Graphs, Constraints and Search for the Abstraction and Reasoning Corpus

Yudong (William) Xu, Elias B. Khalil, Scott Sanner

January 24th, 2023

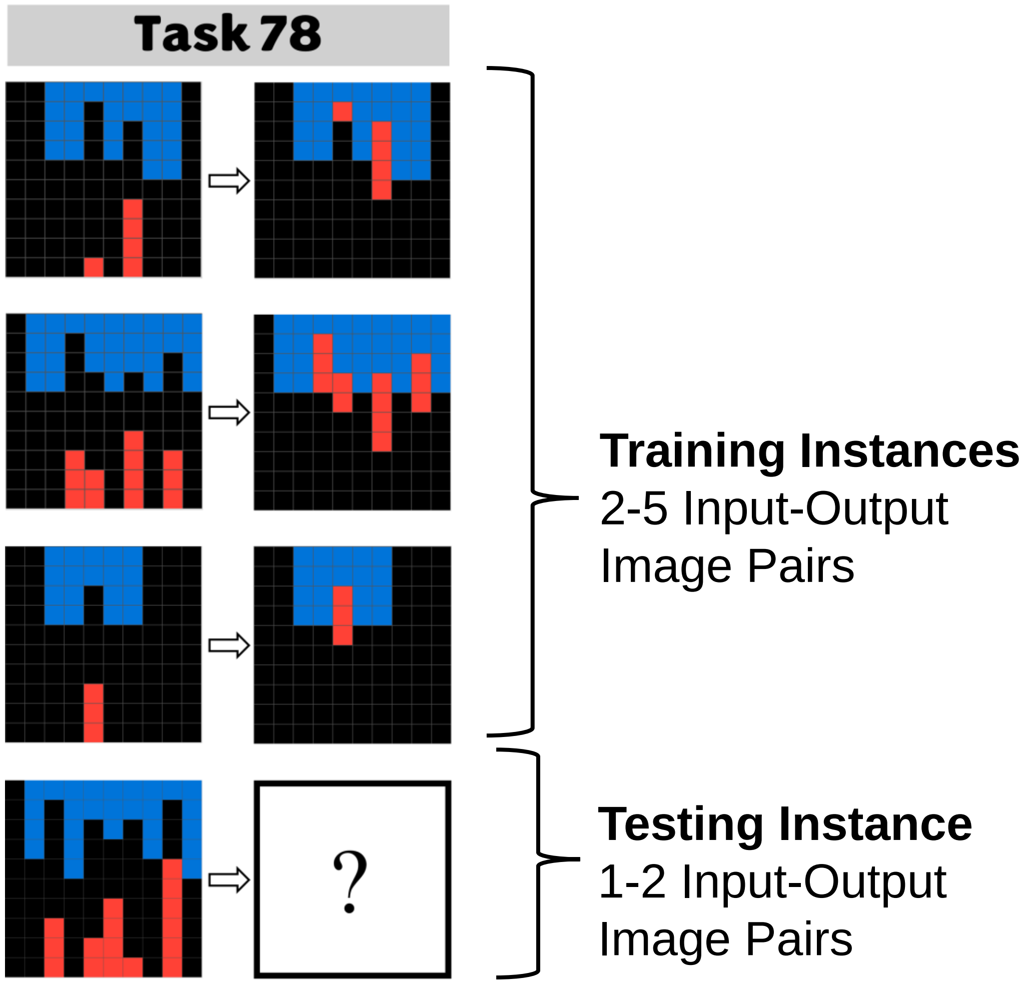
To appear at The 37th AAAI Conference on Artificial Intelligence

UNIVERSITY OF TORONTO

Abstraction and Reasoning Corpus (ARC)

Abstraction and Reasoning Corpus (ARC)

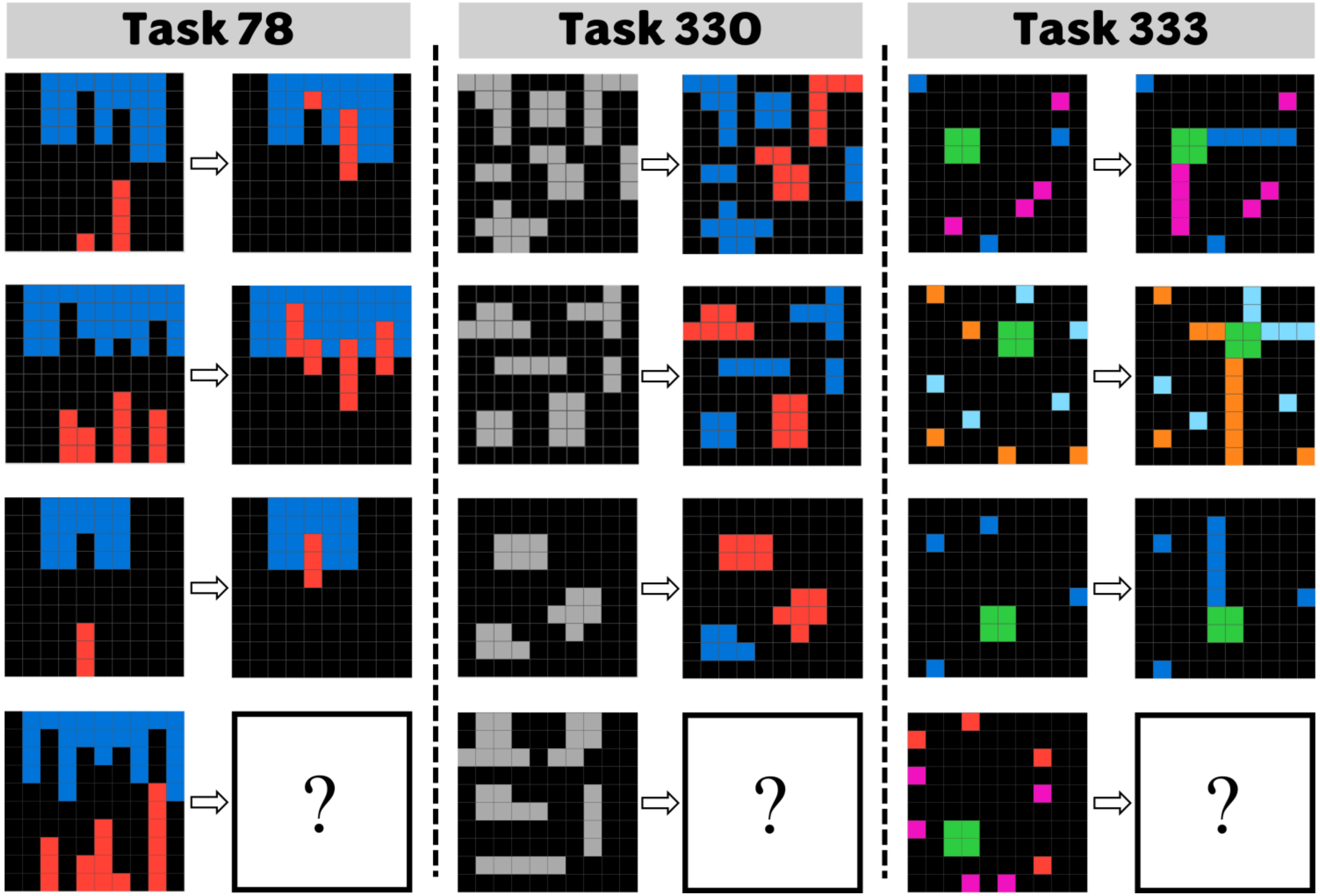
- A collection of 1000 image-based reasoning tasks (800 publicly available)
- Each task:
  given **input image**
  solve for **output image**



UNIVERSITY OF TORONTO

Abstraction and Reasoning Corpus (ARC)

- A collection of 1000 image-based reasoning tasks (800 publicly available)
- Each task:
  given **input image**
  solve for **output image**



Play with ARC: https://arc-editor.lab42.global/

UNIVERSITY OF TORONTO

ARC Kaggle Challenge (2020)

- 3 Months
- 913 Teams
- 3 attempts per task

First Place:
21/100 hidden test tasks solved

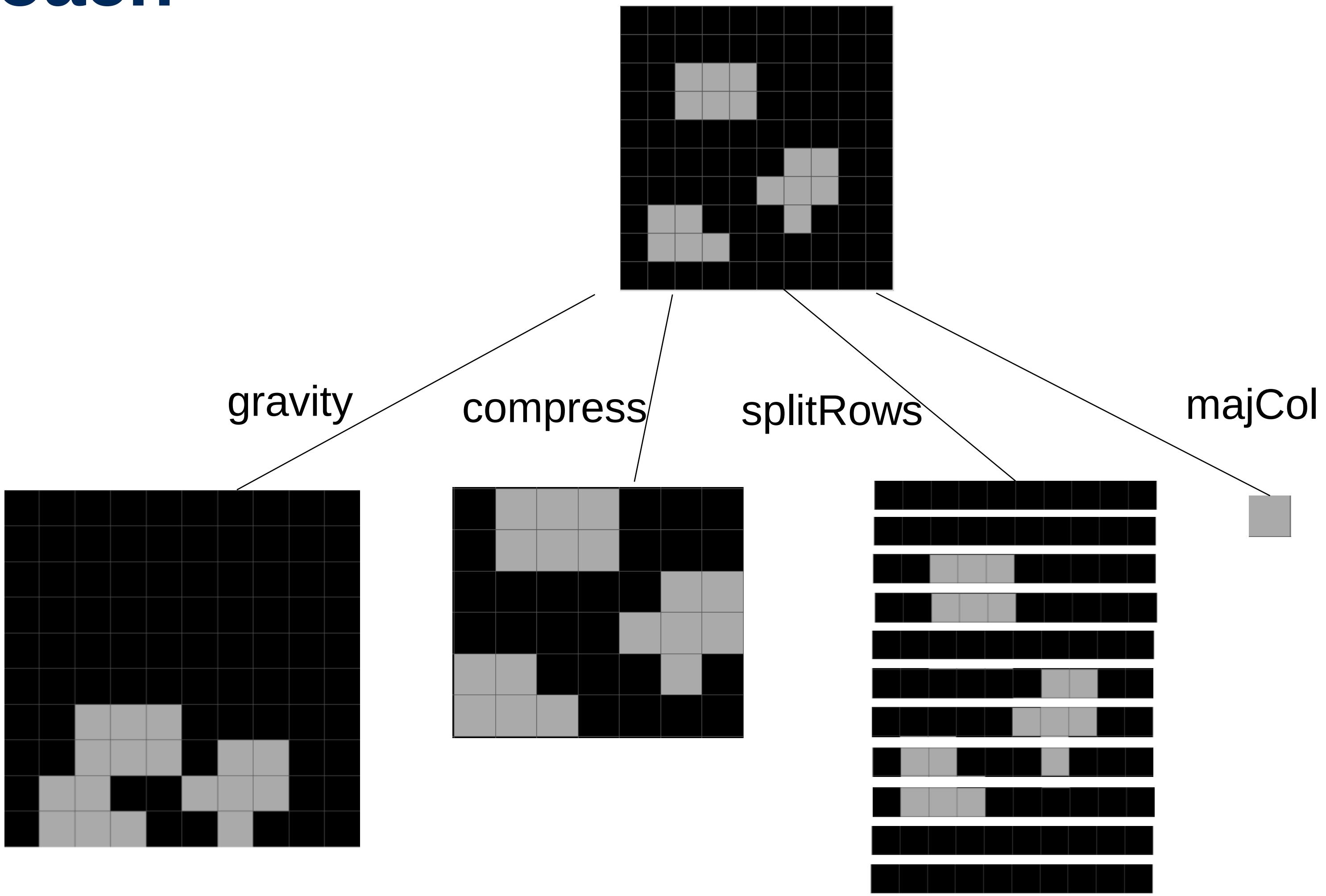
Ensemble of 5 top models:
29/100 solved

State of the Art Approach

- **Develop a Domain Specific Language (DSL):** a set of functions (transformations) that can express the solution.

- **Search through the DSL** to find a program (sequence of functions) that produces the correct output images when applied to input images.

