Brzozowski's algorithm: minimal language acceptors via duality

Helle Hvid Hansen

Radboud Universiteit Nijmegen and CWI

COIN, 4 Oct 2012

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$\mathcal{A}$$

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$rev(\mathcal{A})$$

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$det(rev(\mathcal{A}))$$

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$reach(det(rev(\mathcal{A})))$$

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$reach(det(rev(reach(det(rev(\mathcal{A}))))))$$

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$reach(det(rev(reach(det(rev(\mathcal{A}))))))$$

is a minimal deterministic automaton accepting L.

Brzozowski's Minimisation Algorithm

Brzozowski's Algorithm (1963)

Given a deterministic automaton accepting a language L,

$$reach(det(rev(reach(det(rev(\mathcal{A}))))))$$

is a minimal deterministic automaton accepting L.

Remarks:

- Theoretically, no more efficient than partition refinement.
- In practice, often performs well.
- Works also for nondeterministic automata.

History and Motivation

History:

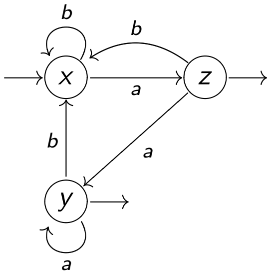
- Joint work with:
  Bonchi, Bonsangue, Panangaden, Rutten, Silva.
- Extends: Bonchi, Bonsangue, Rutten, Silva (2011). (Proof based on duality reachability-observability (Arbib-Manes).
- New: Brzozowski via adjunction of automata,
- New: Generalisation to nondeterministic automata and weighted automata.

Motivation: Gain deeper understanding of

- the construction/algorithm
- relation to similar constructions,

Brzozowski's Algorithm: Example

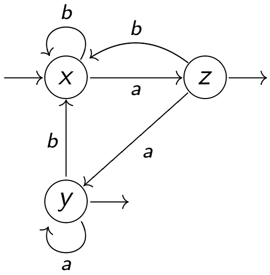
Start:



Accepts $L = (a + b)^* a$

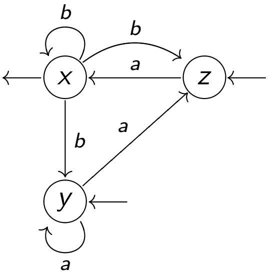
Brzozowski's Algorithm: Example

Start:



Accepts $L = (a+b)^*a$

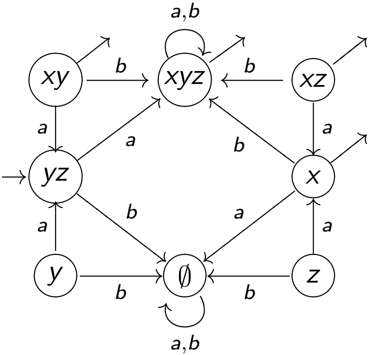
After reversing transitions:



Accepts $rev(L) = a(a+b)^*$

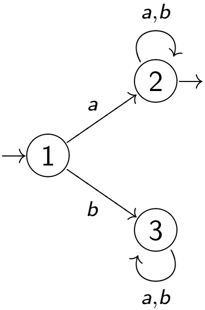
Brzozowski's Algorithm: Example

After determinising:



Accepts $rev(L) = a(a+b)^*$

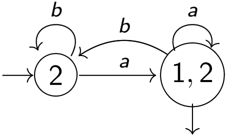
After taking reachable part:



Accepts $rev(L) = a(a+b)^*$

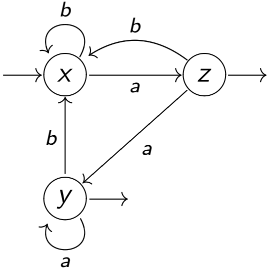
Brzozowski's Algorithm: Example

After doing it all again:



Accepts $rev(rev(L)) = (a + b)^* a$

Original automaton:



Accepts $L = (a + b)^* a$

Overview

- Automata and categories.
- Adjunction of automata via reversal.
- Brzozowski, functorially.
- Generalisations to Moore and weighted automata.
- Related work.

