Løsningsforslag til avsluttende eksamen i HUMIT1750 høsten 2003.

Teksten under har blitt litt "pratsom" fordi jeg har villet forklare hvordan jeg gikk fram. Fra en studentbesvarelse ble det ikke forventet annet enn sluttresultatene. Unntaket er siste spørsmål, hvor man blir bedt om å forklare om beverfunksjonen.

Oppgave 1a.

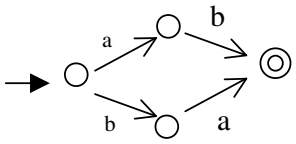
Man kan gå fram på to måter:

Alternativ 1: Først ``ser" man direkte fra grammatikken

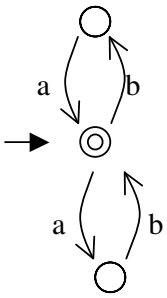
(eller den opplagt ekvivalente grammatikken med produksjoner
$S \to a\,b$ $S \to a\,b\,S$ $S \to b\,a$ $S \to b\,a\,S$)

at den definerer språket $(ab \vee ba)\,(ab \vee ba)^*$

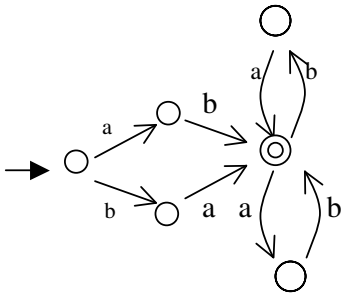
En automat for $(ab \vee ba)$ er



En automat for $(ab \vee ba)^*$ er



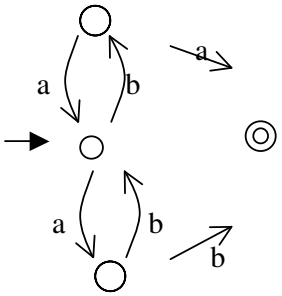
Kopler man de to etter hverandre, får man følgende automat for $(ab \vee ba)\,(ab \vee ba)^*$



Alternativ 2: Hvis vi erstatter grammatikken med den opplagt ekvivalente grammatikken

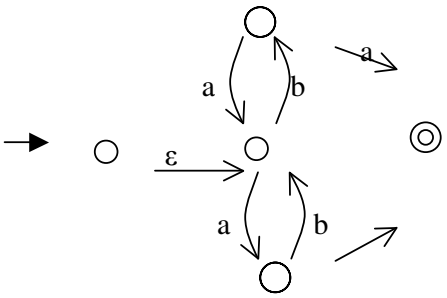
$S \rightarrow a\ B$
$S \rightarrow b\ A$
$A \rightarrow a\ F$
$A \rightarrow a\ S$
$B \rightarrow b\ F$
$B \rightarrow b\ S$
$F \rightarrow \varepsilon$

kan vi bruke metoden i problem 59 baklengs, og få automaten

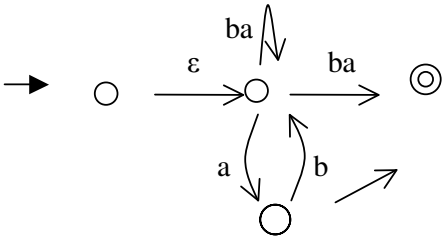


(der tilstandene til venstre er – ovenfra og ned -- A, S, B, og tilstanden til høyre er F)

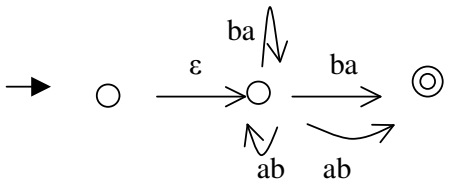
Dette er en annen automat enn vi fikk i første løsningsalternativ, men den er like god. Så bruker vi metoden i problem 52. Da må vi først gjøre start-tilstanden til en "ytre" tilstand, slik:



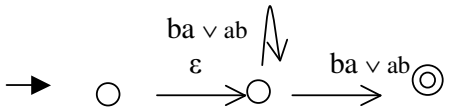
Så kan vi ta vekk den øverste tilstanden. For å sikre at vi fremdeles kan komme oss mellom de gjenværende tilstandene som før, legger vi til nye kanter



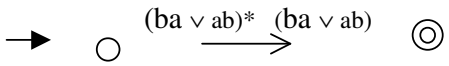
Tilsvarende fjerning av nederste tilstand gir



Ved å knytte sammen alternative transisjoner, får vi så

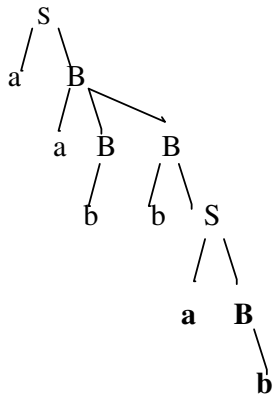


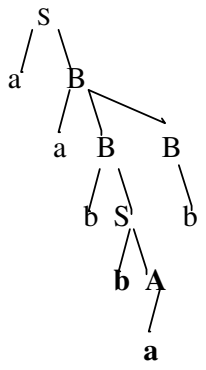
Til slutt tar vi vekk den midterste tilstanden, og får



Det regulære uttrykket (ba ∨ ab)* (ba ∨ ab) blir altså vårt svar.

