Specifying Goals to Deep Neural Networks with Answer Set Programming

Forest Agostinelli, Rojina Panta, Vedant Khandelwal
University of South Carolina

Rojina Panta  Vedant Khandelwal
• Overview
• Heuristic function training
• Goal specification and reaching
• Results
• Future work
Motivation

• Deep reinforcement learning methods, such as DeepCubeA, can learn domain-specific heuristic functions in a largely domain-independent fashion

• Limitations
  • The goal is pre-determined
    • Specifying a new goal requires re-training the entire DNN
  • Hindsight experience replay can be used to generalize over start states and goal states
    • Must know the exact goal state, which is not always feasible
    • Cannot define a set of goal states using a high-level specification language

• Desired solution
  • High-level specifications
    • It should be possible to specify a goal (a set of states), without knowing the elements in the set
    • This will allow us to discover new states by finding a path to a currently unknown state that meets a given specification
  • Flexible specification language
    • The specification language should be able to represent diverse goals
  • Goal agnostic training
    • The training process should not have to be given any information about the goals it will see during testing
    • No re-training necessary

• Can be applied to specifying goals in applications such as chemical synthesis, quantum circuit design, manufacturing

• In our work
  • State descriptors: assignments of values to variables
  • Specification language: Answer set programming (ASP)
  • ASP will be used to describe goals at a high-level using formal logic and an answer set solver will be used to find assignments that represent the a subset of the goal
Outline

• Overview
• Heuristic function training
• Goal specification and reaching
• Results
• Future work
State Representation

• In a given pathfinding domain, there are $V$ variables
  • A variable, $x_i$, can be assigned a single value from its (variable) domain, $D(x_i)$
• An assignment is an assignment is a set of assignments of values to variables $\{x_i = v_i\}$
  • All $v_i \in D(v_i)$
  • If $x_i$ is not in the assignment then it is unassigned
• An assignment is a complete assignment iff all variables have been assigned values
• A state is a complete assignment
• For example, for the Rubik’s cube, variables are stickers and values are their colors
Goal Representation

- An assignment is a **partial assignment** iff at least one variable has not been assigned a value.
- A **goal** is a complete or partial assignment.
- An assignment, \( A \), represents a set of states, \( S_A \).
  - A complete assignment always represents a set of states of size 1.
- A state, \( s \), is in \( S_A \) iff \( A \subseteq s \).
  - In other words, all assignments in \( A \) are present in \( s \).
  - An empty assignment represents the set of all possible states.
- For example, a visualization of an assignment for the “white cross” pattern for the Rubik’s cube and a state that is in the set of states represented by this assignment.
Training

- Generate a start state
- Take a random walk whose length is somewhere between 0 and T
  - Future work could use artificial curiosity
- Convert the end state to its representation as an assignment
- Subsample to obtain a goal
- Convert this representation into one suitable for the DNN
  - One-hot representation
  - Graph
  - Etc.
- RL Update
  - \( L(\theta) = \left( \min_a (c^a(s) + h_\theta(T(s,a),G)) - h_\theta(s,G) \right)^2 \)
Experiments

• ASP will be used to find assignments; therefore, we compare our method, DeepCubeA_g, to other methods capable of finding paths to goals that can be represented as assignments

• 500-1,000 test start and goal pairs

• 200 second time limit to solve test states

• DeepCubeA_g
  • Batch A* search

• DeepCubeA
  • Predefined goal
  • Batch A* search

• Fast Downward Planner
  • A* search
  • Goal count heuristic, fast forward heuristic, causal graph heuristic

• PDBs
  • Predefined goal
  • IDA* search

Performance

- Canon: Canonical goal states
- Rand: Random assignment selected as goal
  - Can be as small as the empty assignment
  - Methods that require a pre-defined goal cannot be applied to this scenario without considerable overhead
- PDBs+: Also includes group theory knowledge
- DeepCubeA\(_g\) consistently outperforms fastdownward in terms of percentage of states solved

