# Bust-a-Move/Puzzle Bobble is NP-Complete

Erik D. Demaine* Stefan Langerman†



**Abstract**

We prove that the classic 1994 Taito video game, known as Puzzle Bobble or Bust-a-Move, is NP-complete. Our proof applies to the perfect-information version where the bubble sequence is known in advance, and it uses just three bubble colors.

*"A girl runs up with somethin' to prove.*
*So don't just stand there. Bust a move!"*
— Young MC [YDD89]

## 1 Introduction

Erik grew up playing the action platform video game *Bubble Bobble* (バブルボブル), starring cute little brontosauruses Bub and Bob,[1] on the Nintendo Entertainment System. (The game was first released by Taito in 1986, in arcades [Thea].) Some years later (1994), Bub and Bob retook the video-game stage with the puzzle game *Puzzle Bobble* (パズルボブル), known as *Bust-a-Move* in the United States [Theb, Wik]. This game essentially got Stefan through his Ph.D.: whenever he needed a break, he would play as much as he could with one quarter. To celebrate the game's 21-year anniversary, we analyze its computational complexity, retroactively justifying the hours we spent playing.

In Puzzle Bobble, the game state is defined by a hexagonal grid, each cell possibly filled with a *bubble* of some color. In each turn, the player is given a bubble of some color, which can be fired in any (upward) direction from the pointer at the bottom center of the board. The fired bubble travels straight, reflecting off the left and right walls, until it hits another bubble or the top wall, in which case it terminates at the nearest grid-aligned position. If the bubble is now in a connected group of at least three bubbles of the same color, then that group disappears ("pops"), and any bubbles now disconnected from the top wall also pop.

Here we study the perfect-information (generalized) form of Puzzle Bobble. We are given an initial board of bubbles and the entire sequence of colored bubbles that will come. The goal is to clear the board using the given sequence of bubbles. (The actual game has an infinite, randomly generated sequence of bubbles, like Tetris [BDH+04].) The game also has a falling ceiling, where all bubbles descend every fixed number of shots; and if a bubble hits the floor, the game ends. We assume that the resolution of the input is sufficiently fine to hit any discrete cell that could be hit

*MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, edemaine@mit.edu

†Directeur de recherches du F.R.S.–FNRS, Départment d'Informatique, Université Libre de Bruxelles, Brussels, Belgium, stefan.langerman@ulb.ac.be

[1]Spoiler: if you finish Bubble Bobble in super mode in co-op, then the true ending reveals that Bub and Bob are in fact human boys, transformed into brontosauruses by the evil whale Baron Von Blubba [Hun11].

by an (infinitely precise) continuous shot. (This assumption seems to hold in the original game, so it is natural to generalize it.)

**Theorem 1** *Puzzle Bobble is NP-complete.*

Membership in NP is easy: specify where to shoot each of the $n$ given bubbles. The rest of this paper establishes NP-hardness.

Our reduction applies to all versions of Puzzle Bobble. Viglietta [Vig12] proved that Puzzle Bobble 3 is NP-complete, by exploiting "rainbow" (wildcard) bubbles. Our proof shows that this feature is unnecessary.

## 2 NP-Hardness

The reduction is from Set Cover: given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_s\}$ of sets where each $S_i \subseteq U$, and given a positive integer $k$, are there $k$ of the sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ whose union hits all elements of $U$?

Figure 1 shows the overall structure of the reduction. The bulk of the construction is in the central small square, which is aligned on the top side of an $m \times m$ square above the floor. By making the central square small enough, the angles of direct shots at the square are close to vertical (which we will need to solve most gadgets), and the rebound angles that hit the square are all approximately 45° (which we will need to solve the crossover gadget below), even after the ceiling falling caused by the shots in the reduction. The player could do multiple rebounds (or destroy bubbles to cause the ceiling to lower prematurely) to make shot angles more horizontal, but this will only make it harder to solve the gadgets.

