INF 2820 V2015: Obligatorisk innleverinsoppgave 1

- **Besvarelsene skal leveres i devilry innen fredag 13.2 kl 18.00**
- **Filene det vises til finner du på /projects/nlp/inf2820/fsa på IFIs linuxmaskiner**
- **Les reglementet for obligatoriske oppgaver:**
  http://www.mn.uio.no/ifi/studier/admin/obliger/index.html
  Dette er en individuell oppgave. Det skal *ikke* levers felles besvarelser.
- **Poengene antyder arbeidsmengden på hvert punkt. Det er 100 poeng i alt.**

Oppgave 1: Endelige tilstandsmaskiner (20 poeng)

Denne oppgaven kan gjøres i JFLAP. Du anbefales likevel å løse den med papir og blyant først for å få eksamenstrening. Så kan du bruke JFLAP til å kontrollere løsningen din.

a. Lag en ikke-deterministisk endelig tilstandsmaskin (NFA) som beskriver språket L1=L(*a*b(a+c)* + ac(b+a)*), der alfabetet er A={*a, b, c*}. (Symbolet + er her disjunksjon).
b. Lag en deterministisk maskin (DFA) som beskriver det samme språket.
c. Lag en tilstandsmaskin som beskriver komplementspråket til L1.

Innlevering: Svar på de tre punktene.

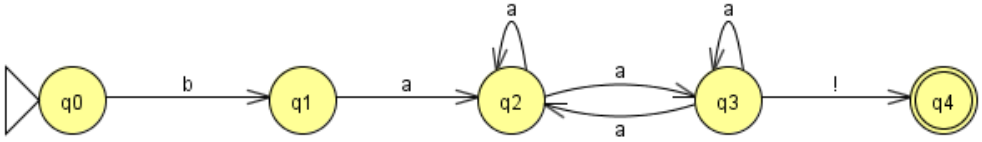
Oppgave 2: Gjenkjenning med DFA (10 poeng)

Se på filen *dfa_recog.py*, som du finner i mappen /projects/nlp/inf2820/fsa/ på IFIs linux-maskiner. Skriv inn maskinen fra oppgave 1b på dette formatet. Prøv den ut på noen strenger med trace=1. Overbevis deg selv om at DFA-en din gjør det den skal og at du forstår programmet *dfa_recog.py*.

Innlevering: Python-representasjon av automaten.

Oppgave 3: Algoritmer for NFA-er (10 poeng)

Vi skal studere forskjellige algoritmer for NFA-er, og se på Python-implementasjonene *nfa_naive_recog.py* og *nfa_smart_recog.py*. Filen *sheep2.nfa* inneholder en representasjon av automaten (som er laget svært flertydig for å gjøre poenget tydelig).



a) Bruk et interaktivt python-vindu (fra idle eller ipython eller lignende) og kjør *nfa_naive_recog.py*. Skriv inn automaten som svarer til nettverket eller les den inn ved

- *nfa2 = NFAFromFile('sheep2.nfa').*

(For at kommandoen skal få denne formen må *sheep2.nfa* ligge i samme mappe som programmet. Ligger den i en annen mappe, må vi skrive stien til filen. I idle og ipython får du autofullføringshjelp til å lete frem stier.)

Bli kjent med representasjonen av automaten ved å gi en del kommandoer som

- *nfa2.start*
- *nfa2.edges*
- *nfa2.finals*

Bli kjent med *nfa_naive_recog.py*. Prøv automaten på en del strenger, både noen som skal aksepteres og noen som ikke skal aksepteres, for eksempel

- *naiveenrec("baaaaaab", nfa2, 1)*

og overbevis deg om at du skjønner hva programmet gjør. (Du behøver ikke studere innlesningsrutinen NFAFromFile.)

b) Gjør det samme med *nfa_smart_recog.py*.

Innlevering: Kjøringseksempel med de to programmene med "trace" av om "baaaaab" anerkjennes.

Oppgave 4: Forkortelser i NFA (20 poeng)

I bruk av regex ser en store besparelser i forkortelser, som [a-z]. Også i NFA-er er det mye å spare på forkortelser, for eksempel for ordklasser i oppgave 5. Oppgave 5 kan løses uten at du bruker forkortelser, men det er en del å spare på å bruke dem. Ta utgangspunkt i programmet *nfa_smart_recog.py*. Innlesningsrutinene er laget for å lese automater både med forkortelser og epsilontransisjoner. Prøv

- nfa3=NFAFromFile('automata/template_abrs.nfa')
- >>> nfa3.edges
- >>> nfa3.abrs

for å se hvordan forkortelser representeres. (Du behøver ikke bry deg om innlesningsrutinen utover dette.) Anerkjenningsprosedyren *nrec()* kan ikke behandle forkortelser. Modifiser *nrec()* til også å kunne bruke automater med forkortelser.

