Supplementary material for PQA: Perceptual Question Answering

Yonggang Qi[1*] Kai Zhang[1*] Aneeshan Sain[2] Yi-Zhe Song[2]
[1]Beijing University of Posts and Telecommunications, CN
[2]SketchX, CVSSP, University of Surrey, UK
{qiyg, forer}@bupt.edu.cn {a.sain, y.song}@surrey.ac.uk

1. Introduction

This supplementary material is to further describe the details of our synthetic dataset PQA. We first compare our proposed PQA dataset with ARC dataset in section 2, then describe the data format of PQA in section 3, and finally explain the detailed steps of data generation for every task of PQA in section 4. More examples of the generated data are attached accordingly as well.

2. PQA vs ARC

As shown in Table 1, we offer some statistics of both ARC and our proposed PQA datasets for comparison. Despite being driven by the same general intuition, we can see that our dataset is specifically designed for perceptual organization, which is constructed based on a set of specific Gestalt laws. Therefore, most of the rules implied in our Q/A pairs are not covered by ARC, which is a general AI benchmark. The rule of rotation symmetry is the only overlapping one. Secondly, ARC has a very limited amount of data-samples (about 1000 in total), where even the implied rule for every data-sample is not provided. As a result, benchmarking for each individual rule is impossible for ARC. On the contrary, our dataset offers 100k different instances for each Gestalt law, thereby facilitating explicit measurement of rule-specific perceptual grouping.

3. Data format

In its raw data representation, each question/answer grid (width w, height h) is composed of $w \times h$ color symbols. Taking a question grid-image as example, it can be represented as $x_q = \{s_{i,j}\}_{i=1,j=1}^{w,h}$, where $s_{i,j}$ is a color symbol at location (i, j). Every $s_{i,j}$ is denoted by one of the 10 pre-defined color symbols as shown in Figure 1 namely, `snow` (0), `dark-orange` (1), `forest-green` (2), `royal-blue` (3), `light-gray` (4), `turquoise` (5), `slate-blue` (6), `fire-brick` (7), `gold` (8), `deep-pink` (9). For example, $s_{i,j} = 1$ if the color is `dark-orange` at location (i, j) of a grid-image. In practice, all data is stored in JSON format.

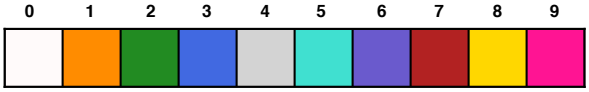

Figure 1. Color symbols and their corresponding codes.

4. Data generation

4.1. T1 Closure Filling

Figure 2 illustrates the grid-image generation process for T1 (closure filling): (i) Generate a blank canvas of size (w, h) for grid-image and set all slots as background with a randomly selected symbol. (ii) Starting from a random location, expand vertically or horizontally, by repetitively placing the same symbol in any one of the 8 slots linked to the current location. After k iterations, a foreground formed from connected symbols is obtained as an answer-grid. (iii) To obtain the question-grid, we remove the internal symbols of the closure region, leaving behind only the boundary symbols. Corresponding pseudo code is shown in Algorithm 1. Some examples of generated data are shown in Figure 10.

4.2. T2 Continuity Connection

Figure 3 illustrates the grid-image generation process for T2 (Continuity Connection): (i) Generate a blank canvas of size (w, h) for the grid-image and set all slots as background with a randomly selected symbol. (ii) Set 5-10 foreground slots at random location in the canvas. (iii) Randomly remove redundant slots till no more than two foreground slots are present in the same line both in vertical and horizontal directions. This forms our question grid. (vi) the answer grid is formed by connecting all foreground slots in the same line. Corresponding pseudo code is shown in Algorithm 2. Some examples of generated data are shown in Figure 11.

*Equal contribution

Table 1. Comparison of ARC and our proposed PQA dataset.

Dataset	Implied Rules	Sub-Rules	#Data	Target
ARC	Objectness priors	Object cohesion	1000 in total	General Core Knowledge
		Object persistence		
		Object influence via contact		
	Goal directedness priors	-		
	Numbers and counting priors	-		
	Basic geometry and topology priors	Lines, rectangular shapes		
		Symmetries, rotations, translations		
		Shape upscaling/downscaling		
		Elastic distortions		
		Containing/being contained		
		Drawing lines, connecting points		
		Orthogonal projections		
		Copying, repeating objects		
PQA	Gestalt laws	Closure filling	100k per law	Specific Gestalt Cognition
		Continuity connection		
		Proximity identification		
		Shape reconstruction		
		Shape & pattern similarity		
		Reflection symmetry		
		Rotation symmetry		

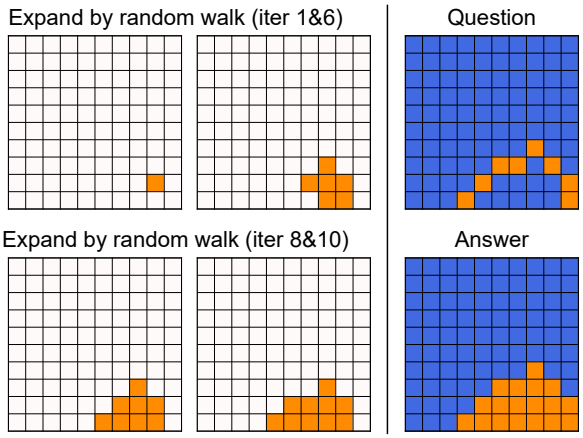

Figure 2. Key steps of data synthesis for T1 (closure filling task).

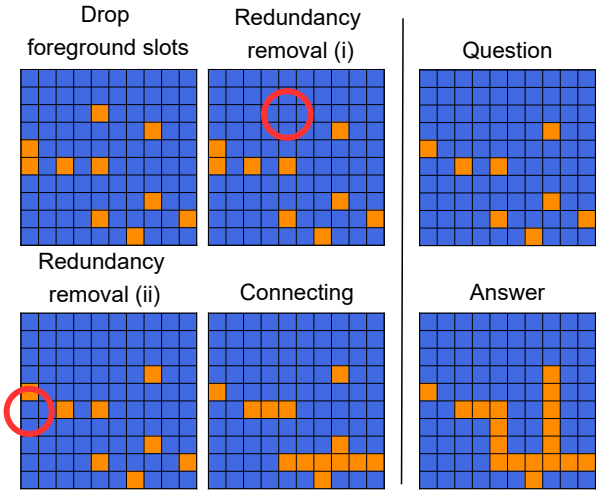

Figure 3. Key steps of data synthesis for T2 (Continuity Connection). Red circle denotes the removed redundant slots.

4.3. T3 Proximity Identification

Figure 4 illustrates the grid-image generation process for T3 (Proximity Identification):(i) Generate a blank canvas of size (w, h) for the grid-image and set all slots as background with a randomly selected symbol. (ii) Randomly select one of 14 pre-defined shapes, which are shown in Figure 5, and place the chosen shape at a random location in canvas. (iii) Place 1-16 single foreground symbols scattered at random locations in the background of canvas to form the question grid. (iv) To construct the answer grid, replace the symbols of the foreground shape with the nearest *single* symbol and remove all the scattered symbols. Corresponding pseudo code is shown in Algorithm 3. Some examples of generated data are shown in Figure 12.

