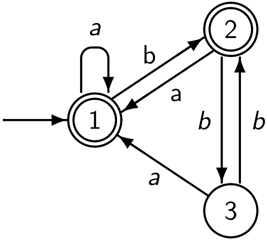
$r^k(i,j)$ expression for $L^k(i,j)$



$r^2(i,j)$	$j = 1$	2	3
$i = 1$	$a^*(baa^*)^*$	$a^*(baa^*)^*b$	$a^*(baa^*)^*bb$
2	$aa^*(baa^*)^*$	$(aa^*b)^*$	$(aa^*b)^*b$
3	$aa^* + a^*baa^*(baa^*)^*$	$a^*b(aa^*b)^*$	$\Lambda + a^*b(aa^*b)^*b$

$$r^3(1,1) = r^2(1,1) + r^2(1,3)r^2(3,3)^*r^2(3,1)$$
$$r^3(1,2) = r^2(1,2) + r^2(1,3)r^2(3,3)^*r^2(3,2)$$

[M] E 3.32

BELOW The *state elimination method* by Brzozowski et McCluskey constructs a regular expression for a given automaton, by iteratively removing the states. The edges of the automaton do not just contain symbols (or Λ) but regular expressions themselves. Thus the graphs are a hybrid form of finite automata and regular expressions. It is rather clear however what they express.

Start by adding a new initial and accepting state; connect the initial state to the old initial state, and connect the old accepting states to the new accepting state, using as label the expression Λ (representing the empty word).

Whenever during this construction two parallel edges (p, r_1, q) and (p, r_2, q) appear, we replace them with a single edge $(p, r_1 + r_2, q)$

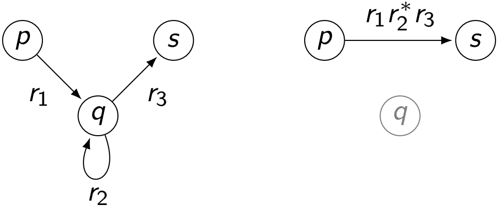
Choose any node q to be removed. Let r_2 be the expression on the loop for q. (If there is no loop we consider this expression to be $\varnothing$.)

For any incoming edge (p, r_1, q) and outgoing edge (q, r_3, s) we add the edge $(p, r_1 r_2^* r_3, s)$ which replace the path from p to s via q.

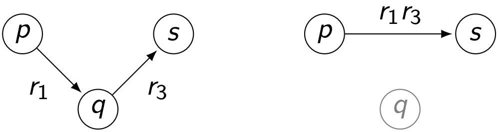
Remove q. Repeat.

When all original nodes are removed, we obtain a graph with single edge; its label represents the language of the original automaton.

