# Appendix for 'ReaSCAN: Compositional Reasoning in Language Grounding'

## A Dataset Generation

### A.1 Action Sequence

Following gSCAN [14], ReaSCAN agents produce strings composed of action symbols {`walk`, `push`, `pull`, `stay`, `L_turn`, `R_turn`}. The actions `push` and `pull` correspond to the "push" and "pull" verbs in the command. For the verb "push", the agent must push the referent object, where pushing requires moving something as far as possible before hitting a wall or another object. For the verb "pull", the agent must pull the referent object, in which case it would pull the object back as far as possible before hitting a wall or another object. Additionally, any object of size 1 or 2 is labeled as *light*, and any object of size 3 or 4 is labeled as *heavy*. If the referent target is a heavy object, the agent needs to push twice or pull twice to move to the next cell (e.g., ⟨`push,push`⟩ or ⟨`pull,pull`⟩ ). The optional adverbs at the end of the command may alter action sequences by inserting actions following the adverb. For example, a list of actions ⟨`L_turn,L_turn,L_turn,L_turn`⟩ is inserted into the action sequence to fulfill the adverbial "while spinning". Since composition generalization on adverbs is not the focus of this paper, we randomly sample adverbs for each command. See the original gSCAN paper for details on adverbs creation [14].

### A.2 Grounded Determiners

ReaSCAN further extends the naturalness of the linguistic input by grounding its determiners. If an NP in the command (e.g., "red circle") is preceded by the definite determiner "the", there is only one red circle in the world. Otherwise, the object is preceded by the indefinite determiner "a".

### A.3 Dataset Statistics

Given the rich structure of our commands (see Section 4.1), the number of possible commands grows quickly with longer patterns. To ensure we can still generate enough shape worlds per command, we have to down-sample our commands significantly for longer commands. For the `Simple` command, we exhaustively collect all commands, totalling 675 commands. For commands with one or more relative clauses, we then sample 2,025 commands for `1-relative-clause` and 3,375 for `2-relative-clauses`. For each command, we sample 180 shape worlds, which is similar to gSCAN. Our framework is also able to generate the full version of ReaSCAN (i.e., considering all combinations of commands), which, though, uses about 250G of disk space.

### A.4 Infrastructure Setups

To generate commands for all three patterns, it takes approximately 16 hours using a single process on a standard CPU cluster. With 50 processes, it takes less than 20 minutes for the largest subset in this paper.

## B Dataset Artifacts

In contrast to realistic datasets, synthetic datasets provide controllable environments for testing a specific aspect of neural models. However, synthetic datasets may produce artifacts induced from the programs generating them. Here, we disclose, as comprehensively as we can, potential artifacts resulting from our data generation process.

### B.1 Non-comprehensive Linguistic Structures

As discussed in Section 4.1, commands from ReaSCAN follow a specific linguistic template and are non-comprehensive in covering all linguistic structures. For examples, there is no confusion about where relative clauses attach. Additionally, the location of occurrence of verbs and adverbs is fixed in our commands. In parallel, we also down-sample our commands for the `1-relative-clause` and `2-relative-clauses` conditions, to make ReaSCAN models trainable with reasonable computing resources.

### B.2 Non-comprehensive Distractors

To generate a complete list of distractors for commands like "walk to the red circle that is in the same row as the blue square", we need to change each attribute (e.g., "red", "circle", "blue" and "square") while holding others constant. Consequently, our `2-relative-clauses` commands could potentially require more than 600 distractors to fulfill a complete sampling of distractors. However, this leads to a dataset that is incomparable

to gSCAN, which samples at most 12 distractors. Thus, we randomly select a set of phrases and make them necessary for each command (see Section 4.2 for details about distractor sampling strategies).

We quantify the percentages of different types of distractors appearing in ReaSCAN (see definitions in Section 4.2 and Figure 5). Those quantifications are based on the distractors we explicitly generate to fulfill these purposes, therefore serving as a lower-bound estimation for all effective distractors within a world. By chance, some might in fact fulfill multiple purposes, or a random distractor might by chance be highly competitive with the target, resulting in a higher number of effective distractors. As shown in Figure 5, relation-based distractors are present in almost all examples with `1-relative-clause` and `2-relative-clauses` commands. For `Simple` and `1-relative-clause`, we sample attribute-based distractors for almost all examples. For `2-relative-clauses`, we only sample attribute-based distractors when applicable. For example, we skip sampling attribute-based distractors when there are more than 2 boxes present in the shape world. As a result, attribute-based distractors are present in about 60% of the examples for `2-relative-clauses` commands. Random distractors are present in close to 100% of worlds for simpler commands (e.g., `Simple` commands), and close to 0% of worlds for commands with more complex structures (e.g., `2-relative-clauses` commands). Additionally, we only sample isomorphism-based distractors when applicable. For example, we only swap attributes between objects that are not the referent target.