Example DSL functions from First Place

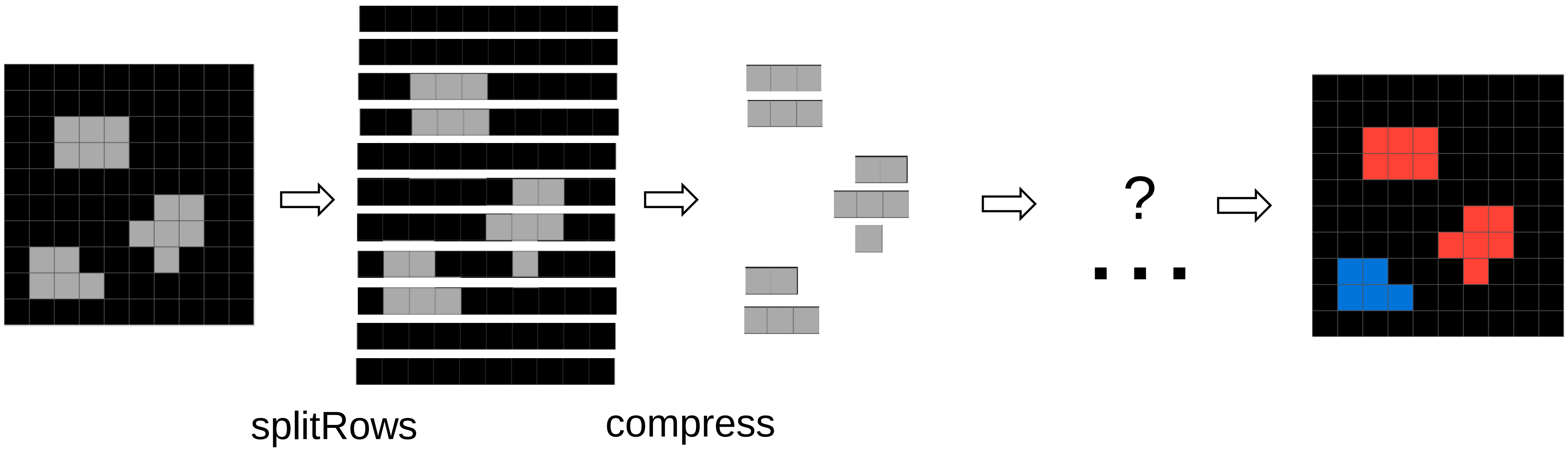


Solved 200 Tasks by hand,

composed DSL including 142 unary transformations

UNIVERSITY OF TORONTO

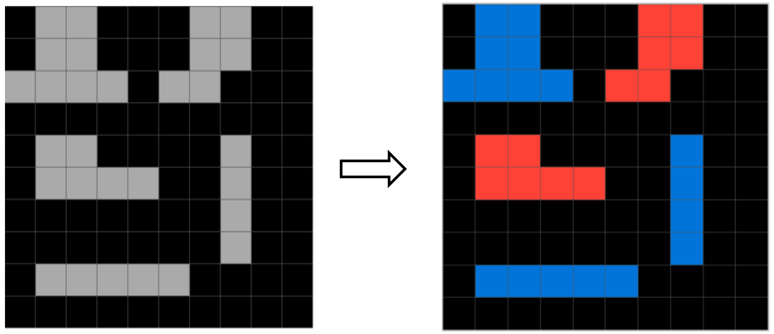
State of the Art Approach



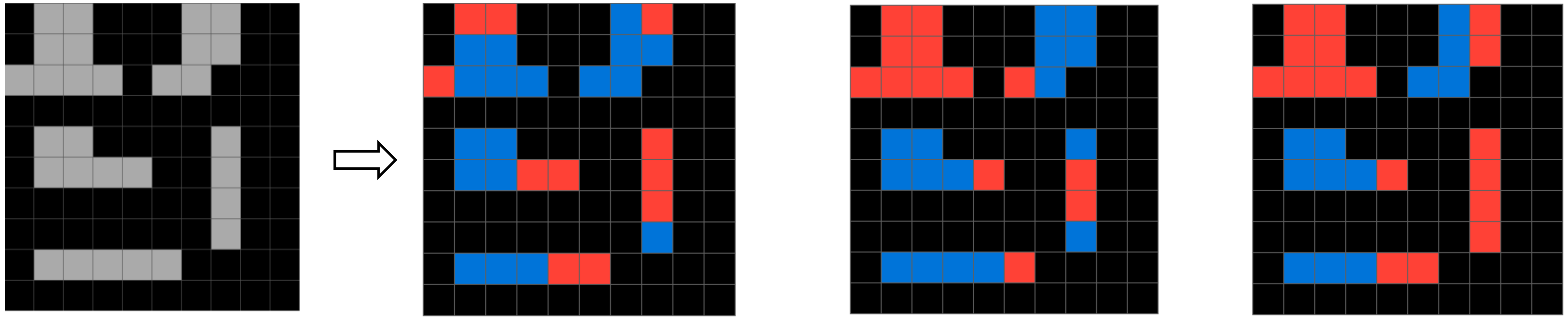
First place can’t solve this simple problem!

Testing Instance

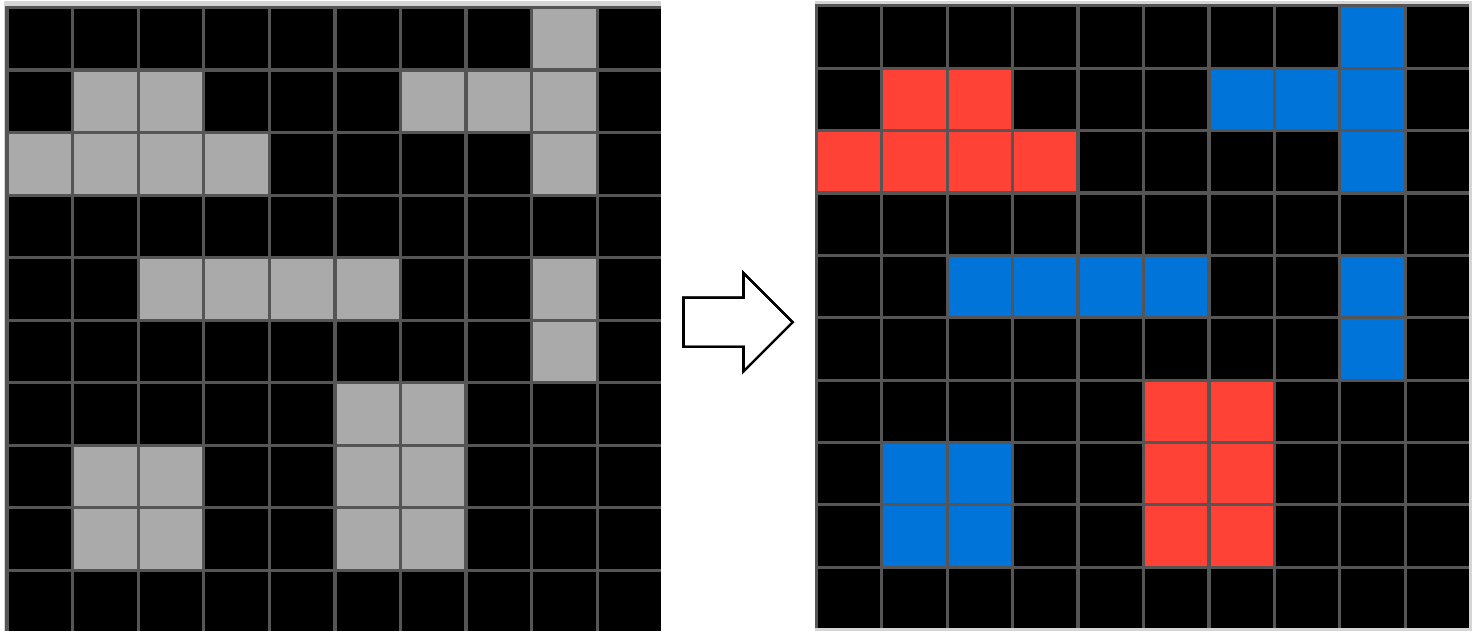
Expected Solution



First Place Top 3 Attempted Solution



How does a human solve it?



“Color the size 6 grey objects red, color the non size 6 grey objects blue”

When humans solve ARC tasks, we tend to provide the solutions in terms of **objects**

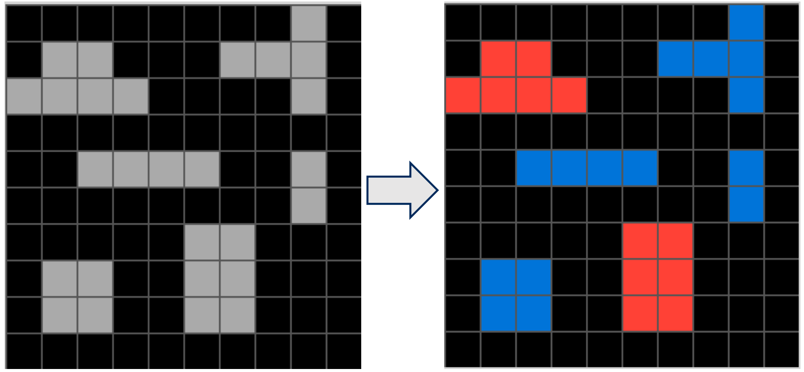
Missing from state of the art approaches

UNIVERSITY OF TORONTO

Abstract Reasoning with Graph Abstractions (ARGA)

ARGA Approach

- **Recognize objects**
- **Develop an object-centric Domain Specific Language (DSL)**
- **Search through the DSL to find modification to the objects**



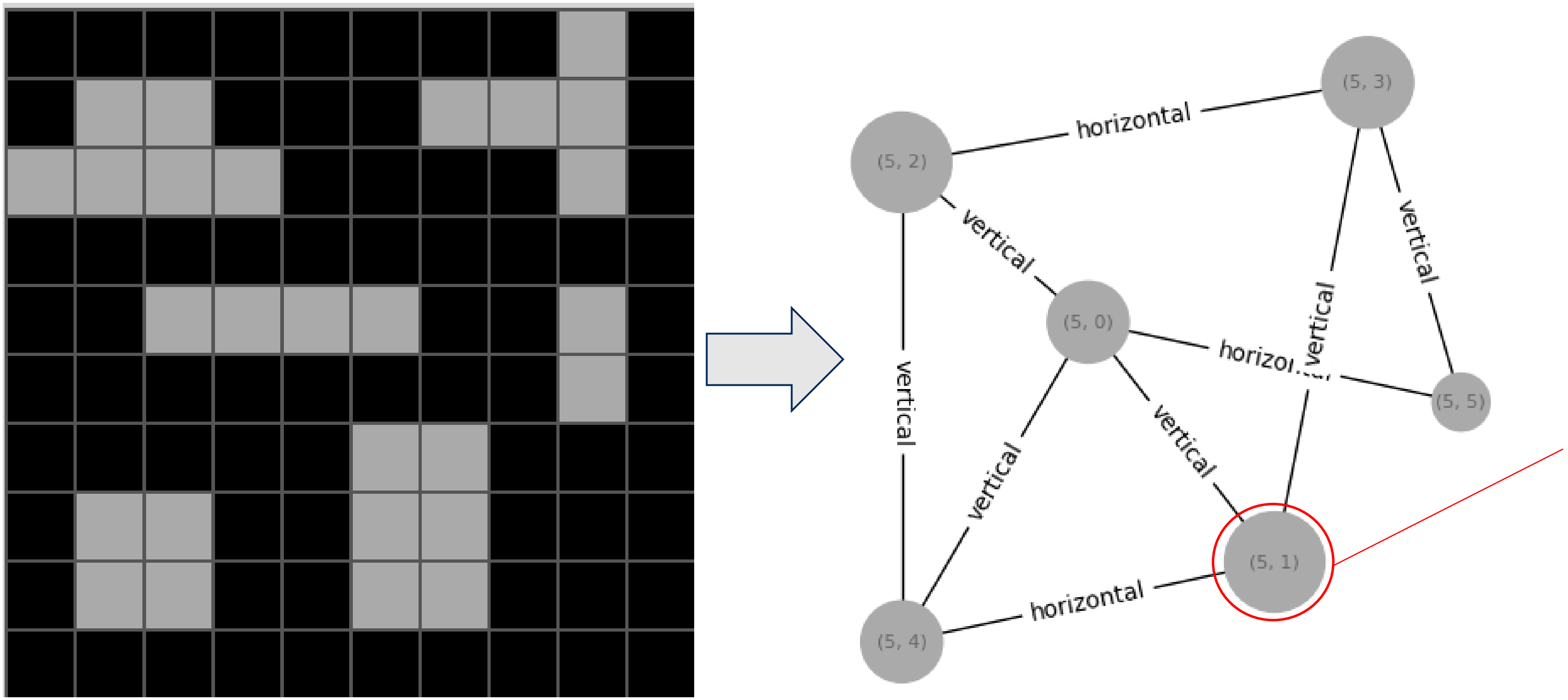
"Color the size 6 grey objects red, color the rest blue"