4.4. T4 Shape Reconstruction

Figure 6 illustrates the grid-image generation process for T4 (Shape Reconstruction): (i) Generate a blank canvas of size (w, h) for the grid-image and set all slots as background with a randomly selected symbol. (ii) Generate two rectangles with height in range $[4, \frac{h}{2}]$ and width in range $[4, \frac{w}{2}]$. (iii) The answer grid can be obtained by placing these two rectangles on the canvas with some free

Algorithm 1 Closure Filling Data Generation

Input h: height , w: width ▷ h and w are both randomly initialized in range of [6,30]
k: number of foreground slots ▷ k is a random number in range of $[1, 2\sqrt{hw}]$
smb_{bg}: background symbol ▷ smb_{bg} is randomly selected in [0, 9]
smb_{fg}: foreground symbol ▷ smb_{fg} is randomly selected in [0, 9], $smb_{fg} \neq smb_{bg}$
Output x_q: question grid, x_a: answer grid

1: **procedure** GENERATE($h, w, k, smb_{bg}, smb_{fg}$)
2: **step (i)**
3: $canvas(:h, :w) \leftarrow smb_{bg}$ ▷ int canvas with all slots in background symbol
4: **step (ii)**
5: $counter_{fg} \leftarrow 0$ ▷ init counter of foreground slots to 0
6: **repeat** $\times 2$ ▷ result in two separate connected areas or a merged connected area
7: $p_0 \leftarrow random(h, w)$ ▷ init starting point p_0 at random location within canvas
8: $p \leftarrow p_0$ ▷ set current location p to p_0
9: $canvas(p) \leftarrow smb_{fg}$ ▷ place foreground symbol smb_{fg} at current location p
10: $counter_{fg} \leftarrow counter_{fg} + 1$ ▷ increase foreground counter by one
11: **while** $counter_{fg} < k$ **do** ▷ do until the number of foreground slots reach k
12: $p \leftarrow random(8 \text{ neighbors of } p)$ ▷ set current location p to a randomly chosen neighbor's location
13: **if** $canvas(p)$ is smb_{bg} **then** ▷ if current location is still a background symbol
14: $canvas(p) \leftarrow smb_{fg}$ ▷ assign current location with foreground symbol
15: $counter_{fg} \leftarrow counter_{fg} + 1$ ▷ increase foreground counter by one
16: $p \leftarrow p_0$ ▷ set current location p to init position p_0
17: **else**
18: **continue** ▷ back to line 9
19: **end if**
20: **end while**
21: **until**
22: $region_{out} \leftarrow outer(foreground)$ ▷ get the outer ring of foreground region
23: $region_{in} \leftarrow inner(foreground)$ ▷ get the internal closure region
24: **step (iii)**
25: $x_a(:h, :w) \leftarrow smb_{bg}$ ▷ init answer x_a in size (h, w) with background symbol
26: $x_a(region_{in} \cup region_{out}) \leftarrow smb_{fg}$ ▷ answer is formed by setting closure regions as foreground
27: $x_q(:h, :w) \leftarrow smb_{bg}$ ▷ init question x_q in size (h, w) with background symbol
28: $x_q(region_{out}) \leftarrow smb_{fg}$ ▷ question is formed by setting outer ring as foreground
29: **if** $connected(background, x_a)$ **then** ▷ guarantee the resulting background in x_a is a single connected area
30: **return** x_q, x_a
31: **else**
32: **restart procedure** ▷ restart the whole procedure otherwise
33: **end if**
34: **end procedure**

space in between (a minimum of 2 slots). (iv) The question grid is formed by randomly converting 50% symbols of each resulting rectangle in the answer grid to their background symbols. Corresponding pseudo code is shown in Algorithm 4. Some examples of generated data are shown in Figure 13.

4.5. T5 Shape & Pattern Similarity

Figure 7 illustrates the grid-image generation process for T5 (Shape Reconstruction): (i) Generate a blank canvas of size (w, h) for the grid-image and set all slots as background with a randomly selected symbol. (ii) Generate two squares (s_a and s_b), each of size 5×5. (iii) Set every slot of each square to a random symbol from a set of symbols (3-5 random symbols which must contain the background symbol). (iv) Select a square (e.g., s_a), and resize it to 10×10 to construct a double size square $\hat{s}_a$. Then we place both s_a and $\hat{s}_a$ on the canvas without overlapping one another to form the answer grid-image. (v) To obtain the question grid we replace all slots in $\hat{s}_a$ with a new symbol, and place another square s_b in the canvas without overlapping on the existing squares. Corresponding pseudo code is shown in Al-

Algorithm 2 Continuity Connection Data Generate

Input h: height, w: width ▷ h and w are both randomly initialized in range of [10,30]
k: number of foreground slots ▷ k is a random number in range of $[5, 10]$
smb_{bg}: background symbol ▷ smb_{bg} is randomly selected in [0, 9]
smb_{fg}: foreground symbol ▷ smb_{fg} is randomly selected in [0, 9], $smb_{fg} \neq smb_{bg}$
Output x_q: question grid, x_a: answer grid

procedure GENERATE($h, w, k, smb_{bg}, smb_{fg}$)
step (i)
  $canvas(:h, :w) \leftarrow smb_{bg}$ ▷ init canvas with all slots with background symbol
step (ii)
  $pos \leftarrow rand\_pos(k)$ ▷ randomly select k different position in canvas
  for $p \in pos$ **do**
    $canvas(p) \leftarrow smb_{fg}$ ▷ assign foreground symbol at every location p in canvas
  end for
step (iii)
  for $line \in (rows \cup cols)$ **do** ▷ for each line in vertical or horizontal direction
    if $|fg\_slot(line)| > 2$ **then** ▷ if more than two foreground slots found in the same line
      $canvas \leftarrow remove\_slot(line)$ ▷ randomly remove redundant foreground slots on canvas
    end if
  end for
  $x_q \leftarrow canvas$ ▷ question grid is obtained
step (iv)
  for $line \in (rows \cup cols)$ **do**
    if $is\_lineup\_slot(line)$ **then** ▷ check if two foreground slots in the same line
      $canvas \leftarrow connect\_slot(line)$ ▷ connect slots by assigning the same symbol to slots in between
    end if
  end for
  $x_a \leftarrow canvas$ ▷ answer grid is obtained
  return x_q, x_a
end procedure

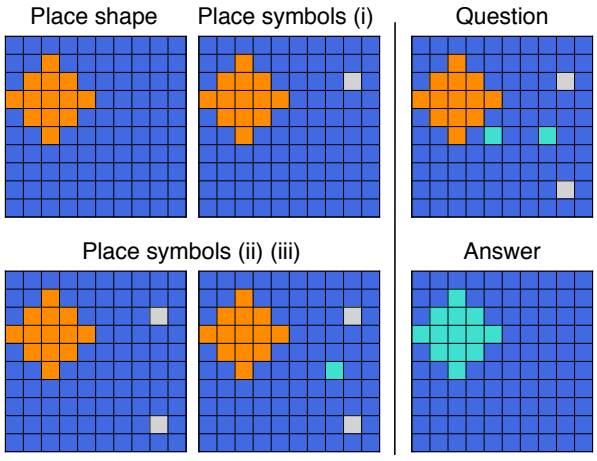

Figure 4. Key steps of data synthesis for T3 (Proximity Identification).

gorithm 5. Some examples of the generated data are shown in Figure 14.

