Supplementary Material: Show, Don't Tell: Evaluating Large Language Models Beyond Textual Understanding with ChildPlay

1 LCL

1.1 Prompts

Validity Testing prompt:
"You will receive a description of a Lego structure, for instance, ((x1, y1, 'color1'), (x2, y2, 'color2')), which lists the coordinates and colors of two pieces. A construct is valid if all Lego pieces are connected but not overlapping. A Lego piece is connected through interlocking pegs, not by merely touching sides. Two Lego pieces overlap when they share the same y-coordinate and any part of their length has the same x-coordinate. If the following structure is valid then reply with valid, otherwise reply with invalid (do not justify your answer): <pieces>"

Figure 1: Validity testing prompt.

Construct Generation prompt:
"A description of a Lego structure consists of a list of tuples, ((x1, y1, 'color1'), (x2, y2, 'color2')), where each tuple shows the coordinates and colors of a piece. Such a structure is valid if all Lego pieces are connected but not overlapping. A Lego piece is connected through interlocking pegs, not by merely touching sides. Two Lego pieces overlap when they share the same y-coordinate and any part of their length has the same x-coordinate. Produce a description of a valid structure using <n pieces> Lego pieces. Reply only with the Lego structure description following the format ((x1, y1, 'color1'), (x2, y2, 'color2'), ...), write nothing else but the structure."

Figure 2: Construct generation prompt.

The prompts written in LaTeX from Fig. 1 and Fig. 2 were used both in the case of GPT-3.5 and GPT-4 in the main text. Notably, these tests are part of the ChildPlay suite. Further tests were conducted but not included in the ChildPlay suite and are illustrated herein. The reason why these tests have not been included in the suite is because they must be written as systematic benchmarks instead of experimental input-output segments. Currently, they stand as illustrative cases of spatial reasoning failure and success that supplement the benchmark but are not aimed at proving the model's capacity either way. They are simply an interesting addition.

2 LCL Syntax

2.1 Definitions in LCL_2

A piece P is defined as a tuple $P = (l, w, (x, y), c, h)$ (see Table 1) where:

1. l is the length of the piece, fixed at 4 units;
2. w is the width of the piece, fixed at 2 units;
3. $x - axis$ corresponds to the position of the studs;
4. $y - axis$ corresponds to layers - the first brick is at layer 0;
5. c is the color of the piece;
6. h is the height of the piece, fixed at 1 unit;

For the sake of brevity, in most of the examples below we omit length (l), color (c), and height (h) since these are set as constants.

Parameter	Description	Value
l	Length of the piece	4 units
w	Width of the piece	2 units
(x, y)	Position of the studs (x-axis), layers (y-axis)	Var
c	Colour of the piece	Var
h	Height of the piece	1 unit

Table 1: Definition of a Piece P

A construction, M, is then a valid construction in LCL_2 if and only if it follows the rules:

1. $P = (4, 2, (x, y), c, 1)$
2. M is composed entirely by P pieces ($\Phi = P$);
3. Every piece P must be connected to at least one other piece P;
4. M is symmetric along the line crossing the 2 by 4 pieces, between its pegs, along the piece's longest side;
5. Pieces in the construct can only be manipulated horizontally in $n * pi$ rotations, with $n \in \mathbb{Z}$ (note that this makes width irrelevant);
6. The position of a piece is defined by its left-most pair of studs;
7. M begins with a piece P at coordinates (0,0);
8. All pieces placed in layer n must be placed before any piece is placed in layer $n + 1$;

Consider constructing a line using three bricks (we omit height h since it is a constant, with value equal to 1). This is counter-intuitive, but note that a line cannot be represented as in Fig 5, because the pieces are disconnected.

LCL_2: $((0, 0), (4, 0), (8, 0))$ is then an example of what one expects to see as representing a line, but it is not valid in LCL. Because the pieces are disconnected from eachother, they just lay next to eachother, one after another in a row. Instead, $((0, 0), (4, 0), (2, 1))$, or $((0, 0), (-2, 1), (2, 1))$, or even $((0, 0), (-2, 1), (4, 1))$ would be valid constructs.

Subsequently, both models were prompted with several additional requests that have not been integrated in the suite yet (see Table 2).

For these experiments, the definition of LCL was provided to the model and it was accompanied by the prompt in Fig. 3.

Prompt: *"I will give you a number of pieces, I will ask you for a shape and you'll output the coordinates per piece to form such a shape. It must be valid in LCL."*

Figure 3: Extra testing prompts not in the suite.

Task	Description
Triangle Construction	*"Make a triangle with 5 bricks."*
Humanoid Figure	*"6 pieces. Build a humanoid figure."*
Bart Simpson-Like Figure	*"Let me help you. Imagine it's Bart Simpson. You have three yellow pieces, one for the head, two for the arms, one red for the torso, and two blue pieces for the legs."*
Tower Construction	*"Produce now a tower with 3 bricks."*

Table 2: Sequence of building prompts.

2.2 Example

A simple example is found in Fig 4. This is a tower constructed from 3 bricks and is a valid LCL_2 construct.

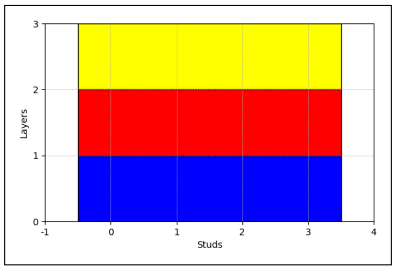


Figure 4: A *valid* tower representation in LCL_2.

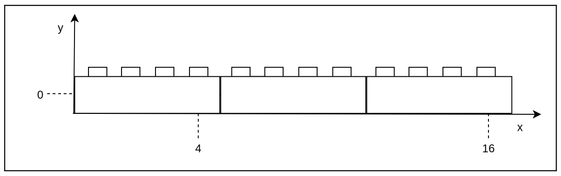


Figure 5: A disconnected line of bricks is not a valid construct in LCL_2: $\{(0,0),(4,0),(8,0)\}$.