Deterministic Automata

$$(1 = \{0\}) \qquad \begin{tikzcd} 1 \arrow[dr, "i"] & & 2 \\ & X \arrow[ur, "f"] \arrow[d, "t"] & \\ & X^A & \end{tikzcd} \qquad (2 = \{0,1\})$$

Deterministic Automata are Algebras and Coalgebras

$$
(1 = \{0\}) \qquad
\begin{tikzcd}
1 \arrow[dr, "i"'] & & 2 \\
& X \arrow[ur, "f"'] \arrow[d, "t"] & \\
& X^A &
\end{tikzcd}
\qquad (2 = \{0, 1\})
$$

$$
\frac{\dfrac{X \to X^A}{A \times X \to X}}{A \to (X \to X)}
$$

transitions are both
algebra and coalgebra

Deterministic Automata are Algebras and Coalgebras

$$(1 = \{0\}) \qquad \begin{tikzcd} 1 \arrow[dr, "i"] & & 2 \\ & X \arrow[ur, "f"] \arrow[d, "t"] & \\ & X^A & \end{tikzcd} \qquad (2 = \{0,1\})$$

$$\frac{\dfrac{X \to X^A}{A \times X \to X}}{A \to (X \to X)}$$

$$\begin{tikzcd} X \arrow[d, "\langle f, t \rangle"] \\ 2 \times X^A \end{tikzcd}$$

transitions are both algebra and coalgebra

output, transitions $2 \times (-)^A$-coalgebra

Deterministic Automata are Algebras and Coalgebras

$$(1 = \{0\}) \qquad \begin{tikzcd} 1 \arrow[dr, "i"] & & 2 \\ & X \arrow[ur, "f"'] \arrow[d, "t"] & \\ & X^A & \end{tikzcd} \qquad (2 = \{0,1\})$$

$$\begin{tikzcd} 1 + A \times X \arrow[d, "{[i,t]}"] \\ X \end{tikzcd} \qquad \frac{\dfrac{X \to X^A}{A \times X \to X}}{A \to (X \to X)} \qquad \begin{tikzcd} X \arrow[d, "{\langle f,t \rangle}"] \\ 2 \times X^A \end{tikzcd}$$

initial state, transitions	transitions are both	output, transitions
$1 + A \times (-)$-algebra	algebra and coalgebra	$2 \times (-)^A$-coalgebra

Initial Algebras and Final Coalgebras

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "i"] & & 2 \\
A^* \arrow[r, dotted, "r"] \arrow[d, "\alpha"'] & X \arrow[r, dotted, "o"] \arrow[d, "t"] \arrow[ur, "f"] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\zeta"] \\
(A^*)^A \arrow[r, dotted, "r^A"] & X^A \arrow[r, dotted, "o^A"] & (2^{A^*})^A
\end{tikzcd}
$$

For all $a \in A, w \in A^*$:

$$
\begin{array}{rcll}
\alpha(w)(a) & = & wa & \text{(append } a\text{)} \\
\zeta(S)(a) & = & \{w \in A^* \mid aw \in S\} = a^{-1}S & \text{(left } a\text{-derivative)}
\end{array}
$$

Initial Algebras and Final Coalgebras

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "i"] & & 2 \\
A^* \arrow[r, dotted, "r"] \arrow[d, "\alpha"'] & X \arrow[r, dotted, "o"] \arrow[d, "t"] \arrow[ur, "f"] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\zeta"] \\
(A^*)^A \arrow[r, dotted, "r^A"] & X^A \arrow[r, dotted, "o^A"] & (2^{A^*})^A
\end{tikzcd}
$$

For all $a \in A, w \in A^*$:

$$
\begin{array}{rcll}
\alpha(w)(a) & = & wa & \text{(append } a\text{)} \\
\zeta(S)(a) & = & \{w \in A^* \mid aw \in S\} = a^{-1}S & \text{(left } a\text{-derivative)} \\
\\
r(w) & = & t(i)(w) & \text{(state reached on input } w\text{)} \\
o(x) & = & \{w \in A^* \mid f(t(x)(w)) = 1\} & \text{(language acepted by } x\text{)}
\end{array}
$$

Reachability, Observability, Minimality

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "i"] & & 2 \\
A^* \arrow[r, dotted, "r"] \arrow[d, "\alpha"'] & X \arrow[ur, "f"] \arrow[r, dotted, "o"] \arrow[d, "t"] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(A^*)^A \arrow[r, dotted, "r^A"] & X^A \arrow[r, dotted, "o^A"] & (2^{A^*})^A
\end{tikzcd}
$$

Def. (Arbib & Manes)

Automaton $\langle X, t, i, f \rangle$ is ...