### 2.1 Bubble Sequence

The sequence of bubbles given to the player is as follows. The very first color appears only $k$ times, where $k$ is the desired set-cover size. Each remaining color appears sufficiently many times ($\Theta(s|U|)$ times, which we will refer to as $\infty$). Unneeded bubbles can be discarded by forming isolated groups of size 3, 4, or 5 off to the side.

$$
\begin{array}{rrrr}
k \text{ blue}, & \infty \text{ yellow}, & \infty \text{ blue}, & \infty \text{ red}; \\
\infty \text{ blue}, & \infty \text{ yellow}, & \infty \text{ blue}, & \infty \text{ red}; \\
\infty \text{ blue}, & \infty \text{ yellow}, & \infty \text{ blue}, & \infty \text{ red}; \\
\vdots & \vdots & \vdots & \vdots \\
\infty \text{ blue}, & \infty \text{ yellow}, & \infty \text{ blue}, & \infty \text{ red}; \\
\infty \text{ red}, & \infty \text{ red}, & \infty \text{ red}, & \ldots
\end{array}
$$

The rough idea is the following. Red bubbles separate vertical layers that unravel sequentially, as enforced by blue buffers. Blue and yellow bubbles form triggers to communicate signals into the next layers, alternately. Blue triggers cup yellow triggers in the next level, and vice versa.

### 2.2 Gadgets

First we have one instance of the choice gadget, shown in Figure 2, which allows triggering $k$ sets (whichever the player chooses).

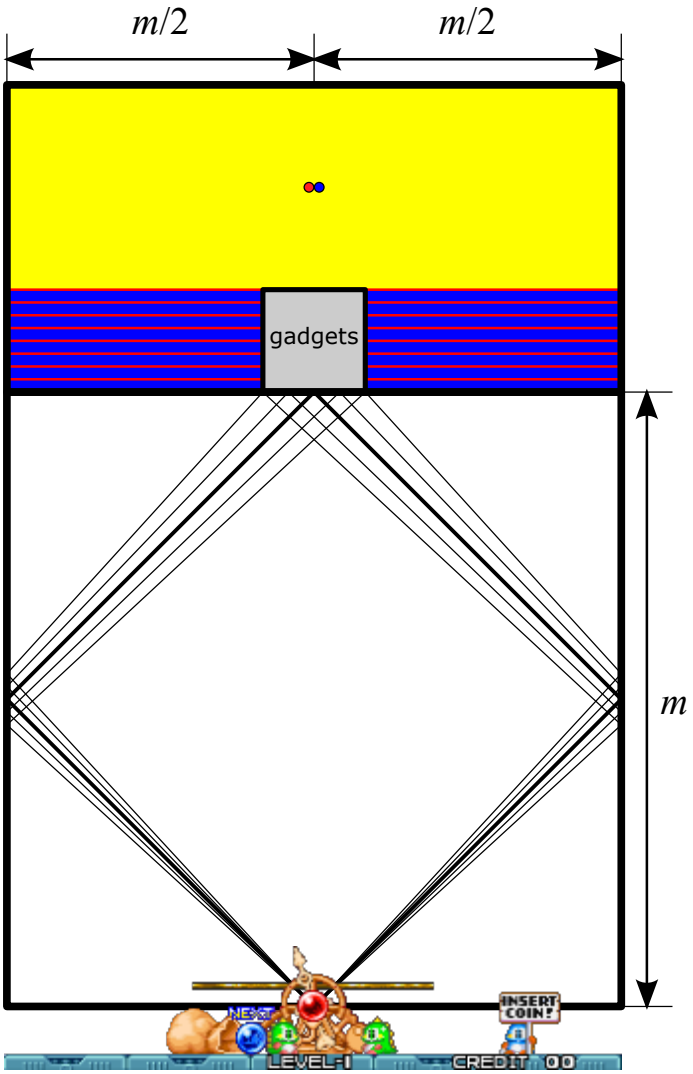


Figure 1: Overall structure of the reduction. All other gadgets lie within a small square at the top of an $m \times m$ square, where $m$ is the width of the game. Red horizontal lines separate the gadgets into layers, with blue fill in between. At the top is a huge rectangle of yellow bubbles with one red bubble and one blue bubble in the middle.

Then we use several split gadgets, shown in Figure 3, to split each trigger for set $S_i$ into $|S_i|$ triggers.

