A Appendix

A.1 Dataset Examples

In this section of the appendix, we present a detailed overview of several representative tasks from each category included in REASONING GYM. For each task, we describe its structure, complexity parameters, and provide examples.

A.1.1 `complex_arithmetic` (Algebra)

Find the solution of an arithmetic operation involving complex numbers.

Default Configuration

```
min_real = -10
max_real = 10
min_imag = -10
max_imag = 10
operations = ('+', '-', '*', '/')
operations_weights = [0.4, 0.4, 0.1, 0.1]
```

Example Task

```
> Question: Subtract the complex numbers: (7.0 - 7.0i) - (-5.0 + 2.0i)

> Answer: 12.0 - 9.0i

> Metadata: {
'source_dataset': 'complex_arithmetic',
'source_index': 2,
'num1': (7.0, -7.0),
'num2': (-5.0, 2.0),
'operation': '-',
'result': (12, -9),
'difficulty': {
  'min_real': -10,
  'max_real': 10,
  'min_imag': -10,
  'max_imag': 10,
  'operations_weights': [0.4, 0.4, 0.1, 0.1]
  }
}
```

A.1.2 `spiral_matrix` (Algorithmic)

Print the elements of a matrix in spiral order.

Default Configuration

```
min_n = 2
max_n = 10
```

Example Task

```
> Question: Given a matrix, your job is to generate a list of elements
in spiral order, starting from the top-left element.

The spiral order is clockwise, starting from the top-left corner.
More precisely:
- Start from the top-left corner and move right.
- Move down towards the bottom-right corner.
- Move left towards the bottom-left corner.
```

```
- Move up towards the top-right corner.
- Repeat the steps for the inner elements of the matrix until every
entry is visited.

Your output should be a space-separated list of integers, e.g.
1 2 3 4 5 6

For the matrix below, what is the list of elements in spiral order?
3 1 3
2 4 9
1 0 8

> Answer: 3 1 3 9 8 0 1 2 4

> Metadata: {
'source_dataset': 'spiral_matrix',
'source_index': 0,
'matrix': [[3, 1, 3], [2, 4, 9], [1, 0, 8]],
'solution': [3, 1, 3, 9, 8, 0, 1, 2, 4],
'n': 3,
'difficulty': {'n': (2, 10)}
}
```

A.1.3 `arc_1d` (ARC)

Find the solution of a 1D version of an ARC problem.

Default Configuration

```
min_size = 10
max_size = 30
num_train = 3
```

Example Task

```
> Question: Find the common rule that maps an input grid to an output
grid, given the examples below.

Example 1:
Input:  0 0 0 2 9 2 3 4 4 0
Output: 2 9 2 3 4 4 0 0 0 0

Example 2:
Input:  0 0 0 0 4 4 2 1 1 0
Output: 0 4 4 2 1 1 0 0 0 0

Example 3:
Input:  0 0 0 7 9 4 9 1 0 0
Output: 7 9 4 9 1 0 0 0 0 0

Below is a test input grid. Predict the corresponding output grid by
applying the rule you found. Describe how you derived the rule and
your overall reasoning process in detail before you submit your answer.
Your final answer should be just the test output grid itself.

Input:
0 0 0 0 0 1 5 0 0 0

> Answer: 0 0 1 5 0 0 0 0 0 0
```

```
> Metadata: {
'source_dataset': 'arc_1d',
'source_index': 0,
'task_name': 'move_3pix_colorful_left',
'train_examples': [
  {'input': [0, 0, 0, 2, 9, 2, 3, 4, 4, 0],
  'output': [2, 9, 2, 3, 4, 4, 0, 0, 0, 0]},
  {'input': [0, 0, 0, 0, 4, 4, 2, 1, 1, 0],
  'output': [0, 4, 4, 2, 1, 1, 0, 0, 0, 0]},
  {'input': [0, 0, 0, 7, 9, 4, 9, 1, 0, 0],
  'output': [7, 9, 4, 9, 1, 0, 0, 0, 0, 0]}],
'test_example': {
  'input': [0, 0, 0, 0, 0, 1, 5, 0, 0, 0],
  'output': [0, 0, 1, 5, 0, 0, 0, 0, 0, 0]},
'difficulty': {'size': (10, 30)}
}
```

A.1.4 `prime_factorization` (Arithmetic)

Factorize a given number down to its primes.

Default Configuration

```
min_value = 2
max_value = 1000
```

Example Task

```
> Question: Find the prime factorization of 656.
Write the factors separated by ×
(Example: for 12 the answer would be: 2 × 2 × 3)

> Answer: 2 × 2 × 2 × 2 × 41

> Metadata: {
'source_dataset': 'prime_factorization',
'source_index': 0,
'number': 656,
'factors': [2, 2, 2, 2, 41],
'difficulty': {'value': (2, 1000)}
}
```

A.1.5 `bf` (Code)

Find the solution of a BF (Brainf*ck) program.