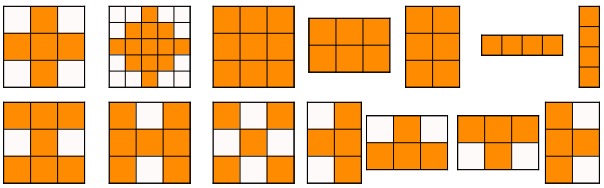

Figure 5. All shapes used in T3 (Proximity Identification)

4.6. T6 Reflection Symmetry

Figure 8 illustrates the grid-image generation process for T6 (Reflection Symmetry): (i) Generate a blank canvas of size (w, h) for the grid-image and set all slots as background with a randomly selected symbol. (ii) Generate a rectangle r such that: if it is flipped along the horizontal it should have its height in range $[3, h]$, width in $[3, \frac{w}{2}]$; whereas for flipping along the vertical it should have its height set to $[3, \frac{h}{2}]$ and width in $[3, w]$. (iii) Randomly replace 30% slots of r with the background symbol. (iv) Generate a symmetry axis along the long or short side of r, and place them on the canvas to form the question. This

Algorithm 3 Proximity Identification Data Generation

Input h: height , w: width ▷ h and w are both randomly initialized in range of [10,30]
$object$: foreground shape ▷ $object$ is randomly select from 14 pre-defined shapes
smb_{obj}: object symbol ▷ smb_{obj} is randomly selected in [0, 9]
smb_{bg}: background symbol ▷ smb_{bg} is randomly selected in [0, 9], $smb_{bg} \neq smb_{obj}$
$smbs_{fg}$: single foreground symbols ▷ $smbs_{fg}$: $1-4$ symbols in [0, 9], $(smb_{bg} \cup smb_{obj}) \cap smbs_{fg} = \varnothing$
k: for every symbol in $smbs_{fg}$, there are k slots placed in canvas ▷ k is up to 4
Output x_q: question grid, x_a: answer grid

procedure GENERATE($h, w, k, object, smb_{object}, smb_{bg}, smbs_{fg}$)
step (i)
  $canvas(:h, :w) \leftarrow smb_{bg}$ ▷ init canvas with all slots using background symbol
step (ii)
  $region_{object} \leftarrow rand\_region(object.size)$ ▷ randomly locate an $object$ region in $canvas$
  $canvas(region_{object}) \leftarrow smb_{obj}$ ▷ assign slots in $region_{object}$ with symbol smb_{obj}
step (iii)
  $slots \leftarrow [\,]$ ▷ init an empty list $slots$
  for $s_{fg} \in smbs_{fg}$ **do**
    $pos \leftarrow rand\_pos(k)$ ▷ randomly select k different locations in canvas
    for $p \in pos$ **do**
      if p not in $region_{object}$ **then**
        $canvas(p) \leftarrow s_{fg}$ ▷ put on the $canvas$
        $slots.insert(\{p, s_{fg}\})$ ▷ assign symbol s_{fg} at location p, inserted in $slots$
      end if
    end for
  end for
  $x_q \leftarrow canvas$ ▷ question grid is obtained
step (iv)
  $distances \leftarrow [\,]$ ▷ init an empty list $distances$
  for $slot \in slots$ **do**
    $d \leftarrow dist(slot, region_{object})$ ▷ calculate the minimum euclidean distance between a $slot$ and the $object$
    $distances.insert(\{d, slot\})$ ▷ insert $dist, slot$ into $distances$
  end for
  $nearest\_slot \leftarrow min(distances)$ ▷ find the nearest $slot$ to $object$
  if $len(nearest\_slot) > 1$ **then** ▷ if more than one nearest $slot$ found
    restart procedure ▷ restart the whole procedure to guarantee an unique nearest $slot$ exists
  end if
  $smb_{nearest} \leftarrow nearest(1).s_{fg}$ ▷ get corresponding foreground symbol of the nearest slot
  $canvas(region_{object}) \leftarrow smb_{nearest}$ ▷ replace $region_{object}$ with symbol of the $nearest$ slot
  $canvas \leftarrow remove(slots)$ ▷ remove all the scattered slots (replace with background)
  $x_a \leftarrow canvas$ ▷ answer grid is obtained
  return x_q, x_a
end procedure

would symbolize the axis for flipping the question pattern. (v) Obtain the mirror shape $\hat{r}$ of r along the symmetry axis to form the answer. Corresponding pseudo code is shown in Algorithm 6. Some examples of generated data are shown in Figure 15.

4.7. T7 Rotation Symmetry

Figure 9 illustrates the grid-image generation process for T7 (Rotation Symmetry): (i) Generate a rectangle r with both height and width between 5 and 15 slots. (ii) Set all slots with randomly selected 3-5 different symbols. (iii) Generate a new rectangle $\hat{r}$ as the answer grid by flipping r twice over – we first flip r along horizontal direction to generate a new rectangle r' (Figure 9), which is then flipped again along vertical direction, giving $\hat{r}$. This forms our answer grid. (iv) Based on the answer grid, generate two rectangle masks to form the corresponding question grid. Pseudo code is shown in Algorithm 7. Some examples of

Algorithm 4 Shape Reconstruction Data Generate

Input h: height, w: width ▷ h and w are both randomly initialized in range of [12,30]
smb_{bg}: background symbol ▷ smb_{bg} is randomly selected in [0, 9]
smb_{fg}: foreground symbol ▷ smb_{fg} is randomly selected in [0, 9], $smb_{fg} \neq smb_{bg}$
Output x_q: question grid, x_a: answer grid

procedure GENERATE(h, w, smb_{bg}, smb_{fg})
step (i)
  $canvas(:h, :w) \leftarrow smb_{bg}$ ▷ init canvas with all slots in background symbol
step (ii)
  $rects_a \leftarrow [\,]$ ▷ init an empty list $rects_a$ for creating two rectangles
  repeat $\times 2$ ▷ generate two rectangles
    $rect_h \leftarrow random([4, \frac{h}{2}))$
    $rect_w \leftarrow random([4, \frac{w}{2}))$
    $rect(:rect_h, :rect_w) \leftarrow smb_{bg}$ ▷ create $rect$ with size $rect_h \times rect_w$
    $rects_a.insert(rect)$ ▷ store rectangle $rect$ in $rects_a$
  until
step (iii)
  $regions \leftarrow rand\_regions(rects_a.size)$ ▷ select two regions for placing $rects_a$ with a minimum gap (2 slots)
  $canvas \leftarrow put(regions, rects_a)$ ▷ place two rectangles $rects_a$ in $canvas$
  $x_a \leftarrow canvas$ ▷ answer grid is obtained
step (iv)
  $rects_q \leftarrow [\,]$ ▷ init an empty list $rects_q$ for generate two masked $rect$
  for $rect \in rects_a$ **do**
    $rect \leftarrow rand\_set(0.5, smb_{bg})$ ▷ randomly set 50% symbols of $rect$ to smb_{bg}
    $sides \leftarrow get\_sides(rect)$ ▷ get 4 sides of $rect$
    for $side \in sides$ **do** ▷ for each side of masked $rect$ at least has two foreground slots
      while $count(smb_{fg}, side) < 2$ **do** ▷ otherwise, we increase the number to two
        $p \leftarrow rand\_pos(side)$ ▷ randomly select a position
        $side(p) \leftarrow smb_{fg}$ ▷ set the corresponding slot to foreground.
      end while
    end for
    $rects_q.insert(rect)$
  end for
  $canvas \leftarrow put(regions, rects_q)$ ▷ place the masked rectangles $rects_q$ in the $canvas$
  $x_q \leftarrow canvas$ ▷ question grid is obtained
  return x_q, x_a
end procedure

the generated data are shown in Figure 16.

