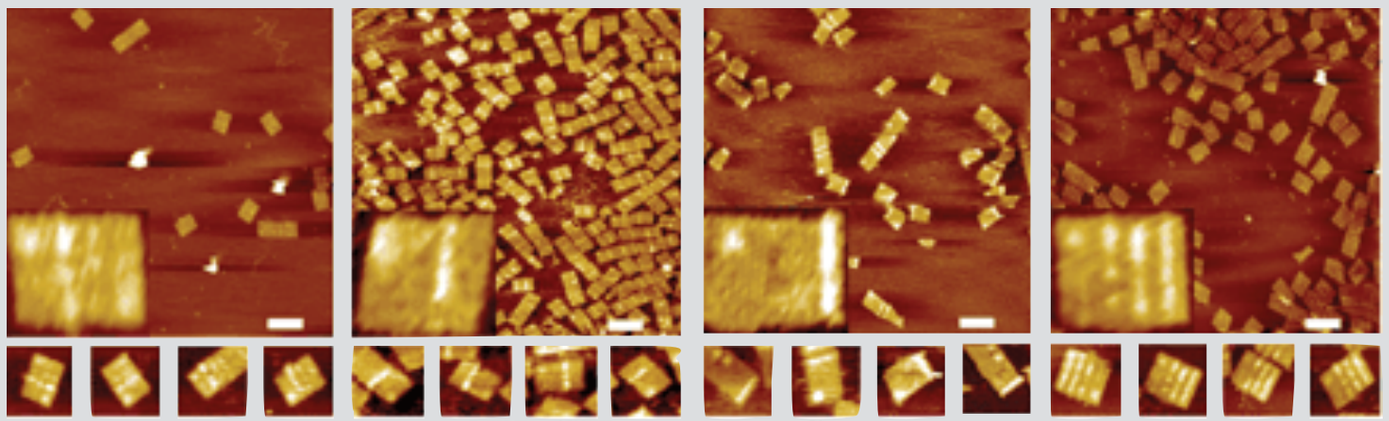



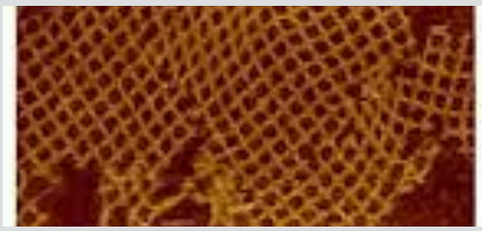



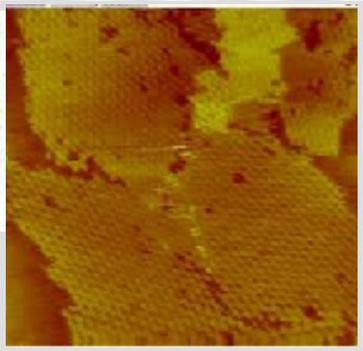



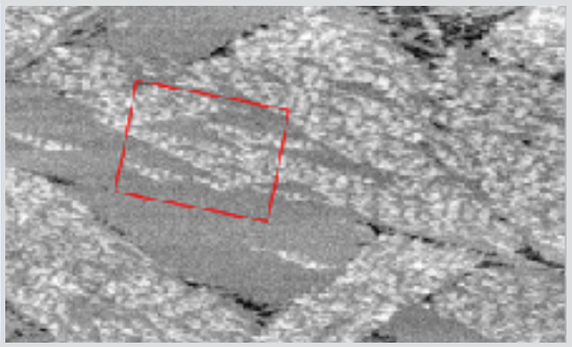



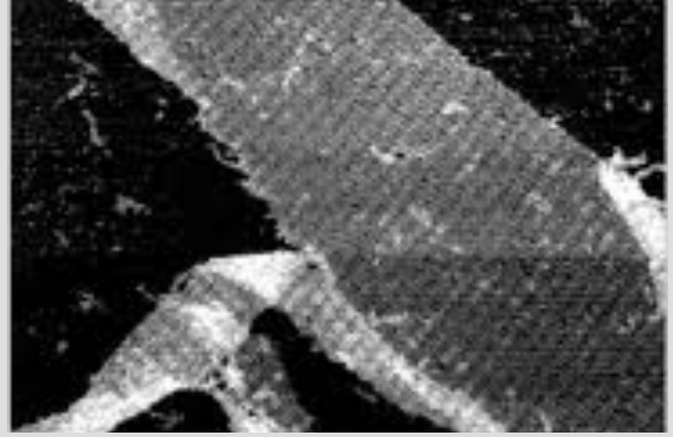



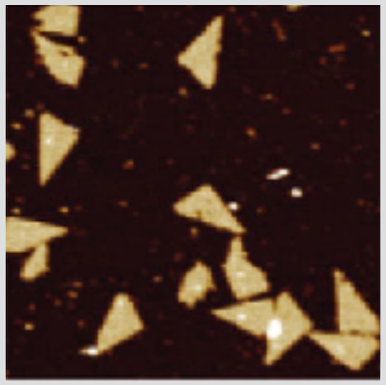


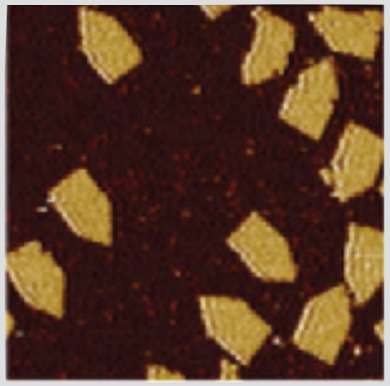


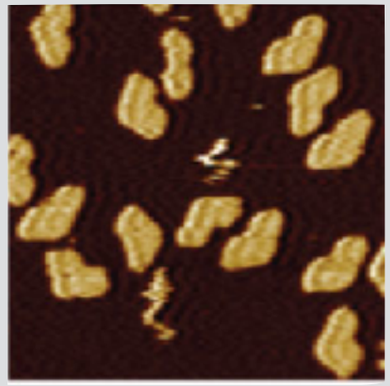


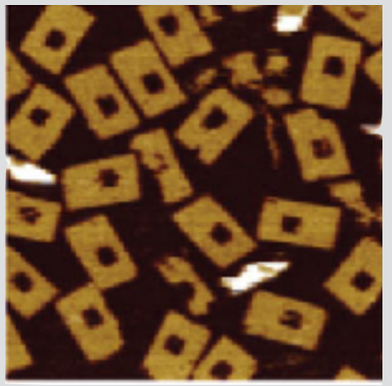




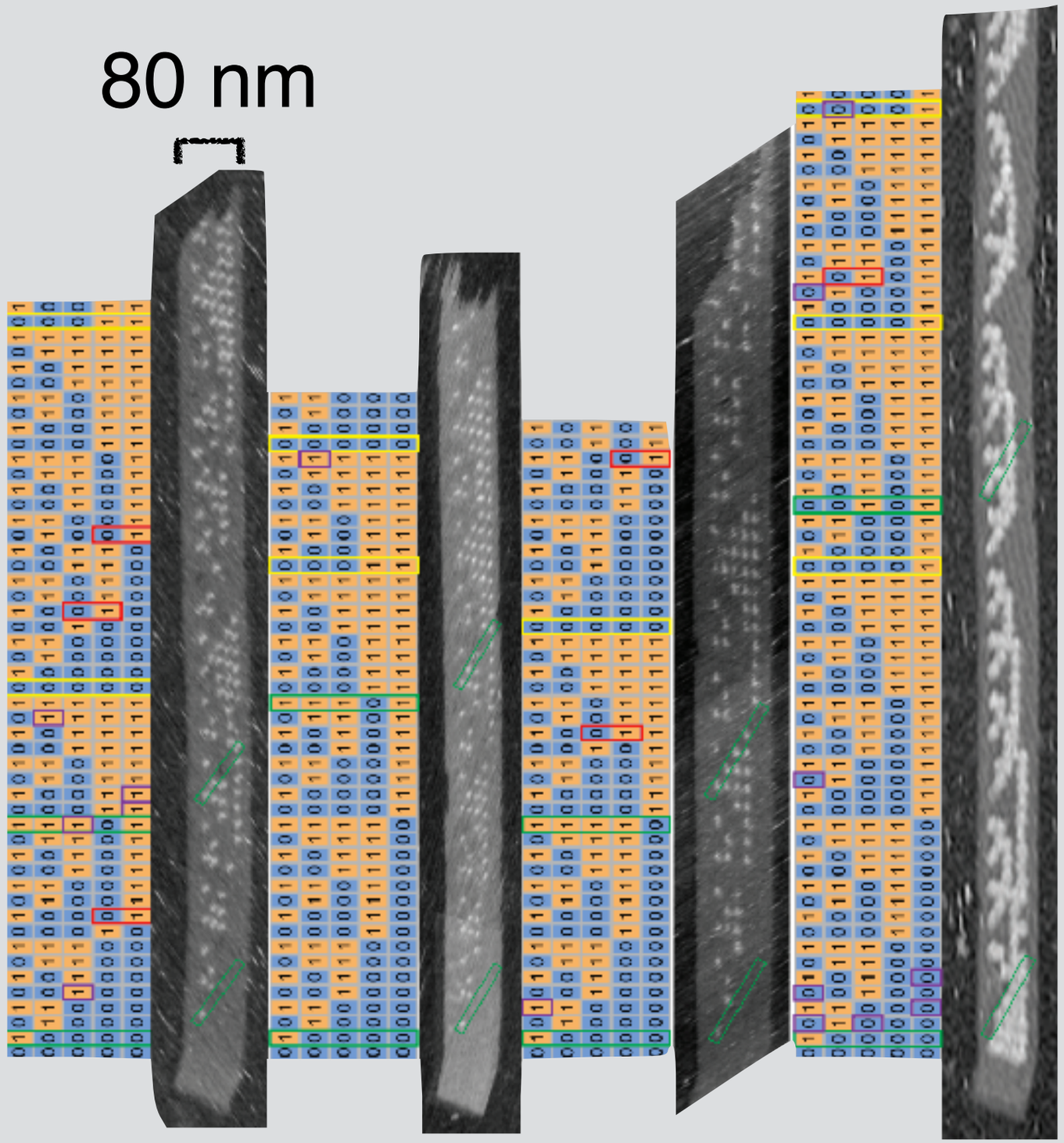


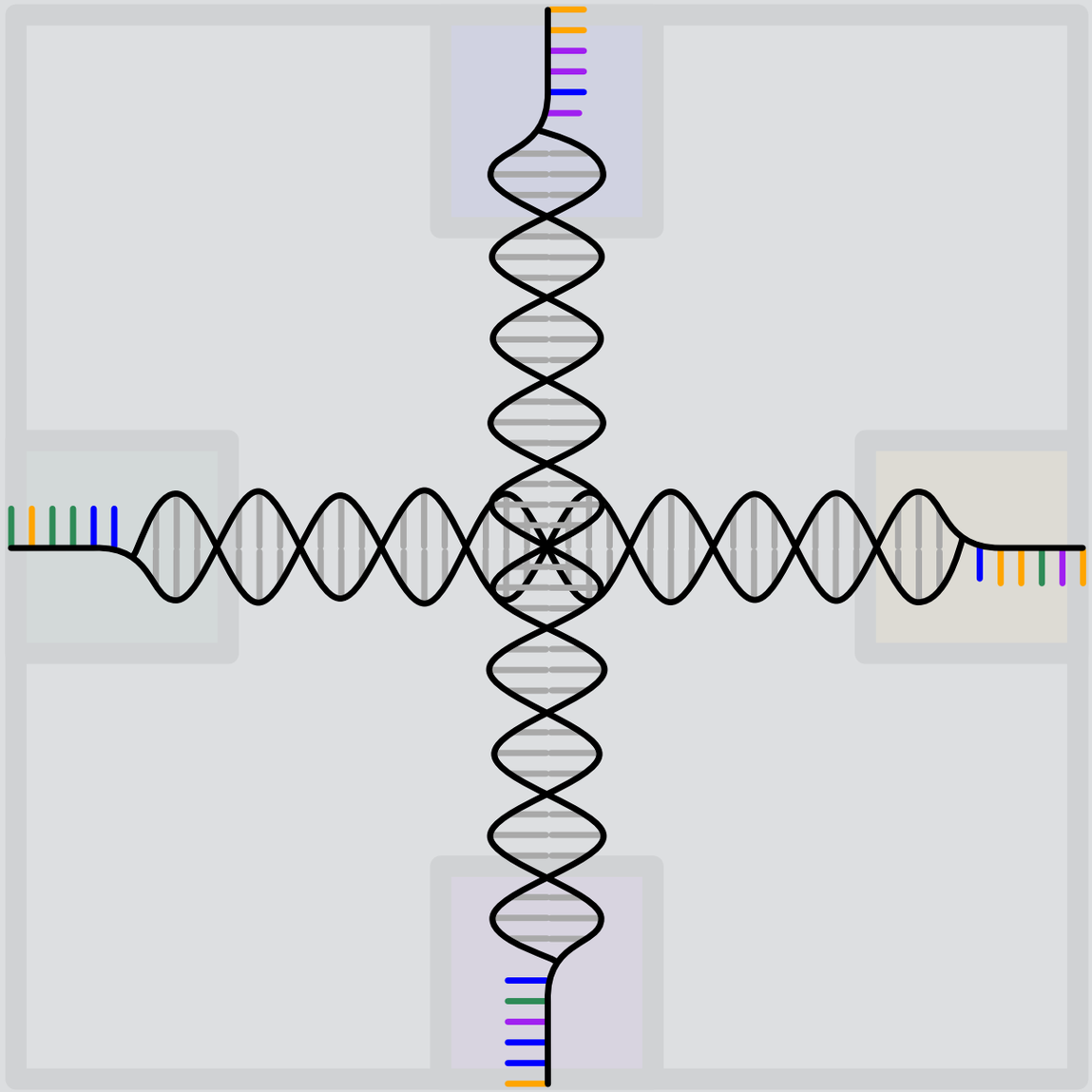




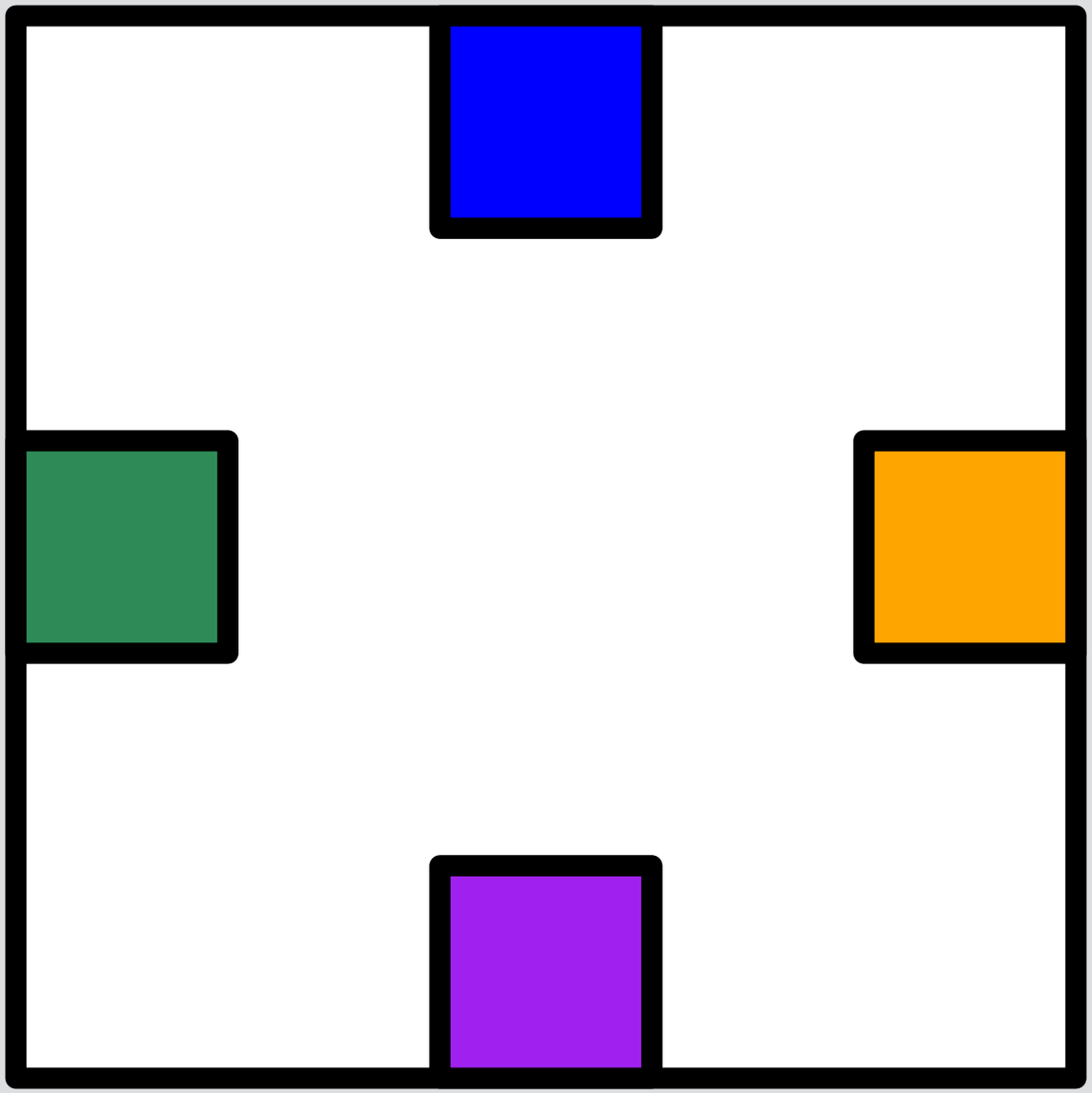
# DNA tile binary counters

[Evans, 2014]





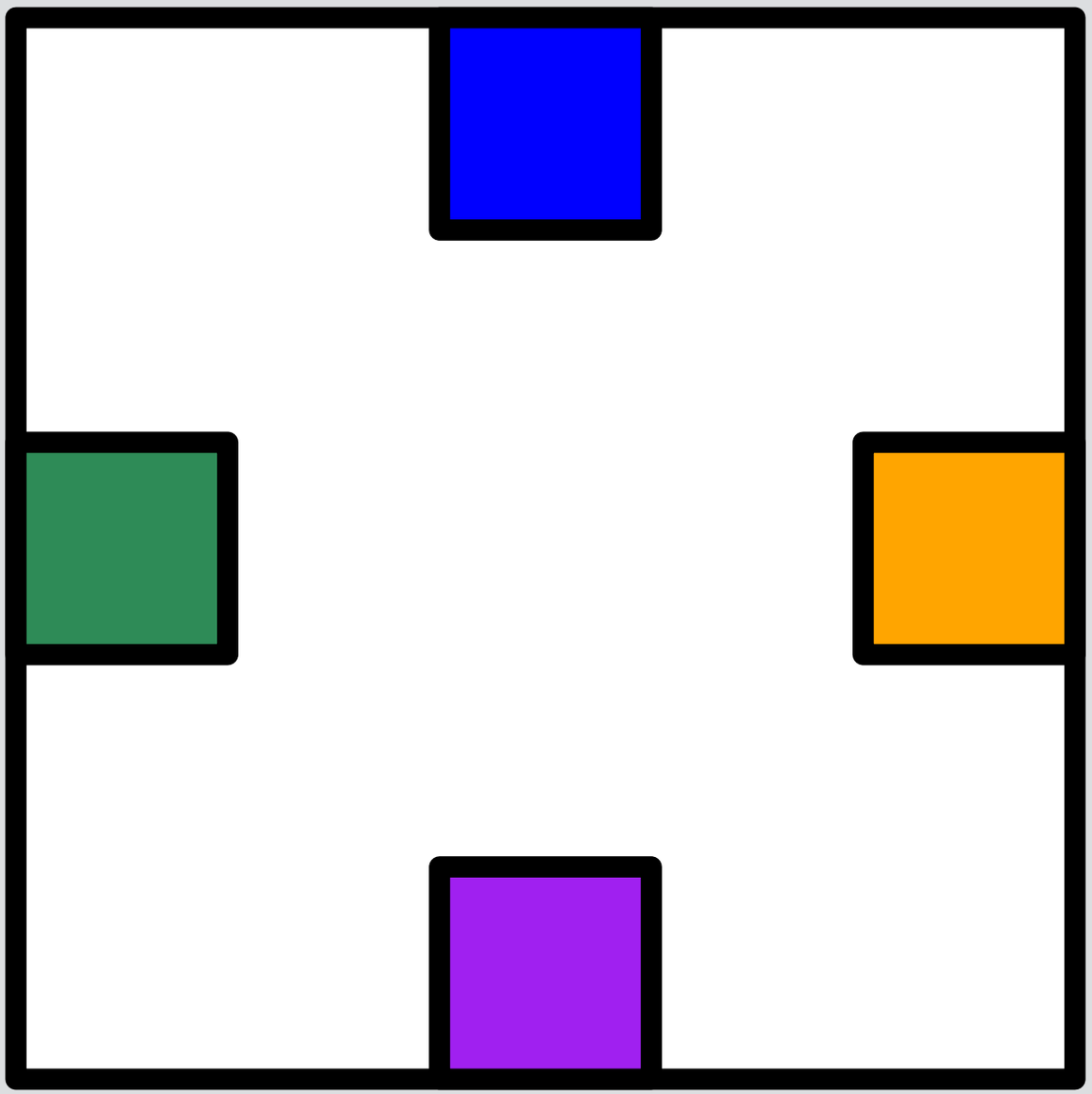
DNA tile



Tile [Winfree 1996]

(Also Adleman? Rothemund?)

# Glues

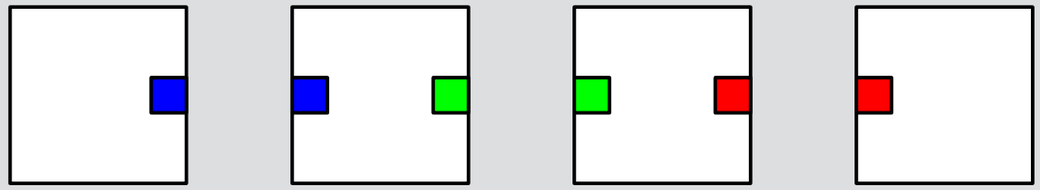


Tile

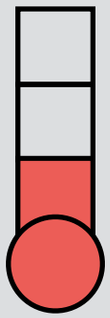
[Winfree 1996]

(Also Adleman? Rothemund?)