join parallel edges



reduce node q



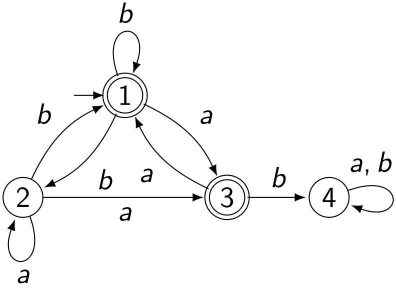
special case: $r_2 = \varnothing$

[M] Exercise 3.54

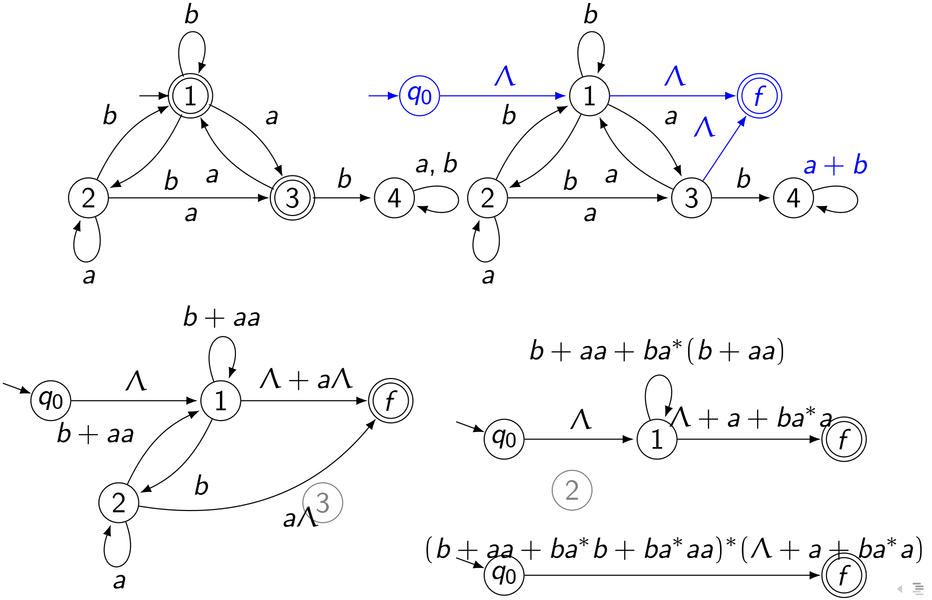
References

R. McNaughton and H. Yamada, Regular expressions and state graphs for automata, IRE Trans. Electronic Computers, vol. 9 (1960), 39–47.
S.C. Kleene. Representation of Events in Nerve Nets and Finite Automata. Automata Studies, Annals of Math. Studies. Princeton Univ. Press. 34 (1956)

state elimination method:
J.A. Brzozowski et E.J. McCluskey, Signal Flow Graph Techniques for Sequential Circuit State Diagrams, IEEE Transactions on Electronic Computers, Institute of Electrical & Electronics Engineers (IEEE), vol. EC-12, no 2, avril 1963, p. 67–76. doi:10.1109/pgec.1963.263416



Eliminate 4,3,2,1


b
1
b
a
2
b
a
3
b
4
a, b
a
a
q0
Λ
1
Λ
f
b
a
Λ
2
b
a
3
b
4
a + b
a
a
b + aa
q0
Λ
1
Λ + aΛ
f
b + aa
2
b
aΛ
3
a
b + aa + ba*(b + aa)
q0
Λ
1
Λ + a + ba* a
f
2
(b + aa + ba* b + ba* aa)*(Λ + a + ba* a)
q0
f

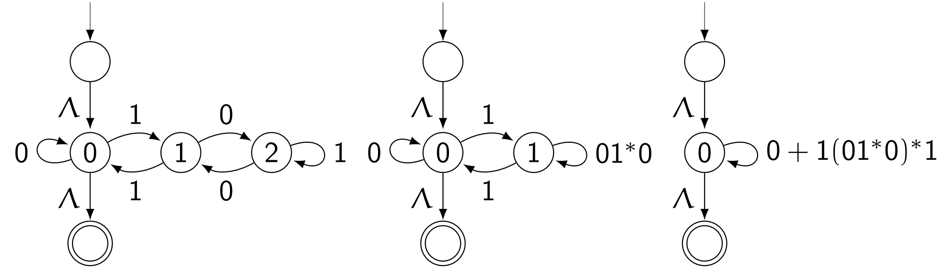
ABOVE

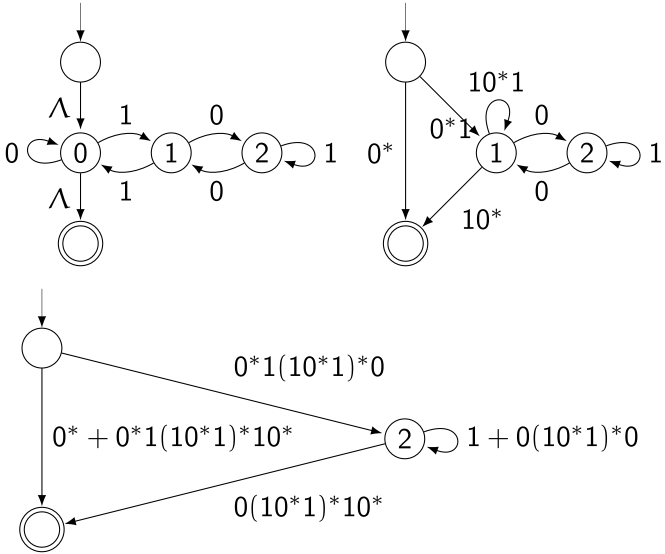
Start by adding new initial and accepting states i and f. Connect these to the original initial and accepting states by edges with the expression Λ.

