Final year intership report

Verification in Isabelle/HOL of Hopcroft's algorithm for minimizing DFAs including runtime analysis

Vincent Trélat

April 11, 2023

Contents

1 Introduction

1.1 Original algorithm

John E. Hopcroft's algorithm for minimizing DFAs was first presented in his original 1971 paper [Hop71] as a formal algorithm. Algorithm 1 is a direct translation of the original algorithm with only slight changes in the notations.

Algorithm 1: Hopcroft's original formal algorithm

Data: Input: a finite DFA $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$
Result: Output: the equivalence class of $\mathcal{Q}$ under state equivalence

Construct $\delta^{-1}(q, a) := \{t \in \mathcal{Q} \mid \delta(t, a) = q\}$ for all $q \in \mathcal{Q}$ and $a \in \Sigma$;
Construct $P_1 := \mathcal{F}$, $P_2 := \mathcal{Q} \setminus \mathcal{F}$ and $s_{a,i} := \{q \in P_i \mid \delta^{-1}(q, a) \neq \varnothing\}$ for all $i \in \{1, 2\}$ and $a \in \Sigma$;
Let $k := 3$;
For all $a \in \Sigma$, construct $L_a := \underset{0 \leq i < k}{\arg\min} \ |s_{a,i}|$;
while $\exists a \in \Sigma, L_a \neq \varnothing$ **do**
    Pick $a \in \Sigma$ such that $L_a \neq \varnothing$ and $i \in L_a$;
    $L_a := L_a \setminus \{i\}$;
    forall $j < k, \exists q \in P_j, \delta(q, a) \in s_{a,i}$ **do**
        $P'_j := \{t \in P_j \mid \delta(t, a) \in s_{a,i}\}$ and $P''_j := P_j \setminus P'_j$;
        $P_j := P'_j$ and $P_k := P''_j$; construct $s_{a,j}$ and $s_{a,k}$ for all $a \in \Sigma$ accordingly ;
        For all $a \in \Sigma$, $L_a := \begin{cases} L_a \cup \{j\} & \text{if } j \notin L_a \wedge |s_{a,j}| \leq |s_{a,k}| \\ L_a \cup \{k\} & \text{otherwise} \end{cases}$;
        $k := k + 1$;
    end
end

1.2 Modern formalisation

Today, the algorithm is usually given in a more mathematical and formalised way[1], as presented below in Algorithm 2.

Definition 1. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, I, F)$ be a DFA. Let P be a partition of Q. Let $B \in P$ and $a \in \Sigma$. We say for $C \in P$ that (a, C) splits B if

$$\exists q_1, q_2 \in B \quad \delta(q_1, a) \in C \wedge \delta(q_2, a) \notin C.$$

[1] see for example [EB23]

If (a, C) is a splitter of B, P can be updated to $P \setminus \{B\} \cup \{B', B''\}$, where

$$B' := \{q \in B \mid \delta(q, a) \in C\} \text{ and } B'' := B \setminus B'.$$

Algorithm 2: Hopcroft's algorithm in a modern style

Data: Input: a finite DFA $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$
Result: Output: the language partition P_ℓ

```
1  if F = ∅ ∨ Q \ F = ∅ then
2  |  return Q
3  else
4  |  P := {F, Q \ F} ;
5  |  W := {(a, min{F, Q \ F}, a ∈ Σ} ;
6  |  while W ≠ ∅ do
7  |  |  Pick (a, B') from W ;
8  |  |  forall B ∈ P do
9  |  |  |  Split B with (a, B') into B_0 and B_1 ;
10 |  |  |  P := (P \ {B}) ∪ {B_0, B_1} ;
11 |  |  |  forall b ∈ Σ do
12 |  |  |  |  if (b, B) ∈ W then
13 |  |  |  |  |  W := (W \ {(b, B)}) ∪ {(b, B_0), (b, B_1)} ;
14 |  |  |  |  else
15 |  |  |  |  |  W := W ∪ {(b, min{B_0, B_1})} ;
16 |  |  |  |  end
17 |  |  |  end
18 |  |  end
19 |  end
20 end
```

2 Proof of correctness

3 Time complexity analysis

We focus on the original algorithm presented in Algorithm 1 in order to work on the arguments given in [Hop71]. The data structures used at that time were mostly linked lists, but let us rather give some requirements for the data structures instead of actual implementations. The goal is to show that the algorithm can be executed in $O(m \cdot n \log n)$ time, where m is the number of symbols in the alphabet and n is the number of states in the DFA.

The following requirements come directly from [Hop71] and are specific to the algorithm presented in Algorithm 1.