Default Configuration

```
difficulty = 1
```

Example Task

```
> Question: This is a BF (Brainf*ck) computer program.
What is the output?

">[-]>[-]<>++++++++++[<+++++++++++>-]<+.-.+++++.--------------.+++++++++
++++++.<"

Respond only with the exact output of the program.

> Answer: onset
```

```
> Metadata: {
'source_dataset': 'bf',
'source_index': 0,
'bfit_code': '\nint main() {\n    print("onset");\n}\n',
'bf_program': '>[-]>[-]<>++++++++++[<++++++++++>-]<+.-.
               +++++.--------------.+++++++++++++++.<',
'difficulty': {'difficulty': 1}
}
```

A.1.6 `figlet_font` (Cognition)

Read the contents of text written with figlet font.

Default Configuration

```
static_word = None
static_font = None
min_word_len = 3
max_word_len = 7
space_letters = True
```

Example Task

```
> Question: What word does this say?

##   ##   ######  ##        ######   ######   ######      ##
### ###  #######  ##        ######  #######  #######    #####
#######  ##       ##          ##    ##       ##         ## ##
#######  #######  ##          ##     #####    #####    ##  ##
## # ##  ##       ##          ##         ##       ##   ######
##   ##  #######  #######   ######  #######  #######  ##   ##
##   ##   ######   ######   ######  ######   ######   ##   ##


> Answer: MELISSA

> Metadata: {
'source_dataset': 'figlet_font',
'source_index': 1,
'font': 'stealth_',
'space_letters': True,
'difficulty': {'word_len': (3, 7)}
}
```

A.1.7 `mini_sudoku` (Games)

Solve a mini (4x4) Sudoku puzzle.

Default Configuration

```
min_empty = 8
max_empty = 12
```

Example Task

```
> Question: In 4x4 Mini Sudoku:
- Each row must contain each number from 1-4 exactly once
- Each column must contain each number 1-4 exactly once
- Each 2x2 subgrid must contain each number 1-4 exactly once

```

```
Solve this 4x4 Mini Sudoku puzzle:
4 _ _ _
_ 3 _ _
_ 1 3 _
_ _ _ _

Format your response as the puzzle above, with spaces separating each
number within a row, and newlines separating rows.

> Answer: 4 2 1 3
1 3 4 2
2 1 3 4
3 4 2 1

> Metadata: {
'source_dataset': 'mini_sudoku',
'source_index': 0,
'puzzle': [
  [4, 0, 0, 0],
  [0, 3, 0, 0],
  [0, 1, 3, 0],
  [0, 0, 0, 0]
],
'solution': [
  [4, 2, 1, 3],
  [1, 3, 4, 2],
  [2, 1, 3, 4],
  [3, 4, 2, 1]
],
'num_empty': 12,
'difficulty': {'empty': (8, 12)}
}
```

A.1.8 `advanced_geometry` (Geometry)

Solve geometry-related problems.

Default Configuration

```
min_coord = -10
max_coord = 10
```

Example Task

```
> Question: For triangle with vertices A=(-1, -6), B=(4, 1), and
C=(-7, 4), determine the orthocenter (intersection of altitudes).
For all geometry problems:
1. Give coordinates in the form (x, y)
2. Round decimal answers to 3 decimal places
3. Use the degree symbol ° for angles
4. Return only the angle, coordinates, or radius as your answer.

> Answer: (0.304, -1.217)

> Metadata: {
'A': (-1, -6),
'B': (4, 1),
'C': (-7, 4),
'ortho': (7/23, -28/23),
'orthocenter_exact': ('7/23', '-28/23'),
```

```
'orthocenter_approx': (0.30434782608695654, -1.2173913043478262),
'source_dataset': 'advanced_geometry',
'source_index': 1,
'task_type': 'orthocenter',
'difficulty': {'min_coord': -10, 'max_coord': 10}
}
```

A.1.9 `shortest_path` (Graphs)

Find the shortest path between a start and an end node.

Default Configuration

```
min_rows = 5
max_rows = 8
min_cols = 5
max_cols = 8
p_blocked = 0.4
```

Example Task

```
> Question: Your task is to find the shortest path from the start to
the destination point in a grid.

The grid is represented as a matrix with the following types of cells:
- *: your starting point
- #: your destination point
- O: an open cell
- X: a blocked cell

Therefore, you need to find the shortest path from * to #,
moving only through open cells.

You may only move in four directions: up, down, left, and right.

If there is no path from * to #, simply write "infeasible".

Your output should be a sequence of directions that leads from * to #,
e.g. right right down down up left

Now, find the length of the shortest path from * to # in the
following grid:
O X X X O
O O X X X
O O # O O
* X O O X
O X X O X

> Answer: up right right

> Metadata: {
'source_dataset': 'shortest_path',
'source_index': 0,
'matrix': [
  ['O', 'X', 'X', 'X', 'O'],
  ['O', 'O', 'X', 'X', 'X'],
  ['O', 'O', '#', 'O', 'O'],
  ['*', 'X', 'O', 'O', 'X'],
  ['O', 'X', 'X', 'O', 'X']
```

```
],
'solution': ['up', 'right', 'right'],
'difficulty': {'rows': (5, 8), 'cols': (5, 8)}
}
```

A.1.10 `acre` (Induction)

Determine whether new combinations of objects will activate a detector using prior observations.

