# Compilers: DFA Minimization

a topic in

DM565 – Formal Languages and Data Processing

Kim Skak Larsen

Department of Mathematics and Computer Science (IMADA)
University of Southern Denmark (SDU)

*kslarsen@imada.sdu.dk*

October, 2023

# DFA Minimization

One can prove that given a regular language $L$, the DFA with the fewest states recognizing $L$ is unique (up to renaming of states).

On the next slides, we will see how to compute such a smallest DFA.

The algorithm we use assumes that all states are reachable, so to arrive at the correct result in all cases, we start by removing unreachable states.

The algorithms in these slides have been taken from the corresponding Wikipedia page.

# Computing Reachable States for DFA

```
let reachable_states := {q0};
let new_states := {q0};
do {
    temp := the empty set;
    for each q in new_states do
        for each c in Σ do
            temp := temp ∪ {p such that p = δ(q,c)};
        end;
    end;
    new_states := temp \ reachable_states;
    reachable_states := reachable_states ∪ new_states;
} while (new_states ≠ the empty set);
unreachable_states := Q \ reachable_states;
```

# DFA Minimization

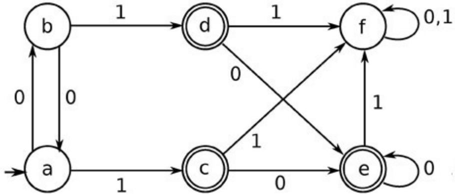
Next, we run the DFA Minimization Algorithm on the next slide with one modification:

It has been proven that it is sufficient to add only one of the sets $F$ or $Q \setminus F$ to $W$. That is what we will do in the example following the presentation of the algorithm, including the smallest of those two sets (for efficiency).

# DFA Minimization Algorithm

```
P := {F, Q \ F};
W := {F, Q \ F};
while (W is not empty) do
     choose and remove a set A from W
     for each c in Σ do
          let X be the set of states for which a transition on c leads to a state in A
          for each set Y in P for which X ∩ Y is nonempty and Y \ X is nonempty do
               replace Y in P by the two sets X ∩ Y and Y \ X
               if Y is in W
                    replace Y in W by the same two sets
               else
                    if |X ∩ Y| <= |Y \ X|
                         add X ∩ Y to W
                    else
                         add Y \ X to W
          end;
     end;
end;
```

# Example DFA




Vevek, Wikipedia

In the example to follow, be aware that there are two $c$'s with different meaning. One is the $c$ from the algorithm and the other is a label from the example DFA. When it appears as $c = 0$ or $c = 1$, it is the $c$ from the algorithm.

# Example

P: $\{\{c, d, e\}, \{a, b, f\}\}$
W: $\{\{c, d, e\}\}$

# Example

P: $\{\{c, d, e\}, \{a, b, f\}\}$
W: $\{\cancel{\{c, d, e\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$

# Example

P: $\{\{c, d, e\}, \{a, b, f\}\}$
W: $\{\cancel{\{c, d, e\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$
$c = 0$:

# Example

P: $\{\{c, d, e\}, \{a, b, f\}\}$
W: $\{\cancel{\{c, d, e\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$
$c = 0$: $X = \{c, d, e\}$

# Example

P: $\{\{c, d, e\}, \{a, b, f\}\}$
W: $\{\cancel{\{c, d, e\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$
$c = 0$: $X = \{c, d, e\}$
$c = 1$:

# Example

P: $\{\{c, d, e\}, \{a, b, f\}\}$
W: $\{\cancel{\{c, d, e\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$
$c = 0\text{: } X = \{c, d, e\}$
$c = 1\text{: } X = \{a, b\}$

# Example

P: $\{\{c,d,e\}, \cancel{\{a,b,f\}}, \{a,b\}, \{f\}\}$
W: $\{\cancel{\{c,d,e\}}\}$

$\underline{\text{1. iteration: } A = \{c,d,e\}}$
$c = 0$: $X = \{c,d,e\}$
$c = 1$: $X = \{a,b\}$

# Example

P: $\{\{c, d, e\}, \cancel{\{a, b, f\}}, \{a, b\}, \{f\}\}$
W: $\{\cancel{\{c, d, e\}}, \{f\}\}$

