Automata & rational languages

<u>Defn</u> Let $\mathcal{A}$ be an alphabet. Any subset $\mathcal{L} \subset \mathcal{A}^*$ is called a language.

Simplest languages: finite languages

step up: rational ones ← to be defined later.

Given an rws $(\mathcal{R}, <)$ we are interested in the language of words reducible w.r.t $\mathcal{R}$.

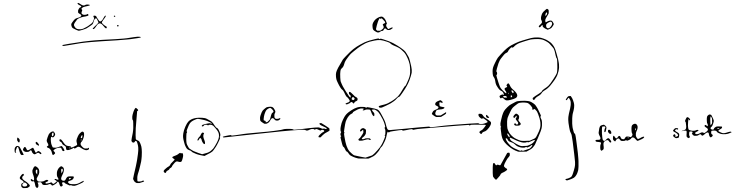
Morover we want a finite procedure to determine if w is reducible w.r.t. $\mathcal{R}$ and (if yes) produce a rule $r \in \mathcal{R}$ which can be applied to w to rewrite it.

<u>Defn</u>: An Automaton over alphabet X is a labeled, directed graph together with two subsets of its vertices: A and Ω.

It's a tuple with

- Σ — the set of vertices (states)
- $L = X \cup \{\varepsilon\}$ — the set of labels
- $E \subset \Sigma \times L \times \Sigma$ — the set of labeled (transitions) edges
- $A \subset \Sigma$ — the set of initial states
- $\Omega \subset \Sigma$ — — " — final —

Ex:



The main aim of automata is to trace.

Defn: Let $\big((\sigma_1, x_1, \sigma_2), (\sigma_2, x_2, \sigma_3), \dots, (\sigma_{n-1}, x_{n-1}, \sigma_n)\big) =: P$ be a directed path in (Σ, E)

the signature $\operatorname{sign}(P)$ is defined to be

$$x_1 x_2 \dots x_{n-1} \in X^*.$$

We say that automaton $\mathcal{A} = (\Sigma, X, E, A, \Omega)$ accepts $w \in X^*$ iff there exist a path P in (Σ, E) s.t.

- $\operatorname{sign}(P) = w$
- $\sigma_1 \in A$
- $\sigma_n \in \Omega$

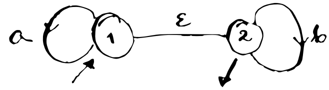
$L(\mathcal{A})$ - the language of an automaton is the set of all words in X^* accepted by $\mathcal{A}$

$$L(\mathcal{A}) = \{ \operatorname{sign}(P) : P - \text{path in } \mathcal{A} \text{ s.t. } \sigma_1 \in A \ \& \ \sigma_n \in \Omega \}$$

Defn / Thm:

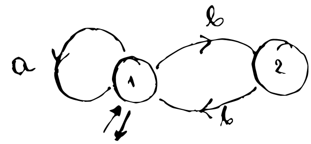
Language $L \subset X^*$ is rational iff there exist a finite automaton $\mathcal{A}$ s.t. $L = L(\mathcal{A})$.

$\mathcal{L}(\mathcal{A}) = \{a,b\}^*$



$\mathcal{L}(\mathcal{A}) = \{a^i b^j : i \geq 0 \quad j \geq 0\}$

$a^* b^*$



$(a^* bb)^* a^*$

<u>Defn</u>: $\mathcal{A}$ - automaton is <u>deterministic</u> iff

- $|A| \leq 1$ (at most one starting state)
- $E \subset \Sigma \times X \times \Sigma$ (no edge is labeled by ε)
- if $(\sigma, x, \tau_1), (\sigma, x, \tau_2) \in \Sigma \Rightarrow \tau_1 = \tau_2$
  (there is at most one edge starting at σ labeled x).

$\mathcal{A}$ is <u>complete</u> iff $\mathcal{A}$ is deterministic, $|A| = 1$,
$\forall \sigma \in \Sigma, x \in X \ \exists \tau \in \Sigma : (\sigma, x, \tau) \in E$.