more closely resemble human solutions

Recognizing Objects: Graph Abstractions

Recognize Objects: Graph Abstractions

ARGA recognizes the objects by transforming the images into graph abstractions



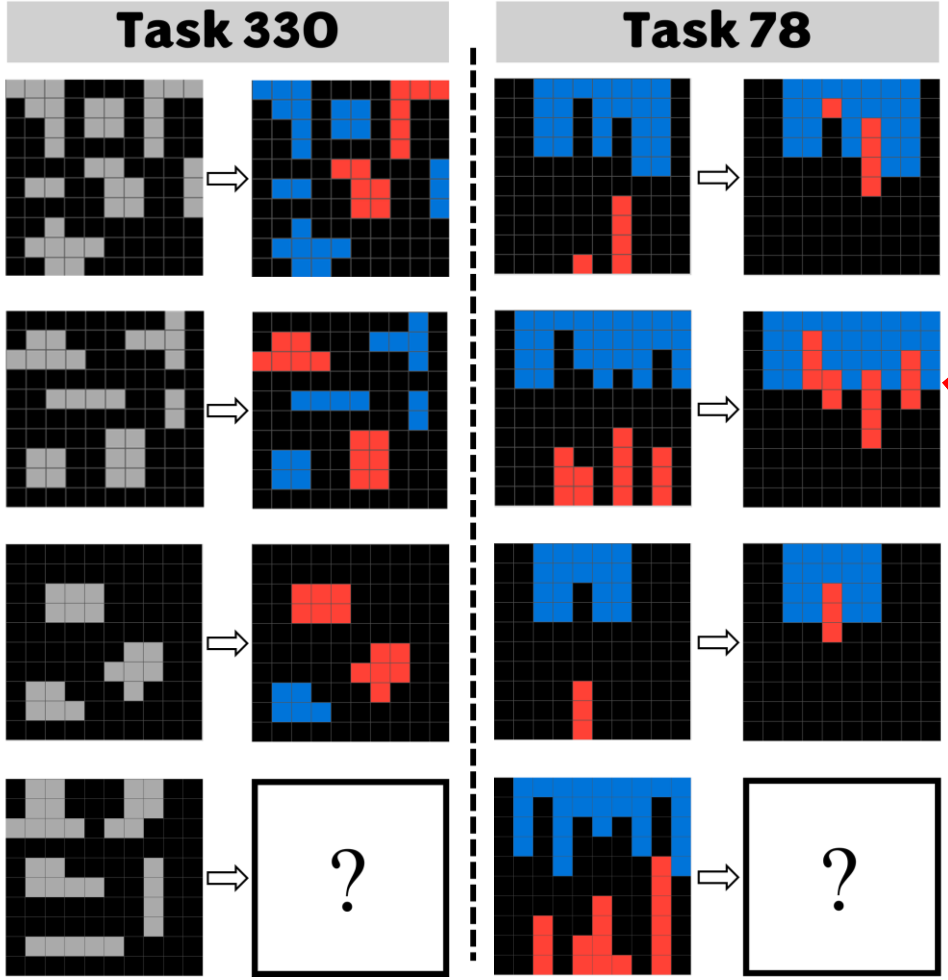
Node:

Color = Grey
Size = 6
Contained Pixels=
(6,5), (6,6), (7,5),
(7,6), (8,5), (8,6)

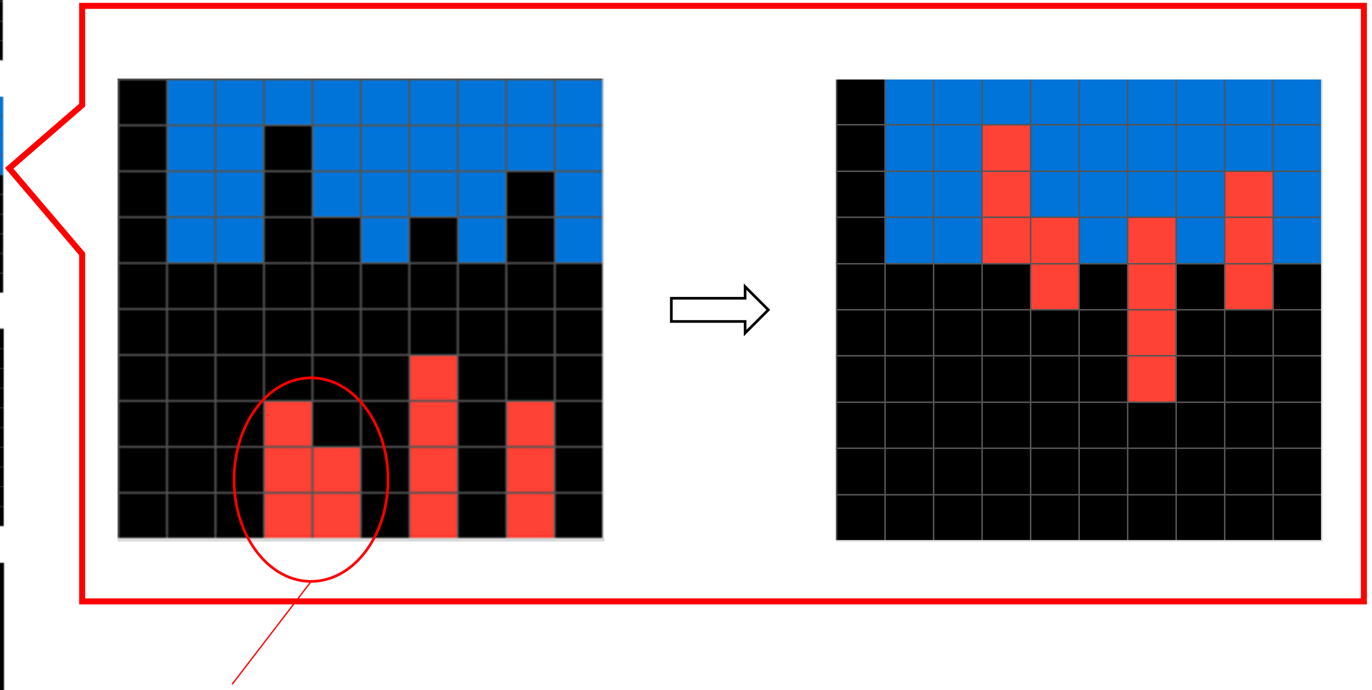
Single-colored connected
pixels graph

UNIVERSITY OF TORONTO

Recognize Objects: Graph Abstractions



Does the same abstraction work for all tasks?

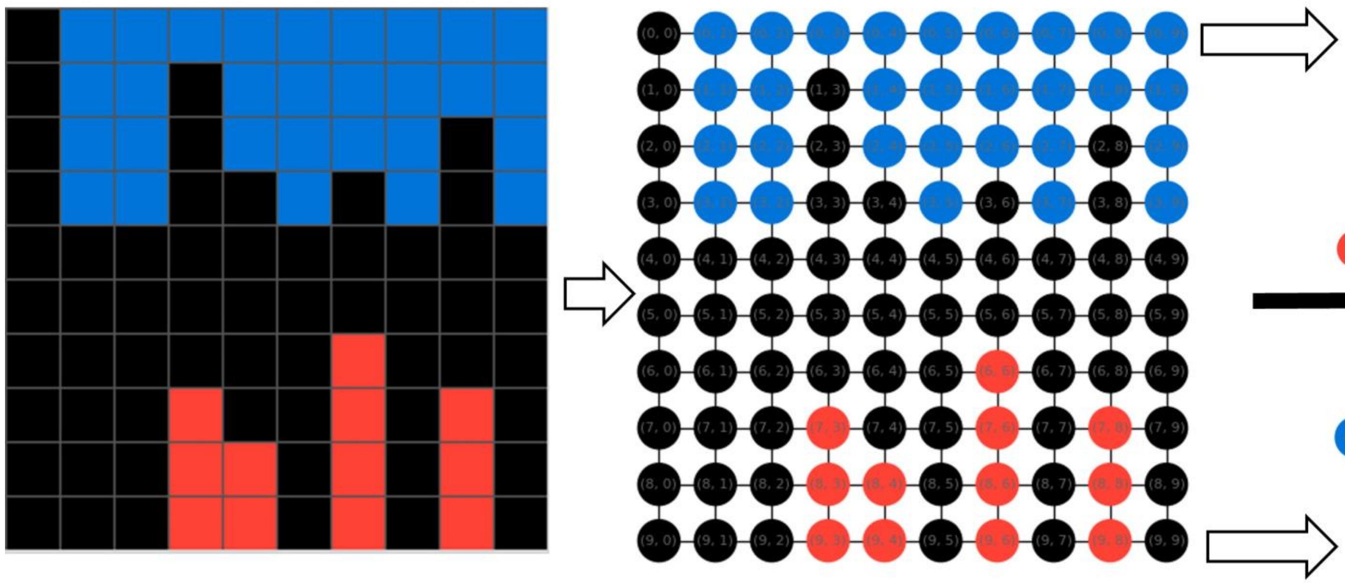


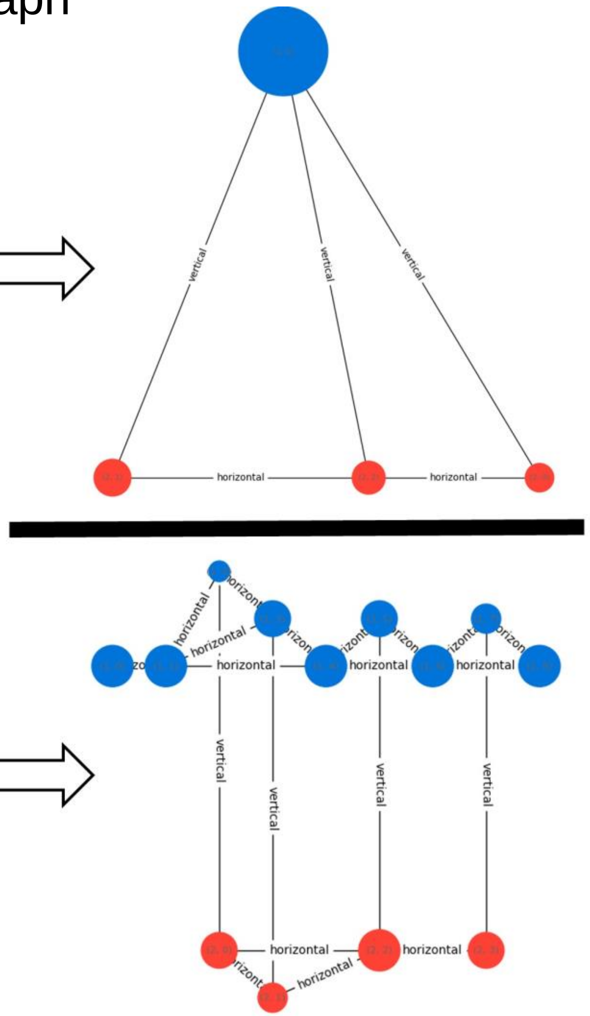
There are two objects here!

Recognize Objects: Graph Abstractions

Need different definitions for objects!
Achieved by defining different abstractions

Single-colored connected pixels graph





Single-colored vertically connected pixels graph

UNIVERSITY OF TORONTO

ARGA Approach

- **Recognize objects** → Graph Abstraction

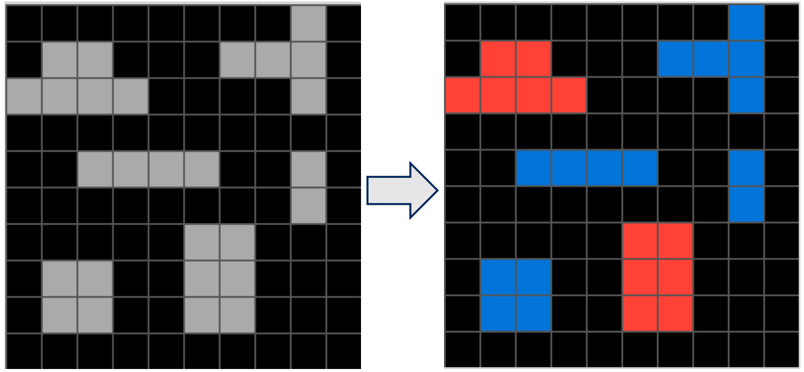
- **Develop an object-centric Domain Specific Language (DSL)**

- **Search through the DSL to find modification to the objects**

Object-centric DSL

UNIVERSITY OF TORONTO

Object-centric DSL



“Color the size 6 grey objects red, color the non-size 6 grey objects blue”

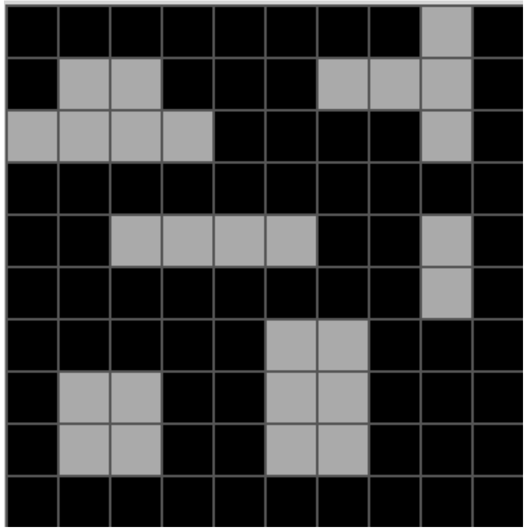
Filters:
Look for objects that satisfy some condition
- *size 6 objects*
- *non-size 6 objects*
- *grey objects*

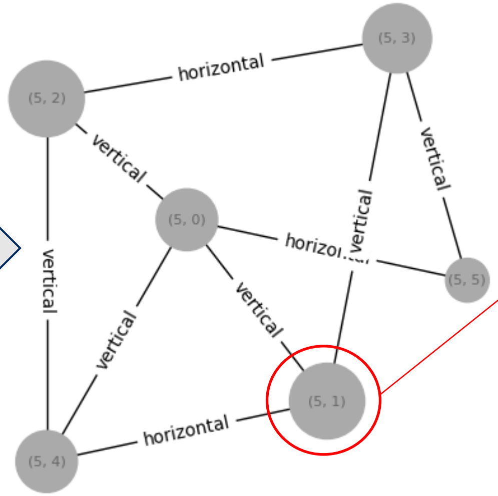
Transformations:
Update object properties
- *color object red*
- *color object blue*

UNIVERSITY OF TORONTO

Object-centric DSL: Filters

How do we filter for size 6 grey objects?