1. iteration: $A = \{c, d, e\}$

$c = 0$: $X = \{c, d, e\}$
$c = 1$: $X = \{a, b\}$

# Example

P: $\{\{c, d, e\}, \cancel{\{a, b, f\}}, \{a, b\}, \{f\}\}$
W: $\{\cancel{\{c, d, e\}}, \cancel{\{f\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$
$c = 0$: $X = \{c, d, e\}$
$c = 1$: $X = \{a, b\}$

$\underline{\text{2. iteration: } A = \{f\}}$

# Example

P: $\{\{c,d,e\}, \cancel{\{a,b,f\}}, \{a,b\}, \{f\}\}$
W: $\{\cancel{\{c,d,e\}}, \cancel{\{f\}}\}$

$\underline{\text{1. iteration: } A = \{c,d,e\}}$
$c = 0$: $X = \{c,d,e\}$
$c = 1$: $X = \{a,b\}$

$\underline{\text{2. iteration: } A = \{f\}}$
$c = 0$:

# Example

P: $\{\{c, d, e\}, \cancel{\{a, b, f\}}, \{a, b\}, \{f\}\}$
W: $\{\cancel{\{c, d, e\}}, \cancel{\{f\}}\}$

1. iteration: $A = \{c, d, e\}$

$c = 0$: $X = \{c, d, e\}$
$c = 1$: $X = \{a, b\}$

2. iteration: $A = \{f\}$

$c = 0$: $X = \{f\}$

# Example

P: $\{\{c, d, e\}, \cancel{\{a, b, f\}}, \{a, b\}, \{f\}\}$
W: $\{\cancel{\{c, d, e\}}, \cancel{\{f\}}\}$

$\underline{1.\ \text{iteration: } A = \{c, d, e\}}$
$c = 0$: $X = \{c, d, e\}$
$c = 1$: $X = \{a, b\}$

$\underline{2.\ \text{iteration: } A = \{f\}}$
$c = 0$: $X = \{f\}$
$c = 1$:

# Example

P: $\{\{c, d, e\}, \cancel{\{a, b, f\}}, \{a, b\}, \{f\}\}$
W: $\{\cancel{\{c, d, e\}}, \cancel{\{f\}}\}$

1. iteration: $A = \{c, d, e\}$

$c = 0$: $X = \{c, d, e\}$
$c = 1$: $X = \{a, b\}$

2. iteration: $A = \{f\}$

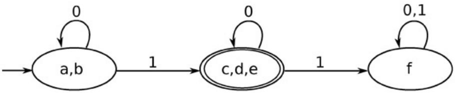
$c = 0$: $X = \{f\}$
$c = 1$: $X = \{c, d, e, f\}$

# Example

P: $\{\{c, d, e\}, \cancel{\{a, b, f\}}, \{a, b\}, \{f\}\}$
W: $\{\cancel{\{c, d, e\}}, \cancel{\{f\}}\}$

$\underline{\text{1. iteration: } A = \{c, d, e\}}$
$c = 0$: $X = \{c, d, e\}$
$c = 1$: $X = \{a, b\}$

$\underline{\text{2. iteration: } A = \{f\}}$
$c = 0$: $X = \{f\}$
$c = 1$: $X = \{c, d, e, f\}$

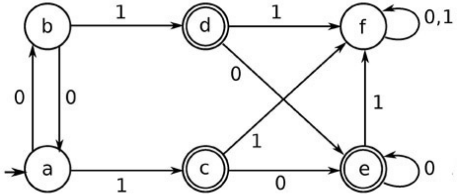



Vevek, Wikipedia

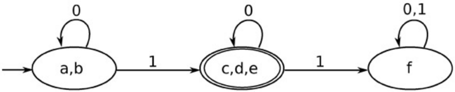
# Constructing the Minimal DFA

- The set $P$ is the set of states.
- The start state is the state containing the original start state.
- Any state containing an original final state is a final state.
- If original states $a$ and $b$ are contained in new states $S_a$ and $S_b$ and there is a transition on $\alpha \in \Sigma$ from $a$ to $b$, then there is a transition on $\alpha$ from $S_a$ to $S_b$.

# Example (continued)




Vevek, Wikipedia




Vevek, Wikipedia

# An Application in Type Checking

See the lecture notes for a discussion of how this can be used to decide structural equivalence very efficiently.