<table>
<thead>
<tr>
<th>Puzzle</th>
<th>Solver</th>
<th>Path Cost</th>
<th>% Solved</th>
<th>% Opt</th>
<th>Nodes/Sec</th>
<th>Secs</th>
<th>Nodes/Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC (Canon)</td>
<td>PDBs(^+)</td>
<td>20.67</td>
<td>100.00%</td>
<td>100.0%</td>
<td>2.05E+06</td>
<td>2.20</td>
<td>1.79E+06</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA</td>
<td>21.50</td>
<td>100.00%</td>
<td>60.3%</td>
<td>6.62E+06</td>
<td>24.22</td>
<td>2.90E+05</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA(_g)</td>
<td>22.03</td>
<td>100.00%</td>
<td>35.0%</td>
<td>2.44E+06</td>
<td>41.99</td>
<td>5.67E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>-</td>
<td>0.00%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>-</td>
<td>0.00%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>-</td>
<td>0.00%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RC (Rand)</td>
<td>DeepCubeA(_g)</td>
<td>15.22</td>
<td>99.40%</td>
<td>-</td>
<td>1.91E+06</td>
<td>32.24</td>
<td>5.19E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>7.18</td>
<td>32.80%</td>
<td>-</td>
<td>2.67E+05</td>
<td>13.79</td>
<td>1.41E+05</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>6.49</td>
<td>31.20%</td>
<td>-</td>
<td>4.87E+05</td>
<td>13.83</td>
<td>2.93E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>7.85</td>
<td>33.80%</td>
<td>-</td>
<td>1.12E+06</td>
<td>11.62</td>
<td>5.81E+04</td>
</tr>
<tr>
<td>PDBs</td>
<td>PDBs(^+)</td>
<td>52.02</td>
<td>100.00%</td>
<td>100.0%</td>
<td>3.22E+04</td>
<td>0.002</td>
<td>1.45E+07</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA</td>
<td>52.03</td>
<td>100.00%</td>
<td>99.4%</td>
<td>3.85E+06</td>
<td>10.28</td>
<td>3.93E+05</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA(_g)</td>
<td>52.02</td>
<td>100.00%</td>
<td>100.0%</td>
<td>1.81E+05</td>
<td>2.61</td>
<td>6.94E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>36.75</td>
<td>0.80%</td>
<td>0.80%</td>
<td>9.05E+07</td>
<td>102.11</td>
<td>8.66E+05</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>52.75</td>
<td>80.80%</td>
<td>24.80%</td>
<td>2.92E+06</td>
<td>42.11</td>
<td>6.93E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>41.95</td>
<td>4.40%</td>
<td>1.20%</td>
<td>2.00E+07</td>
<td>80.58</td>
<td>2.47E+05</td>
</tr>
<tr>
<td>15-P (Canon)</td>
<td>FastDown (GC)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15-P (Rand)</td>
<td>DeepCubeA(_g)</td>
<td>33.98</td>
<td>100.00%</td>
<td>-</td>
<td>1.11E+05</td>