- reachable if r is surjective (no algebraic redundancy).
- observable if o is injective (no coalgebraic redundancy).
- minimal if it is *reachable and observable*.

Categories of Automata

Aut = category of all deterministic automata, and automaton morphisms:

$$
\begin{tikzcd}
1 \arrow[d, "i"'] \arrow[rrd, "j"] & & 2 \\
X \arrow[rr, "h"] \arrow[rru, "f"'] \arrow[d, "t"'] & & Y \arrow[u, "g"'] \arrow[d, "s"] \\
X^A \arrow[rr, "h^A"] & & Y^A
\end{tikzcd}
$$

Note:

- Automaton morphisms preserve language.
- No initial object, no final object in Aut.

The Category Aut(L)

Aut(L) = subcategory of Aut of automata accepting L.

The Category Aut(L)

Aut(L) = subcategory of Aut of automata accepting L.

Initial and final objects regained:

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "i"] \arrow[drr, bend left, "L"] & & 2 \\
A^* \arrow[r, dotted, "r"] \arrow[urr, bend left, "\chi_L"] \arrow[d, "\alpha"'] & X \arrow[r, dotted, "o"] \arrow[ur, "f"] \arrow[d, "t"] & 2^{A^*} \arrow[u, "\epsilon?"'] \arrow[d, "\beta"] \\
(A^*)^A \arrow[r, dotted, "r^A"] & X^A \arrow[r, dotted, "o^A"] & (2^{A^*})^A
\end{tikzcd}
$$

The Category Aut(L)

Aut(L) = subcategory of Aut of automata accepting L.

Initial and final objects regained:

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "i"] \arrow[drr, bend left, "L"] & & 2 \\
A^* \arrow[r, dotted, "r"] \arrow[d, "\alpha"'] \arrow[urr, bend left, "\chi_L"] & X \arrow[r, dotted, "o"] \arrow[d, "t"] \arrow[ur, "f"] & 2^{A^*} \arrow[u, "\epsilon?"'] \arrow[d, "\beta"] \\
(A^*)^A \arrow[r, dotted, "r^A"] & X^A \arrow[r, dotted, "o^A"] & (2^{A^*})^A
\end{tikzcd}
$$

Automaton $\langle X, t, i, f \rangle$ in Aut(L) is ...

- reachable if <u>initial morphism r is surjective</u>.
- observable if <u>final morphism o is injective</u>.

Myhill-Nerode via Aut(L)

$$\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[r, "\chi_L"', phantom] \arrow[dr, bend left, "L"] & 2 \\
A^* \arrow[d, "\alpha"'] \arrow[r, dotted, "r=o"] \arrow[ur, bend left, "\chi_L"] & 2^{A^*} \arrow[u, "\epsilon?"'] \arrow[d, "\zeta"] \\
(A^*)^A \arrow[r, dotted, "r^A=o^A"] & (2^{A^*})^A
\end{tikzcd}$$

- $o(w) = \{u \in A^* \mid wu \in L\} = w^{-1}L$

Myhill-Nerode via Aut(L)

$$\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "L"] & 2 \\
A^* \arrow[r, dotted, "r=o"] \arrow[ur, "\chi_L"] \arrow[d, "\alpha"'] & 2^{A^*} \arrow[u, "\epsilon?"'] \arrow[d, "\zeta"] \\
(A^*)^A \arrow[r, dotted, "r^A=o^A"] & (2^{A^*})^A
\end{tikzcd}$$

- $o(w) = \{u \in A^* \mid wu \in L\} = w^{-1}L$
- $\ker(o)$ is Myhill-Nerode-equivalence:
  $$w \equiv_L v \quad \text{iff} \quad \forall u \in A^* : wu \in L \iff vu \in L$$
- $\mathrm{img}(o)$ is set of left-quotients of L.
- $|\mathrm{img}(o)| = \mathrm{index}(\equiv_L)$