Node:

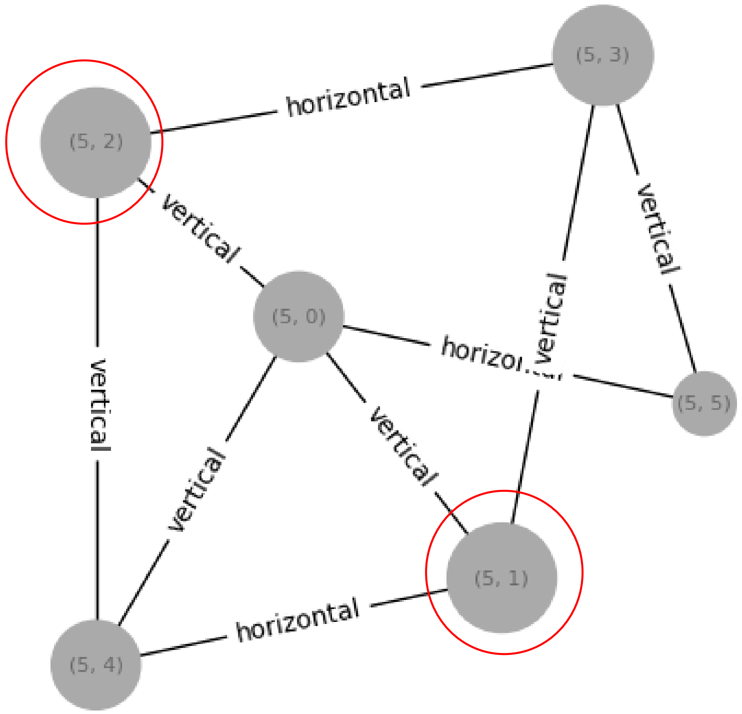
Color = Grey
Size = 6
Contained Pixels=
(6,5), (6,6), (7,5),
(7,6), (8,5), (8,6)

Node *n*
color(n, grey) = True
size(n, 6) = True
containsPixel(n, (6,5)) = True
containsPixel(n, (6,6)) = True
containsPixel(n, (7,5)) = True
containsPixel(n, (7,6)) = True
containsPixel(n, (8,5)) = True
containsPixel(n, (8,6)) = True

logically defined relations

Object-centric DSL: Filters

How do we filter for size 6 grey objects?



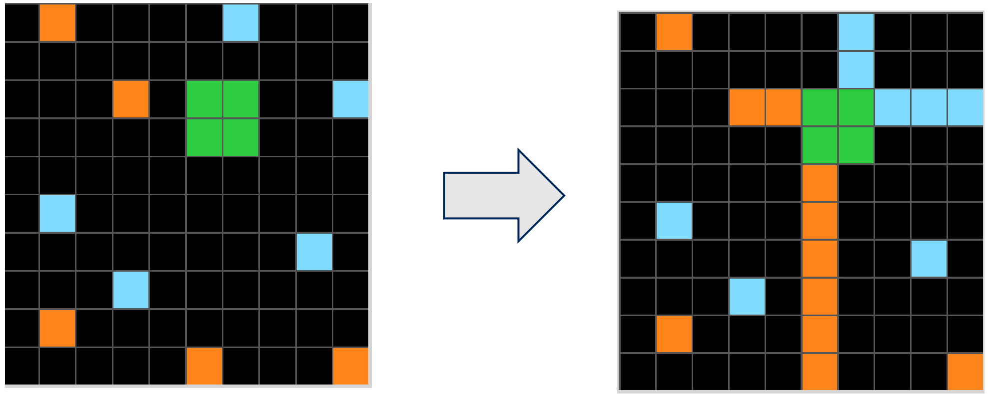
$$Node(n) \wedge Size(n, 6) \wedge Color(n, grey)$$

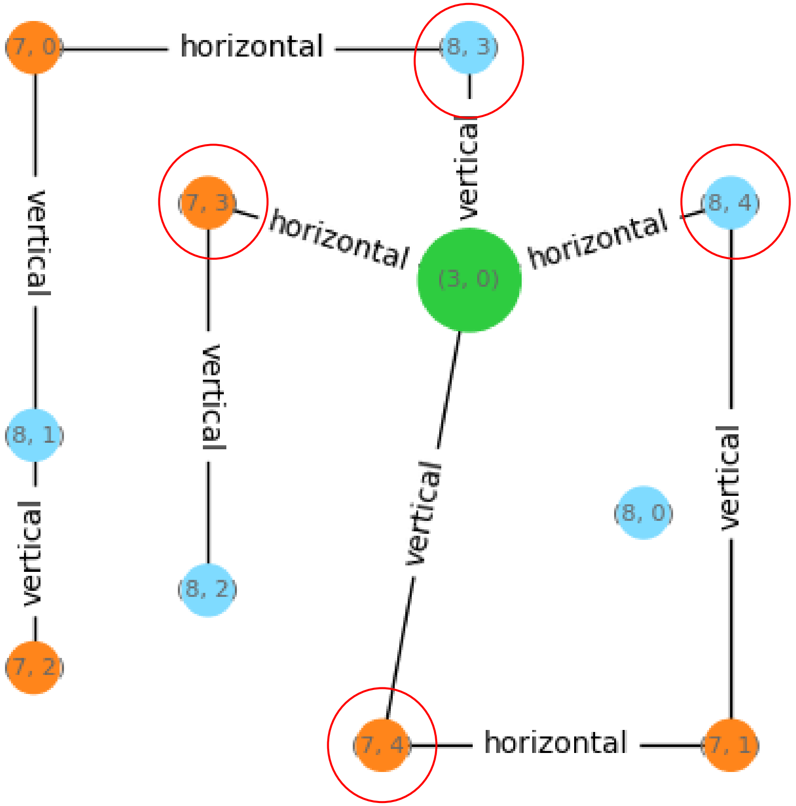
Evaluates to True for objects with size 6 and color grey

Full Filter Grammar:

$$
\begin{aligned}
Filter(x) &::= Type(x) \\
&::= Filter(x) \wedge Filter(x) \\
&::= Filter(x) \vee Filter(x) \\
&::= \neg Filter(x) \\
&::= \exists y\, Rel(x, y) \wedge Filter(y) \\
&::= \exists y\, Rel(y, x) \wedge Filter(y) \\
&::= \forall y\, Rel(x, y) \implies Filter(y) \\
&::= \forall y\, Rel(y, x) \implies Filter(y) \\
&::= Rel(x, c)\ [c \text{ is a constant}] \\
&::= Rel(c, x)\ [c \text{ is a constant}]
\end{aligned}
$$

Object-centric DSL: Filters





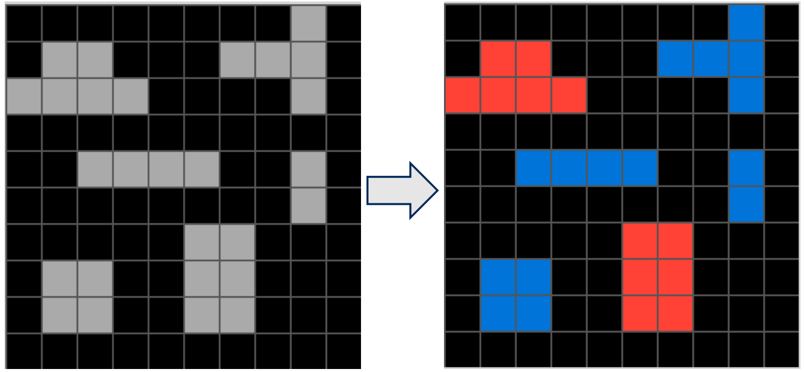
$$\exists y \;\; Neighbor(n, y)$$
$$\wedge Color(y, green)$$

Evaluates to True for objects with green neighbor

Full Filter Grammar:

$$
\begin{aligned}
Filter(x) &::= Type(x) \\
&::= Filter(x) \wedge Filter(x) \\
&::= Filter(x) \vee Filter(x) \\
&::= \neg Filter(x) \\
&::= \exists y \, Rel(x, y) \wedge Filter(y) \\
&::= \exists y \, Rel(y, x) \wedge Filter(y) \\
&::= \forall y \, Rel(x, y) \implies Filter(y) \\
&::= \forall y \, Rel(y, x) \implies Filter(y) \\
&::= Rel(x, c) \; [c \text{ is a constant}] \\
&::= Rel(c, x) \; [c \text{ is a constant}]
\end{aligned}
$$

Object-centric DSL



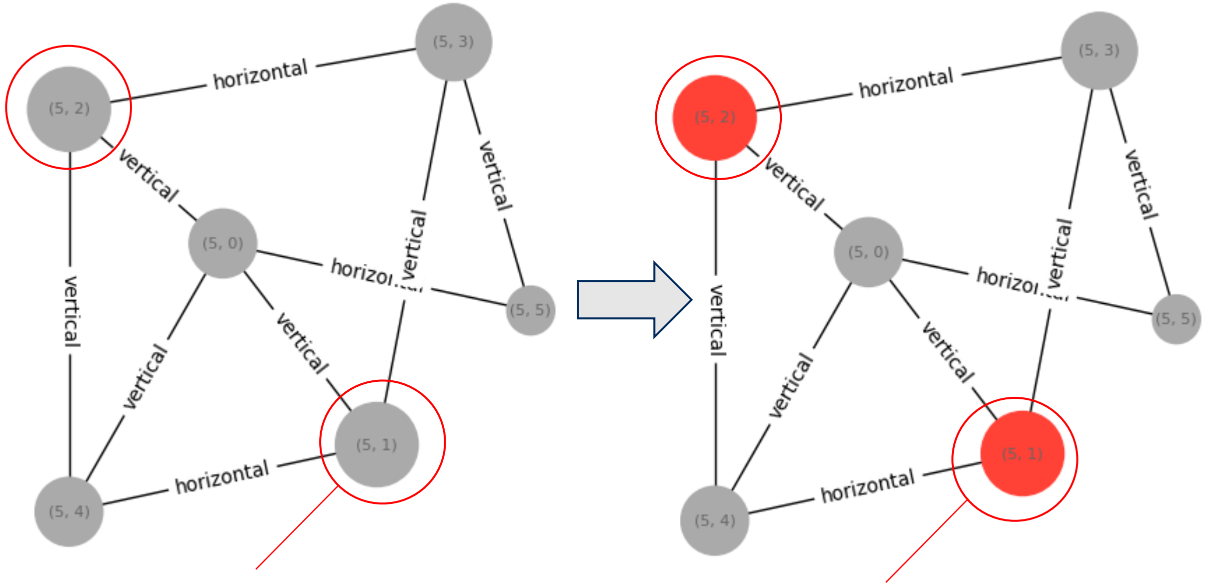
“Color the size 6 grey objects red, color the non-size 6 grey objects blue”

Filters:
Look for objects that satisfy some condition
- *size 6 objects*
- *non-size 6 objects*
- *grey objects*

Transformations:
Update object properties
- *color object red*
- *color object blue*

Object-centric DSL: Transformation

We have filtered for the objects to be colored. How do we update them?



