Mandatory assignment 1, INF2820, 2013

- **Deadline: February 14, 6:00 pm**
- **To be delivered in devilry.**
- **You shoul put your username in the comments of the submitted files, like**
  `# Oblig 1, jtl`
- **The mentioned files can be found at /projects/nlp/inf2820/fsa**

Exercise 1

A) People laugh in various ways. Some laugh “haha”, others laugh “hehe”. Create a DFA that recognizes the language {ha, haha, hahaha, …, he, hehe, hehehe, …, hi, hihi, hihihi, …, ho, hoho, hohoho, …}. Expressions like ”aha” or ”hahe” should not be in the language. Construct first the DFA in JFLAP. The set of symbols (the alphabeth) should be A = {h, a, e, i, o}. There should only be one symbol on an edge.(JFLAP allows an edge to be tagged with, for example *ha*, but then you have an NFA, not a DFA.)

B) Then take a look at the Python program *dfa_recog.py*. Enter the same DFA in this format. Try it out on a couple of strings with trace=1. Make sure that the DFA does what it should and that you understand *dfa_recog.py*.

Submission: JFLAP-automaton and Python representation of automaton.

Exercise 2: Abbreviations in NFA

In using regex there is much to save by using abbreviations, like [a-z]. Also when constructing finite-state automata one may save by using abbreviations, for example for word classes in the following exercises. We will work with NFA implemented in Python, see the file *nfa_smart_recog.py*. In the file we have designed the NFA-class with an argument *abrs* for abbreviations. To see how it is supposed to represent abbreviations, you may read in *template.nfa* as described. The recognition procedure *nrec()* does not handle NFAs with abbreviations. Modify the *nrec()* procedure to handle abbreviations. (Beware that *template.nfa* also contains jump arcs (empty transitions). To test your new recognition procedure, you should make an NFA with abbreviations, but without jump arcs.)

Submission: Modified *nfa_smart_recog.py* saved under a new name. Examples of execution of recognition with abbreviations.

Exercise 3: Time-of-day expressions

(Cf. Exercise 2.6 in J&M). Make an NFA that recognizes time-of-day expressions. You may choose language, either Norwegian or some variant of English. A minimum requirement is that it recognizes at least one expression for every minute throughout the day. In this exercise you may let the symbols be whole words, e.g. A ={two, three, minutes, clock,…} for English.

If you choose English, you should try to be consistent and choose one variant of English. If you choose Norwegian, you may use 12 hour clock and “old way”, i.e. “ti over halv fire” – not ”femten førti”.

Examples:

Included	Excluded
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

The NFA shall be written in the same format as *template.nfa*. If you choose Norwegian, test it on the examples above. If you choose English, construct five different types of expressions you think should belong to the language and five expressions that should not belong to the language and test your automaton on these examples.

Submission: The NFA file and the runs with the examples.

Exercise 4: NFA for sentences

This exercise should be done either for Norwegian or for English – you don't have to do both!

Exercise 4A: Norwegian

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 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.	

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*. Det skal være minst fem forskjellige ord i hver ordklasse (Negennanvn, Nfellesnavn, P, V, A) og minst tre verb av hver verbtype. 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.*

Innlevering: filen med NFA og resultatet av å teste med *test()* og *norsk_testset.py.*

Exercise 4B: English

You shall construct an NFA for the structure of some English sentences. An overall requirement is that all included sentences are grammatical. We cannot expect to describe all possible English sentences. We start with some simple sentences and see what more to include

- Kari smiled.
- Ola slept.

NPs

We will include (i) proper names, (ii) singular nouns preceded by the definite article: *the*, the indefinite: *a*, or a quantifier: *every, some*. We will also include any number of adjectives.

Included	Excluded
Kari slept.	
The child slept.	
A child slept.	
A small pretty bay slept.	
Kari said that a small child slept.	

VPs

We will include intransitive, transitive and ditransitive verbs. Some verbs can be both transitive and intransitive; other verbs only belong to one class.

Included	Excluded
Kari gave the baby the house.	Kari gave
Kari handed the animal the apple.	Kari gave the animal
Kari sold the house	Kari sold the baby the house
Kari ate the apple	Kari sold
Kari ate	Kari slept the apple
Kari smiled	

We also include verbs taking a sentence complement and verbs taking both a nominal and a sentential complement.

Included	Excluded
Kari told the child that the monkey smiled.	Kari said the child that the monkey smiled
Kari told that the monkey smiled.	Kari sold that the monkey smiled
Kari said that the monkey smiled	
Kari told the animal that Ola said that the monkey smiled.	

PP

We include PPs. They may modify NPs or VPs.

Included	Illustrates
The child with the monkey slept.	NP-modification
The monkey slept in the house.	VP-modification
The child with the monkey slept in the house.	Both types

The NFA should be written in the same format as *template.nfa*.It shall contain at least five words of each word class (Common Noun, Proper Noun, Preposition, Verb, Adjective) and at least three verbs of each subcategory. Check that it works with your modified *nrec()*. Finally test your nfa on the examples in *englsih_testset.py* by the help of the function *test()* from *test.py*

Submission: The file with the NFA and the result of the test run.

Exercise 5: E-transitions

Neither *nfa_naive_recog.py* nor *nfa_smart_recog.py* handles ε-transitions. We may write an ε-transition as 5 '#' 7 in a file, cf. *template.py*. We will try to modify the NFA-recognition procedures to include such transitions.

a) We first consider the naïve recognition algorithm from *nfa_naive_recog.py*. We may take one of two different approaches. When there are no epsilon transitions each step will (i) change state, and (ii) move one square in input. With epsilon transitions there is a choice whether a step should describe

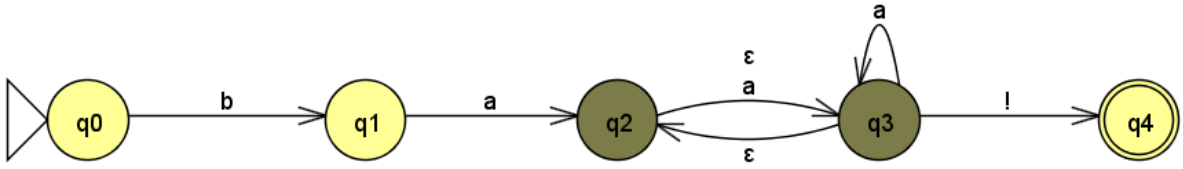
i. change state (and not move in input for an epsilon transition), or
ii. move one square on the tape (and possibly perform several state changes if there are epsilon transitions)

JFLAP illustrates the difference between the two by "step by state" and "step with closure".

nfa_naive_recog.py can be modified fairly easily to cope with epsilon transitions following strategy (i). First modify *nfa_naive_recog.py* to handle abreviations similarly as you did with *nfa_smart_recog.py* in exercise 2. Then extend this to also handle epsilon transitions following strategy (i). Which properties must the DFA satisfy for this to work?

Submission: Modified *nfa_naive_recog.py.*

b) (A challenge:) To implement strategy (ii), we will make use of the epsilon closure of a state. Make a procedure which takes a state and an NFA as arguments and constructs the epsilon closure of the state. Then use this to modify *nfa_smart_recog.py* to handle epsilon transitions. Test your result on the following NFA



Submission: Modified *nfa_smart_recog.py* with comments where you have made changes.

The End