A Details of rule-RAVEN dataset

A.1 Rules in rule-RAVEN

In the joint training stage, a specific rule contains 4 elements on different attributes, e.g., `(Number:add, Color:progression_2, Size:constant, Shape:max)`. These rules meet the structural and functional requirements stated in Section 3.2 The embodiment of each rule on each attribute are shown in Figure 7. For example, the '`Color:progression_2`' means: in the same row of panels, the image color gradually darkens by 2 degrees from left to right.

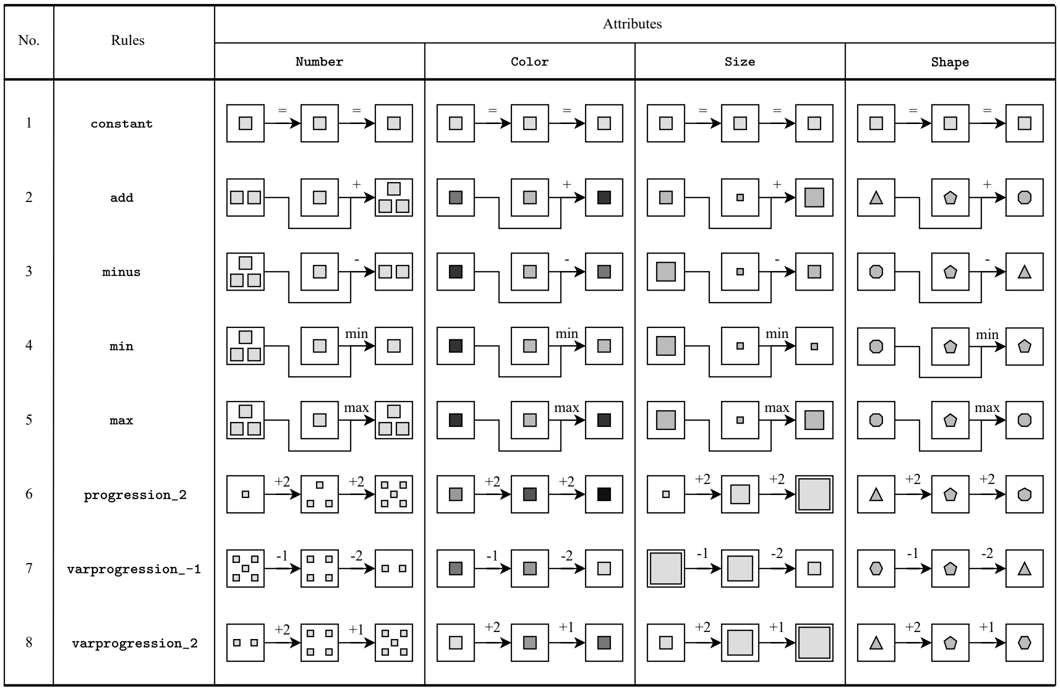

Figure 7: The embodiment of all 8 rules involved in the joint training stage on the 4 attributes, i.e., `(Number, Color, Size, Shape)`.

In the speaker pre-training stage, the rules involved in each attribute are shown in Figure 8, which do not overlap with those in Figure 7, and have similar semantics to rules No.6, No.7, and No.8 in Figure 7. Therefore, pre-training the speaker with these rules does not provide agents with prior knowledge about the rules in the joint training stage, which only helps the speaker acquire reasoning capabilities to cope with the drifting context.

A.2 A RPM case in rule-RAVEN

Figure 9 shows one RPM case of the rule-RAVEN dataset used in our experiments. The form of this case is consistent with Figure2. Figure2 is just for the convenience of illustration, only 1 attribute of the rule is drawn, but the actual rule consists of 4 attributes, i.e., `(Number, Color, Size, Shape)`.