### B.3 Shapes and Relations Biases

As discussed in Section 4.1 and Appendix C, we sample commands following a set of rules, which may lead to imbalanced sampling for shapes and relations. For example, "box" may only appear after the relational clause "is inside of". As a result, the frequencies of "box" and the relational clause "is inside of" are drastically different from the others. Additionally, we disallow unnatural commands such as "walk to the red circle that is in the same color as the square" or "walk to the circle that is in the same color as the red square", where the relation is redundant or the unnatural command can be simplified. Following these rules, the frequencies of different relations and attributes could be further stratified.

Figure 5 also includes frequency plots for different sizes, colors, shapes, and relations in ReaSCAN. We include frequency distributions for the commands and the shape worlds separately. For colors, frequencies are evenly distributed. For the sizes in commands (e.g., "big" or "small"), frequencies are evenly distributed. For the actual sizes in specific shape worlds, smaller sizes have lower frequencies. This is an artifact due to the shape noun "box". Examples including larger boxes may tend to be valid examples compared to smaller ones, which impose spatial limitations. For shapes except the box, frequencies are evenly distributed. "Box" is less frequent, as it can only follow the specific relation "is inside of". For selected relations, frequencies are extremely biased. As shown in Figure 5, relations such as "same shape as", "same color as", and "same size as" are much less frequent than the others. This is due to the fact that we exclude a significant number of unnatural commands that contain these relations (see Appendix C for details).

### B.4 Self-exclusiveness

We assume every object mentioned maps to a unique object in the generated world. For example, if the command is "walk to the object that is in the same color as the square", the target is not allowed to be the square itself. This is generally not true in the real world.

### B.5 Other Induced Artifacts

Figure 5 includes the distributions for verbs and adverbs present in our commands. As shown in the plot, both verbs and adverbs distribute uniformly. We also include distributions for agent-facing directions and agent–target relative directions at the start. As in gSCAN [14], our agent always starts facing east. Additionally, the relative directions between the agent and the target at the start are distributed uniformly.

## C Rule-based Command Samplings

As described in Section 4.1, our command generation process involves building relational clauses between objects. To prevent generated commands from being ungrammatical, we enforce some explicit rules when sampling commands:

- `Rule 1A`: When a "same shape as" relational clause is present in the command (e.g., "`OBJ1` that is in the same shape as `OBJ2`"), both objects for this clause cannot contain any shape descriptor. For example, we consider "a red object that is in the same shape as a red square" as unnatural since one could just say "a red square". If `OBJ1` contains a shape descriptor already, then the relational clause is unnecessary.

