Oppgaver INF 2820 V2012

Obligatorisk innleverinsoppgave 1, deloppgave A

- **Frist 10.2 kl 23.59 – strikt – se beskrivelse på semestersiden**
- **Besvarelsene skal leveres i devilry.**
- **Filene det vises til finner du på /projects/nlp/inf2820/finitestate**

Oppgave 1: En latterlig oppgave

Folk ler forskjellig. Noen ler ”haha” andre ler ”hehehe”. Lag en DFA som beskriver språket {ha, haha, hahaha, …, he, hehe, hehehe, …, hi, hihi, hihihi, …, ho, hoho, hohoho, …}. Uttrykk som ”aha” eller ”hahe” skal ikke være i språket. Lag først DFA-en i JFLAP. Se deretter på Python-programmet dfa.py. Skriv inn DFA-en i dette formatet. Prøv den på et par strenger med trace=1. Sjekk at DFA-en din gjør det den skal og at du har forstått dfa.py

Innlevering: JFLAP-automat og Python-representasjon av automat.

Felles for de resterende oppgavene

Oppgave 2 og 3 kan løses direkte i nfa.py. Men hvis du gjør oppgave 4 – og kanskje også oppgave 5 – først, vil det kunne bli mindre arbeid med oppgave 2 og 3.

Oppgave 2: Klokkeslettuttrykk

(Jfr. Oppgave 2.6 i J&M). Lag en NFA for norske klokkeslettutrykk. Du kan bruke 12 timers klokke og ”gammel tellemåte”, eks. ”ti over halv fire” – ikke ”femten førti”.

Eksempler:

Skal være med	Skal ikke være med
tre	halv
halv fem	kvart på halv ni
ti på sju	fem over kvart på ti
fem over halv åtte	ti på halv
kvart over ni	halv over to
kvart på ti	halv på tolv

Det skal være minst ett uttrykk for alle minutter gjennom dagen. NFA-en skal skrives på samme format som template.nfa og testes med nfa.py på eksemplene over.
Innlevering: filen med NFA og kjøringseksempel med eksempeluttrykkene

Oppgave 3: NFA for norske setninger

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. Vi begyner med enkle setninger som

- Kari smilte.
- Barnet sov.

og ser hva mer vi vil ta med

VP-ledd

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	Skal ikke være med
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	Skal ikke være med
Kari fortale 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.	

NP-ledd

Vi begrenser oss til (i) egennavn, (ii) substantiv i bestmt 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	Skal ikke være med
Kari sov.	
Barnet sov.	
Et barn sov.	
Et lite pent barn sov.	Et pene barn sov.
Kari sa at et lite barn sov.	

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.nfa. Sjekk at det virker med (din modifiserte) nfa.py. Kjør til slutt test() fra test_sentences.py.

Innlevering: filen med NFA, evt modifikasjoner til nfa.py og resultatet av å teste med test().

Oppgave 4: Forkortelser i NFA

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 forrige oppgave. Vi har lagt opp til bruk av forkortelser i Python-formatet vårt for NFA-er. Dette er vist i fila template.nfa. Programmet nfa.py er i stand til å lese inn forkortelser, men ikke i stand til å bruke dem. Modifiser programmet nfa.py slik at det kan takle forkortelser.

Innlevering: Modifisert nfa.py. Kjøringseksempel med en NFA med forkortelser.

Oppgave 5: E-transisjoner

Implementasjonen nfa.py takler ikke NFA-er med ε-transisjoner. Vi kan skrive en ε-transisjon som: 5 '#' 7.

Vi skal prøve å modifisere nfa.py til å takle slike kanter.

a) Vi ser først på den naive baktrackingstilnærmingen, recognize4(). Her kan en ta to ulike tilnærminger. Så lenge det ikke er hoppekanter vil hvert trinn (i) flytte seg en kant og (ii) bevege seg en posisjon i input. Med hoppekanter kan vi velge mellom:

i. flytte seg en kant (og ikke flytte seg i input hvis dette er en hoppekant), og
ii. flytte seg en posisjon i input (og samtidig flytte seg flere kanter hvis det er hoppekanter)

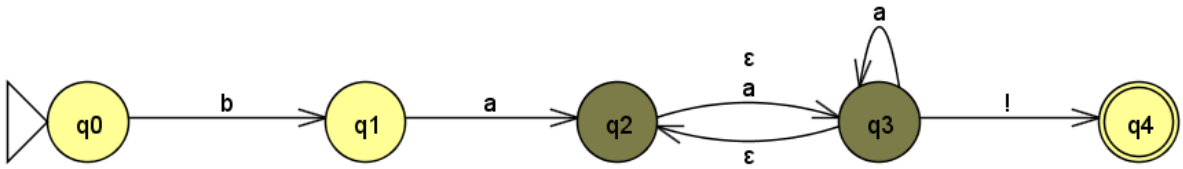
JFLAP illustrerer forskjellen mellom de to med "step by state" og "step with closure".

recognize4() kan modifiseres forholdsvis enkelt til å takle hoppekanter med strategi (i). Gjør dette. Hvilken egenskap må vi forlange av NFA-en for at dette ikke skal gi problemer?

Innlevering: Modifisert recognize4().

b) (En litt større utfordring:) For å følge strategi (ii), beregner vi epsilontillukningen til en tilstand. Lag en prosedyre som implementerer epsilontillukningen til en tilstand og modifiser deretter NFAFromFile.recog() til å takle hoppekanter.

Du kan teste resultatet på den litt ekstreme NFA-en



Innlevering: Modifisert nfa.py med kommentarer der det er gjort endringer.