Kompression von Zeichenreihen

Wir betrachten nur Komprimierungen

$$compress : \Sigma^* \longrightarrow \Delta^*$$

die man ohne Informationsverlust rückgängig machen kann:

$$decompress(compress(w)) = w.$$

Kompressionsrate: $\frac{|w|}{|compress(w)|}$

(i) Statistische Methoden: berechne das in einem Kontext erwartete Zeichen und kodiere das tatsächliche durch die Differenz zum erwarteten

(ii) Wörterbuchmethoden: ersetze *Blöcke* B (= Teilwörter von w beschränkter Länge) durch Codes $c(B) \in \Delta$ aus einem Wörterbuch

$$Dict = \{(B_i, c(B_i)) \mid i = 1, \ldots, k\}.$$

(a) *statisches Wörterbuch*: *Dict* ist unabhängig von w

(b) *dynamisches Wörterbuch*: *Dict* wird während der Kompression von w berechnet.

Finde deterministisch eine Zerlegung $w = B_{i_1} \cdots B_{i_n} \in Dict^*$ mit „kurzem“ $c(w)$ für

$$c(w) := c(B_{i_1}) \cdots c(B_{i_n}).$$

Beispiel: Ziv-Lempel-Kompression (Unix: compress, gzip)

KOMPRESSION LEMPEL-ZIV-77

Parameter Fenstergröße N (≤ 8192), Puffergröße F (≤ 20)

Eingabe: ein Text t über dem Alphabet Σ

Ausgabe: eine Folge von Tripeln $(i, j, c) \in (N - F) \times F \times \Sigma$

(i) ein *Fenster* der Länge N gleitet durch den Text t,

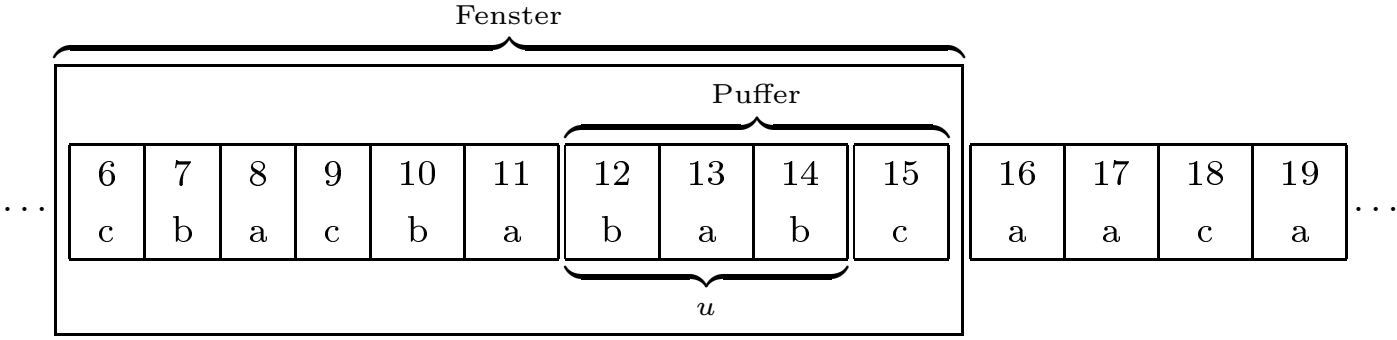
(ii) die ersten $N - F$ Zeichen im Fenster sind behandelt, die letzten F Zeichen bilden die *Vorausschau* (Puffer),

(iii) suche das längste echte Präfix u des Puffers, das an einer Fensterposition $N - F - i$ beginnt (ggf. im Puffer endet),

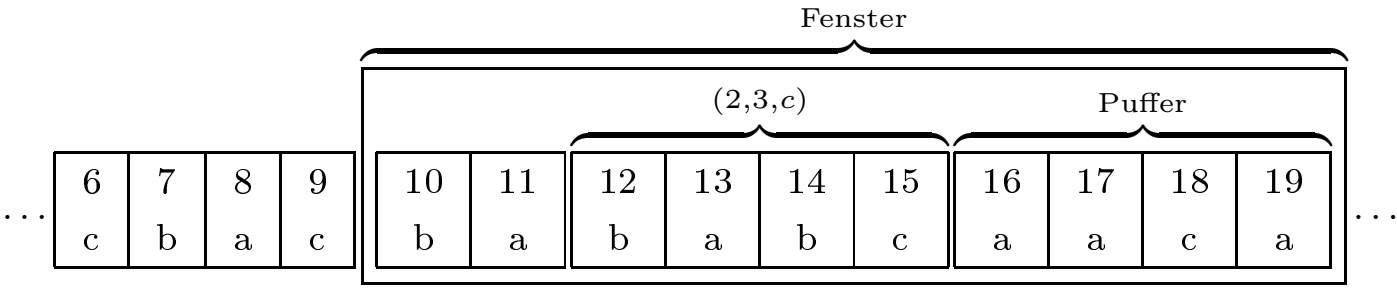
(iv) Kodiere u durch $(i, |u|, a)$, wobei ua ein Pufferpräfix ist, das nicht an der Fensterposition $N - F - i$ vorkommt,

(v) verschiebe das Fenster um $|u| + 1$ Zeichen

Expl. 3 $N = 10, F = 4$



Komprimiere $uc = babc$ durch $(2, 3, c)$ and betrachte dann:



(i) Fenster und Puffer beschränken die Suche und die Größe der „Zeiger“ i und $|u|$ in $c(B) = c(ua) = (i, |u|, a)$.