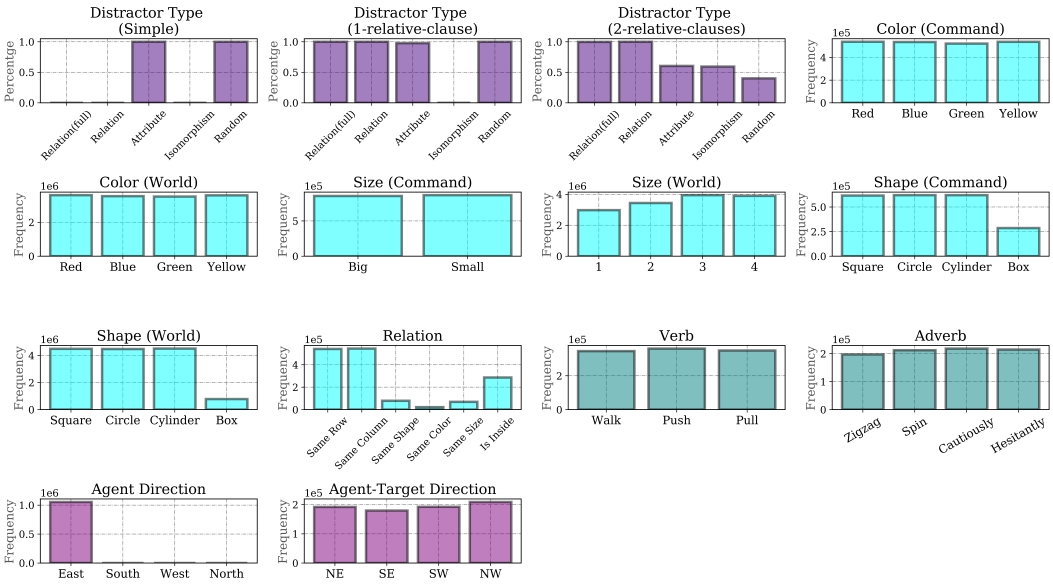


**Figure 5:** Statistics of ReaSCAN.

- `Rule 1B`: When the "same color as" relational clause is present in the command, both objects for this clause cannot contain any color descriptor.
- `Rule 1C`: When the "same size as" relational clause is present in the command, both objects for this clause cannot contain any size descriptor. For size, we only allow two sizes for any objects when size descriptors are mentioned (e.g., a circle can only be in either of two random sizes if "small circle" appears in any command).
- `Rule 2`: The following shape of the "is inside of" relational clause must be a box.
- `Rule 3`: For any object term that has relational clauses following it, it cannot be over-specified with descriptors. For example, if we have "`OBJ1` that is in the same shape as `OBJ2`", we only allow relational conjunction clauses other than "same shape as".
- `Rule 4`: In contrast to gSCAN [14], we enforce an order of modifiers where size descriptors proceed color descriptors (e.g., we allow "small red circle" but not "red small circle").

These rules may over-sample or down-sample certain relations and shapes. We discuss these potential artifacts results in Section B.

# D Sub-graph Matching

## D.1 Multi-edge Graph Representation

Our distractor sampling strategies incentivize models to learn compositional reasoning. At the same time, these strategies increase the chance that the referent target becomes unidentifiable. For example, even if we randomly place objects in a world, they may form relations consistent with the command by chance. We address this issue by solving constraints using a graph representation.

We represent each world as a graph where nodes represent objects and edges represent relations between objects (see Figure 3 for an example). Each node has attribute-based relations with other nodes. For example, a "red circle" node will have a `SAME_COLOR` edge to a "red square" node. To make sure the referent target is unique for every command–world pair, we ensure only one referent target can be identified by querying the graph with our command. We simplify this problem as a problem of sub-graph matching. We represent each command as a sub-graph where nodes represent objects mentioned in the command and edges encode relations between nodes described in the command. Then, we use a sub-graph matching algorithm [49] to ensure that the sub-graph representing the command appears only once in the world graph. Sub-graph matching is a NP-hard problem, so we locally optimize our algorithm to have $O(n^k)$ time complexity where $k$ represents the number of clauses in the command.

### D.2 Locally Optimized Sub-graph Matching Algorithm

To ensure that there is only one referent target, we make sure that the sub-graph representing the command only appears once in the graph of the shape world, as illustrated in Figure 6. We include both the complete matching algorithm, which uses `VF2` as the main algorithm (see Alg. 1), and our locally optimized algorithm (see Alg. 2) for the sub-graph matching problem. Note that this optimized algorithm only applies to three commands mentioned in Section 4.1, and may need minor modifications to adapt to other commands. We use the `VF2` algorithm from the `NetworkX` package for sub-graph matching [6]. Full implementations of our algorithms can be found in our code repository.

**Algorithm 1 Complete Multi-edge Sub-graph Matching**

**Require** multi-edge directional graphs $\texttt{G}_w$ for the world and $\texttt{G}_c$ for the command
**Require** referred object `O` for the command
**Return** matching referent targets `R`

```
import networkx as nx
R ← {}
LiG_w ← nx.LineGraph(G_w)
LiG_c ← nx.LineGraph(G_c)
DiGM ← nx.VF2(LiG_w, LiG_c)
for g_s ← DiGM.subgraph_isomorphisms_iter() do
    isValid ← True
    for pair_w, pair_c ← g_s.items() do
        rel_w ← get_relations(pair_w)
        rel_c ← get_relations(pair_c)
        if not rel_w ∩ rel_c do
            isValid ← False
            break
    end for
    if isValid do
        node ← get_correspond_node(O)
        R ← R + {node}
end for
return R
```

## E Models and Experiments Setups

For our `M-LSTM`[7] and `GCN-LSTM`[8] models, we adapt code from the original repositories. For both models, we optimize for cross-entropy loss using Adam with default parameters [50]. The learning rate starts at $1e^{-4}$ and decays by $0.9$ every 20,000 steps for the `M-LSTM` model. The learning rate starts at $8e^{-4}$ for the `GCN-LSTM` model with the same learning rate decaying schedule. We train for 200,000 steps, with batch size 2000 for the `M-LSTM` model, and train for 100 epochs with batch size 200 for the `GCN-LSTM` model. We choose the best model during training by the performance on a smaller development set of 2,000 examples, which is consistent with the training pipeline proposed in Ruis et al. [14] for gSCAN. For `M-LSTM`, we choose the kernel size for the CNN to be 7. For `GCN-LSTM`, we choose the number of message passing iterations to be 4. We adapt the code released by each paper, which only supports single-GPU training. The training time is about 3 days on a Standard GeForce RTX 2080 Ti GPU with 11GB memory. To foster reproducibility, we release our adapted evaluation scripts in our code repository.