Myhill-Nerode via Aut(L)

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "L" near start] & 2 \\
A^* \arrow[ur, "\chi_L"' near end] \arrow[r, dotted, "r=o"] \arrow[d, "\alpha"'] & 2^{A^*} \arrow[u, "\epsilon?"'] \arrow[d, "\zeta"] \\
(A^*)^A \arrow[r, dotted, "r^A=o^A"] & (2^{A^*})^A
\end{tikzcd}
$$

Characterisation:

L regular

iff index($\equiv_L$) is finite

iff left-quotients(L) is finite

- $o(w) = \{u \in A^* \mid wu \in L\} = w^{-1}L$
- ker(o) is Myhill-Nerode-equivalence:
  $w \equiv_L v \quad \text{iff} \quad \forall u \in A^* : wu \in L \iff vu \in L$
- img(o) is set of left-quotients of L.
- $|\mathrm{img}(o)| = \mathrm{index}(\equiv_L)$

Adjoint Automata: Main tools

- Contravariant powerset functor $2^{(-)}$: $(2=\{0,1\})$

$$
\begin{array}{rcl}
X & \mapsto & 2^X = \{g\colon X \to 2\} \\
f\colon X \to Y & \mapsto & 2^f = f^{-1}\colon 2^Y \to 2^X \\
 & & f^{-1}(S \subseteq Y) = \{x \in X \mid f(x) \in S\}
\end{array}
$$

Adjoint Automata: Main tools

- Contravariant powerset functor $2^{(-)}$: $(2=\{0,1\})$

$$
\begin{array}{rcl}
X & \mapsto & 2^X = \{g\colon X \to 2\} \\
f\colon X \to Y & \mapsto & 2^f = f^{-1}\colon 2^Y \to 2^X \\
 & & f^{-1}(S \subseteq Y) = \{x \in X \mid f(x) \in S\}
\end{array}
$$

- Adjunction of state spaces:

$$
\begin{tikzcd}
\mathsf{Set} \arrow[r, bend left, "2^{(-)}"] \arrow[r, phantom, "\perp"] & \mathsf{Set}^{\mathrm{op}} \arrow[l, bend left, "2^{(-)^{\mathrm{op}}}"]
\end{tikzcd}
\qquad\qquad
\frac{X \to 2^Y \quad \text{in Set}}{2^X \to Y \quad \text{in Set}^{\mathrm{op}}}
$$

Adjoint Automata: Main tools

- Contravariant powerset functor $2^{(-)}$: $(2 = \{0,1\})$

$$
\begin{array}{rcl}
X & \mapsto & 2^X = \{g\colon X \to 2\} \\
f\colon X \to Y & \mapsto & 2^f = f^{-1}\colon 2^Y \to 2^X \\
 & & f^{-1}(S \subseteq Y) = \{x \in X \mid f(x) \in S\}
\end{array}
$$

- Adjunction of state spaces:

$$
\begin{tikzcd}
\mathsf{Set} \arrow[r, bend left, "2^{(-)}"] \arrow[r, phantom, "\perp"] & \mathsf{Set}^{\mathrm{op}} \arrow[l, bend left, "2^{(-)^{\mathrm{op}}}"]
\end{tikzcd}
\qquad\qquad
\frac{X \to 2^Y \quad \text{in Set}}{2^X \to Y \quad \text{in Set}^{\mathrm{op}}}
$$

- Exponential transpose:

$$
\frac{g\colon X \to 2^Y \quad \text{in Set}}{\hat{g}\colon Y \to 2^X \quad \text{in Set}}
$$

Reversing an Automaton

- $2^{(-)}$ reverses transitions and determinises:

$$t = (t_a \colon X \to X)_{a \in A} \quad \longmapsto \quad ({t_a}^{-1} \colon 2^X \to 2^X)_{a \in A} =: 2^t$$

Reversed transitions: $S \xrightarrow{a} t_a^{-1}(S)$ (a-predecessors of S)

Reversing an Automaton

- $2^{(-)}$ reverses transitions and determinises:

$$t = (t_a \colon X \to X)_{a \in A} \qquad \longmapsto \qquad ({t_a}^{-1} \colon 2^X \to 2^X)_{a \in A} =: 2^t$$

Reversed transitions: $S \xrightarrow{a} t_a^{-1}(S)$ (a-predecessors of S)

- initial becomes final:

$$i \colon 1 \to X \quad \longmapsto \quad 2^i \colon 2^X \to 2^1 = 2$$

In reversed automaton: S is final iff $i \in S$.