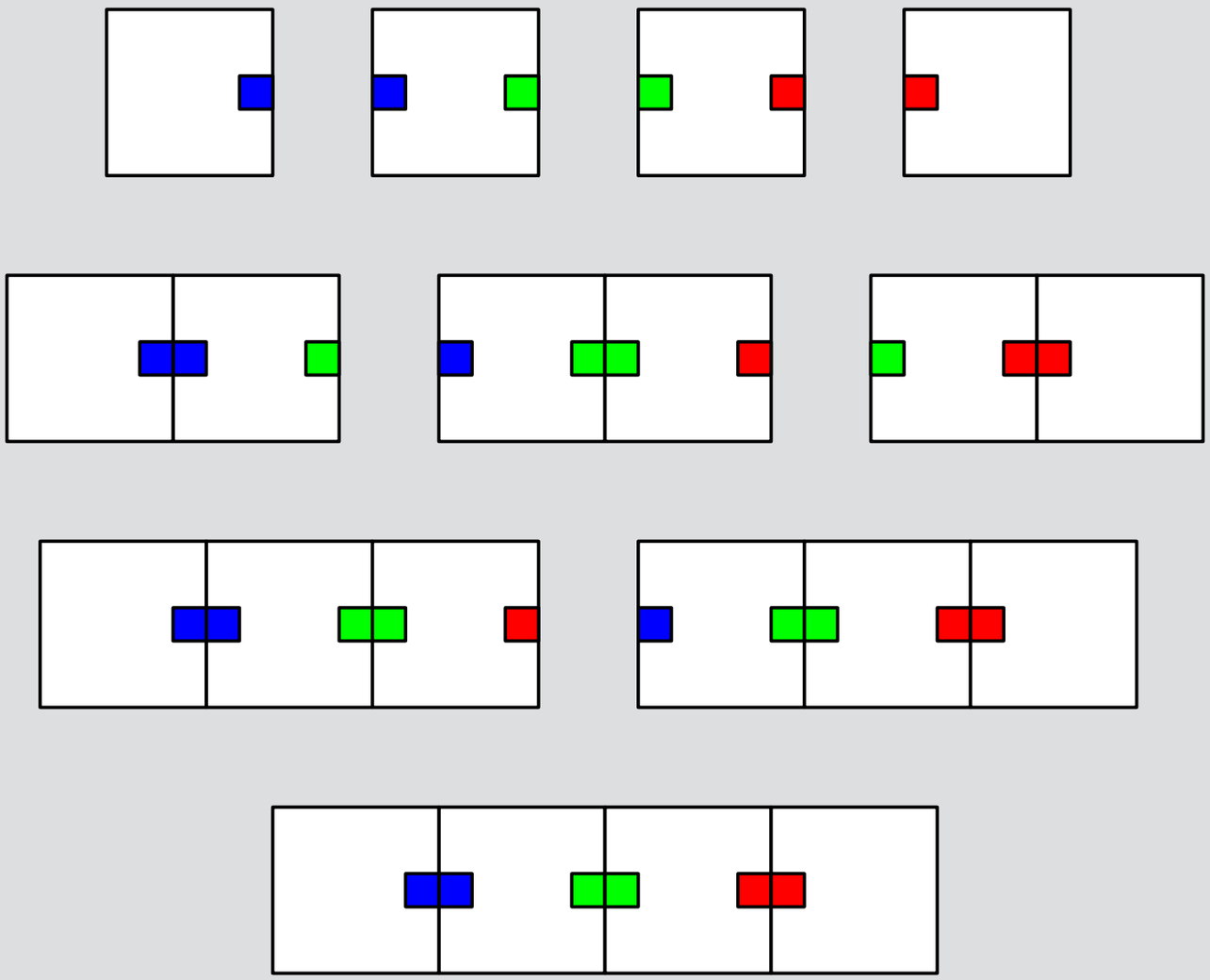
# Two-handed tile assembly



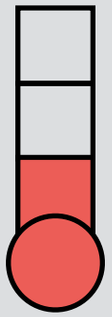
## Tile types



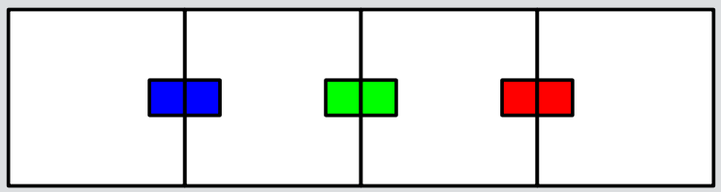
# Two-handed tile assembly



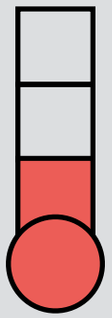
Producible assemblies



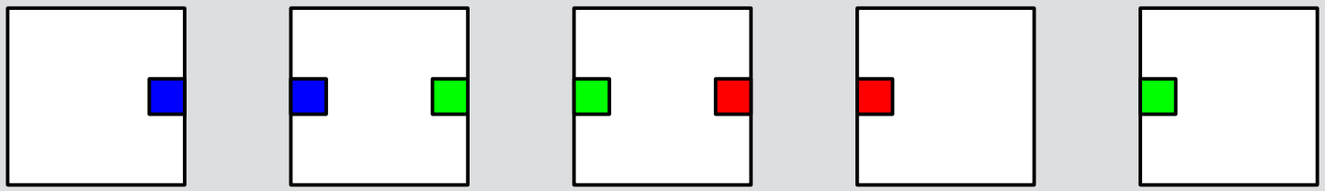
# Two-handed tile assembly



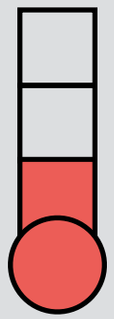
Terminal assemblies



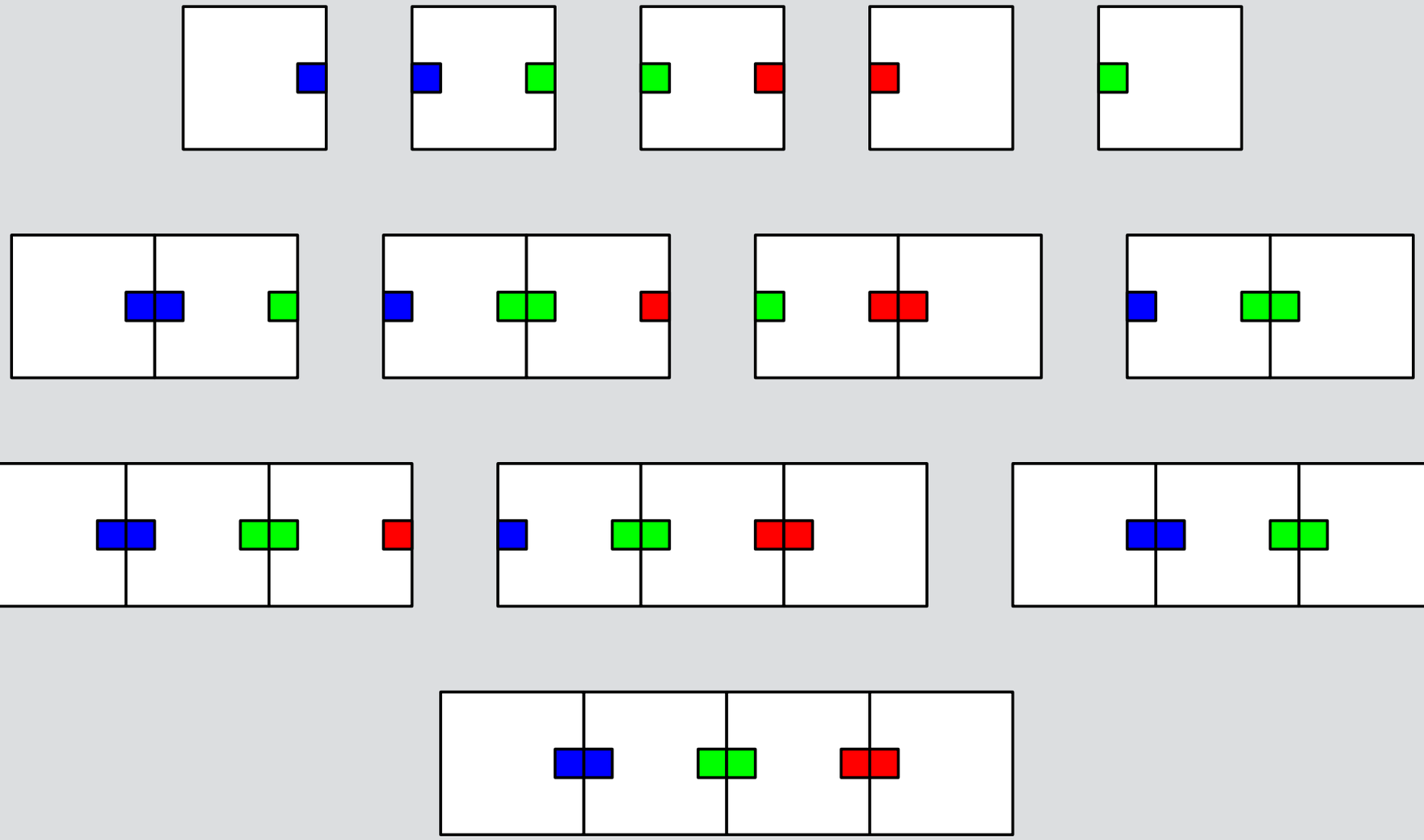
# Two-handed tile assembly



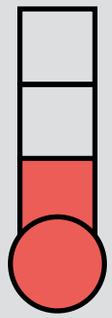
Tile types



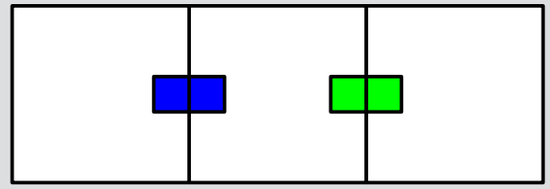
# Two-handed tile assembly

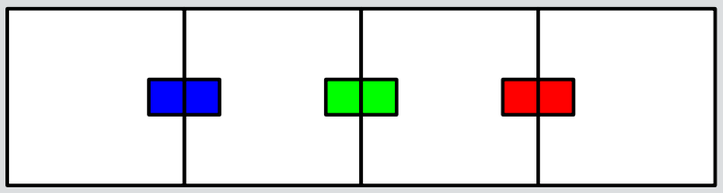


Producible assemblies

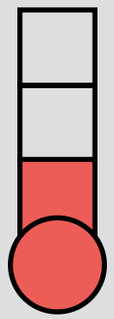


# Two-handed tile assembly

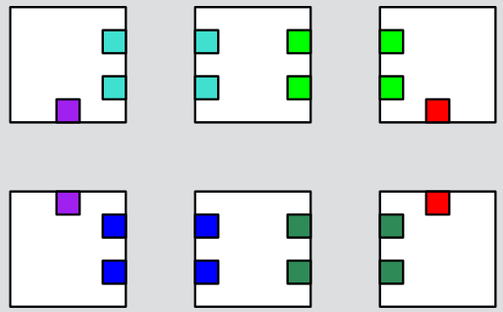




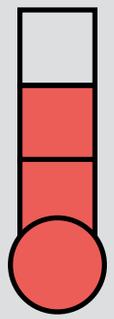
## Terminal assemblies



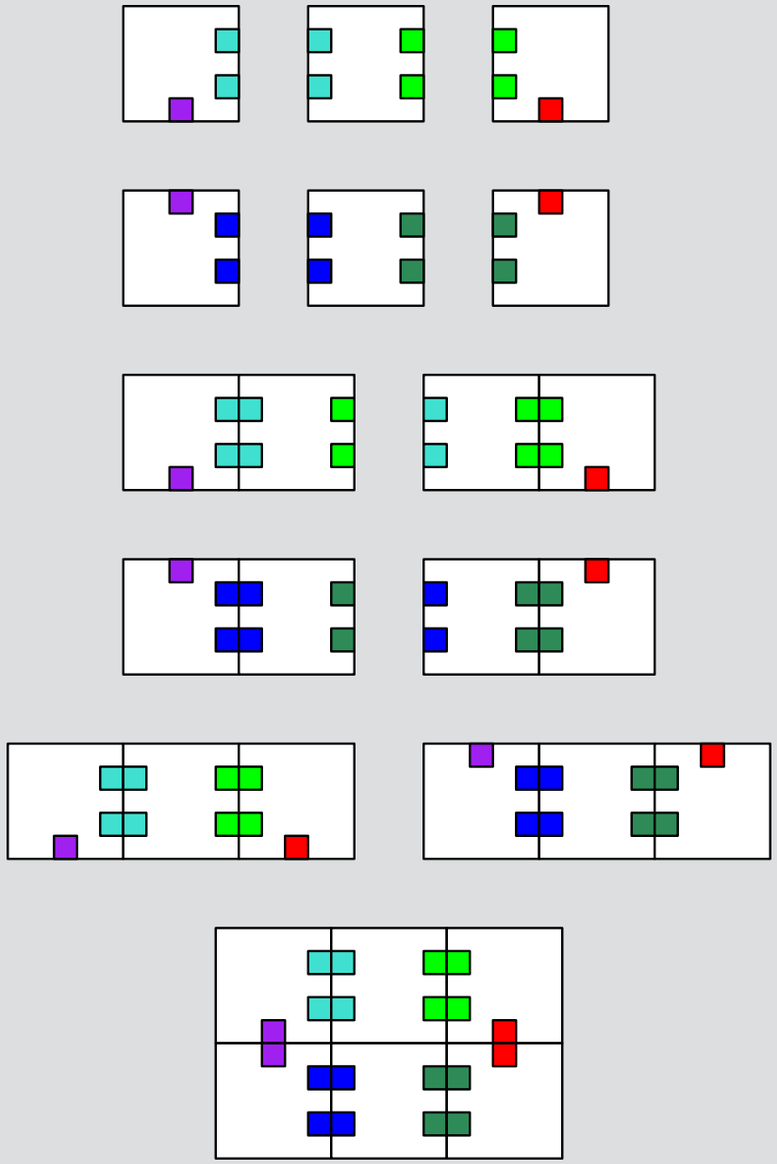
# Two-handed tile assembly



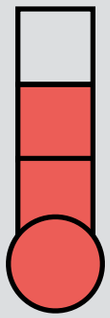
Tile types



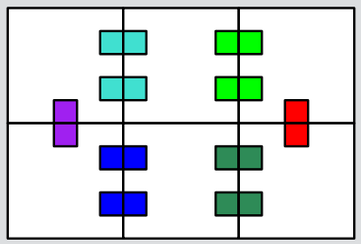
# Two-handed tile assembly



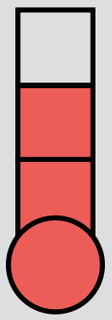
# Producible assemblies



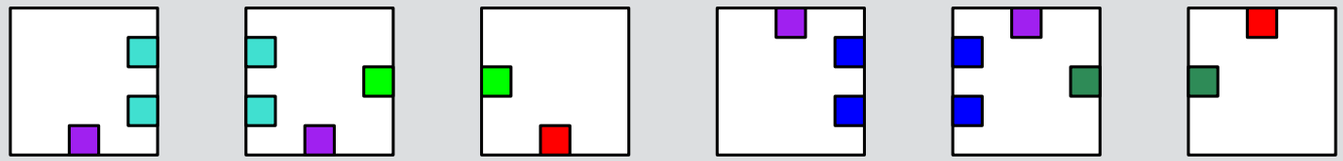
# Two-handed tile assembly



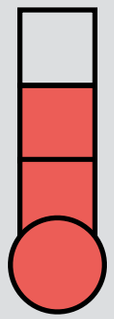
Terminal assemblies



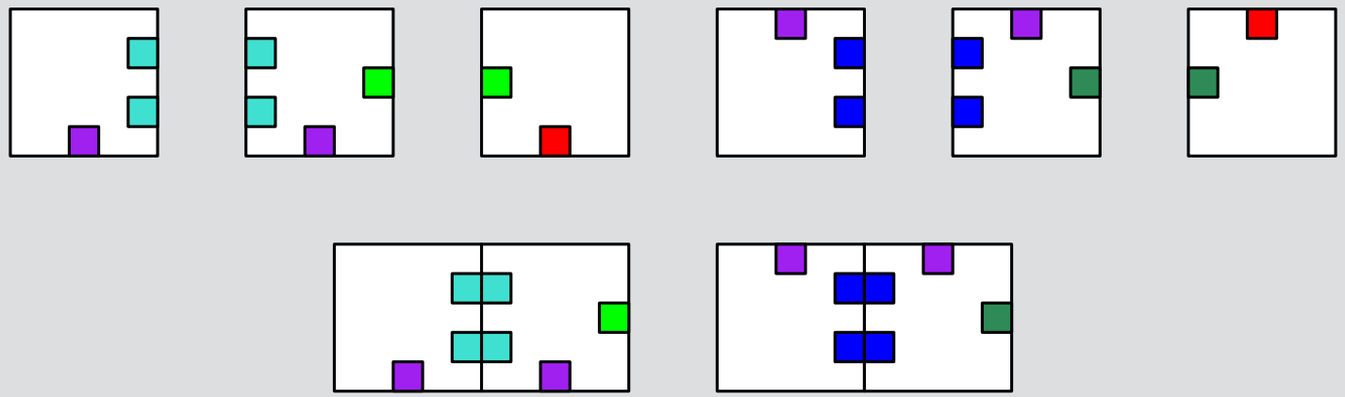
# Two-handed tile assembly

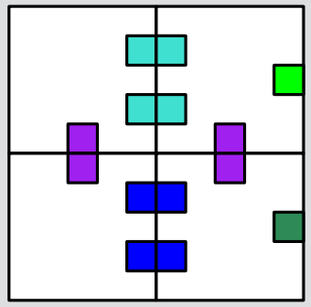


Tile types

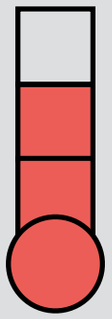


# Two-handed tile assembly

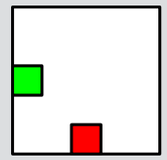


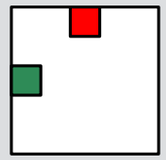


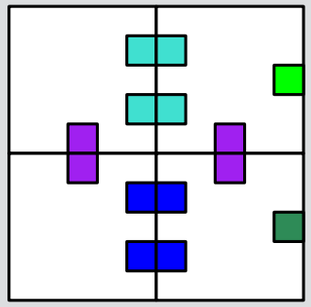
# Producible assemblies



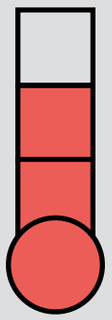
# Two-handed tile assembly



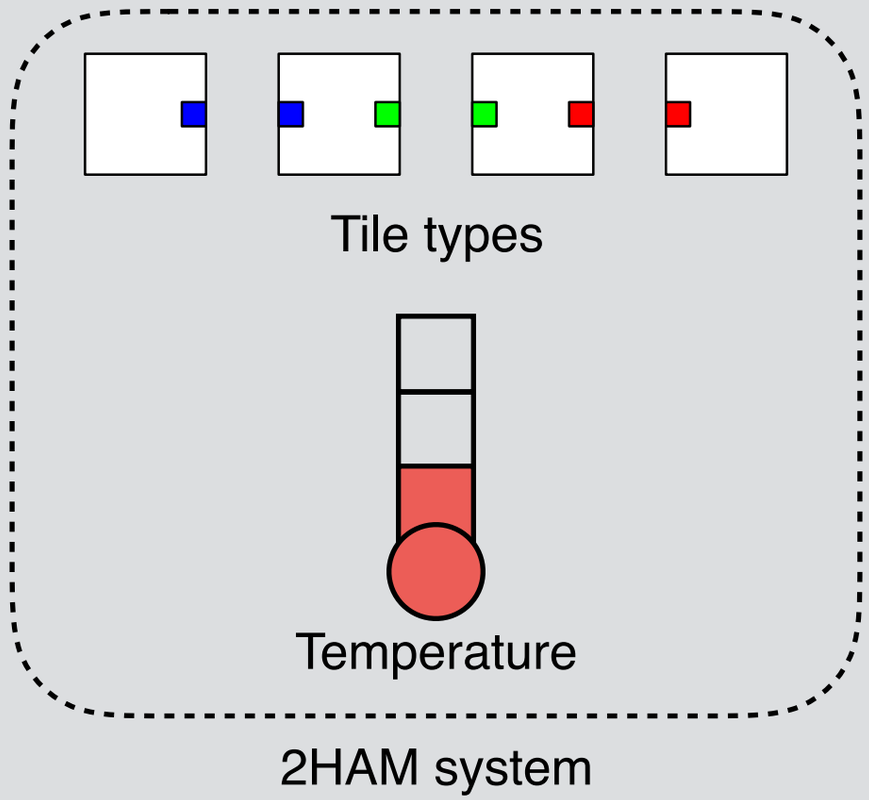




## Terminal assemblies



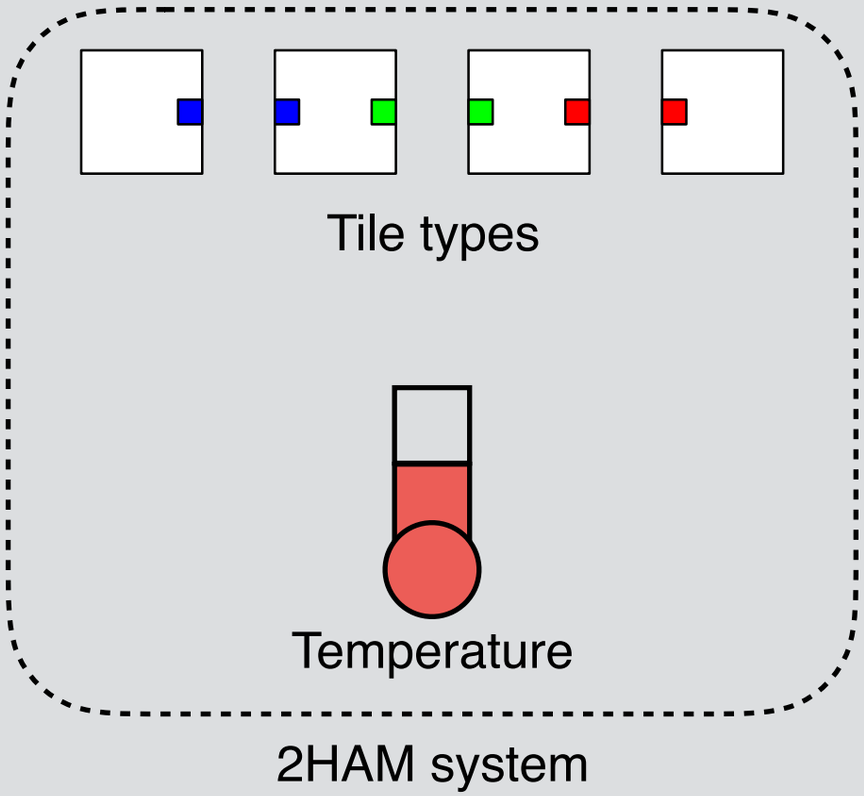
# Two-handed tile assembly model (2HAM)




This paper is about **verifying** the **behavior** of 2HAM systems.

# Verification problems

## Input




?

maybe some
other stuff

## Output

“Yes”

(system has a specific behavior)

or

“No”

(system does not have that behavior)

# Verification problems

Input

Output

Tile

ific behavior)

Temp

e that behavior)

2HAM

?

maybe some
other stuff

**VEMDP 2014**

**Verification of Engineered Molecular Devices and Programs**

**July 17, 2014, Vienna, Austria**

**An affiliated workshop of CAV 2014**

**VEMDP 2015**

**Verification of Engineered Molecular Devices and Programs**

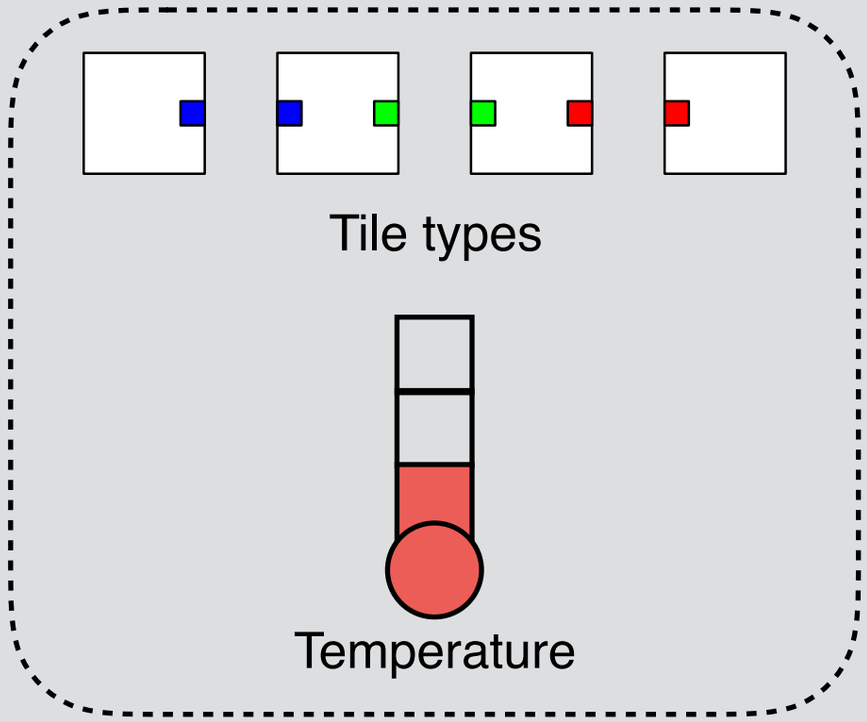
**July 19, 2015, San Francisco, USA**

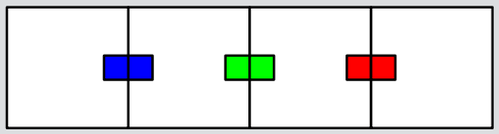
**An affiliated workshop of CAV 2015, co-sponsored by the NSF Molecular Programming Project**

# Specific verification problems

# 2HAM Unique Assembly Verification (UAV) Problem

## Input

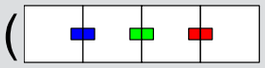


Assembly

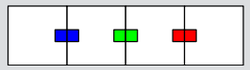
## Output

“Yes”

(  is the unique terminal assembly of the input system)

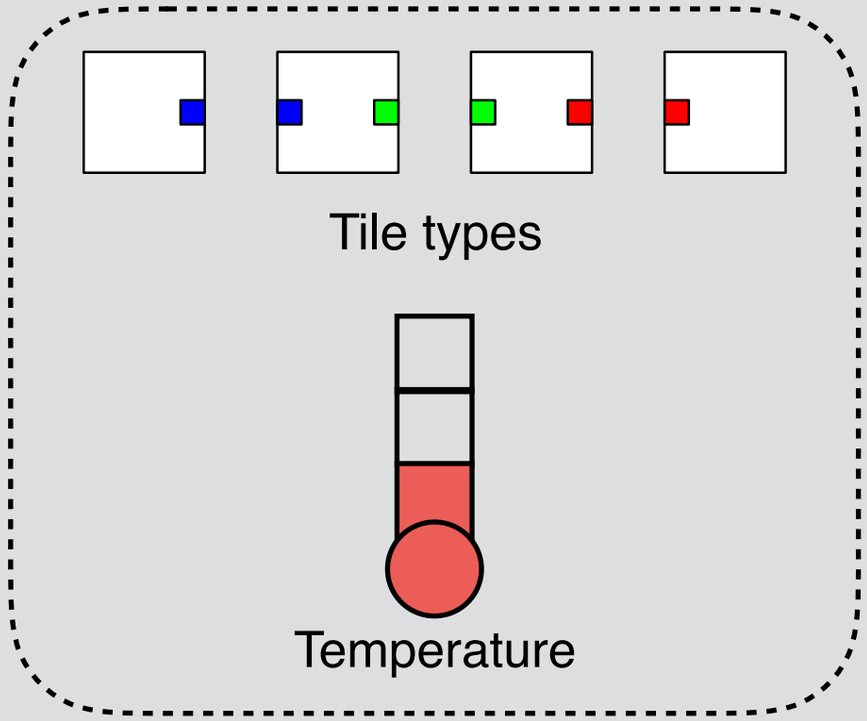
or

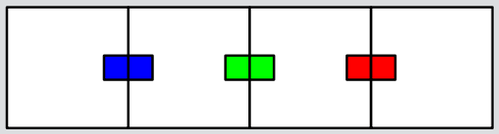
“No”