Algorithm 5 Shape & Pattern Similarity Data Generation

Input h: height, w: width ▷ h and w are both randomly initialized in range of [12,30]
smb_{mask}: mask symbol used for question grid ▷ smb_{mask} is randomly selected in [0, 9]
smb_{bg}: background symbol ▷ smb_{bg} is randomly selected in [0, 9], $smb_{bg} \neq smb_{mask}$
$smbs_{fg}$: foreground symbols ▷ $smbs_{fg}$: $3-5$ symbols in [0, 9], $smb_{bg} \in smbs_{fg}, smb_{mask} \notin smbs_{fg}$
Output x_q: question grid, x_a: answer grid

procedure GENERATE($h, w, smb_{mask}, smb_{bg}, smbs_{fg}$)
step (i)
  $canvas(:h, :w) \leftarrow smb_{bg}$ ▷ init canvas with all slots in background symbol
step (ii) & (iii)
  for $i \in \{a, b\}$ **do** ▷ creating two squares s_a and s_b
    $s_i(:5, :5) \leftarrow smb_{bg}$ ▷ create square in size 5×5
    $s_i \leftarrow rand\_set(smbs_{fg})$ ▷ randomly set each slot in s_i with one of $smbs_{fg}$
  end for
step (iv)
  $\hat{s}_a \leftarrow resize(s_a, \{10, 10\})$ ▷ resize s_a to a double size square $\hat{s}_a$
  $s_{ar} \leftarrow rand\_region(\{5, 5\})$ ▷ select a region for s_a
  $s_{br} \leftarrow rand\_region(\{5, 5\})$ ▷ select a region for s_b
  $\hat{s}_{ar} \leftarrow rand\_region(\{10, 10\})$ ▷ select a region for $\hat{s}_a$
  if $is\_overlapping(s_{ar}, s_{br}, \hat{s}_{ar})$ **then**
    restart procedure ▷ guarantee all regions are non-overlapping
  end if
  $canvas(s_{ar}) \leftarrow s_a$ ▷ place s_a in canvas
  $canvas(\hat{s}_{ar}) \leftarrow \hat{s}_a$ ▷ place $\hat{s}_a$ in canvas
  $x_a \leftarrow canvas$ ▷ answer grid is obtained
step (v)
  $canvas(\hat{s}_{ar}) \leftarrow smb_{mask}$ ▷ mask region $\hat{s}_{ar}$ for constructing question
  $canvas(s_{br}) \leftarrow s_b$ ▷ place s_b in canvas
  $x_q \leftarrow canvas$ ▷ question grid is obtained
  return x_q, x_a
end procedure

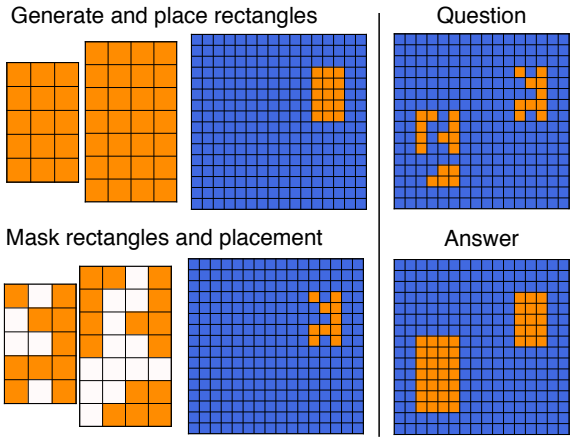

Figure 6. Key steps of data synthesis for T4 (Shape Reconstruction).

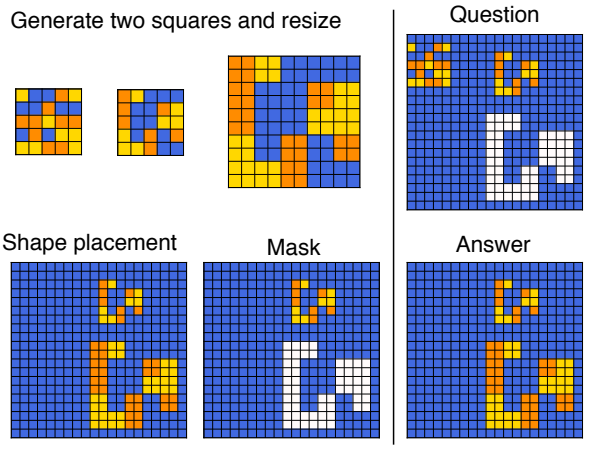

Figure 7. Key steps of data synthesis for T5 (Shape & Pattern Similarity Generate Algorithm).

Algorithm 6 Reflection Symmetry Data Generation

Input h: height, w: width ▷ h and w are both randomly initialized in range of [10,30]
smb_{sa}: symmetry axis symbol ▷ smb_{sa} is randomly selected in [0, 9]
smb_{bg}: background symbol ▷ smb_{bg} is randomly selected in [0, 9], $smb_{bg} \neq smb_{sa}$
smb_{fg}: foreground symbol ▷ smb_{fg} is randomly selected in [0, 9], $smb_{fg} \neq smb_{bg}, smb_{fg} \neq smb_{bg}$
Output x_q: question grid, x_a: answer grid

procedure GENERATE($h, w, smb_{sa}, smb_{bg}, smb_{fg}$)
step (i)
  $canvas(:h, :w) \leftarrow smb_{bg}$ ▷ init canvas with all slots in background symbol
step (ii)
  $sa \leftarrow rand(top, bottom, left, right)$ ▷ select a side in (top, bottom, left, right) as symmetry axis
  if $sa \in \{top, bottom\}$ **then** ▷ if would flip along vertical direction
    $r \leftarrow rand\_size([3, \frac{h}{2}], [3, w])$ ▷ create rectangle r with height in range of $[3, \frac{h}{2}]$, width in range of $[3, w]$
  else if $sa \in \{left, right\}$ **then** ▷ if would flip along horizontal direction
    $r \leftarrow rand\_size([3, h], [3, \frac{w}{2}])$ ▷ create rectangle r with height in range of $[3, h]$, width in range of $[3, \frac{w}{2}]$
  end if
step (iii)
  $r \leftarrow rand\_set(\{0.7, smb_{fg}\}, \{0.3, smb_{bg}\})$ ▷ set 30% slots of r to background and the rest to foreground
  $region_r \leftarrow rand\_region(r.size)$ ▷ randomly select a region fits the size of r
  $canvas(region_r) \leftarrow r$ ▷ place r in canvas
step (iv)
  $region_{sa} \leftarrow get\_side(region_r)$ ▷ get symmetry axis region
  $canvas(region_{sa}) \leftarrow smb_{sa}$ ▷ set symmetry axis region with smb_{sa} in $canvas$
  $x_q \leftarrow canvas$ ▷ question grid is obtained
step (v)
  $\hat{r} \leftarrow symmetry(r, sa)$ ▷ get the mirror shape $\hat{r}$ of r along symmetry axis sa
  $region_{\hat{r}} \leftarrow symmetry\_region(region, sa)$ ▷ get the region of $\hat{r}$ in canvas
  $canvas(region_{\hat{r}}) \leftarrow \hat{r}$ ▷ place $\hat{r}$ in canvas
  $x_a \leftarrow canvas$ ▷ answer grid is obtained
  return x_q, x_a
end procedure

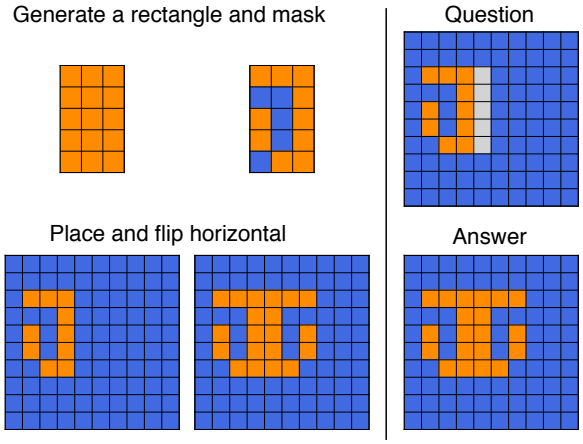

Figure 8. Key steps of data synthesis for T6 (Reflection Symmetry).