This sequence forms the construction of a 3-brick line, each brick having a width of 4 units. But since this construction is composed of three columns, one piece P each, it can be broken apart and is not a topological object (each piece can be moved individually). The **correct** construct with three bricks has many possible solutions. For a centre piece with two pieces on the bottom or two pieces on the top, we find 24 possible solutions. In eq. 1 is the general formula with s being the amount of studs:

$$\begin{aligned} f(0) &= 0 \\ f(s) &= 4*(s-1)+f(s-1) \end{aligned} \tag{1}$$

And its non-recursive form:

$$\begin{aligned} f(0) &= 0 \\ f(s) &= 2(s-1)s \end{aligned} \tag{2}$$

We show two more simple examples:

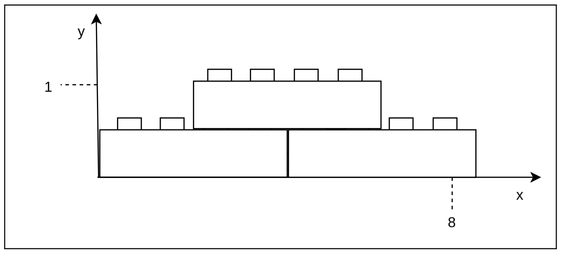


Figure 6: A possible representation of the requested "line" as a valid construct in LCL_2: $\{(0,0),(4,0),(2,1)\}$.

, and:

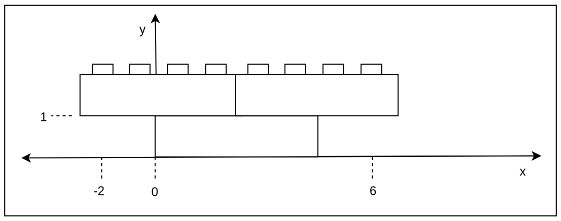


Figure 7: Another possible representation of the requested "line" as a valid construct in LCL_2: $\{(0,0),(-2,1),(2,1)\}$.

The "three-in-a-line" can only be loosely interpreted in LCL_2, due to rule (2) - that pieces cannot be moved independently from the rest of the model. For this reason, one can imagine many more structures that loosely fall under the definition of a "line" or "wall", for example:

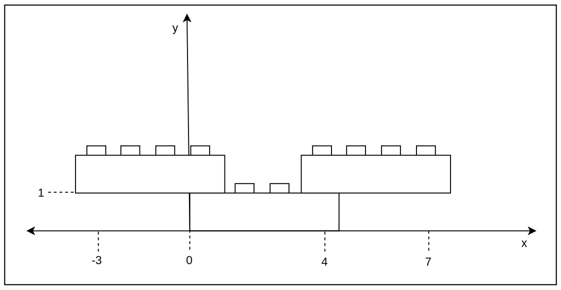


Figure 8: Another possible construct for the requested "line". This is also a valid construct in LCL_2: $\{(0,0),(-2,1),(4,1)\}$.

Or even a stair-like structure:

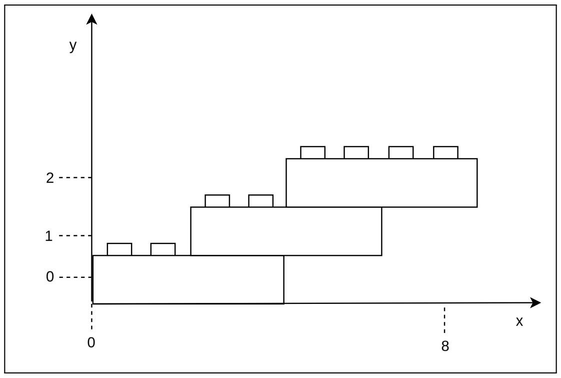


Figure 9: Stair-like construct for the requested "line". This is also a valid construct in LCL_2: $\{(0,0),(2,1),(4,2)\}$.

A humanoid could also be easily represented in LCL_2 as:

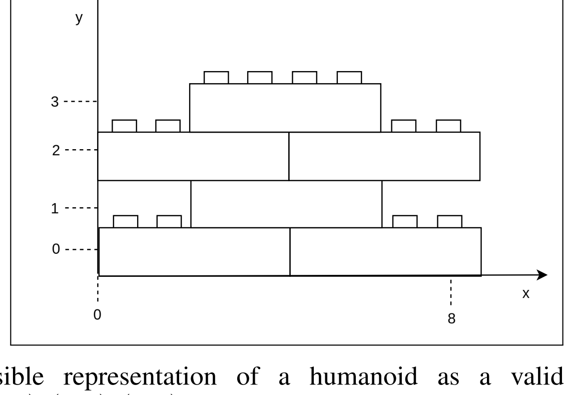
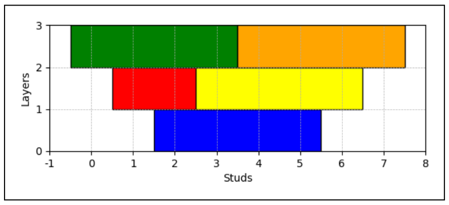
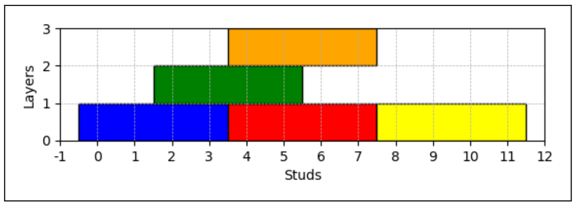


Figure 10: A possible representation of a humanoid as a valid construct in LCL_2: $\{(0,0),(4,0),(2,1),(0,2),(4,2),(2,3)\}$.

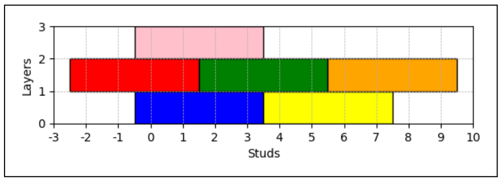


(a) GPT-3.5.

