Towards Automated Discovery of God-like Folk Algorithms for Rubik's Cube Supplementary Material: Proofs

Garrett E. Katz, Naveed Tahir
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY, 13244
{gkatz01,ntahir}@syr.edu

Lemma 1. *τ is always non-empty when line 19 of Algorithm 1 is executed.*

Proof. The paths produced by SCRAMBLES$(M-D)$ have length $T < M-D$ and $s^{(T)} = s^*$. By line 2 of Algorithm 2, the initial rule $\mathcal{R}_0$ always has prototype $S_0 = s^*$ and cost $\ell_0 = 0$. Therefore there is at least one $t \leq T$ (namely, $t = T$) and rule (namely $r = 0$) for which $s^{(t)} = S_r$ and $D + t + \ell_r \leq M$. So τ is always non-empty on line 19. ☐

Lemma 2. *Rules created during Algorithm 2 have distinct prototype states, i.e., $S_r = S_{r'}$ implies $r = r'$.*

Proof. Let $r < r'$ index two distinct rules in $\mathcal{R}$. When r' is first added on line 21 of Algorithm 1, the rule search from $s^{(0)}$ has failed on line 16. This means $s^{(0)}$ does not match any existing prototype state, including S_r. Additionally, $s^{(0)}$ is used as the new prototype state $S_{r'}$. Therefore $S_{r'} \neq S_r$. Furthermore, S_r and $S_{r'}$ are never changed once they are added. Therefore, $r \neq r'$ implies $S_r \neq S_{r'}$. The lemma follows by contrapositive. ☐

Lemma 3. *For any rule r, there exists a sequence of rules $\langle r^*_n \rangle_{n=0}^N$ with $r^*_0 = r$ and the following properties:*

- $S_{r^*_{n+1}} = m_{r^*_n}(S_{r^*_n})$ *for $n < N$*
- $S_{r^*_N} = s^*$*, the solved state*
- $\ell_{r^*_0} = \sum_{n=0}^N |m_{r^*_n}| < M - D$

Proof. By induction. In the base case ($|\mathcal{R}| = 1$), the properties are satisfied with $N = 0$ and $r^*_0 = 0$ because $S_0 = s^*$ and $\ell_0 = |m_0| = 0$ from line 2 in Algorithm 2. For the inductive case, consider a new rule R added on line 21 of Algorithm 1 with state $S_R = s^{(0)}$ and macro $m_R = \langle a^{(t)} \rangle_{t=1}^{\hat{t}}$. By line 19 of Algorithm 1, $m_R(S_R) = s^{(\hat{t})}$ is an existing prototype $S_{\hat{r}}$ with $\hat{r} < R$. By the inductive hypothesis, there is a rule sequence $\langle r^*_n \rangle_{n=0}^N$ with $r^*_0 = \hat{r}$ whose chained macros transform $S_{\hat{r}}$ into s^* with $\ell_{\hat{r}}$ total actions. Prepending the new rule $r = R$ with macro m_R to this rule sequence will transform S_R into s^* with $\ell_R = |m_R| + \ell_{\hat{r}}$ total actions. Furthermore, by line 19 in Algorithm 1, $\ell_R = \hat{t} + \ell_{\hat{r}} < M - D$. ☐

Algorithm 1: INCORPORATE$(\mathcal{R}, \langle s^{(t)} \rangle_{t=0}^T, \langle a^{(t)} \rangle_{t=1}^T)$

Input:
$\mathcal{R}$: A (potentially incomplete) macro database
$\langle s^{(t)} \rangle_{t=0}^T, \langle a^{(t)} \rangle_{t=1}^T$: A path with $s^{(T)} = s^*$ and $T \leq M - D$
Output:
$\mathcal{R}$: A (potentially modified) copy of the macro database
ϕ: True if $\mathcal{R}$ was unmodified, False otherwise