Reversing an Automaton

- $2^{(-)}$ reverses transitions and determinises:

$$t = (t_a \colon X \to X)_{a \in A} \quad \longmapsto \quad ({t_a}^{-1} \colon 2^X \to 2^X)_{a \in A} =: 2^t$$

Reversed transitions: $S \xrightarrow{a} t_a^{-1}(S)$ (a-predecessors of S)

- initial becomes final:

$$i \colon 1 \to X \quad \longmapsto \quad 2^i \colon 2^X \to 2^1 = 2$$

In reversed automaton: S is final iff $i \in S$.

- final becomes initial:

$$f \colon X \to 2 = 2^1 \quad \longmapsto \quad \hat{f} \colon 1 \to 2^X$$

In reversed automaton: initial state is set of final states $\hat{f}$.

Reversing an Automaton

$$
\begin{tikzcd}
1 \arrow[dr, "i"] & & 2 & & 1 \arrow[dr, "\hat{f}"] & & 2 \\
& X \arrow[ur, "f"] \arrow[d, "t"] & \arrow[rr, mapsto, "\overline{2}"] & & & 2^X \arrow[ur, "2^i"'] \arrow[d, "2^t"] & \\
& X^A & & & & (2^X)^A &
\end{tikzcd}
$$

Reversing an Automaton

$$
\begin{tikzcd}
1 \arrow[dr, "i"'] & & 2 & & 1 \arrow[dr, "\hat{f}"] & & 2 \\
& X \arrow[ur, "f"'] \arrow[d, "t"] & & \overset{\overline{2}}{\longmapsto} & & 2^X \arrow[ur, "2^i"'] \arrow[d, "2^t"] & \\
& X^A & & & & (2^X)^A &
\end{tikzcd}
$$

Theorem:

- Reversing is functor $\overline{2}\colon \mathrm{Aut} \to \mathrm{Aut}^{\mathrm{op}}$.
- If $\mathcal{X}$ accepts L, then $\overline{2}(\mathcal{X})$ accepts $rev(L) = \{w^R \mid w \in L\}$.
- Reversing is functor $\overline{2}\colon \mathrm{Aut}(L) \to \mathrm{Aut}(rev(L))^{\mathrm{op}}$.

Theorem: Reversal lifts dual adjunction on Set to dual adjunction of automata:

$$\begin{tikzcd}
\mathrm{Aut}(L) \arrow[rr, bend left, "\overline{2}"] \arrow[rr, phantom, "\perp"] \arrow[d] & & \mathrm{Aut}(rev(L))^{\mathrm{op}} \arrow[ll, bend left, "\overline{2}^{\mathrm{op}}"] \arrow[d] \\
\mathrm{Set} \arrow[rr, bend left, "2"] \arrow[rr, phantom, "\perp"] & & \mathrm{Set}^{\mathrm{op}} \arrow[ll, bend left, "2^{\mathrm{op}}"]
\end{tikzcd}$$

Corollary (duality): Let $\mathcal{A}$ be initial object in Aut(L), $\mathcal{Z}$ the final object in Aut($rev(L)$), and let $\mathcal{X}$ be an automaton in Aut(L).

$$r\colon \mathcal{A} \twoheadrightarrow \mathcal{X} \qquad \xmapsto{\overline{2}} \qquad o\colon \overline{2}(\mathcal{X}) \rightarrowtail \overline{2}(\mathcal{A}) = \mathcal{Z}$$

$\mathcal{X}$ reachable $\qquad \Longrightarrow \qquad$ $\overline{2}(\mathcal{X})$ observable

Brzozowski's algorithm, functorially

- Let: $$\begin{array}{rcl} \mathrm{rAut}(L) & = & \text{reachable automata accepting } L, \\ \mathrm{oAut}(L) & = & \text{observable automata accepting } L, \\ \mathrm{mAut}(L) & = & \text{minimal automata accepting } L. \end{array}$$
- Reachability is functor $R\colon \mathrm{Aut}(L) \to \mathrm{rAut}(L)$ (coreflector). Restricts to $R\colon \mathrm{oAut}(L) \to \mathrm{mAut}(L)$.

