Generalized Planning for the Abstraction and Reasoning Corpus

Chao Lei[1] Nir Lipovetzky[1] Krista A. Ehinger[1]

[1]The University of Melbourne

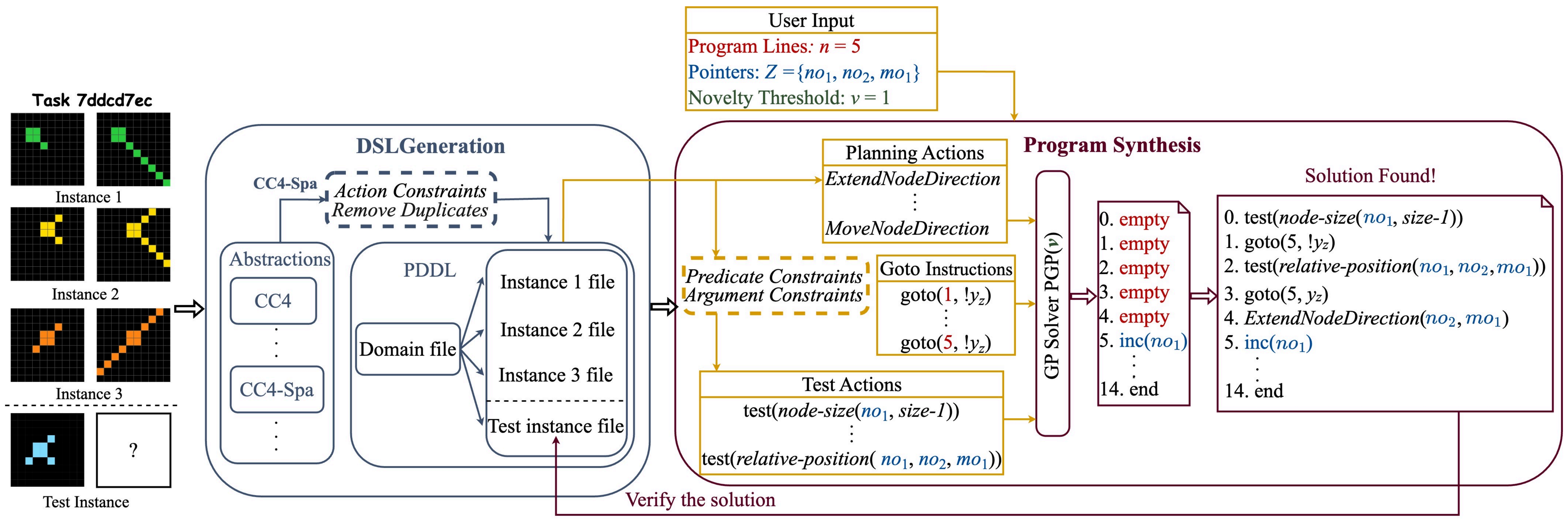


Figure 1. Pipeline sketch of **Generalized Planning for Abstract Reasoning (GPAR)**, a two-stage system that employs GP to solve ARC tasks. The DSL generation stage encompasses a collection of abstractions to generate a domain file and associated instance files for each ARC task. The program synthesis stage uses a generalized planning solver PGP(v) to generate a program that can map the input image to the output image by executing the planning program on the corresponding initial state in each training instance.

Scan here to read the full paper.

Introduction

- Abstraction and Reasoning Corpus (**ARC**) is a set of **abstract visual reasoning tasks** that measure the gap of **abstract reasoning** and **generalization capacities** between **humans** and **AI**[1].