<u>Proposition</u>: In a deterministic automaton a path is determined by its starting point & signature.

$\square$

ALGORITHM : trace

Input : • $\mathcal{A}$ - automaton (deterministic)
• w - word in X^*
• σ - the starting state // usually an initial state

Output : • k - the length of traced path
• τ - the end point

```
begin
  τ = σ
  for (i, l) in enumerate(w)
      if (τ, l, τ') ∈ A
          τ = τ'
      else
          return (i-1, τ)
      end
  end
  return length(w), τ   // we succesfully traced the whole w
end
```

Now it's straightforward to decide if $w \in \mathcal{L}(\mathcal{A})$:

If for any initial state $\sigma \in A$ we have

$$k, \tau = \mathrm{trace}(\mathcal{A}, w, \sigma) \text{ with}$$

• $k = \mathrm{length}(w)$ and
• $\tau \in \Omega$,

then $w \in \mathcal{L}(\mathcal{A})$

Index Automaton

$(R, <)$ - rws (reduced)

<u>Defn</u>: <u>Index automaton</u> is a complete automaton recognizing the language of words in X^* which are reducible w.r.t $(R, <)$.

<u>Note</u>: If A - index for $(R, <)$, P - path in A from the initial state to $\omega \in \Omega$.

then $W = \mathrm{sign}(P)$ is reducible w.r.t. $(R, <)$.

$\Rightarrow$ W contains as subword lhs for a rwrule in R

<u>Rule identifier</u> is a function

$$f : \Omega \to \mathrm{rwrules}(R)$$

$f(\omega) = A \to B \iff$ for every path $P = \alpha \rightsquigarrow \omega$ sign(P) contains A as subword

Algorithm : rewrite

Input : • W - word to be rewritten

• $\mathcal{A}$ - index automaton with rule identifier f.

Output : • V - rewritten W.

```
begin
  V = ε
  P = [initial_state(𝒜)]      // Path in 𝒜
  while !isone(W)
    x = popfirst!(W)
    σ = last(P)^x              // last(P) --x--> σ
                               //   is an edge in 𝒜
    if !isfinal(σ)
      push!(P, σ)
      push!(V, x)
    else
      A → B = f(σ)             // σ is final
      resize!(V, length(V) - Length(A) + 1)
      resize!(P, length(P) - Length(A) + 1)
      prepend!(W, B)
    end
  end
  return V
end
```

Notes : • for every $i \geq 0$ $\underbrace{\mathrm{sign}(P[1:i+1])}_{\text{well defined!}} = V[1:i]$

• V is always irreducible w.r.t. R.

• If $\sigma \in \Omega$ then V_x ends with <u>lhs</u> of $f(\sigma)$.

Constructing Index Automaton

$\mathcal{L} = \{$lhses of rules from $\mathcal{R}\}$

$\Sigma = \{$prefixes of elements from $\mathcal{L}\}$

Edges:

$E_1 = \{ (L, x, L) : L \in \mathcal{L}, x \in X \}$

(loops on the final states)

$E_3 = \{ (u, x, ux) : u \in \Sigma \setminus \mathcal{L},\ ux \in \Sigma \}$

(direct paths)

$E_2 = \{ (u, x, v) : u \in \Sigma \setminus \mathcal{L},\ ux \notin \Sigma,$
v - the longest suffix of ux which $\in \Sigma \}$