In generating random splits (Section 6.1), we randomly partition train/dev/test after command–world pairs are generated. As shown in Table 2, we generate more than 1M example–world pairs in total. Our `Simple` set is comparable to `gSCAN` [14]. However, our `Simple` set is smaller in size since `gSCAN` permutes based on the agent's relative direction against the referent target. Note that for `1-relative-clause` and `2-relative-clauses`, we down-sample our commands since we keep world per command approximately the same as `gSCAN` to ensure a fair comparison. See Appendix A for detailed dataset statistics. To evaluate our distractor sampling strategies, we regenerate a new dataset for `2-relative-clauses` containing only random distractors and analyze model

[6] https://networkx.org/documentation/stable/reference/algorithms/isomorphism.vf2.html
[7] https://github.com/LauraRuis/multimodal_seq2seq_gSCAN
[8] https://github.com/HQ01/gSCAN_with_language_conditioned_embedding

**Algorithm 2 Locally Optimized Multi-edge Sub-graph Matching**

**Require** multi-edge directional graphs $\texttt{G}_w$ for the world and $\texttt{G}_c$ for the command
**Require** referred object `O` for the command
**Return** matching referent targets `R`

```
R ← {}
for n_w ← G_w.get_nodes() do
    M ← {}
    rel_w ← n_w.get_edges()
    rel_o ← G_c.get_edges(O)
    if |rel_o ∩ rel_w|==|rel_o| do
        # found a potential candidate, checking nbrs
        for nbr ← n_w.get_nbrs() do
            for n_c ← G_c.get_nodes() where n_c != O do
                rel_nbr ← nbr.get_edges()
                rel_c ← n_c.get_edges()
                if |rel_nbr ∩ rel_c|==|rel_c| do
                    # add nbr in to potential matching list
                    M[n_c] ← M[n_c] + nbr
        isValid ← True
        for n_c ← G_c.get_nodes() where n_c!=O do
            if |M[n_c]|==0 do
                isValid ← False
                break
        if isValid && at_least_one_unique_for_each(M) do
            R ← R + {n_w}
end for
return R
```

performance results. Finally, we combine all three subsets for all patterns and evaluate aggregated performance (see Table 3 for per pattern performance).

## F ReaSCAN Generation Workflow

Figure 6 illustrates our main data generation workflow with an example with a single relational clause. The data generation workflow contains five main steps:

- `Step 1`: We first sample a command pattern from the `command` generator. The command pattern contains the basic structure of the command as in Section 4.1
- `Step 2`: We fill out the command pattern by supplying its semantics and generate a fully-formed command.
- `Step 3`: Using our simulator, we generate a shape world containing objects conforming to our command as well as distractors. We have four types of distractors, as described in Section 4.2
- `Step 4`: We then build a graph describing objects and their relations in the shape world generated from the previous step. Using our sub-graph matching algorithm (Appendix D), we validate whether there is only one referent target presented in the shape world grounding the command.
- `Step 5`: If "yes", we record the command–world pair. If "no", we fall back to `Step 3` and generate a new shape world.

## G ReaSCAN Examples

Figure 7 shows more examples for different patterns of commands in ReaSCAN.

## H ReaSCAN as an Abstract Reasoning Challenge

Figure 8 shows two simplified examples of how we can transform ReaSCAN into an abstract reasoning challenge following the Abstract Reasoning Corpus proposed by Chollet [51]. Instead of generating the action sequence