Node *n*
color(n, grey) = True
color(n, red) = False
size(n, 6) = True
containsPixel(n, (6,5)) = True
…

Node *n*
color(n, grey) = False
color(n, red) = True
size(n, 6) = True
containsPixel(n, (6,5)) = True
…

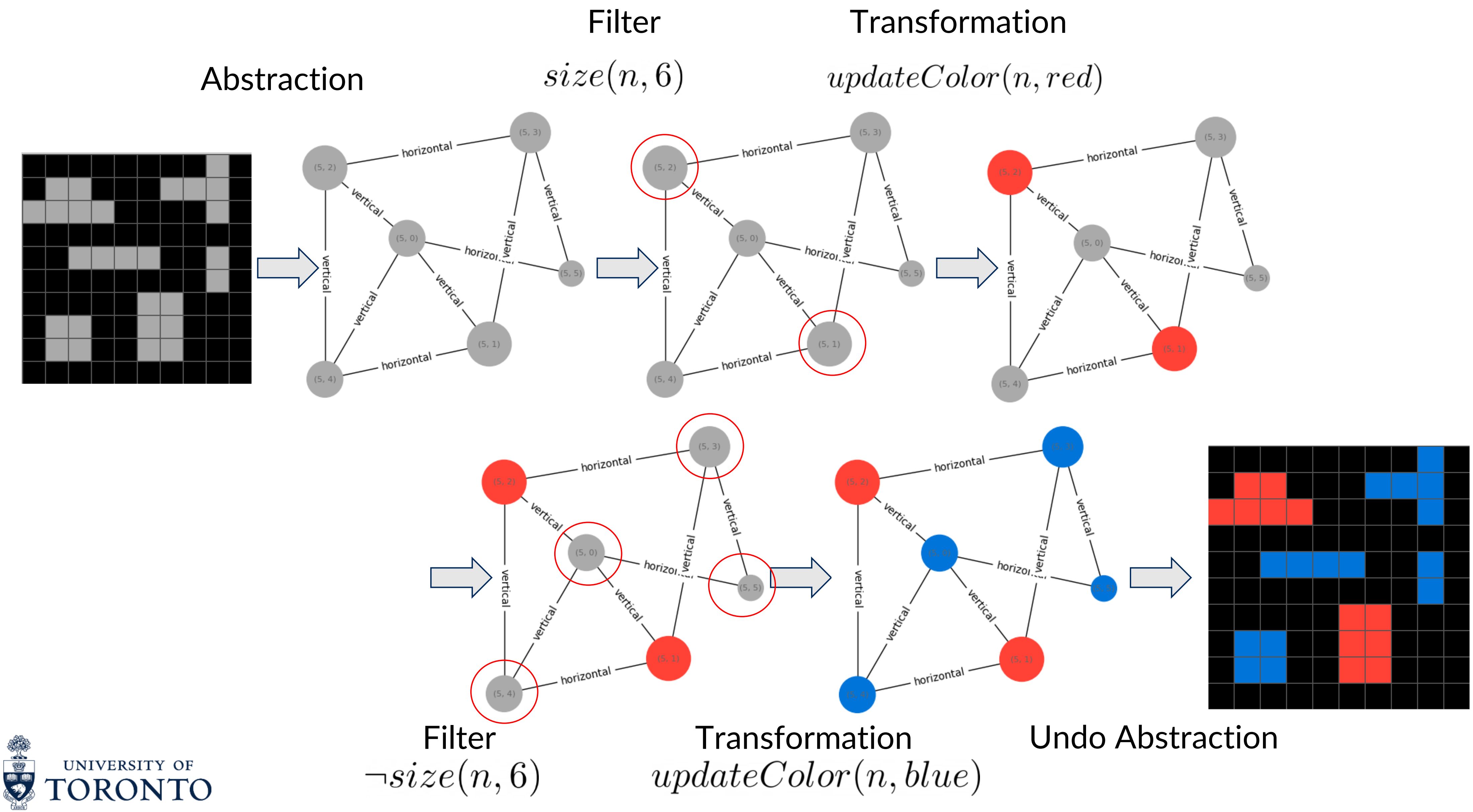
Define transformation:

$$updateColor(n : Node, c : Color) \longrightarrow color(n, c) \wedge \neg color(n, c') \quad \forall c' \in Color \text{ s.t. } c' \neq c$$

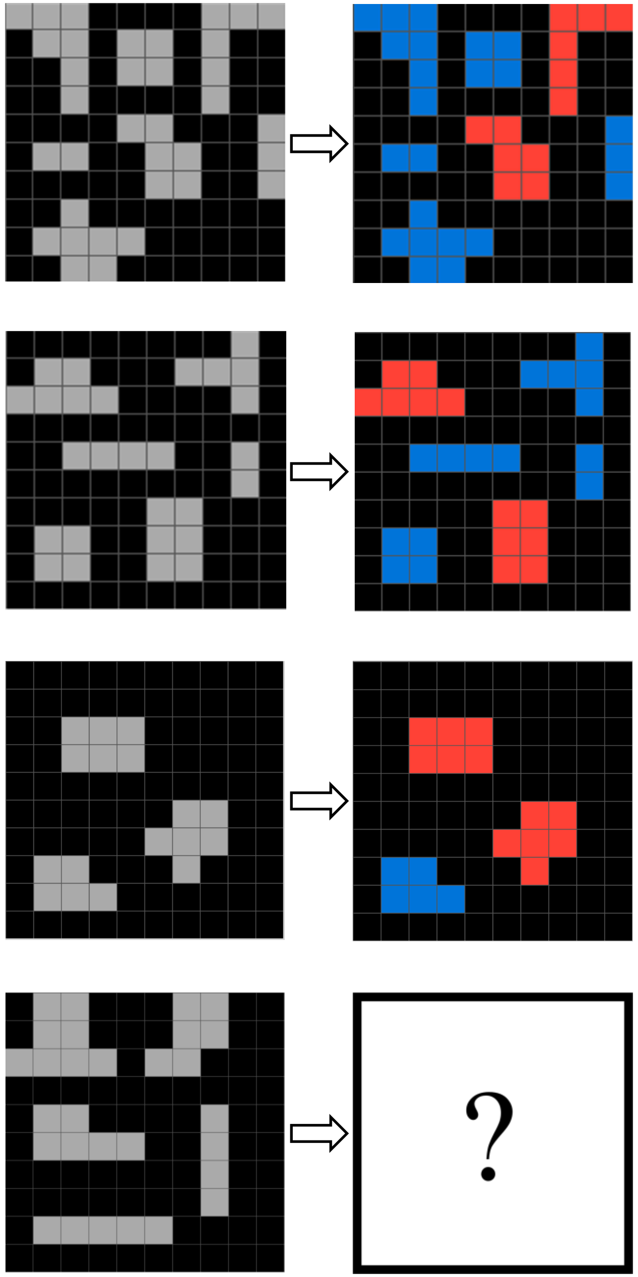
Object-centric DSL: Transformation

Transformation	Description
updateColor(Node, Color)	Update color of Node to Color
move(Node, Direction)	Update pixels of Node to move in Direction
moveMax(Node, Direction)	Update pixels of Node to move in Direction until it collides with another node
rotate(Node)	Update pixels of Node to rotate it clockwise
fillRectangle(Node, Color)	Fill background nodes in rectangle enclosed by the node with Color
hollowRectangle(Node, Color)	Color all nodes in rectangle enclosed by the node with Color
addBorder(Node, Color)	Add additional pixels to Node in Direction
insertPattern(Node, Pattern)	Insert Pattern at Node
mirror(Node, Pixel, Direction)	Mirror Node toward Direction around Pixel
extend(Node, Direction)	Add additional pixels to Node in Direction
flip(Node, Direction)	Flip Node in place in some direction

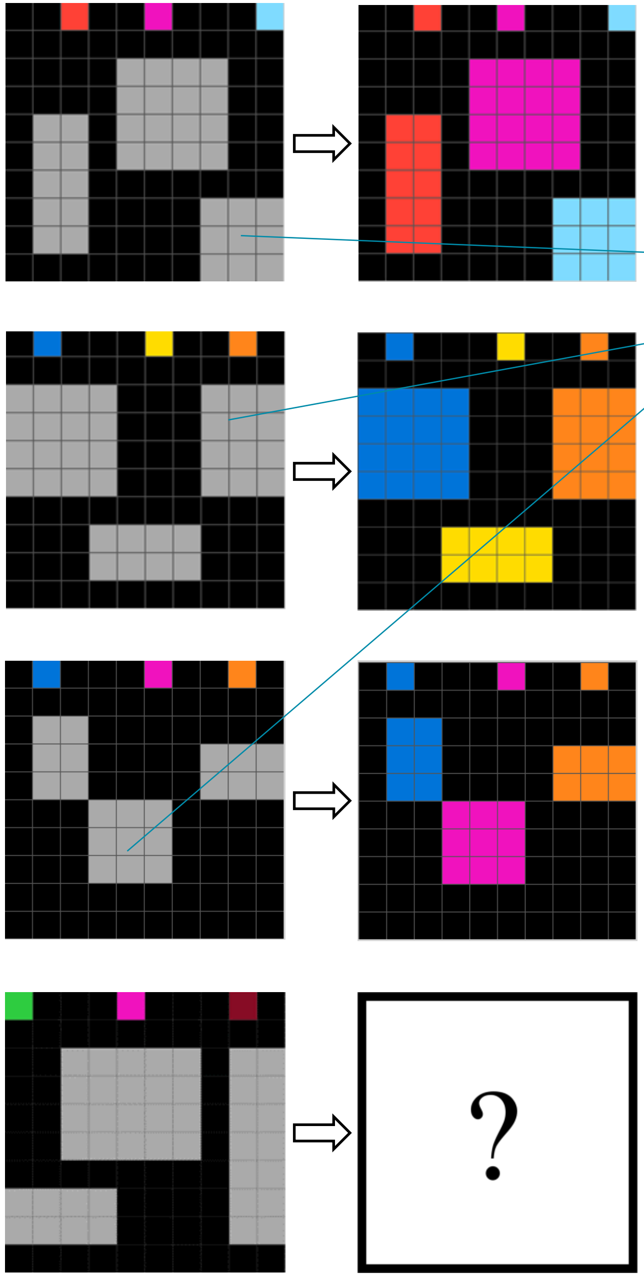
ARGA Solution



Task 330



Task 354



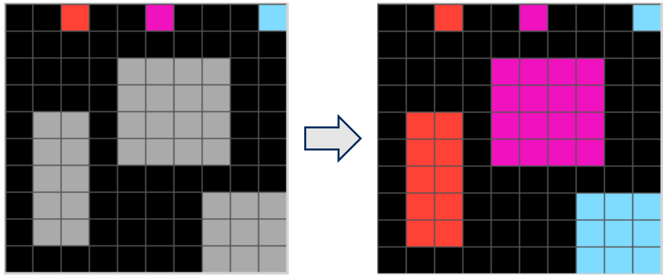
What color should these objects be updated to?

⇩

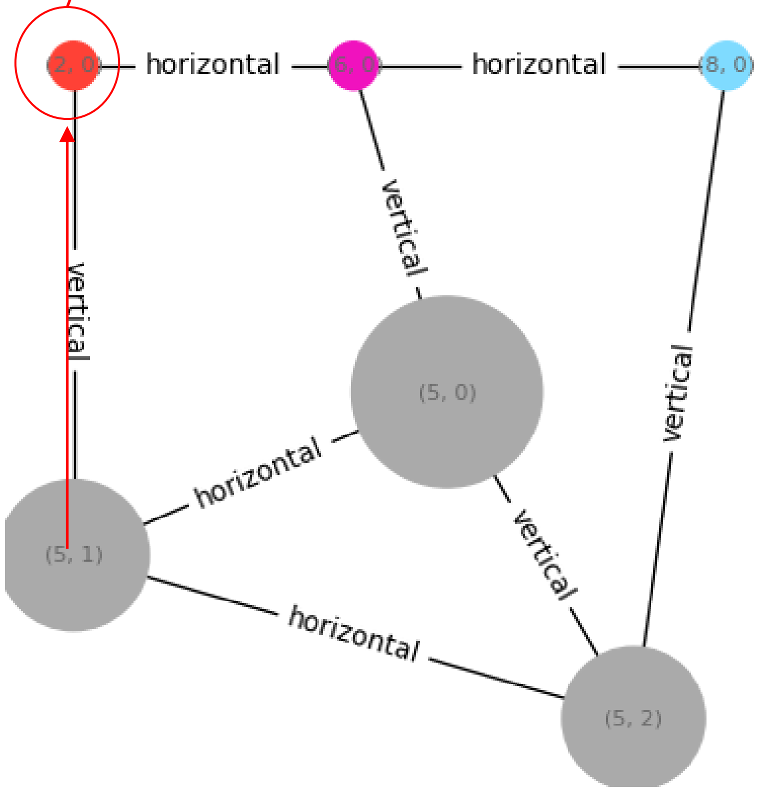
The color of their size-1 neighbor

The color needs to be dynamically determined for each instance!