Requirement 1. Sets such as $\delta^{-1}(q, a)$ and L_a must be represented in a way that allows $O(1)$ time for addition and deletion in front position.

Requirement 2. Vectors must be maintained to indicate whether a state is in a given set.

Requirement 3. Sets such as P_i must be represented in a way that allows $O(1)$ time for addition and deletion at any given position.

Requirement 4. For a state q in a set P_i or $s_{a,i}$, its position must be determined in $O(1)$ time.

VT: Maybe not necessary? This should be provable from Req. 2 and Req. 3.

Lemma 1. Lines 1 to 4 can be executed in $O(|\Sigma| \cdot |\mathcal{Q}|)$ time.

Proof. The non trivial part is the computation of the inverse transition function $\delta^{-1}(q, a)$, for all $q \in \mathcal{Q}$ and $a \in \Sigma$. This can be done in $O(|\Sigma| \cdot |\mathcal{Q}|)$ time by iterating over Σ and traversing the automaton (e.g. with a DFS) while keeping track of the predecessor at each step. ■

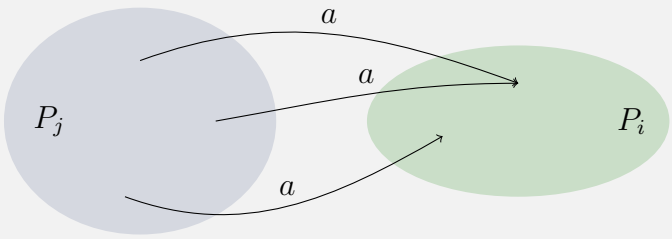
Lemma 2. An iteration of the loop at line 5 taken for a letter a and an index $i \in L_a$ takes a time proportional to the number of transitions terminating in P_i and the number of symbols in the alphabet, i.e. $\Theta(|\Sigma| \cdot |s_{a,i}|)$ time.

Proof. We pick an $a \in \Sigma$ such that $L_a \neq \varnothing$ and an $i \in L_a$. We need to examine all $j < k$ such that $\exists q \in P_j, \delta(q, a) \in s_{a,i}$ to construct the sets corresponding to splitting the block P_j w.r.t. a and P_i.

Let $j < k$. From the definition of $s_{a,i}$, we obtain the following:

$$\begin{aligned} \exists q \in P_j, \delta(q, a) \in s_{a,i} &\iff \exists q \in P_j, \delta(q, a) \in P_i \wedge \underbrace{\delta^{-1}(\delta(q, a), a) \neq \varnothing}_{\text{true}} \\ &\iff \exists q \in P_j, \delta(q, a) \in P_i \end{aligned}$$

Which corresponds to finding states in P_j having an outgoing a-transition to a state in P_i, as represented in the following scheme:



This set of states can be expressed via the inverse transition function:

$$\{q \in P_j \mid \delta(q,a) \in P_i\} = \left(\bigcup_{q \in P_i} \delta^{-1}(q,a)\right) \cap P_j$$

Since δ^{-1} was already computed in the first step of the algorithm, we can determine using req. 3 whether a state of $\bigcup_{q \in P_i} \delta^{-1}(q,a)$ is also in P_j in $\Theta(1)$ time.

Thus, instead of examining P_j for all $j < k$, we rather go through the table of δ^{-1} and for each state q such that $\delta(q,a) \in P_i$, we know from req. 4 that we can determine the index $j < k$ (because there are k blocks) of the block P_j containing q in $\Theta(1)$ time. The sets P'_j and $P''_j = P_k$ resulting from the partition can be constructed on the fly without any additional time cost. The construction of the sets b_j and b_k as well as the update of L_b for all $b \in \Sigma$ can also be done on the fly but require $\Theta(1)$ time for each symbol $b \in \Sigma$ and thus add up to a total of $\Theta(|\Sigma|)$ time. Finally, notice that

$$\left|\bigcup_{q \in P_i} \delta^{-1}(q,a)\right| = \left|\{q \in P_i \mid \delta^{-1}(q,a) \neq \varnothing\}\right| = |s_{a,i}|$$

Overall, an iteration of the loop takes $\Theta\left(|\Sigma| \cdot |s_{a,i}|\right)$ time.

■

VT: $|\Sigma|$ should not appear here!!! Otherwise it would result in a $O(|\Sigma|^2|\mathcal{Q}|\log|\mathcal{Q}|)$ complexity!

We will now justify the logarithmic factor in the time complexity. We briefly explain why a logarithm stands out and prove the statement by induction. The idea is that for each symbol $a \in \Sigma$, a state $q \in \mathcal{Q}$ can be in at most one of the splitters $s_{a,i}$. When the loop iterates over this splitter, it will split the block and keep the smaller one, whose size will be at most the size of the splitter divided by two. This means that $s_{a,i}$ can be processed at most $\log|s_{a,i}|$ times. We now properly state and prove the property by induction.