Default Configuration

```
train = 1
```

Example Task

```
> Question: You are a researcher studying causal relationships using
Blicket experiments. In these experiments, certain objects (called
'blickets') have the hidden property of activating a detector,
causing its light to turn on.

Each example shows the results of placing different combinations of
objects on the detector. Each object is described by color, material
and shape. Your task is to determine whether a new combination of
objects will cause the detector to activate.

After observing the previous examples, respond with:
- "on" if you can determine the detector light will turn on
- "off" if you can determine the detector light will stay off
- "undetermined" if there is insufficient evidence to reach a conclusion

Do not use quotation marks in your answer.

Previous experimental results:
yellow rubber cylinder → on
red rubber sphere → off
yellow rubber cylinder, red rubber sphere → on
yellow metal sphere, red metal cylinder, brown rubber cylinder,
    purple rubber sphere, yellow rubber cube → on
yellow rubber cube, brown rubber cylinder, purple rubber sphere → off
yellow metal sphere, red metal cylinder → on

New test case:
yellow rubber cylinder

What is the detector light status?

> Answer: on

> Metadata: {
'source_dataset': 'acre',
'source_index': 0
}
```

A.1.11 `knights_knaves` (Logic)

Determine which individuals are truth-telling, and which are liars.

Default Configuration

```
n_people = 2
depth_constraint = 2
width_constraint = 2
```

Example Task

```
> Question: A very special island is inhabited only by sages and fools.
Sages always tell the truth, and fools always lie.
You meet 2 inhabitants: Zoey, and Riley.
Zoey commented, "Riley is a fool".
In Riley's words: "Zoey is a sage or Riley is a sage".
So who is a sage and who is a fool?
(Format your answer like: "Zoey is a sage/fool, and Riley is a sage/fool")

> Answer: Zoey is a fool, and Riley is a sage.

> Metadata: {
'source_dataset': 'knights_knaves',
'source_index': 0,
'statements': (
  ('lying', 1), ('or', ('telling-truth', 0), ('telling-truth', 1))
),
'solution': (False, True),
'names': ['Zoey', 'Riley'],
'knight_knave_terms': {
  'knight': 'sage',
  'knave': 'fool',
  'a_knight': 'a sage',
  'a_knave': 'a fool',
  'Knight': 'Sage',
  'Knave': 'Fool'
},
'difficulty': {
  'n_people': 2,
  'depth_constraint': 2,
  'width_constraint': 2}
}
```

A.2 Zero-Shot Evaluation: Configs

Below are the configuration files used to procedurally generate data for the zero-shot evaluation benchmarks. Each dataset lists parameters with values for the easy setting, while the hard setting values are shown in comments.

`complex_arithmetic`

```yaml
min_real: -10  # -100
max_real: 10  # 100
min_imag: -10  # -100
max_imag: 10  # 100
operations_weights: [0.4, 0.4, 0.1, 0.1]  # [0.25, 0.25, 0.25, 0.25]
```

`intermediate_integration`

```yaml
problem_type_weights: [0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12]
                    # [0, 0, 0, 1, 0, 0, 0, 0]
```

`polynomial_equations`

```yaml
min_degree: 1  # 2
max_degree: 3  # 3
min_terms: 2  # 3
max_terms: 4  # 4
```

polynomial_multiplication

```yaml
min_terms: 2  # 4
max_terms: 4  # 8
min_value: 1  # 10
max_value: 100  # 10000
min_degree: 0  # 1
max_degree: 3  # 4
min_polynomials: 2  # 3
max_polynomials: 3  # 6
```

simple_equations

```yaml
min_terms: 2  # 3
max_terms: 4  # 10
min_value: 1  # 10
max_value: 100  # 10000
operators_weights: [0.4, 0.4, 0.2]  # [0.35, 0.35, 0.3]
```

simple_integration

```yaml
min_terms: 2  # 3
max_terms: 5  # 4
```

ab

```yaml
length: 10  # 25
```

base_conversion

```yaml
min_base: 2  # 9
max_base: 16  # 18
min_value: 0  # 10000
max_value: 1000  # 100000
```

binary_alternation

```yaml
min_n: 10  # 50
max_n: 30  # 500
```

binary_matrix

```yaml
p_zero: 0.25  # 0.25
min_n: 3  # 25
max_n: 10  # 50
```

caesar_cipher

```yaml
min_rotation: 1  # 15
max_rotation: 25  # 25
min_words: 3  # 15
max_words: 20  # 25
```