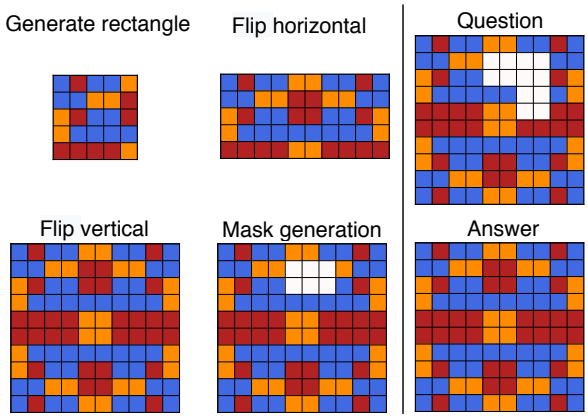

Figure 9. Key steps of data synthesis for T7 (Rotation Symmetry).

Algorithm 7 Rotation Symmetry Data Generation

Input h: height, w: width ▷ h and w are both randomly initialized in range of [5, 15]
smb_{mask}: mask symbol ▷ always set to 0 at now
$smbs$: symbols ▷ $smbs$: $3-5$ symbols in [0, 9] to construct the pattern of r, $smb_{mask} \notin smbs$
Output x_q: question grid, x_a: answer grid

procedure GENERATE($h, w, smb_{mask}, smbs$)
step (i) & (ii)
  $r \leftarrow empty(h, w)$ ▷ create a blank rectangle r in size $h \times w$
  $r \leftarrow rand\_set(smbs)$ ▷ randomly set every slot of r with one symbol in $smbs$
step (iii)
  $r \leftarrow flip\_horizontal(r)$ ▷ flipping r along vertical direction firstly.
  $\hat{r} \leftarrow flip\_vertical(r)$ ▷ flip the updated rectangle r again along horizontal direction.
  $x_a \leftarrow \hat{r}$ ▷ answer grid is obtained
step (iv)
  $size_{mask} \leftarrow rand\_size(\{1, \frac{h}{2}\}, \{1, w\})$ ▷ randomly get a size with height in $[1, \frac{h}{2}]$ and width in $[1, w]$
  $region_{mask} \leftarrow rand\_region(\hat{r}, size_{mask}, tb)$ ▷ get a mask region at top/bottom half of $\hat{r}$
  $\hat{r}(region_{mask}) \leftarrow smb_{mask}$ ▷ mask $\hat{r}$ within $region_{mask}$ with smb_{mask}
  $size_{mask} \leftarrow rand\_size(\{1, h\}, \{1, \frac{w}{2}\})$ ▷ randomly get a size with height in $[1, h]$ and width in $[1, \frac{w}{2}]$
  $region_{mask} \leftarrow rand\_region(\hat{r}, size_{mask}, lr)$ ▷ get a mask region at left/right half of $\hat{r}$
  $\hat{r}(region_{mask}) \leftarrow smb_{mask}$ ▷ mask $\hat{r}$ within $region_{mask}$ with smb_{mask}
  $x_q \leftarrow \hat{r}$ ▷ question grid is obtained
  return x_q, x_a
end procedure

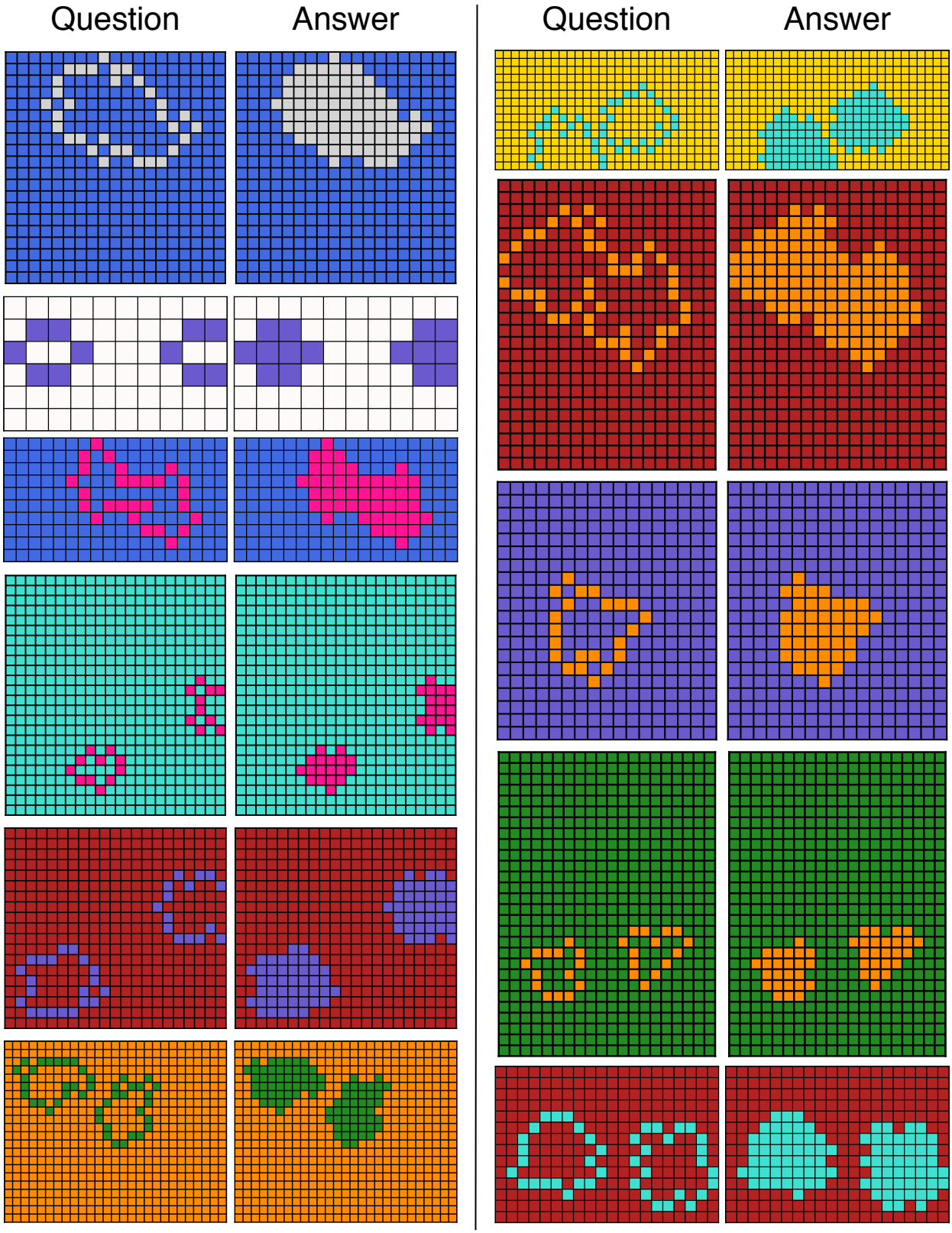

Figure 10. Sample of T1 (Closure Filling)