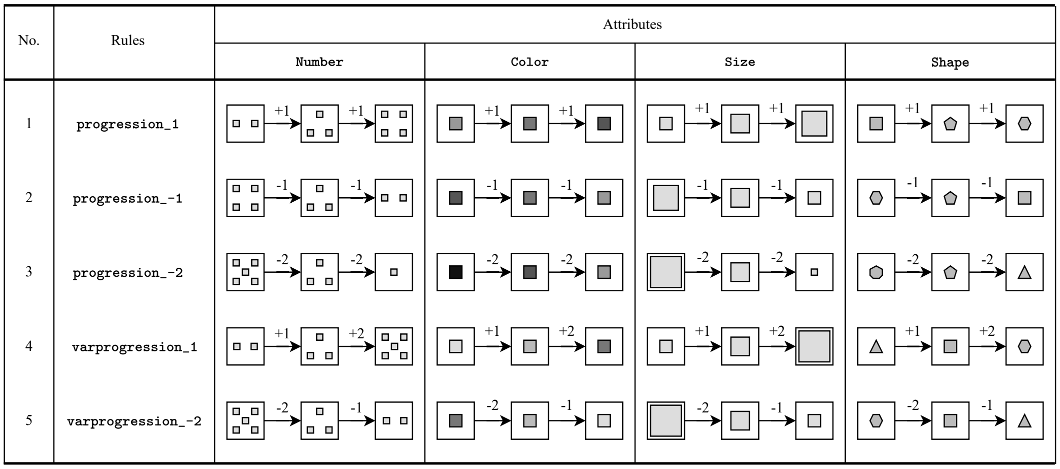

Figure 8: The embodiment of 4 extra rules involved in the speaker pre-training stage on the 4 attributes, i.e., `(Number, Color, Size, Shape)`. These rules involved in the pre-training phase do not overlap with rules in the joint training stage.

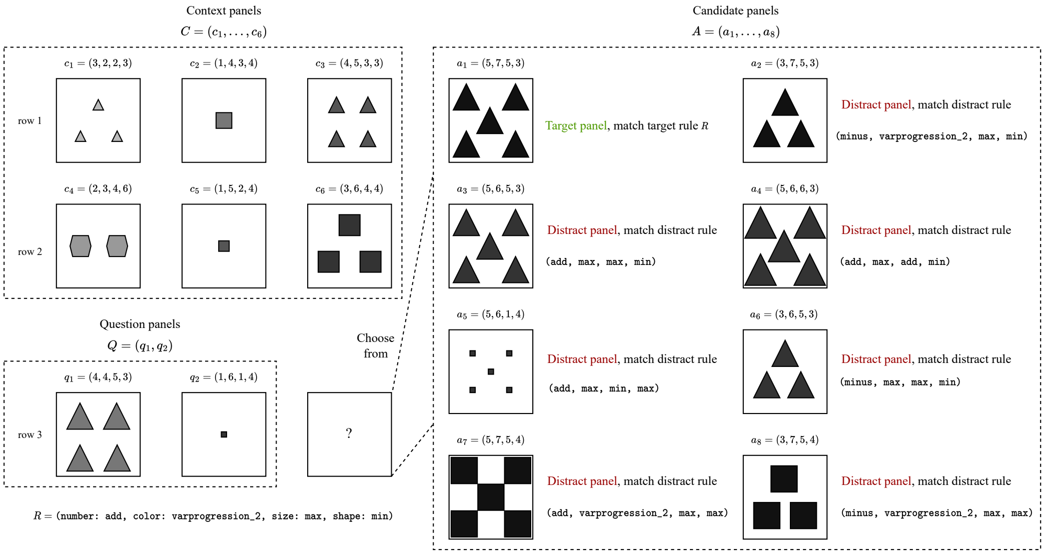

Figure 9: An example problem of the rule-RAVEN dataset with 4 attributes (i.e., `(Number, Color, Size, Shape)`).

A.3 Details of generalization data splits

Figure 10 shows an example of four levels of generalization data splits described in Section 4.2.