count_primes

```yaml
min_n: 1  # 10000
max_n: 10000  # 50000
```

cryptarithm

```yaml
min_words: 2  # 5
max_words: 3  # 10
```

game_of_life

```yaml
grid_size_x: 10  # 50
grid_size_y: 10  # 50
filled_cells_weights: 0.1  # 0.2
simulation_steps: 1  # 2
```

game_of_life_halting

```yaml
grid_size_x: 12  # 50
grid_size_y: 12  # 50
difficulty: 1  # 2
num_oscillators: 5  # 7
max_simulation_steps: 20  # 50
```

graph_color

```yaml
min_num_vertices: 10  # 10
max_num_vertices: 10  # 20
num_colors: 3  # 4
```

group_anagrams

```yaml
min_anagram_groups: 2  # 10
max_anagram_groups: 10  # 50
min_words_per_group: 2  # 2
max_words_per_group: 5  # 5
```

isomorphic_strings

```yaml
min_string_length: 2  # 50
max_string_length: 10  # 100
```

jugs

```yaml
num_jugs: 3  # 4
difficulty: 10  # 10
```

letter_counting

```yaml
min_words: 5  # 25
max_words: 15  # 50
```

letter_jumble

```yaml
min_word_len: 1  # 5
max_word_len: 64  # 30
min_words: 3  # 25
max_words: 20  # 50
min_corruption_level: 0.1  # 0.3
max_corruption_level: 0.9  # 0.6
```

manipulate_matrix

```yaml
min_rows: 2  # 25
max_rows: 10  # 50
min_cols: 2  # 25
max_cols: 10  # 50
```

```
min_transforms: 1  # 3
max_transforms: 10  # 10
```

number_filtering

```
min_numbers: 3  # 50
max_numbers: 10  # 100
min_decimals: 0  # 2
max_decimals: 4  # 4
min_value: -100.0  # -500
max_value: 100.0  # 500
```

number_sorting

```
min_numbers: 3  # 50
max_numbers: 10  # 100
min_decimals: 0  # 2
max_decimals: 2  # 4
min_value: -100.0  # -500
max_value: 100.0  # 500
```

palindrome_generation

```
min_length: 3  # 50
max_length: 10  # 100
```

palindrome_partitioning

```
min_string_len: 5  # 5
max_string_len: 15  # 15
min_substring_palindrome_len: 1  # 1
max_substring_palindrome_len: 5  # 5
```

pool_matrix

```
min_rows: 2  # 25
max_rows: 10  # 50
min_cols: 2  # 25
max_cols: 10  # 50
min_pool_size: 1  # 5
max_pool_size: 3  # 7
```

ransom_note

```
min_note_length: 1  # 50
max_note_length: 10  # 100
min_magazine_length: 2  # 100
max_magazine_length: 30  # 500
```

rotate_matrix

```
min_n: 2  # 25
max_n: 10  # 50
min_rotations: 0  # 5
max_rotations: 10  # 15
```

rotten_oranges

```
min_n: 10  # 25
max_n: 30  # 50
```

sentence_reordering

```yaml
min_words_in_sentence: 3  # 20
max_words_in_sentence: 20  # 50
```

spell_backward

```yaml
min_word_len: 3  # 5
max_word_len: 10  # 20
```

spiral_matrix

```yaml
min_n: 2  # 25
max_n: 10  # 50
```

string_insertion

```yaml
min_string_length: 5  # 50
max_string_length: 20  # 100
```

string_manipulation

```yaml
min_string_length: 5  # 50
max_string_length: 20  # 100
```

string_splitting

```yaml
min_initial_machines: 0  # 50
max_initial_machines: 5  # 100
```

string_synthesis

```yaml
min_initial_blocks: 0  # 50
max_initial_blocks: 5  # 100
```

word_ladder

```yaml
min_word_length: 4  # 3
max_word_length: 4  # 5
```

word_sequence_reversal

```yaml
min_words: 3  # 25
max_words: 8  # 50
```

word_sorting

```yaml
min_words: 3  # 25
max_words: 10  # 50
min_word_length: 3  # 5
max_word_length: 12  # 10
```

arc_1d

```yaml
min_size: 10  # 25
max_size: 30  # 50
```

arc_agi

```yaml
rotations_weights: [0.25, 0.25, 0.25, 0.25]  # [0.15, 0.3, 0.25, 0.3]
mirrors_weights: [0.2, 0.2, 0.2, 0.2, 0.2]  # [0.2, 0.2, 0.2, 0.2, 0.2]
```

rearc

```yaml
pso_difficulty_weights: [0.14, 0.14, 0.14, 0.14, 0.14, 0.14, 0.14]
                        # [0, 0, 0, 1, 0, 0, 0]
```

```
rng_difficulty_weights: [0.14, 0.14, 0.14, 0.14, 0.14, 0.14, 0.14]
                        # [0, 0, 0, 1, 0, 0, 0]
```

basic_arithmetic

```
min_terms: 2  # 5
max_terms: 6  # 10
min_digits: 1  # 2
max_digits: 4  # 5
```