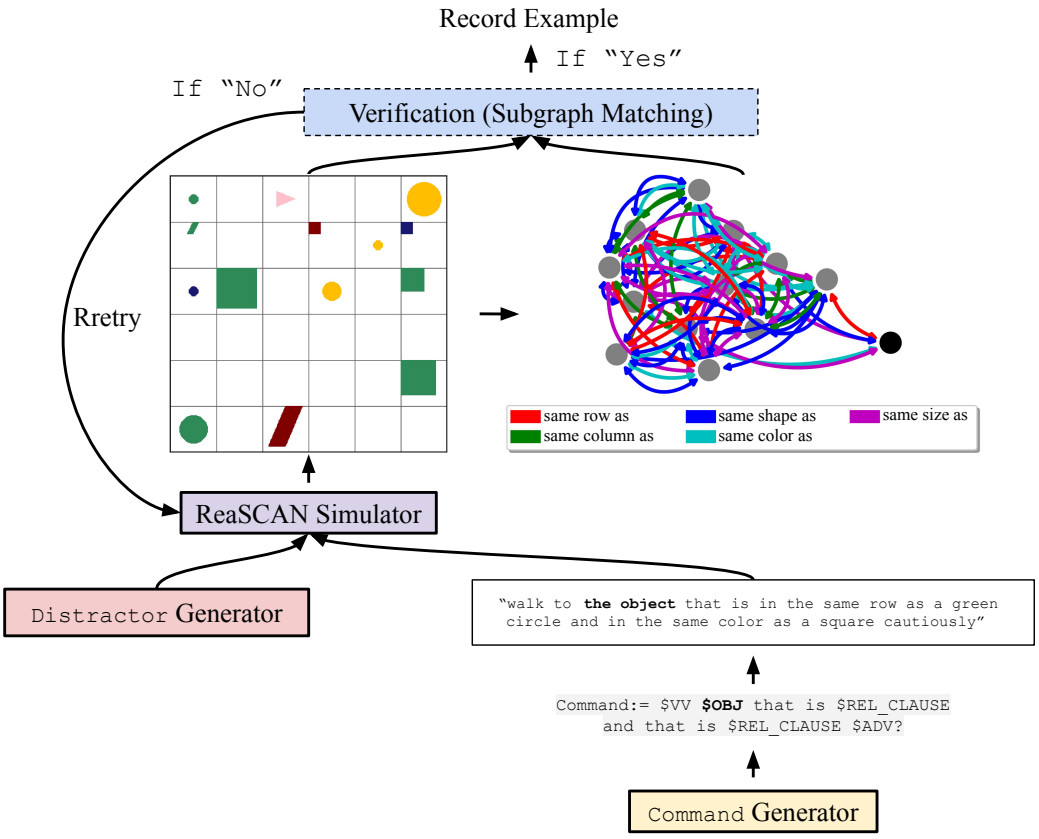


**Figure 6:** Data generation workflow with a simplified example.

based on a command–world pair, the task is now defined as predicting the output of a test input world, given multiple pairs of input–output worlds as training examples. This can be seen as a program induction or program synthesis task as well. Our pipeline is able to generate the reasoning logic involved in each task with natural language. This task mimics abstract reasoning tests for humans. To generate reasoning tasks, we first extend our current framework to generate multiple shape worlds for each command along with the position of the referent targets for each world. Then, we define some primitive actions (e.g., `DRAW` for changing color; `CHANGE` for changing shape) that can be operated on the referent targets. Finally, we generate a new set of shape worlds with operated referent targets.

# I Dataset Documentation

We have made our dataset and the framework to generate the dataset publicly avaliable at `https://github.com/frankaging/Reason-SCAN`. We bear all responsibility in case of violation of rights. The dataset is released with Creative Commons Attribution 4.0 International License. Updates will be reflected in our code repository. The first release is versioned as ReaSCANv1.0. Any subsequent releases will have higher version numbers.

The dataset and its metadata can be found in the code repository. Additionally, we provide detailed steps for how to regenerate ReaSCAN in our code repository. Since the data generation framework is novel, it is self-contained as well. The dataset and its code repository will remain publicly available. We include our datasheets.