Oppgave 1b.





Oppgave 1c

Vi oppgir stakkautomaten ved hjelp av en tabell, som følger:

Når vi er i denne tilstanden,	og leser dette fra input,	og finner dette øverst på stakken, da kan vi fjerne dette fra stakken,	gå til denne tilstanden,	og legge disse symbolene inn på stakken. (Symbolet lengst til høyre nederst, det lengst til venstre øverst)
P		S	P	aB
P		S	P	bA
P		A	P	a
P		A	P	aS
P		B	P	b
P		B	P	bS
P		A	P	bAA
P		B	P	aBB
P	a	a	P	
P	b	b	P	

P er start-tilstand og aksepterende tilstand, og start-stakk er en stakk med nøyaktig ett symbol, nemlig S.

Oppgave 2a.
Vi skal altså skrive et regulært uttrykk for språket som inneholder strenger av typen

$P \wedge \neg Q \wedge R \vee R \wedge \neg P \wedge \neg Q$

og så videre. Alfabetet består da av de seks symbolene P Q R $\wedge \vee \neg$

Når vi skriver regulære uttrykk, bruker vi tegnet $\vee$ for å uttrykke union. For å unngå forveksling mellom de to tegnene, skrives union under ved hjelp av tegnet $\cup$.

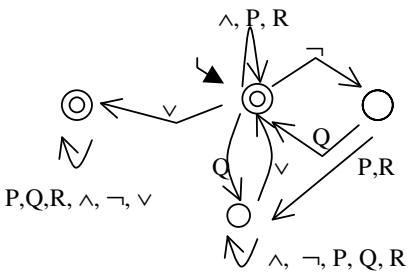
Et regulært uttrykk for det aktuelle språket blir da:

$(\neg \cup \varepsilon)(P \cup Q \cup R)\ (\wedge(\neg \cup \varepsilon)(P \cup Q \cup R))^*\ (\vee(\neg \cup \varepsilon)(P \cup Q \cup R)\ (\wedge(\neg \cup \varepsilon)(P \cup Q \cup R))^*)^*$

Liten forklaring: (Ikke nødvendig som del av besvarelse.)

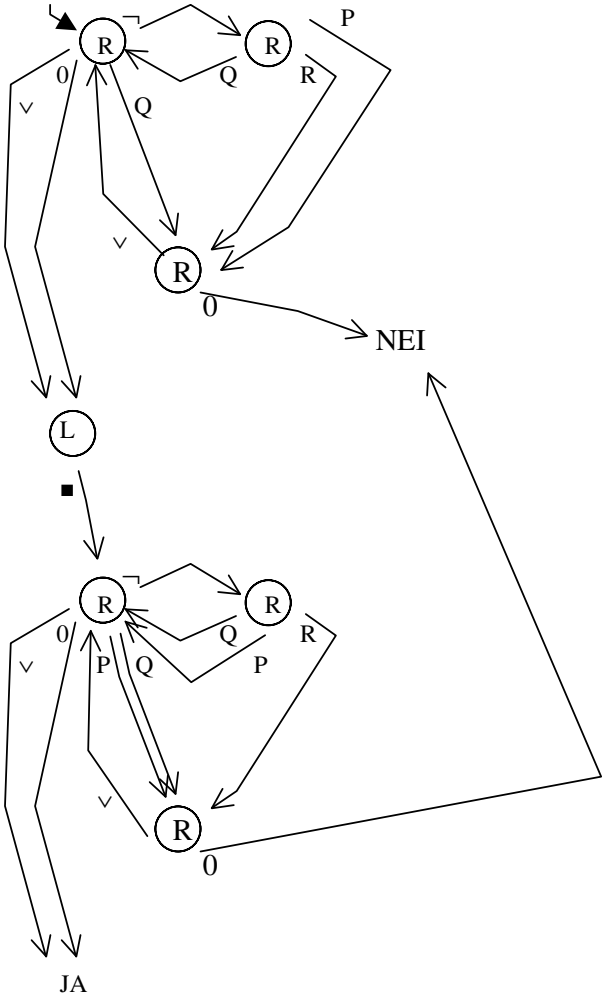
Språket av litteraler angis ved $(\neg \cup \varepsilon)(P \cup Q \cup R)$ Kall dette LITT.
Språket av (ikke-tomme) konjunksjoner av litteraler angis ved LITT $(\wedge$ LITT$)^*$ Kall dette KONJ.
Språket av (ikke-tomme) disjunksjoner av slike konjunksjoner angis ved KONJ $(\vee$ KONJ$)^*$