<td>1.60</td>
<td>6.16E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>14.92</td>
<td>38.00%</td>
<td>-</td>
<td>1.61E+07</td>
<td>18.77</td>
<td>5.46E+05</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>32.66</td>
<td>89.20%</td>
<td>-</td>
<td>1.24E+06</td>
<td>17.39</td>
<td>5.63E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>20.45</td>
<td>51.20%</td>
<td>-</td>
<td>3.90E+06</td>
<td>21.41</td>
<td>1.20E+05</td>
</tr>
<tr>
<td>PDBs</td>
<td>PDBs(^+)</td>
<td>89.41</td>
<td>100.00%</td>
<td>100.00%</td>
<td>8.19E+10</td>
<td>4239.54</td>
<td>1.91E+07</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA</td>
<td>89.49</td>
<td>100.00%</td>
<td>96.98%</td>
<td>6.44E+06</td>
<td>19.33</td>
<td>3.34E+05</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA(_g)</td>
<td>90.47</td>
<td>100.00%</td>
<td>55.24%</td>
<td>3.38E+05</td>
<td>5.22</td>
<td>6.48E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>-</td>
<td>0.00%</td>
<td>0.00%</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>81.00</td>
<td>1.01%</td>
<td>0.40%</td>
<td>2.68E+06</td>
<td>89.84</td>
<td>2.91E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>-</td>
<td>0.00%</td>
<td>0.00%</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24-P (Canon)</td>
<td>DeepCubeA(_g)</td>
<td>66.28</td>
<td>99.60%</td>
<td>-</td>
<td>3.10E+05</td>
<td>4.91</td>
<td>6.16E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>9.86</td>
<td>10.00%</td>
<td>-</td>
<td>9.54E+06</td>
<td>11.88</td>
<td>4.27E+05</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>26.35</td>
<td>26.00%</td>
<td>-</td>
<td>5.99E+05</td>
<td>19.57</td>
<td>2.41E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>13.75</td>
<td>12.60%</td>
<td>-</td>
<td>1.42E+06</td>
<td>14.42</td>
<td>6.85E+04</td>
</tr>
<tr>
<td>24-P (Rand)</td>
<td>DeepCubeA</td>
<td>32.88</td>
<td>100.00%</td>
<td>-</td>
<td>5.01E+03</td>
<td>2.71</td>
<td>1.84E+03</td>
</tr>
<tr>
<td></td>
<td>DeepCubeA(_g)</td>
<td>32.02</td>
<td>100.00%</td>
<td>-</td>
<td>1.80E+04</td>
<td>0.95</td>
<td>1.79E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (GC)</td>
<td>31.94</td>
<td>99.80%</td>
<td>-</td>
<td>3.17E+06</td>
<td>5.93</td>
<td>5.85E+05</td>
</tr>
<tr>
<td></td>
<td>FastDown (FF)</td>
<td>33.15</td>
<td>100.00%</td>
<td>-</td>
<td>2.92E+04</td>
<td>0.32</td>
<td>7.49E+04</td>
</tr>
<tr>
<td></td>
<td>FastDown (CG)</td>
<td>33.12</td>
<td>100.00%</td>
<td>-</td>
<td>4.43E+04</td>
<td>0.51</td>
<td>7.25E+04</td>
</tr>
</tbody>
</table>