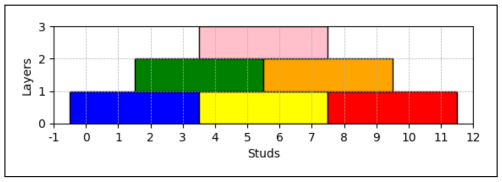


(b) GPT-4.

Figure 11: Model responses to the query: *"Make a triangle with 5 bricks."*, randomised colours. This is impossible to satisfy.

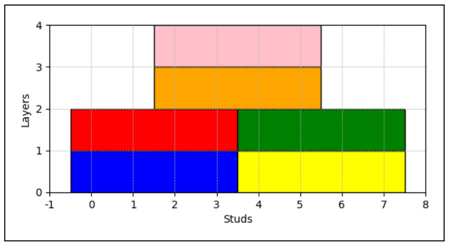


(a) GPT-3.5.

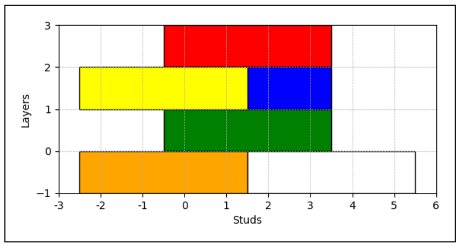


(b) GPT-4.

Figure 12: Model responses to the query: *"Make a triangle with 6 bricks."*, randomised colours.

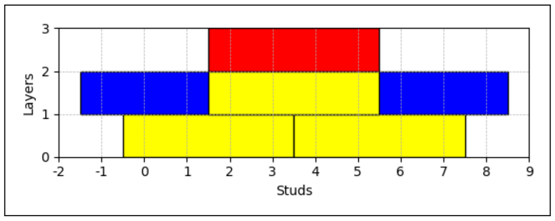


(a) GPT-3.5

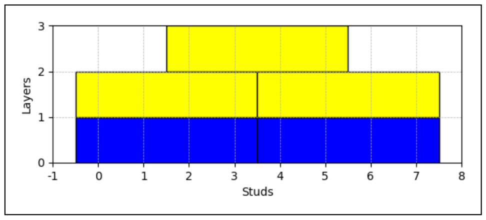


(b) GPT-4

Figure 13: Model responses to the query: *"You have 6 pieces. Build a humanoid figure."*, randomised colours.



(a) GPT-3.5



(b) GPT-4

Figure 14: Model response to the query: *"Imagine it's bart simpson. You have three yellow pieces, one for the head, two for the arms, one red for the torso, and two blue pieces for the legs."* - colours specified.

We show the model's incorrect answers in Figs. 11, 13, and 14 and correct answers in Figs. 12b. Essentially, both GPT-3.5 and GPT-4 were not far from the expected target, but failed to respect LCL_2 rules in most cases. For example, pieces are found in an impossible superposition in Fig. 11a (red piece is in the same position as yellow piece), 13b (blue piece is in the same position as yellow piece), and 14b (red piece is in the same position as middle yellow pieces). In Fig. 14a, GPT-3.5 erroneously swapped the middle yellow piece with the red piece and the blue pieces with the bottom yellow pieces, even though it first declared in plain English the correct organisation of the 6 pieces. The positive result is that models manage to assemble a tower of three pieces and GPT-4 was capable of assembling a triangle (see Table 3). None of the models recognised that they were asked an impossible task, namely building a triangle with only 5 pieces (see Fig. 11).

Model		Responses	
Category	N(P)	GPT-3.5	GPT-4
Tower	3	Correct	Correct
Impossible Triangle	5	Incorrect	Incorrect
Triangle	6	Incorrect	Correct
Humanoid	6	Incorrect	Incorrect
Bart Simpson	6	Incorrect	Incorrect

Table 3: Comparison of Responses by GPT-3.5 and GPT-4.

2.3 Small Dataset for Future Experiments

The dataset defined herein contains several example prompts that are more complex and do not follow the 2x4 assumption, each consisting of a request followed by a LEGO kit of fewer than 15 pieces to which the agent is bound.

LEGO Kits

Apple

Possible prompt: *"Construct a LEGO apple with a mix of red and green colors, resembling a typical apple shape using slopes and bricks."*

- Green Slope 45 2 x 1 - Code: 3040 (Quantity: 1)
- Red Slope 45 2 x 2 - Code: 3039 (Quantity: 2)
- Lime Slope, Inverted 45 2 x 2 - Code: 3660 (Quantity: 2)
- Red Brick 2 x 3 - Code: 3002 (Quantity: 1)
- Lime Plate 2 x 2 - Code: 3022 (Quantity: 1)
- Lime Brick 1 x 2 - Code: 3004 (Quantity: 1)

Yellow Hut

Possible prompt: *"Build a hut with a purple and yellow color scheme, featuring a simple structure and a sloped roof."*

- Trans-Clear Brick 1 x 2 without Bottom Tube - Code: 3065 (Quantity: 2)
- Medium Nougat Brick 2 x 2 - Code: 3003 (Quantity: 1)
- Lime Plate 2 x 6 - Code: 3795 (Quantity: 1)
- Bright Light Yellow Brick 1 x 2 - Code: 3004 (Quantity: 4)
- Bright Light Yellow Brick 2 x 2 - Code: 3003 (Quantity: 1)
- Medium Lavender Slope 45 2 x 2 - Code: 3039 (Quantity: 4)

Fortress

Possible prompt: *"Create a medieval-themed LEGO fortress with arches, walls, and defensive structures, symbolizing a stronghold."*

- Green Plate 2 x 8 - Code: 3034 (Quantity: 1)

- Light Bluish Gray Arch 1 x 4 x 2 - Code: 6182 (Quantity: 2)
- Sand Green Brick 1 x 2 - Code: 3004 (Quantity: 2)
- Light Bluish Gray Brick 1 x 2 - Code: 3004 (Quantity: 2)
- Dark Bluish Gray Brick 1 x 2 - Code: 3004 (Quantity: 2)
- Light Bluish Gray Brick 2 x 2 - Code: 3003 (Quantity: 1)
- Reddish Brown Brick, Round 1 x 1 Open Stud - Code: 3062b (Quantity: 2)