Then we use several crossover gadgets, shown in Figure 4, to bring together all the triggers for element $x$, for every element $x$. More precisely, Figure 4 shows how to copy all other wire values while swapping an adjacent pair. Adjacent swaps suffice to bubble sort $S_1, S_2, \ldots, S_s$ from being in order by set to being in order by element. In fact, we can use the parallel sorting algorithm *odd–even sort* by executing several swaps in one layer, and use only $2\sum_i |S_i|$ layers.

Next, for each element, we merge all the triggers for that element (coming from sets that contain the element), using the OR gadget in Figure 5. In this gadget, any input trigger enables the output trigger. By combining several OR gadgets, we end up with one trigger per element in $U$, indicating whether that element was covered by the $k$ chosen sets.

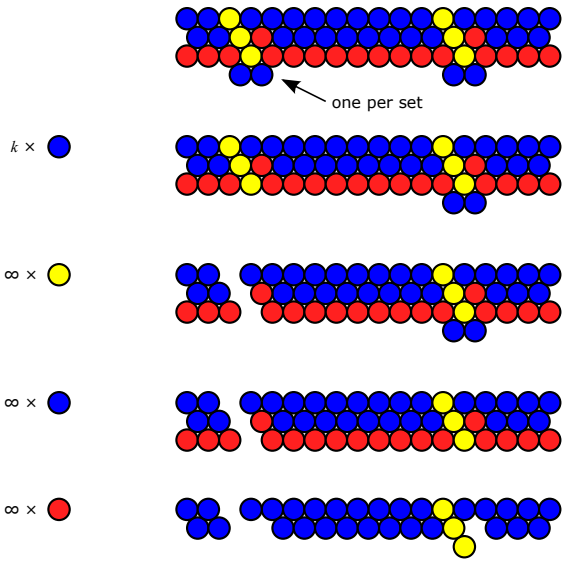


Figure 2: Choice gadget, shown here with $s = 2$ sets. (Left) Behavior of a chosen set. (Right) Behavior of an unchosen set.

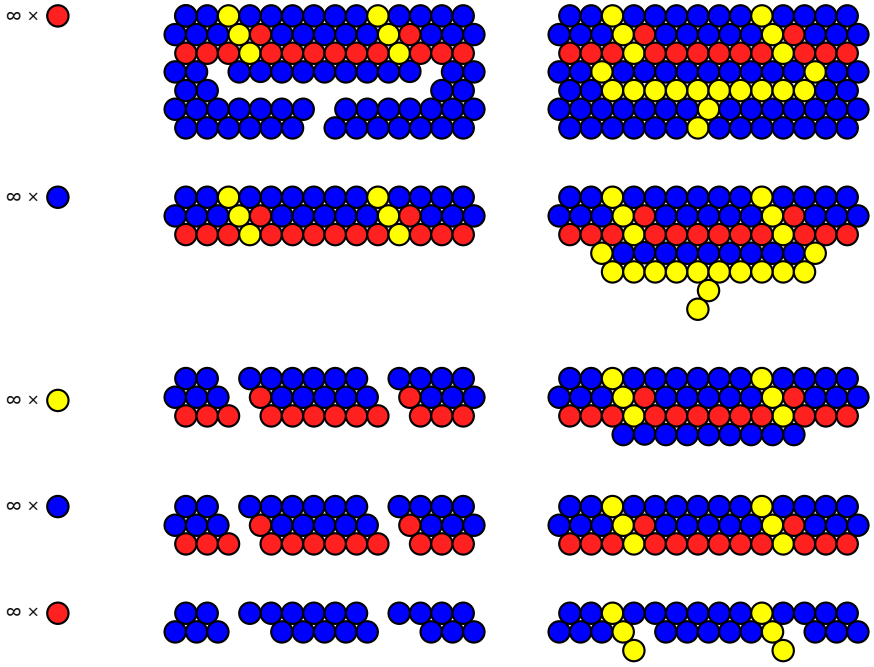


Figure 3: Split gadget. (Left) Behavior of a chosen set. (Right) Behavior of an unchosen set.

Finally, we combine the element triggers using the AND gadget in Figure 6. In this gadget, the output triggers only if all inputs trigger. By combining several AND gadgets, we end up with one trigger indicating that all elements are covered, i.e., we found a set cover of size $k$.