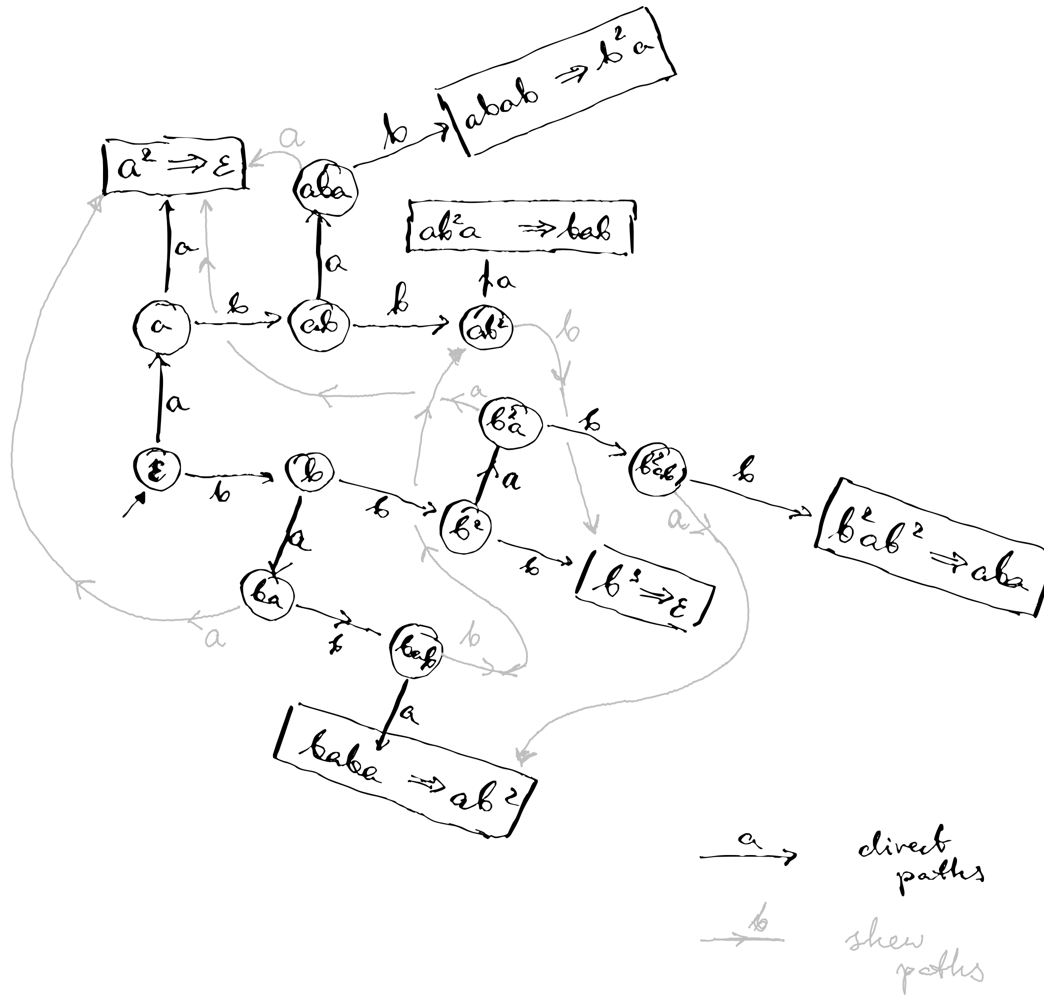
(skew paths)

$$\mathrm{I}(\mathcal{R}) = \mathcal{A}(\Sigma, X, E_1 \cup E_2 \cup E_3, \{\varepsilon\}, \mathcal{L})$$

If $(\mathcal{R}, <)$ is <u>reduced</u> $\Rightarrow A \in \mathcal{L}$ determines uniquely rwrule $A \to B \in \mathcal{R}$

rule identifier: $f(A) = A \to B$.

$$
R \begin{cases}
a^2 \to \varepsilon \\
b^3 \to \varepsilon \\
abab \to b^2a \\
ab^2a \to bab \\
baba \to ab^2 \\
b^2ab^2 \to aba
\end{cases}
$$