Dinghy

Possible prompt: *"Assemble a small LEGO dinghy with a white sail and a mast."*

- Dark Tan Plate 2 x 4 - Code: 3020 (Quantity: 1)
- Tan Slope, Inverted 33 3 x 2 with Flat Bottom Pin and Connections - Code: 3747b (Quantity: 1)
- White Slope 45 2 x 2 - Code: 3039 (Quantity: 3)
- White Brick 2 x 2 - Code: 3003 (Quantity: 1)
- White Brick 1 x 2 - Code: 3004 (Quantity: 1)
- Tan Brick 2 x 3 - Code: 3002 (Quantity: 1)
- Reddish Brown Brick, Round 2 x 2 with Axle Hole - Code: 3941 (Quantity: 1)

Blue Bot

Possible prompt: *"Construct a LEGO robot with a humanoid structure, featuring a distinguishable head, body, arms, and legs."*

- Medium Blue Brick 2 x 2 - Code: 3003 (Quantity: 1)
- Brick, Modified 2 x 3 with Curved Top - Code: 6215 (Quantity: 1)
- Brick 2 x 4 - Code: 3001 (Quantity: 1)
- Brick 1 x 2 - Code: 3004 (Quantity: 2)
- Brick, Round 2 x 2 with Grille - Code: 92947 (Quantity: 1)
- Plate 2 x 2 - Code: 3022 (Quantity: 1)
- Brick, Modified 1 x 2 with Studs on 1 Side - Code: 11211 (Quantity: 1)
- Brick 1 x 2 without Bottom Tube - Code: 3065 (Quantity: 1)
- Tile 1 x 1 Round - Code: 98138 (Quantity: 1)
- Brick, Round 2 x 2 Dome Top, with Bottom Axle Holder - Code: 553c (Quantity: 1)

Toy Car

Possible prompt: *"Build a LEGO toy car with a compact design, featuring wheels, and a sloped windshield."*

- Brick 2 x 6 - Code: 2456 (Quantity: 1)
- Slope 2 x 2 45° - Code: 3039 (Quantity: 1)
- Brick 1 x 2 without Bottom Tube - Code: 3065 (Quantity: 1)
- Brick 1 x 2 - Code: 3004 (Quantity: 1)
- Plate 2 x 2 with Wheel Holders - Code: 4600 (Quantity: 2)
- Wheel 8mm D. x 6mm with Slot - Code: 34337 (Quantity: 4)
- Tire Offset Tread Small - Band Around Center of Tread - Code: 87414 (Quantity: 4)

Goldfish

Possible prompt: *"Create a LEGO goldfish with fins and tail, featuring elements for eyes."*

- Brick 2 x 4 - Code: 3001 (Quantity: 2)
- Brick 1 x 2 with Pin Hole - Code: 3700 (Quantity: 1)
- Brick, Modified 1 x 2 with Studs on 1 Side - Code: 11211 (Quantity: 2)
- Brick 2 x 3 - Code: 3002 (Quantity: 1)
- Slope 45° 2 x 2 - Inverted - Code: 3660 (Quantity: 1)
- Slope 2 x 1 - 45° - Code: 3040 (Quantity: 4)
- Tile 1 x 1 Round with Eye Pattern - Code: 98138pb007 (Quantity: 2)
- Slope 30° 1 x 2 x 2/3 - Code: 85984 (Quantity: 1)

Baby Elephant

Possible prompt: *"Assemble a LEGO baby elephant with a focus on its trunk, ears, and body structure."*

- Brick 2 x 6 - Code: 2456 (Quantity: 1)
- Brick 1 x 2 - Code: 3004 (Quantity: 3)
- Brick 1 x 4 - Code: 3010 (Quantity: 1)
- Brick 1 x 1 with Stud on 1 Side - Code: 87087 (Quantity: 2)
- Tile 1 x 1 Round with Eye Pattern - Code: 98138pb027 (Quantity: 2)
- Brick 2 x 4 - Code: 3001 (Quantity: 1)

Flamingo

Possible prompt: *"Construct a LEGO flamingo with pink bricks, designed to stand on one leg and feature a long neck and beak."*

- Brick 1 x 2 - Code: 3004 (Quantity: 3)
- Brick, Modified 2 x 3 with Curved Top - Code: 6215 (Quantity: 2)
- Brick 1 x 1 with Stud on 1 Side - Code: 87087 (Quantity: 2)
- Plate 2 x 3 - Code: 3021 (Quantity: 1)
- Slope 2 x 2 - 45° - Code: 3039 (Quantity: 1)
- Tile 1 x 1 Round with Eye Closed Pattern - Code: 98138pb028 (Quantity: 2)

Twin Engine Airplane

Possible prompt: *"Build a LEGO twin-engine airplane, with a body, wings, and a tail."*

- Plate 2 x 8 - Code: 3034 (Quantity: 2)
- Brick 1 x 2 x 2 with Inside Stud Holder - Code: 3245c (Quantity: 1)
- Brick, Modified 1 x 1 x 1 2/3 with Studs on 1 Side - Code: 32952 (Quantity: 2)
- Brick 1 x 4 with 4 Studs on 1 Side - Code: 30414 (Quantity: 2)
- Slope 2 x 2 - 45° - Code: 3039 (Quantity: 1)
- Brick 1 x 2 without Bottom Tube - Code: 3065 (Quantity: 1)