bitwise_arithmetic

```
difficulty: 2  # 5
```

calendar_arithmetic

```
tasks: [
    'weekday_offset',
    'weekday_of_date',
    'weekday_of_date_from_first_date',
    'recurring_event_day',
    'count_days',
    'count_business_days',
    'is_leap_year'
]
# [
#    'weekday_of_date',
#    'is_leap_year',
#    'weekday_offset',
#    'count_days',
#    'count_business_days'
# ]
offset_upper_bound: 100  # 200
```

chain_sum

```
min_terms: 2  # 5
max_terms: 6  # 8
min_digits: 1  # 4
max_digits: 4  # 6
```

count_bits

```
min_n: 1  # 1000000
max_n: 2147483647  # 100000000
```

decimal_arithmetic

```
min_num_decimal_places: 3  # 5
max_num_decimal_places: 3  # 8
precision: 12  # 10
min_terms: 2  # 5
max_terms: 6  # 8
```

decimal_chain_sum

```
min_terms: 2  # 5
max_terms: 6  # 8
min_digits: 1  # 4
max_digits: 4  # 8
min_decimal_places: 1  # 4
max_decimal_places: 4  # 6
```

dice

```
num_dice: 4  # 6
max_dice_size: 20  # 25
```

fraction_simplification

```
min_value: 1  # 100
max_value: 1000  # 1000
min_factor: 1  # 10
max_factor: 100  # 100
```

gcd

```
min_numbers: 2  # 3
max_numbers: 2  # 4
min_value: 1  # 1000
max_value: 1000  # 10000
```

gsm_symbolic

```
# no parameters to override
```

lcm

```
min_numbers: 2  # 3
max_numbers: 2  # 4
min_value: 1  # 1000
max_value: 100  # 10000
```

leg_counting

```
min_animals: 3  # 20
max_animals: 10  # 30
min_instances: 1  # 64
max_instances: 15  # 256
```

number_format

```
min_num_candidates: 2  # 25
max_num_candidates: 5  # 100
min_n: 1000  # 100000
max_n: 1000000000  # 1000000
max_delta: 10.0  # 0.001
```

power_function

```
min_exponent: 0  # 4
max_exponent: 8  # 8
```

prime_factorization

```
min_value: 2  # 1000
max_value: 1000  # 5000
```

products

```
min_terms: 2  # 4
max_terms: 2  # 8
min_digits: 1  # 4
max_digits: 5  # 8
```

time_intervals

```yaml
max_time_difference_seconds: 86400  # 21600
max_date_difference_days: 100  # 30
```

bf

```yaml
difficulty: 1  # 2
```

codeio

```yaml
difficulty: -1  # 7
```

color_cube_rotation

```yaml
min_rotations: 1  # 10
max_rotations: 3  # 50
```

figlet_font

```yaml
min_word_len: 3  # 5
max_word_len: 7  # 10
```

modulo_grid

```yaml
size_x: 20  # 40
size_y: 20  # 40
max_holes: 1  # 5
max_divisor: 20  # 7
max_target: 20  # 3
```

needle_haystack

```yaml
min_num_statements: 10  # 100
max_num_statements: 100  # 500
```

number_sequence

```yaml
min_terms: 4  # 5
max_terms: 8  # 10
min_value: -100  # -500
max_value: 100  # 500
max_complexity: 3  # 3
```

rectangle_count

```yaml
max_rectangles: 10  # 15
```

rubiks_cube

```yaml
cube_size: 3  # 5
min_scramble_steps: 3  # 25
max_scramble_steps: 10  # 50
```

countdown

```yaml
min_numbers: 4  # 3
max_numbers: 6  # 9
min_target: 100  # 100
max_target: 999  # 1000
min_value: 1  # 1
max_value: 100  # 100
```

emoji_mystery

```
min_words_in_sentence: 3  # 10
max_words_in_sentence: 35  # 30
```

futoshiki

```
min_board_size: 4  # 6
max_board_size: 9  # 7
min_difficulty: 0  # 1
max_difficulty: 3  # 2
```

knight_swap

```
min_nodes: 6  # 6
max_nodes: 9  # 8
min_pieces: 2  # 3
max_pieces: 2  # 4
min_steps: 4  # 1
max_steps: 20  # 20
```

mahjong_puzzle

```
min_num_rounds: 10  # 50
max_num_rounds: 50  # 100
```

maze

```
min_grid_size: 5  # 25
max_grid_size: 10  # 50
min_dist: 5  # 10
max_dist: 10  # 15
```

mini_sudoku

```
min_empty: 8  # 6
max_empty: 12  # 10
```

n_queens

```
n: 8  # 8
min_remove: 1  # 4
max_remove: 7  # 6
```

puzzle24

```
min_value: 1  # 1
max_value: 10  # 6
```

rush_hour

```
min_moves: 1  # 25
max_moves: 50  # 50
```

sokoban

```
min_w: 6  # 10
max_w: 10  # 15
min_h: 6  # 10
max_h: 10  # 15
```

sudoku

```
min_empty: 30  # 30
max_empty: 50  # 50
```

tower_of_hanoi

```yaml
min_disks: 3  # 5
max_disks: 7  # 10
min_pegs: 3  # 3
max_pegs: 4  # 4
```