(ii) Wörterbuch: Teilwörter der Länge $1 \leq k \leq F$ des Präfixes der Länge $N - 2$ des Fensterinhalts.

Komprimieren

(i) Zustand:

$$\mathit{Codes}\boxed{\mathit{Fenster}}\mathit{Resteingabe} = p_0 \cdots p_k \boxed{c_6 \ldots c_1.b_1 \ldots b_4}\, v$$

(ii) Kodiere $(i, |u|, a)$, mit $ua =$ längstes Präfix des Puffers, wovon u bei Offset $i > 0$ vor dem Puffer beginnt.

(iii) Verschiebe das Fenster um $|ua|$ Positionen.

Beisp. 4 *LZ*-77 mit $N = 10, F = 4$, $w = a^{15}$

$$\begin{aligned}
\boxed{\epsilon^6.\epsilon^4}\,a^{15} &\Rightarrow \boxed{\epsilon^6.a^4}\,a^{11} \\
&\Rightarrow (1,0,a)\boxed{\epsilon^5 a.a^4}\,a^{10} \\
&\Rightarrow (1,0,a)(1,3,a)\boxed{\epsilon a^5.a^4}\,a^6 \\
&\Rightarrow (1,0,a)(1,3,a)(1,3,a)\boxed{a^6.a^4}\,a^2 \\
&\Rightarrow (1,0,a)(1,3,a)(1,3,a)(1,3,a)\boxed{a^6.a^2} \\
&\Rightarrow (1,0,a)(1,3,a)(1,3,a)(1,3,a)(1,1,a)
\end{aligned}$$

Beachte: a^F wird zu $(1,0,a)(1,F-1,a)$ kodiert, also exponentiell kürzer bei log.Darstellung von $F - 1$.

Dekomprimieren

Zustand: *Dekodiert* (i,m,a) *Codes* $= u(i,m,a)\cdots(i_k,m_k,a_k)$:

(i) i Schritte zurück in u gehen;

(ii) von dort m Buchstaben abschreiben, dann a;

(iii) restliche *Codes* $= \cdots(i_k,m_k,a_k)$ dekodieren.

Beisp. 3 *LZ*-77 mit $N=10, F=4$, $w=a^{15}$

$$
\begin{aligned}
&(1,0,a)(1,3,a)(1,3,a)(1,3,a)(1,1,a)\\
\Rightarrow\quad & a.(1,3,a)(1,3,a)(1,3,a)(1,1,a)\\
\Rightarrow\quad & a.aaaa.(1,3,a)(1,3,a)(1,1,a)\\
\Rightarrow\quad & a.aaaa.aaaa.(1,3,a)(1,1,a)\\
\Rightarrow\quad & a.aaaa.aaaa.aaaa.(1,1,a)\\
\Rightarrow\quad & a.aaaa.aaaa.aaaa.aa
\end{aligned}
$$

Beisp. 4 *LZ*-77-code $(N = 10, F = 4)$ of $w = abbbaabbabbb$:

	B_0	B_1	B_2	B_3	B_4
$w =$	a	b	bba	abb	$abbb$
$LZ\text{-}77(w) =$	$(1,0,a)$	$(1,0,b)$	$(1,2,a)$	$(5,3,a)$	$(3,3,b)$

$$
\begin{aligned}
& \boxed{\epsilon^6.\epsilon^4}\, abbbaabbabbb \\
\Rightarrow\quad & \boxed{\epsilon^6.\underline{a}bbb}\, aabbabbb \\
\Rightarrow\quad & (1,0,a)\boxed{\epsilon^5 a.\underline{b}bba}\, abbabbb \\
\Rightarrow\quad & (1,0,a)(1,0,b)\boxed{\epsilon^4 ab.\underline{bb}aa}\, bbabbb \\
\Rightarrow\quad & (1,0,a)(1,0,b)(1,2,a)\boxed{\epsilon abbba.\underline{abb}a}\, bbb \\
\Rightarrow\quad & (1,0,a)(1,0,b)(1,2,a)(5,3,a)\boxed{baabba.\underline{bbb}\epsilon} \\
\Rightarrow\quad & (1,0,a)(1,0,b)(1,2,a)(5,3,a)(3,3,b)\boxed{babbb\epsilon.\epsilon^4}
\end{aligned}
$$

LEMPEL-ZIV-78 COMPRESSION

Die *LZ-Blockzerlegung von* $w \in \Sigma^+$ ist die Folge $B_0, \ldots, B_m$ von Teilwörtern, so daß

(i) $B_0 :=$ der Anfangsbuchstabe von w,

(ii) Ist $w = B_0 \cdots B_{n-1} v$ für ein $v \in \Sigma^+$, so ist

$$B_n := \begin{cases} \text{kürzestes Präfix} \neq \epsilon \text{ von } v, \text{ das} \\ \text{nicht in } \{B_0, \ldots, B_{n-1}\} \text{ ist,} & \text{falls vorhanden,} \\ v & \text{sonst.} \end{cases}$$

Bem: Die Menge der *LZ*-Blöcke von w ist präfixabgeschlossen.