This trigger is connected to a huge ($n^{1-\varepsilon}$-area) rectangle of yellow bubbles at the top of the board, with one red bubble in the middle, as shown in Figure 1. If the yellow triggers, the player wins the game immediately (as the red falls). Otherwise, only red bubbles come, so the player

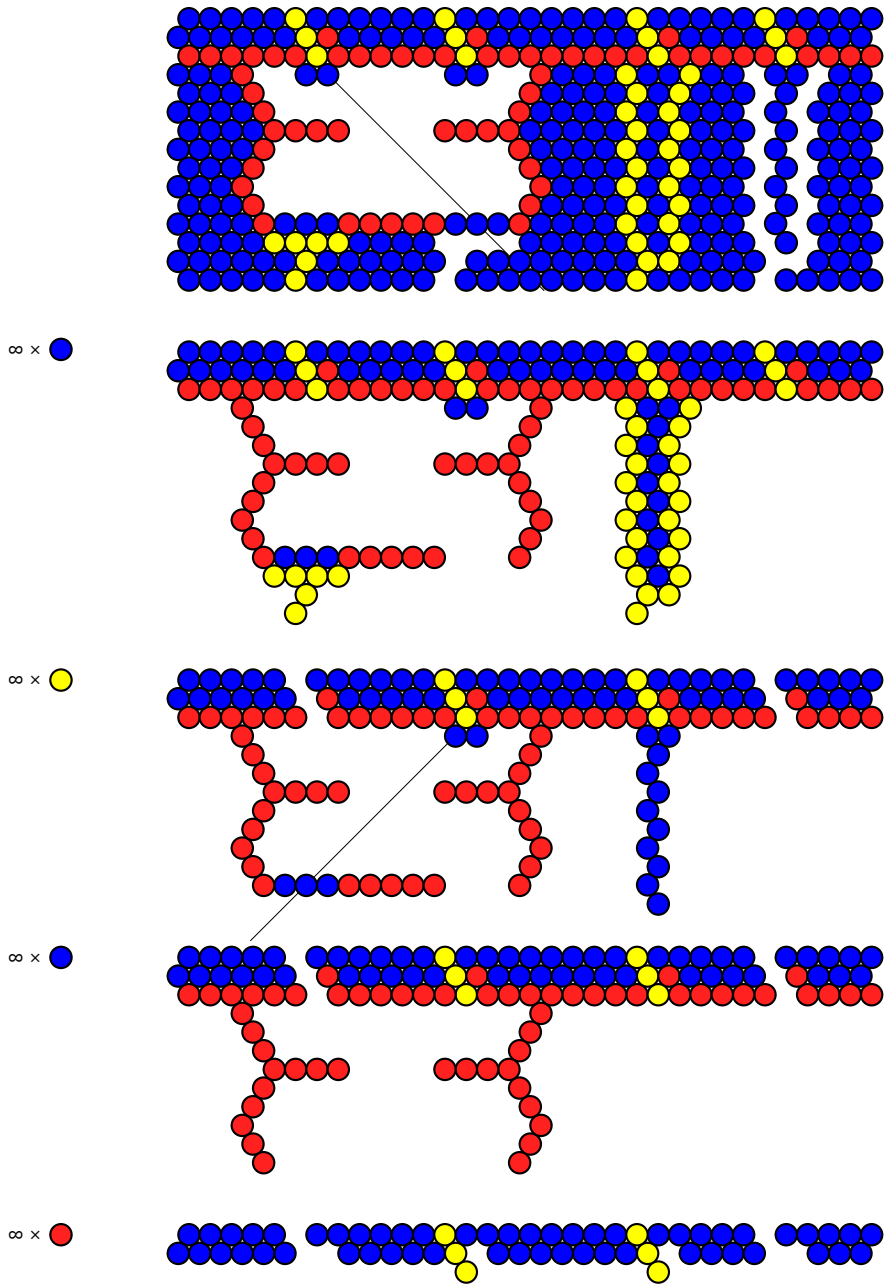


Figure 4: Crossover gadget, shown here with left side inactive and right side active. On the right are other wires whose values are simply copied.

eventually dies when the yellow rectangle reaches the floor. (We include the red and blue bubbles in the middle of the yellow bubbles because, in the actual game, only present bubbles can be presented for shooting, so if there were only yellow bubbles left, the player would get to shoot yellow and win.)