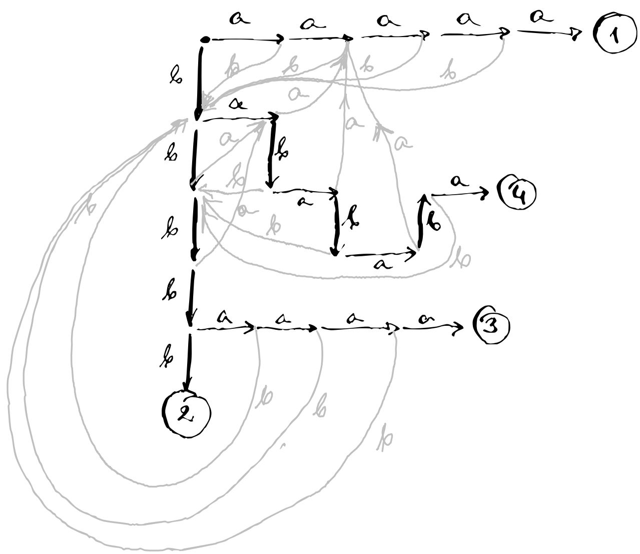


1) $a^5 \to \varepsilon$

2) $b^5 \to \varepsilon$

3) $b^4 a^4 \to (ab)^4$

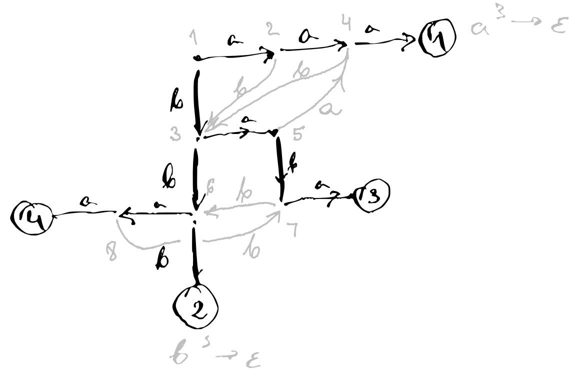
4) $(ba)^4 \to a^4 b^4$



1, 2 $a^3 = b^3 = \varepsilon$

3 $baba \to a^2 b^2$

4 $b^2 a^2 \to abab$



Algorithm : isconfluent

Input : • $(R,<)$ – reduced rws
• $\mathcal{A}$ – index automaton for $(R,<)$ with rule identifier f

Output : true/false + witness of failure

```
begin
  P = [ ]   // empty path
  UE = [ ]  // unexplored edges
  for (lhs → rhs) in rules(R)
      S = lhs[2:end]   // proper suffix
      push!(P, trace(A, S)[2])
      backtrack = false
      while !isempty(P) && !backtrack
          if isterminal(P[end])   // ....
              // process the overlap of (lhs → rhs) and f(P[end])
              // return if failure
              backtrack = true
          end
          if !backtrack
              push!(UE, alphabet(R))
              l = pop!(last(UE))
              push!(P, trace(A, l, P[end]))
          end
          while backtrack
              if !isempty(last(UE))
                  l = pop!(last(UE))
                  P[end] = trace(A, l, P[end-1])
                  backtrack = false
              else
                  pop!(UE)
                  pop!(P)
              end
          end
      end
  end
  return true
end
```

check if the path really defines A ≥ B

we've reached the end of a path time to backtrack:

if we're not backtracking, try to extend the path

we're backtracking: either take the next branch, or we're done with the branches, so move up!

<u>Note</u>: Since Index Automaton may contain directed loops this backtrack may not finish!

What is an additional condition that should put us in the backtrack mode?

(we're only looking for completions which are of the same length as their signature)

Problems with using index automata in Knuth-Bendix completion:

- the language $\mathcal{L} = \mathcal{L}(A)$ constantly changes so we need to keep A in sync
- for large rwses rebuilding index from scratch is expensive.
- is it easy to add a new rule?
- it's <u>hard</u> to remove one (note: to keep the size of the node constant we <u>don't</u> want to store in-edges and that makes this modification <u>hard</u>).
- it's way easier to rebuild the automaton using the datastructures for nodes already present in A.

How: → project!

Theorem (Higman, 1951)

Let $G = \mathrm{Grp}\langle r, s, t \mid t^{-1} r t r^{2} = r^{-1} s r s^{2} = s^{-1} t s t^{2} = 1 \rangle$

G is trivial.