Der Block B_n wird durch seinen Endbuchstaben a und einen Zeiger k auf den Präfixblock B_{k-1} dargestellt.

Die *LZ-78-Komprimierung* $LZ(w)$ von w ist die Folge $p_0 \cdots p_{m-1}$ der Blockdarstellungen $p_n = (k, a) \in \mathbb{N} \times \Sigma$ so daß

$$\begin{aligned} p_n = (0, a) &\iff B_n = a, \\ p_n = (j+1, a) &\iff B_n = B_j a. \end{aligned}$$

Beisp. 5 *LZ*-78-Komprimierung von $w = a^{15}$:

	B_0	B_1	B_2	B_3	B_4
$w =$	a	aa	aaa	$aaaa$	$aaaaa$
$LZ(w) =$	$(0,a)$	$(1,a)$	$(2,a)$	$(3,a)$	$(4,a).$

Beisp. 6 *LZ*-78-Komprimierung von $w = abbbaabbabbb$:

	B_0	B_1	B_2	B_3	B_4	B_5
$w =$	a	b	bb	aa	bba	bbb
$LZ(w) =$	$(0,a)$	$(0,b)$	$(2,b)$	$(1,a)$	$(3,a)$	$(3,b).$

Baumdarstellung der Blockfolge

Während der Komprimierung werden die Blöcke $B_0, B_1, \ldots \in \Sigma^+$ in einem Baum gespeichert, wobei gemeinsame Präfixe nur einmal dargestellt werden. Der nächste Block ist das erste Präfix der Resteingabe, das keinen Pfad im Baum markiert.

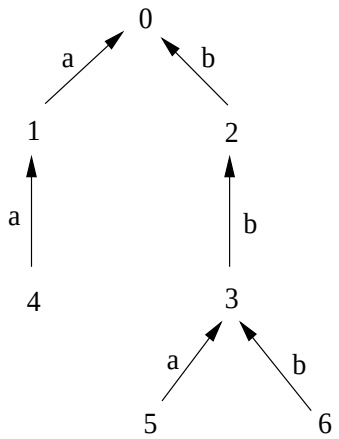
Die lineare Ordnung der Blöcke von $LZ(w)$ entspricht einer *Numerierung* der Baumknoten: Block B_k markiert den Weg von der Wurzel 0 zum Knoten $k+1$.

Beisp. 6 LZ-78-Komprimierung von $w = abbbaabbabbb$:

	B_0	B_1	B_2	B_3	B_4	B_5
$w =$	a	b	bb	aa	bba	bbb
$LZ(w) =$	$(0,a)$	$(0,b)$	$(2,b)$	$(1,a)$	$(3,a)$	$(3,b).$

Die Aufzählung der Paare von $LZ(w)$ durch Anhängen einer Nummer $1, 2, \ldots$ ergibt den Baum

$$LZ'(w) = (0,a,1)(0,b,2)(2,b,3)(1,a,4)(3,a,5)(3,b,6),$$



Das Tupel $p_n = (k,a)$ von $LZ(w)$ ist die Kante $k \xleftarrow{a} n+1$.

LZ-78-Tries

Dieser Baum ist ein Σ-*trie*, d.h. verschiedene von einem Knoten ausgehende Zweige sind mit verschiedenen Buchstaben markiert (falls w ein besonderes Endsymbol hat).

Def. 7 Ein (kantenmarkierter) Σ-Baum $(T, \leq, \langle \xleftarrow{a} \rangle_{a\in\Sigma}, 0)$ ist ein Σ-*Trie*, wenn es zu jedem Knoten $n \in T$ und jedem $a \in \Sigma$ höchstens ein $n' \in T$ mit $n \xleftarrow{a} n'$ gibt. Ein *numerierter* Σ-*Trie*

$$\mathcal{T} = (T, \leq, \langle \xleftarrow{a} \rangle_{a\in\Sigma}, 0, succ),$$

oder *LZ-78-Trie*, ist ein Σ-Trie mit einer Relation *succ* auf T, die mit der Baumordnung $\leq$ verträglich ist. OBdA sei $T = \{0, 1, 2, \ldots, m\}$ und $succ(i, j) \iff i + 1 = j$.

Prop. 8 Man kann $LZ(w)$ in $O(|w|)$ Schritten berechnen.

Bew. Man findet den nächsten Block $B_n = B_j a$ der Resteingabe, indem man buchstabenweise im Trie $\mathcal{T}_{n-1}$ absteigt, bis man vom zuletzt besuchten Knoten k eine a-Kante zu einem neuen Knoten $n + 1$ einfügen muß; dann gibt man die Blockdarstellung $p_n = (k, a)$ aus. Für jeden Buchstaben der Eingabe macht das einen festen Aufwand.

Die Dekomprimierung geht ebenfalls in Linearzeit (abhängig von der Trie-Größe).