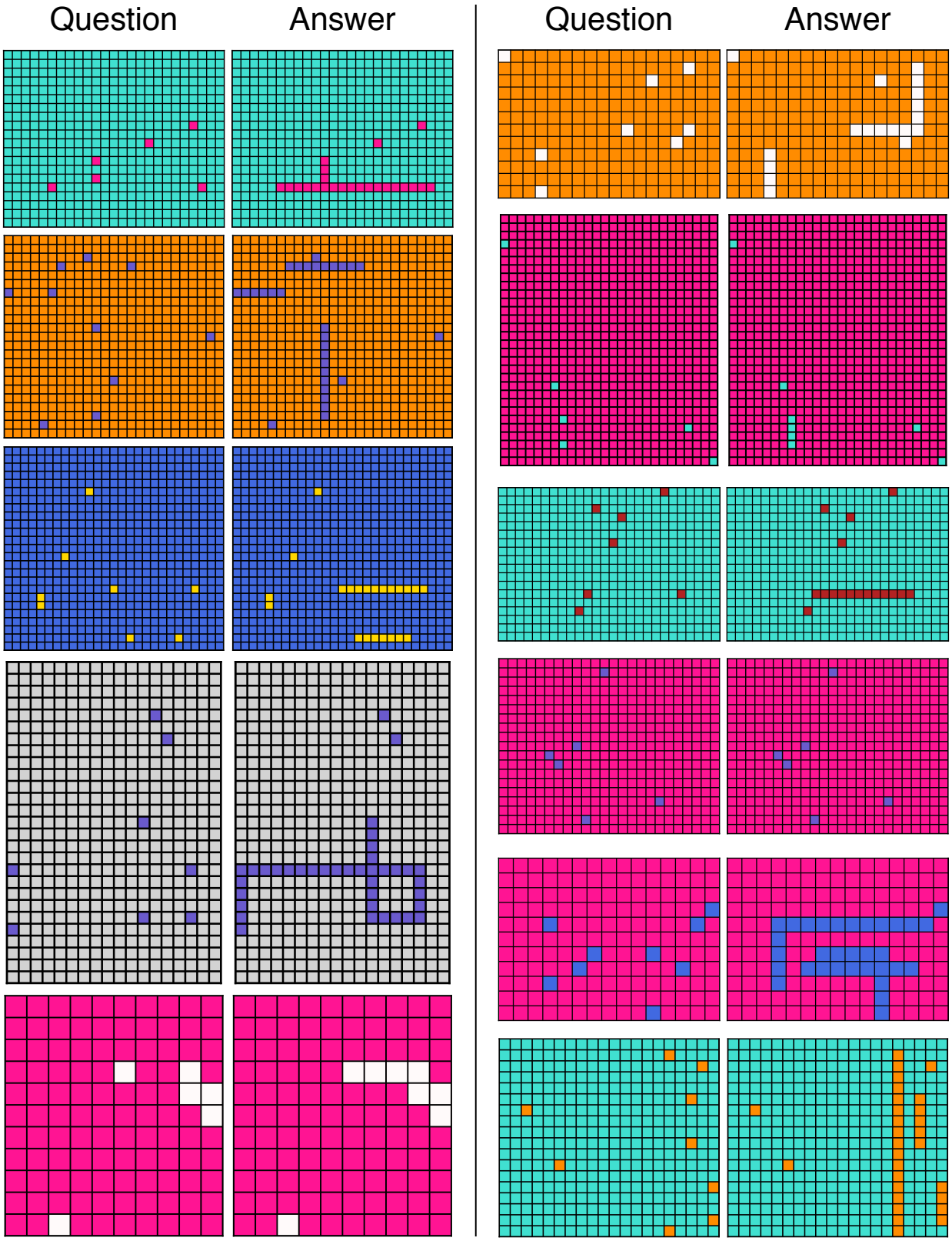

Figure 11. Sample of T2 (Continuity Connection)

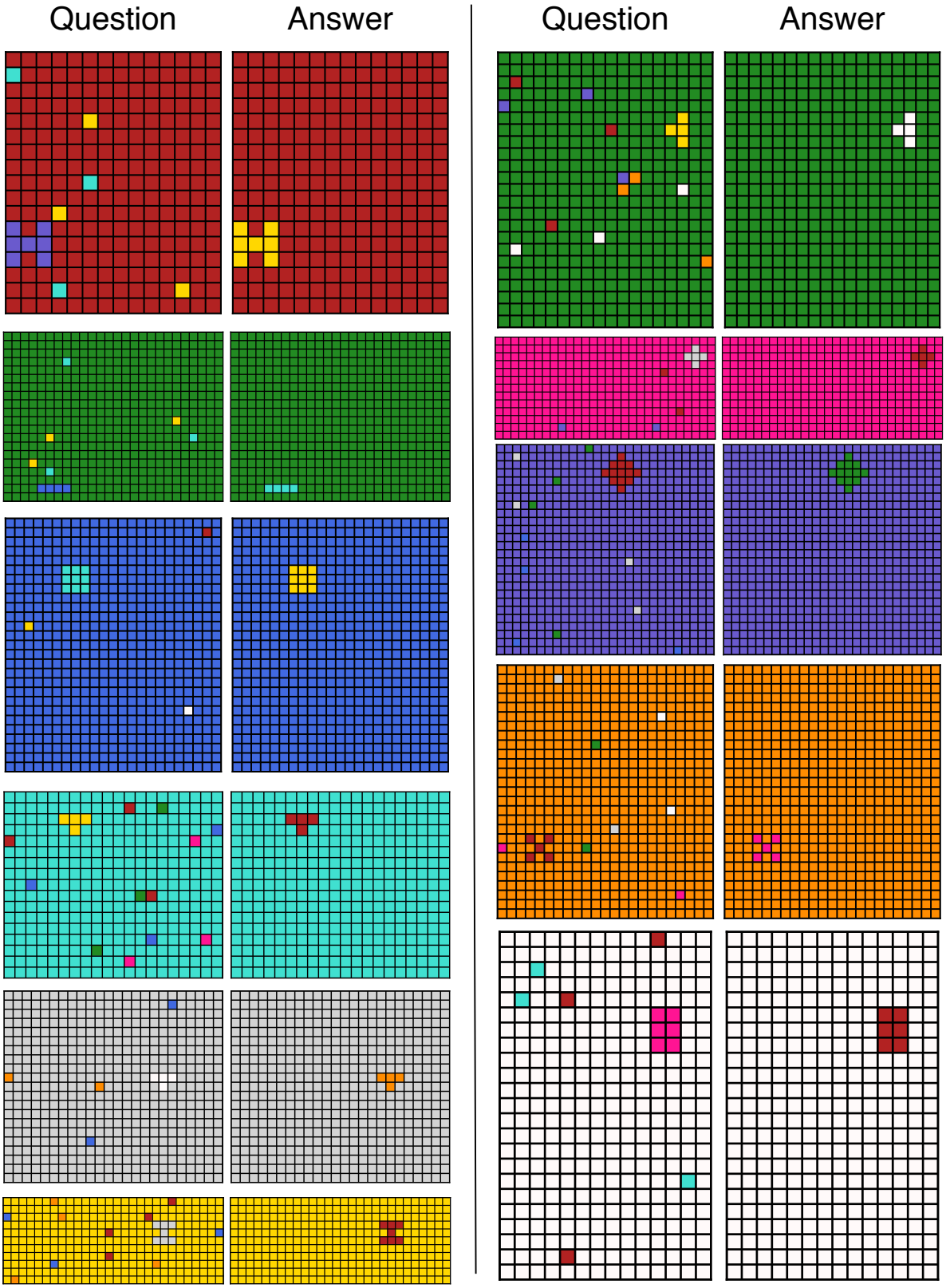

Figure 12. Sample of T3 (Proximity Identification)