Sokoban
Outline

• Overview
• Heuristic function training
• Goal specification and reaching
• Results
• Future work
An **answer set program (ASP)** is a set of sentences in first order logic that defines a set of **stable models** (also known as answer sets)
- We obtain assignments from stable models

ASP solvers, such as clingo, can also make use of choice rules, aggregates, and classical negation

- \( \alpha(\Pi) \) is the set of all possible assignments that can be obtained from \( \Pi \)
- A **candidate state** is a state that is a superset of some assignment in \( \alpha(\Pi) \)
- A **goal state** is a state that is in \( \alpha(\Pi) \)

**Monotonic specification:** All candidate states are goal states

**Non-monotonic specification:** Some candidate states are not goal states
ASP Specifications: Rubik’s Cube Example

• Define basic background knowledge
  • Colors, faces, cubelets
  • Constraints: Cannot have two stickers of the same color on the same cubelet, cannot have two stickers from the same cubelet on opposite faces

• Given basic background knowledge, specifications often only require a few lines of code
  • face_same(F) :- face_col(F, FCol), #count{Cbl : onface(Cbl, FCol, F)}=9.
  • canon_solved :- #count{F : face_same(F)}=6.

• Our specifications contain combinations of common patterns
  • Note: the training procedure is unaware of what the specification will be at test time
Goal Reaching: Monotonic Specification

\[ S_{\Pi} = S_{\Pi_{\text{m}}} \]

- If our specification behaves \textit{monotonically}, then all candidate states are goal states
  - Therefore, we can randomly sample assignments from \( \Pi \) until we find one that we can reach
  - Some of these assignments may represent the empty set
  - The answer set solver (we use clingo) used is agnostic to the cost of a shortest path

**Abbreviations and Definitions**

- \( \Pi \): Answer set program
- \( S_{\Pi} \): set of states represented by program
- \( S_{\Pi_{\text{m}}} \): set of states represented by assignment

\[ S_{A_0} \xrightarrow{} S_{A_1} \xrightarrow{} S_{A_2} \]

### Graph Elements

- \( S_0 \) and \( S_{\Pi'} \)
- \( S_{A_0}, S_{A_1}, S_{A_2} \)

- Monotonic Specification

### Mathematical Notation

\[ S \rightarrow S \]

\[ = \]

\[ \Pi \]

\[ \Pi_{\text{m}} \]

\[ \}

\[ \} \]
Handling Non-Monotonicity

- If negation as failure is used in a program, $\Pi$, then $\Pi$ can exhibit non-monotonic behavior
  - A logic program is non-monotonic if some atoms that were previously derived can be retracted by adding new knowledge
  - Therefore, we can have a state that is a candidate state but not a goal state
- For example, a white cross with no yellow stickers on the white face
  - The assignment for this specification is just a white cross
  - However, there can be a state that is a specialization of this assignment, but has yellow on the white face
To reduce the size of candidate states while ensuring there is still at least one goal state, find another minimal assignment, $A_2$, such that

$A \subset A_2$

$A_2 \in \alpha(\Pi)$
Outline

• Overview
• Heuristic function training
• Goal specification and reaching
• Results
• Future work
### Results

<table>
<thead>
<tr>
<th>Goal</th>
<th>Path Cost</th>
<th>% Solved</th>
<th># Models</th>
<th>Model Time</th>
<th>Search Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rubik’s Cube (Canon)</td>
<td>24.41</td>
<td>100%</td>
<td>1</td>
<td>0.37</td>
<td>4.39</td>
</tr>
<tr>
<td>Rubik’s Cube (Cross6)</td>
<td>13.11</td>
<td>100%</td>
<td>1</td>
<td>0.41</td>
<td>2.14</td>
</tr>
<tr>
<td>Rubik’s Cube (Cup4)</td>
<td>24.33</td>
<td>100%</td>
<td>42.5</td>
<td>34.65</td>
<td>374.11</td>
</tr>
<tr>
<td>Rubik’s Cube (CupSpot)</td>
<td>17.99</td>
<td>100%</td>
<td>27.68</td>
<td>38.66</td>
<td>241.08</td>
</tr>
<tr>
<td>Rubik’s Cube (Checkers)</td>
<td>23.85</td>
<td>100%</td>
<td>1</td>
<td>0.49</td>
<td>4.2</td>
</tr>
<tr>
<td>Sokoban (Immov)</td>
<td>35.15</td>
<td>100%</td>
<td>6.37</td>
<td>6.83</td>
<td>16.16</td>
</tr>
<tr>
<td>Sokoban (BoxBox)</td>
<td>33.77</td>
<td>88%</td>
<td>1.89</td>
<td>0.58</td>
<td>6.08</td>
</tr>
<tr>
<td>Sokoban (AgentInBox)</td>
<td>34.42</td>
<td>77%</td>
<td>1.26</td>
<td>0.38</td>
<td>4.09</td>
</tr>
</tbody>
</table>
### Results