Object-centric DSL: Parameter Binding



Color *c* = *red*



$$\exists y \;\; neighbor(n, y) \wedge size(y, 1) \wedge color(y, c)$$

retrieve this:

color of size 1 neighbor of node n

Grammar:

$Param(x, v)$
$::= v = c$ [c is a constant]
$::= Rel(x, v)$
$::= Rel(v, x)$
$::= \exists y \; Rel(x, y) \wedge Filter(y) \wedge Param(y, v)$
$::= \exists y \; Rel(y, x) \wedge Filter(y) \wedge Param(y, v)$

UNIVERSITY OF TORONTO

ARGA Solution

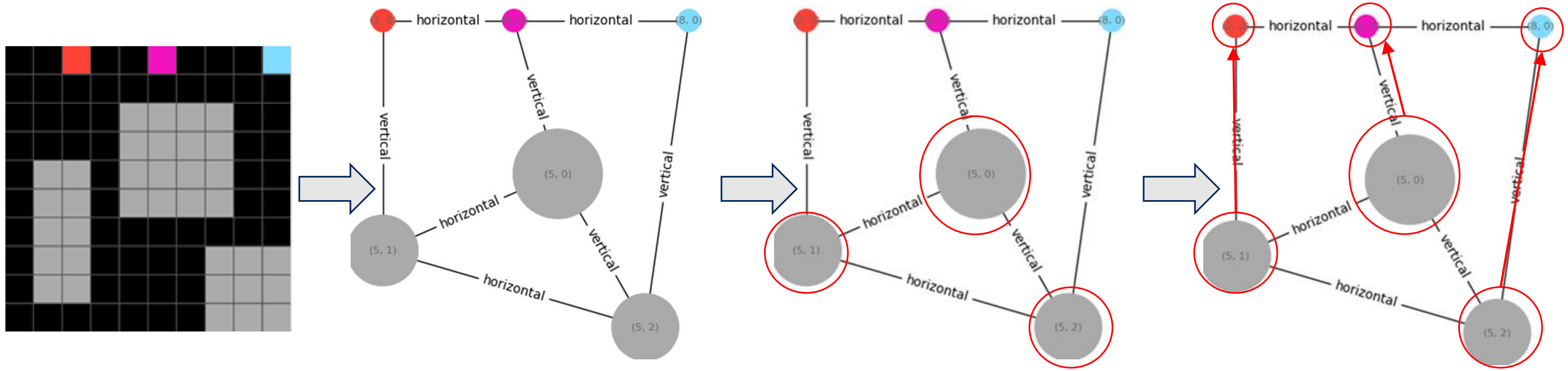
Abstraction

Filter

$color(n, grey)$

Parameter Binding

$$\exists y \;\; neighbor(x, y)$$
$$\wedge size(y, 1)$$
$$\wedge color(y, c)$$

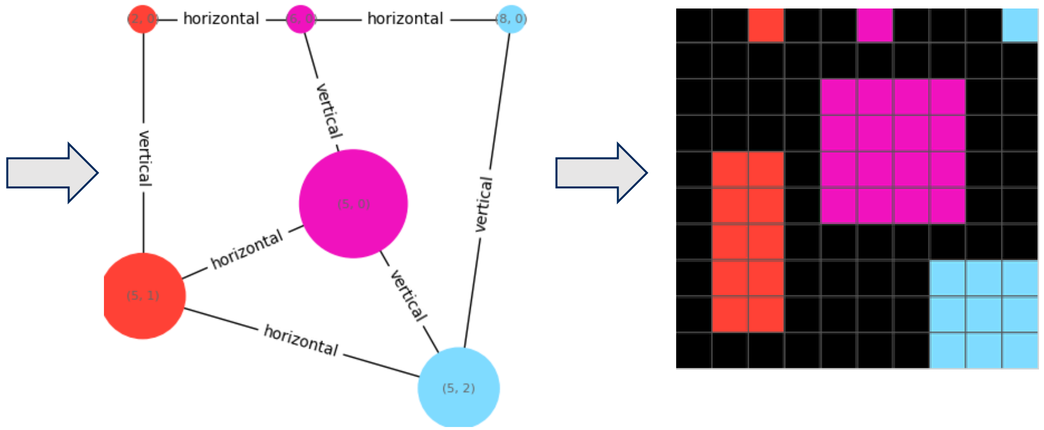


for each $\quad n \in Node$

if $\quad filter(n)$

then $\quad v_i \leftarrow \{v | Param_i(n, v)\}$ for $i \in \{1 \ldots k\}$

$\quad transform(n, v_1, \ldots, v_k)$



Transformation

$updateColor(n, c)$

Undo Abstraction

ARGA Approach

- **Recognize objects** → Abstraction

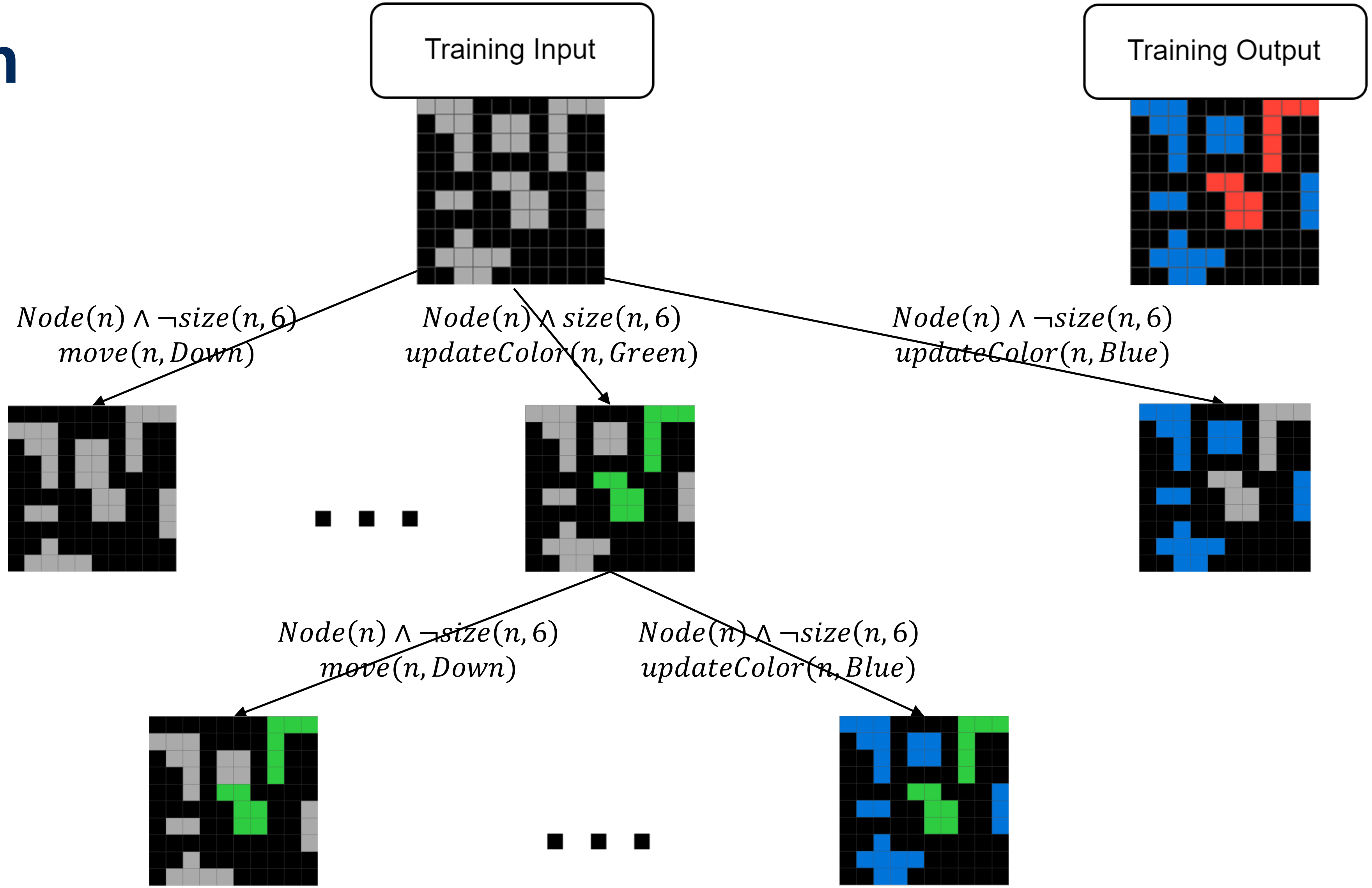
- **Develop an object-centric Domain Specific Language (DSL)** → Filters, Transformations, Parameter Binding

- **Search through the DSL to find modification to the objects**

Solution Synthesis:
Searching the DSL for solution

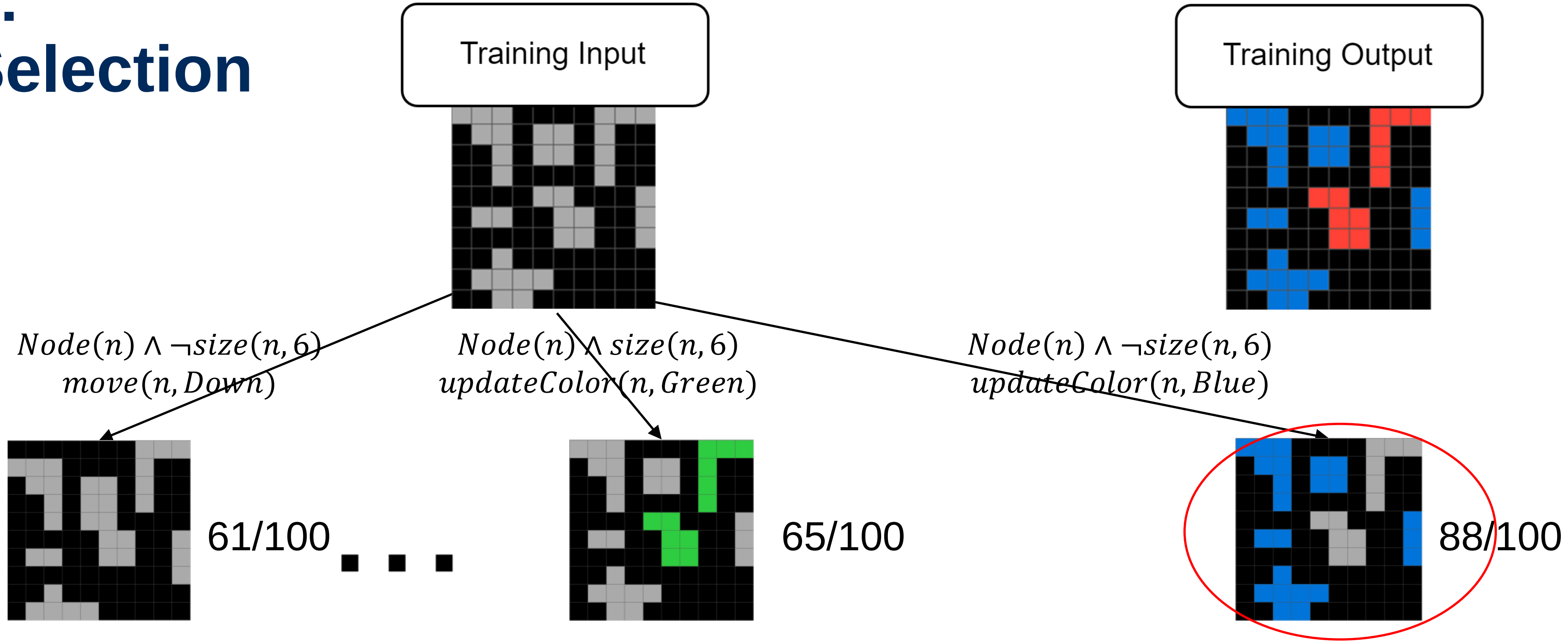
UNIVERSITY OF
TORONTO

Search



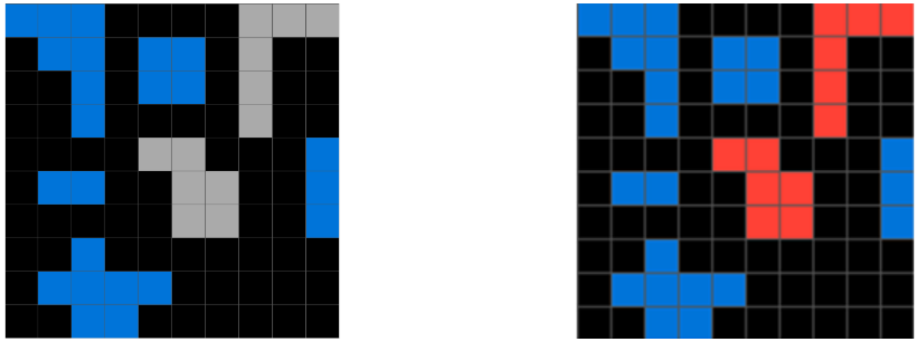
UNIVERSITY OF TORONTO

Search: Node Selection



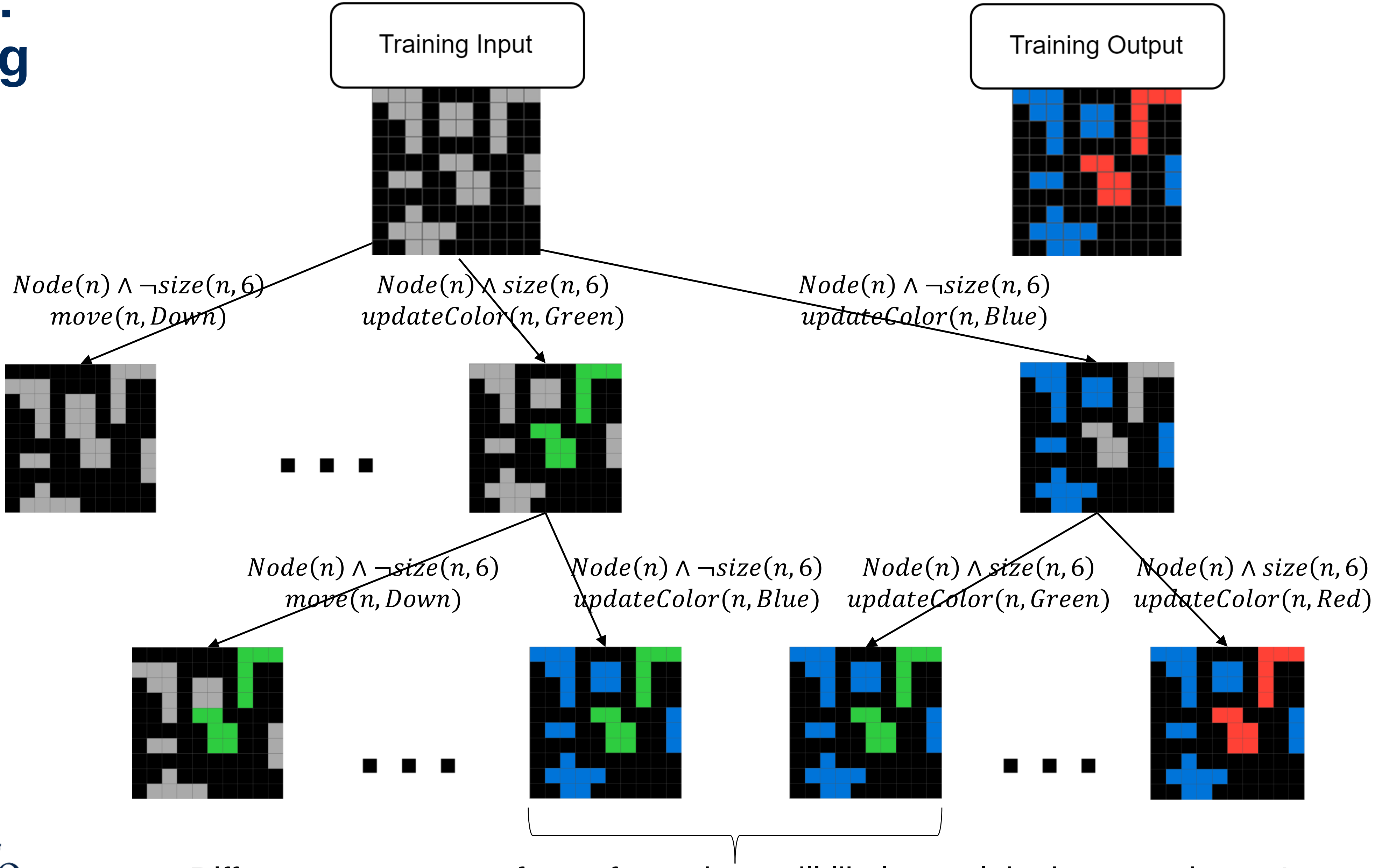
Which node to expand next?

