Brzozowski's Algorithm for Automata Minimization Verified in Coq

Filipe Ramos[1]
ofiliperamos@gmail.com

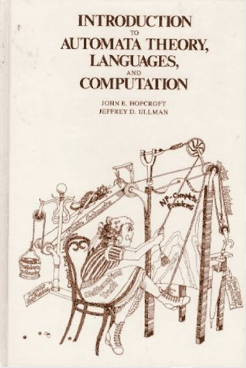
Karina Girardi Roggia[1]
karina.roggia@udesc.br

Rafael Castro G. Silva[2]
rasi@di.ku.dk

[1]Universidade do Estado de Santa Catarina
[2]University of Copenhagen

2024

Introduction



- Why minimize DFA?
  - To reduce memory usage
  - To simplify the automaton for reasoning
  - To obtain the canonical DFA for a given regular language
- Table-filling algorithm [8]
- Brzozowski's algorithm [5]
  - Simpler to understand
  - Easier to implement
  - Despite $\mathcal{O}(2^n)$, it frequently behaves well in practice [3]
- Lack of correctness proofs of the algorithm in Coq

Introduction

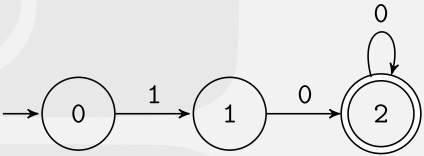
Coq Proof Assistant [2]:

- Formal proof management system
- Tactics allowing manageable steps
- Automation for simpler proofs
- Broad user community

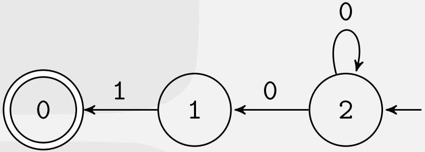
Proof goals:

- reversal produces an automaton that accepts the strings reversed from the source language;
- after reversal and determinization, the paths of the new automaton connect set states that have original states connected by reverse paths;
- the minimized automaton is deterministic;
- the language of the minimized DFA is the same as that of the input automaton;
- all constructed states are reachable and distinguishable.

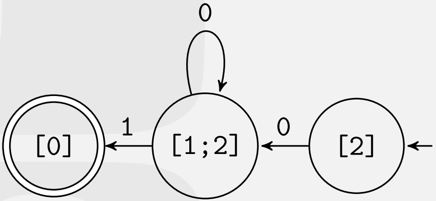
Brzozowski's Algorithm



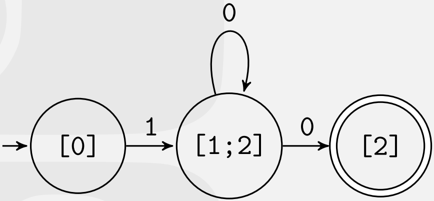
Brzozowski's Algorithm



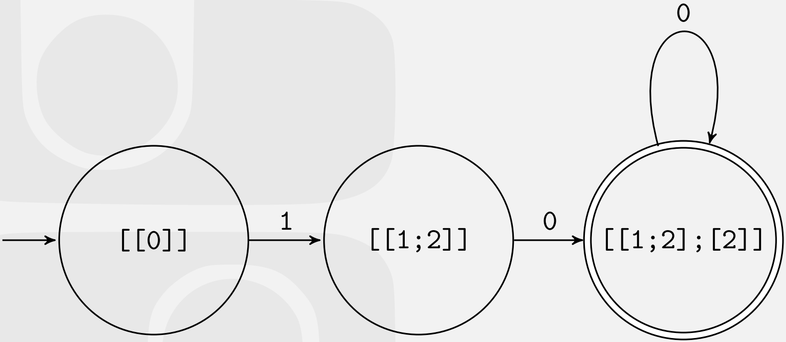
Brzozowski's Algorithm



Brzozowski's Algorithm



Brzozowski's Algorithm



Finite Automata in Coq

$\langle Q, \Sigma, \delta, I, F \rangle$

```
Context {State Symbol : Type}.

Record NFA := {
    states : list State;
    alphabet : list Symbol;
    transition : State -> Symbol ->
list State;
    start_states : list State;
    accept_states : list State
}.
```