Substitute:

$R_1 = t^{-1} r t r^{2}$

$S_1 = r^{-1} s r s^{2}$

$T_1 = s^{-1} t s r^{2}$ and form

$$G_2 = \mathrm{Grp}\langle r, s, t \mid T_1^{-1} R_1 T_1^{-1} R_1^{2} = R_1^{-1} S_1 R S_1^{2} = S_1^{-1} T_1 S_1 T_1^{2} = 1 \rangle$$

Theorem:

G_2 is trivial

Knuth-Bendix proves this by limiting the length of confluence conflicting words to 26.

Other uses of automata: - prove infiniteness.

If $\mathcal{L} = \mathcal{L}(\mathcal{A})$ is a rational language, then $X^* - \mathcal{L}$ is rational as well.

Consider $\mathcal{I}(\mathcal{R})$ - index automaton for rws $(\mathcal{R}, <)$.

$\mathcal{L}(\mathcal{I}(\mathcal{R}))$ - the set of words in X^* reducible w.r.t. $\mathcal{R}$

if $\mathcal{R}$ - confluent & reduced

$\mathcal{C} = X^* - \mathcal{L}(\mathcal{I}(\mathcal{R}))$ the set of canonical forms

$\mathcal{C} \xleftrightarrow{1-1}$ monoid elements

Corollary: If $X^* - \mathcal{L}(\mathcal{I}(\mathcal{R}))$ is infinite, so is M - monoid presented by $(\mathcal{R}, <)$.

Trim Automata:

$\mathcal{A} = (\Sigma, X, E, A, \Omega)$

Defn: • $\sigma \in \Sigma$ is accessible iff there exist a path $P \subset \mathcal{A}$, $first(P) \in A$, $last(P) = \sigma$.

• $\sigma \in \Sigma$ is coaccessible iff there exist a path $P \subset \mathcal{A}$, $first(P) = \sigma$, $last(P) \in \Omega$.

• $\sigma \in \Sigma$ is trim iff it's both accessible and coaccessible.

Let $\Sigma_t = \{\sigma \in \Sigma : \sigma \text{ is trim}\}$

We call $\mathcal{A}_t = (\Sigma_t', X, E', A', \Omega')$ the restriction of $\mathcal{A}$ to Σ_t.

Proposition: $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_t)$.

Proposition: Let $\mathcal{A}$ be trim.

- $\mathcal{L}(\mathcal{A}) = \emptyset$ iff the set of states is empty.
- $\mathcal{L}(\mathcal{A}) \neq \{\varepsilon\}$ iff $\mathcal{A}$ has a non-trivial label on one of its edges.

Proof: • If there are trim states in $\mathcal{A}$ then there are paths from A to Ω and their signatures are in $\mathcal{L}(\mathcal{A})$.

• If $e = (\sigma, x, \tau)$ belongs to E then there exists a path from A to Ω containing e
$\Rightarrow$ its sign $\neq \varepsilon \Rightarrow \mathcal{L}(\mathcal{A}) \neq \{\varepsilon\}$.

<u>Corollary:</u> Given an automaton we can decide whether $\mathcal{L}(\mathcal{A})$ contains a non-empty word.

$\mathcal{A} \rightsquigarrow \mathcal{A}_t \rightsquigarrow$ find an edge with non-trivial label.

<u>Proposition:</u> Let $\mathcal{A}$ - finite automaton. $\mathcal{L}(\mathcal{A})$ is infinite iff $\mathcal{A}_t$ contains a directed loop with non-trivial signature.

<u>Proof:</u> we can replace $\mathcal{A}$ by $\mathcal{A}_t$ without changing the language.

($\Leftarrow$) let $C =$  directed loop on σ.

Since σ - trim : $\exists\, P \quad \alpha \rightsquigarrow \sigma$ for some $\alpha \in A$

$\exists\, Q \quad \sigma \rightsquigarrow \omega$ for some $\omega \in \Omega$

then $\forall n$ paths $P\,C^n Q$ lead from α to ω and produce distinct words $\mathrm{sign}(P)\cdot\mathrm{sign}(C)^n\cdot\mathrm{sign}(Q)$ in $\mathcal{L}(\mathcal{A})$.