Note we also replaced the parallel edges a, b (loops on node 4) with the expression $a + b$.

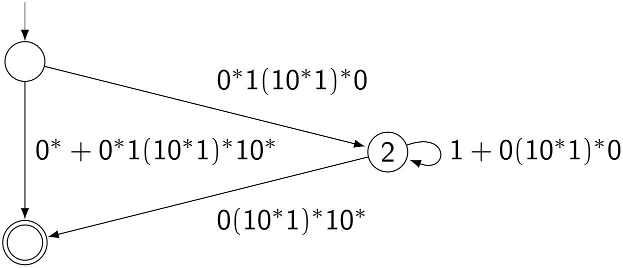
The first node that is eliminated is 4. The proces is not visible here, as there are no pairs (i, j) such that there are edges $(i, R_1, 4)$ and $(4, R_2, j)$, because there are no outgoing edges from 4. Thus no edges are constructed.

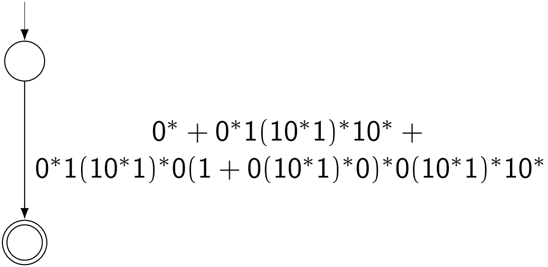
The second node eliminated is 3, as shown.


Λ
1
0
0
0
1
2
1
1
0
Λ
Λ
1
0
0
1
01*0
1
Λ
Λ
0
0 + 1(01*0)*1
Λ


Λ
1
0
0
0
1
2
1
Λ
1
0
10*1
0
0*1
0*
1
2
1
0
10*
0*1(10*1)*0
0* + 0*1(10*1)*10*
2
1 + 0(10*1)*0
0(10*1)*10*

Example divisible by 3, 2





ABOVE

We compute a regular expression from the given automaton in two different reduction orders.

The first example reduces nodes in the order $2, 1, 0$. The result is $(\ 0 + 1(01^*0)^*1\ )^*$

(The removal of the last loop was left as an exercise.)

The second example in the order $0, 1, 2$. The result $0^*{+}0^*110^*{+}0^*10(1{+}00)^*010^*$ $0^* + 0^*1(10^*1)^*10^* + 0^*1(10^*1)^*0(1 + 0(10^*1)^*0)^*0(10^*1)^*10^*$

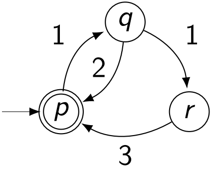
The result differs in structure and size.

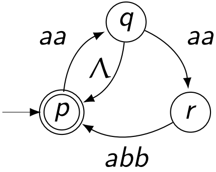
$h : \Sigma_1 \to \Sigma_2^*$ letter-to-string map

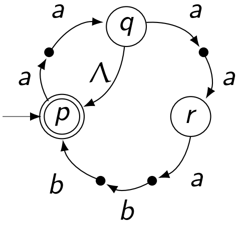
$$h : \begin{array}{ccc} 1 & \mapsto & aa \\ 2 & \mapsto & \Lambda \\ 3 & \mapsto & abb \end{array}$$

$h : \Sigma_1^* \to \Sigma_2^*$ string-to-string map
$h(\sigma_1\sigma_2 \ldots \sigma_k) = h(\sigma_1)h(\sigma_2)\ldots h(\sigma_k)$ $h(12113) = aa \cdot \Lambda \cdot aa \cdot aa \cdot abb$

$K \subseteq \Sigma_1^*$ language-to-language map
$h(K) = \{\ h(x) \mid x \in K\ \}$