LEMPEL-ZIV-WELCH COMPRESSION

LZW-block decomposition

For $w \in \Sigma^+$, the *LZW-block decomposition of* w is the sequence $B_0, \ldots, B_{m-1}$ of subwords such that

(i) T_0 is the trie with root 0 and branches to nodes $1, \ldots, q$ labeled $a_1, \ldots, a_q$, where $\Sigma = \{a_1, \ldots, a_q\}$.

(ii) If trie T_n and subwords $B_0, \ldots, B_{n-1}$ are constructed such that $w = B_0 \cdots B_{n-1} v$ for some $v \in \Sigma^+$, then

$$B_n := \left\{ \begin{array}{l} \text{longest non-empty prefix of } v \\ \text{labeling a path in } T_n \end{array} \right.$$

If $v \neq B_n$, then

$$T_{n+1} := \left\{ \begin{array}{l} T_n \text{ extended by a branch from the} \\ \text{node reached by } B_n \text{ to a new node } |T_n|, \\ \text{labeled with the first letter of } v \text{ after } B_n. \end{array} \right.$$

LZW-compression (pointers only!)

If $w = B_0 \cdots B_{m-1}$ is the *LZW* block decomposition, then

$$LZW(w) = p_0 \cdots p_{m-1}$$

where p_i is the node reached in T_{m-1} by the path labeled B_i.

LZW-Decompression

(i) T_0 is known from Σ;

(ii) if T_n and $B_0 \cdots B_{n-1} p_n \cdots p_{m-1}$ are reconstructed:

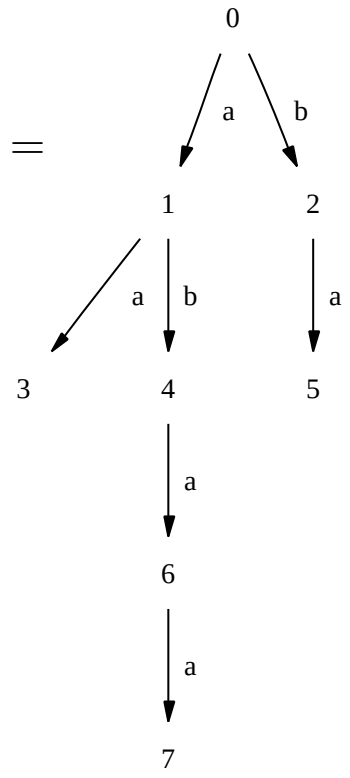
(iii) get B_n by reading the path from 0 to p_n in T_n;

(iv) get T_{n+1} by adding an edge from p_n to new node $|T_n|$, labeled by the first letter of B_{n+1}, which is the first label on the path to node $p_n \in T_n$, if $p_{n+1} \notin T_n$, else to p_{n+1}.

Beisp. 9 *LZW*-compression of $w = aababababaaa$:

	B_0	B_1	B_2	B_3	B_4	B_5
$w =$	a	a	b	ab	aba	aa
$LZW(w) =$	1	1	2	4	6	3

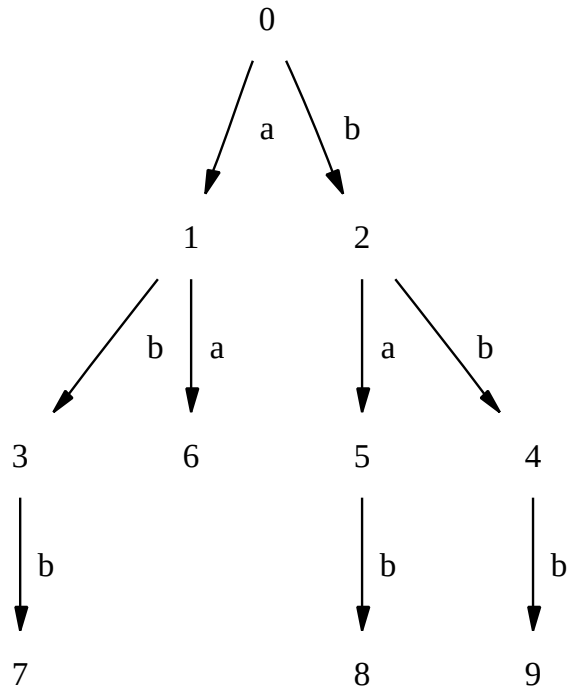
$T =$



$T_n = T \restriction \{k \mid k \leq n+2\}$

Beisp. 10 *LZW*-compression of $w = abbaabbabbb$:

	B_0	B_1	B_2	B_3	B_4	B_5	B_6	B_7
$w =$	a	b	b	a	ab	ba	bb	b
$LZW(w) =$	1	2	2	1	3	5	4	2

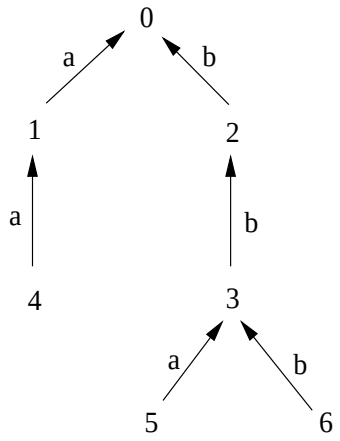


Suche in *LZ*-78-komprimierten Dateien

Erinnerung: die *LZ*-78-Komprimierung macht aus $w \in \Sigma^+$ einen numerierten Σ-Trie. Für $w = abbbaabbabbb$:

	B_0	B_1	B_2	B_3	B_4	B_5
$w =$	a	b	bb	aa	bba	bbb
$LZ(w) =$	$(0,a)$	$(0,b)$	$(2,b)$	$(1,a)$	$(3,a)$	$(3,b).$

Das Tupel $p_n = (k, a)$ von $LZ(w)$ ist die Kante $k \xleftarrow{a} n+1$.