For `ID`, `Inpo-ood`, and `Expo-ood-L2` generalization levels, we first get $7^4 = 2401$ rule combinations based on rules No.1 to 7 in Figure 7 on all 4 attributes. We then randomly sample 300 rule combinations from these 2401 rule combinations and generate 10 problems for each rule combination as the `Inpo-ood` data split (with a total of $300 \times 10 = 3000$ problems). For each of the remaining $7^4 - 300 = 2101$ rule combinations, we generate 10 problems as the train data split (for `ID`, `Inpo-ood`, and `Expo-ood-L2`) and 10 problems as the `ID` data split (with a total of $2101 \times 10 = 21010$ problems for training and 21010 for `ID`). We finally combine rule No.8, excluded in the previous process, with the rules on other attributes to get $8^4 - 7^4 = 1695$ rule combinations and generate 10 problems for each rule combination as `Expo-ood-L2` data split (with a total of $1695 \times 10 = 16950$ problems).

For the `Expo-ood-L1` generalization level, we exclude rule No.6 on attribute 1, rule No.3 on attribute 2, rule No.4 on attribute 3, and rule No.8 on attribute 4, getting 2401 rule combinations remaining. By comparing the accuracy difference between `Expo-ood-L1` and `Expo-ood-L2` levels (both `Expo-ood-L1` and `Expo-ood-L2` exclude one rule on each attribute, but on `Expo-ood-L1` level, rules excluded on one attribute appear on other attributes), we can check whether the language describing the rules can help the listener apply the rules across attributes. To compare fairly with `Expo-ood-L2` level, we randomly exclude 300 rule combinations and generate 10 problems for each of the remaining 2101 rule combinations as the training set for `Expo-ood-L1` level (with a total of $2101 \times 10 = 21010$ problems). We combine the rules excluded in the previous process with the rules on other attributes to get $8^4 - 7^4 = 1695$ rule combinations and generate 10 problems for each rule combination as `Expo-ood-L1` data split (with a total of $1695 \times 10 = 16950$ problems).

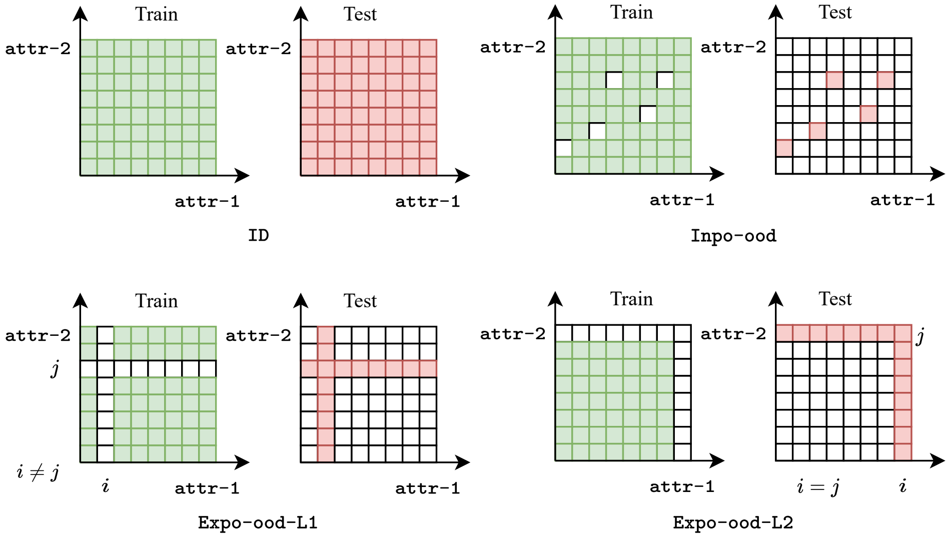

Figure 10: An example (with 2 attributes, `(attr-1, attr-2)`) to illustrate 4 generalization levels.

B Details of Agent Model

The model diagram of the speaker and listener is shown in Figure 11. Some hyperparameters of the model are listed in Table 4. Please refer to the source code in the supplementary materials for more implementation details.