Oppgave 2b.
Vi skal nå tegne en endelig automat som leser inn slike utsagn, og aksepterer dem som er sanne i den angitte tolkningen. Hva maskinen gjør hvis input ikke er et slikt utsagn, trenger vi ikke å bry oss om. Da er fire tilstander nok:



Noen av pilene har flere tegn på seg, for eksempel den øverste. Vi leser dette som en forkortet skrivemåte for tre transisjoner på en gang: Vi blir værende i start-tilstanden hvis vi leser $\wedge$, og hvis vi leser P, og hvis vi leser R. Tilstanden i midten er altså start-tilstanden. Vi kunne kalt den "denne konjunksjonen er sann så langt." Vi blir værende her (med unntak av avstikkere til høyre når vi leser negasjon) så lenge vi leser sanne litteraler. Herfra kan vi enten gå videre til tilstanden til venstre, som vi kunne kalt "OK". Dit kommer vi når vi har funnet en hel konjunksjon som bare består av sanne litteraler. Vi vet at vi har nådd slutten av en konjunksjon når vi ser tegnet $\vee$. Den nederste tilstanden kunne vi kalt "bedre lykke enste gang". Hit havner vi fra tilstanden over når vi har oppdaget et usant litteral. Vi blir værende her til en ny $\vee$ dukker opp og signaliserer en ny sjanse.

Oppgave 2b.
Vi tenker oss en Turingmaskin med uendelig tape mot høyre, med symbolene P,Q,R, $\wedge$, $\neg$, $\vee$, 0 (blank) og ■
Som input får vi en tape med ■ først (altså til venstre), deretter utsagnet vi skal teste, og så blanke. Maskinen trenger aldri skrive noe: Det er tilstrekkelig at den først går fra venstre mot høyre mens den gjør akkurat det samme som maskinen over. Vi sjekker altså først for sannhet i tolkningen hvor P,R er sanne og Q er gal. Er utsagnet ikke sant her, kan maskinen stoppe og si nei med en gang. I motsatt fall går den tilbake til starten av tapen og etterlikner så en helt tilsvarende endelig automat som sjekker for sannhet i forhold til tolkningen hvor P og Q er usanne mens R er sann:



Denne maskinen skriver altså ikke. På hver transisjonspil finnes det derfor bare ett symbol. Dette står ved starten av pilen, og angir lest tegn. Fra start-tilstanden er det tegnet ut-transisjoner for symbolene Q, $\neg$, $\vee$, 0. Husk på at det er underforstått i vår måte å tegne på at det da også går transisjoner fra denne tilstanden til seg selv for de øvrige symbolene P,R, $\wedge$, ■.

Hvis vi skal sjekke logisk gyldighet av slike utsagn, må vi sjekke sannhet i forhold til alle de åtte mulige tolkningene. Dette kan gjøres ved å kople åtte maskiner etter hverandre på måten vi her har gjort med disse to.

Oppgave 3.
En flittig bever er en bever som skriver minst like mange ett-tall som alle de andre beverne med samme antall tilstander.

La n være et hvilket som helst tall; da finnes det bare endelig mange (forskjellige) bevere med n tilstander. Dette er fordi en bever bare bruker symbolene 0,1, ■, og det bare er et endelig antall kombinasjonsmuligheter for hvordan en maskin kan oppføre seg når den leser disse tegnene mens den er i de n forskjellige tilstandene.

Av de endelig mange beverne med n tilstander, vil det alltid være noen som skriver minst like mange 1-tall som alle de andre, og for hver n vil det derfor finnes flittige bevere med n tilstander. (Det kan være flere av dem, men de skriver like mange 1-tall.) Beverfunksjonen tar inn et tall n og gir tilbake antallet 1-tall som en flittig bever med n tilstander skriver. Vi skriver B for beverfunksjonen.

Vi har alltid $B(n) \leq B(m)$ hvis $n \leq m$:

For det første, hvis n=m så har vi selvfølgelig $B(n)= B(m)$ og derfor også $B(n) \leq B(m)$.

Hvis $n<m$, resonnerer vi som følger: Ta en hvilken som helst flittig bever med n tilstander, og gjør ingenting annet med den enn å legge til m-n nye tilstander som vi aldri kan komme til. (Ta en tegning av den flittige beveren, og tegn inn m-n nye tilstander med ingen transisjoner inn til seg fra den opprinnelige maskinen.) Den nye beveren oppfører seg nøyaktig som den forrige, men har m tilstander. Den er en bever, og skriver B(n) ett-tall. Nå finnes det også en *flittig* bever med m tilstander, og den produserer pr. def. minst like mange 1-tall, altså B(n) eller flere. Med andre ord, $B(m) \geq B(n)$.