($\Rightarrow$) let $n = |E|$ and pick $w \in \mathcal{L}(\mathcal{A})$ s.t. $\mathrm{length}(w) > n$. Let P be the path from $\alpha \in A$ to $\omega \in \Omega$ with $\mathrm{sign}(P) = w$. Some edge $e = (\sigma, u, \tau)$ will occur on P more than once. Write $P = P'\,C\,Q$ where C begins with the first occurence of e and Q begins with the next one. Then C is a directed loop in $\mathcal{A}$.

Corollary: Given $\mathcal{A}$ - f.s.a. it's possible to decide whether $\mathcal{L}(\mathcal{A})$ is infinite.

Proof: replace $\mathcal{A}$ by $\mathcal{A}_+$ if necessary.

let $\mathcal{A} = (\Sigma, X, E, A, \Omega)$. for every $\sigma \in \Sigma$

we can decide whether $\mathcal{A}_\sigma = (\Sigma, X, E, \{\sigma\}, \{\sigma\})$

contains a non-trivial word.

$\Rightarrow$ we can decide whether $\mathcal{A}$ contains a directed loop with non-trivial signature.

$\square$

DFA
(trim)
deterministic automaton

automaton
NFA

(accessible) subset construction

additional fail non-final state

complete automaton (trim)
CFA

Constructions of automata:

Let

$\mathcal{A}_1 = (\Sigma_1, X, E_1, A_1, \Omega_1)$
$\mathcal{A}_2 = (\Sigma_2, X, E_2, A_2, \Omega_2)$
be two finite automata (labeled by the same alphabet.)

Let $\mathcal{L}_1 = \mathcal{L}(\mathcal{A}_1)$, $\mathcal{L}_2 = \mathcal{L}(\mathcal{A}_2)$.

Theorem:

$\mathcal{L}_1 \cup \mathcal{L}_2$, $\mathcal{L}_1 \cap \mathcal{L}_2$, $X^* - \mathcal{L}_1$, $\mathcal{L}_1\mathcal{L}_2$, $(\mathcal{L}_1)^*$

are rational languages.

Proof: We'll construct automata recognizing each of these languages.

Assumption: $\Sigma_1 \cap \Sigma_2 = \emptyset$

1) $\mathcal{A}^{\cup} = (\Sigma_1 \cup \Sigma_2, X, E_1 \cup E_2, A_1 \cup A_2, \Omega_1 \cup \Omega_2)$

recognizes $\mathcal{L}_1 \cup \mathcal{L}_2$

Note: $\mathcal{A}^{\cup}$ is not deterministic

2) $\mathcal{A}^{\cap} = (\Sigma_1 \times \Sigma_2, X, E^{\cap}, A_1 \times A_2, \Omega_1 \times \Omega_2)$

$$E^{\cap} = \{((\sigma_1, \sigma_2), x, (\tau_1, \tau_2)) : (\sigma_1, x, \tau_1) \in E_1 \ \& \ (\sigma_2, x, \tau_2) \in E_2\}$$

if $P \subset \mathcal{A}_\cap$ - a path from (α_1, α_2) to (ω_1, ω_2)

$\Rightarrow$ $proj_1(P)$ - path $\alpha_1 \rightsquigarrow \omega_1$ in $\mathcal{A}_1$

$proj_2(P)$ — " — $\alpha_2 \rightsquigarrow \omega_2$ in $\mathcal{A}_2$

$sign(P) = sign(proj_i(P))$

$\Rightarrow \mathcal{A}_\cap$ recognizes $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Note: only accessible part of $\mathcal{A}_\cap$ should be constructed.

If both $\mathcal{A}_1, \mathcal{A}_2$ are deterministic so is $\mathcal{A}_\cap$.