Problem 1 Welche Eigenschaften von Strings $w = a_1 \cdots a_n \in \Sigma^+$ kann man am Trie $LZ(w)$ ablesen?

Satz 2 (Büchi) Für $L \subseteq \Sigma^+$ sind äquivalent:

(i) L ist regulär

(ii) L ist durch eine Aussage φ in zweitstufiger Logik mit Quantifizierung über Mengen definierbar, genauer:

$$L = \{w \in \Sigma^+ \mid \mathcal{O}_w \models \varphi\}, \quad \text{wobei}$$

$\mathcal{O}_w = (\{1, \ldots, |w|\}, \langle U_a \mid a \in \Sigma\rangle, \leq)$ mit $U_a = \{i \mid a_i = a\}$.

Für $w = a_1 \cdots a_n$ ist $\mathcal{O}_w$ die lineare Ordnung $1 \leq \ldots \leq n$ mit einer Zerlegung der Punktmenge nach „Farben“ U_a, $a \in \Sigma$.

In den Formeln φ können wir $\max, \min, +1, -1$ benutzen.

Beisp. 3 Die reguläre Sprache sei $L = a(bc)^*a$.

$$\begin{aligned}\varphi = \ & \text{„}U_a, U_b, U_c \text{ bilden eine Zerlegung“} \wedge U_a(\min) \wedge U_a(\max) \wedge \\ & \exists X(\forall i < \max(i \in X \vee i+1 \in X) \wedge \\ & \quad \forall i \in X \; U_b(i) \wedge U_c(i+1))\end{aligned}$$

Proposition: Jede Aussage φ über gefärbte Ordnungen kann man in eine Aussage ψ über LZ-78-Tries übersetzen, sodaß

$$\mathcal{O}_w \models \varphi \iff \mathcal{T}_w \models \psi$$

Idee: Position i in $\mathcal{O}_w$ wird durch das Paar $(k,l) \in \mathcal{T}^2_{LZ(w)}$ kodiert, wobei i in B_l an der durch das Präfix B_k von B_l bestimmten Position liegt.

Aber: Quantifizierung über Mengen in $\mathcal{O}_w$ entspricht Quantifizierung über 2-stellige Relationen in $\mathcal{T}_{LZ(w)}$ – das ist zu teuer.

Frage 1 Kann man Eigenschaften ψ von LZ-78-Tries $\mathcal{T}$ durch geeignete Trie-Automaten $\mathcal{A}_\psi$ erkennen, sodaß also

$$\mathcal{T}_{LZ(w)} \models \psi \iff \mathcal{A}_\psi \text{ akzeptiert } \mathcal{T}_{LZ(w)} \; ?$$

Approximate notion of automata: m-*LZ*-78-trie-automata

The *hemisphere of radius m around n* in a graph (G, E) is

$$hs_m(n) := \{k \in G \mid E^{\leq m}(n, k)\}.$$

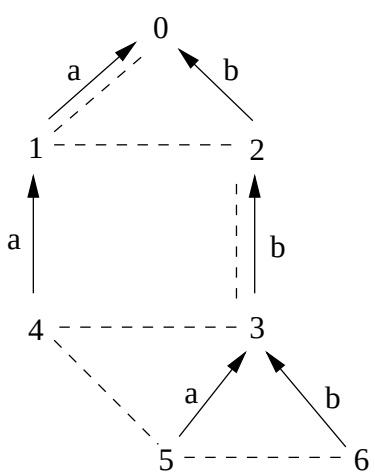
Def. 4 Let $\mathcal{T}$ be an *LZ*-78-trie and

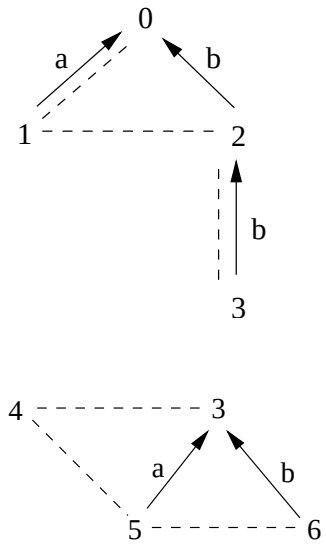
$$E \quad := \quad \bigcup \{ \xleftarrow{a} \mid a \in \Sigma\} \cup \{Succ\}.$$

The *bottom-up (resp. top-down) LZ-78-hemisphere of radius m around $n \in T$* is

$$\begin{aligned} bu\text{-}hs_m^{\mathcal{T}}(n) &:= \mathcal{T}\restriction_{hs_m(n)} \quad \text{in } (T, E), \\ td\text{-}hs_m^{\mathcal{T}}(n) &:= \mathcal{T}\restriction_{hs_m(n)} \quad \text{in } (T, \breve{E}). \end{aligned}$$

Expl. 5 (Cont.) The top-down resp. bottom-up 2-hemispheres of node 3 (with dashed edges for *Pred* resp. *Succ*):





m-*LZ*-78-**trie-automata**

Def. 6 A finite *bottom-up m-LZ-automaton* $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ over Σ consists of

- a finite set Q of *states*,
- sets $I, F \subseteq Q$ of *initial* and *final* states,
- a finite *transition relation* δ of pairs (P, q) (or: $P \to q$), where $q \in Q$ and P is a bottom-up *LZ*-78-hemisphere of radius m whose nodes except the root are labelled by elements of Q.

