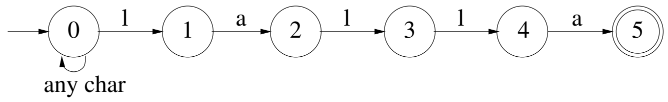
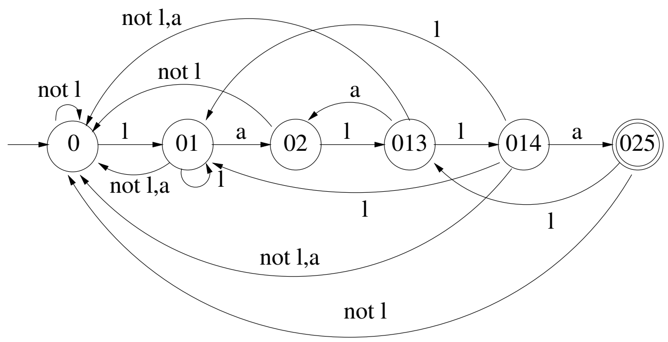
Module Title: Informatics 2A
Exam Diet: Dec 2016–17
Brief notes on answers:

1. (a) The state diagram for the obvious N is



[2 marks if correct, 1 mark if nearly correct.]

(b) The corresponding DFA N is:



[2 marks for correctly labelled states. 5 marks for transitions (roughly 1/3 mark per transition.)]

(c) Run the longer string through M. Each time we enter an accepting state, signal that there is an occurrence of *lalla* ending at the current read position. [1 mark]

2. (a) Bookwork. The Pumping Lemma states that every regular language L has the following 'pumping property': there exists $k \geq 0$ such that for any string $xyz \in L$ with $|y| \geq k$, there is some decomposition of y as uvw with $v \neq \epsilon$ such that $xuv^iwz \in L$ for all i.

[2 marks for evidence of understanding; 3 marks for a fully correct statement. The contrapositive form is also acceptable. But be fairly strict here in requiring the order and type of the quantifiers to be correct.]

(b) A typical attempt at using the Pumping Lemma might run as follows: given $k \geq 0$, consider say $x = a^m$, $y = b^k$, $z = \epsilon$ where $m \neq k$, so that $xyz \in L$. Now given a decomposition $y = uvw$ with $v \neq \epsilon$, it need not be the case that $|v|$ divides $m - k$, and if not, we won't be able to choose i such that $xuv^iwz \notin L$, as this would require that $uv^iw = b^m$.

[Up to 4 marks. A slightly non-standard question type, so award marks for anything showing evidence of good understanding.]

(c) The language $K = \{a^mb^n\}$ is regular, and regular languages are closed under complement and intersection. So if L were regular, then $L' \cap K = \{a^nb^n\}$ would also be regular, and we know from lectures that it is not.

[1 mark for appealing to closure under complement; 1 mark for appealing to $\{a^nb^n\}$; 1 mark for correct use of closure under intersection.]

3. The known words have one possible tag apiece, while each unknown word has three. Hence we only need to consider the probabilities for the possible taggings of *mimsy*

and *borogoves*. Since these are not adjacent, it suffices to compute probability for the following sequences:

MOD mimsy/MOD VB	$0.5 \times 0.8 \times 0.2 = 0.08$
MOD mimsy/NN VB	$0.3 \times 0.5 \times 0.5 = 0.075$
MOD mimsy/VB VB	$0.2 \times 0.2 \times 0.0 = 0.0$
DT borogoves/MOD STOP	$0.3 \times 0.8 \times 0.0 = 0.0$
DT borogoves/NN STOP	$0.7 \times 0.5 \times 0.5 = .175$
DT borogoves/VB STOP	$0.0 \times 0.2 \times 0.1 = 0.0$

The final tagging is: all/MOD mimsy/MOD were/VB the/DT borogoves/NN.

4. The two possible parses are:

```
S(                            % 1.0 x
  NP(                         % 0.7 x
    NN(Scientists))           % 0.3 x
  VP(                         % 0.4 x  <== different from below
    VB(count)                 % 1.0 x
    NP(                       % 0.7 x
      NN(whales))             % 0.2 x
    PP(                       % 1.0 x
      Prep(from)              % 1.0 x
      NP(                     % 0.7 x
        NN(space)))))         % 0.5
                              % = .004116
```

with and:

```
S(                            % 1.0 x
  NP(                         % 0.7 x
    NN(Scientists))           % 0.3 x
  VP(                         % 0.6 x  <== different from above
    VB(count)                 % 1.0 x
    NP(                       % 0.3 x  <== different from above
     NP(                      % 0.7 x
       NN(whales))            % 0.2 x
     PP(                      % 1.0 x
       Prep(from)             % 1.0 x
       NP(                    % 0.7 x
         NN(space))))))       % 0.5
                              % = .0018522
```

The parser chooses the first parse, attaching the PP to the verb.