tsumego

```yaml
min_board_size: 9  # 5
max_board_size: 13  # 15
max_stones: 15  # 10
```

advanced_geometry

```yaml
min_coord: -10  # -100
max_coord: 10  # 100
```

simple_geometry

```yaml
min_sides: 3  # 10
max_sides: 6  # 15
```

course_schedule

```yaml
min_num_courses: 5  # 25
max_num_courses: 10  # 50
min_num_prerequisites: 1  # 3
max_num_prerequisites: 2  # 4
min_cycle_length: 3  # 3
max_cycle_length: 5  # 4
```

family_relationships

```yaml
min_family_size: 4  # 5
max_family_size: 8  # 9
```

largest_island

```yaml
min_rows: 5  # 25
max_rows: 10  # 50
min_cols: 5  # 25
max_cols: 10  # 50
min_num_islands: 0  # 5
max_num_islands: 5  # 10
min_island_size: 0  # 5
max_island_size: 10  # 20
```

quantum_lock

```yaml
difficulty: 10  # 5
```

shortest_path

```yaml
min_rows: 5  # 25
max_rows: 8  # 50
min_cols: 5  # 25
max_cols: 8  # 50
```

acre

```yaml
# no parameters to override
```

list_functions

```yaml
# no parameters to override
```

aiw

```yaml
task_type_weights: [0.33, 0.33, 0.33]  # [0.5, 0.25, 0.25]
max_entities: 6  # 10
```

circuit_logic

```yaml
min_terms: 3  # 10
max_terms: 5  # 20
min_inputs: 2  # 4
max_inputs: 4  # 8
```

knights_knaves

```yaml
n_people: 2  # 3
depth_constraint: 2  # 3
width_constraint: 2  # 3
```

propositional_logic

```yaml
min_vars: 2  # 4
max_vars: 4  # 8
min_statements: 2  # 4
max_statements: 4  # 8
min_complexity: 1  # 2
max_complexity: 3  # 4
```

self_reference

```yaml
difficulty: 5  # 5
```

syllogism

```yaml
allow_all: True  # True
allow_no: True  # True
allow_some: True  # False
allow_some_not: True  # False
```

zebra_puzzles

```yaml
num_people: 4  # 5
num_characteristics: 4  # 5
```

A.3 Zero-Shot Evaluation: Per-dataset performance on Hard settings

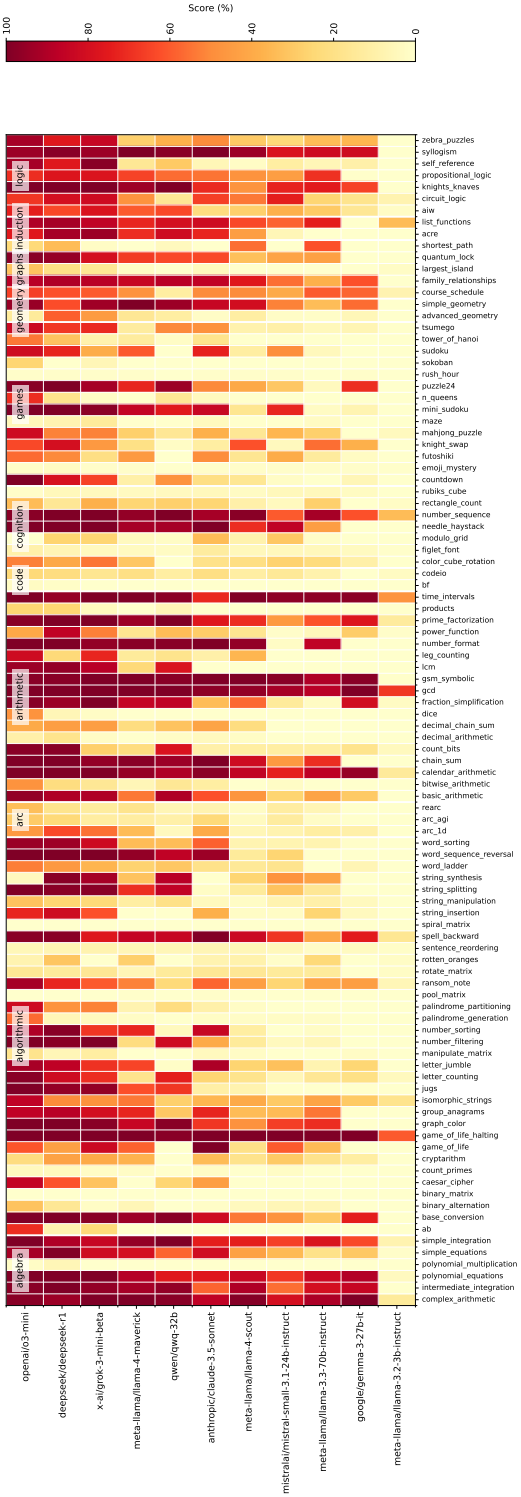

Figure 7: Per-task reasoning accuracy. Performance quickly drops beyond basic skills, and even the top model (o3-mini) falters on long-horizon puzzles such as `rush_hour`, `rubiks_cube` and `rotten_oranges`, underscoring the benchmark's value for probing advanced reasoning.