Thus, even approximating the maximum number of poppable bubbles better than a factor of $n^{1-\varepsilon}$ is NP-hard (similar to Tetris [BDH+04]).

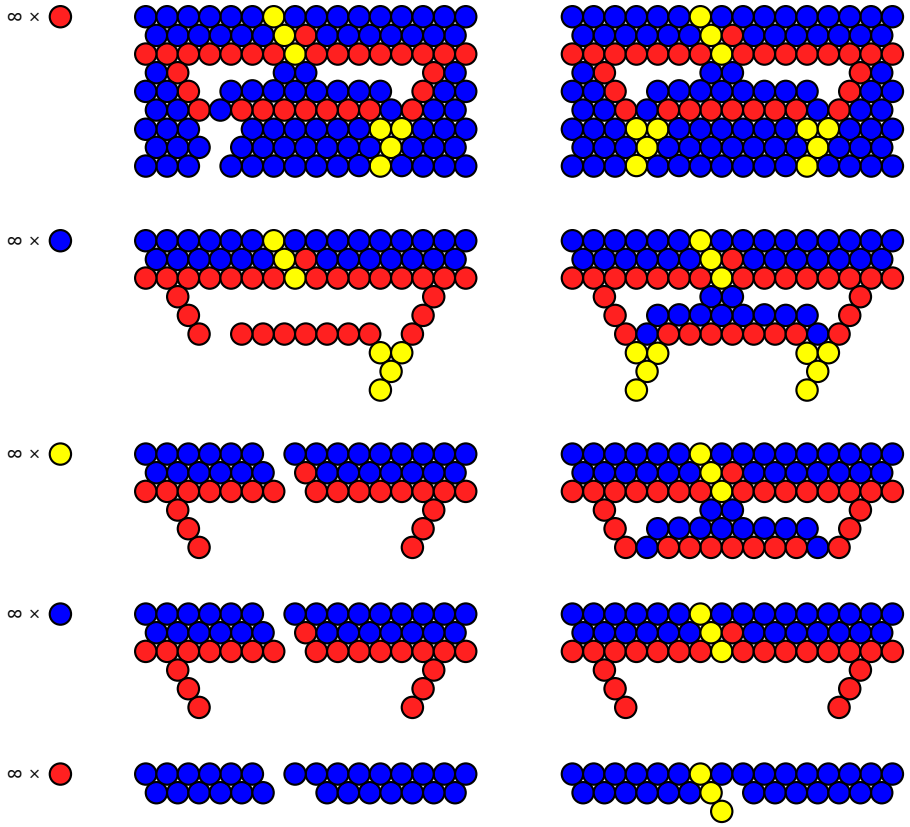


Figure 5: OR gadget. (Left) One input active, triggering output. (Right) No inputs active.

### 2.3 Putting It Together

Figures 7 and 8 show an example of how the gadgets fit together in a real example. In particular, it illustrates how to stretch gadgets horizontally so that their inputs and outputs align, and how to stack the layers of gadgets (each gadget is placed on the row immediately after the previous).

The bijection between solutions of the Puzzle Bobble instance and the Set Cover instance come from which triggers get popped by the first $k$ blue shots on the Choice gadget. (Fewer than $k$ triggers could be popped, corresponding to smaller-than-$k$ set covers.) The correctness follows from the claimed properties of the gadgets, which can be verified from the figures implementing a greedy algorithm of popping all possible bubbles of each provided color (which can only help for these instances).

A key lemma for correctness is that, during the $i$th blue–yellow–blue–red phase of the bubble sequence, only bubbles in the $i$th layer of the construction can be directly popped, with spillover into the next layer only from triggered yellow bubble wires. The $i$th red layer prevents any nonred bubbles from physically reaching the next layer in the $i$th phase, because the gaps between red bubbles are designed to be strictly less than one bubble width. (Precisely, the gap width is $\sqrt{3}-1 \approx 0.73$.) At the end of the phase when firing red bubbles, the blue in the next layer uses the same $< 1$ gaps (when the yellow has been triggered) to prevent any red bubbles from reaching the next layer. So the lemma follows.