Brzozowski's algorithm, functorially

- Let:
$$\begin{array}{lcl} \mathrm{rAut}(L) & = & \text{reachable automata accepting } L, \\ \mathrm{oAut}(L) & = & \text{observable automata accepting } L, \\ \mathrm{mAut}(L) & = & \text{minimal automata accepting } L. \end{array}$$
- Reachability is functor $R\colon \mathrm{Aut}(L) \to \mathrm{rAut}(L)$ (coreflector). Restricts to $R\colon \mathrm{oAut}(L) \to \mathrm{mAut}(L)$.
- Brzozowski's algorithm is $R \circ \overline{2}^{\mathrm{op}} \circ R^{\mathrm{op}} \circ \overline{2}$:

$$\begin{tikzcd}
\mathrm{Aut}(L) \arrow[r, "\overline{2}"] \arrow[d, dotted, "O"'] & \mathrm{Aut}(rev(L))^{\mathrm{op}} \arrow[d, "R^{\mathrm{op}}"] \\
\mathrm{oAut}(L) \arrow[d, "R"'] & \mathrm{rAut}(rev(L))^{\mathrm{op}} \arrow[l, "\overline{2}^{\mathrm{op}}"'] \\
\mathrm{mAut}(L) &
\end{tikzcd}$$

Brzozowski for Moore Automata

Deterministic Automata

$$\begin{tikzcd} 1 \arrow[dr, "i"] & & 2 \\ & X \arrow[ur, "f"] \arrow[d, "t"] & \\ & X^A & \end{tikzcd}$$

Moore Automata

$$\begin{tikzcd} 1 \arrow[dr, "i"] & & B \\ & X \arrow[ur, "f"] \arrow[d, "t"] & \\ & X^A & \end{tikzcd}$$

- Moore automata accept B-weighted languages $\sigma \colon A^* \to B$
- Reversal functor $B^{(-)} = \mathrm{Set}(-, B)$.
- Adjunction for Moore automata, Brzozowski algorithm ✓

Brzozowski for Weighted Automata

Weighted Automaton in Set

$$\begin{tikzcd}
1 \arrow[dr, "i"] & & \mathbb{S} \\
& X \arrow[ur, "f"] \arrow[d, "t"] & \\
& V(X)^A &
\end{tikzcd}$$

Moore Automaton in SMod

$$\begin{tikzcd}
1 \arrow[dr, "i"] & & \mathbb{S} \\
& V(X) \arrow[ur, "f^\sharp"] \arrow[d, "t^\sharp"] & \\
& V(X)^A &
\end{tikzcd}$$

- $\mathbb{S}$ is a commutative semiring $(S, +, \cdot, 0, 1)$ (e.g. $(\mathbb{N}, +, \cdot, 0, 1)$.
- SMod $=$ $\mathbb{S}$-semimodules and $\mathbb{S}$-linear maps
- $V(X) = \{s_1x_1 + \ldots + s_nx_n \mid s_i \in \mathbb{S}, x_i \in X\}$ (free on X)

Brzozowski for Weighted Automata

Weighted Automaton in Set

$$\begin{tikzcd} 1 \arrow[dr, "i"'] & & {\color{blue}\mathbb{S}} \\ & X \arrow[ur, "f"'] \arrow[d, "t"] & \\ & V(X)^A & \end{tikzcd}$$

Moore Automaton in SMod

$$\begin{tikzcd} 1 \arrow[dr, "i"'] & & {\color{blue}\mathbb{S}} \\ & V(X) \arrow[ur, "f^\sharp"'] \arrow[d, "t^\sharp"] & \\ & V(X)^A & \end{tikzcd}$$

- $\mathbb{S}$ is a commutative semiring $(S, +, \cdot, 0, 1)$ (e.g. $(\mathbb{N}, +, \cdot, 0, 1)$.
- SMod $=$ $\mathbb{S}$-semimodules and $\mathbb{S}$-linear maps
- $V(X) = \{s_1x_1 + \ldots + s_nx_n \mid s_i \in \mathbb{S}, x_i \in X\}$ (free on X)
- Reversal functor: ${\color{blue}\mathbb{S}^{(-)} = \mathrm{SMod}(-, \mathbb{S})}$ (taking dual space)
- Dual adjunction of Moore automata over SMod (check...)