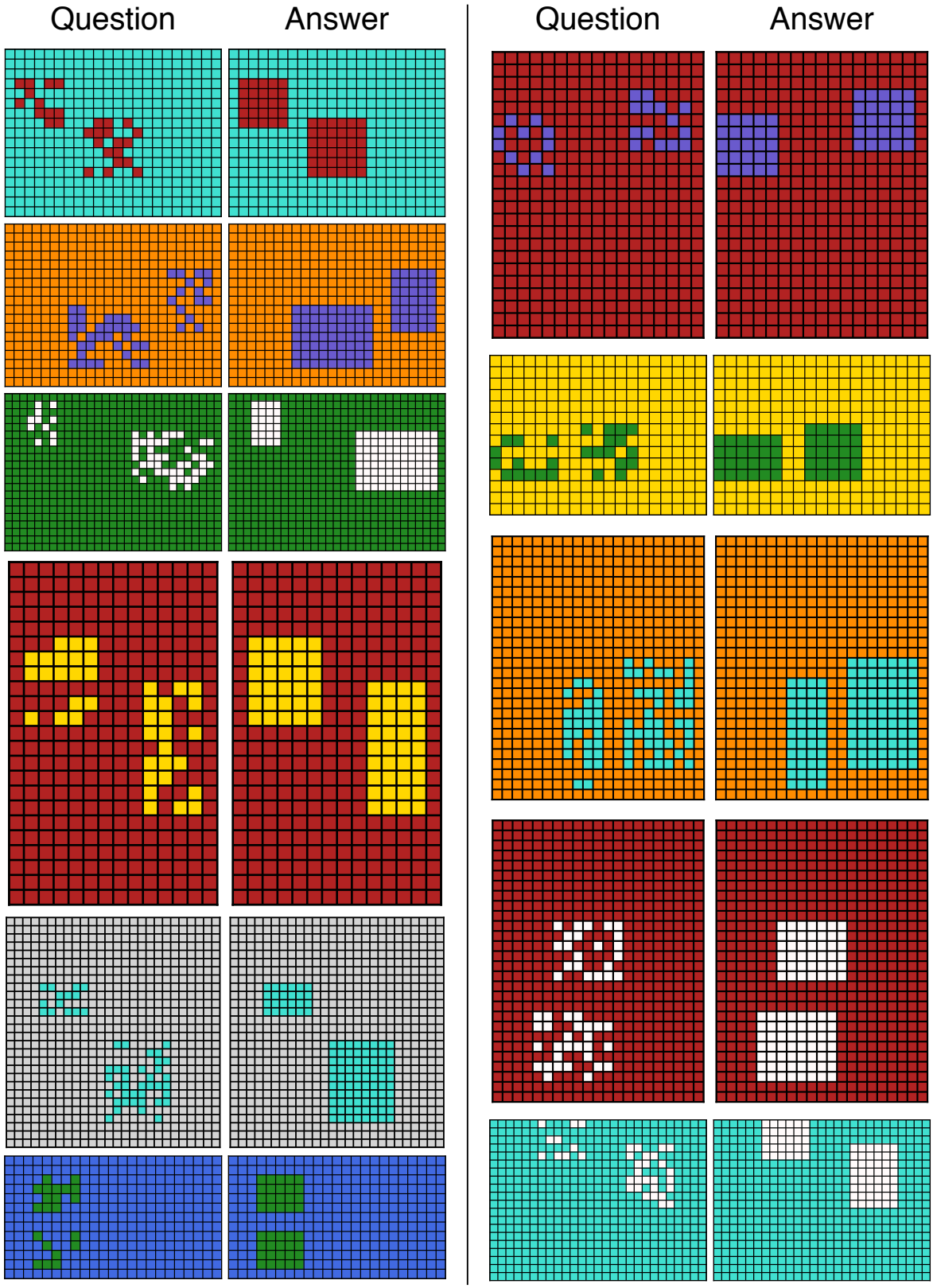

Figure 13. Sample of T4 (Shape Reconstruction)

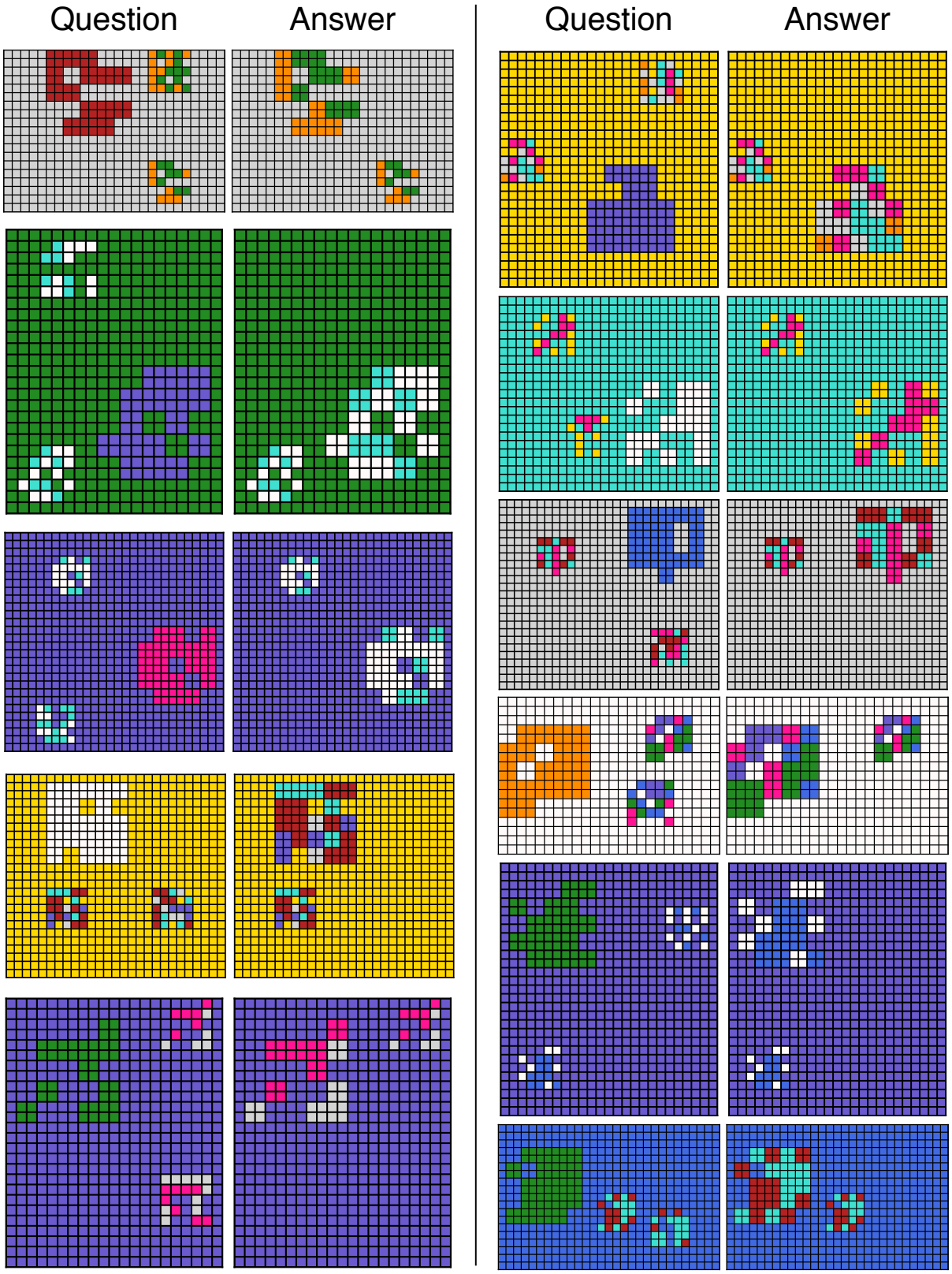

Figure 14. Sample of T5 (Shape & Pattern Similarity)