(**Utfordring**: Anerkjenningsprosedyren *nrec()* kan heller ikke behandle hoppekanter (epsilontransisjoner). Innlesingsrutinen tar høyde for hoppekanter i automatene representert ved hash. Du kan se på *template_abrs_epsilon.nfa* og hvordan den representeres etter innlesning med *NFAFromFile*. Hvis du ønsker å bruke hoppekanter, må du modifisere *nrec()* til også å behandle dem. Dette er ikke noe krav, men et tilbud til de som ønsker større utfordringer.)

Innlevering: Modifisert *nfa_smart_recog.py* med kommentarer der det er gjort endringer. Kjøringseksempel *med template_abrs.nfa*.

Oppgave 5: NFA for norske setninger (40 poeng)

Vi skal lage en NFA for setningsstrukturen til noen norske setninger. Vi skal passe på at den ikke får med for mye. Alt som anerkjennes skal være grammatiske norske setninger. Vi kan selvsagt ikke forvente å beskrive alle mulige norske setninger.

Byggestenene vil være norske ord, som *ga*, *barnet, huset*, mao. vil disse utgjøre "alfabetet" for det formelle språket. Ord kan deles inn i ordklasser. De største og viktigste er

- N (*substantiv* eller *nomen*): Kari, barn, barnet, …
- V (*verb*): ga, solgte, spiste, …
- A (*adjektiv*): stort, pent, underlig, …
- P (*preposisjoner*): fra, til, på,…

Innenfor ordklassene kan det være underklasser. For substantiv skiller vi mellom egennavn, *Kari*, og fellesnavn, *barn*. For verb skiller vi bl.a. mellom intransitive*, sov*, transitive, *anerkjenne*, ditransitive, *ga*. Ord kommer også i forskjellige former, som bestemt, *barnet* og ubestemt*, barn*. Underklasse og form kan være bestemmende for hvor et ord kan forekomme.
I denne oppgaven kan vi dele ord inn i grupper ut i fra ordklasse og evt. underklasse og form. Så kan vi bruke forkortelser som navn på slike grupper, og lage en NFA som beskriver setninger basert på forkortelser og ord.

Vi begyner med enkle setninger som

- Kari smilte.
- Barnet sov.

og ser hva mer vi vil ta med. Slike enkle setninger er bygget opp av et substantivledd (NP), *barnet*, og et verballedd (VP), *sov*.

NP

For NP vil vi begrenser oss til (i) egennavn, (ii) substantiv i bestemt form entall, (iii) nøytrumsubstantiv i ubestemt form entall med en foranstilt artikkel eller kvantor. For disse ubestemte vil vi også ta med et ubegrenset antall modifiserende adjektiv.

Skal være med i språket	Skal ikke være med i språket
Kari sov.	
Barnet sov.	
Et barn sov.	
Et lite pent barn sov.	Et pene barn sov.
Kari sa at et lite barn sov.	

VP

Vi vil ha med intransitive, transitive og ditransitive verb. Noen verb kan være både transitive og intransitive, andre kan bare være i en klasse.

Skal være med i språket	Skal ikke være med i språket
Kari ga barnet huset	Kari ga
Kari overrakte dyret eplet	Kari ga dyret
Kari solgte huset	Kari solgte barnet huset
Kari spiste eplet	Kari solgte
Kari spiste	Kari sov eplet
Kari smilte	

Vi skal også ta med verb som tar setningskomplement. Noen av disse tar både et nominalkomplement og et setningskomplement.

Skal være med i språket	Skal ikke være med i språket
Kari fortalte barnet at dyret smilte	Kari sa barnet at dyret smilte
Kari fortalte at dyret smilte	Kari solgte at dyret smilte.
Kari sa at dyret smilte	
Kari fortalte barnet at Ola sa at dyret smilte.	

PP

Vi tar med preposisjonalledd, disse kan modifisere et nominalledd eller VP.

Skal være med	Type
Dyret med barnet sov.	NP-modifikasjon
Dyret sov i huset.	VP-modifikasjon
Dyret med barnet sov i huset.	Begge typer i denne setningen.

NFA-en skal skrives på samme format som *template_abrs.nfa*. Det skal være minst fem forskjellige ord i hver (under)ordklasse (Negennanvn, Nfellesnavn, P, V, A) og minst tre verb av hver type av verb (intransitiv, verb som tar NP-komplement, verb av typen "V at …", osv.) Vi vil ha flere ord i hver klasse for å få prinsipielle løsninger og unngå snarveier. Sjekk at NFA-en virker med (din modifiserte) *nfa_smart_recog.py*. Test nfa-en din på eksemplene i *norsk_testset.txt*. Du kan bruke funksjonen *test()* fra *test.py.* Hvis du ikke vet hvordan du skal lese inn funksjoner fra flere filer, kan du sette sammen filene *nfa_smart_recog.py* og *test.py* til en fil.

Innlevering: Filen med NFA og resultatet av å teste med test().

Lykke til!