- Need for predicates
- Difficult function handling in definitions and proofs
- Impossible to apply induction directly
- Complex to apply transformations (e.g. reversal)

Finite Automata in Coq

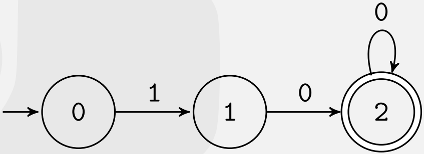
$\langle Q, \Sigma, \delta, I, F \rangle$

```
Variant NFA_comp :=
    | state (q:State)
    | symbol (a:Symbol)
    | start (q:State)
    | accept (q:State)
    | transition (q1:State)
      (a:Symbol) (q2:State).

Definition NFA := list NFA_comp.
```

- Less predicates
- Transitions computed using pattern-matching functions (e.g. `transitionf`)
- Simple induction
- Easy to transform through pattern-matching

Finite Automata in Coq



- [state 0; state 1; state 2; start 0; accept 2; transition 0 1 1; transition 1 0 2; transition 2 0 2]
- [start 0; accept 2; transition 0 1 1; transition 1 0 2; transition 2 0 2]

Deterministic Finite Automata

```
Variable nfa : NFA.

Definition start_singleton := ∃ q, In q (start_sts nfa) ∧
∀ q1 q2, In q1 (start_sts nfa) → In q2 (start_sts nfa) →
q1 = q2.

Definition transitionf_det := ∀ q a q1 q2,
let s := transitionf [q] a in In q1 s → In q2 s → q1 = q2.

Definition is_dfa := start_singleton ∧ transitionf_det.
```

- Same representation for both NFA and DFA
- NFA $\leftrightarrow$ DFA conversion facilitated

Automaton Reversal in Coq

1. `start q1` becomes `accept q1`;
2. `accept q2` becomes `start q2`;
3. `transition q3 a q4` becomes `transition q4 a q3`.

Reversed Language

```
Inductive path (g:NFA) : State → State → Word → Prop :=
  | path_nil q : path g q q nil
  | path_trans q1 q2 q3 a w : In (transition q1 a q2) g →
    path g q2 q3 w → path g q1 q3 (a::w).
```

$$\text{path g q1 q2 w} \Leftrightarrow \text{path } g^R \text{ q2 q1 } w^R$$

$$L(g^R) = \{w^R | w \in L(g)\}$$

$$L(g_{min}) = \{(w^R)^R | w \in L(g)\}$$

Automaton Determinization in Coq

1. `start (start_states g') :: transition Q a (transitionf g' Q a)`
2. State list normalization
   - `[0; 1; 2]` $\equiv$ `[1; 2; 0]`
   - Reduction of redundancy
   - Consistency in state representation

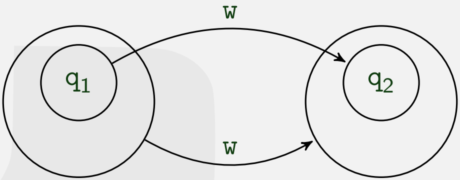
     `transitionf g' [0; 1; 2] a = [1; 2; 0]`
3. Accepting states appending
   - The previously generated states with an accepting state
4. Removal of unreachable states
   - Pumping lemma

Determinization Correctness

By applying induction over the input automaton, we can obtain:

- Only one start state is generated
- `transitionf (det g') [Q1] a` = `[Q2; ...; Q2]`
  since only one `transition Q1 a Q2` is generated

Proof of Equivalence



$$\mathtt{q}_1 \in \mathtt{Q} \wedge \mathtt{q}_2 \in \mathtt{ext\_transitionf\ g'\ [q_1]\ w} \Leftrightarrow$$
$$\mathtt{q}_2 \in \mathtt{Q'} \in \mathtt{ext\_transitionf\ (det\ g')\ [Q]\ w}$$

Let $\texttt{Q}_1$ and $\texttt{Q}_2$ be two indistinguishable states in $\texttt{det g}^R$:

$\texttt{ext\_transitionf (det g}^R\texttt{) [Q}_1\texttt{] w} = \texttt{[Q'}_1\texttt{; ...]}$
$\texttt{ext\_transitionf (det g}^R\texttt{) [Q}_2\texttt{] w} = \texttt{[Q'}_2\texttt{; ...]}$
$\texttt{Q'}_1 \in \texttt{accept\_sts (det g}^R\texttt{)} \Leftrightarrow \texttt{Q'}_2 \in \texttt{accept\_sts (det g}^R\texttt{)}$

for all $\texttt{w}$.