<table>
<thead>
<tr>
<th>Goal</th>
<th>Path Cost</th>
<th>% Solved</th>
<th># Models</th>
<th>Model Time</th>
<th>Search Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rubik’s Cube (Canon)</td>
<td>24.41</td>
<td>100%</td>
<td>1</td>
<td>0.37</td>
<td>4.39</td>
</tr>
<tr>
<td>Rubik’s Cube (Cross6)</td>
<td>13.11</td>
<td>100%</td>
<td>1</td>
<td>0.41</td>
<td>2.14</td>
</tr>
<tr>
<td>Rubik’s Cube (Cup4)</td>
<td>24.33</td>
<td>100%</td>
<td>42.5</td>
<td>34.65</td>
<td>374.11</td>
</tr>
<tr>
<td>Rubik’s Cube (CupSpot)</td>
<td>17.99</td>
<td>100%</td>
<td>27.68</td>
<td>38.66</td>
<td>241.08</td>
</tr>
<tr>
<td>Rubik’s Cube (Checkers)</td>
<td>23.85</td>
<td>100%</td>
<td>1</td>
<td>0.49</td>
<td>4.2</td>
</tr>
<tr>
<td>Sokoban (Immov)</td>
<td>35.15</td>
<td>100%</td>
<td>6.37</td>
<td>6.83</td>
<td>16.16</td>
</tr>
<tr>
<td>Sokoban (BoxBox)</td>
<td>33.77</td>
<td>88%</td>
<td>1.89</td>
<td>0.58</td>
<td>6.08</td>
</tr>
<tr>
<td>Sokoban (AgentInBox)</td>
<td>34.42</td>
<td>77%</td>
<td>1.26</td>
<td>0.38</td>
<td>4.09</td>
</tr>
</tbody>
</table>

In Table 2 we see that the path cost for finding the Cross6 goal is almost half that of finding the canonical goal, even though the canonical goal is a subset of the Cross6 goal. This indicates that the trained heuristic function is capable of estimating the cost-to-go to a closest state in a set of goal states without needing to be explicitly told of a closest state. This ability to discover paths to goal states, which themselves, are not known until a path is found, could be extended to domains such as chemical synthesis. For example, this would allow practitioners to specify properties a molecule should or should not have, discover synthesis routes to such molecules and, as a result, discover molecules that meet these specifications.

In Algorithm 1, we sample a new stable model if we fail to find a goal state. From Table 2, we see that the number of models we need to sample for the canonical Rubik’s cube goal state and Cross6 is only one. However, for Cup4 and CupSpot, we must sample, on average, 42.5 and 27.68 models, respectively, to find a goal state. In cases where a goal state was not found, A* search failed to find a path to the sampled stable model. This may be because the sampled stable models represented only unreachable states. We discuss ways to overcome this in the Future Work Section.

For Sokoban, we see that the BoxBox and AgentInBox goals did not achieve a 100% success rate. Since we did not set a maximum iteration for Algorithm 1, all failure cases involved the algorithm terminating because all models were banned. Therefore, A* search failed to find a path to all stable models, which may indicate that the goal was not reachable for these start states. Figure 10 shows start states that failed to reach both the BoxBox and AgentInBox goals. The figure shows that there was not enough room to reach these goals.

#### Discussion

**Table 2: Performance of DeepCubeA when reaching goals specified with ASP.**

- **Figure 7:** Reached goal where all boxes are immoveable.
- **Figure 8:** Reached goal where all boxes form a larger box.
- **Figure 9:** Reached goal where four boxes are at the four corners of the agent.

**Figure 10:** Start states that failed to reach both BoxBox and AgentInBox.

**Related Work**

Action Schema Networks (ASNets) (Toyer et al. 2020) are neural networks that exploit the structure of the Planning Domain Definition Language (PDDL) to learn a policy that generalizes across problem instances. However, ASNets are...
Outline

• Overview
• Heuristic function training
• Goal specification and reaching
• Results
• Future work
Goal Reaching: Non-monotonic

$\Pi$: Answer set program
$\mathcal{S}_\Pi$: set of states represented by program
$\mathcal{S}_A$: set of states represented by assignment

Combine this with a conflict-driven branch-and-bound search
The results in Table 1 show that expressing goals using NAF significantly improves path cost for both the random and conflict-driven specialization operators. Comparisons are along the conflict-driven specialization operator is significantly faster and finds significantly shorter paths. The average number of seconds it took to do a single specialization, the average number of seconds it took to find a single path, the average percentage of reached assignments that were not goal states, the average number of experiments are shown in Table 1. Figures showing goals reached are shown in Figures 2, 3, and 4. Platinum CPU, otherwise. We give a time limit of 500 seconds for each test state. Results for our experiments are shown in Table 1. Figures showing goals reached are shown in Figures 2, 3, and 4.