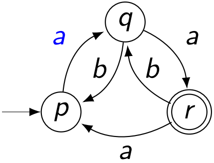


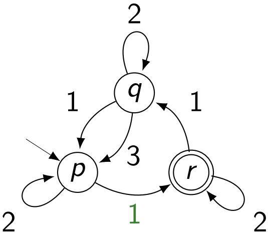


$h : \Sigma_1 \to \Sigma_2^*,\ L \subseteq \Sigma_2^*$ $\quad h^{-1}(L) = \{\, x \in \Sigma_1^* \mid h(x) \in L \,\}$

$$h: \begin{array}{ccc} 1 & \mapsto & aa \\ 2 & \mapsto & \Lambda \\ 3 & \mapsto & abb \end{array}$$

$$\begin{array}{lllllll} \Sigma_1^* \quad h^{-1}(L) & \ni & 1 & 2 & 1 & 3 & 1 \\ \downarrow h & & & & & & \\ \Sigma_2^* \quad L & \ni & aa & \Lambda & aa & abb & aa \end{array}$$





Regular languages are closed under
– Boolean operations (complement, union, intersection, minus)
– Regular operations (union, concatenation, star)
– Reverse (mirror)
– [inverse] Homomorphism

Section 4

Context-Free Languages

$\langle$assignment$\rangle$ ::= $\langle$variable$\rangle$ = $\langle$expression$\rangle$

$\langle$statement$\rangle$ ::= $\langle$assignment$\rangle$ |
  $\langle$compound-statement$\rangle$ |
  $\langle$if-statement$\rangle$ |
  $\langle$while-statement$\rangle$ | ...

$\langle$if-statement$\rangle$ ::=
  if $\langle$test$\rangle$ then $\langle$statement$\rangle$ |
  if $\langle$test$\rangle$ then $\langle$statement$\rangle$ else $\langle$statement$\rangle$

$\langle$while-statement$\rangle$ ::=
  while $\langle$test$\rangle$ do $\langle$statement$\rangle$

Definition (well-formed formulas)

. . . by using the construction rules below, and only those, finitely many times:

– every propositional atom $p, q, r, \ldots$ is a wff

– if ϕ is a wff, then so is $(\neg\phi)$

– if ϕ and ψ are wff, then so are $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$,

BNF Backus Naur form

$\psi ::= p \mid (\neg\psi) \mid (\psi \wedge \psi) \mid (\psi \vee \psi) \mid (\psi \rightarrow \psi)$

M.Huet & M.Ryan, Logic in Computer Science

$AnBn = \{ a^n b^n \mid n \geqslant 0 \} \subseteq \{a, b\}^*$

Example

– $\Lambda \in AnBn$

– for every $x \in AnBn$, also $axb \in AnBn$

[M] E 1.18

Example

– $\Lambda \in AnBn$ (basis)
– for every $x \in AnBn$, also $axb \in AnBn$ (induction)

$S \rightarrow \Lambda$
$S \rightarrow aSb$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aa\,bb$$
$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaa\,bbb$$

if $S \Rightarrow^* x$ then also $S \Rightarrow^* axb$

$Pal \subseteq \{a, b\}^*$

Example

– $\Lambda, a, b \in Pal$
– for every $x \in Pal$, also $axa, bxb \in Pal$

[M] E 1.18

Example

– Λ, a, $b \in Pal$

– for every $x \in Pal$, also axa, $bxb \in Pal$

$S \rightarrow \Lambda \mid a \mid b$
$S \rightarrow aSa$
$S \rightarrow bSb$

$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aababaa$

$AnBn = \{ a^n b^n \mid n \geqslant 0 \}$

variants

$\{ a^n b^{n+1} \mid n \geqslant 0 \}$

$S \to b$ (end with extra b)
$S \to aSb$

$\{ a^i b^j \mid i \leqslant j \}$

$S \to \Lambda$
$S \to aSb \mid Sb$ (free b's)

$\{ a^i b^j \mid i \neq j \}$

$S \to A \mid B$ (choice!)
$A \to aAb \mid aA \mid a$ $(i > j)$
$B \to aBb \mid Bb \mid b$ $(i < j)$

Balanced $\subseteq \{(,)\}^*$: $\Lambda, (), (()), ()(), ((())), (())(), \ldots$