3 Three Board Games: Tic-Tac-Toe, Connect-Four, and Battleship

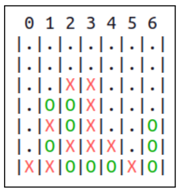
3.1 Prompts

Game	Introductory Prompt
Battleship	*"Battleship is a two-player guessing game where each player has a fleet of ships on a secret grid and then takes turns guessing the locations of the opponent's ships. The objective is to sink all of the opponent's ships by correctly guessing their locations. O's in a board mean that the player selected a square to attack and there was no ship there - it's a miss. Had there been a ship there, instead of an O you would see an X. In your board, an <S> signifies a ship position, and a <~> signifies the sea. Your input is just two numbers with a space in between, one for the row (from 0 to <self.board_size-1>) and one for the column (from 0 to <self.board_size-1>), like: 0 0, nothing else. Do not output anything else but the row col values."*
Tic-Tac-Toe	*"Tic-Tac-Toe is a two-player game played on a 3x3 grid. Players take turns placing their mark, X or O, in an empty square. The first player to place three of their marks in a horizontal, vertical, or diagonal row wins the game. You will play as player 1, therefore you play with X while your adversary plays with the symbol O. Your input is then a number (from 0 to 2) for the row followed by a space and another number (from 0 to 2) for the column, nothing else. Do not output anything else but the row col values else you lose."*
Connect-Four	*"Connect-Four is a two-player game. The pieces fall straight down, occupying the next available space within a column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. In a board, player 1, you, plays with symbol X, while player 2, your opponent, plays with symbol O. Your input is just a number from 0 to 6, nothing else. Do not output anything else but the col value else you lose."*

Table 4: The three introductory prompts used for the board games in the ChildPlay suite.

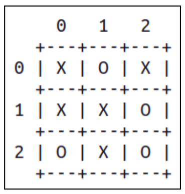
3.2 Example

Note that in the case of Connect-Four, a move consists of a singular scalar. A board state is shown after each play. Examples can be found in Fig. 15.



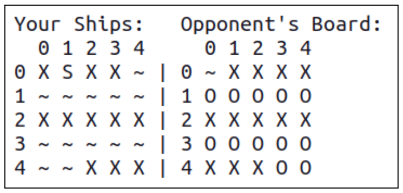
```
 0 1 2 3 4 5 6
|.|.|.|.|.|.|.|
|.|.|.|.|.|.|.|
|.|.|X|X|.|.|.|
|.|O|O|X|.|.|.|
|.|X|O|X|.|.|O|
|.|O|X|X|X|.|O|
|X|X|O|O|O|X|O|
```

(a) Player X has won the Connect-Four game.



```
   0   1   2
 +---+---+---+
0 | X | O | X |
 +---+---+---+
1 | X | X | O |
 +---+---+---+
2 | O | X | O |
 +---+---+---+
```

(b) Players tied at the Tic-Tac-Toe game.



```
Your Ships:   Opponent's Board:
  0 1 2 3 4     0 1 2 3 4
0 X S X X ~ | 0 ~ X X X X
1 ~ ~ ~ ~ ~ | 1 O O O O O
2 X X X X X | 2 X X X X X
3 ~ ~ ~ ~ ~ | 3 O O O O O
4 ~ ~ X X X | 4 X X X O O
```

(c) A player has won the Battleship game.

Figure 15: Examples of final board states in the three different board games.

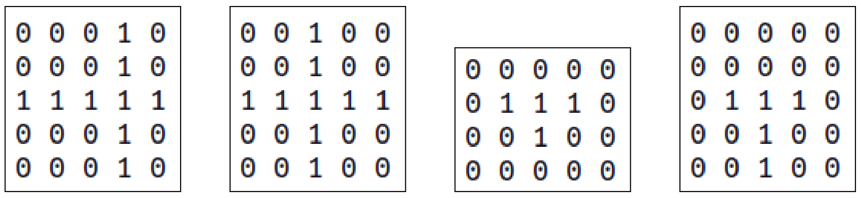
4 Shapes - short experiments (not included in ChildPlay)

4.1 Prompts

Test	Prompt
Introductory prompt	*"Below is a 15 by 15 grid of 0s. I have flipped some 0s into 1s such that a basic geometrical shape has formed. Can you tell me what shape it is?"*
Square (feedback)	*"That's incorrect. The shape is a square. Can you tell me the length and width?"*
Circle (feedback)	*"That's incorrect. The shape is a circle. Can you tell me the coordinates of the center?"*
Triangle (feedback)	*"That is incorrect. It is in fact a triangle. Can you tell the length of the base?"*
Cross A	*"Can you tell me the coordinates of the center of the cross and the length of each line, horizontal and vertical?"*
Cross B	*"Draw a cross in a 5 by 5 grid, with horizontal and vertical axes of 3 units of length with the center at (3,3)."*

Table 5: Introductory and correction prompts for identifying and detailing specific geometrical shapes in a grid environment.

In the shape detection tests, both GPT-3.5 and GPT-4 demonstrated limited comprehension and ability to accurately interpret or draw shapes. When tasked with drawing a cross (see Fig. 16), GPT-3.5 and GPT-4 initially failed to produce a correct cross, but slightly improved after feedback. In Table 6, both models often misidentified or misrepresented the shapes requested, such as describing a circle as a "diamond shape" (GPT-3.5) and an "arrow pointing upwards" (GPT-4). Additionally, neither model could fully comprehend geometric properties, frequently providing incorrect dimensions and centers for squares, triangles, and crosses.



```
0 0 0 1 0
0 0 0 1 0
1 1 1 1 1
0 0 0 1 0
0 0 0 1 0
```

(a) GPT-3.5's, before receiving feedback.

```
0 0 1 0 0
0 0 1 0 0
1 1 1 1 1
0 0 1 0 0
0 0 1 0 0
```

(b) GPT-3.5's, after receiving feedback.

```
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
```

(c) GPT-4's, before receiving feedback.

```
0 0 0 0 0
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 1 0 0
```

(d) GPT-4's, after receiving feedback.

Figure 16: Querying the models to draw a cross with side length of 3 on a 5 by 5 matrix with center at $(3, 3)$.