(  is not a terminal assembly or there is another terminal assembly of the input system)

# 2HAM Unique Assembly Verification (UAV) Problem

## Input

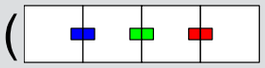


Assembly

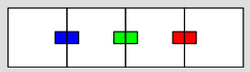
## Output

“Yes”

(  is the unique terminal assembly of the input system)

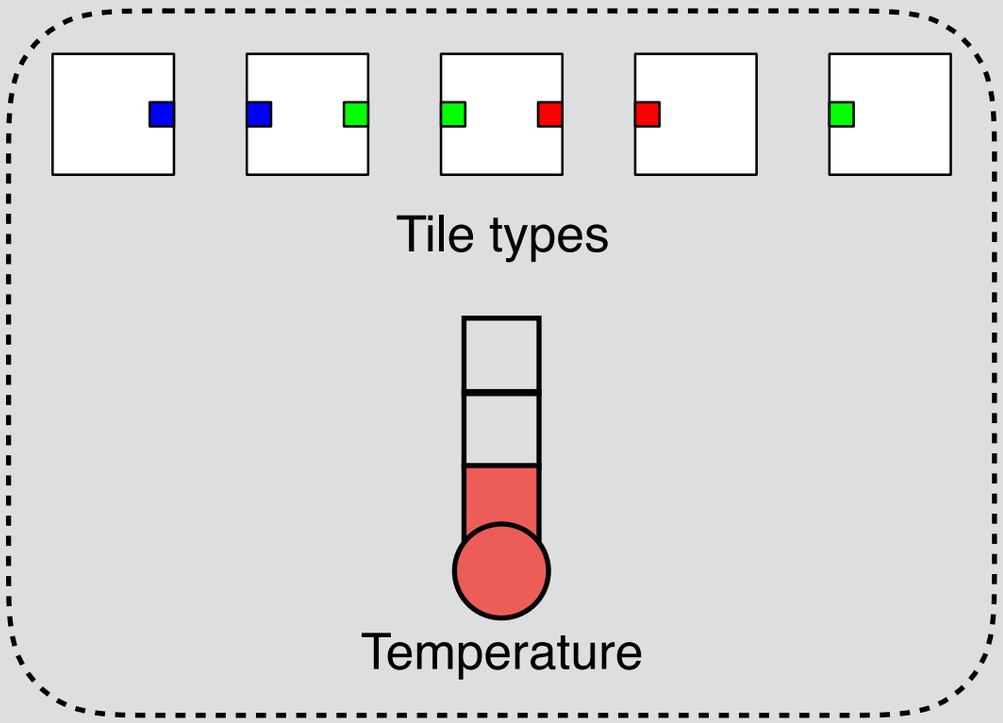
or

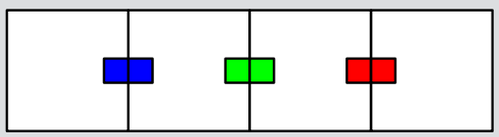
“No”

(  is not a terminal assembly or there is another terminal assembly of the input system)

# 2HAM Unique Assembly Verification (UAV) Problem

## Input

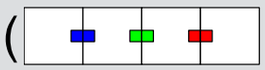


Assembly

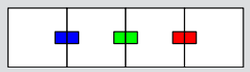
## Output

“Yes”

(  is the unique terminal assembly of the input system)

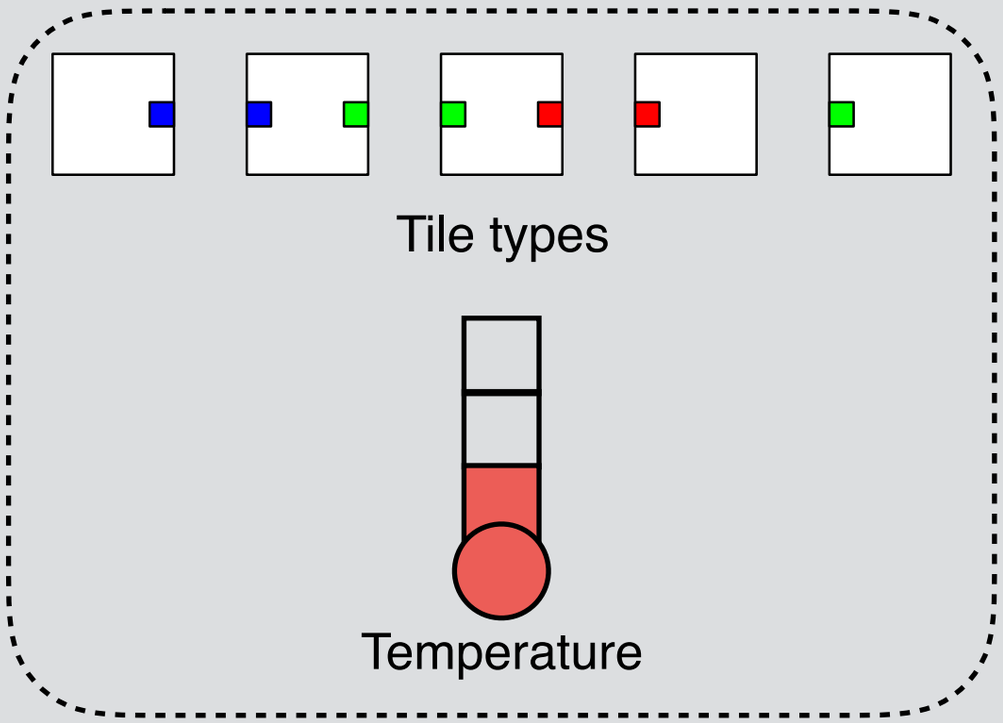
or

“No”

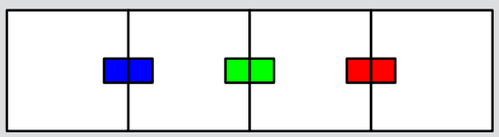
(  is not a terminal assembly or there is another terminal assembly of the input system)

# 2HAM Unique Assembly Verification (UAV) Problem

## Input



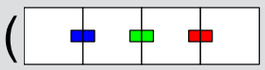

2HAM system



Assembly

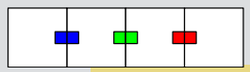
## Output

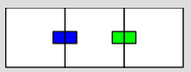
“Yes”

(  is the unique terminal assembly of the input system)

or

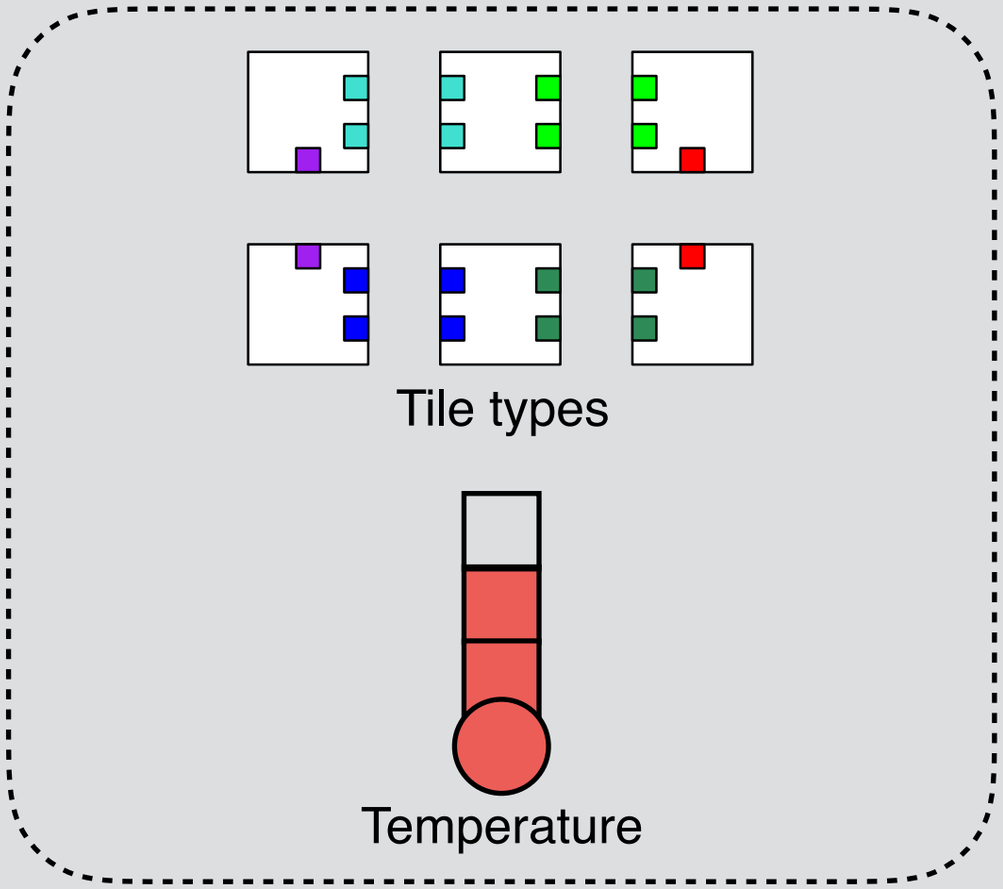
“No”

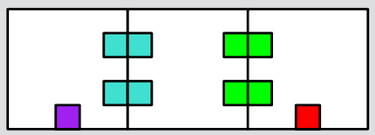
(  is not a terminal assembly or there is another terminal assembly of the input system)



# 2HAM Unique Assembly Verification (UAV) Problem

## Input



Assembly

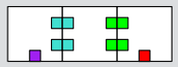
## Output

“Yes”

(  is the unique terminal assembly of the input system)

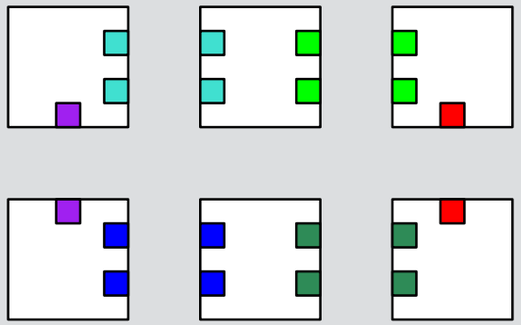
or

“No”

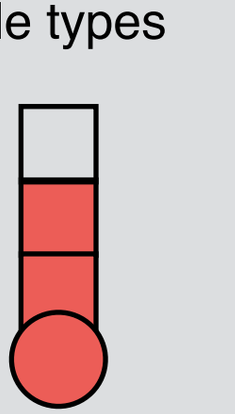
(  is not a terminal assembly or there is another terminal assembly of the input system)

# 2HAM Unique Assembly Verification (UAV) Problem

## Input

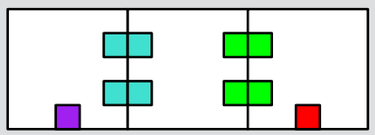


Tile types



Temperature

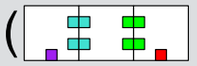
2HAM system



Assembly

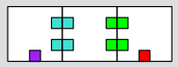
## Output

“Yes”

(  is the unique terminal assembly of the input system)

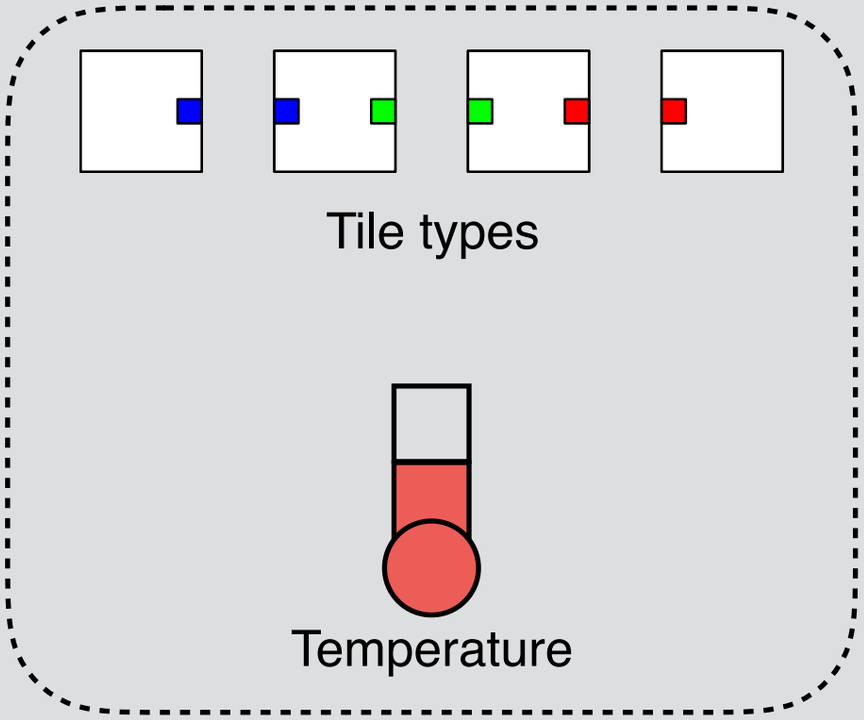
or

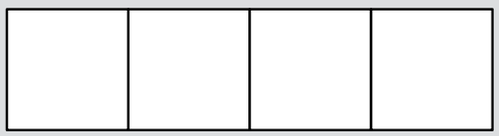
“No”

(  is not a terminal assembly or there is another terminal assembly of the input system)

# 2HAM Unique Shape Verification (USV) Problem

## Input




## Output

“Yes”

( is the shape of every terminal assembly of the input system)

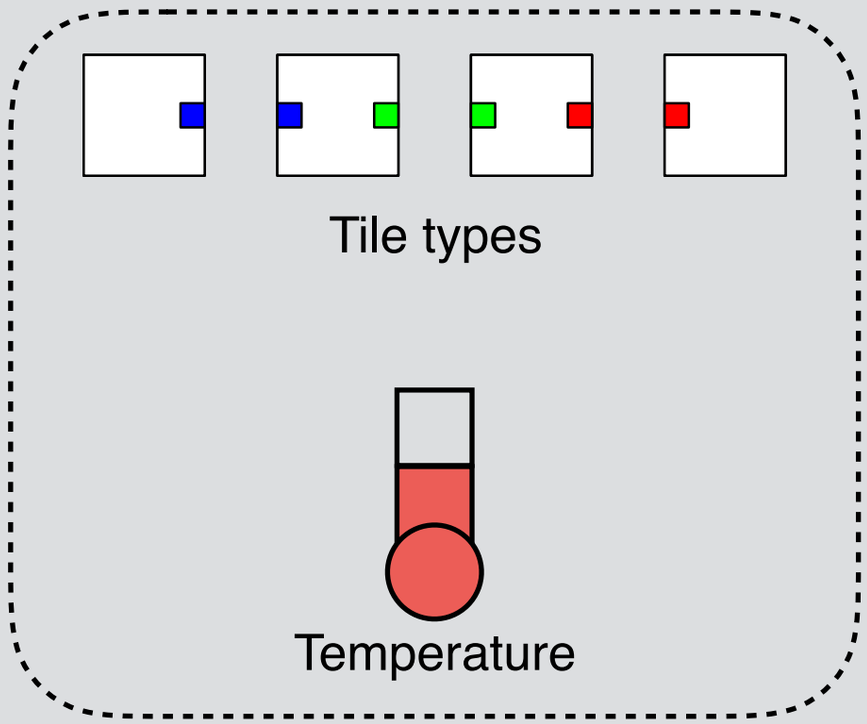
or

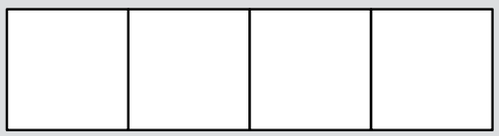
“No”

(there is a terminal assembly of the input system without shape )

# 2HAM Unique Shape Verification (USV) Problem

## Input




## Output

“Yes”

( is the shape of every terminal assembly of the input system)

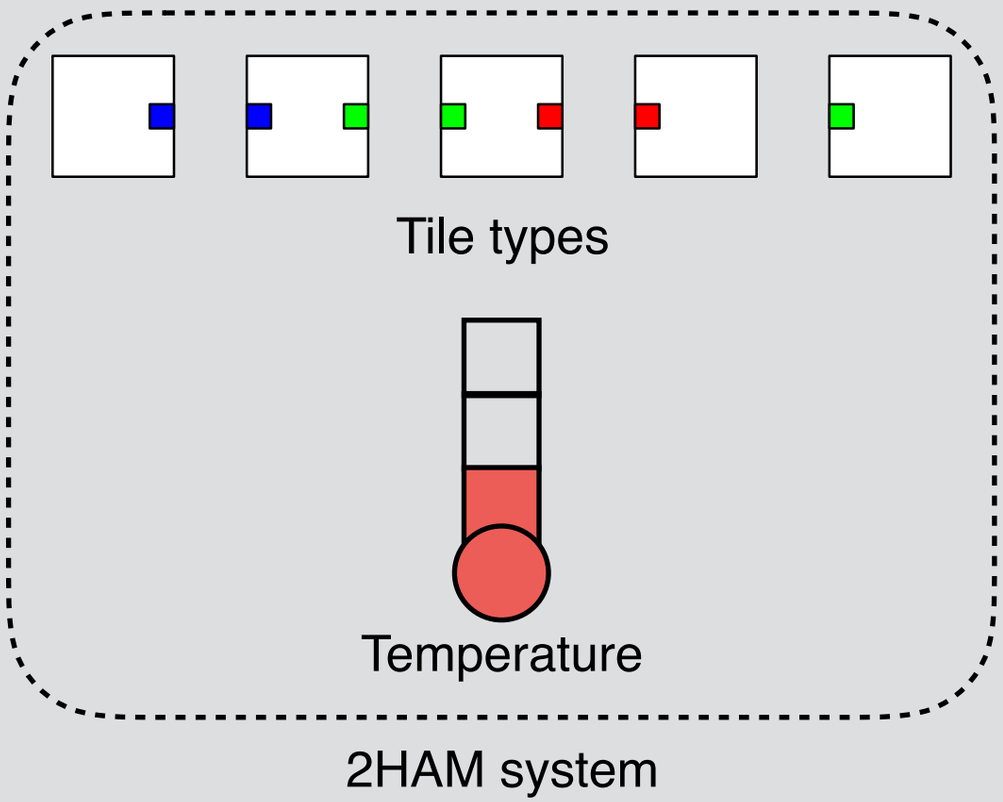
or

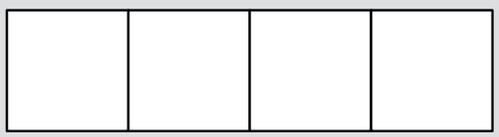
“No”

(there is a terminal assembly of the input system without shape )

# 2HAM Unique Shape Verification (USV) Problem

## Input



Shape

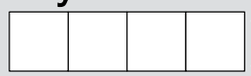
## Output

“Yes”

(  is the shape of every terminal assembly of the input system)

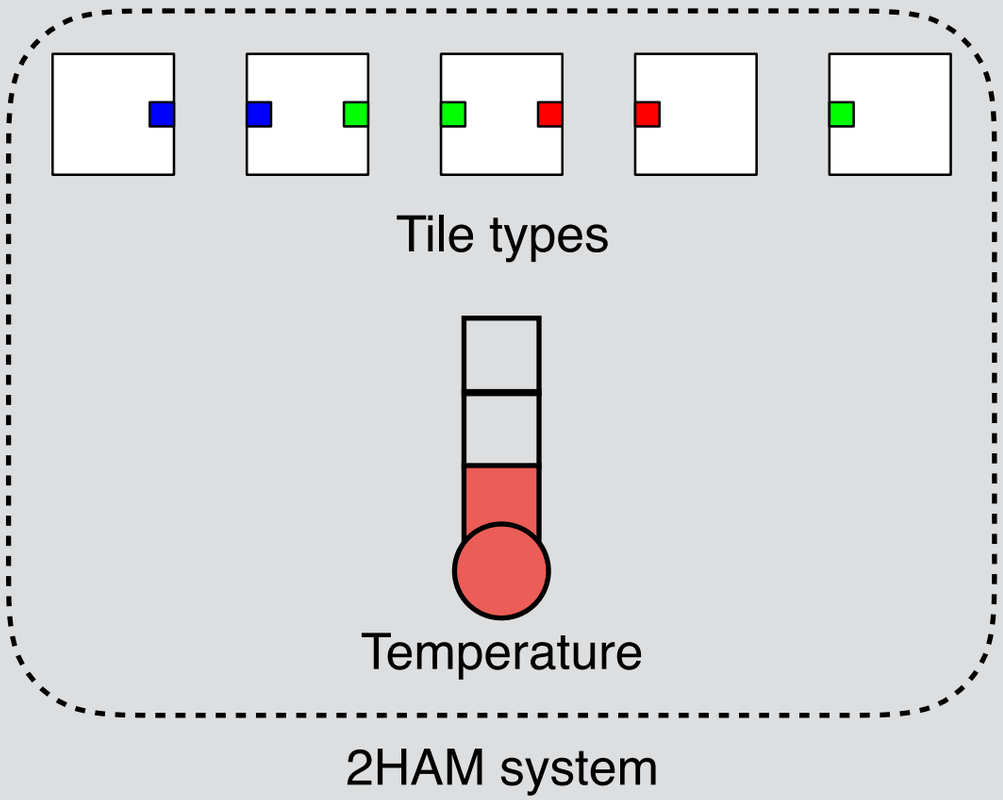
or

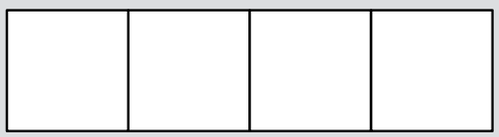
“No”

(there is a terminal assembly of the input system without shape  )

# 2HAM Unique Shape Verification (USV) Problem

## Input



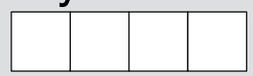

## Output

“Yes”

(  is the shape of every terminal assembly of the input system)

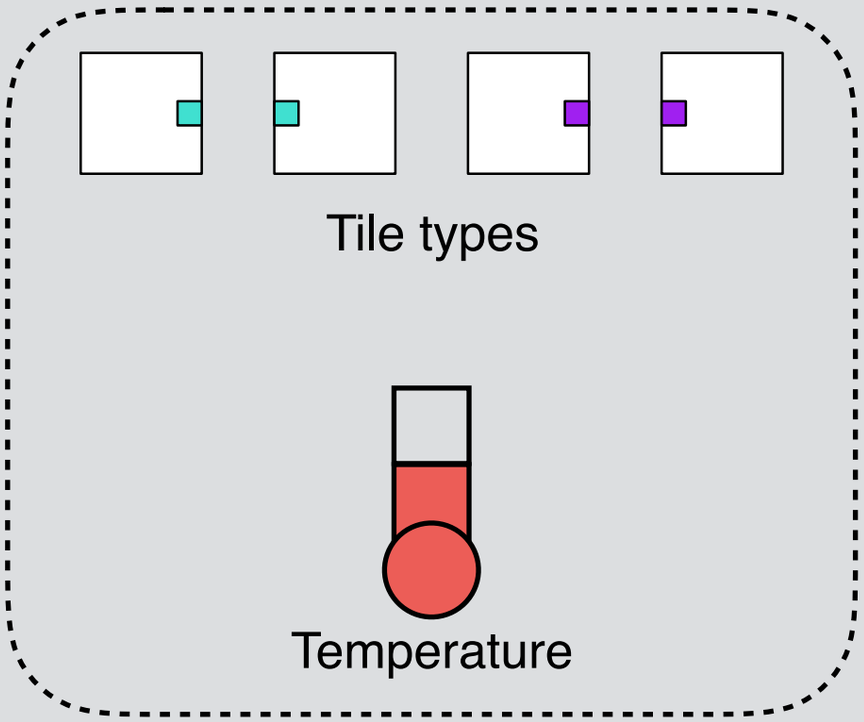
or

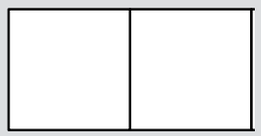
“No”

(there is a terminal assembly of the input system without shape  )

# 2HAM Unique Shape Verification (USV) Problem

## Input



Shape

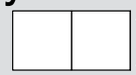
## Output

“Yes”

(  is the shape of every terminal assembly of the input system)

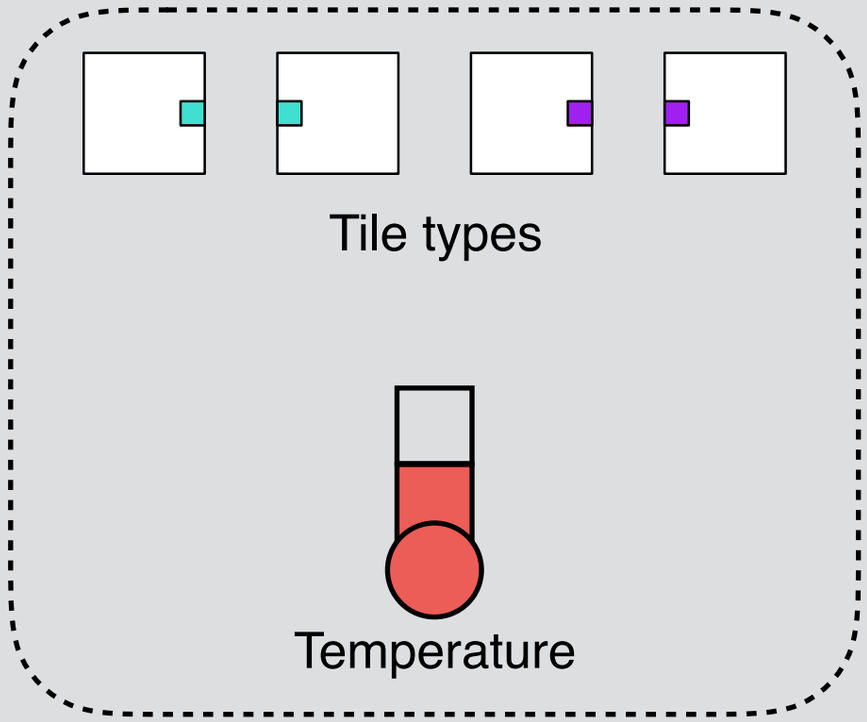
or

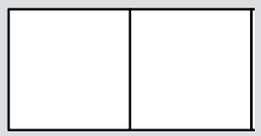
“No”

