

The following is a list of topics in this course from which the midterm exam questions will be chosen. Questions will be designed to test

1. Whether you can understand, correctly use and define major concepts introduced
2. Whether you can carry out certain important algorithms, or apply them in simple applications
3. Whether you can apply theorems and properties characterizing languages
4. Whether you can recognize simple, useful consequences of the definitions and theorems you have studied, and whether can persuasively demonstrate how to deduce them.

The emphasis will not be upon proofs of theorems: you will not be expected to recall details of inductive proofs of theorems in the text. However, you will be expected to be able to derive (prove) consequences that follow from known results. At least one short derivation of this sort will be on the exam.

The exam will consist of several short questions, and a few longer questions requiring somewhat greater time.

You may bring your text, notes, homeworks and solutions, etc. to the exam in any form (paper, electronic, etc.). *Beware*, however, of the time overheads that will occur if you scan your materials for a concept or example.

Reference	Applications	Concepts/Definitions	Algorithms
Intro lecture		automaton language	
Prelim lecture Chapt. 0	digraphs bit matrix reachability reflx., symm. trans. expressing properties using relational calc. everywhere describing sets mathematical terminology	binary relations relational calculus: R^{-1} , R^k , $R \circ S$ R^* closures induction language ops: \cup , \cdot , $*$ regular expressions	matrix powers
Text p. 16 Notes		DFA configuration, yield in 1 step, yield* computation (seq. of config.) acceptance, language of M	
Text 1.1 Text 1.1 Text 1.2	designing FA regular operations non-determinism specification	NFA acceptance by NFA	
Text 1.3	editors prog. language definition	subset construction ϵ -move, ϵ -closure	Rabin-Scott ϵ -move removal
Text 1.3	from specification to machine		NFA \leftrightarrow reg. exp.
Notes	sequential circuits text scanning	finite transducer	
Text 1.1 Notes	constructing machines from others		closure properties
Text 1.4 Notes	proofs of non-regularity	pumping lemma	
Notes		decision problem	decision algorithms

The following problems are suggested as preparation for the midterm exam on Friday 7 October. They are *not* to be handed in.

- 1 A relation R is a *partial order* if it is reflexive, transitive and antisymmetric. Express " R is a partial order" in relational calculus terms. Prove that if R is a partial order, its graph cannot have any cycles (paths which connect back to their starting node) except for self-loops.
- 2 Show that

$$\varepsilon + 1^*(011)^*(1^*(011)^*)^* = (1 + 011)^*$$

- 3 Diagram a DFA accepting the set of strings over $\Sigma = \{0, 1\}$ which begin and end with 0 and every 1 is followed by at least three zeros.
- 4 A state q in an NFA $M = (K, \Sigma, \delta, s, F)$ is said to be *reachable* iff there is some path from s to q in M . A state p is *coreachable* iff there is some path from p to some state in F .
 - (a) Describe an algorithm for determining the set $K^r \subseteq K$ of reachable states of M . Code is unnecessary.
 - (b) Based upon your observations in part (a), describe an algorithm for determining the set $K^c \subseteq K$ of *coreachable* states.
- 5 The UNIX command

```
egrep ^expr$ filename
```

is a useful text processing tool. It compares each line of the named file to the regular expression `expr`. Each line in the set denoted by the regular expression is written to the user's terminal. Note that the *entire line* must be accepted by (must match) the expression or it is not printed. For example, `(a+b)` will print only lines consisting of a single lower case `a` or a single lower case `b`.

`Egrep` allows some handy shorthand for commonly used regular expressions. A string enclosed in brackets `[]` matches any *single* character from the string. Ranges of characters may be abbreviated as in `[a-zA-Z0-9]` which matches a single lower case letter or digit. The dash reverts to its usual meaning outside of brackets. For example, to get all lines containing alternating letters and dashes, and ending with two blanks, one can use `[A-Za-z](-[A-Za-z])*\b\b`. (We have made the blank `\b` visible for clarity). A caret `^` inside the brackets matches the *complement* of the set of characters specified. Thus `[^a-z]` matches any character that is *not* a lower case alphabetic.

Give `egrep` expressions which would list the following at your terminal:

- all lines beginning a paragraph (paragraph indent = 5 blanks)
- any line having a word with `q` not followed by `u`, unless the `q` is the last character of the word (e.g., `Iraq`). Words are delimited by blanks or line ends.
- any line with a word having `ei` unless it occurs immediately after `c`.
- any line having a colon which is followed by fewer than 2 blanks
- any line with a telephone number (or what looks like one) on it.

You may assume that the file contains text over the alphabet `{0-9A-Za-z . , : - \b}`.

- 6 Text Exercises/Problems 1.18, 1.19, 1.22, 1.21(b) (use the Text algorithm: node elimination), 1.17, 1.23, 1.36, 1.40(a), 1.29