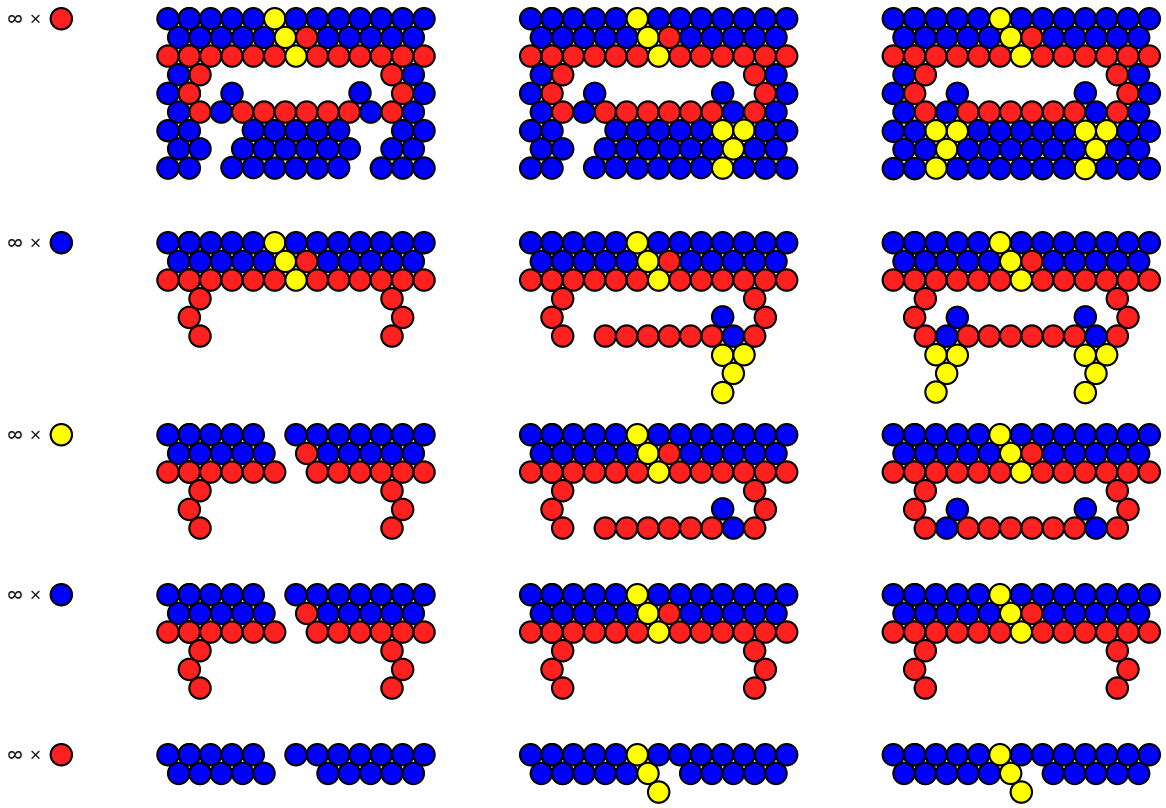


Figure 6: AND gadget. (Left) Two inputs active, triggering output. (Middle) One input active. (Right) No inputs active.

## 3 Open Problems

We have proved NP-hardness for just three colors. What about just two colors? Or even one color?

## Acknowledgments

We thank Giovanni Viglietta for helpful discussions, in particular for pointing out bugs in earlier versions of this proof.

## References

[BDH+04] Ron Breukelaar, Erik D. Demaine, Susan Hohenberger, Hendrik Jan Hoogeboom, Walter A. Kosters, and David Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry and Applications*, 14(1–2):41–68, 2004.

[Hun11] Stuart Hunt. Bubble memories: 25 years of Bubble Bobble. *Retro Gamer*, 95:26–35, 2011.

[Thea] The International Arcade Museum. Bubble Bobble. http://www.arcade-museum.com/game_detail.php?game_id=7222.

[Theb] The International Arcade Museum. Puzzle Bobble. http://www.arcade-museum.com/game_detail.php?game_id=9169.