Example

– $\Lambda \in$ *Balanced*
– for every $x, y \in$ *Balanced*, also $xy \in$ *Balanced*
– for every $x \in$ *Balanced*, also $(x) \in$ *Balanced*

[M] E 1.19

Recursion

$Expr \subseteq \{a, +, *, (,)\}$

Example

– $a \in Expr$
– for every $x, y \in Expr$, also $x + y \in Expr$ and $x * y \in Expr$
– for every $x \in Expr$, als $(x) \in Expr$

[M] E 1.19

Example

– $a \in Expr$
– for every $x, y \in Expr$, also $x + y \in Expr$ and $x * y \in Expr$
– for every $x \in Expr$, als $(x) \in Expr$

$S \to a \mid S + S \mid S * S \mid (S)$
derivation(s) for $a + (a * a)$ and $a + a * a \ldots$
ambiguity
[M] E 4.2

$NonPal \subseteq \{a, b\}^*$
$x = abbbaaba \in NonPal$
[M] E 4.3

$NonPal \subseteq \{a, b\}^*$
$x = abbbaaba \in NonPal$

Example

– for every $A \in \{a, b\}^*$, aAb and bAa are elements of $NonPal$
– for every S in $NonPal$, aSa and bSb are in $NonPal$

$A \to \Lambda \mid aA \mid bA$
$S \to aAb \mid bAa \mid aSa \mid bSb$
[M] E 4.3

$NonPal \subseteq \{a, b\}^*$
$x = abbbaaba \in NonPal$

Example

– for every $A \in \{a, b\}^*$, aAb and bAa are elements of $NonPal$
– for every S in $NonPal$, aSa, bSb, aSb and bSa are in $NonPal$

$A \to \Lambda \mid aA \mid bA$
$S \to aAb \mid bAa \mid aSa \mid bSb \mid aSb \mid bSa$
[M] E 4.3

alphabet $\{\ 1, 2, 5, =\ \}$

$\{\ x{=}y \mid x \in \{1,2\}^*, y \in \{5\}^*, n_1(x) + 2n_2(x) = 5n_5(y)\ \}$

$n_\sigma(x)$ number of σ occurrences in x

$212{=}5 \quad 22222{=}55 \quad 12(122)^3 2{=}5^4$

The problem with most solutions is that when read from left to right the initial string over $\{1, 2\}$ cannot always be chopped into part with exact value 5, without chopping the symbol 2.
The solution is like a finite automaton, which reads $1, 2$ and 'saves' the values until the value 5 is reached, then we write a 5 to the right.

$\Sigma = \{\, 1, 2, 5, = \,\}$

variables S_i, $0 \leqslant i \leqslant 4$

axiom S_0

productions

$S_0 \to 1S_1 \mid 2S_2$
$S_1 \to 1S_2 \mid 2S_3$
$S_2 \to 1S_3 \mid 2S_4$
$S_3 \to 1S_4 \mid 2S_05$
$S_4 \to 1S_05 \mid 2S_15$
$S_0 \to =$

Definition

context-free grammar (CFG) 4-tuple $G = (V, \Sigma, S, P)$
- V alphabet *variables* / *nonterminals*
- Σ alphabet *terminals* disjoint $V \cap \Sigma = \varnothing$
- $S \in V$ *axiom*, *start symbol*
- P finite set rules, *productions*
  of the form $A \to \alpha$, $A \in V$, $\alpha \in (V \cup \Sigma)^*$

derivation step $\alpha = \alpha_1 A \alpha_2 \Rightarrow_G \alpha_1 \gamma \alpha_2 = \beta$ for $A \to \gamma \in P$

Definition

language generated by G
$L(G) = \{ x \in \Sigma^* \mid S \Rightarrow_G^* x \}$

[M] Def 4.6 & 4.7

NonPal, its grammar components

$$A \to \Lambda \mid aA \mid bA$$
$$S \to aAb \mid bAa \mid aSa \mid bSb$$

variables $V = \{ S, A \}$
terminals $\Sigma = \{ a, b \}$
axiom S
productions
$P = \{A \to \Lambda, A \to aA, A \to bA, S \to aAb, S \to bAa, S \to aSa, S \to bSb\}$