(there is a terminal assembly of the input system without shape  )

# 2HAM Unique Shape Verification (USV) Problem

## Input




## Output

“Yes”

( is the shape of every terminal assembly of the input system)

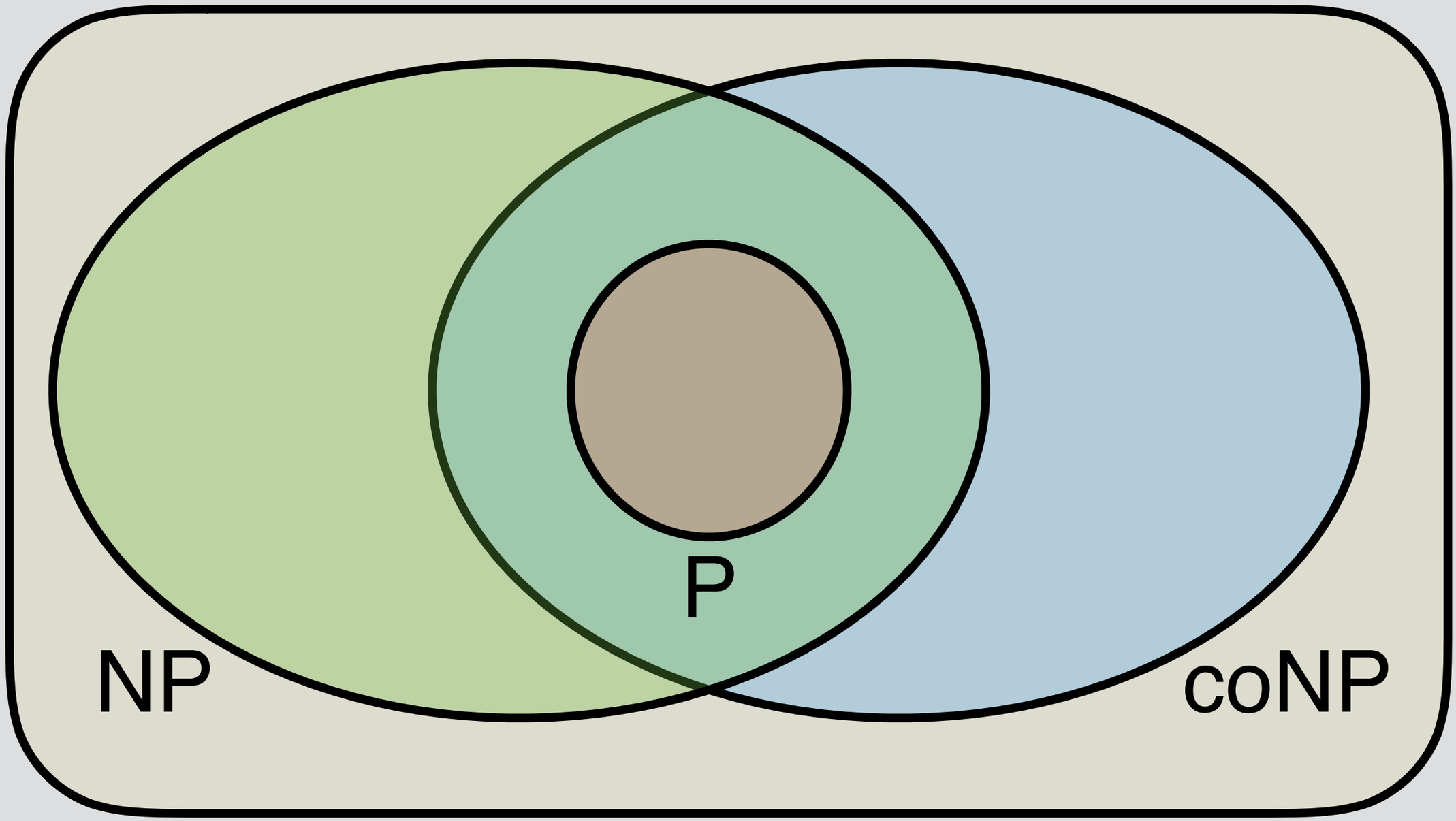
or

“No”

(there is a terminal assembly of the input system without shape )

# Our first result

# Complexity classes



## NP

"Yes" can be verified in polynomial time.

Returns "Yes" if any branch returns "Yes".

NP-complete:
does a Boolean formula have satisfying assignment?

## coNP

"No" can be verified in polynomial time.

Returns "No" if any branch returns "No".

coNP-complete:
Is every assignment of a Boolean formula satisfying?

# Our results

# Our results

1. 2HAM USV is coNP$^{NP}$-complete.

# Our results

1. 2HAM USV is coNP$^{NP}$-complete.

An *oracle*

# Our results

1. 2HAM USV is $coNP^{NP}$-complete.

An *oracle*

The coNP algorithm can make
O(1)-time calls to any NP algorithm.

# Our results

1. 2HAM USV is $coNP^{NP}$-complete.

An *oracle*

The coNP algorithm can make
O(1)-time calls to any NP algorithm.

Already known: 2HAM UAV is coNP-complete.
[Unpublished]

# 2HAM USV is $coNP^{NP}$-hard

# Reduction is from $\forall\exists$SAT:

## Input:

A boolean formula with variables

$x_1, x_2, \ldots, x_j, y_1, y_2, \ldots, y_k$

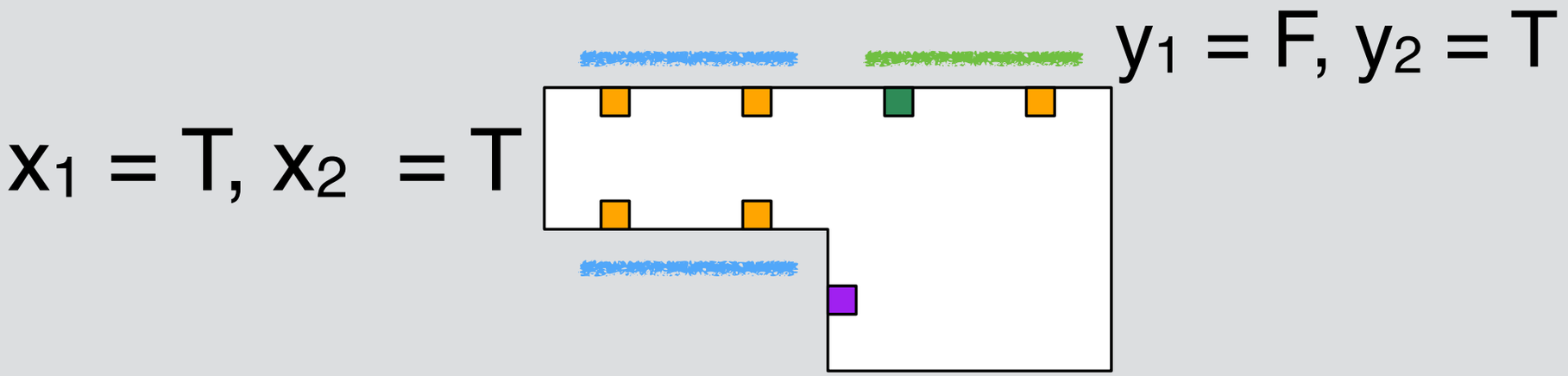
## Output:

Does every assignment of $x_1, x_2, \ldots, x_j$
have an assignment of $y_1, y_2, \ldots, y_k$
that satisfies the formula?

$\forall\exists$SAT proved $coNP^{NP}$-hard by [Stockmeyer 1975]

# 2HAM USV is coNP$^{NP}$-hard

(The assemblies involved in the reduction)

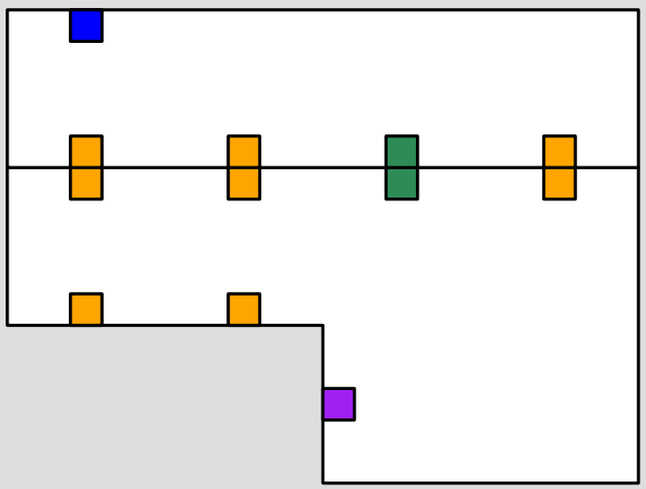



Assignment assembly

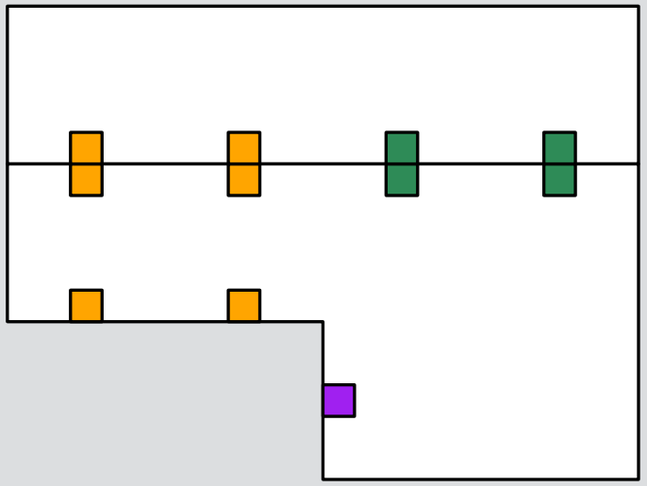
# 2HAM USV is coNP$^{NP}$-hard

(The assemblies involved in the reduction)

Satisfying assignment

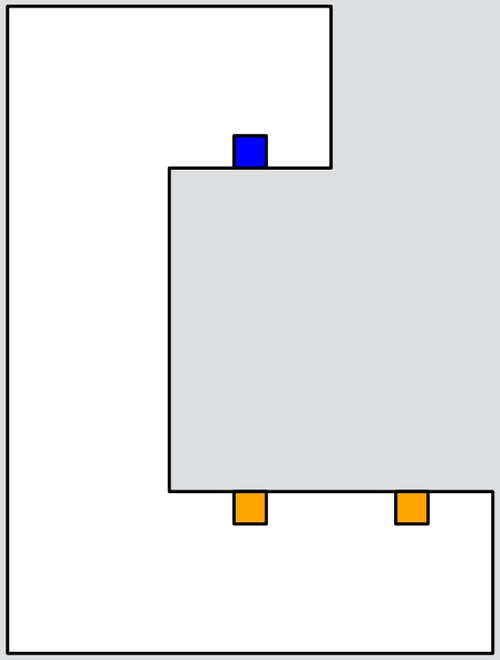


~~Satisfying assignment~~

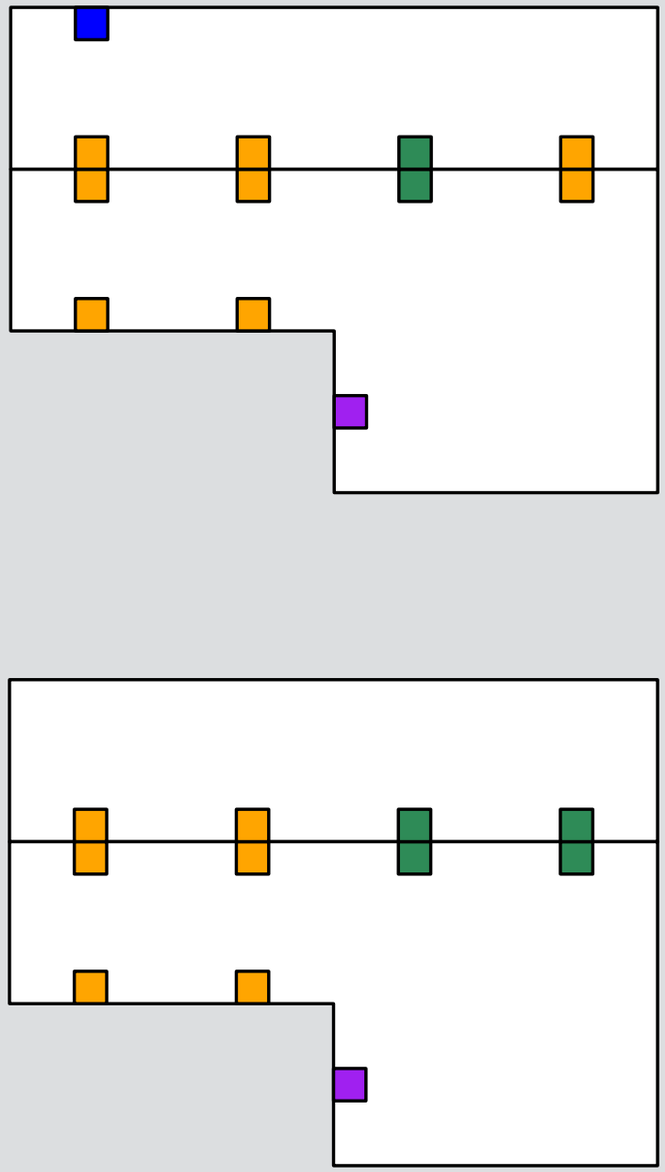


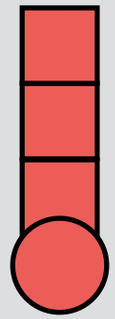
# 2HAM USV is coNP$^{NP}$-hard

(The assemblies involved in the reduction)



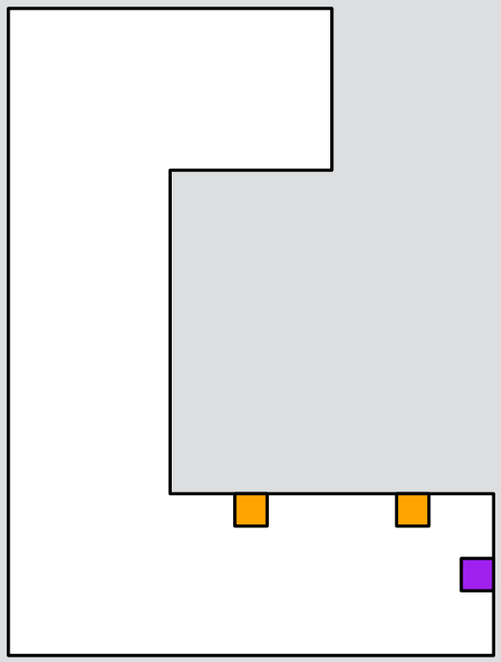
Checker assembly



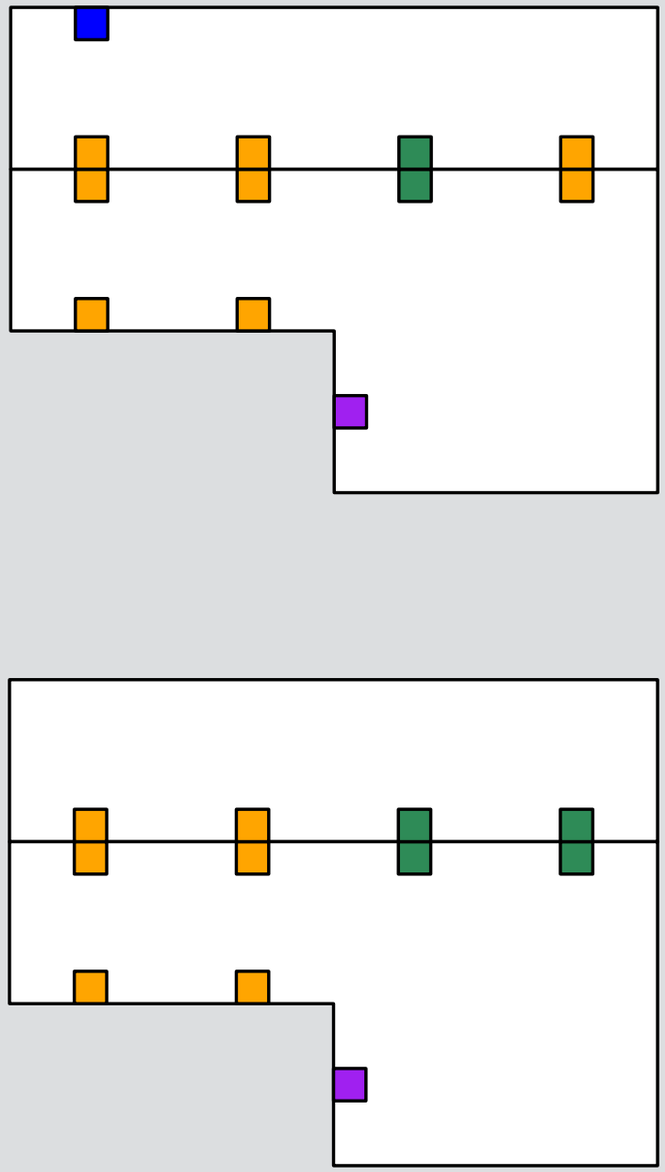


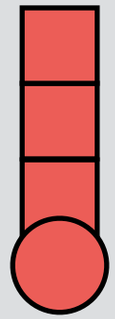
# 2HAM USV is coNP$^{NP}$-hard

(The assemblies involved in the reduction)



Filler assembly

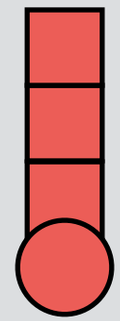




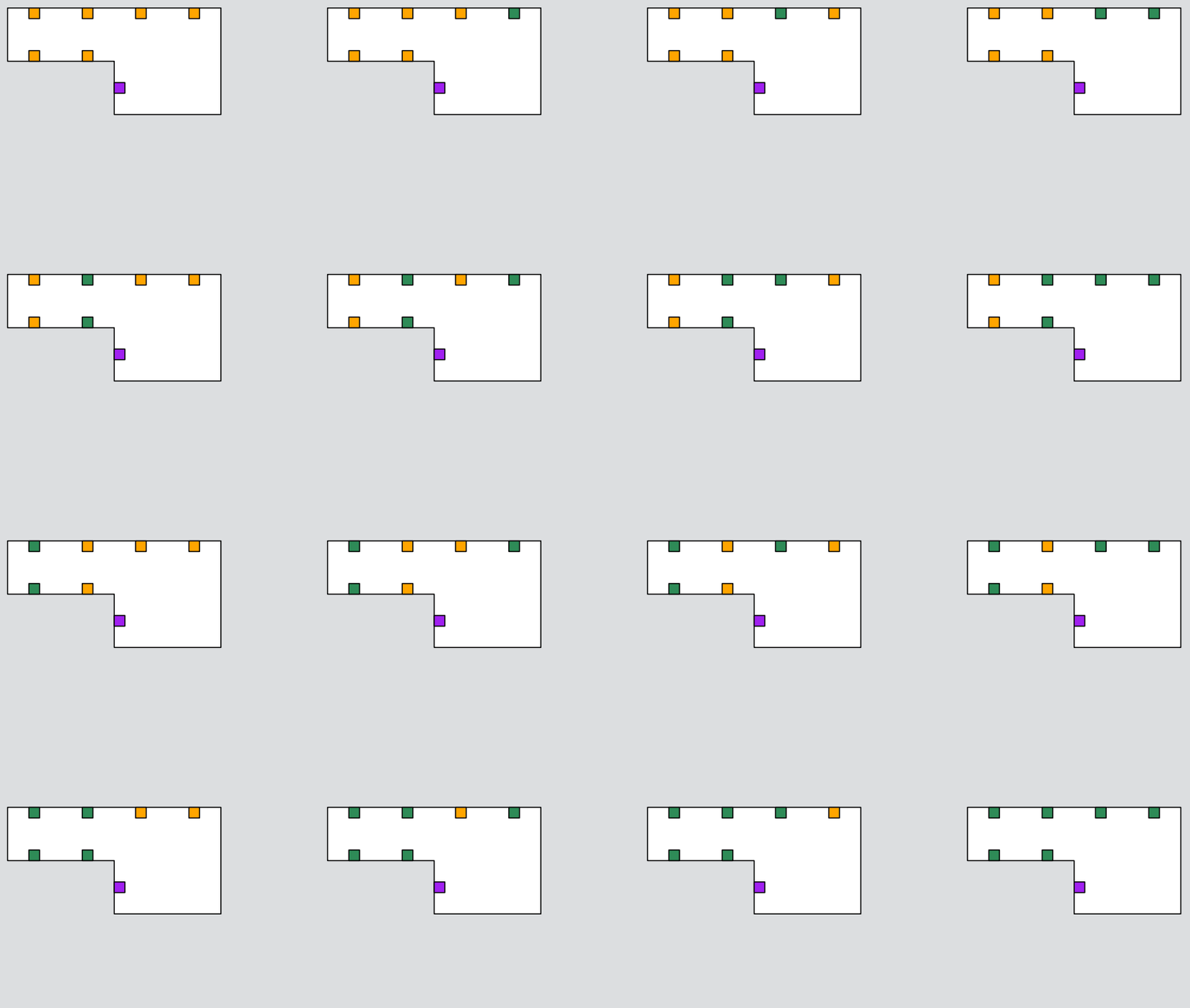
Does every assignment of $x_1$, $x_2$ have an assignment of $y_1$, $y_2$, that satisfies the formula?

$y_1$, $y_2$ assignments

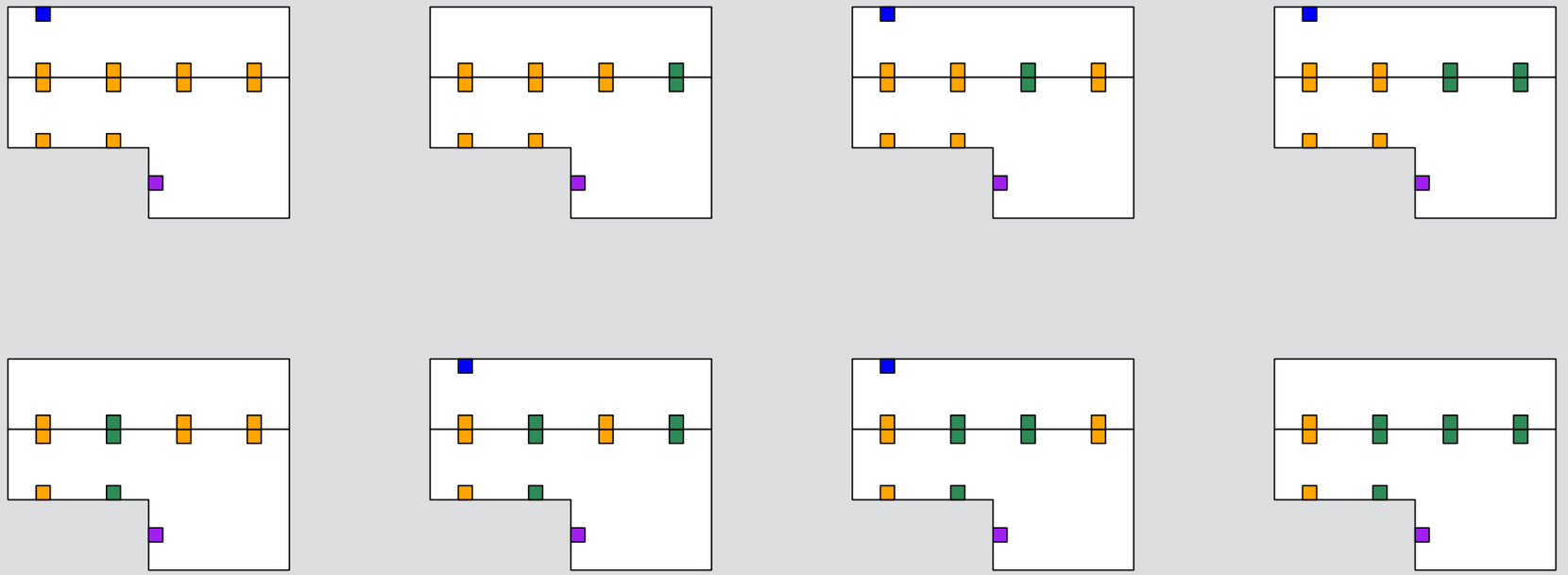
$x_1$, $x_2$ assignments

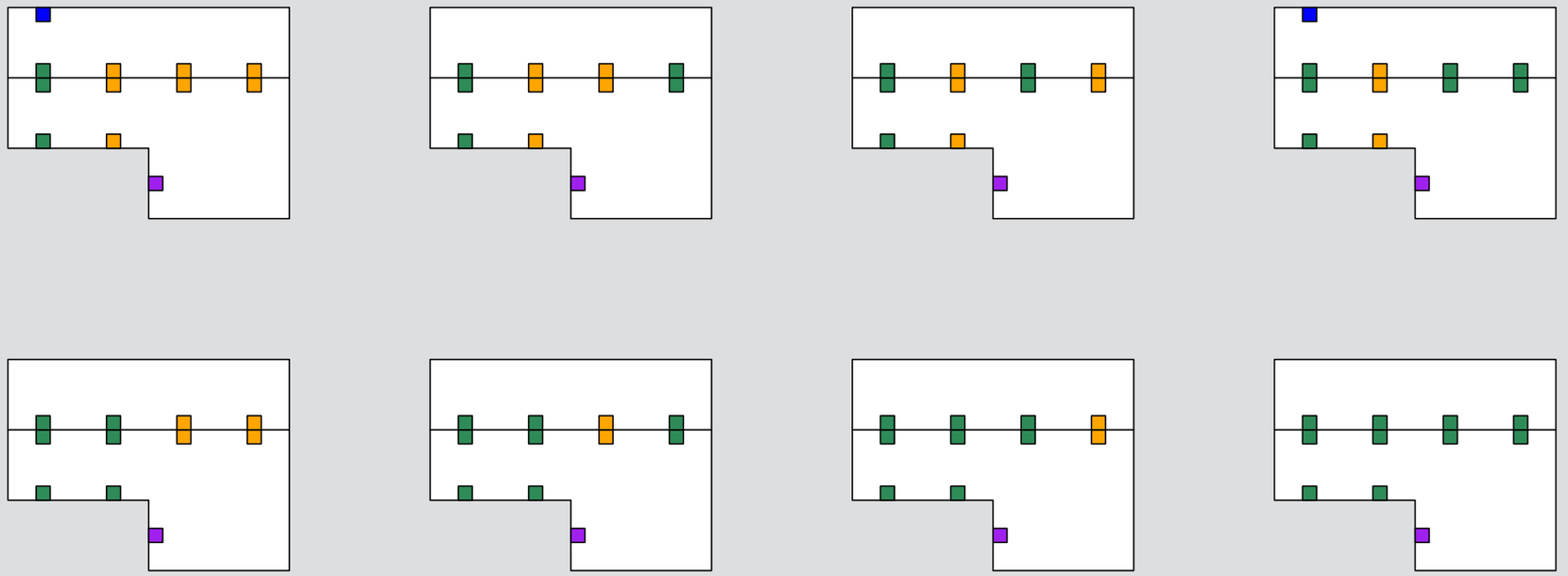


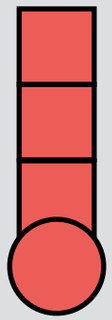
Does every assignment of $x_1$, $x_2$ have
an assignment of $y_1$, $y_2$, that satisfies the formula?