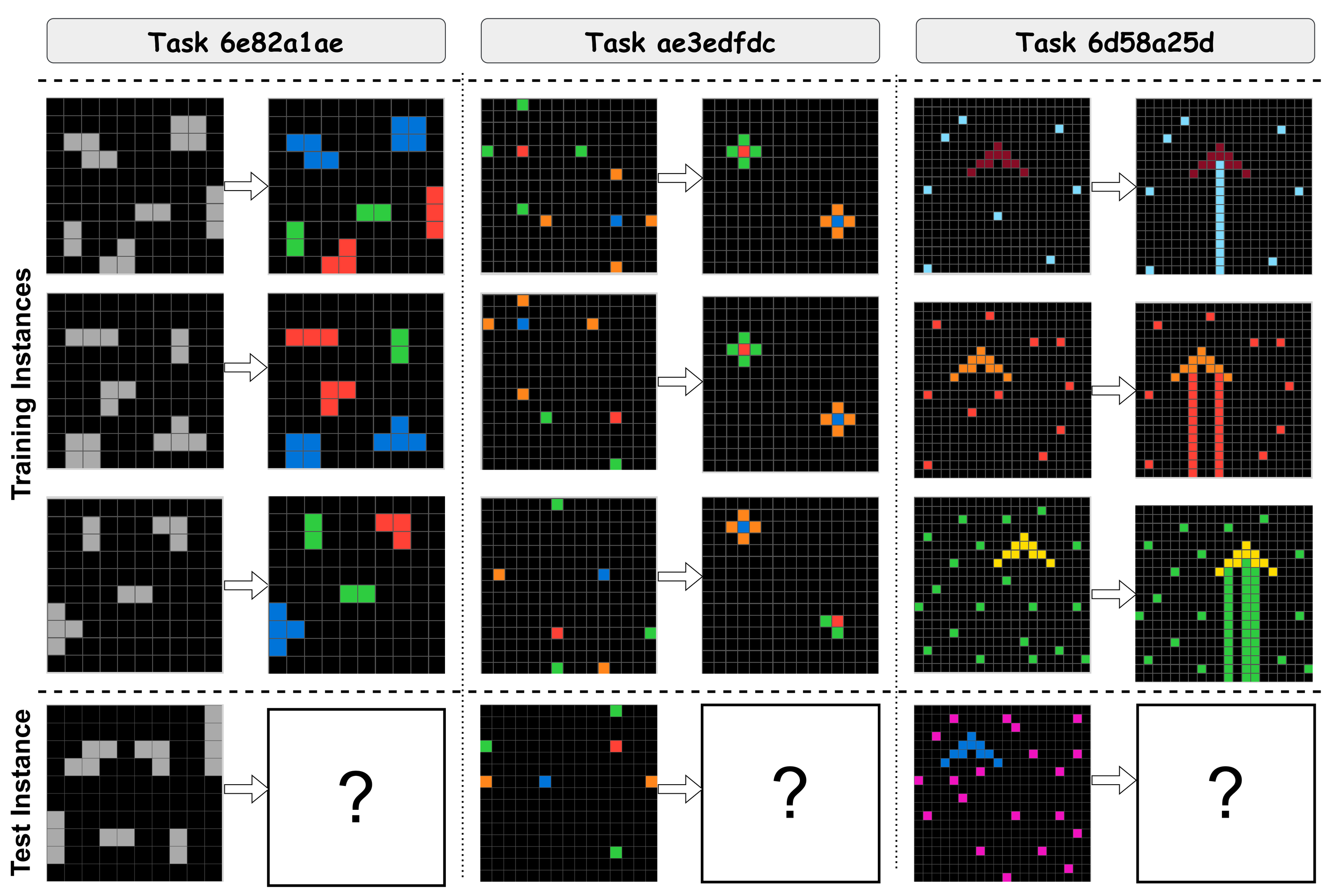


Figure 2. Three example tasks from the ARC. For a given task, each row contains an input-output image pair as a training instance, and the bottom row is the test instance. The goal of the solver is to learn from the training instances how to generate the output for the test instance.

Method Part 1: Domain-Specific Knowledge

Abstraction:

- Abstraction enables object awareness in GPAR to allow actions to modify a group of pixels at once rather than individually, resulting in a smaller search space.
- Multiple abstraction considerations in GPAR can compensate for the limitations of a certain abstraction.

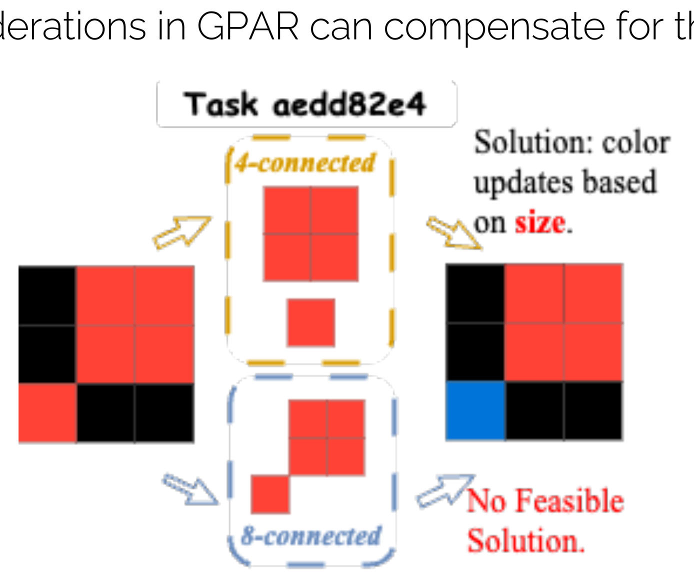
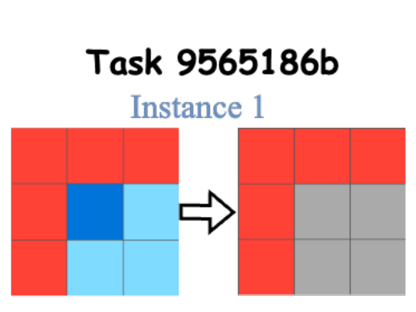


Figure 3. A 4- vs. 8-connected abstractions example.