$\Rightarrow_G^*$ is the *transitive and reflexive closure* of $\Rightarrow_G$

zero, one or more steps

general case $\quad \alpha = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n = \beta$

$\alpha \Rightarrow_G^* \beta$ iff there are strings $\alpha_0, \alpha_1, \ldots, \alpha_n$ such that

- $\alpha_0 = \alpha$
- $\alpha_n = \beta$
- $\alpha_i \Rightarrow \alpha_{i+1} \quad$ for $0 \leqslant i < n$.

special case $\quad n = 0 \quad \alpha = \alpha_0 = \beta$

Variables can be rewritten regardless of context

Lemma

If $u_1 \Rightarrow^* v_1$ *and* $u_2 \Rightarrow^* v_2$*, then* $u_1 u_2 \Rightarrow^* v_1 v_2$*.*

Lemma

If $u \Rightarrow^* v_1 v v_2$ *and* $v \Rightarrow^* w$*, then* $u \Rightarrow^* v_1 w v_2$*.*

Lemma

If $u \Rightarrow^* v$ *and* $u = u_1 u_2$*,*
then $v = v_1 v_2$ *such that* $u_1 \Rightarrow^* v_1$ *and* $u_2 \Rightarrow^* v_2$*.*

$AeqB = \{ x \in \{a, b\}^* \mid n_a(x) = n_b(x) \}$

$aaabbb, ababab, aababb, \ldots$

[M] E 4.8

From lecture 6:

– Even number of both a and b

two letters together

aa and bb keep both numbers even [odd]

ab and ba switch between even and odd, for both numbers

$$(aa + bb + (ab + ba)(aa + bb)^*(ab + ba))^*$$

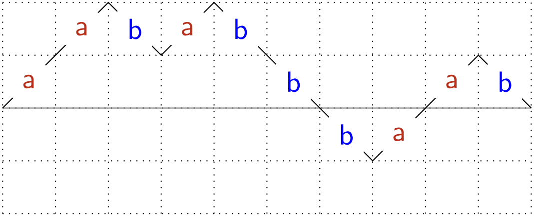
[M] E 3.4

$AeqB = \{ \, x \in \{a, b\}^* \mid n_a(x) = n_b(x) \, \}$

$aaabbb, ababab, aababb, \ldots$

[M] E 4.8

$AeqB = \{\ x \in \{a, b\}^* \mid n_a(x) = n_b(x)\ \}$



$AeqB = \{ x \in \{a, b\}^* \mid n_a(x) = n_b(x) \}$

$aaabbb, ababab, aababb, \ldots$

$$S \to \Lambda \mid aB \mid bA$$
$$A \to aS \mid bAA \qquad A \text{ generates } n_a(x) = n_b(x) + 1$$
$$B \to bS \mid aBB \qquad B \text{ generates } n_a(x) + 1 = n_b(x)$$

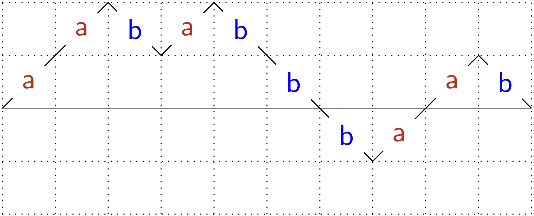
$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow \ldots$ (different options)

(1) $aabB \Rightarrow aabaBB \Rightarrow aababSB \Rightarrow aababB \Rightarrow aababbS \Rightarrow aababb$
(2) . . . (ambiguous, later)

[M] E 4.8

When a string has multiple variables, like aabSB in the above example, then we are not forced to rewrite the first variable, we can as well rewrite another one.
Thus we can do $\mathsf{aab\underline{S}B} \Rightarrow \mathsf{aabB}$, but also $\mathsf{aabS\underline{B}} \Rightarrow \mathsf{aabSaBB}$, for instance.

