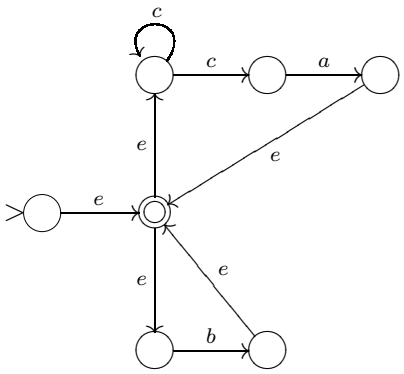
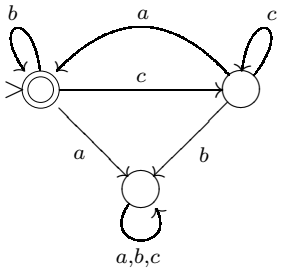
MA217 Videregående diskret matematikk
Onsdag 11. juni 2003
løsningsforslag

Norges teknisk-naturvitenskapelige universitet
Institutt for matematiske fag

1 **a)** For eksempel noe sånt.



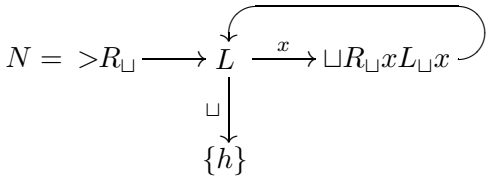
b) En deterministisk Turingmaskin som avgjør L er



c) Maskinen i punkt b) er standardmaskinen fordi de to ikkeaksepterende tilstandene blir skilt av symbolet a.

2 **a)** Konfigurasjonen blir $(s, \triangleright \sqcup \underline{\sqcup} abab)$.

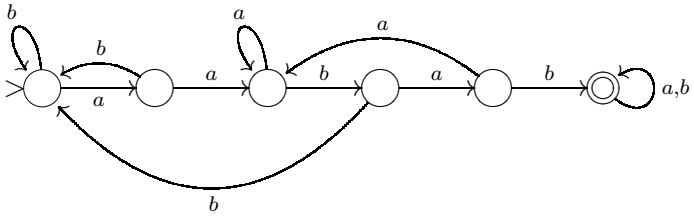
b) Maskinen N som vist på figuren under gjør jobben.



3 **a)** Anta at $M = (K, \Sigma, \delta, s, F)$ er standardautomaten til språket L. Dersom $q \in K$ er en tilstand la M_q betegne automaten $(K, \Sigma, \delta, s, \{q\})$. Automatene M og M_q er like bortsett fra at det er bare q som er sluttilstand i M_q.

Språkene $\{L(M_q) \mid q \in K\}$ danner en partisjon av Σ^*, og dette er partisjonen assosiert med ekvivalensrelasjonen $\approx_L$.

b) Ekvivalensklassene kan ordnes på en rekke hvor klassen lengst til høyre er språket L. Den neste fra høyre består av ord som ikke er med i L, og som slutter med $aaba$. Den tredje fra høyre består av ord som ikke er med i L, og som slutter med aab. Den fjerde fra høyre består av ord som ikke er med i L, og som slutter med aa. Den femte fra høyre består av ord som ikke er med i L, som slutter med a, men ikke med aa eller $aaba$. Den første tilstanden består av alle andre ord inkludert det tomme ordet. Tilsammen er det 6 ekvivalensklasser som på naturlig vis svarer til tilstandene i standardmaskinen til L, som er angitt under.



4 **a)**

$$\begin{aligned} \text{plus}(x, 0) &= x, \\ \text{plus}(x, y+1) &= \text{plus}(x, y) + 1 = \text{succ}(\text{plus}(x, y)). \end{aligned}$$

Altså kan vi skrive $\text{plus} = \text{rek}(g, h)$, der $g = \text{id}_{1,1}$ og $h = \text{succ} \circ \text{id}_{3,3}$.

$$\begin{aligned} \text{pred}(0) &= 0, \\ \text{pred}(x+1) &= x. \end{aligned}$$

Altså kan vi skrive $\text{pred} = \text{rek}(g, h)$, der $g = \text{zero}_0$ og $h = \text{id}_{2,1}$.

$$\begin{aligned} \text{minus}(x, 0) &= x, \\ \text{minus}(x, y+1) &= \text{pred}(\text{minus}(x, y)). \end{aligned}$$

Altså kan vi skrive $\text{minus} = \text{rek}(g, h)$, der $g = \text{id}_{1,1}$ og $h = \text{pred} \circ \text{id}_{3,3}$.

b) I hele dette punktet skriver vi rekursiv funksjon for primitivt rekursiv funksjon.

Alle rekursive funksjoner kan skrives som strenger over et endelig alfabet som inneholder symbolene id, zero, succ, $\circ$, rek, komma og paranteser, samt tallene 0 og 1. (Strenger av disse gir oss blant annet alle nødvendige indekser.)

De strengene som svarer til rekursive funksjoner danner et språk, og det er ikke så veldig vanskelig å vise at dette er et Turingavgjørbart språk. (Mange forskjellige strenger kan gi samme funksjon.)

Vi kan ordne strengene leksikografisk og vi kan dermed snakke om den n'te rekursive funksjonen f_n.

Vi kan nå lage en Turingmaskin som beregner funksjonen g, der $g(n) = f_n(n) + 1$.