PDDL Representation:

- PDDL describes each ARC task through a single domain file and a finite set of instance files, one for each input-output image pair.



File for Domain

```
(define (domain ARC-9565186b)
(:types node pixel size color - object)
(:predicates (node-color ?no - node ?co - color)
 (node-size ?no - node ?si - size)
 (contain-pixel ?no - node ?pi - pixel)
 (pixel-color ?pi - pixel ?co -color))
(:action UpdateColor
 :parameters (?no - node, ?co1 - color ?co2 - color)
 :precondition (node-color ?no ?co1)
 :effect @PixelColorUpdate(?no, ?co2)))
```

File for Instance 1

```
(define (problem 9565186b-1)
 (:domain ARC-9565186b)
 (:objects pixel-0-0 pixel-0-1 pixel-0-2 pixel-1-0 pixel-1-1 pixel-1-2
      pixel-2-0 pixel-2-1 pixel-2-2 - pixel
      node-1 node-2 node-3 - node
      size-1 size-3 size-4 size-5 - size
      red blue grey cyan - color)
(:INIT (node-size node-1 size-5) (node-color node-1 red)
 (node-size node-2 size-1) (node-color nod-2 blue)
 (node-size node-3 size-3) (node-color nod-3 cyan)
 (pixel-color pixel-0-0 red),...,(pixel-color pixel-2-2 cyan)
 (contain-pixel node-1 pixel-0-0),...,(contain-pixel node-3 pixel-2-2)
(:Goal(AND (pixel-color pixel-0-0 red),...,(pixel-color pixel-2-2 grey))))
```

Figure 4. A PDDL example for a fragment of an ARC task. Parameters of action schemes and predicates are preceded by the "?" symbol, and external functions are preceded by the "@" symbol.

Method Part 2: Generalized Planning

Domain Knowledge:

i. Duplicated Abstraction Removal: Abstractions that generate identical instances are avoided.

ii. Action Pruning: Actions that result in necessary nodes' positions, colors, or sizes updating are considered.

iii. Predicate Constraint: A predicate can work as a condition, iff the condition is not always true among all training and test input images.

iv. Argument Constraint: The arguments chosen for predicates describe attributes that exist in all training and test input images.

v. Structural Restrictions: Part of the planning program that iterates over all possible combinations of pointer values is automatically generated before the search starts. Other instructions are restricted by appearance sequences.

GPAR Program Synthesis Process:

- GPAR leverages PGP(v) as a GP solver, taking program lines n, pointers Z, and novelty thresholds v as input. The solution is a planning program that can map the input image to the output image.