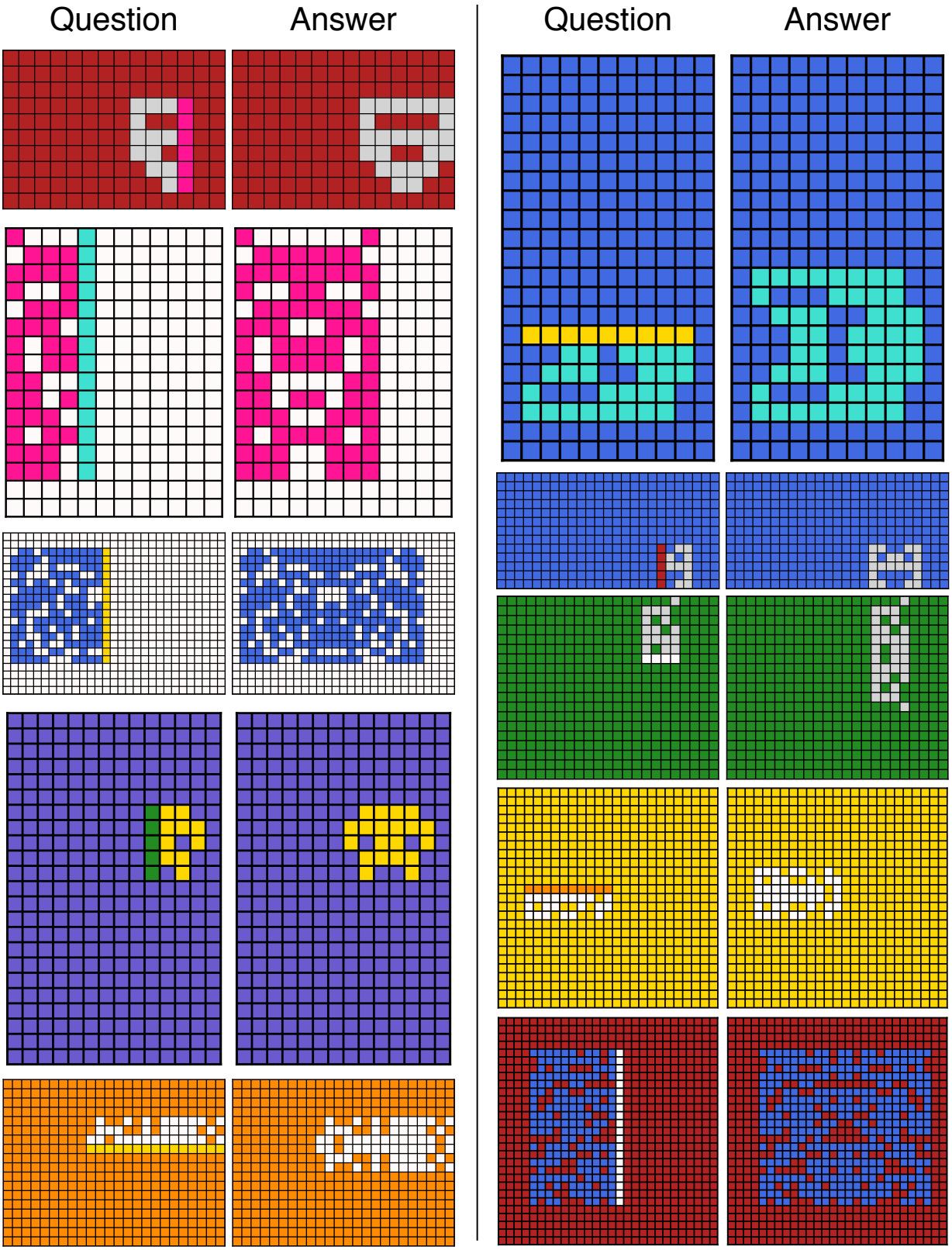

Figure 15. Sample of T6 (Reflection Symmetry)

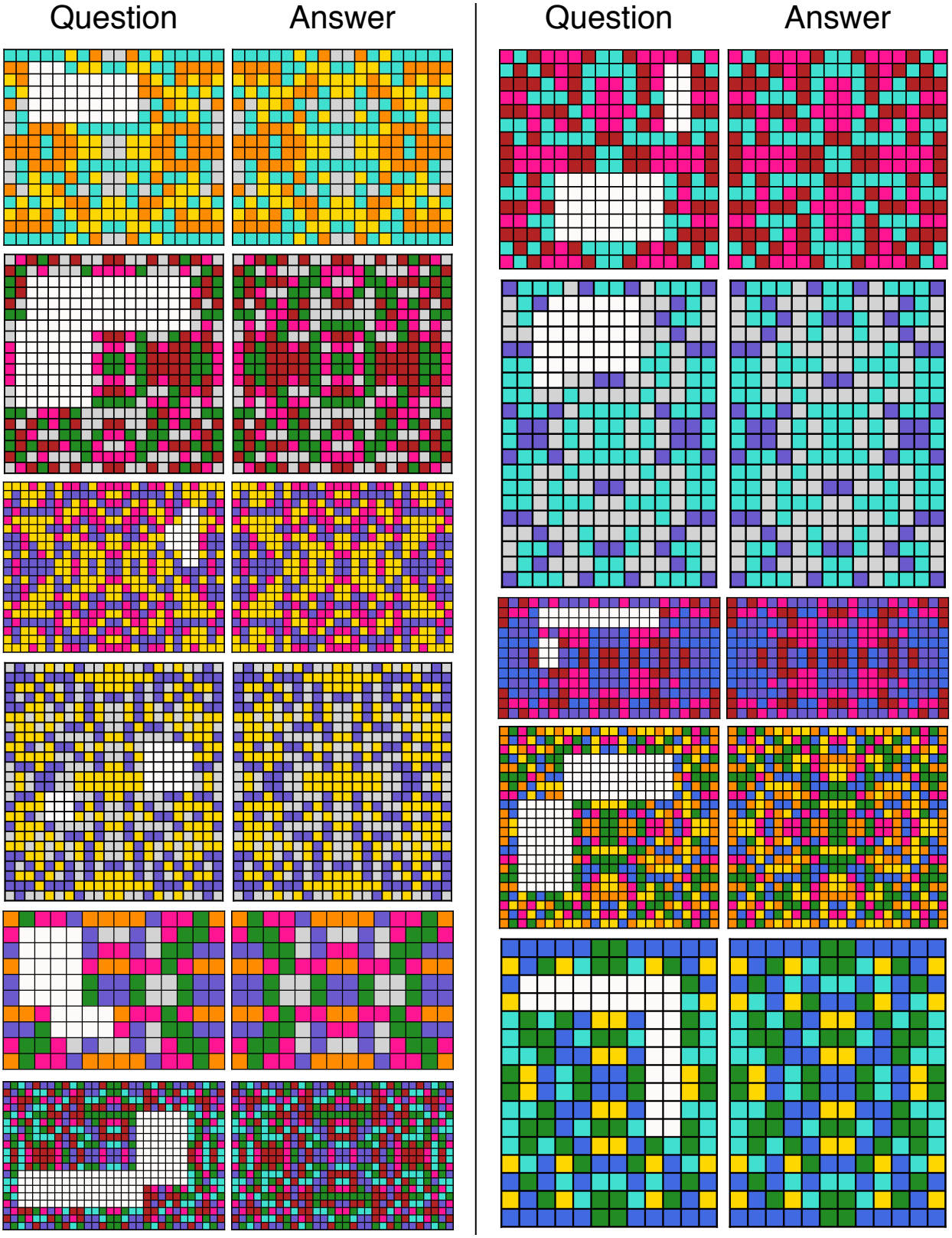

Figure 16. Sample of T6 (Rotation Symmetry)