<table>
<thead>
<tr>
<th>Goal</th>
<th>SpecOp</th>
<th>Cost</th>
<th>%Solve</th>
<th>#Itr</th>
<th>#Assign</th>
<th>%reach</th>
<th>%not goal</th>
<th>Secs Spec</th>
<th>Secs Path</th>
<th>Secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC: ∀diffCtrW</td>
<td>-</td>
<td>11.54</td>
<td>70</td>
<td>3.34</td>
<td>33.43</td>
<td>7.68</td>
<td>0</td>
<td>12.77</td>
<td>7.5</td>
<td>564.94</td>
</tr>
<tr>
<td>RC: ¬∃sameCtrW</td>
<td>Rand</td>
<td>1.67</td>
<td>99</td>
<td>7.2</td>
<td>63.02</td>
<td>87.84</td>
<td>69.06</td>
<td>0.06</td>
<td>1.04</td>
<td>95.46</td>
</tr>
<tr>
<td></td>
<td>Conflict</td>
<td>1.26</td>
<td>100</td>
<td>5.43</td>
<td>36.31</td>
<td>99.34</td>
<td>52.36</td>
<td>0.06</td>
<td>0.07</td>
<td>5.98</td>
</tr>
<tr>
<td>24p:r0SumEven</td>
<td>-</td>
<td>24.55</td>
<td>100</td>
<td>9.24</td>
<td>92.4</td>
<td>100</td>
<td>0</td>
<td>0.2</td>
<td>0.23</td>
<td>42.52</td>
</tr>
<tr>
<td>24p:¬r0SumOdd</td>
<td>Rand</td>
<td>3.16</td>
<td>100</td>
<td>4.27</td>
<td>33.6</td>
<td>100</td>
<td>38.71</td>
<td>0.2</td>
<td>0.03</td>
<td>6.64</td>
</tr>
<tr>
<td></td>
<td>Conflict</td>
<td>2.51</td>
<td>100</td>
<td>4.06</td>
<td>31.6</td>
<td>100</td>
<td>22.13</td>
<td>0.21</td>
<td>0.04</td>
<td>6.58</td>
</tr>
<tr>
<td>24p:∀rSumEven</td>
<td>-</td>
<td>83.71</td>
<td>100</td>
<td>9.19</td>
<td>91.9</td>
<td>50.41</td>
<td>0</td>
<td>0.88</td>
<td>1.77</td>
<td>250.18</td>
</tr>
<tr>
<td>24p:¬∀rSumOdd</td>
<td>Rand</td>
<td>17.07</td>
<td>100</td>
<td>10.23</td>
<td>92.05</td>
<td>99.98</td>
<td>85.51</td>
<td>0.1</td>
<td>0.08</td>
<td>21.72</td>
</tr>
<tr>
<td></td>
<td>Conflict</td>
<td>12.87</td>
<td>100</td>
<td>8.66</td>
<td>77.1</td>
<td>100</td>
<td>79.72</td>
<td>0.11</td>
<td>0.08</td>
<td>17.08</td>
</tr>
</tbody>
</table>

All stickers on the white face are different than the center sticker

All rows sum to an even number

Agostinelli, F., A Conflict-Driven Approach for Reaching Goals Specified with Negation as Failure. ICAPS HAXP Workshop 2024
Questions?

- Code
  - Code available on GitHub
  - https://github.com/forestagostinelli/SpecGoal

Rojina Panta
Vedant Khandelwal

Email: foresta@cse.sc.edu
Website: https://cse.sc.edu/~foresta/