Heuristics:
Select the node with the highest pixel-wise Accuracy



88/100 pixels accurate

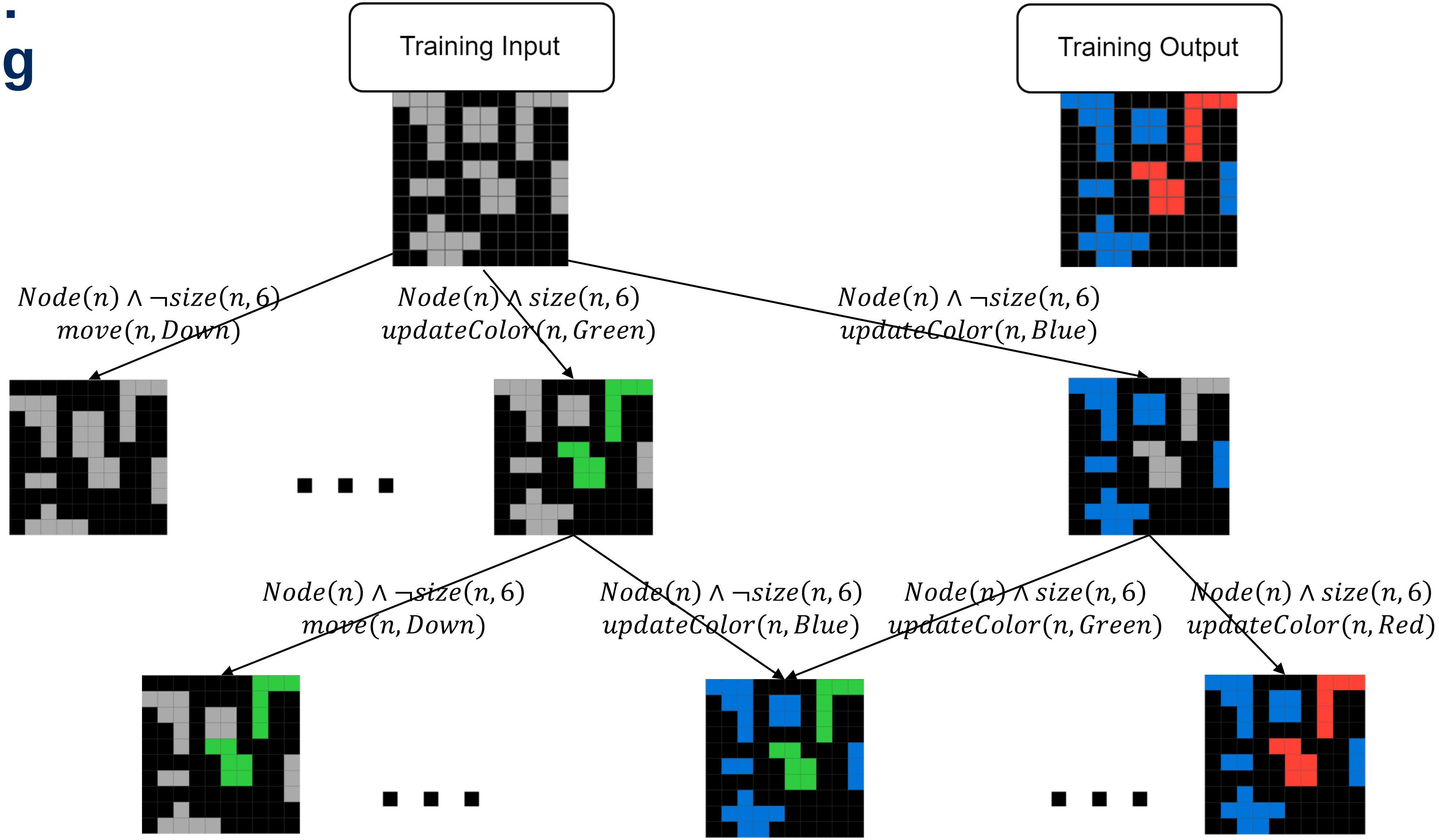
Search: Hashing



Different sequences of transformations will likely result in the same image!

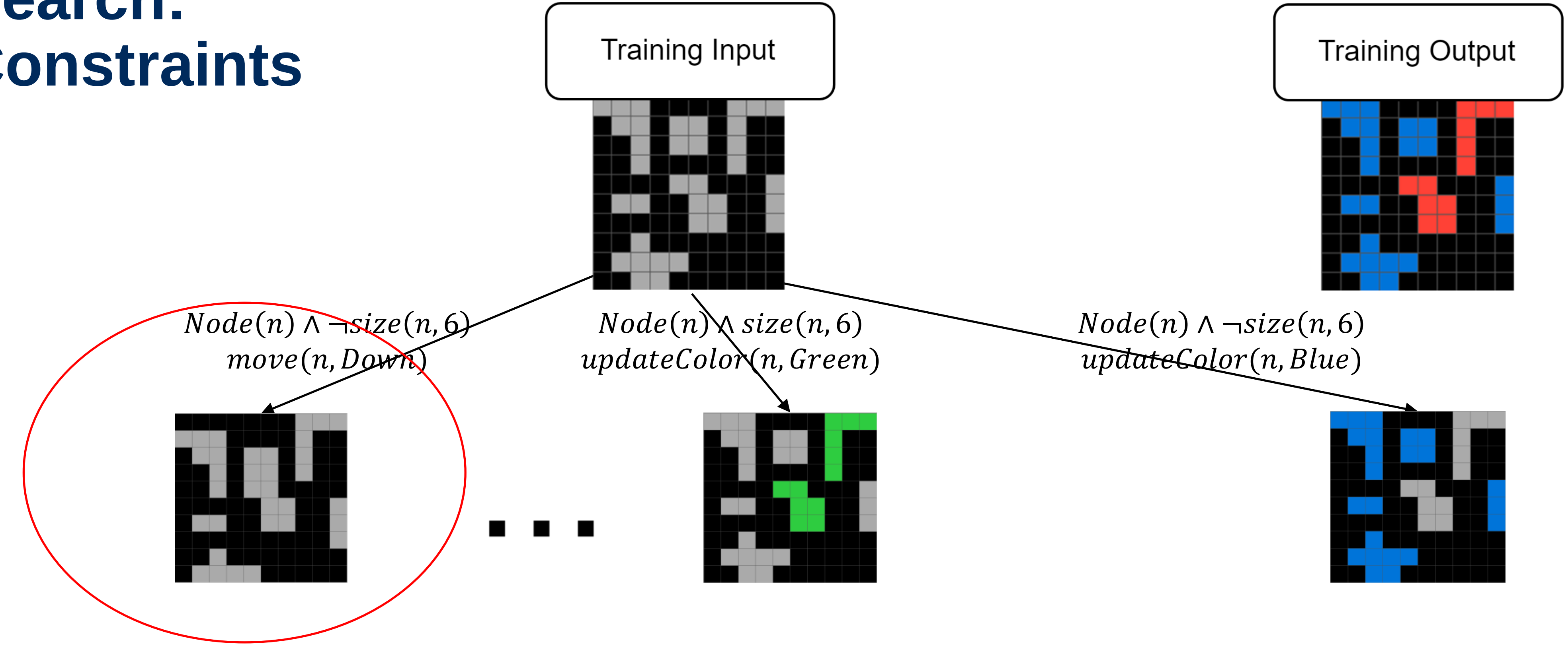
UNIVERSITY OF TORONTO

Search: Hashing

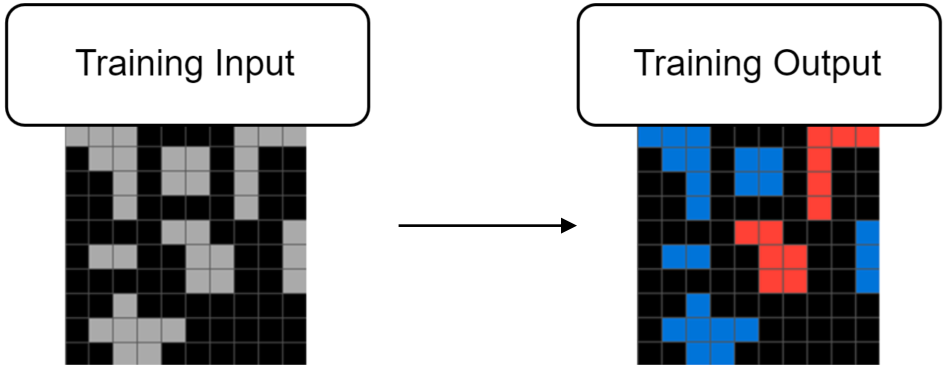


UNIVERSITY OF TORONTO

Search: Constraints



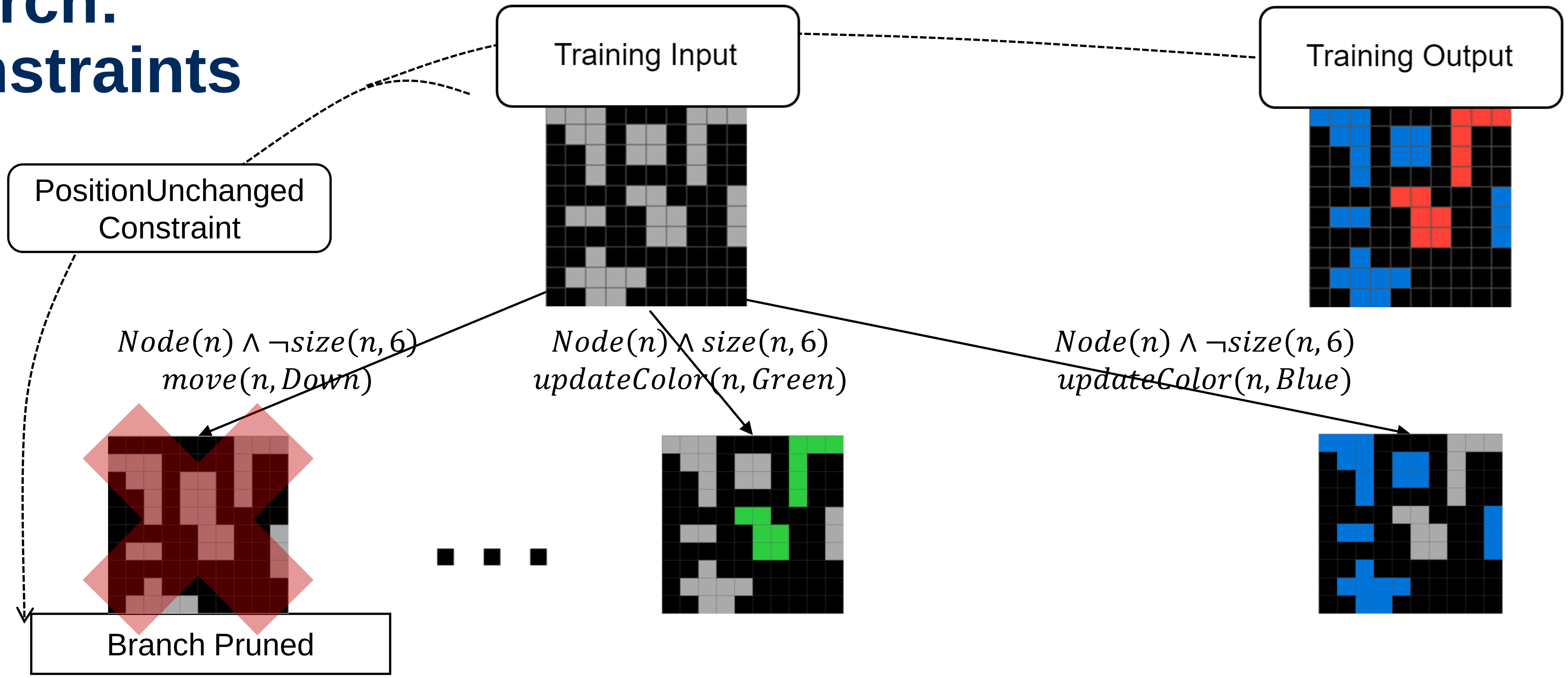
Should this branch be explored at all?