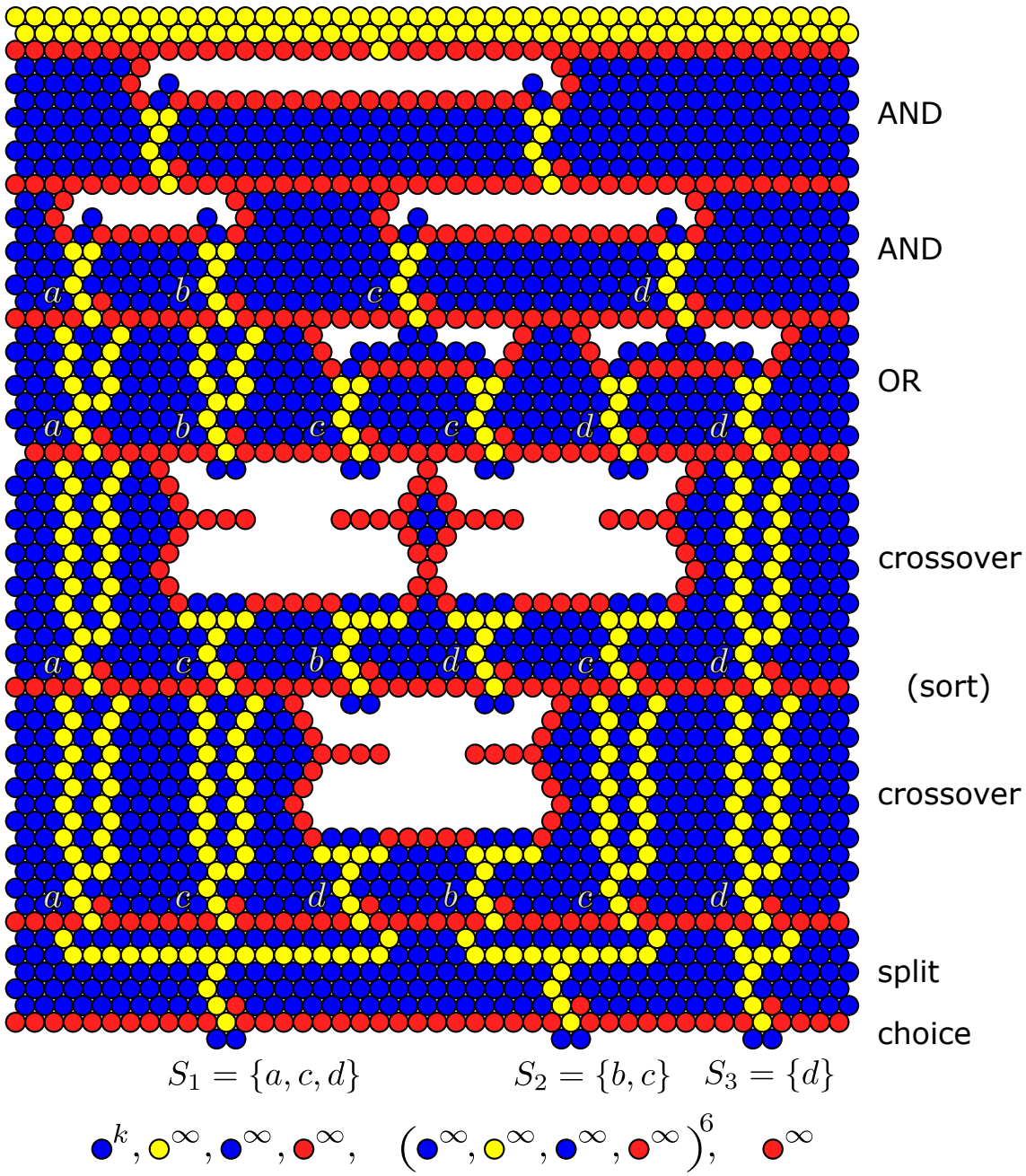


Figure 7: Example of the main construction (the gray box in Figure 1) with three sets and four elements. The bubble sequence at the bottom can solve the puzzle for $k = 2$ and $k = 3$, but not for $k = 1$.


[Vig12] Giovanni Viglietta. Gaming is a hard job, but someone has to do it! In *Proceedings of the 6th International conference on Fun with Algorithms*, pages 357–367, 2012.

[Wik] Wikipedia, The Free Encyclopedia. Puzzle Bobble. http://en.wikipedia.org/wiki/Puzzle_Bobble.