A.4 Zero-Shot Evaluation: Hard vs. Easy

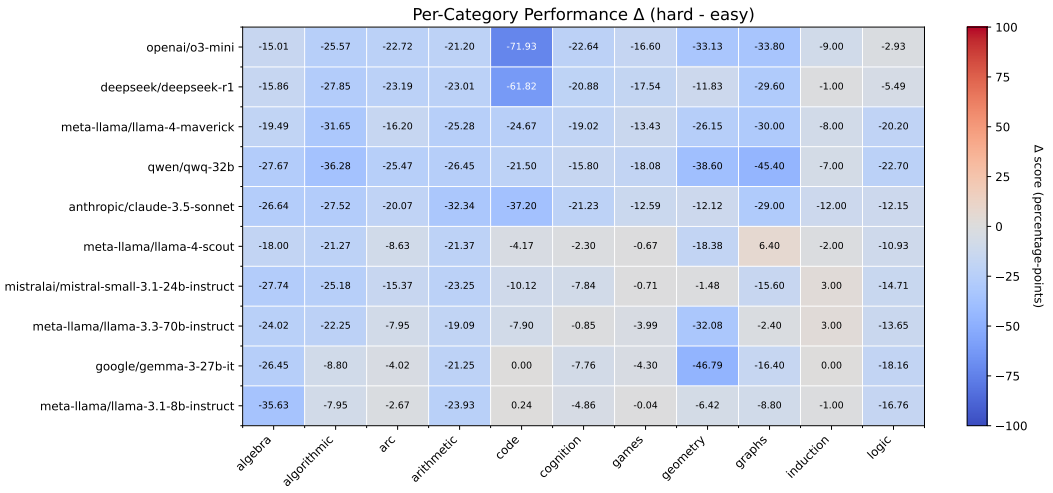

Figure 8: Impact of dataset difficulty on per-category accuracy. Raising generator difficulty consistently depresses performance across most categories, with larger drops visible in compositional-reasoning tasks. The decline is pronounced for both high-capacity models such as o3-mini and smaller instruction-tuned models like gemma-3-27b-it, highlighting that increased problem complexity strains models irrespective of scale.

B Related Work

C Training Approach

Here we briefly describe the approach taken to training models with RLVR. Full training code and configurations are available in our open-source repository at https://github.com/open-thought/reasoning-gym.

We use the verl open-source library for most training runs including all intra-domain and inter-domain generalization experiments, as well as curriculum learning experiments. Training runs are conducted using a 4xA6000 GPU node on Runpod.

For the Qwen2.5-3B-Instruct-RG-Math model trained as part of external generalization experiments, we use separate training code which is also included and documented in the training section of our repository, under the qwen-math subdirectory.

Below we show an example of a verl training config with custom REASONING GYM modifications. We omit two sections, reward_model and critic, which are required by verl but have no effect on the training when using GRPO due to the lack of reward and critic models.