We see splitters as updatable program variables. This means that $s_{a,i}$ as a set may differ along the loop, however we know there exists some $q \in \mathcal{Q}$ such that $s_{a,i}$ is the unique set containing q throughout the execution. This gives a way to characterize splitters throughout the whole execution.

Lemma 3. Let $a \in \Sigma$. Each splitter $s_{a,i}$ where $i \in L_a$ is processed – i.e. picked at line 6 – at most $\lfloor \log|s_{a,i}| \rfloor$ times.

Proof. We first show the following statement:

During an iteration, the chosen splitter $s_{a,i}$ will either be removed from the set of splitters (i.e. $L_a := L_a \setminus \{i\}$) or its size will be reduced by at least $\frac{|s_{a,i}|}{2}$ after the iteration.

Let $\{P_1, \cdots, P_\ell\}$ be the current partition and let $s_{a,i}$ be the selected splitter. Thus, $s_{a,i}$ is no longer in the set of splitters, i.e. $i \notin L_a$. Since we go over all splittable blocks, P_i may also be split by $s_{a,i}$.

- If P_i is not split by $s_{a,i}$, then $s_{a,i}$ was already removed from the splitters. Note that it can be added again later if it is split by another splitter.
- If P_i is split into P_i' and P_i'', then the smaller set is added to the splitters. If P_i'' is the smaller one, a new splitter $s_{a,k}$ is added to the splitters[a] and $s_{a,i}$ was already removed from the set of splitters. If P_i' is the smaller one, then P_i is updated to P_i' and thus $s_{a,i}$ is updated to a set whose size is at most $\frac{|s_{a,i}|}{2}$.

Therefore, since a splitter cannot be empty, $s_{a,i}$ can be processed at most m times where m is such that

$$1 \leq \frac{|s_{a,i}|}{2^m} < 2$$

which is equivalent to

$$m \leq \log |s_{a,i}| < m + 1 \quad \text{i.e.} \quad \lfloor \log |s_{a,i}| \rfloor = m$$

■

[a] Note that $|s_{a,k}| \leq \frac{|s_{a,i}|}{2}$

Lemma 4. Let $a \in \Sigma$. Let us consider some step in the algorithm such that $\mathcal{P}$ is the current partition. The total time spent in the loop until termination for any symbol a is bounded by

$$T_a := \theta \left(\sum_{i \in L_a} |s_{a,i}| \log |s_{a,i}| + \sum_{i \in \{1, \cdots, |\mathcal{P}|\} \setminus L_a} |s_{a,i}| \log \frac{|s_{a,i}|}{2} \right)$$

where θ is the constant of proportionality that may be obtained from lemma 2.

Proof. We show the result by induction over the steps.

Base case: the current partition is $\{P_1, P_2\}$ and we may assume w.l.o.g. that $L_a = \{1\}$. We have to show that the total time spent in the loop for a is bounded by $T_a = |s_{a,1}| \log |s_{a,1}| + |s_{a,2}| \log \frac{|s_{a,2}|}{2}$.

- We know from lemma 2 that an iteration of the loop for $s_{a,i}$ takes $\theta\,|s_{a,i}|$ time. We also know from lemma 3 that $s_{a,i}$ can be processed at most $\log |s_{a,i}|$ times[a], the total time for $s_{a,1}$ is bounded by $\theta\,|s_{a,1}| \log |s_{a,1}|$.
- For $s_{a,2}$, we know that it is not in the set of splitters, but it can be added if P_2 is split by $s_{a,1}$ into P_2' and P_2''. Since L_a is empty, the smaller one is added as a splitter and thus has a size at most $\frac{|s_{a,2}|}{2}$ and thus can be processed at most this many times. An iteration over this new splitter will take less than $\theta\,|s_{a,2}|$ time, thus the total time for $s_{a,2}$ is bounded by $\theta\,|s_{a,2}| \log \frac{|s_{a,2}|}{2}$.

Inductive step: Let $\{P_1, \cdots, P_\ell\} := \mathcal{P}$ be the current partition. The induction hypothesis states that the total time spent in the loop until termination for a is bounded by

$$T_a := \theta \left(\sum_{i \in L_a} |s_{a,i}| \log |s_{a,i}| + \sum_{i \in \{1,\cdots,\ell\} \setminus L_a} |s_{a,i}| \log \frac{|s_{a,i}|}{2} \right)$$

By going through one more step, some blocks of $\mathcal{P}$ may be split and we define a new time $\hat{T}_a$ over this new partition and we have to show that $\hat{T}_a \leq T_a$.