5. (a) There are many possible answers. Here's one:

$$\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{VP} &\rightarrow \text{VB ADVP-POS} \mid \text{NEG VB ADVP-NEG} \\
\text{NP} &\rightarrow \text{you} \mid \text{the film} \\
\text{NEG} &\rightarrow \text{did not} \\
\text{VB} &\rightarrow \text{like} \\
\text{ADVP-POS} &\rightarrow \text{somewhat} \\
\text{ADVP-NEG} &\rightarrow \text{at all}
\end{aligned}$$

(b) Again, many possible answers. Example:

$$\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{VP} &\rightarrow \text{VB ADVP[POS]} \mid \text{NEG VB ADVP[NEG]} \\
\text{NP} &\rightarrow \text{you} \mid \text{the film} \\
\text{NEG} &\rightarrow \text{did not} \\
\text{VB} &\rightarrow \text{like} \\
\text{ADVP-POS} &\rightarrow \text{somewhat} \\
\text{ADVP-NEG} &\rightarrow \text{at all}
\end{aligned}$$

(c) If the grammars are designed as above, they are equally expressive.

6. (a) The set E of potentially empty non-terminals is just $\{\mathsf{opts}, \mathsf{qualifier}, \mathsf{args}\}$. The *First* sets are:

$$\begin{aligned}
\mathit{First}(\mathsf{args}) &= \{\epsilon, \mathit{str}\} & \mathit{First}(\mathsf{file}) &= \{\mathit{str}, -\mathtt{jar}\} \\
\mathit{First}(\mathsf{qualifier}) &= \{\epsilon, :, = & \mathit{First}(\mathsf{opt}) &= \{-\} \\
\mathit{First}(\mathsf{opts}) &= \{\epsilon, -\} & \mathit{First}(\mathsf{command}) &= \{\mathtt{java}\}
\end{aligned}$$

[1 mark for E, half a mark for each *First* set. These are all very easy.]

(b) The *Follow* sets are:

$$\begin{aligned}
\mathit{Follow}(\mathsf{command}) &= \{\$\} & \mathit{Follow}(\mathsf{args}) &= \{\$\} \\
\mathit{Follow}(\mathsf{file}) &= \{\mathit{str}, \$\} & \mathit{Follow}(\mathsf{opts}) &= \{\mathit{str}, -\mathtt{jar}\} \\
\mathit{Follow}(\mathsf{opt}) &= \{-, \mathit{str}, -\mathtt{jar}\} & \mathit{Follow}(\mathsf{qualifier}) &= \{-, \mathit{str}, -\mathtt{jar}\}
\end{aligned}$$

[1 mark per *Follow* set. These are harder than the *First* sets.]

(c) The parse table is:

	`java`	−	−`jar`	:	=	*str*	$
command	`java` opts file args						
opts		opt opts	ϵ			ϵ	
opt		− *str* qualifier					
qualifier		ϵ	ϵ	: *str*	= *str*	ϵ	
file			−`jar` *str*			*str*	
args						*str* args	ϵ

[2 marks for a table of the right format, including column for $. Roughly half a mark per correct entry. I expect it to be easy to get at least 5 marks, but quite hard to get the full 9 marks.]

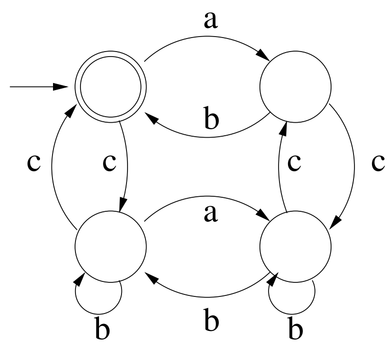
(d) The computation proceeds as follows:

Operation	Input remaining	Stack
	`java` *str* −`jar` *str*	command
Lookup `java`, command	`java` *str* −`jar` *str*	`java` opts file args
Match `java`	*str* −`jar` *str*	opts file args
Lookup *str*, opts	*str* −`jar` *str*	file args
Lookup *str*, file	*str* −`jar` *str*	*str* args
Match *str*	−`jar` *str*	args
Lookup −`jar`, args		

At this point, a blank entry in the table is encountered: '−`jar` found where args expected'.