[YDD89] Marvin C. Young, Matt Dike, and Michael Doss. Bust a move. In *Stone Cold Rhymin'*. Delicious Vinyl, 1989.

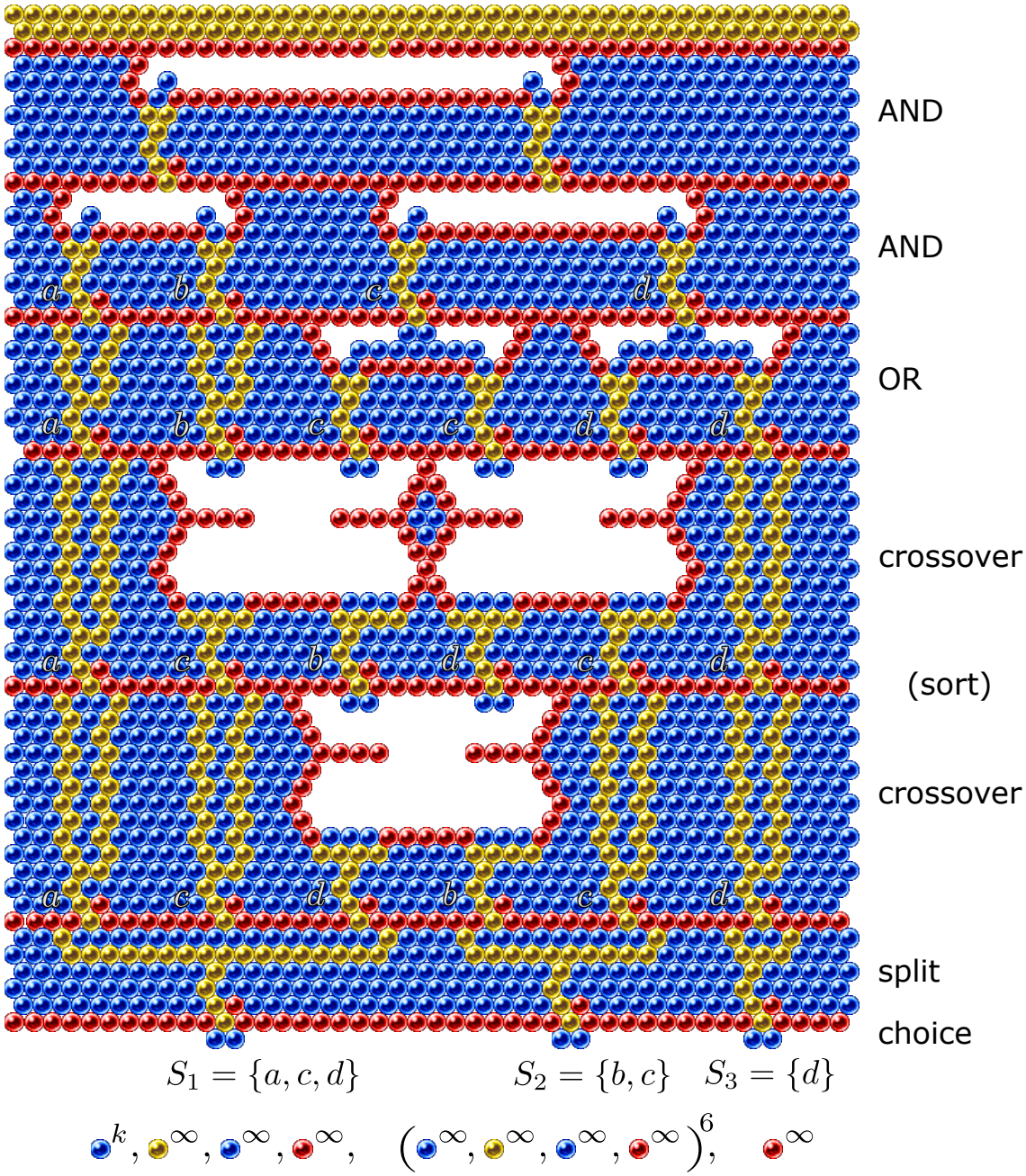


Figure 8: Figure 7 using actual Puzzle Bobble sprites, thanks to The Spriters Resource.