Does every assignment of $x_1$, $x_2$ have
an assignment of $y_1$, $y_2$, that satisfies the formula?

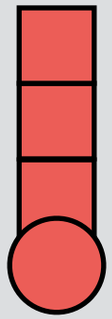




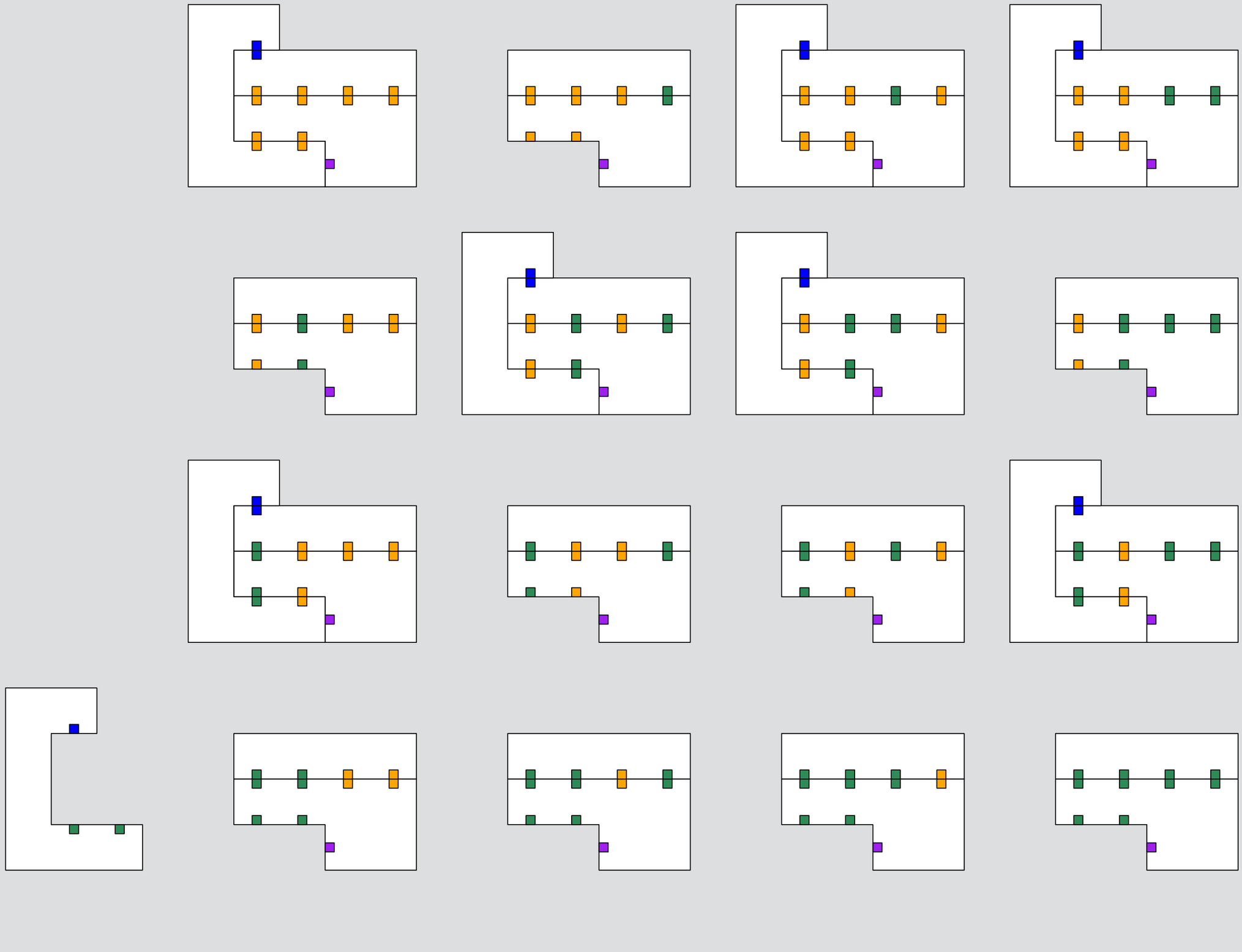


# Does every assignment of $x_1$, $x_2$ have an assignment of $y_1$, $y_2$, that satisfies the formula?

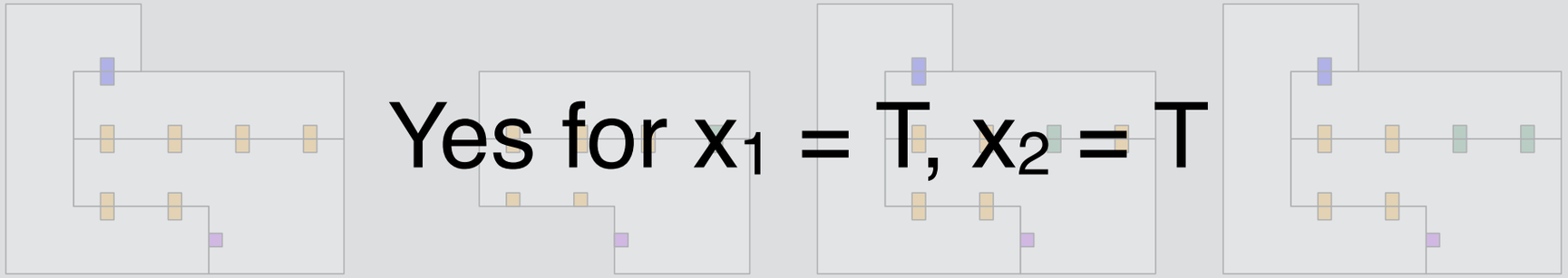
Checker assembly



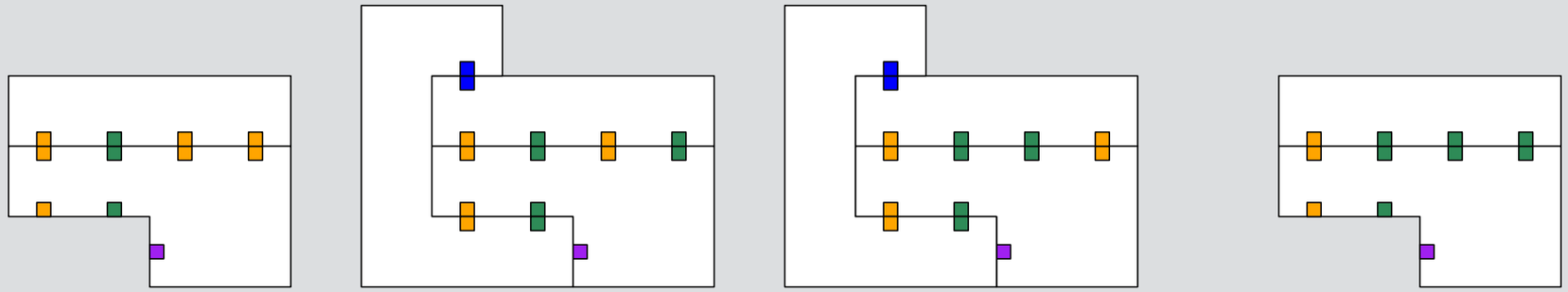
# Does every assignment of $x_1$, $x_2$ have an assignment of $y_1$, $y_2$, that satisfies the formula?

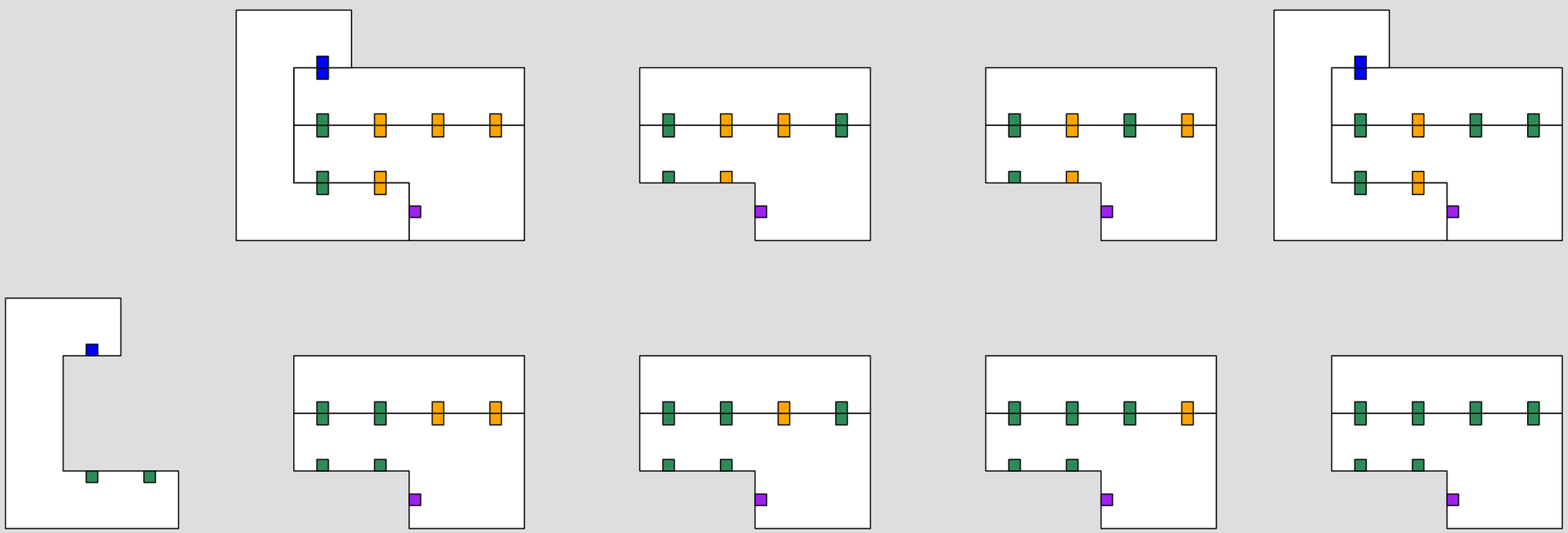


Does every assignment of $x_1$, $x_2$ have an assignment of $y_1$, $y_2$, that satisfies the formula?

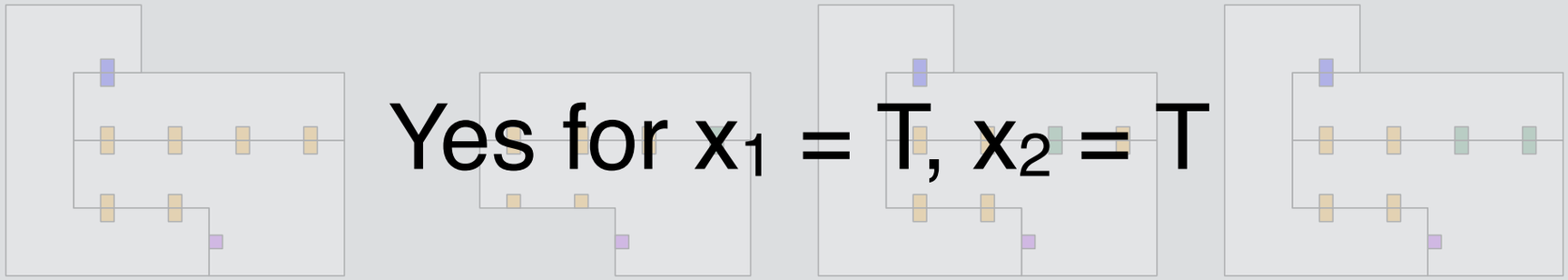


Yes for $x_1 = T$, $x_2 = T$

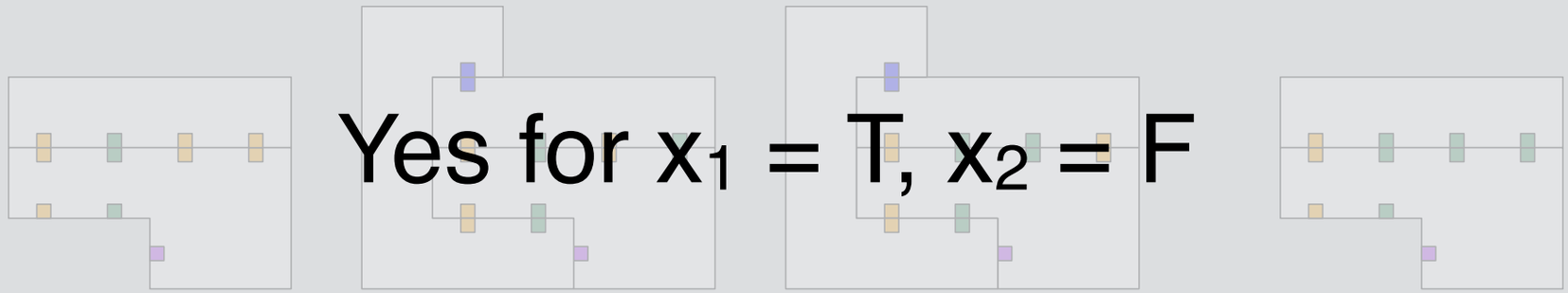




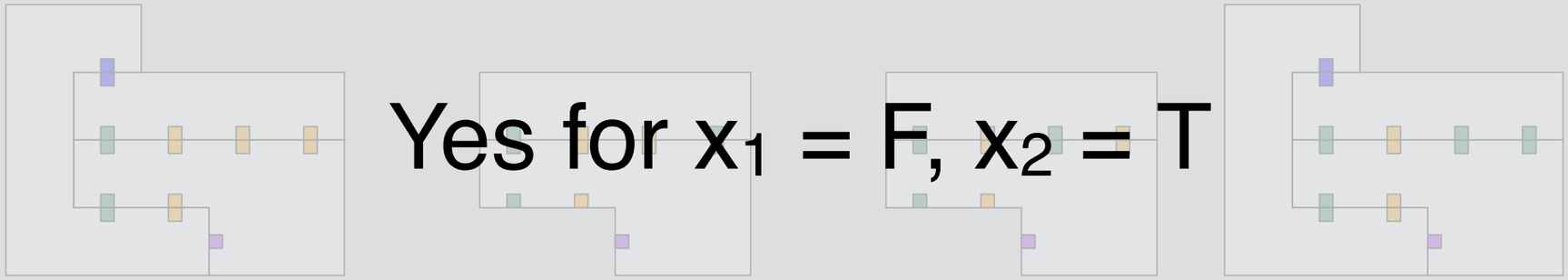
Does every assignment of $x_1$, $x_2$ have an assignment of $y_1$, $y_2$, that satisfies the formula?



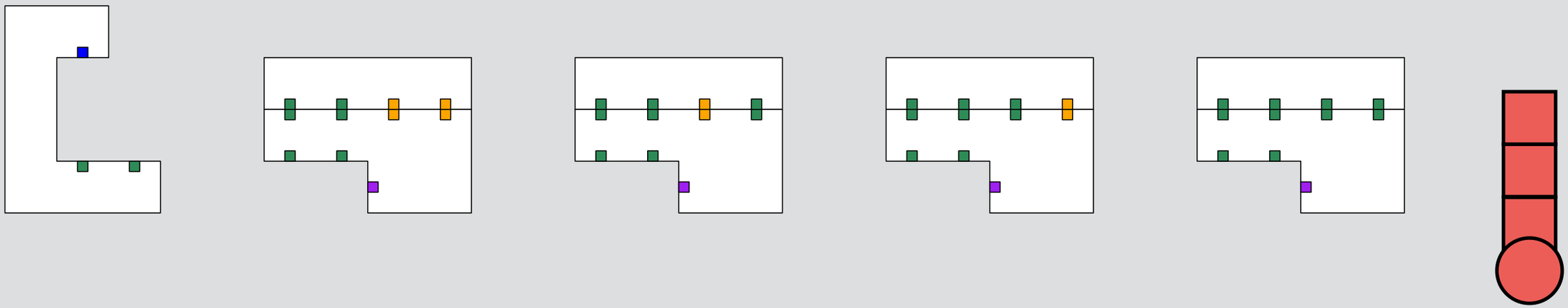
Yes for $x_1 = T$, $x_2 = T$



Yes for $x_1 = T$, $x_2 = F$



Yes for $x_1 = F$, $x_2 = T$



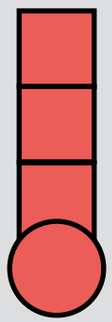
Does every assignment of $x_1$, $x_2$ have an assignment of $y_1$, $y_2$, that satisfies the formula?