Suppose P_j is split into P_j' and P_j'' by any splitter. Let $\tilde{s}_{a,j}$ be the new value for $s_{a,j}$ and $s_{a,\ell+1}$ the new potential splitter. The partition is updated as well, i.e. P_j is updated to P_j' and $P_{\ell+1} := P_j''$ is added to the partition. We have to consider the cases where $j \in L_a$ and $j \notin L_a$ separately.

- $j \in L_a$: is added to the set of splitters, so now $\ell + 1 \in L_a$ and the partition is updated as well. Note that

$$|s_{a,\ell+1}| = |s_{a,j}| - |\tilde{s}_{a,j}|$$

Thus, instead of taking $\theta\,|s_{a,j}|\log|s_{a,j}|$ time for $s_{a,j}$, it now takes

$$\theta\left(|\tilde{s}_{a,j}|\log|\tilde{s}_{a,j}| + |s_{a,\ell+1}|\log|s_{a,\ell+1}|\right)$$

i.e.

$$\theta\left(|\tilde{s}_{a,j}|\log|\tilde{s}_{a,j}| + (|s_{a,j}| - |\tilde{s}_{a,j}|)\log(|s_{a,j}| - |\tilde{s}_{a,j}|)\right)$$

By using concavity of the logarithm, we have:

$$|\tilde{s}_{a,j}|\log|\tilde{s}_{a,j}| + (|s_{a,j}| - |\tilde{s}_{a,j}|)\log(|s_{a,j}| - |\tilde{s}_{a,j}|) \leq |s_{a,j}|\log|s_{a,j}|$$

- $j \notin L_a$: the smaller splitter between $\tilde{s}_{a,j}$ and $s_{a,\ell+1}$ is added to the set of splitters. If σ is the smaller splitter and ξ is the other one, then it is added as splitter, i.e. its index is added to L_a. Thus, the corresponding term in the sum $|s_{a,j}|\log\frac{|s_{a,j}|}{2}$ is updated as follows:

$$\theta\left(|\sigma|\log|\sigma| + \theta\,|\xi|\log\left(\frac{|\xi|}{2}\right)\right)$$

Since $|\xi| = |s_{a,j}| - |\sigma|$, we have:

$$\underbrace{\theta\,|\sigma|\log|\sigma|}_{\text{because } \sigma \text{ is a splitter}} + \underbrace{\theta(|s_{a,j}| - |\sigma|)\log\left(\frac{|s_{a,j}| - |\sigma|}{2}\right)}_{\text{because the other splitter is not added as a splitter}}$$

By using the fact that $|\sigma| \leq \left|\frac{s_{a,j}}{2}\right|$, we have:

$$\begin{aligned}
&|\sigma|\log|\sigma| + (|s_{a,j}| - |\sigma|)\log\left(\frac{|s_{a,j}|}{2} - |\sigma|\right) - |s_{a,j}|\log\frac{|s_{a,j}|}{2} \\
&\leq |\sigma|\log\frac{|s_{a,j}|}{2} + (|s_{a,j}| - |\sigma|)\log\left(\frac{|s_{a,j}|}{2} - |\sigma|\right) - |s_{a,j}|\log\frac{|s_{a,j}|}{2} \\
&\leq (|s_{a,j}| - |\sigma|)\log\left(\frac{\frac{|s_{a,j}|}{2} - |\sigma|}{\frac{|s_{a,j}|}{2}}\right) \leq 0
\end{aligned}$$

Overall, splitting a block of $\mathcal{P}$ does not increase the total time spent over the loop, hence $\hat{T}_a \leq T_a$.

■

[a] We drop the floor operator for simplicity. Note that since we are giving upper bounds, this is completely valid.

From lemma 4, we obtain in particular that for the initial partition, the

following holds:

$$\forall a \in \Sigma, T_a \leq \theta(|P_1| + |P_2|) \log(|P_1| + |P_2|) = \theta|\mathcal{Q}| \log |\mathcal{Q}|$$

By iterating over all symbols in Σ, we eventually obtain the following theorem.

Theorem 1. Algorithm 1 runs in $O\left(|\Sigma| \cdot |\mathcal{Q}| \log |\mathcal{Q}|\right)$ time.

References

[EB23] Javier Esparza and Michael Blondin. *Automata Theory: An Algorithmic Approach*. 2023.

[Hop71] John E. Hopcroft. *An n Log n Algorithm for Minimizing States in a Finite Automaton*. Stanford University, Stanford, CA, USA, 1971.