Test	Query	Correct Answer	GPT-3.5 Response	GPT-4 Response
Circle	Shape Center	Circle (7,7)	"diamond shape" "(7,7)"	"arrow pointing upwards" "(7,7)"
Square	Shape Dimensions	Square (3,4)	"square" "(4,4)"	"'O'" "(3,3)"
Triangle	Shape Base Length	Triangle 7 units	"diamond" "7"	"arrow pointing upwards" "6"
Cross	Shape Center Line Lengths	Cross (5,5) 5	"square" "(7,7)" "5"	" 'plus' sign (+)" "(6,5)" "4"

Table 6: Comparison of Responses by GPT-3.5 and GPT-4 in Shape Detection Tests

5 Conway's Game of Life - a short experiment in state prediction (not included in ChildPlay)

We were interested in seeing if LLMs could predict states based on very simple rules. Initially we tested this by generating sequential states in Conway's Game of Life and feeding them to GPT-3.5 and GPT-4 while prompting the models for two things: the rules and the next state. We do not include this as a benchmark because the experiment required hand segmentation of areas of interest in simulated states and repeatability was not achieved except for in the patterns of interest, of course. We hope to include a version of this task later on in the ChildPlay suite.

In Conway's Game of Life, a cellular automaton devised by mathematician John Horton Conway and introduced to the public by Gardner, cells perpetuate or perish given certain simple rules. We simulated rule B3/S23 also known as "blinking" in varying board sizes. In this rule, a cell is born if it has exactly three live neighbours and survives if it has either two or three live neighbours, dying otherwise. Rule B3/S23 is known to generate a behaviour exhibited by certain configurations of cells that alternate between two or more states over successive generations. These configurations are generally known as oscillators, which are a type of pattern in the game that returns to its initial configuration after a fixed number of generations, repeating indefinitely. In our case, we prompted the models with three configurations of increasing complexity. One consisting of three iterations of the oscillator (3 horizontal cells and 3 vertical cells), another with the same oscillator in the same position but now noise was added in the shape of 4 cells that did not change. Finally, a 5-iteration long pattern was used showcasing cells moving in a circle formation outwardly across the board. These were obtained by repeated simulation using custom software developed specifically for this study.

5.1 Prompts

In the Conway's Game of Life tests, neither GPT-3.5 nor GPT-4 managed to consistently identify or predict the evolving patterns correctly. Table 8 summarizes their performance, where both models only succeeded in identifying a simple blinking pattern. In more complex scenarios involving patterns before or after a given state, both models returned incorrect responses. Even when explicitly provided with the game's rules, GPT-3.5 and GPT-4 failed to accurately predict the next pattern or the pattern before.

Test	Introductory Prompt	Second Prompt
Test 1: GPT-4 and GPT-3.5	*"The following was produced using Conway's Game of Life, namely rule 'survival': [2, 3], 'birth': [3]. Print the state prior to the first one and the one after the last one."*	
Test 2: GPT-4 and GPT-3.5	*"Print the next pattern. The following are two iterations of the game of life by Conway. You cannot use code"*	*"That's wrong. The rule is Rule: 'Blinking': 'survival': [2, 3], 'birth': [3]. Try again."*
Test 3: GPT-4 and GPT-3.5	*"Print the next pattern. The following are two iterations of the game of life by Conway"*	*"That's wrong. The rule is Rule: 'Blinking': 'survival': [2, 3], 'birth': [3]. Try again."*
Requesting Rules: GPT-3.5 and GPT-4	*"I will show you two iterations of Conway's game of life. The first generated the second. You must deduce the survival and birth rules. You must only print these rules, nothing else. Understood?"*	

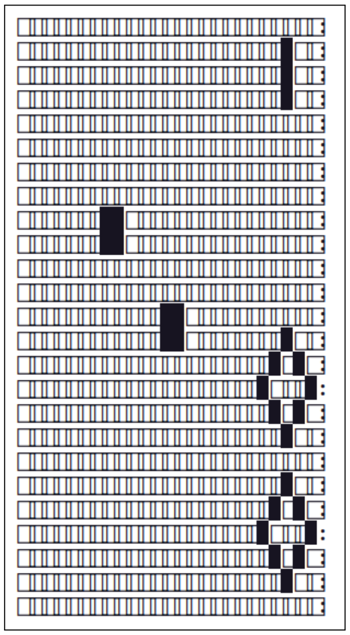
Table 7: Prompts for tests related to Conway's Game of Life.

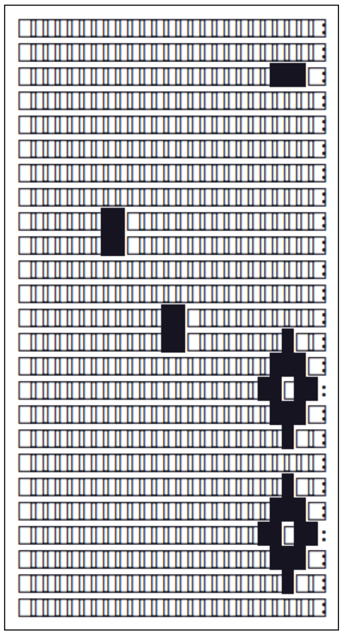
Test	Description	Query[1]	GPT-3.5 Response	GPT-4 Response
Test 1	Blinking pattern	Identify the rule	Correct	Correct
Test 2	Blinking pattern	Next pattern (no rule)	Incorrect	Incorrect
	Blinking pattern[1]	Next pattern[1]	Incorrect	Incorrect
Test 3	Complex pattern	Pattern before [1]	Incorrect	Incorrect
	Complex pattern	Pattern after [1]	Incorrect	Incorrect