1: $\langle (S_r, W_r, m_r, \ell_r) \rangle_{r=1}^R \leftarrow \mathcal{R}$
2: $\phi \leftarrow$ True
3: $r \leftarrow$ QUERY$(\mathcal{R}, s^{(0)})$
4: **if** $r \neq$ False **then**
5: $\quad M' \leftarrow M - (D + |m_r|)$
6: $\quad v, \mathbf{p}, \langle \overline{r}_n \rangle_{n=1}^N, \langle \overline{s}^{(t_n)} \rangle_{n=1}^N \leftarrow \mathcal{A}(M', \mathcal{R}, m_r(s^{(0)}))$
7: $\quad$ **if** $v =$ False **then**
8: $\quad\quad \phi \leftarrow$ False
9: $\quad\quad \overline{s}^{(t_0)}, \overline{r}_0 \leftarrow s^{(0)}, r$
10: $\quad\quad \omega \leftarrow \{(n,k) \mid (0 \leq n \leq N) \land (S_{\overline{r}_n,k} \neq \overline{s}_k^{(t_n)})\}$
11: $\quad\quad$ Choose one $(\hat{n}, \hat{k})$ from ω
12: $\quad\quad W_{\overline{r}_{\hat{n}},\hat{k}} \leftarrow 0$
13: $\quad\quad \mathcal{R} \leftarrow \langle (S_r, W_r, m_r, \ell_r) \rangle_{r=1}^R$
14: $\quad$ **end if**
15: **else**
16: $\quad r', s', \mathbf{p}' \leftarrow$ RULE-SEARCH$(\mathcal{R}, s^{(0)})$
17: $\quad$ **if** $r' =$ False **then**
18: $\quad\quad \phi \leftarrow$ False
19: $\quad\quad \tau \leftarrow \{(t, r) \mid (s^{(t)} = S_r) \land (D + t + \ell_r \leq M)\}$
20: $\quad\quad$ Choose one $(\hat{t}, \hat{r})$ from τ
21: $\quad\quad \mathcal{R} \leftarrow$ ADD_RULE$(\mathcal{R}, s^{(0)}, \langle a^{(t)} \rangle_{t=1}^{\hat{t}}, \hat{t} + \ell_{\hat{r}})$
22: $\quad$ **end if**
23: **end if**
24: Return $\mathcal{R}, \phi$

Algorithm 2: RCONS($\mathcal{H}$)

Input:
$\mathcal{H} = \langle \mathcal{R}_i \rangle_{i=1}^{I}$: Initial modification history
Output:
$\mathcal{H}$: Updated modification history

1: **if** $\mathcal{H} = \langle\rangle$ **then**
2: $\quad \mathcal{R} \leftarrow \langle (s^*, \mathbf{0}, \langle\rangle, 0) \rangle$
3: **else**
4: $\quad \mathcal{R} \leftarrow \mathcal{R}_I$
5: **end if**
6: **repeat**
7: $\quad \varphi \leftarrow$ True
8: $\quad$ **for** $\mathbf{s}, \mathbf{p} \in$ SCRAMBLES$(M - D)$ **do**
9: $\quad\quad \mathcal{R}, \phi \leftarrow$ INCORPORATE$(\mathcal{R}, \mathbf{s}, \mathbf{p})$
10: $\quad\quad \varphi \leftarrow \varphi \wedge \phi$
11: $\quad\quad$ **if** $\neg\phi$ **then**
12: $\quad\quad\quad \mathcal{H} \leftarrow \mathcal{H} \oplus \langle \mathcal{R} \rangle$
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: **until** φ
16: Return $\mathcal{H}$

Lemma 4. *ω is always non-empty when line 10 of Algorithm 1 is executed.*

Proof. By contradiction. Assume (for contradiction) that when line 10 is executed, ω is empty, i.e.,

$$\overline{s}^{(t_n)} = S_{\overline{r}_n} \text{ for every } n \in \{0, 1, ..., N\} \tag{1}$$