Clearly, none of the objects should move!

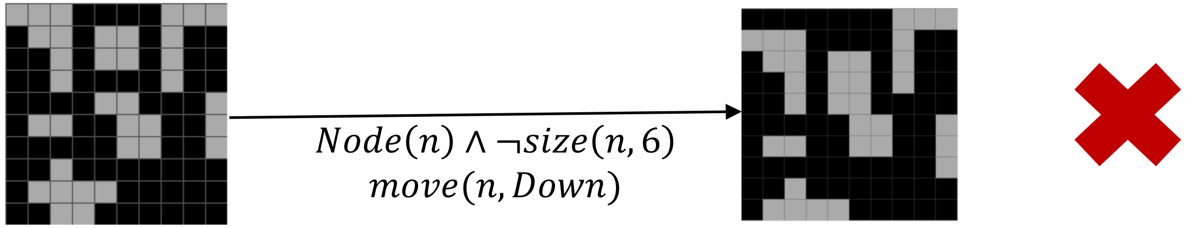
UNIVERSITY OF TORONTO

Search: Constraints

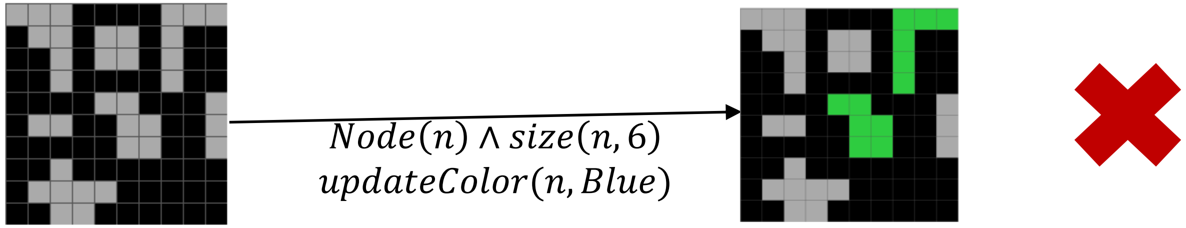


Constraints

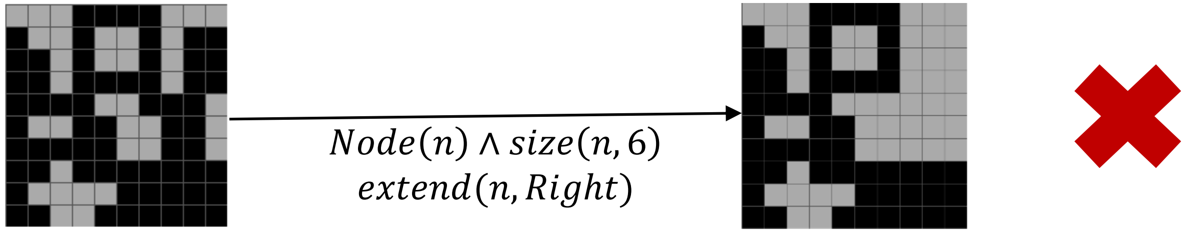
positionUnchanged:
Node does not change position after update



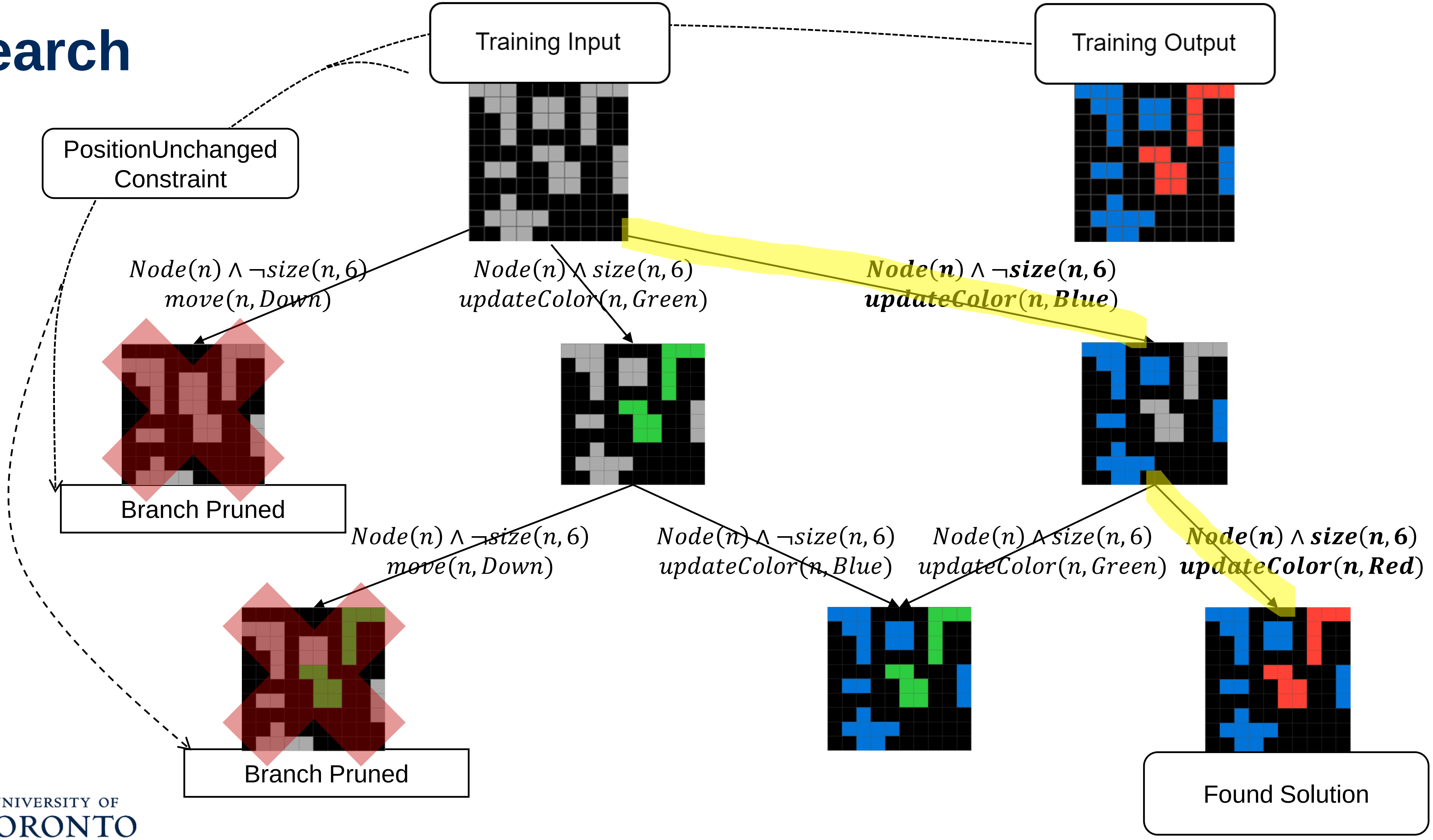
colorUnchanged:
Node does not change color after update



sizeUnchanged:
Node does not change in size after update



Search



UNIVERSITY OF TORONTO

ARGA Approach

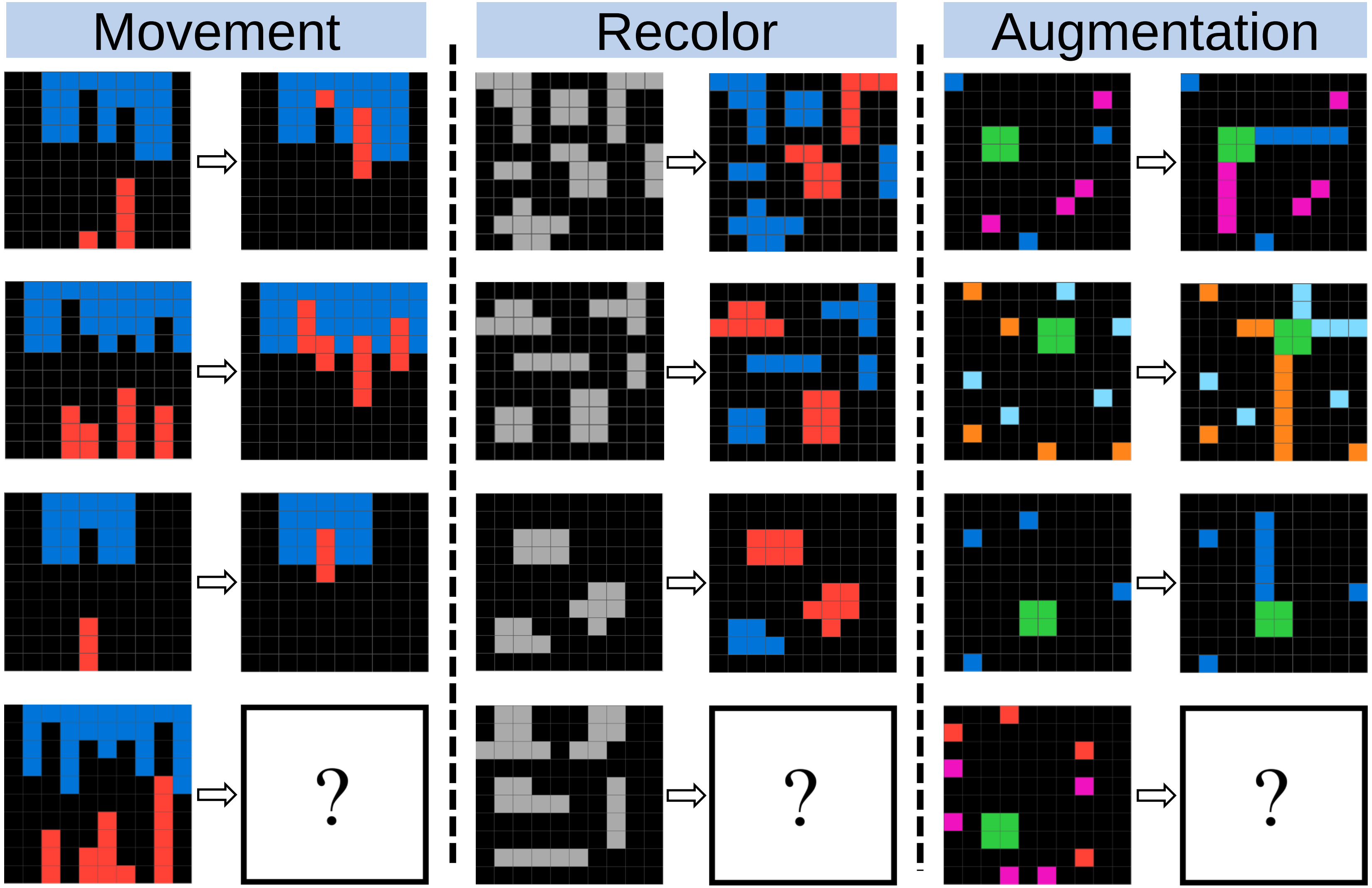
- **Recognize objects** → Abstraction
- **Develop an object-centric Domain Specific Language (DSL)** → Filters, Transformations, Parameter Binding
- **Search through the DSL to find modification to the objects** → Greedy Search, Constraints, Hashing

ARGA Results

UNIVERSITY OF TORONTO

Results

Identified a subset of 160 object-centric tasks



Results

Model	Task Type	# Training Correct	# Testing Correct	Average Nodes	Average Time (sec.)
ARGA	movement	18/31 (58.06%)	17/31 (54.84%)	3830.35	89.75
	recolor	25/62 (40.32%)	23/62 (37.10%)	12316.87	326.83
	augmentation	20/67 (29.85%)	17/67 (25.37%)	4668.82	67.09
	all	63/160 (39.38%)	57/160 (35.62%)	7504.81	178.66
Kaggle First Place	movement	21/31 (67.74%)	15/31 (48.39%)	2176777.67	62.45
	recolor	23/62 (37.10%)	28/62 (45.16%)	2290441.32	93.19
	augmentation	35/67 (52.24%)	21/67 (31.34%)	2248151.10	66.07
	all	79/160 (49.38%)	64/160 (40.00%)	2249924.92	77.08

Conclusion

- Abstraction and Reasoning Corpus (ARC)
  - Hard to solve
  - State of the art solution does not utilize objects
- ARGA
  - Object-centric DSL
  - Graph abstraction to recognize objects
  - Constraint-guided search to find solution