Let $\texttt{Q}_1$ and $\texttt{Q}_2$ be two indistinguishable states in $\texttt{det g}^\texttt{R}$:

$\texttt{ext\_transitionf (det g}^\texttt{R}\texttt{) [Q}_1\texttt{] w} = \texttt{[Q'}_1\texttt{; ...]}$
$\texttt{ext\_transitionf (det g}^\texttt{R}\texttt{) [Q}_2\texttt{] w} = \texttt{[Q'}_2\texttt{; ...]}$
$\texttt{Q'}_1 \ni \texttt{q0}_1 \Leftrightarrow \texttt{Q'}_2 \ni \texttt{q0}_2$

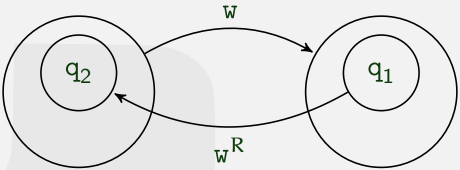
for all $\texttt{w}$ and some start states $\texttt{q0}_1$ and $\texttt{q0}_2$ in g.

Let $\texttt{Q}_1$ and $\texttt{Q}_2$ be two indistinguishable states in $\texttt{det g}^R$:

$\texttt{ext\_transitionf (det g}^R\texttt{) [Q}_1\texttt{] w} = \texttt{[Q'}_1\texttt{; ...]}$
$\texttt{ext\_transitionf (det g}^R\texttt{) [Q}_2\texttt{] w} = \texttt{[Q'}_2\texttt{; ...]}$
$\texttt{Q'}_1 \ni \texttt{q0} \Leftrightarrow \texttt{Q'}_2 \ni \texttt{q0}$

for all w and the start state q0 of g, assuming g is deterministic.

Minimization Correctness



$$Q \ni q_2 \in \texttt{ext\_transitionf g } [q_1] \texttt{ w}^R \Leftrightarrow$$
$$q_1 \in Q' \in \texttt{ext\_transitionf (det g}^R\texttt{) [Q] w}$$

Minimization Correctness

Let `Q`$_1$ and `Q`$_2$ be two indistinguishable states in `det g`^{R}:

`ext_transitionf (det g`^{R}`) [Q`$_1$`] w` $=$ `[Q'`$_1$`; ...]`
`ext_transitionf (det g`^{R}`) [Q`$_2$`] w` $=$ `[Q'`$_2$`; ...]`
`Q'`$_1$ $\ni$ q0 $\Leftrightarrow$ `Q'`$_2$ $\ni$ q0
`ext_transitionf g [q0] w`^{R} $\subseteq$ `Q`$_1$ $\Leftrightarrow$
`ext_transitionf g [q0] w`^{R} $\subseteq$ `Q`$_2$

for all `w` and the start state q0 of g, assuming g is deterministic.

Which means every reachable state of g is in `Q`$_1$ iff it is in `Q`$_2$.

Let $\mathtt{Q}_1$ and $\mathtt{Q}_2$ be two indistinguishable states in $\mathtt{det}\ \mathtt{g}^{\mathrm{R}}$:

$\mathtt{ext\_transitionf\ (det\ g}^{\mathrm{R}}\mathtt{)\ [Q}_1\mathtt{]\ w} = \mathtt{[Q'}_1\mathtt{;\ \ldots]}$
$\mathtt{ext\_transitionf\ (det\ g}^{\mathrm{R}}\mathtt{)\ [Q}_2\mathtt{]\ w} = \mathtt{[Q'}_2\mathtt{;\ \ldots]}$
$\mathtt{Q'}_1 \ni \mathtt{q0} \Leftrightarrow \mathtt{Q'}_2 \ni \mathtt{q0}$
$\mathtt{ext\_transitionf\ g\ [q0]\ w}^{\mathrm{R}} \subseteq \mathtt{Q}_1 \Leftrightarrow$
$\mathtt{ext\_transitionf\ g\ [q0]\ w}^{\mathrm{R}} \subseteq \mathtt{Q}_2$