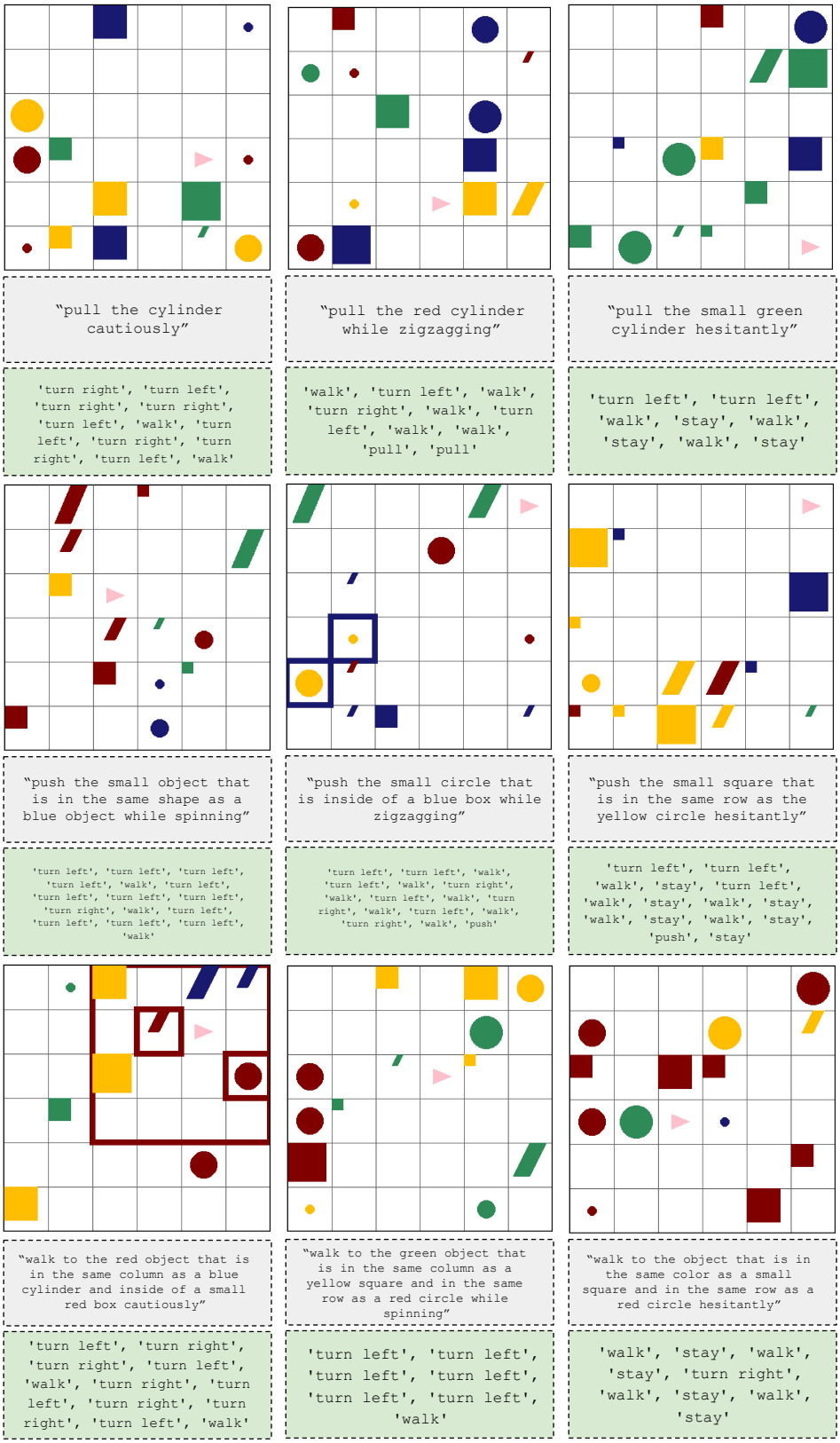


**Figure 7:** ReaSCAN examples with varying command patterns. The navigation commands and the target action sequences are in the grey boxes and green boxes respectively.