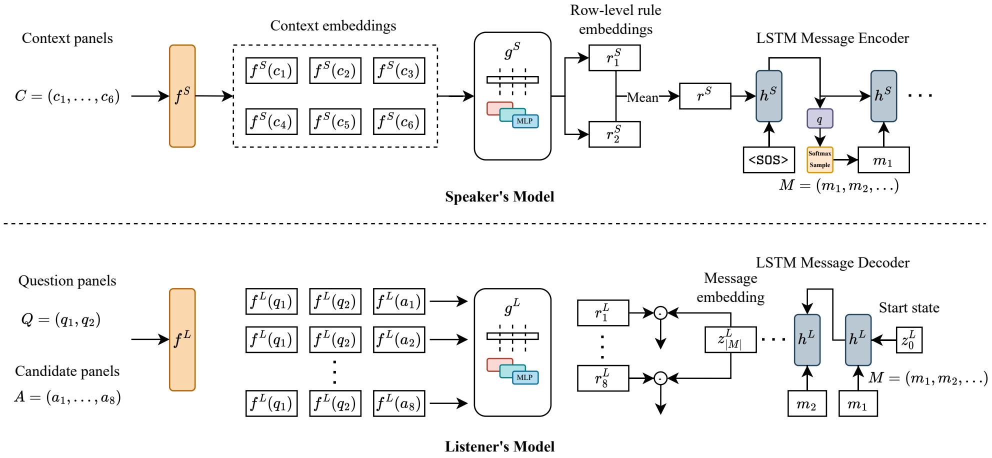

Figure 11: The Speaker's and Listener's Model.

Table 4: Some hyperparameters of the model.

Hyperparameter	Value of $N \in \{20, 30, 40, 80\}$
f^S and f^L output dim	$[80, 120, 160, 240]$
Groups of g^S and g^L	$[80, 120, 160, 240]$
Experts of g^S and g^L	5
g^S and g^L output dim	$[400, 600, 800, 1200]$
h^S and h^L hidden dim	$[400, 600, 800, 1200]$

C Results of Diverse Language Sizes

The proposed two-stage training method does not introduce constraints on the size of the emerged language (i.e., `message_length` and `vocabulary_size`). The reason is that the speaker updates f^S, g^S, and h^S in the first stage of training, while only load parameters of f^S, g^S (without h^S) in the second stage. Therefore, even if the $\mathbb{1}(r_i, m_i)$ operator on message encoder h^S constrains the equal size between message m_i and rule r_i in the first stage, we can the training a new h^S with reconfigurable language size in the second stage (joint) training.

We also provide results with diverse language sizes (Table 5) in our reasoning game and get similarly high generalization performance (accuracy ~ 0.95).

Table 5: Generalization accuracy with different language sizes (`message_length` M, `vocabulary_size` V) and attribute values N.

(M, V, N)	`ID`	`Inpo-ood`
$(6, 15, 20)$	0.9539 ± 0.0018	0.9528 ± 0.0015
$(6, 30, 20)$	0.9522 ± 0.0015	0.9513 ± 0.0031
$(6, 15, 30)$	0.9429 ± 0.0053	0.9385 ± 0.0045
$(6, 30, 30)$	0.9411 ± 0.0022	0.9362 ± 0.0014

D Token-level Analysis of the Emerged Language

We compute the probability distribution $P(message|attribute, rule)$ of a randomly selected seed and provide the most probable tokens for a given attribute and rule, as shown in Table 6.

Table 6: Given attributes and rules, the most probable tokens at each position.

Rules	`Color`	`Number`	`Size`	`Shape`
`add`	**K**, C, K, H	B, C, B, C	**K**, L, K, K	B, K, B, C
`minus`	**B**, L, B, B	K, O, G, O	**G**, G, G, H	J, J, J, J
`min`	**K**, C, K, C	J, L, J, C	**K**, O, K, K	B, B, B, C
`max`	**B**, B, B, K	K, O, K, K	**K**, G, J, B	F, B, K, C
`constant`	**K**, B, K, K	B, K, C, K	**K**, C, K, C	K, B, K, C
`progression_2`	**B**, L, B, B	K, O, K, O	**G**, G, H, H	J, O, J, J
`varprogression_-1`	**K**, C, K, **C**	B, J, J, **C**	**K**, O, K, **C**	B, K, B, **C**

The results indicate that tokens exhibit regular patterns (i.e., language-like syntax and compositionality) for different attributes and rules. For example, almost all rules related to attribute '`color`' start with tokens 'K' or 'B', and attribute '`size`' start with tokens 'K' or 'G'. On another dimension, the rule '`varprogression_-1`' across all attributes ends with token 'C'.