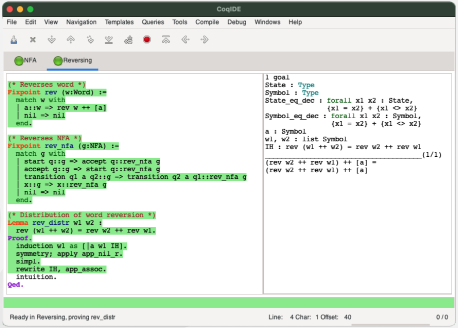
for all w and the start state q0 of g, assuming g is deterministic.

> Which means every state of g is in $\mathtt{Q}_1$ iff it is in $\mathtt{Q}_2$, assuming all states of g are reachable. Thus, $\mathtt{Q}_1 = \mathtt{Q}_2$ as we wanted to prove.

Since `det g`^{R} is deterministic and all its states are reachable, `det (det g`^{R}`)`^{R} is minimal for any finite automaton g.

Qed!

Conclusion



- Finite automata as lists!
- Proofs guided by **tactics** with well-defined syntax and step-by-step procedures
- Brzozowski's algorithm verified with Coq's standard library
- We have shown it works for both NFA and DFA, with multiple start states
- Approximately 2400 lines of proofs, 580 lines of specifications

References

- Athalye, A.: CoqIOA: a formalization of IO automata in the Coq proof assistant. Ph.D. thesis, Massachusetts Institute of Technology (2017), https://dspace.mit.edu/handle/1721.1/112831
- Bertot, Y., Castran, P.: Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions. Springer Publishing Company, Incorporated, 1st edn. (2010)
- Bonchi, F., Bonsangue, M.M., Rutten, J.J., Silva, A.: Brzozowski's algorithm (co) algebraically. In: Logic and Program Semantics, pp. 12–23. Springer (2012), https://link.springer.com/chapter/10.1007/978-3-642-29485-3_2
- Braibant, T., Pous, D.: Deciding Kleene algebras in Coq. Logical Methods in Computer Science **8** (2012), https://lmcs.episciences.org/1043

References

- Brzozowski, J., Tamm, H.: Theory of átomata. In: Theoretical Computer Science. vol. 539, pp. 13–27. Elsevier (2014), https://www.sciencedirect.com/science/article/pii/S0304397514002953
- Cassandras, C.G., Lafortune, S.: Introduction to discrete event systems. Springer Science+Business Media, New York, 2 edn. (2008)
- Doczkal, C., Kaiser, J.O., Smolka, G.: A constructive theory of regular languages in Coq. In: International Conference on Certified Programs and Proofs. pp. 82–97. Springer (2013), https://link.springer.com/chapter/10.1007/978-3-319-03545-1_6
- Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. Pearson/Addison Wesley, Boston, 3 edn. (2006)
- Jan, H.: Proof of Brzozowski's algorithm for DFA minimization (2019), https://cs.stackexchange.com/questions/105574/proof-of-brzozowskis-algorithm-for-dfa-minimization

References

- Paulson, L.C.: A formalisation of finite automata using hereditarily finite sets. In: International Conference on Automated Deduction. pp. 231–245. Springer (2015), https://link.springer.com/chapter/10.1007/978-3-319-21401-6_15
- Pierce, B.C., de Amorim, A.A., Casinghino, C., Gaboardi, M., Greenberg, M., Hriţcu, C., Sjöberg, V., Yorgey, B.: Logical Foundations, vol. 1. UPenn CIS, Pennsylvania, 6.6 edn. (2024)
- Ramos, F.: Prova da minimização de autômatos finitos determinísticos pelo algoritmo de brzozowski assistida por computador (2021), https://pergamumweb.udesc.br/acervo/152736, supervisors: Karina Girardi Roggia, Rafael Castro G. Silva

Brzozowski's Algorithm for Automata Minimization Verified in Coq

Filipe Ramos[1]
ofiliperamos@gmail.com

Karina Girardi Roggia[1]
karina.roggia@udesc.br

Rafael Castro G. Silva[2]
rasi@di.ku.dk

[1]Universidade do Estado de Santa Catarina
[2]University of Copenhagen

2024