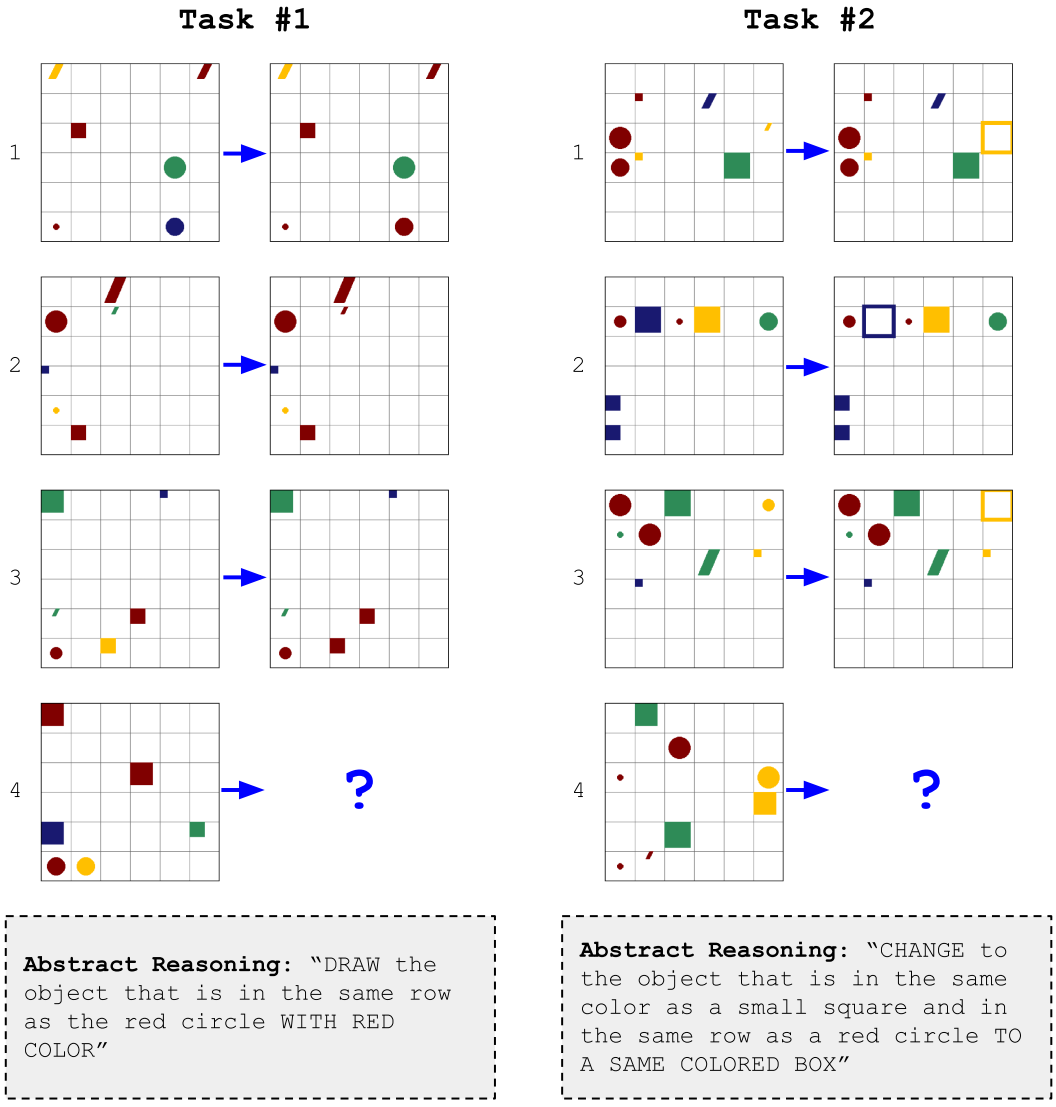


**Figure 8:** Two simplified abstract reasoning challenges with ReaSCAN. The task mimics human reasoning tests: given a set of input–output (input on the left and output on the right) pairs, the task taker needs to guess the output for the last input. For each task, we provide one potential abstract reasoning statement to solve the task.

# Datasheet for 'ReaSCAN: Compositional Reasoning in Language Grounding'

## A Motivation

### A.1 For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description

The dataset was created as a new benchmark for evaluating compositional generalization of neural models by addressing the limitations of gSCAN. We find that gSCAN has three major limitations: (1) its set of instructions is so constrained that compositional interpretation is not required; (2) the distractor objects in its grounded scenarios are mostly not relevant for accurate language understanding; and (3) in many examples, not all modifiers in the command are required for successful navigation, which further erodes the need for compositional interpretation and inflates model performance scores.

### A.2 Who created this dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?

The dataset was created by Zhengxuan Wu, Elisa Kreiss, and Christopher Potts at Stanford University, and Desmond C. Ong at National University of Singapore.

### A.3 Who funded the creation of the dataset? If there is an associated grant, please provide the name of the grantor and the grant name and number

This research is supported in part by the National Research Foundation, Singapore, under its AI Singapore Program (AISG Award No: AISG2-RP-2020-016), and in part by a Stanford HAI Hoffman–Yee grant.

## B Composition

### B.1 What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description.

This dataset is a synthetic dataset. Each instance contains:

1. A command–world pair where the command is a synthetic English sentence and the world is a synthetic $n \times n$ grid-world, where we fix $n = 6$, using the open-sourced MiniGym from Open-AI.[9] The world is represented by a list of objects that are present in the world. Each object is defined by a unique tensor.
2. An agent initial position and facing direction.
3. The position of the referent target.
4. The gold label, which is the correct action sequence the agent can execute to reach and operate on the referent target.

### B.2 Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set? If the dataset is a sample, then what is the larger set? Is the sample representative of the larger set (e.g., geographic coverage)? If so, please describe how this representativeness was validated/verified. If it is not representative of the larger set, please describe why not (e.g., to cover a more diverse range of instances, because instances were withheld or unavailable).

ReaSCANv1.0 is provided in our project Github repository.[10] It is sampled from a larger set. As our dataset provides a general framework that can scale up to many more examples, it reaches a point where regular computing resources might not sufficient. As a result, we randomly sample a subset from our larger pool. We document known potential artifacts from our generation process in Appendix B.

[9] https://github.com/maximecb/gym-minigrid

[10] https://github.com/frankaging/Reason-SCAN

### B.3 What data does each instance consist of? "Raw" data (e.g., unprocessed text or images)or features? In either case, please provide a description

We provide details about our instances in Section B.1.

### B.4 Is there a label or target associated with each instance? If so, please provide a description.

The gold label is the correct action sequence. The agent can execute the action sequence to reach and operate on the referent target.

### B.5 Is any information missing from individual instances? If so, please provide a description, explaining why this information is missing (e.g., because it was unavailable). This does not include intentionally removed information, but might include, e.g., redacted text.

Everything is included. No data is missing.

### B.6 Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)? If so, please describe how these relationships are made explicit.

N/A.