Let $\langle r_n^* \rangle_{n=0}^{N'}$ with $r_0^* = \overline{r}_0$ be the rule sequence provided by Lemma 3. We will show that $\langle r_n^* \rangle_{n=0}^{N'} = \langle \overline{r}_n \rangle_{n=0}^{N}$ by inductivion on n. For the base case, we already have $r_0^* = \overline{r}_0$. For the inductive case from n to $n+1$, let $s' = m_{\overline{r}_n}(\overline{s}^{(t_n)})$. We have:

$$\begin{aligned} s' &= m_{\overline{r}_n}(S_{\overline{r}_n}) && \text{(by assumption, Eq. 1)} && (2) \\ &= m_{r_n^*}(S_{r_n^*}) && \text{(by the inductive hypothesis)} && (3) \\ &= S_{r_{n+1}^*} && \text{(by Lemma 3)} && (4) \end{aligned}$$

Since s' matches a rule (namely, r_{n+1}^*), BFS rule search from s' in $\mathcal{A}$ will not proceed past depth 0. Therefore, $s' = \overline{s}^{(t_{n+1})}$. Since $s' = S_{r_{n+1}^*}$ also (by Eq. 4), we have:

$$\begin{aligned} S_{r_{n+1}^*} &= \overline{s}^{(t_{n+1})} && && (5) \\ S_{r_{n+1}^*} &= S_{\overline{r}_{n+1}} && \text{(by assumption, Eq. 1)} && (6) \\ r_{n+1}^* &= \overline{r}_{n+1} && \text{(by Lemma 2)} && (7) \end{aligned}$$

Therefore, by induction, $r_n^* = \overline{r}_n$ up to $n = N'$, at which point $\overline{s}^{(t_{N'})} = s^*$, the solved state. Furthermore, again by Lemma 3, the total number of actions $|m_{\overline{r}_0}| + \sum_{n=1}^{N'} |m_{\overline{r}_n}| < M - D$, and therefore $\sum_{n=1}^{N'} |m_{\overline{r}_n}| < M - D - |m_{\overline{r}_0}| = M'$.

It follows that the invocation of $\mathcal{A}$ on line 6 reaches s^* in $N = N'$ rule applications and at most M' actions, so it returns $v =$ True. But if $v =$ True, the condition on line 7 is False, and line 10 is not executed, which contradicts the initial assumption. $\square$

Proposition 1. *Construction can always proceed and terminates in finite time.*

Proof. By Lemmas 1 and 4, there is always at least one choice available for modifying wildcards (line 10) or adding rules (line 19) when Algorithm 1 needs to. Hence construction can always proceed.

Algorithm 1 add rules but never removes them, and by Lemma 2, each rule has a distinct state. Hence the total number of rules is monotonically non-decreasing and bounded above by the total number of possible states. Therefore $|\mathcal{R}|$ converges to a fixed point at some finite iteration I of line 9 in Algorithm 2.

Algorithm 1 never sets wildcards of existing rules to 1, only to 0 (on line 12). Furthermore, no new rules are added after iteration I. Therefore, after iteration I, the total number of non-zero wildcards ($\sum_{r,k} W_{r,k}$) is monotonically non-increasing and bounded below by 0. Therefore the set of non-zero wildcards also converges to a fixed point at some finite iteration $J \geq I$.

The code branches that set ϕ to False in Algorithm 1 are the same branches that add new rules and set wildcards to zero. Therefore, Algorithm 2's first full pass over scrambled states after iteration J, ϕ will never be set to False, and the outer loop (lines 6-15) will halt. $\square$

Proposition 2. *When construction terminates, the returned rule set $\mathcal{R}$ is correct: i.e., for any state s, $\mathcal{A}(M, \mathcal{R}, s)$ returns a path from s to s^* in at most M actions.*

Proof. When Algorithm 2 terminates, ϕ was never set to False in the last outer iteration (lines 6-15). In particular, the conditions on lines 7 and 17 of Algorithm 1 are always False. This means that every state $s \in \mathcal{S}$ is within D steps of at least one s' matching a rule, and once a matching rule is applied to any state s', $\mathcal{A}$ finds a path from s' to s^* in at most $M - D$ steps. Therefore $\mathcal{A}$ will find a path from any state s to s^* in at most M steps. $\square$