$\mathcal{A}$ is *deterministic* if $|I| = 1$ and $q = q'$ when $(P, q), (P, q') \in \delta$.

Then $r : T \to Q$ is an (*accepting*) *run* of $\mathcal{A}$ on *LZ*-78-trie $\mathcal{T}$ if

- $r(\max) \in I$ (and $r(0) \in F$),
- for each $n \in T$ there is some $(P, q) \in \delta$ such that $bu\text{-}hs_m^{\mathcal{T}}(n)$, expanded by the node-labelling r, is isomorphic to P with label q at its root,

The *class of LZ-78-tries accepted by* $\mathcal{A}$ is

$$L(\mathcal{A}) := \{\mathcal{T} \mid \mathcal{A} \text{ accepts } \mathcal{T}\}.$$

Similarly, define *top-down m-LZ-automata.*

Note: While $\mathcal{A}$ sequentially follows the enumeration of $\mathcal{T}_\alpha$, it can access some states reached at suffixes (resp. prefixes) of α.

Strictness of the automata hierarchy

The bottom-up resp. top-down hierarchies of m-LZ-automata are strict, and no level exhausts the MSO-properties of tries:

Thm. 1 For every m, there is an MSO-sentence defining a class of LZ-tries that is not accepted by any m-LZ-automaton.

Top-down LZ-trie-automata

G.Navarro (2003) has shown that finite automata on strings can be simulated on LZ-78-compressed strings. In fact:

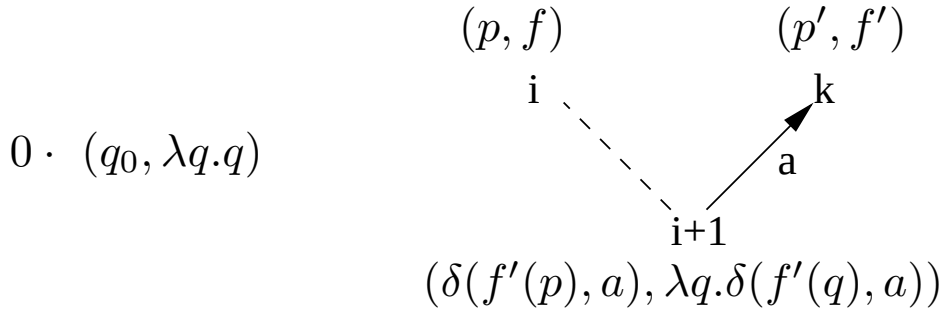
Thm. 2 The LZ-compression $\{\mathcal{T}_{LZ(w)} \mid w \in R\}$ of any regular set $R \subseteq \Sigma^+$ is accepted by a deterministic top-down 1-LZ-automaton.

Bew.: Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a DFA accepting R. Define a det. top-down 1-LZ-automaton $\mathcal{A}' = (Q', \Sigma, \delta', q'_{in}, F')$ by

(i) $Q' := Q \times (Q \to Q)$,

(ii) $q'_{in} := (q_0, \lambda q.q)$, $F' := F \times (Q \to Q)$,

(iii) δ' according to the following transitions:



for each $a \in \Sigma$ (including the case $i = k$).

Suppose r' is a run of $\mathcal{A}'$ on $\mathcal{T}_{LZ(w)}$ where $w = B_0 \cdots B_{m-1}$ are the LZ-blocks. By induction we see that for each $i < m$,

$$r'(i) \quad = \quad (\delta(q_0, B_0 \cdots B_{i-1}), \lambda p.\delta(p, B_{i-1})). \qquad (3)$$

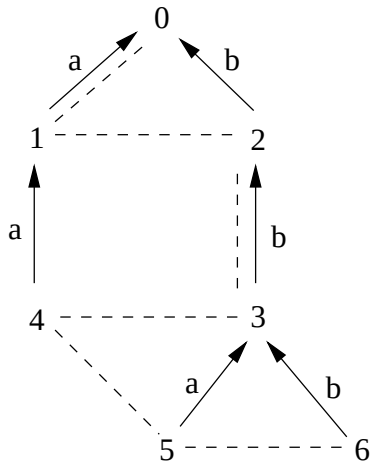
Hence $\mathcal{A}'$ accepts $\{\mathcal{T}_{LZ(w)} \mid w \in R\}$, because

$$w \in R \iff \delta(q_0, B_0 \cdots B_{m-1}) \in F \iff r'(m) \in F'.$$

□

Proof of (3)