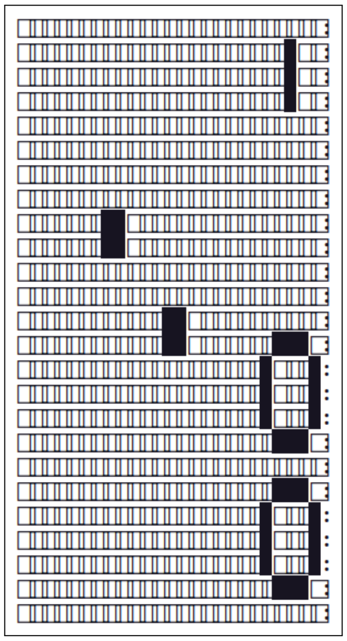
Table 8: Evaluation of GPT-3.5 and GPT-4 Responses in Conway's Game of Life Test Scenarios

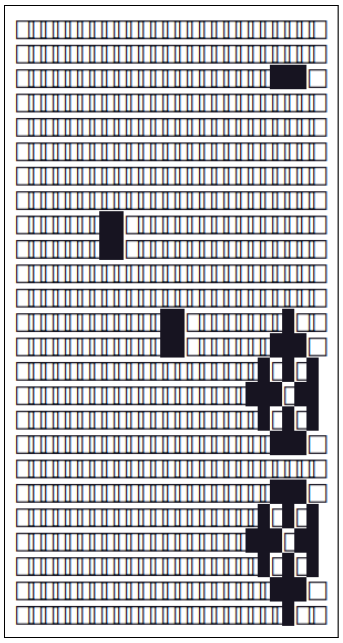
[1]Queries conducted with the explicit rule revealed.

5.2 Test 1


(a) First iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.


(b) Second iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.


(c) Third iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.


(d) Fourth iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.

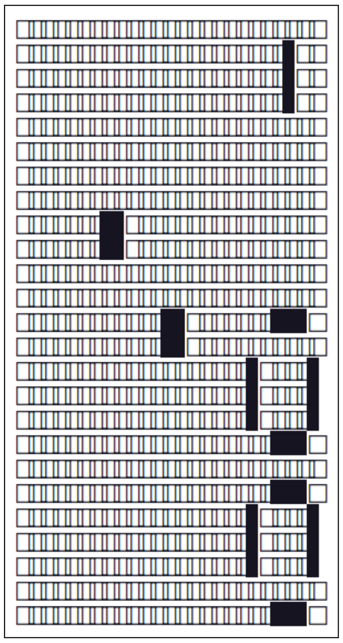
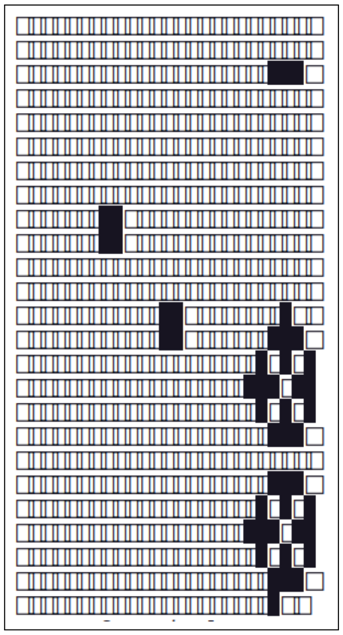
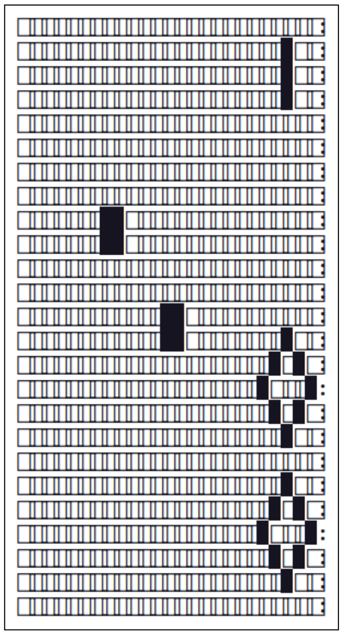

(e) Final iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.

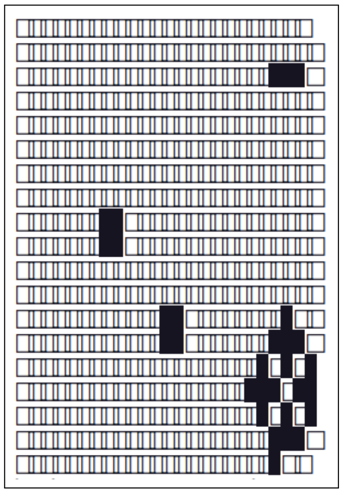
Figure 17: Sample taken from 100 iterations of rule B3/S23.



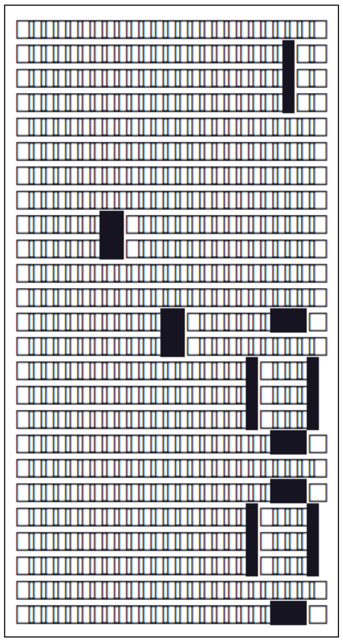
(a) GPT-3.5's guess of the first iteration after seeing the three iterations that follow.



(b) First iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.



(c) GPT-3.5's guess of the final iteration after seeing the three iterations prior.



(d) Final iteration of the 5 iterations generated using rule B3/S23 of Conway's Game of Life.

Figure 18: Prompting GPT-3.5 for the first and last iteration of a 5-sequence long sample from rule B3/S23 of Conway's Game of Life after showing the middle 3 iterations.

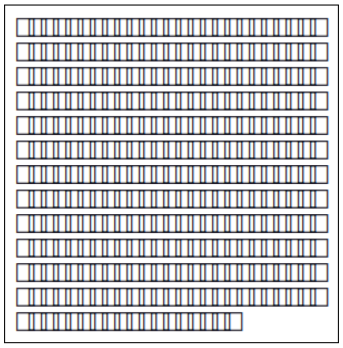


Figure 19: Prompting GPT-4 for the first and last iteration of a 5-sequence long sample from rule B3/S23 of Conway's Game of Life after showing the middle 3 iterations. The model failed to produce an image for the last iteration and only half-completed the first iteration. This is after several trial runs.