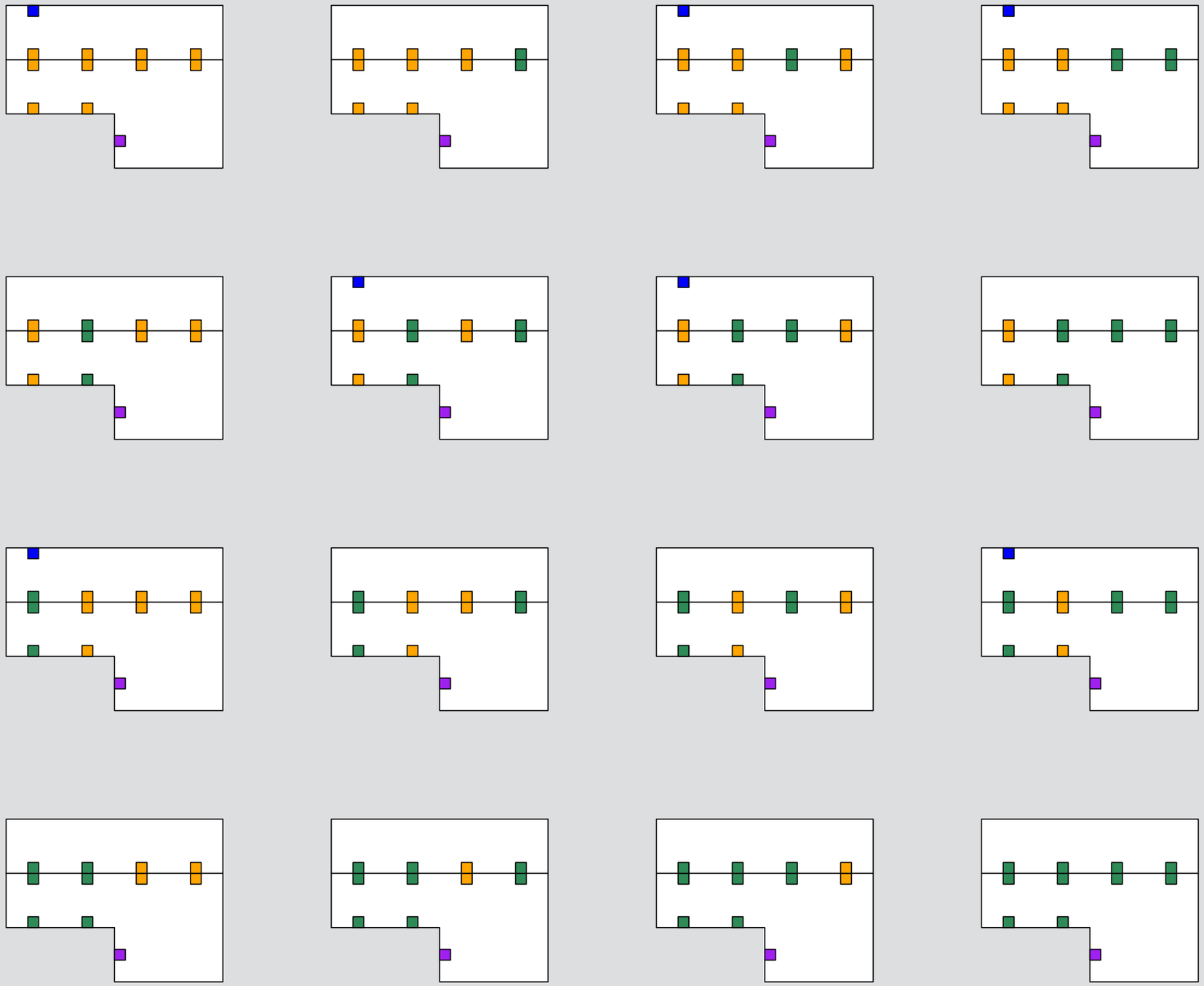
Yes for $x_1 = T$, $x_2 = T$

Yes for $x_1 = T$, $x_2 = F$

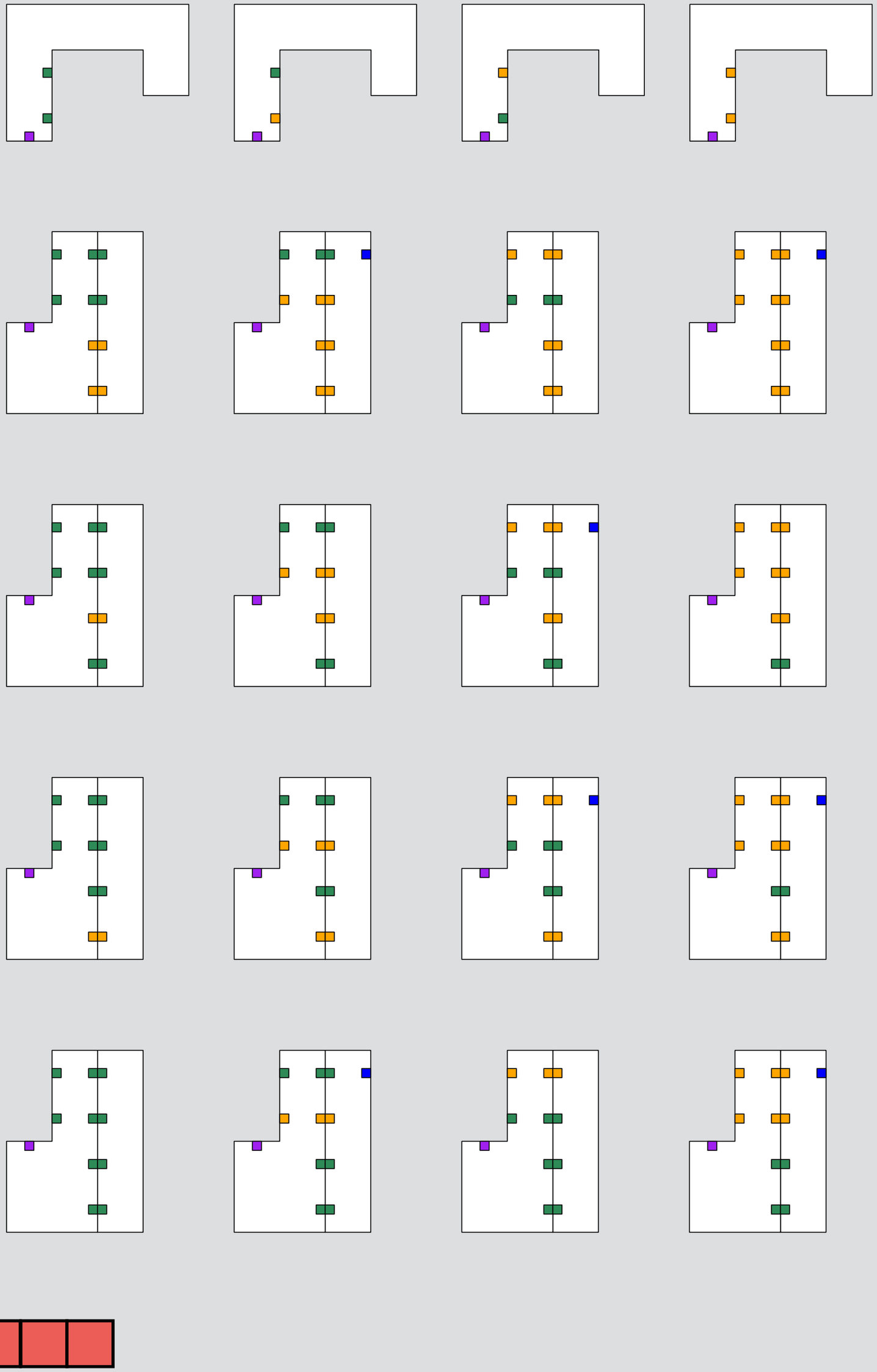
Yes for $x_1 = F$, $x_2 = T$

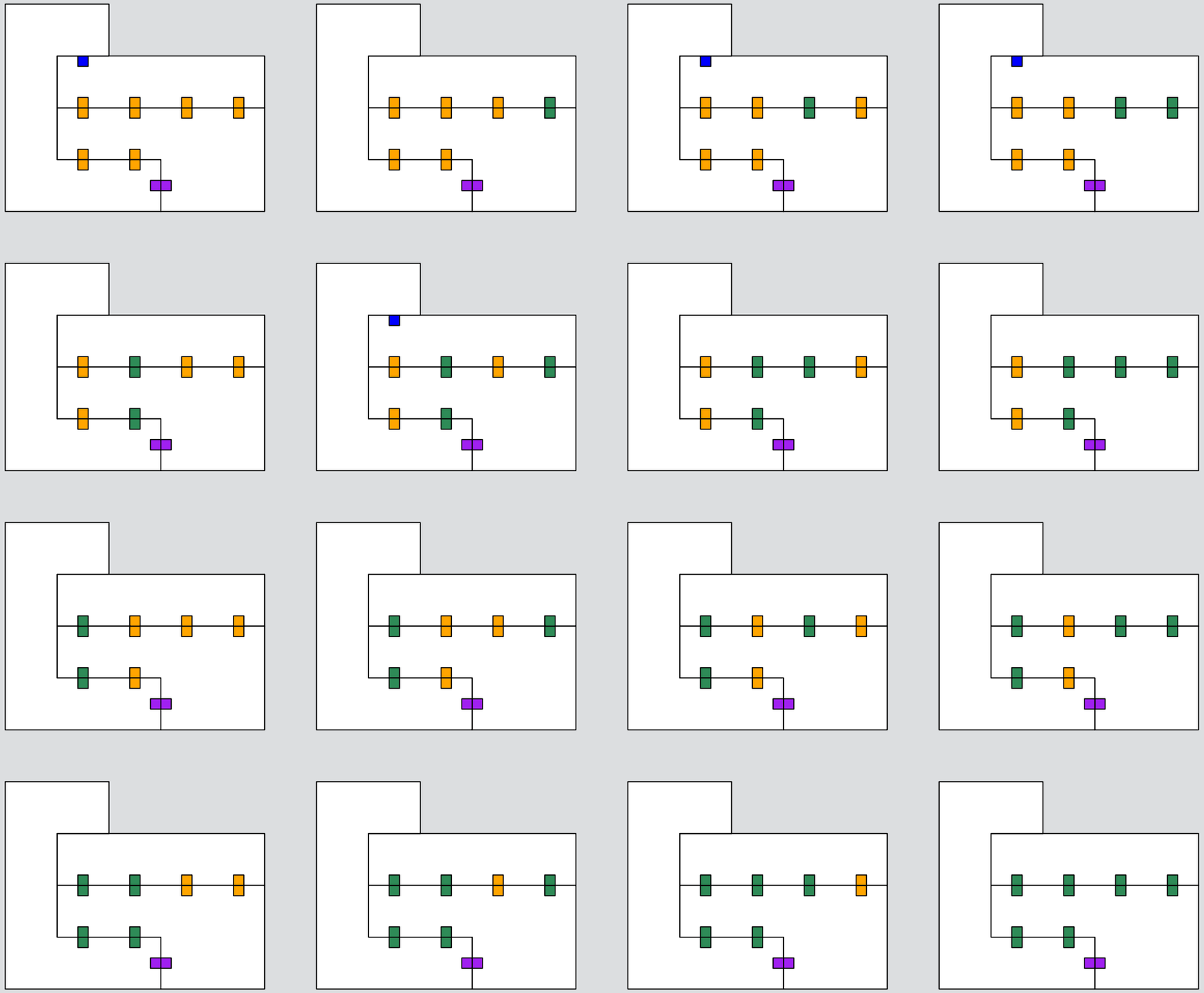
No for $x_1 = F$, $x_2 = F$





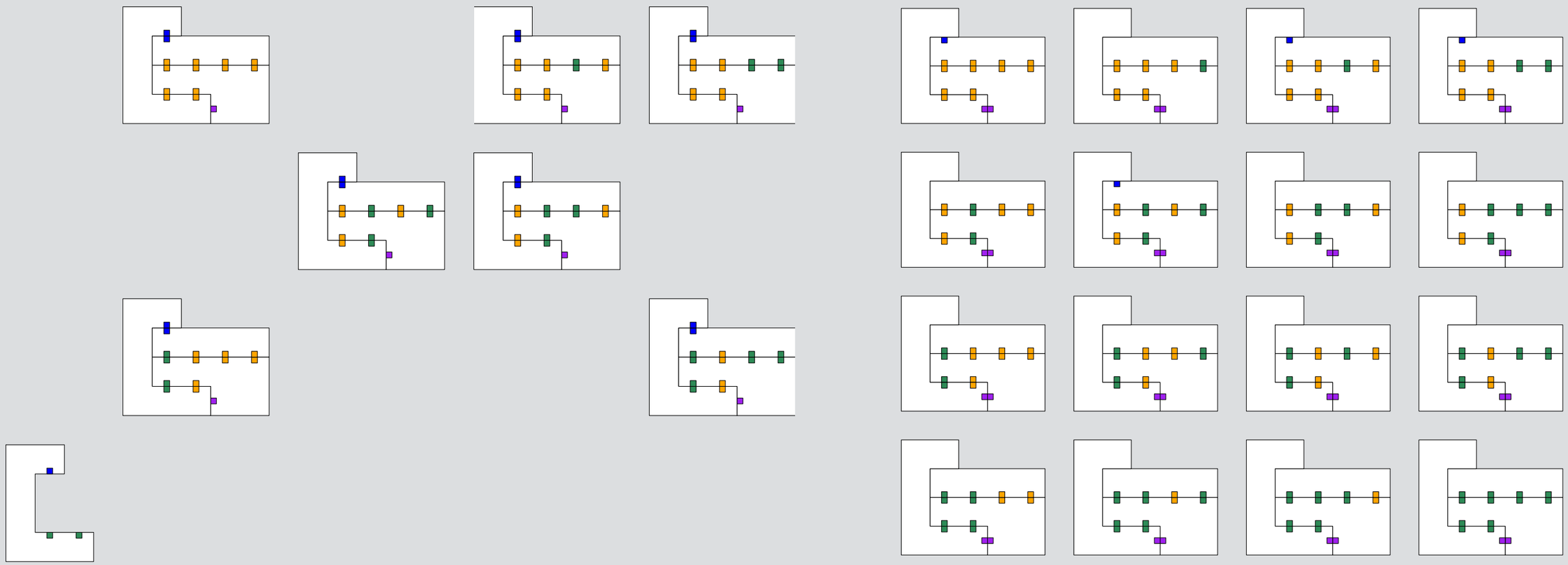
# Filler assembly





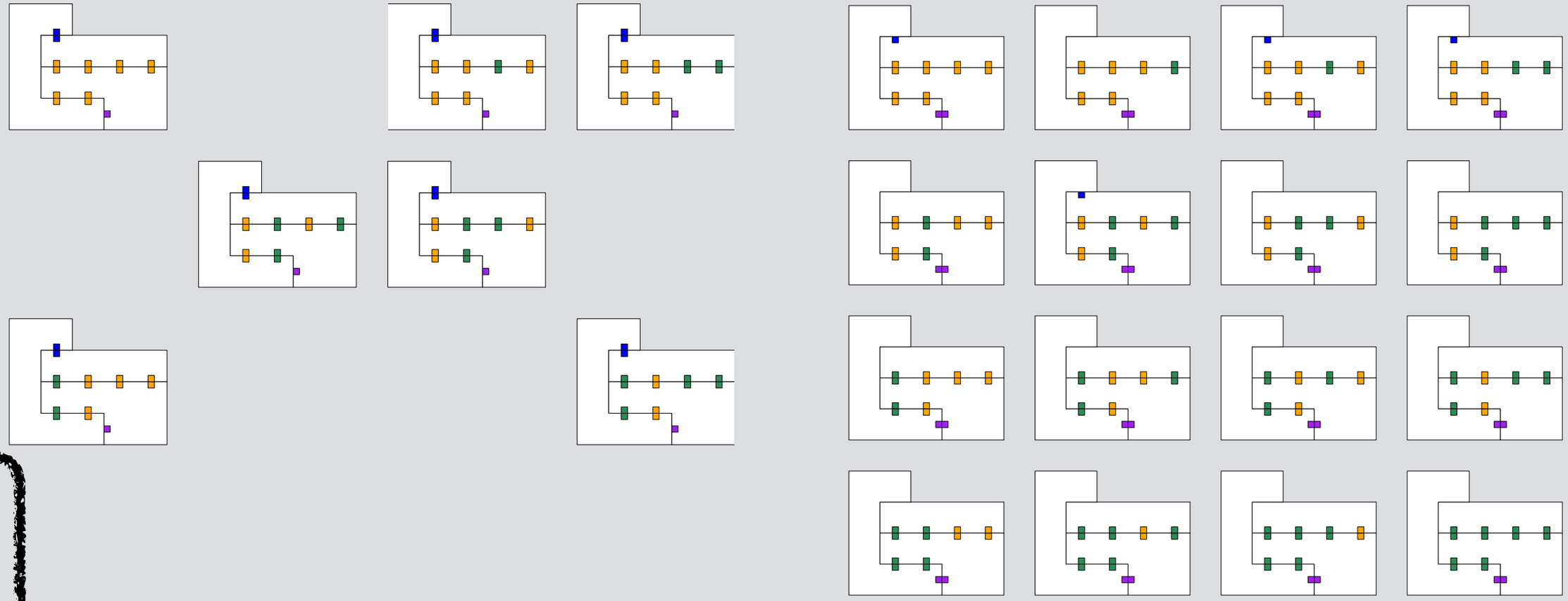
# Reduction from to ∀∃SAT to 2HAM USV

## Terminal assemblies



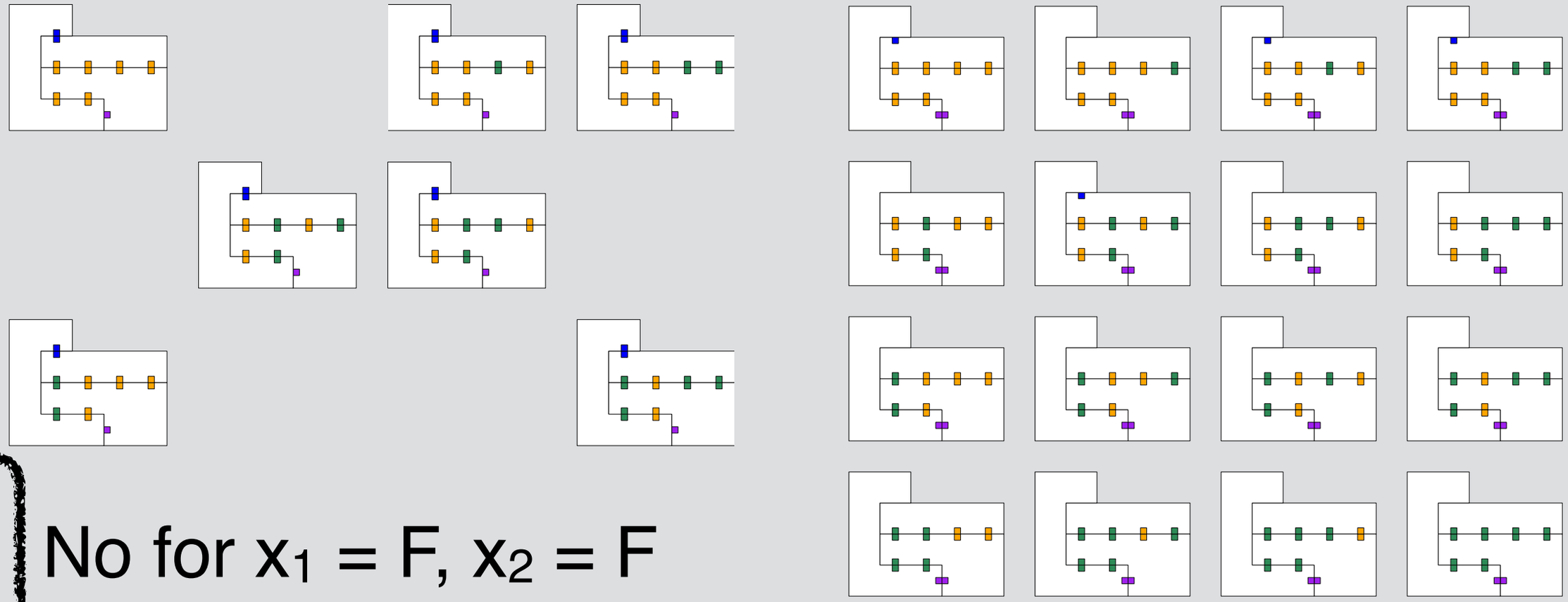
# Reduction from to ∀∃SAT to 2HAM USV

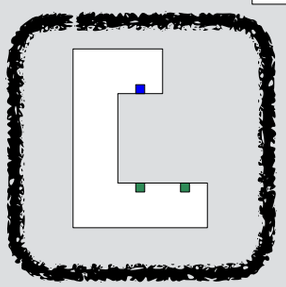
## Terminal assemblies



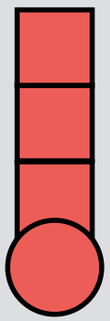
# Reduction from to ∀∃SAT to 2HAM USV

## Terminal assemblies



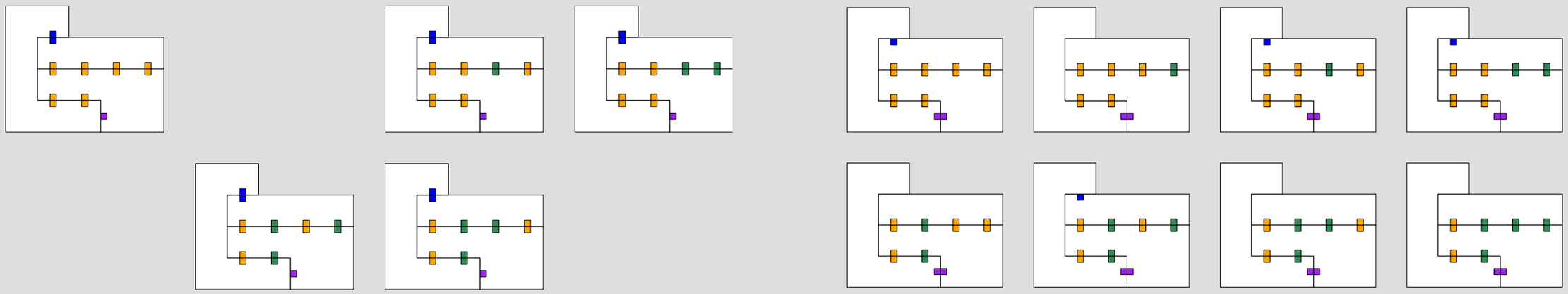


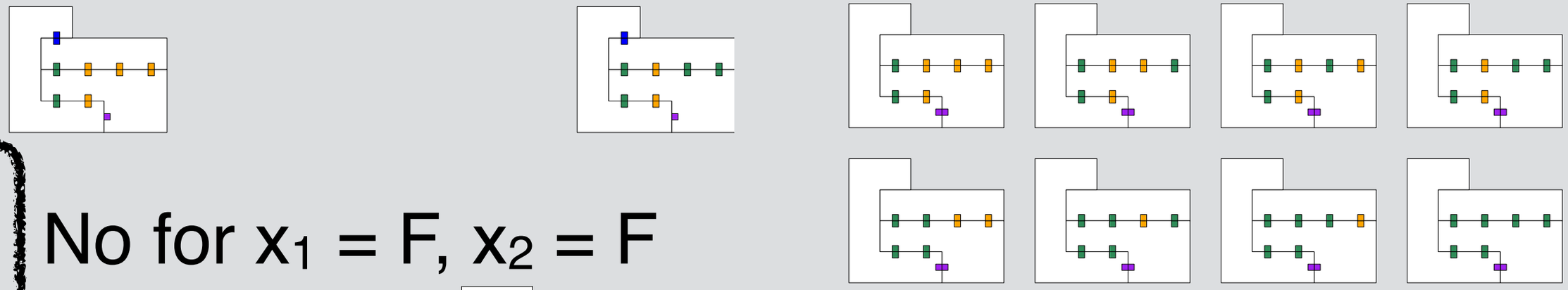
No for $x_1$ = F, $x_2$ = F



# Reduction from to ∀∃SAT to 2HAM USV

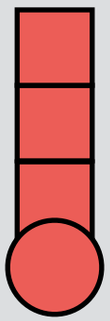
## Terminal assemblies



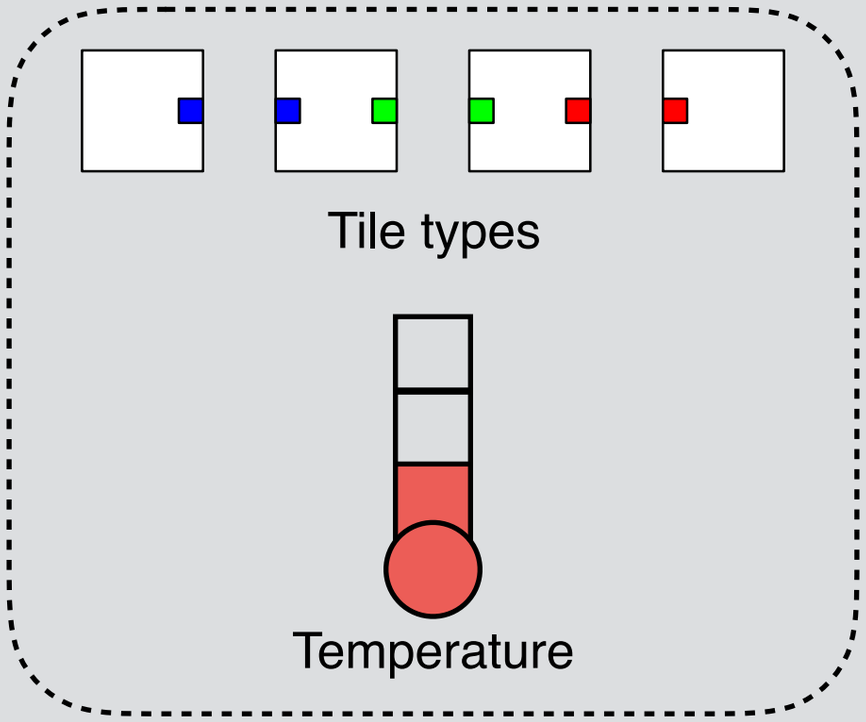


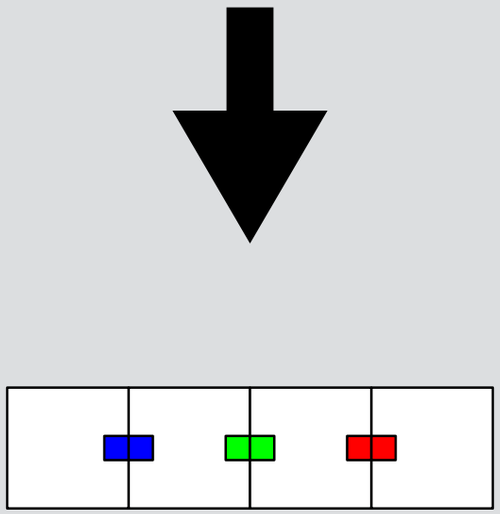
No for $x_1$ = F, $x_2$ = F

Shape is not  ➡ USV answer is “No”.