Brzozowski for Weighted Automata

$$
\begin{tikzcd}
\text{WAut over Set} \arrow[d, "lin.ext"] \arrow[drr, bend left] & & \\
\text{Moore over SMod} \arrow[rr, "\mathbb{S}^{(-)}"] & & \text{Moore over SMod}^{\mathrm{op}} \arrow[d, "\text{forget}"] \\
& & \text{Moore over Set}^{\mathrm{op}} \arrow[d, "R^{\mathrm{op}}"] \\
\text{oMoore over Set} \arrow[d, "R"] & & \text{rMoore over Set}^{\mathrm{op}} \arrow[ll, "B^{(-)^{\mathrm{op}}}"'] \\
\text{mMoore over Set} & &
\end{tikzcd}
$$

Brzozowski for Weighted Automata

$$
\begin{tikzcd}
\textcolor{blue}{\mathsf{WAut\ over\ Set}} \arrow[d, "lin.ext"] \arrow[dr, bend left] & \\
\mathsf{Moore\ over\ SMod} \arrow[r, "\mathbb{S}^{(-)}"] & \mathsf{Moore\ over\ SMod}^{\mathrm{op}} \arrow[d, "\mathsf{forget}"] \\
& \mathsf{Moore\ over\ Set}^{\mathrm{op}} \arrow[d, "R^{\mathrm{op}}"] \\
\mathsf{oMoore\ over\ Set} \arrow[d, "R"] & \mathsf{rMoore\ over\ Set}^{\mathrm{op}} \arrow[l, "B^{(-)^{\mathrm{op}}}"'] \\
\textcolor{blue}{\mathsf{mMoore\ over\ Set}} &
\end{tikzcd}
$$

Note: Nondeterministic automata are weighted automata over Boolean semiring $\mathbb{S} = (2, \vee, \wedge, 0, 1)$.

Related Work

- Bezhanishvili, Kupke, Panangaden (WoLLIC 2012): minimisation via dual equivalence coalgebra-algebra (deterministic, linear weighted, belief automata).

$$
\begin{tikzcd}
\mathrm{Coalg}(T) \arrow[rr, bend left] \arrow[rr, phantom, "\cong"] \arrow[dd] & & \mathrm{Alg}(L)^{\mathrm{op}} \arrow[ll, bend left] \arrow[dd] \\
& & \\
\mathsf{C} \arrow[rr, bend left] \arrow[rr, phantom, "\cong"] & & \mathsf{D}^{\mathrm{op}} \arrow[ll, bend left]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{PODFA} \arrow[rr, bend left] \arrow[rr, phantom, "\cong"] \arrow[dd] & & \mathsf{FBAO}^{\mathrm{op}} \arrow[ll, bend left] \arrow[dd] \\
& & \\
\mathsf{FinSet} \arrow[rr, bend left] \arrow[rr, phantom, "\cong"] & & \mathsf{FinBA}^{\mathrm{op}} \arrow[ll, bend left]
\end{tikzcd}
$$

- Roumen (M.Sc. thesis, 2012): generalised duality of automata (quantum automata, Boolean automata)
- Gehrke, Pin: duality between languages and Stone spaces.
- ...

Conclusion

Summary:

- Brzozowski algorithm via dual adjunction of automata.
- Generalisations: given Moore/nondeterm/weighted automaton accepting L, construct minimal Moore automaton accepting L (language equivalence!).
- Future work: other automaton types (probabilistic, ω-automata, ...), combination with generalised powerset construction, algebraic-coalgebraic automata theory.

Message:

- duality $\rightsquigarrow$ algorithms.
- category theory $\rightsquigarrow$ generalisations.