Dersom den *beregnbare* funksjonen g hadde vært rekursiv så ville den vært med på listen, altså ville det ha eksistert et naturlig tall m slik at g er funksjonen som er gitt av (programmet eller strengen) f_m, som igjen ville ha medført at $f_m(m) = g(m) = f_m(m) + 1$, hvilket er umulig. Følgelig har vi funnet en beregnbar funksjon g som ikke er rekursiv.

5 Starter vi med $s(x_1) = 0$ får vi $\{(x_2), (\overline{x_2}), (x_2 \vee \overline{x_3}), (x_3 \vee \overline{x_4}), (\overline{x_3} \vee \overline{x_4})\}$, en selvmotsigelse. Starter vi derimot med $s(x_1) = 1$ får vi $\{(x_4), (x_2 \vee \overline{x_3}), (x_3 \vee \overline{x_4}), (\overline{x_3} \vee \overline{x_4})\}$. Vi må derfor ha $s(x_4) = 1$. I neste omgang gir dette $\{(x_2 \vee \overline{x_3}), (x_3), (\overline{x_3})\}$, også en selvmotsigelse.

Dette viser at strengen ikke er med i 2-SATISFIABILITY.

6 **a)** Det finnes et polynom P, slik at for ethvert naturlig tall m og ethvert par av konfigurasjoner C og D, der $C = (s, \triangleright\underline{\sqcup}w)$ og $C \vdash_M^{(m)} D$, så er $m \leq P(|w|)$.

Med andre ord så vil maskinen på inndata w, alltid stoppe før den har tatt $P(|w|) + 1$ skritt.

(Et polynom er en funksjon av formen $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, der n er et naturlig tall som kalles graden til P, og $|w|$ betyr lengden av w.)

b) At M avgjør L betyr at følgende er oppfylt:

1) Enhver kjøring av M stopper.

2) Dersom $w \notin L$ så fører alle kjøringer til tilstanden n.

3) Dersom $w \in L$ så fører minst en kjøring til tilstanden y.

Eller litt mer formelt.

For alle $w \in \Sigma^*$ finnes et naturlig tall n slik at for ethvert naturlig tall m, og for ethvert par av konfigurasjoner C og D, der $C = (s, \triangleright\underline{\sqcup}w)$ og $C \vdash_M^{(m)} D$, så er $m \leq n$, og

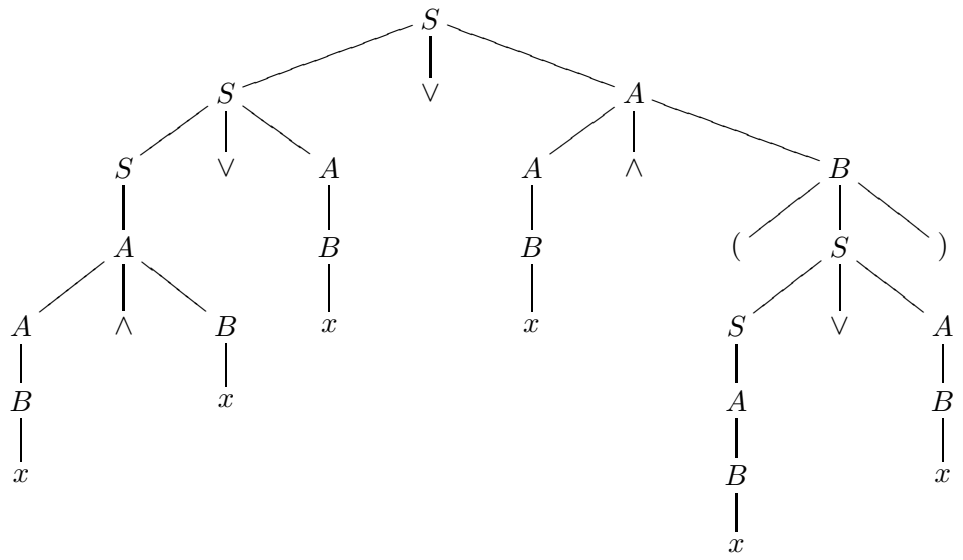
$w \in L$ hvis og bare hvis det finnes en konfigurasjon $D = (y, \triangleright x\underline{a}z)$ slik at $C \vdash_M^* D$.

c) Klassen $\mathcal{NP}$ består av alle språk som kan avgjøres av en polynom tid (ikkedeterministisk) Turingmaskin.

At L kan reduseres til L' i polynom tid betyr at det finnes en polynom tid Turingmaskin som beregner en funksjon $f : \Sigma^* \to \Sigma^*$, slik at $w \in L \Leftrightarrow f(w) \in L'$.

7 **a)** Det er bare en derivasjon som gir et ord med bare en terminal nemlig $S \to A \to B \to x$.

b)



c) Gramatikken blir

$$
\begin{array}{lcl}
S & \to & SS_1 \mid AS_1 \mid BS_1 \mid xS_1 \mid AA_1 \mid BA_1 \mid xA_1 \mid (B_1 \\
S_1 & \to & \vee A \mid \vee B \mid \vee x \\
A & \to & AA_1 \mid BA_1 \mid xA_1 \mid (B_1 \\
A_1 & \to & \wedge B \mid \wedge x \\
B & \to & (B_1 \\
B_1 & \to & S) \mid A) \mid B) \mid x)
\end{array}
$$

Tilsammen 22 regler

d) Den dynamiske algoritmen gir følgende resultat.

S, A					A_1	x
					$\wedge$	
A, B	B_1		B_1	)		
	S	S_1	x			
		$\vee$				
	x					
(						

Siden S er med i den øverste boksen til venstre, så er ordet med i språket.