Example Training Config for verl

```yaml
reasoning_gym:
  dataset_size: 20000
  developer_prompt: DeepSeekZero
  datasets:
    ab:
      weight: 1
    base_conversion:
      weight: 1
    binary_alternation:
      weight: 1
      config:
        p_solvable: 0.9
    binary_matrix:
      weight: 1
```

```yaml
      config:
        min_n: 2
        max_n: 6
    caesar_cipher:
      weight: 1
      config:
        max_words: 10
    cryptarithm:
      weight: 1
    isomorphic_strings:
      weight: 1
      config:
        max_string_length: 8
    jugs:
      weight: 1
      config:
        difficulty: 6
    rotate_matrix:
      weight: 1
      config:
        min_n: 2
        max_n: 6
    string_manipulation:
      weight: 1
      config:
        max_string_length: 15
        max_num_rules: 6

curriculum:
    enabled: False
    schedule:
      automatic: True
      update_steps: 30 # automatic curriculum updating after 50 steps
    last_k: 20
    success_threshold: 0.70
    failure_threshold: 0.10
    curricula:
      spell_backward:
        attribute_levels:
          word_len: 0
reward:
  use_accuracy: True
  secondary_rewards:
   - name: format
     scaling_factor: 0.2
     kwargs:
        preappend_thinking_token: False
   - name: length
     scaling_factor: 0.2

data:
  tokenizer: null
  train_files: train.parquet # unused due to RG procedural dataset generators
  val_files: test.parquet # unused due to RG procedural dataset generators
  prompt_key: prompt
  max_prompt_length: 4096
  max_response_length: 2048
  train_batch_size: 32
  val_batch_size: 64
  return_raw_chat: True
  return_raw_input_ids: True
actor_rollout_ref:
  hybrid_engine: True
  model:
    path: Qwen/Qwen2.5-3B-Instruct
```

```yaml
  external_lib: null
  override_config: { }
  enable_gradient_checkpointing: True
  use_remove_padding: True
actor:
  strategy: fsdp  # This is for backward-compatibility
  ppo_mini_batch_size: 16
  ppo_micro_batch_size: null # will be deprecated, use
  ↪  ppo_micro_batch_size_per_gpu
  ppo_micro_batch_size_per_gpu: 8
  use_dynamic_bsz: False
  ppo_max_token_len_per_gpu: 49152 # n * ${data.max_prompt_length} +
  ↪  ${data.max_response_length}
  grad_clip: 1.0
  clip_ratio: 0.2
  entropy_coeff: 0.001
  use_kl_loss: True # True for GRPO
  kl_loss_coef: 0.001 # for grpo
  kl_loss_type: low_var_kl # for grpo
  ppo_epochs: 1
  shuffle: False
  ulysses_sequence_parallel_size: 1 # sp size
  optim:
    lr: 1e-6
    lr_warmup_steps_ratio: 0.  # the total steps will be injected during runtime
    min_lr_ratio: null   # only useful for warmup with cosine
    warmup_style: constant  # select from constant/cosine
    total_training_steps: 500  # must be override by program
  fsdp_config:
    wrap_policy:
      # transformer_layer_cls_to_wrap: None
      min_num_params: 0
    param_offload: False
    optimizer_offload: False
    fsdp_size: -1
ref:
  fsdp_config:
    param_offload: True
    wrap_policy:
      # transformer_layer_cls_to_wrap: None
      min_num_params: 0
  log_prob_micro_batch_size: null # will be deprecated, use
  ↪  log_prob_micro_batch_size_per_gpu
  log_prob_micro_batch_size_per_gpu: 160
  log_prob_use_dynamic_bsz: ${actor_rollout_ref.actor.use_dynamic_bsz}
  log_prob_max_token_len_per_gpu:
  ↪  ${actor_rollout_ref.actor.ppo_max_token_len_per_gpu}
  ulysses_sequence_parallel_size:
  ↪  ${actor_rollout_ref.actor.ulysses_sequence_parallel_size} # sp size
rollout:
  name: vllm
  temperature: 1.0
  top_k: -1 # 0 for hf rollout, -1 for vllm rollout
  top_p: 1
  prompt_length: ${data.max_prompt_length}  # not use for opensource
  response_length: ${data.max_response_length}
  # for vllm rollout
  dtype: bfloat16 # should align with FSDP
  gpu_memory_utilization: 0.7
  ignore_eos: False
  enforce_eager: True
  free_cache_engine: True
  load_format: dummy_dtensor
  tensor_model_parallel_size: 4
  max_num_batched_tokens: 12288
```

```yaml
    max_num_seqs: 1024
    log_prob_micro_batch_size: null # will be deprecated, use
    ↪  log_prob_micro_batch_size_per_gpu
    log_prob_micro_batch_size_per_gpu: 160
    log_prob_use_dynamic_bsz: ${actor_rollout_ref.actor.use_dynamic_bsz}
    log_prob_max_token_len_per_gpu:
    ↪  ${actor_rollout_ref.actor.ppo_max_token_len_per_gpu}
    disable_log_stats: True
    enable_chunked_prefill: True # could get higher throughput
    # for hf rollout
    do_sample: True
    use_fire_sampling: False
    max_model_len: 12288
    # number of responses (i.e. num sample times)
    n: 8 # > 1 for grpo
    val_kwargs:
      do_sample: True

algorithm:
  gamma: 1.0
  lam: 1.0
  adv_estimator: grpo
  kl_penalty: kl  # how to estimate kl divergence
  kl_ctrl:
    type: fixed
    kl_coef: 0.001

verbose: True

trainer:
  balance_batch: True
  total_epochs: 1
  total_training_steps: 500
  project_name: inter-domain-generalisation
  experiment_name: inter_reasoning_algorithmic_qwen_3b_composite
  logger: [ 'console', 'wandb' ]
  val_generations_to_log_to_wandb: 0
  nnodes: 1
  n_gpus_per_node: 4
  save_freq: 100
  # auto: find the last ckpt to resume. If can't find, start from scratch
  resume_mode: auto # or auto or resume_path if
  resume_from_path: False
  test_freq: 100
  critic_warmup: 0
  default_hdfs_dir: null
  remove_previous_ckpt_in_save: False
  del_local_ckpt_after_load: False
  default_local_dir: checkpoints/${trainer.project_name}/${trainer.experiment_name}
```