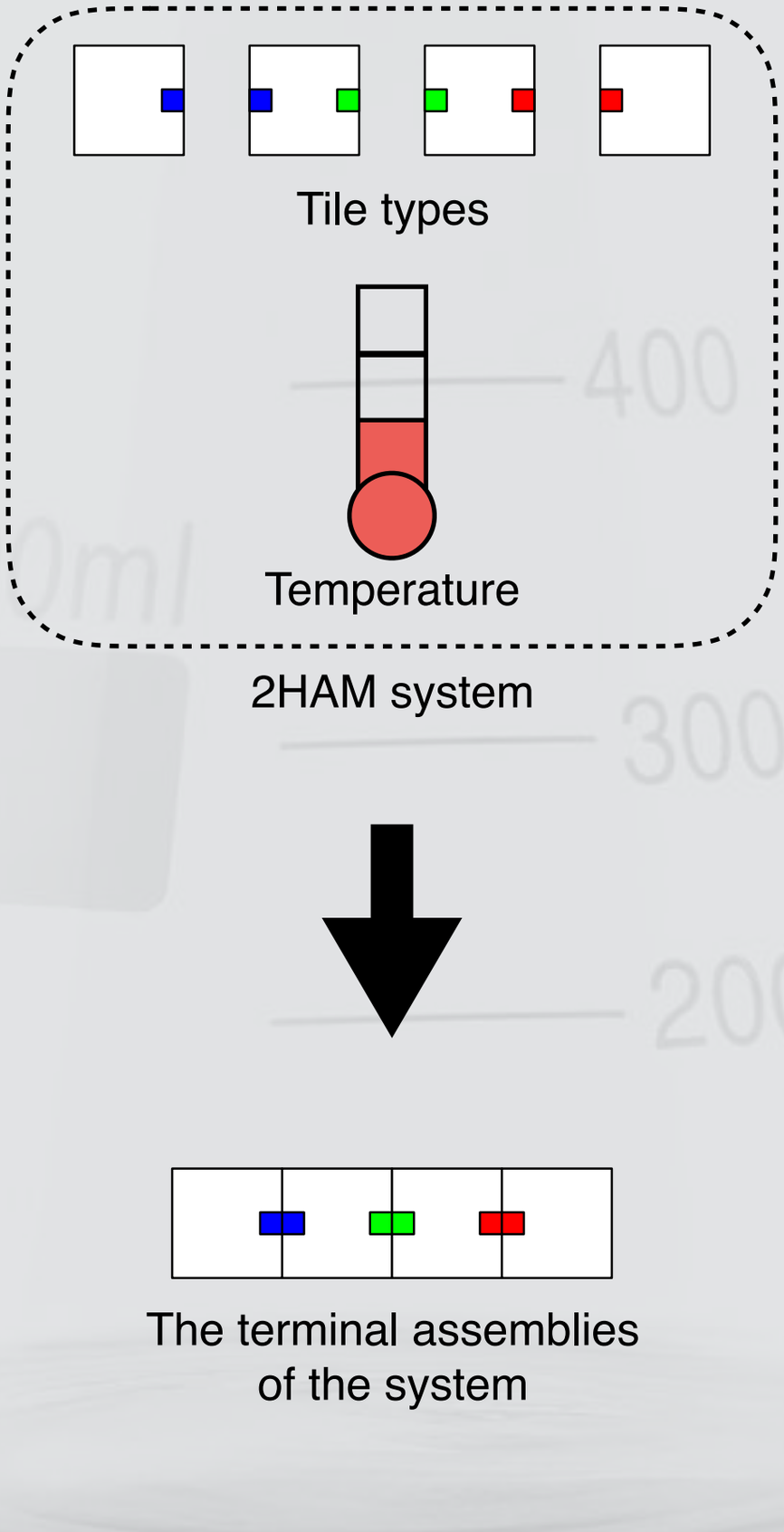


# Staged tile self-assembly model



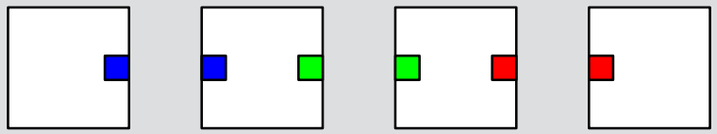



The terminal assemblies
of the system

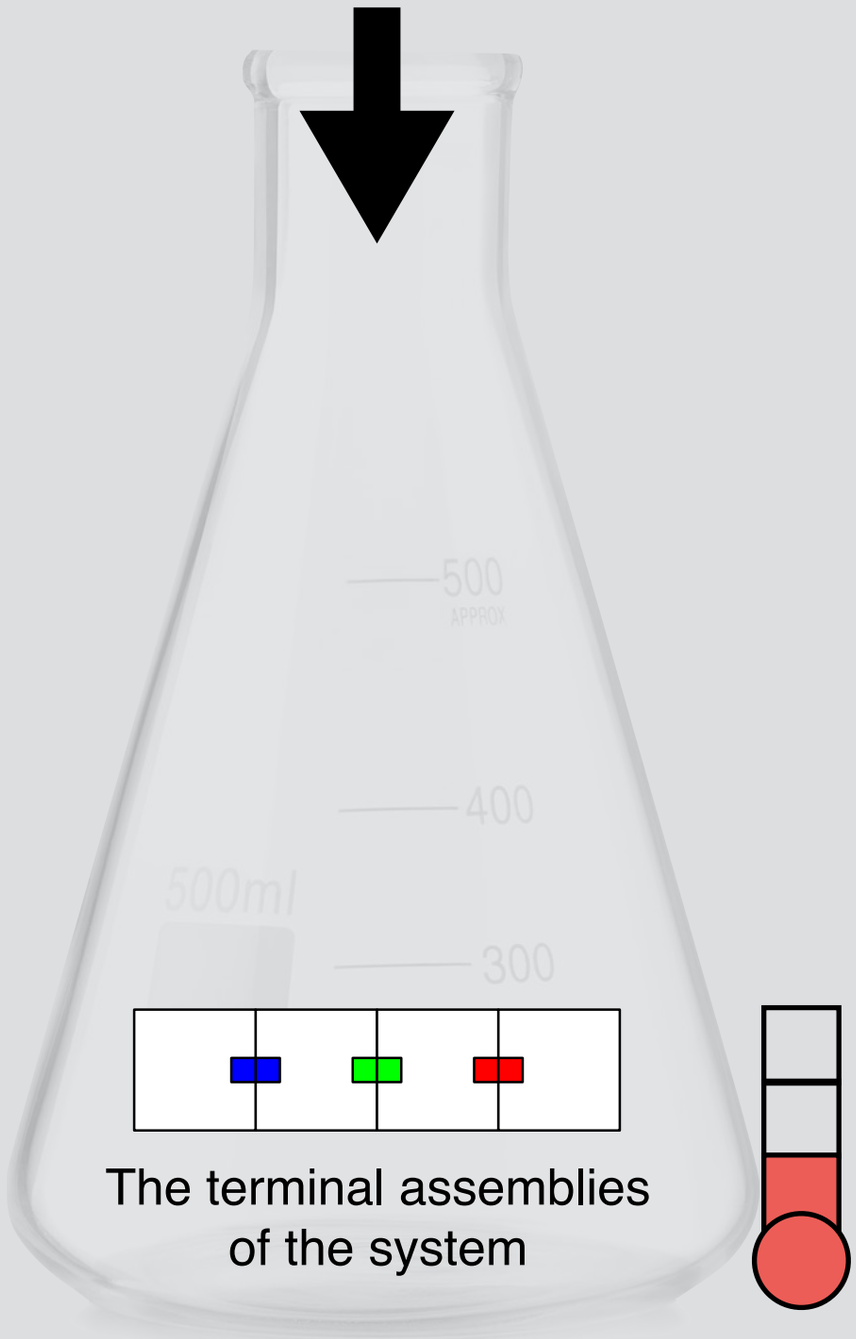





The terminal assemblies
of the system

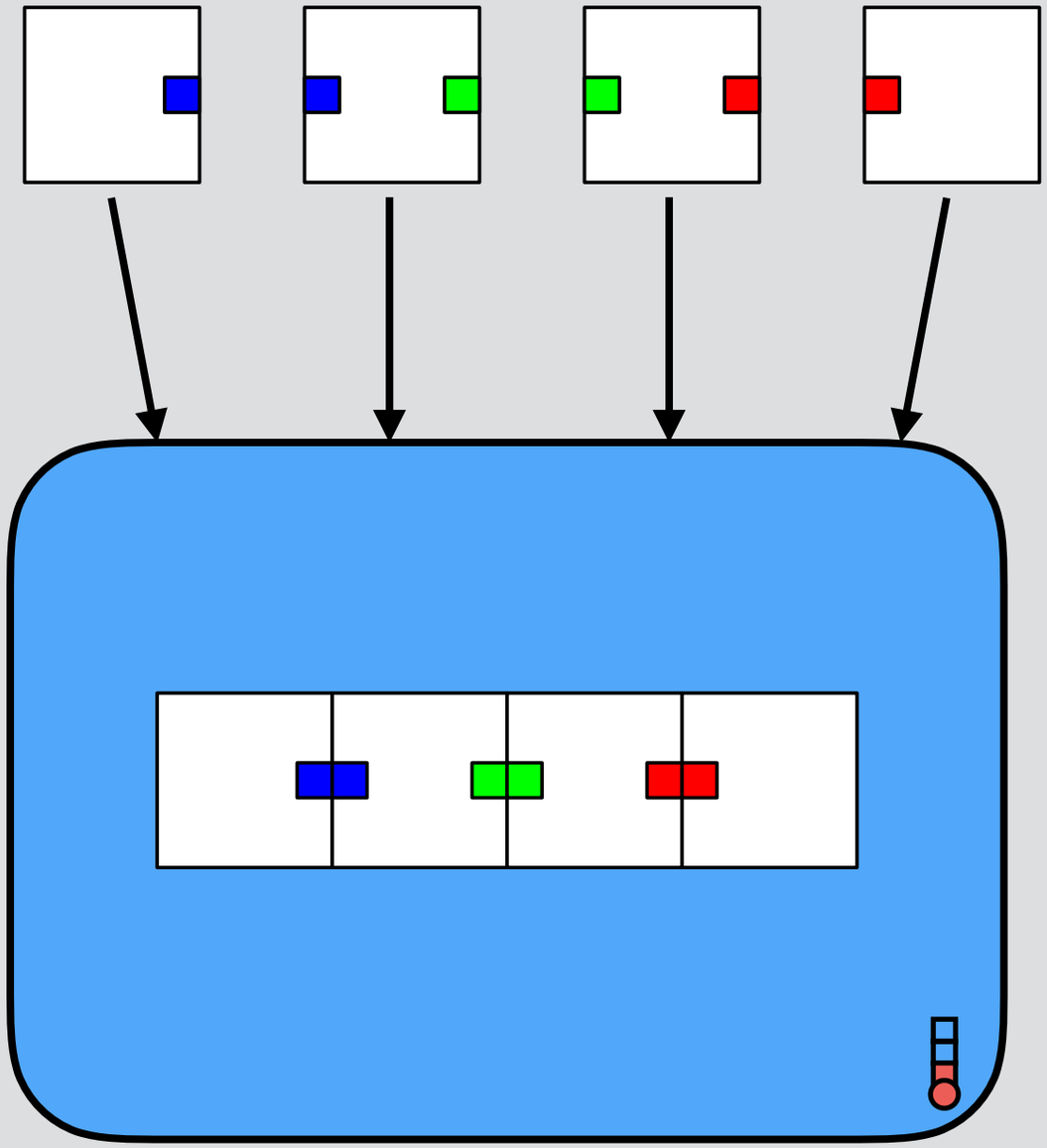


Tile types



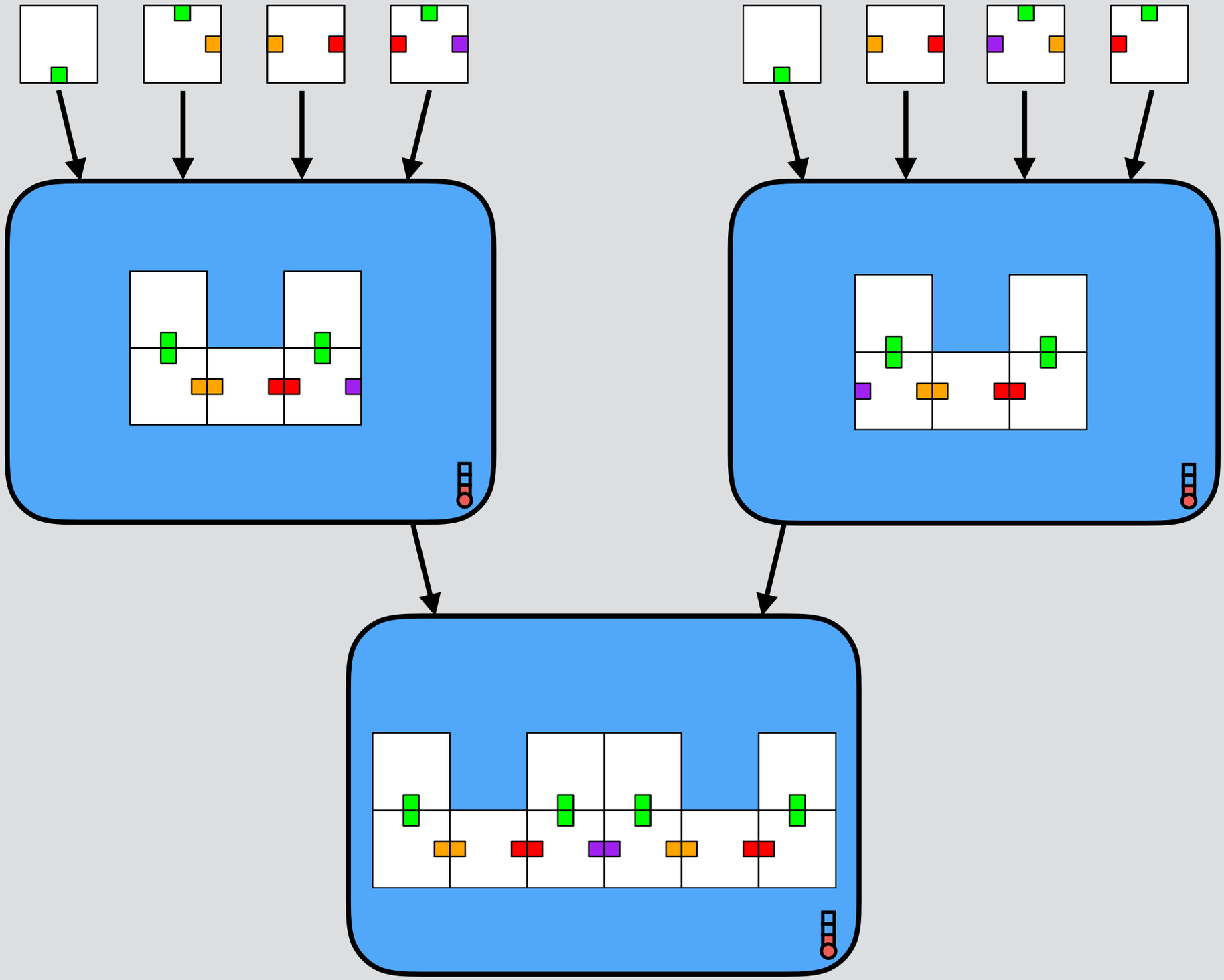



The terminal assemblies
of the system



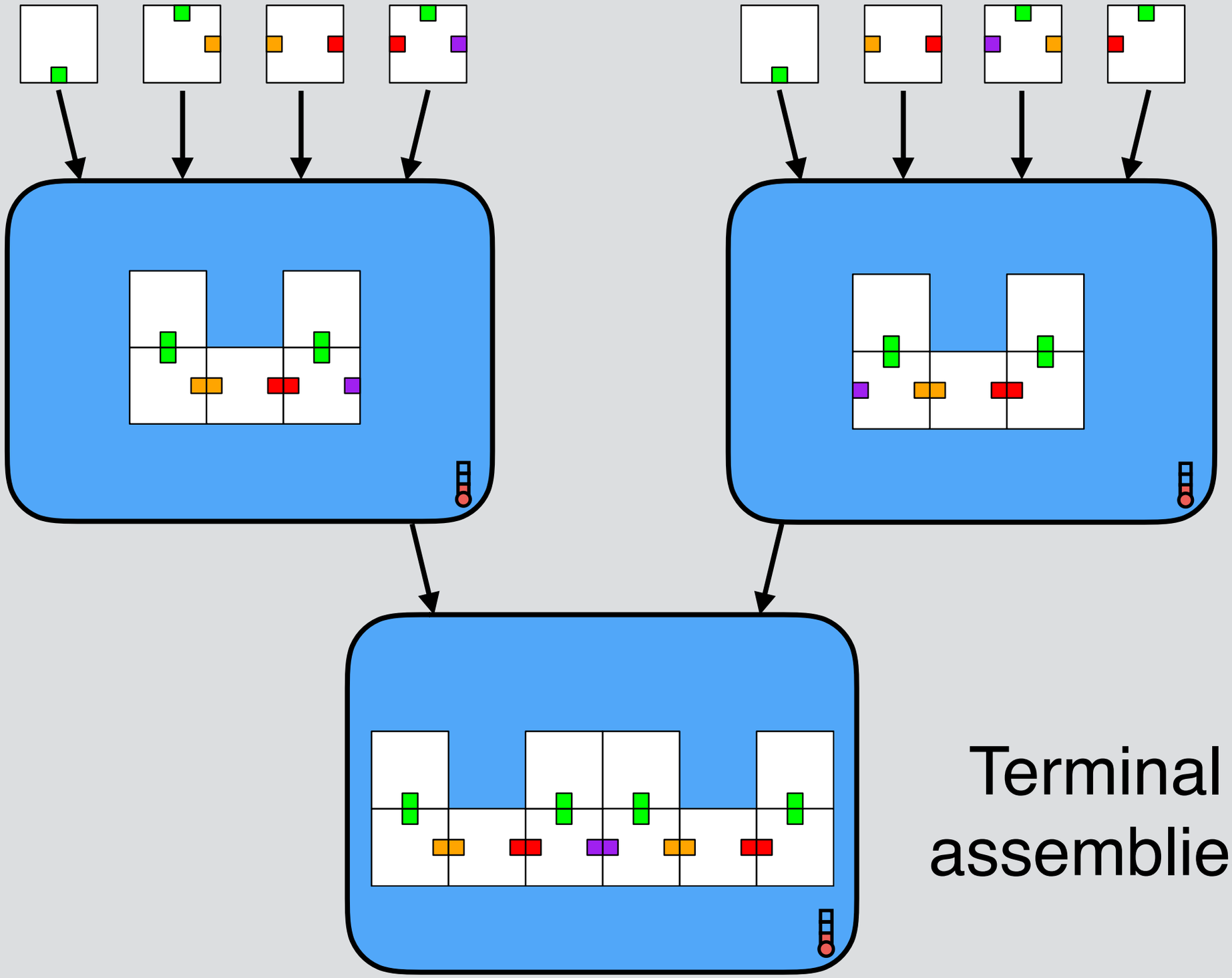
# Staged tile assembly model

[Demaine et al. 2008]



# Staged tile assembly model

[Demaine et al. 2008]

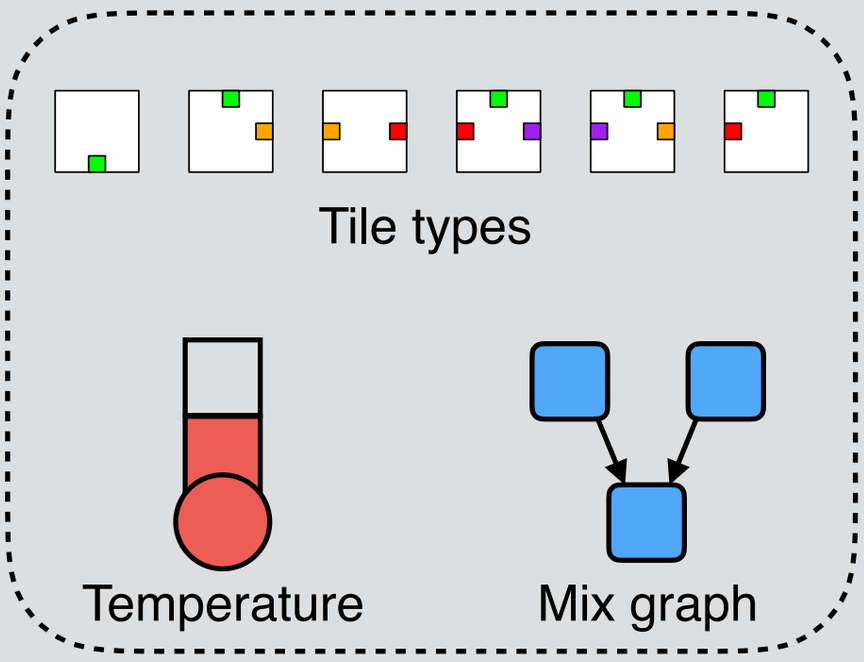


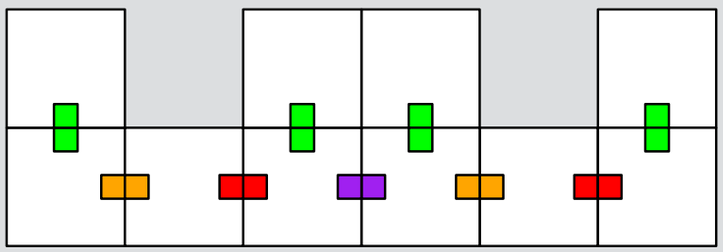
Terminal assemblies

Our second and third results

# Staged UAV Problem

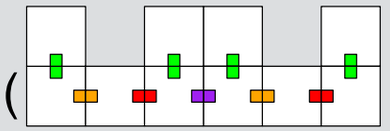
## Input



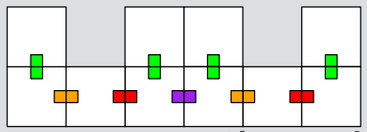

## Output

“Yes”

(  is the unique terminal assembly of the input system)

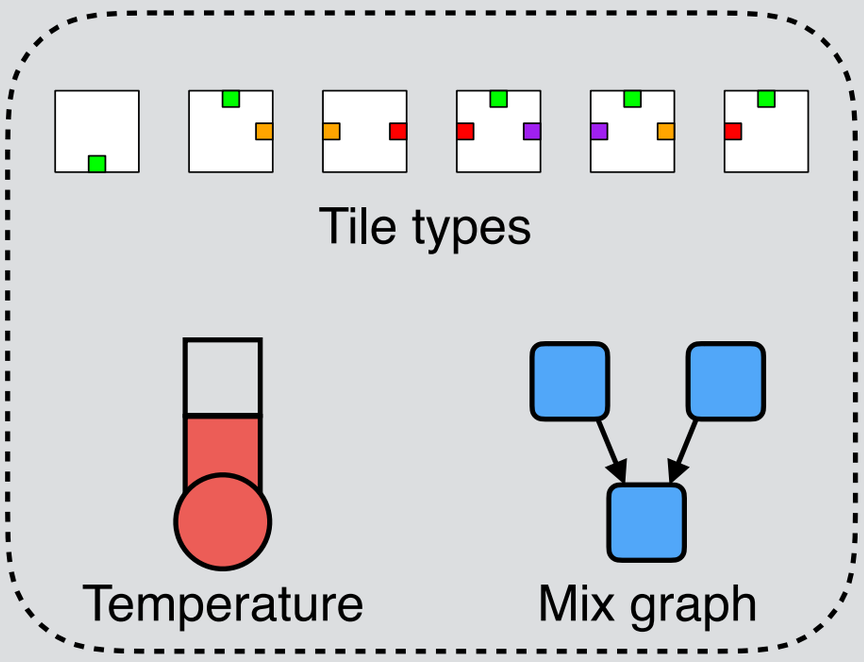
or

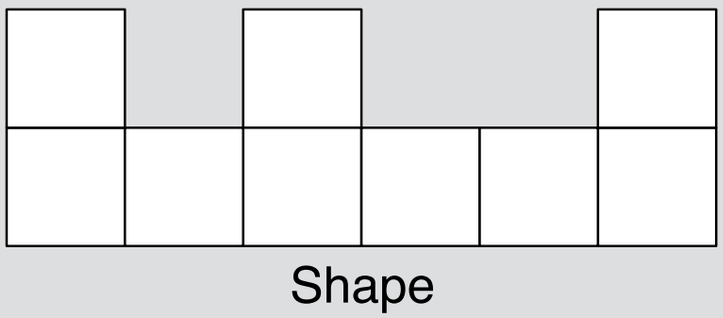
“No”

(  is not a terminal assembly or there is another terminal assembly of the input system)

# Staged USV Problem

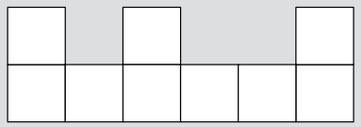
## Input



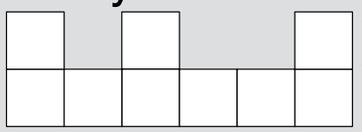

## Output

“Yes”

(  is the shape of every terminal assembly of the input system)

or

“No”

(there is a terminal assembly of the input system without shape  )

# Our results

1. 2HAM USV is coNP$^{NP}$-complete.

# Our results

1. 2HAM USV is coNP$^{NP}$-complete.

2. Staged UAV and USV are coNP$^{NP}$-hard.

# Our results

1. 2HAM USV is coNP$^{NP}$-complete.

2. Staged UAV and USV are coNP$^{NP}$-hard.

3. Staged UAV and USV are in PSPACE.

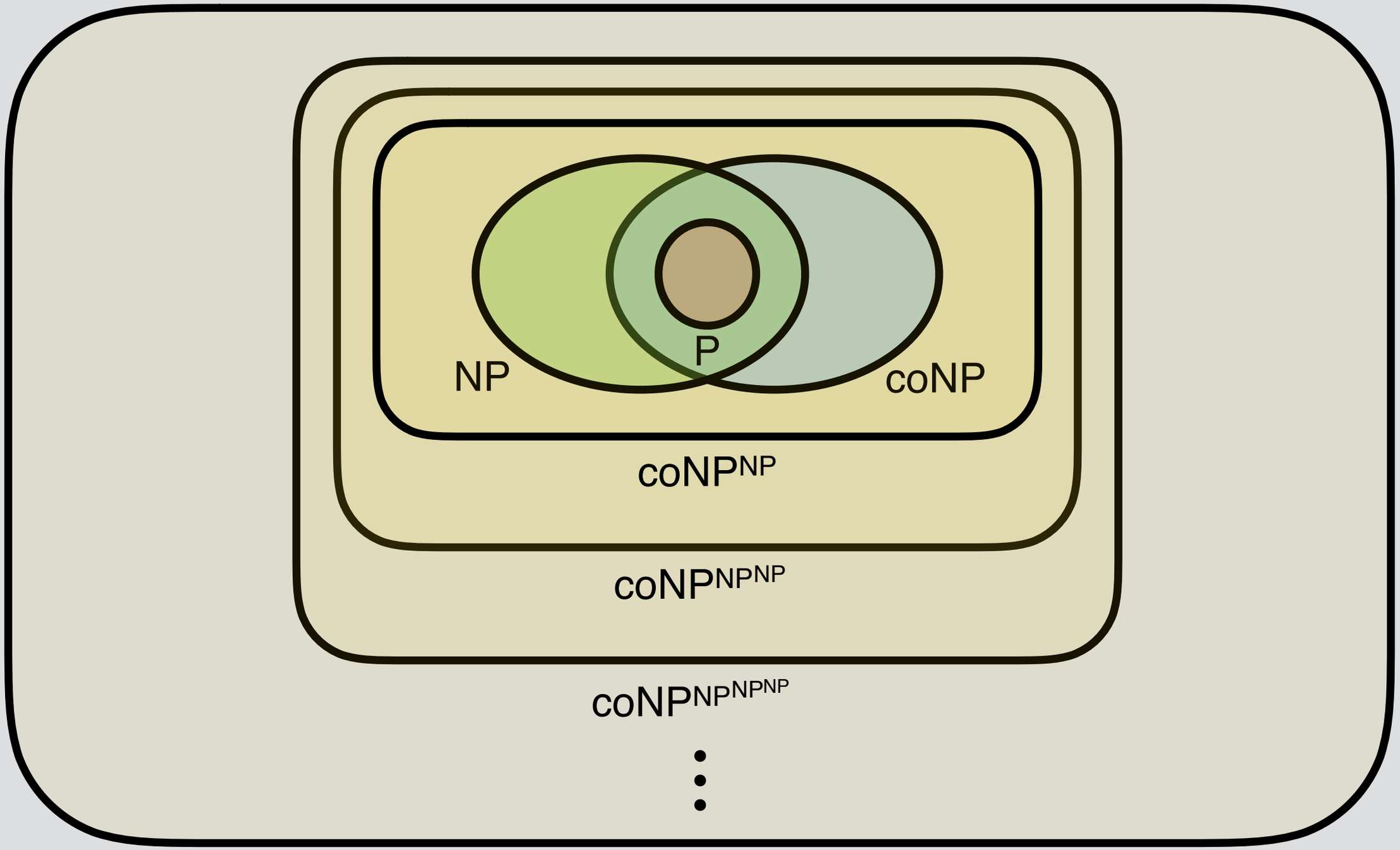
# Our results

1. 2HAM USV is coNP$^{NP}$-complete.