3) $\mathcal{A}_1^-$

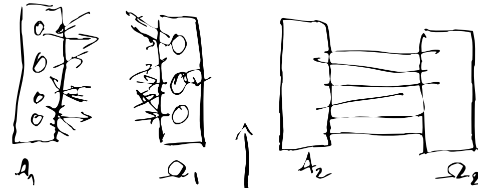
$\mathcal{A}_1 \rightsquigarrow \mathcal{A}_1^c$ (complete with the same language)

then $\mathcal{A}_1^- = (\Sigma_1^c, X, E_1^c, A_1^c, \Sigma_1^c - \Omega_1^c)$

recognizes $X^* - \mathcal{L}_1$

for $w \in X^*$ $\to$ unique path $P \in \mathcal{A}_1^c$, let $\sigma = last(P)$

$w \in \mathcal{L}_1 \Leftrightarrow \sigma \in \Omega_1^c$, $w \notin \mathcal{L}_1 \Leftrightarrow \sigma \in \Sigma_1^c - \Omega_1^c$.

4) 

E_0 = add all possible edges here, all labeled by ε.

$\mathcal{A}^{1\cdot 2} = (\Sigma_1 \cup \Sigma_2, X, E_1 \cup E_2 \cup E_0, A_1, \Omega_2)$ recognizes $\mathcal{L}_1 \mathcal{L}_2$.

5) $\mathcal{A}^* =$ 

recognizes $(\mathcal{L}_1)^*$

<u>Corollary:</u> Let $M = \langle X \mid R \rangle$ be a f.p. monoid.

Suppose that $S = RC(X, R, <)$ (reduced, confluent rewriting system) is finite w.r.t. wordering $<$.

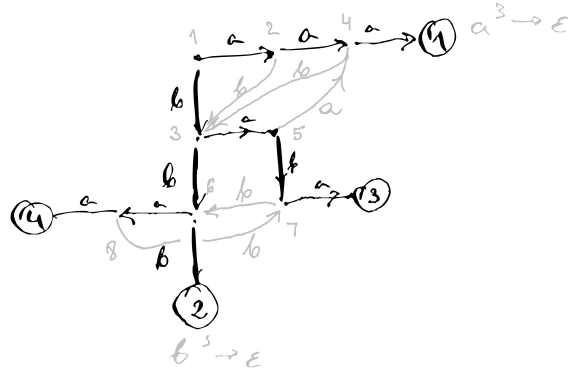
- $\mathcal{I}$ - the ideal of words in X^* reducible w.r.t. the rws.
- $X^* - \mathcal{I}$ - irreducible words $\overset{1-1}{\longleftrightarrow}$ canonical forms $\longleftrightarrow$ elements of M.
- $\mathcal{I}(S)$ - index automaton for S recognizes $\mathcal{I}$
- $(\mathcal{I}(S))^c$ recognizes $X^* - \mathcal{I}$

Thus M is infinite iff $(\mathcal{I}(S))^c$ contains a directed cycle with non-trivial signature.

$a^3 = b^3 = \varepsilon$

$baba \to a^2 b^2$

$b^2 a^2 \to abab$



$\mathcal{B}_1 = \mathcal{A}$

$\mathcal{B}_2 = \{ \sigma \in \mathcal{B}_1 : \exists (\tau, l, \sigma) : \tau \in \mathcal{B}_1 \}$

$= \{ 2, 3, 4, 5, 6, 7, 8 \}$

$\mathcal{B}_3 = \{ \sigma \in \mathcal{B}_2 : \exists (\tau, l, \sigma) : \tau \in \mathcal{B}_2 \}$

$= \{ 3, \ldots, 8 \}$

$\mathcal{B}_4 = \{ \sigma \in \mathcal{B}_3 : \exists (\tau, l, \sigma) : \tau \in \mathcal{B}_3 \}$

$= \{ 3, \ldots, 8 \}$

$\vdots$

Theorem: let $\mathcal{A}$ be trim;

- $\exists k : \mathcal{B}_{k+1} = \mathcal{B}_k$;
- $\mathcal{L}(\mathcal{A})$ is infinite if $\mathcal{B}_k = \mathcal{B}_{k+1} = \ldots$ has an edge with a non-trivial label.