Consider node 5 in our LZ-trie



Its top-down 1-hemisphere contains nodes 4 and 3. Recall that B_i labels the path from the root to node $i+1$. By induction,

$$\begin{array}{rcccc} r'(4) & = & (& \delta(q_0, B_0 \cdots B_3) & , \quad \ldots \quad), \\ r'(3) & = & (& \ldots & , \quad \lambda q.\delta(q, B_2) \quad). \end{array}$$

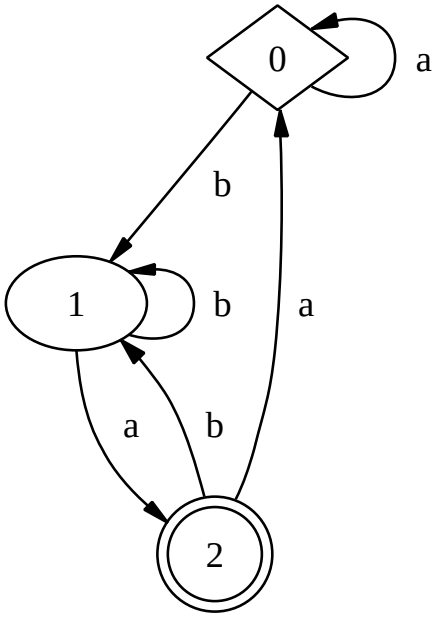
Note that B_4 labels the path to node 5, and $B_4 = B_2 a$, so

$$\begin{array}{rcl} r'_2(5) = \lambda q.\delta(q, B_4) & = & \lambda q.\delta(r'_2(3)(q), a), \\ r'_1(5) = \delta(q_0, B_0 \cdots B_4) & = & \delta(r'_1(4), B_4) = r'_2(5)(r'_1(4)) \end{array}$$

Beispiel: Suche ba in LZ-78$(abbbaabbabbb)$

Methode:

(i) Konstruiere für $L = \Sigma^* ba$ einen DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$:



Sei $\delta_c : Q \to Q$ die Funktion $\delta_c(q) := \delta(q, c)$, hier also:

$$\delta_a = \{(0,0), (1,2), (2,0)\}, \quad \delta_b = \{(0,1), (1,1), (2,1)\}.$$

(ii) Bilde den $\mathcal{A}$ simulierenden top-down 1-LZ-78-DFA $\mathcal{A}' = (Q', \Sigma, \delta', q_0', F')$ mit

(a) $Q' := Q \times (Q \to Q)$,

(b) $q_0' := (q_0, \lambda q : Q.q)$, $F' := \{q_2\} \times (Q \to Q)$,

(c) und als δ' die top-down-1-Umgebungen, für $c \in \Sigma$:

$$0 \cdot \ (q_0, \lambda q.q) \qquad \begin{matrix} (p, \_) & & (\_, f') \\ \text{i} & & \text{k} \\ & \searrow \quad \nearrow \text{c} & \\ & \text{i+1} & \\ (\delta_c \circ f'(p), \delta_c \circ f') & & \end{matrix} \tag{4}$$

Wegen $\delta_a = \{(0,0),(1,2),(2,0)\}$ ist

$$\delta_a \circ f' = \{(0,0),(1,0),(2,0)\} \tag{5}$$

außer für die Fälle von f' mit $1 \in \mathit{range}(f')$:

f'	$\delta_a \circ f'$
$(0,0),(1,0),(2,1)$	$(0,0),(1,0),(2,2)$
$(0,0),(1,1),(2,0)$	$(0,0),(1,2),(2,0)$
$(0,0),(1,1),(2,1)$	$(0,0),(1,2),(2,2)$
$(0,0),(1,1),(2,2)$	$(0,0),(1,2),(2,0)$
$(0,0),(1,2),(2,1)$	$(0,0),(1,0),(2,2)$
$(0,1),(1,0),(2,0)$	$(0,2),(1,0),(2,0)$
$(0,1),(1,0),(2,1)$	$(0,2),(1,0),(2,2)$
$(0,1),(1,0),(2,2)$	$(0,2),(1,0),(2,0)$
$(0,1),(1,1),(2,0)$	$(0,2),(1,2),(2,0)$
$(0,1),(1,1),(2,1)$	$(0,2),(1,2),(2,2)$
$(0,1),(1,1),(2,2)$	$(0,2),(1,2),(2,0)$
$(0,1),(1,2),(2,0)$	$(0,2),(1,0),(2,0)$
$(0,1),(1,2),(2,1)$	$(0,2),(1,0),(2,2)$
$(0,1),(1,2),(2,2)$	$(0,2),(1,0),(2,0)$

(6)

Da δ_b immer den Wert 1 liefert, ist für jedes f'

$$\delta_b \circ f' = \{(0,1),(1,1),(2,1)\} \tag{7}$$

Durch Einsetzen dieser Funktionen in (4) sind die 1-Umgebungen für die Transition δ' bestimmt.

(iii) Lies mit $\mathcal{A}'$ die komprimierte Folge $LZ\text{-}78(w)$.

$\mathcal{A}'$ bestimmt den Zustand bei $p_n = (k+1, a)$ mit a, dem Zustand bei p_{n-1} *und dem Zustand bei* p_k.