2. Staged UAV and USV are coNP$^{NP}$-hard.

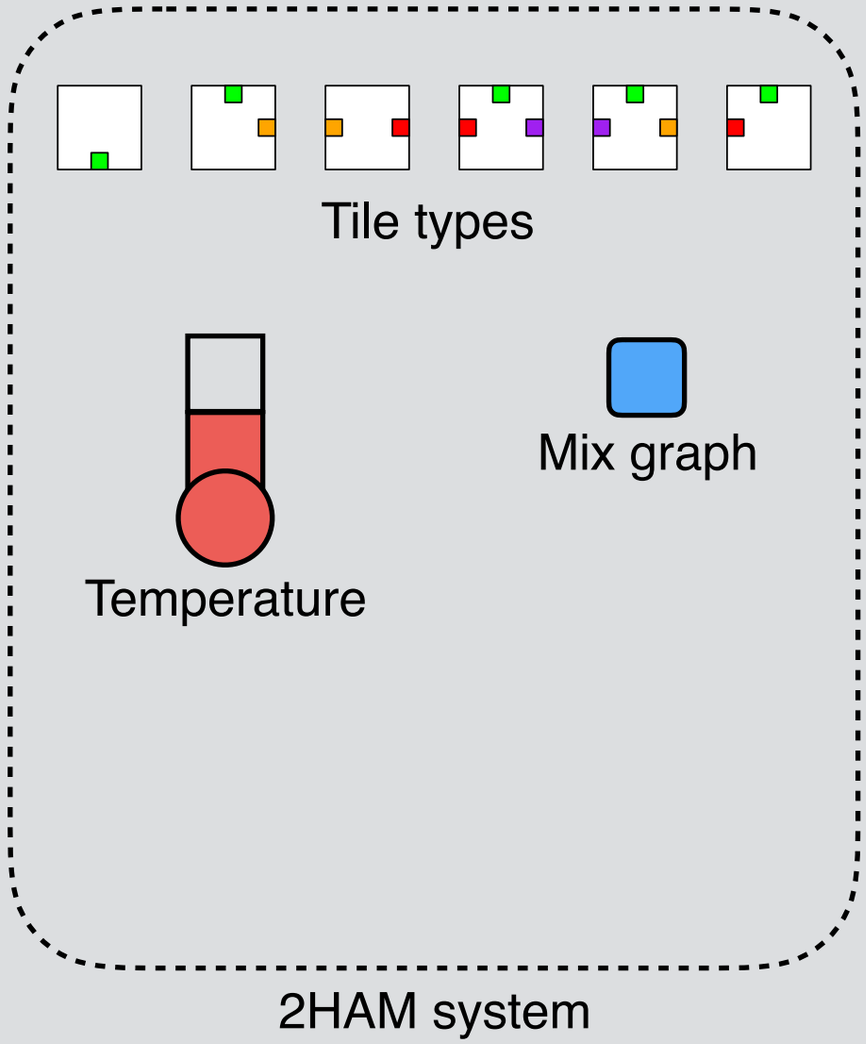
3. Staged UAV and USV are in PSPACE.

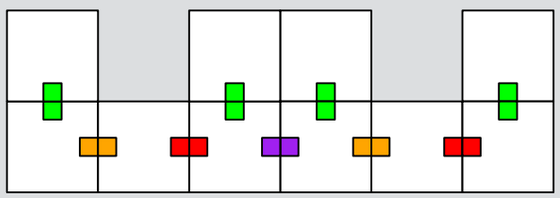
Actually, we prove something stronger…



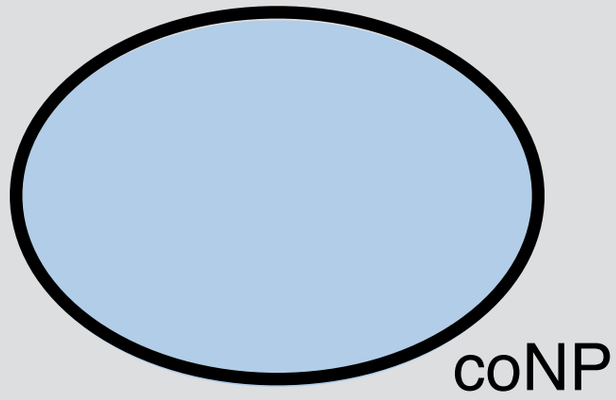

PSPACE

# Input

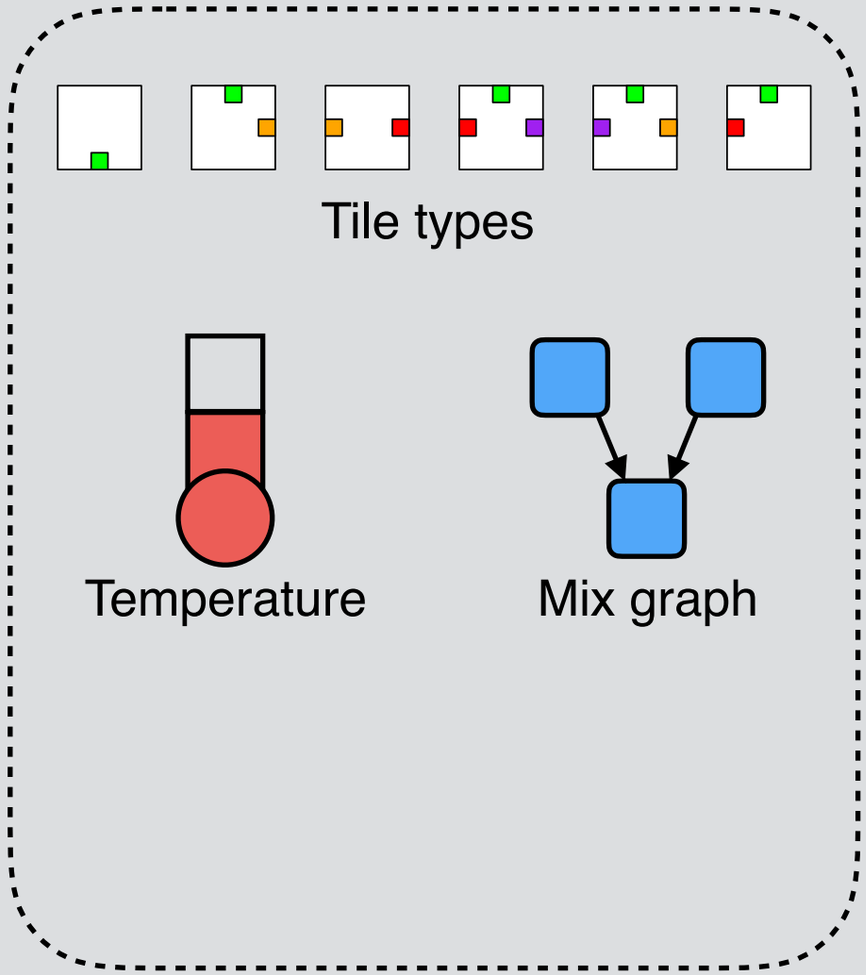


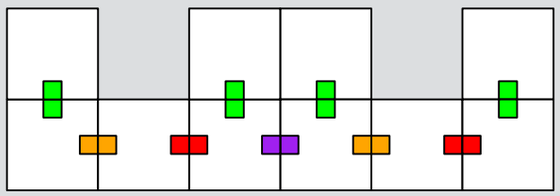




# Staged UAV Complexity

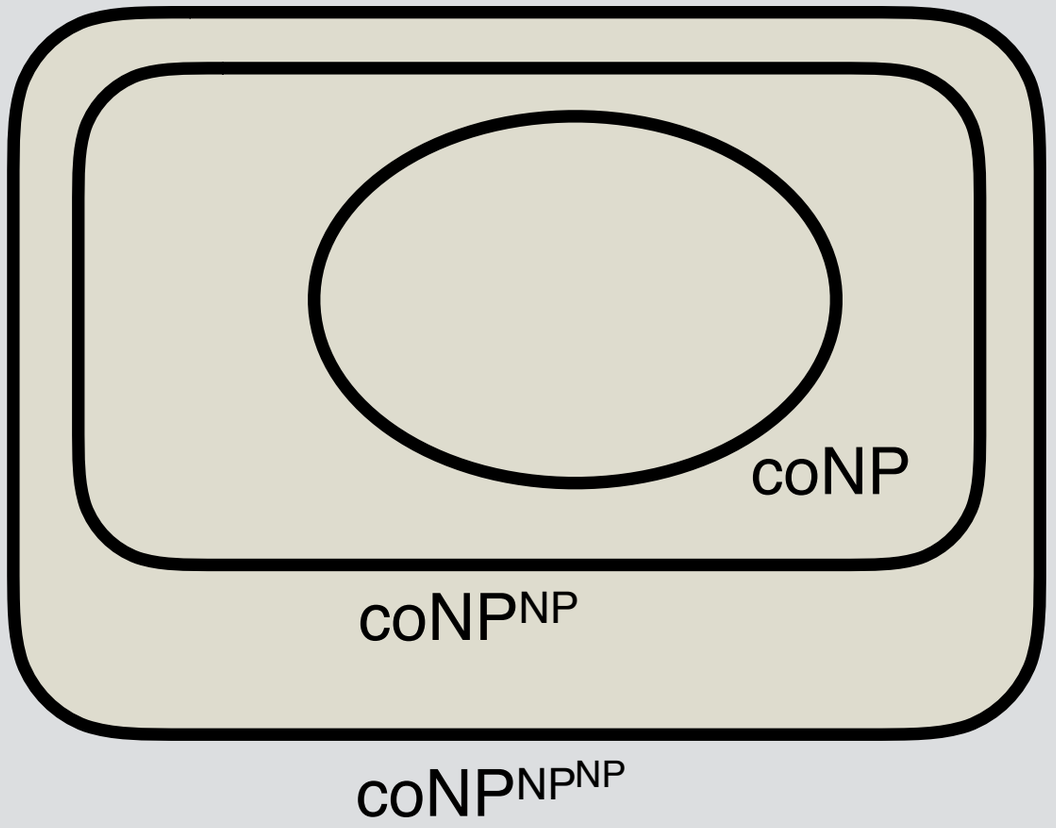


# Input

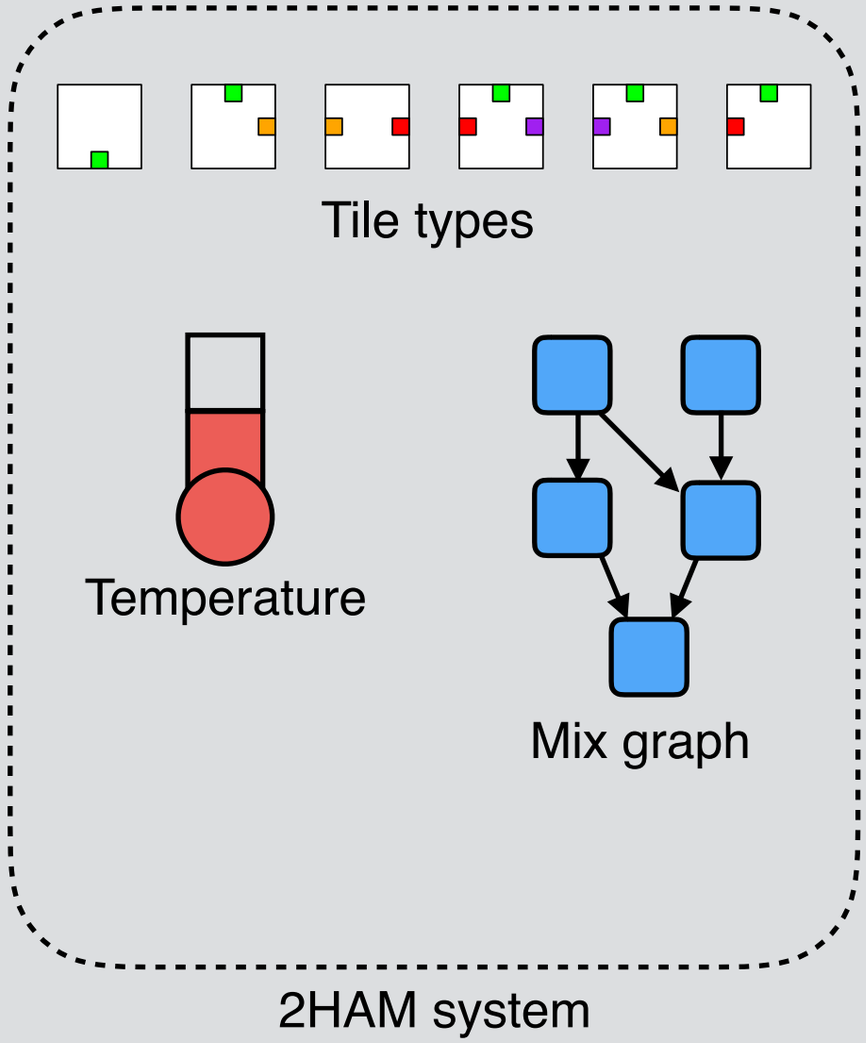


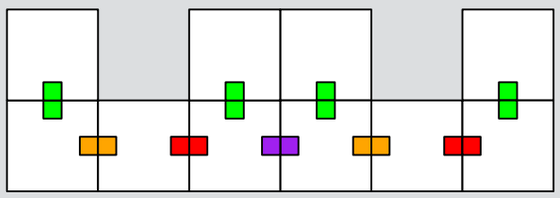




# Staged UAV Complexity

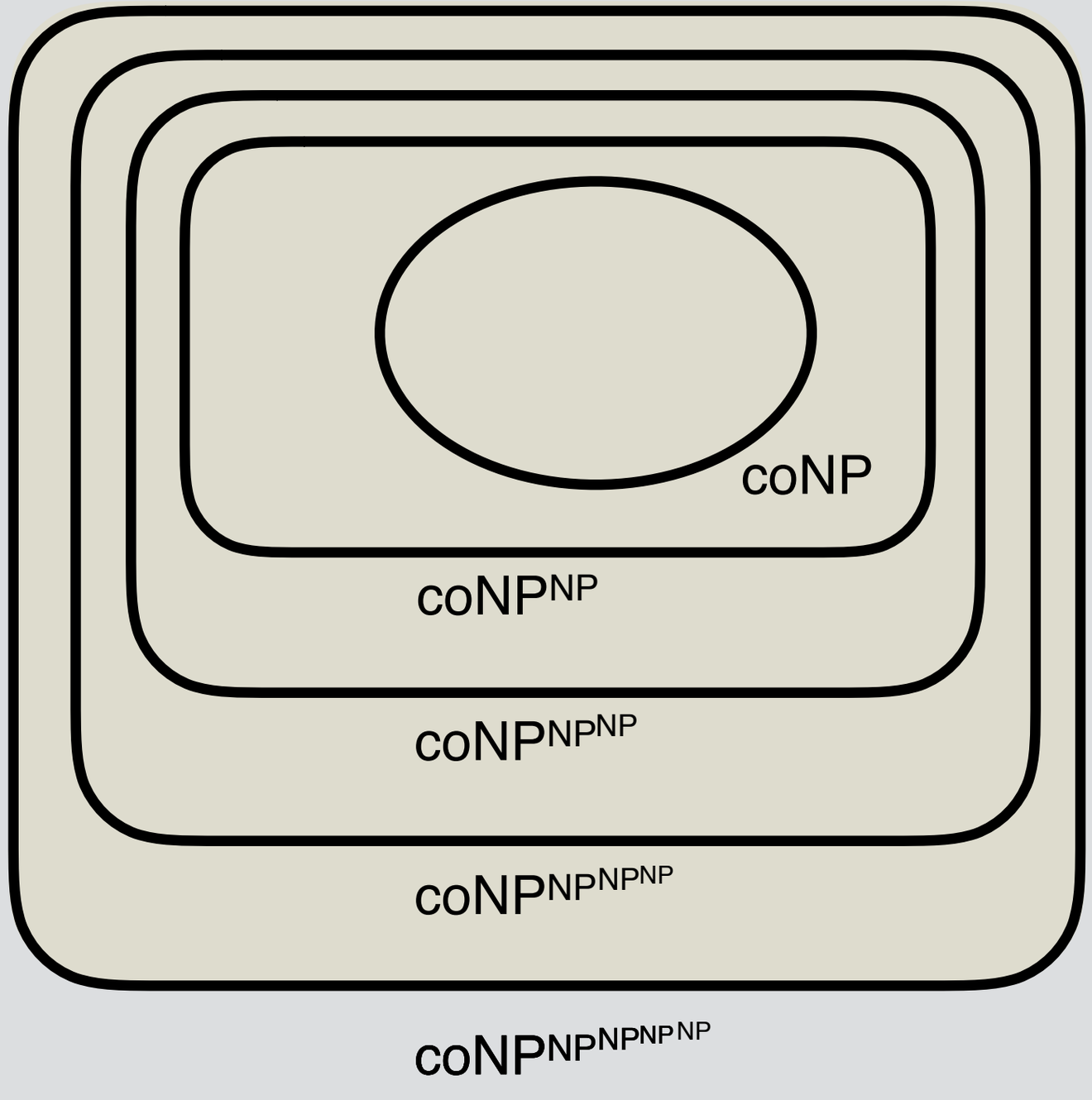


# Input



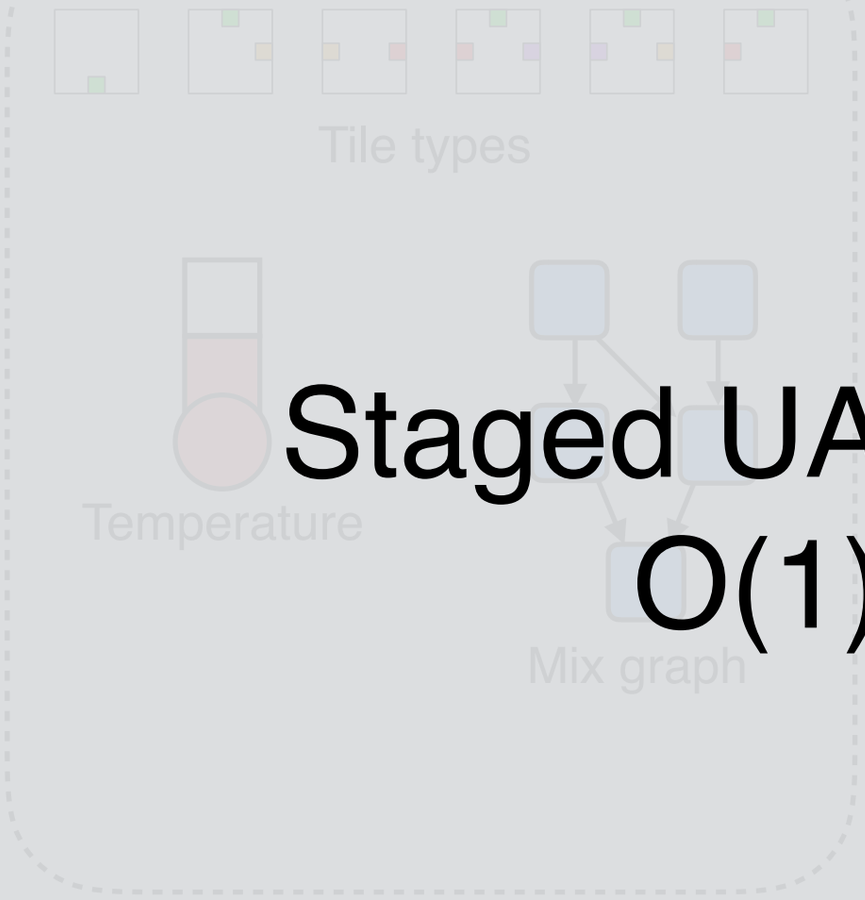




# Staged UAV Complexity



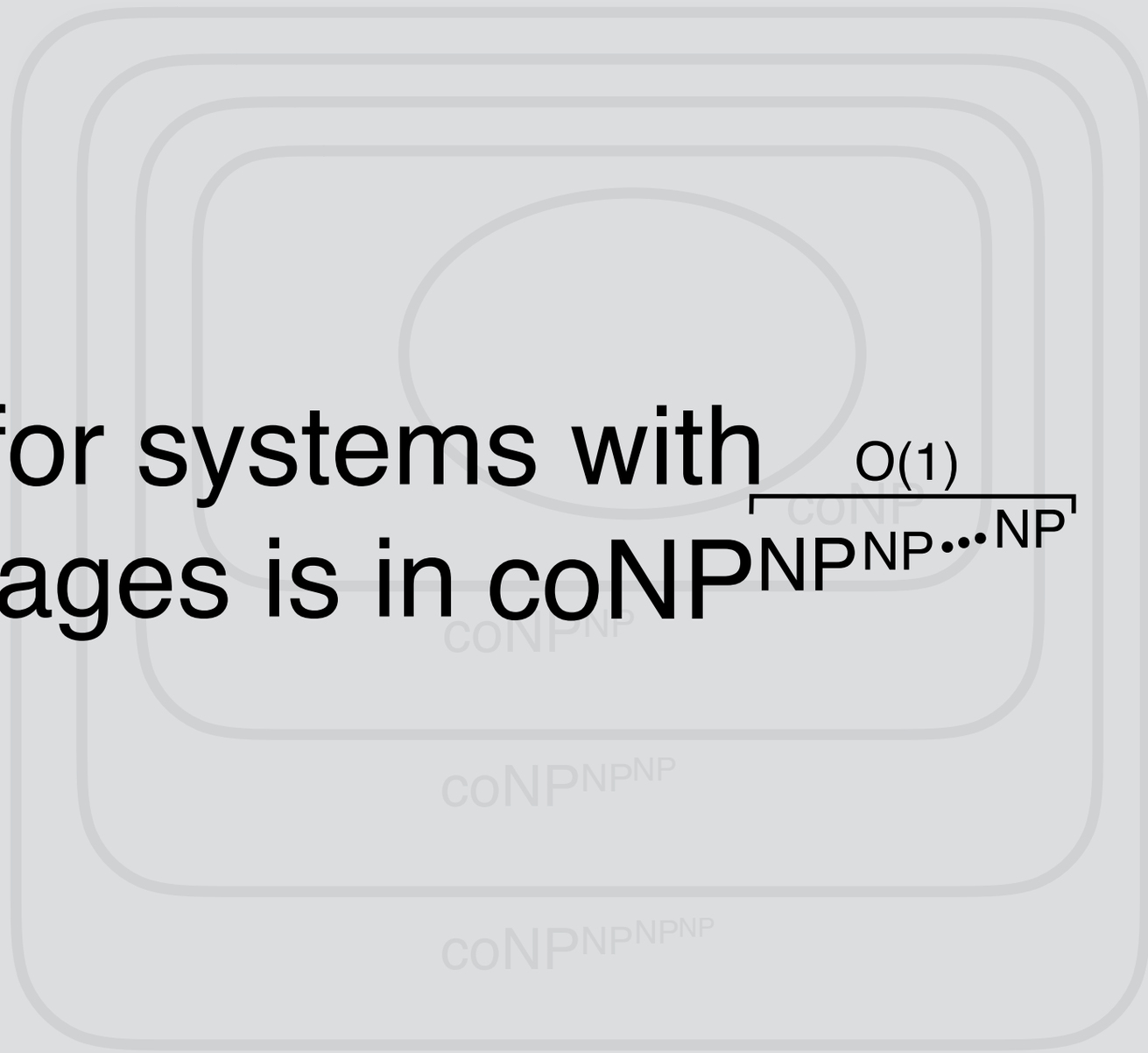
Input

Staged UAV Complexity










# Staged UAV for systems with O(1) stages is in $\text{coNP}^{\text{NP}^{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}}$ (O(1))

# Our results

1. 2HAM USV is $\text{coNP}^{\text{NP}}$-complete.

2. Staged UAV and USV are $\text{coNP}^{\text{NP}}$-hard.

3. Staged UAV and USV are in PSPACE.

    3.5 For O(1) stages, in $\text{coNP}^{\overbrace{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}^{O(1)}}$

# Open problems

# Open problems

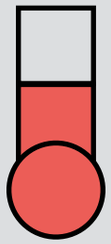
Are staged UAV and USV PSPACE-hard?

For O(1) stages, $\text{coNP}^{\overbrace{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}^{O(1)}}$-hard?

# Open problems

Are staged UAV and USV PSPACE-hard?

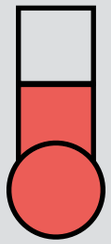
For O(1) stages, $\text{coNP}^{\overbrace{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}^{O(1)}}$-hard?

Do results change if only  allowed?

# Open problems

Are staged UAV and USV PSPACE-hard?

For O(1) stages, $\text{coNP}^{\overbrace{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}^{O(1)}}$-hard?

Do results change if only  allowed?

Are there other self-assembly problems with complexity $\text{coNP}^{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}$ or $\text{NP}^{\text{NP}^{\text{NP}^{\cdots^{\text{NP}}}}}$?

# Verification in Staged Tile Self-Assembly

Robert Schweller* Andrew Winslow* Tim Wylie*

**Abstract**

We prove the unique assembly and unique shape verification problems, benchmark measures of self-assembly model power, are $\mathsf{coNP}^{\mathsf{NP}}$-hard and contained in $\mathsf{PSPACE}$ (and in $\Pi^{\mathsf{P}}_{2s}$ for staged systems with $s$ stages). En route, we prove that unique shape verification problem in the 2HAM is $\mathsf{coNP}^{\mathsf{NP}}$-complete.

## 1 Introduction

Here we consider the complexity of two standard problems in tile self-assembly: deciding whether a system uniquely assembles a given assembly or shape. These so-called *unique assembly* and *unique shape verification* problems are benchmark problems in tile assembly, and have been studied in a variety of models, including the aTAM [1, 2], the $q$-tile model [6], and the 2HAM [3].

The unique assembly and unique shape verification problems ask whether a system behaves as expected: does a given system yield a unique given assembly or assemblies of a given unique shape? The distinct rules by which assemblies form in various tile assembly models