### B.7 Are there recommended data splits (e.g., training, development/validation, testing)? If so, please provide a description of these splits, explaining the rationale behind them.

As our dataset is designed for compositional generalization, we provide train/dev/test splits as well as compositional splits in our released dataset. We also provide scripts to generate these splits in our code repository.

### B.8 Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description.

N/A.

### B.9 Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?

Yes, the dataset is self-contained.

### B.10 Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctorpatient confidentiality, data that includes the content of individuals non-public communications)? If so, please provide a description.

No, this is a synthetic dataset.

### B.11 Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? If so, please describe why.

No, this is a synthetic dataset containing synthetic navigation instructions in English and synthetic grid worlds.

### B.12 Does the dataset relate to people? If not, you may skip the remaining questions in this section.

No, this is a synthetic dataset and does not contain any human-assigned labels.

### B.13 Does the dataset identify any subpopulations (e.g., by age, gender)? If so, please describe how these subpopulations are identified and provide a description of their respective distributions within the dataset.

N/A.

**B.14 Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset? If so, please describe how.**

N/A.

**B.15 Does the dataset contain data that might be considered sensitive in any way (e.g., data that reveals racial or ethnic origins, sexual orientations, religious beliefs, political opinions or union memberships, or locations; financial or health data; biometric or genetic data; forms of government identification, such as social security numbers; criminal history)?**

N/A.

## C Collection Process

N/A. This is a synthetic dataset containing synthetic navigation instructions in English and synthetic grid worlds. As a result, we do not collect any human data.

## D Preprocessing/cleaning/labeling

**D.1 Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? If so, please provide a description. If not, you may skip the remainder of the questions in this section.**

No, the synthetic dataset is provided as-is.

**D.2 Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? If so, please provide a link or other access point to the "raw" data.**

N/A.

**D.3 Is the software used to preprocess/clean/label the instances available? If so, please provide a link or other access point**

N/A.

## E Use

**E.1 Has the dataset been used for any tasks already? If so, please provide a description.**

No, this is our first release of the dataset.

**E.2 Is there a repository that links to any or all papers or systems that use the dataset? If so, please provide a link or other access point.**

No, this is our first release of the dataset.

**E.3 What (other) tasks could the dataset be used for?**

This dataset is designed for evaluating compositional generalization of neural models as a synthetic navigation task. This task can be used for referring expression resolution as well.

### E.4 Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? For example, is there anything that a future user might need to know to avoid uses that could result in unfair treatment of individuals or groups (e.g., stereotyping, quality of service issues) or other undesirable harms (e.g., financial harms, legal risks) If so, please provide a description. Is there anything a future user could do to mitigate these undesirable harms?

There is minimal risk for harm: this is a synthetic dataset and does not contain any human labels.

### E.5 Are there tasks for which the dataset should not be used? If so, please provide a description.

No, this dataset is used for training neural models that solve the synthetic task posed by the dataset only. The dataset should not be used directly in any real-world applications.

## F Distribution

### F.1 Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? If so, please provide a description.

Yes, the dataset is publicly available on the internet.

### F.2 How will the dataset will be distributed (e.g., tarball on website, API, GitHub)? Does the dataset have a digital object identifier (DOI)?

The dataset is publicly avaliable at our Github repository.

### F.3 When will the dataset be distributed?

The dataset is first released in 2021.

### F.4 Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so, please describe this license and/or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions.

Our dataset has a Creative Commons Attribution 4.0 International License. The dataset is publicly available on the internet. People are allowed to use our scripts to generate their own version of ReaSCAN as well.

### F.5 Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any supporting documentation.

Unknown.

## G Maintenance

### G.1 Who is supporting/hosting/maintaining the dataset?

Zhengxuan Wu is supporting/maintaining the dataset.

### G.2 How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

wuzhengx@stanford.edu

### G.3 Is there an erratum? If so, please provide a link or other access point.

There is not an explicit erratum, but updates and fixes with the dataset will be reflected in our Github repository. The first release is versioned as ReaSCANv1.0. Any subsequent releases will have higher version numbers.

**G.4 Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)? If so, please describe how often, by whom, and how updates will be communicated to users (e.g., mailing list, GitHub)?**

This will be posted on the dataset Github repository.

**G.5 If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)? If so, please describe these limits and explain how they will be enforced.**

N/A.

**G.6 Will older versions of the dataset continue to be supported/hosted/maintained? If so, please describe how. If not, please describe how its obsolescence will be communicated to users.**

N/A.

**G.7 If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? If so, please provide a description. Will these contributions be validated/verified? If so, please describe how. If not, why not? Is there a process for communicating/distributing these contributions to other users? If so, please provide a description.**

Others may do so and should contact the original authors about incorporating fixes/extensions.