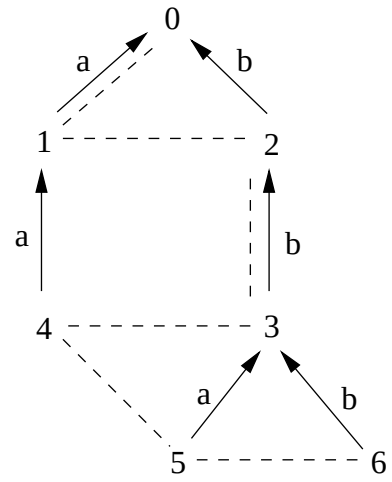
Suche von ba mit $\mathcal{A}'$ in LZ-78$(abbbaabbabbb)$

Für $w = abbbaabbabbb = a.b.bb.aa.bba.bbb$ war

$$\begin{aligned} LZ\text{-}78(w) &= (0,a)(0,b)(2,b)(1,a)(3,a)(3,b), \\ &= (0,a,1)(0,b,2)(2,b,3)(1,a,4)(3,a,5)(3,b,6), \end{aligned}$$

oder als LZ-Trie

$\mathcal{T} =$



Ein Lauf $r : T \to Q'$ von $\mathcal{A}'$ auf $\mathcal{T}$ beginnt bei der Wurzel 0, die mit dem Zustand

$$r(0) = q'_0 = (q_0, \lambda q.q) = (0, \lambda q.q)$$

markiert wird. Nach der Umgebung von Knoten 1 erhält er die Markierung

$$r(1) = (\delta_a \circ f'(p), \delta_a \circ f') = (\delta_a(0), \delta_a) = (0, \delta_a)$$

mit $r(0) = (p, \_)$ und $r(k) = (\_, f')$ für $k \xleftarrow{a} 1$

Analog geht es weiter:

- Da $2-1=1$ und $0 \xleftarrow{b} 2$ sind, erhält man aus

$$\begin{aligned} r(1) &= (p, \_) = (0, \_), \\ r(0) &= (\_, f') = (\_, \lambda q.q) \end{aligned}$$

$$r(2) = (\delta_b \circ f'(p), \delta_b \circ f') = (\delta_b(0), \delta_b) = (1, \delta_b)$$

- Da $3-1=2$ und $2 \xleftarrow{b} 3$ sind, erhält man aus

$$\begin{aligned} r(2) &= (p, \_) = (1, \_), \\ r(2) &= (\_, f') = (\_, \delta_b) \end{aligned}$$

$$r(3) = (\delta_b \circ f'(p), \delta_b \circ f') = (\delta_b(1), \delta_b) = (1, \delta_b)$$

- Da $4-1=3$ und $1 \xleftarrow{a} 4$ sind, erhält man aus

$$\begin{aligned} r(3) &= (p, \_) = (1, \_), \\ r(1) &= (\_, f') = (\_, \delta_a) \end{aligned}$$

$$r(4) = (\delta_a \circ f'(p), \delta_a \circ f') = (\delta_a^2(1), \delta_a^2) = (0, \lambda q.0)$$

- Da $5-1=4$ und $3 \xleftarrow{a} 5$ sind, erhält man aus

$$\begin{aligned} r(4) &= (p, \_) = (0, \_), \\ r(3) &= (\_, f') = (\_, \delta_b) \end{aligned}$$

$r(5) = (\delta_a \circ f'(p), \delta_a \circ f') = (\delta_a \circ \delta_b(1), \delta_a \circ \delta_b) = (2, \lambda q.2)$

Wegen $2 \in F$ ist das ein Treffer, d.h. wegen

$$r(5) = (\delta(q_0, B_0 \cdots B_{5-1}), \_) = (2, \_)$$

ist $B_0 \cdots B_4 = a.b.bb.aa.bba. \in \Sigma^* ba.$

- Da $6-1=5$ und $3 \xleftarrow{b} 6$ sind, erhält man aus

$$\begin{aligned} r(5) &= (p, \_) = (2, \_), \\ r(3) &= (\_, f') = (\_, \delta_b) \end{aligned}$$

$$r(6) = (\delta_b \circ f'(p), \delta_b \circ f') = (\delta_b^2(1), \delta_b^2) = (1, \delta_b),$$

was kein Endzustand ist, also ist $w \notin \Sigma^* ba$.

Warum werden Vorkommen von ba übersehen?

Wir finden das erste Vorkommen von ba deshalb nicht, weil es nicht am Blockende auftrat, der Automat aber nur $\Sigma^* ba$ erkennt, also nur Endvorkommen von ba.

Der Automat zu $L' = \Sigma^* ba \Sigma^*$ würde alle Wörter erkennen, in denen ba vorkommt.

Aber: die Simulation ist zu grob, um die Vorkommen des Suchworts, die nicht am Blockende stehen, zu melden: die Funktionen $f' : Q \to Q$ in

$$r(k) = (\_, f') = (\_, \lambda q.\delta(q, B_{k-1}))$$

sind die δ-Übergänge beim Lesen ganzer Blöcke, sie melden nicht, wenn ein Endzustand beim buchstabenweisen Lesen der Blöcke erreicht würde.