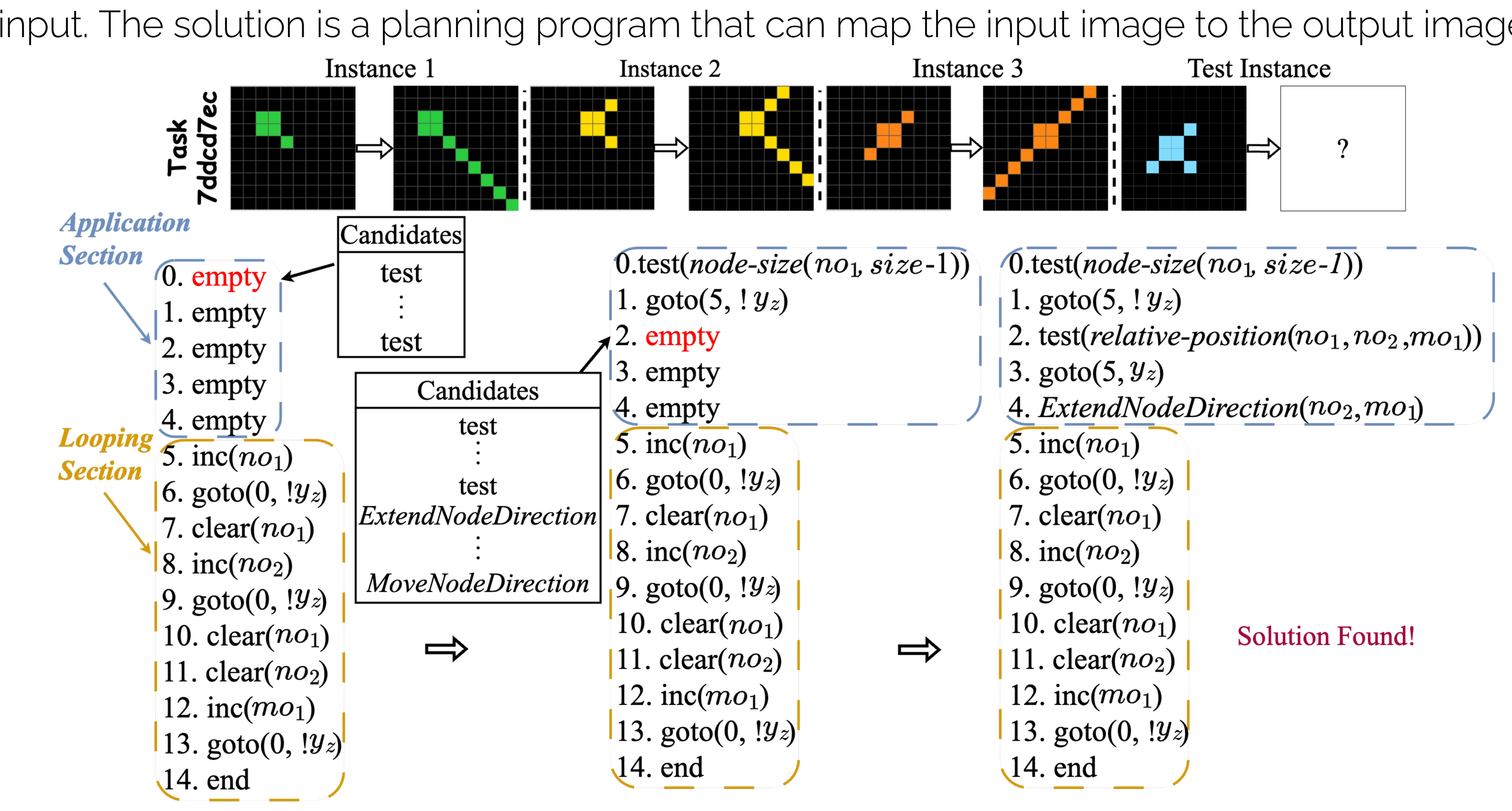


Figure 1. An illustration of the planning process with the application section and the looping section. Lines 0 and 1 ensure that no_1 indexes the square node, and lines 2 and 3 constrain the no_2 to point to the single-pixel node, while mo_1 indexes the correct spatial relation between no_1 and no_2.

Results

Model	Task Type	Training Accuracy		Testing Accuracy	
ARGA	movement	18/31	(58.06%)	17/31	(54.84%)
	recolor	25/62	(40.32%)	23/62	(37.10%)
	augmentation	20/67	(29.85%)	17/67	(25.37%)
	all	63/160	(39.38%)	57/160	(35.62%)
Kaggle First Place	movement	**21**/31	(**67.74%**)	15/31	(48.39%)
	recolor	23/62	(37.10%)	28/62	(45.16%)
	augmentation	**35**/67	(**52.24%**)	21/67	(31.34%)
	all	79/160	(49.38%)	64/160	(40.00%)
GPAR	movement	20/31	(64.52%)	**19**/31	(**61.30%**)
	recolor	**41**/62	(**66.13%**)	**39**/62	(**62.90%**)
	augmentation	25/67	(37.31%)	**23**/67	(**34.33%**)
	all	**86**/160	(**53.75%**)	**81**/160	(**50.63%**)

Table 1. Performance of Abstract Reasoning with Graph Abstractions (ARGA)[2], Kaggle First Place and GPAR over 160 object-centric ARC tasks. Training accuracy is the number of tasks whose solutions solve all the training instances. Testing accuracy is the number of tasks whose solutions also generate the correct output images for all test instances.

Contributions

- GPAR achieves **state-of-the-art performance** over the ARC benchmark.
- A novel method to solve abstract reasoning tasks based on **generalized planning**.
- A domain-specific language encoding based on **PDDL**.
- The usage of novel **ARC domain knowledge** to reduce the size of the solution space.

References

[1] Chollet, F. 2019. On the Measure of Intelligence. *arXiv preprint arXiv:1911.01547*.

[2] Xu, Y.; Khalil, E. B.; and Sanner, S. 2023. Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, AAAI, 4115–4122.

AAAI 2024 Main Track