5.3 Test 2

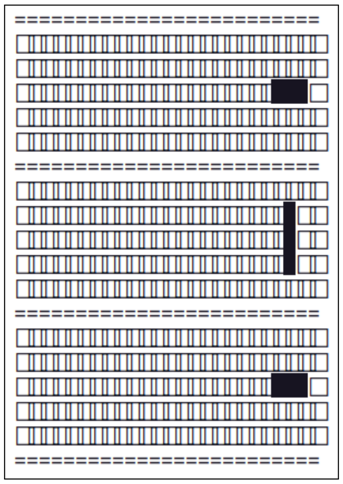
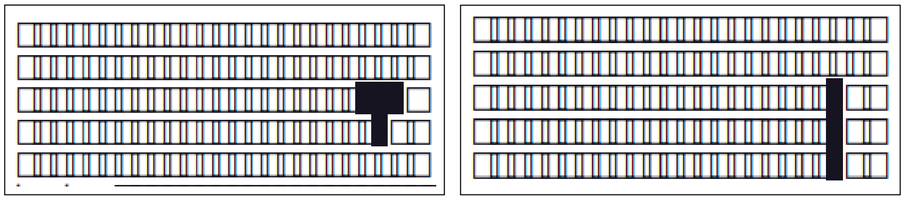
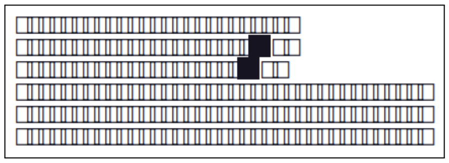


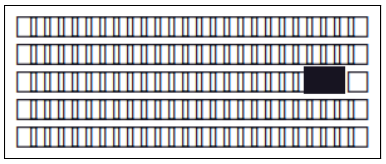
Figure 20: Test 2 consisting of three iterations of a 'blinking' or 'flashing lights' object generated by rule B3/S23.



(a) GPT-3.5's guess of the iteration after seeing the first two iterations of test 2.

(b) GPT-3.5's guess of the iteration after seeing the first two iterations of test 2 and receiving feedback including the clue about rule B3/S23.


(a) GPT-4's guess of the iteration after seeing the first two iterations of test 2.


(b) GPT-4's guess of the iteration after seeing the first two iterations of test 2 and receiving feedback including the clue about rule B3/S23.

5.4 Test 3

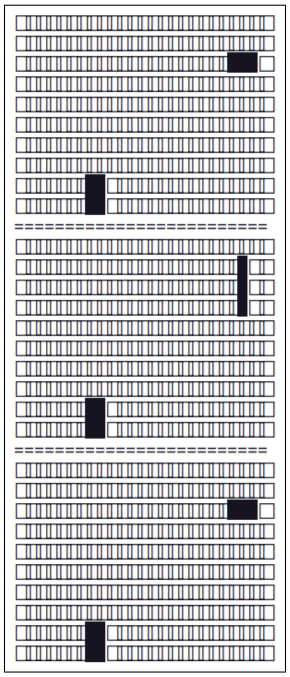
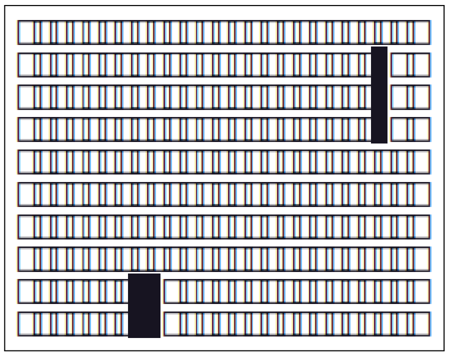
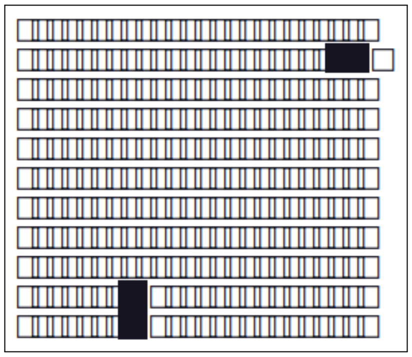
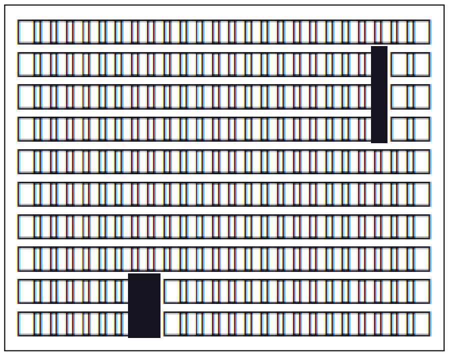


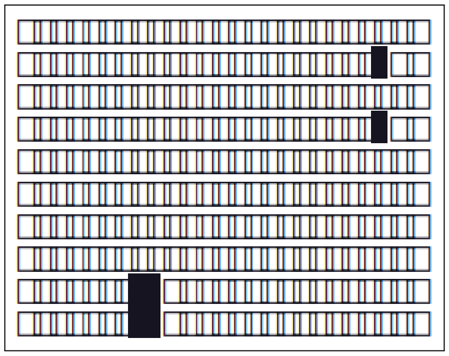
Figure 23: Test 3 consisting of three iterations of a 'blinking' or 'flashing lights' object generated by rule B3/S23 plus an inert mass.


(a) GPT-3.5's guess of the final iteration after seeing the two iterations prior.


(b) GPT-3.5's guess after seeing the first two iterations of test 3 and receiving feedback including the clue about rule B3/S23.



(a) GPT-4's guess of the iteration after seeing the first two iterations prior.



(b) GPT-4's guess after seeing the first two iterations of test 3 and receiving feedback including the clue about rule B3/S23.