[5 marks for the course of computation, 1 mark for pinpointing the error.]

7. (a) The state diagram for N is:



[2 marks for a sensible array of states with the right start and accepting state. 3 marks for the right transitions.]

(b) In general, we should take $N = (Q, \Delta, S, F)$ where:

- $Q = Q_0 \times Q_1$,
- $\Delta = \{((p_0, p_1), s, (q_0, p_1)) \mid (p_0, s, q_0) \in \Delta_0\} \cup \{((p_0, p_1), s, (p_0, q_1)) \mid (p_1, s, q_1) \in \Delta_1\}$,
- $S = S_0 \times S_1$,
- $F = F_0 \times F_1$.

[3 marks for Δ, 2 marks altogether for Q, S, F. Hopefully they will be able to abstract from part (a) to arrive at the general definition.]

(c) For each $a \in \Sigma$, let L_a be the language $\{a\}$; then the interleaving of all these languages is exactly $L(\Sigma)$. Moreover, each L_a is regular, as shown by an obvious two-state NFA. By applying the construction of (b) to all these NFAs, we obtain an NFA for $L(\Sigma)$ with $2^{|\Sigma|}$ states.

[2 marks for noting that $L(\Sigma)$ is the interleaving of the L_a. 1 mark for justifying that each L_a is regular. 1 mark for invoking the construction of part (b). 1 mark for the number of states.]

(d) An NFA is minimal if no two distinct states have the same associated language (where the language associated with a state q is the set of strings that take us from q to an accepting state). The NFA from (c) is indeed minimal: each state corresponds to a subset $S \subseteq \Sigma$ (the set of symbols that must appear in the string in order to arrive in that state). The associated language for such a state consists of all strings listing the symbols of $\Sigma - S$ in some order, and this is clearly different for each subset S.

[2 marks for definition of minimality; 1 mark for saying the NFA is minimal; 2 marks for justification.]

(e) Given any k, let $s = a^{2k}$. Then M_s has $2k+1$ states, so M_s^2 has $(2k+1)^2$ states. However, in this case L_s^2 consists of just the single string a^{4k}, so the minimal DFA has $4k+1$ states. But $(2k+1)^2 = 4k^2 + 4k + 1 > 4k^2 + k = k(4k+1)$.

[2 marks for picking a suitable s dependent on k; 3 marks for the rest of the argument.]

8. (a) Here's a very simple solution, showing only one example for both NN and JJ categories (the rest are similar).

S → a NP is JJ $\{\exists x.NP.sem(x) \wedge JJ.sem(x)\}$
NN → table $\{\lambda x.\text{TABLE}(\text{X})\}$
JJ → blue $\{\lambda x.\text{GREEN}(\text{X})\}$

(b) Here's a highly simplified solution. To correctly index nonterminals in the semantic representation, we use coindexes; these are ignored in the syntax.

S → a NP_1 and NP_2 are JJ_1 and JJ_2 , respectively
$\{\exists x.\exists y.NP_1.sem(x) \wedge JJ_1.sem(x) NP_2.sem(x) \wedge JJ_2.sem(x)\}$
NN → table $\{\lambda x.\text{TABLE}(\text{X})\}$
JJ → blue $\{\lambda x.\text{GREEN}(\text{X})\}$

(c) The grammar below is linguistically dubious, but it produces the correct string language.

S → a NN , RC , and JJ , respectively
RC → NN , RC , JJ | and NN are JJ
NN → table $\{\lambda x.\text{TABLE}(\text{X})\}$
JJ → blue $\{\lambda x.\text{GREEN}(\text{X})\}$

(d) No, because each sentence has a substring of the form NN^m *and NN are* JJ^m which can be shown by the pumping lemma to be non-regular. (It is obvious context-free since we can write a CFG for it.)

(e) Suppose we have an RC with n items. Then the NN at depth 1 is related to the JJ at depth n, the NN at depth 2 is related to the JJ at depth $n-1$, and so on. So this is not possible using any mechanism the students learned in class, since these only relate items that appear in the same clause.

(f) Respectively resembles the cross-serial construction in Dutch and Swiss German: it coordinates expressions in ways that violate the nesting of the context-free derivation, though in this case, the string language itself is still context-free.