$AeqB = \{ x \in \{a, b\}^* \mid n_a(x) = n_b(x) \}$



$S \to \Lambda \mid aSb \mid bSa \mid SS$

$S \Rightarrow SS \Rightarrow a_1 S b_6 S \Rightarrow a_1 a_2 S b_3 S b_6 S \Rightarrow \ldots$

$S \Rightarrow a_1 S b_{10} \Rightarrow \ldots$

[M] Exercise 1.66

$i = j + k$ vs $j = i + k$

$L_1 = \{ a^i b^j c^k \mid i = j + k \}$ $\quad aaa\,b\,cc$

$L_1 = \{ a^i b^j c^k \mid i = j + k \}$ $\quad aaa\,b\,cc$

generate as $\quad a^{k+j}\ b^j\ c^k = \underbrace{a^k\ \underbrace{a^j\ b^j}\ c^k}$

$$S \to aSc \mid T$$
$$T \to aTb \mid \Lambda$$

$$S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaTcc \Rightarrow aaaTbcc \Rightarrow aaabcc$$

$L_1 = \{\ a^i b^j c^k \mid i = j + k\ \}$ $\quad aaa\,b\,cc$

generate as $\quad a^{k+j}\ b^j\ c^k = \underbrace{a^k\ \underbrace{a^j\ b^j}\ c^k}$

$$S \to aSc \mid T$$
$$T \to aTb \mid \Lambda$$

$$S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaTcc \Rightarrow aaaTbcc \Rightarrow aaabcc$$

$L_2 = \{\ a^i b^j c^k \mid j = i + k\ \}$ $\quad a\,bbb\,cc$

$L_1 = \{ a^i b^j c^k \mid i = j + k \}$ $\quad aaa\, b\, cc$

generate as $\quad a^{k+j}\ b^j\ c^k = \underbrace{a^k\ \underbrace{a^j\ b^j}\ c^k}$

$S \rightarrow aSc \mid T$
$T \rightarrow aTb \mid \Lambda$

$S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaTcc \Rightarrow aaaTbcc \Rightarrow aaabcc$

$L_2 = \{ a^i b^j c^k \mid j = i + k \}$ $\quad a\, bbb\, cc$

generate as $\quad a^i\ b^{i+k}\ c^k = \underbrace{a^i\ b^i}\ \underbrace{b^k\ c^k}$

$S \rightarrow XY$ (concatenate)
$X \rightarrow aXb \mid \Lambda$
$Y \rightarrow bYc \mid \Lambda$

$S \Rightarrow \underline{X}\, Y \Rightarrow a\underline{X}b\, Y \Rightarrow ab\, \underline{Y} \Rightarrow ab\, b\underline{Y}c \Rightarrow ab\, bb\underline{Y}cc \Rightarrow abbbcc$
$S \Rightarrow X\, \underline{Y} \Rightarrow \underline{X}\, bYc \Rightarrow aXb\, b\underline{Y}c \Rightarrow a\underline{X}b\, bbYcc \Rightarrow ab\, bb\underline{Y}cc \Rightarrow abbbcc$

(a priori there is no prescribed order rewriting X or Y)

Using building blocks

Theorem

If L_1, L_2 are CFL, then so are $L_1 \cup L_2$, $L_1 L_2$ and L_1^*.

[M] Thm 4.9

Using building blocks

Theorem

If L_1, L_2 are CFL, then so are $L_1 \cup L_2$, $L_1 L_2$ and L_1^*.

$G_i = (V_i, \Sigma, S_i, P_i)$, having no variables in common.
[M] Thm 4.9

Using building blocks

Theorem

If L_1, L_2 are CFL, then so are $L_1 \cup L_2$, $L_1 L_2$ and L_1^*.

$G_i = (V_i, \Sigma, S_i, P_i)$, having no variables in common.

Construction

$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, S, P)$, new axiom S
– $P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$ $L(G) = L(G_1) \cup L(G_2)$
– $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$ $L(G) = L(G_1) L(G_2)$

$G = (V_1 \cup \{S\}, \Sigma, S, P)$, new axiom S
– $P = P_1 \cup \{S \to SS_1, S \to \Lambda\}$ $L(G) = L(G_1)^*$

[M] Thm 4.9