Informatics 2A 2012–13

Lecture 31

Revision Lecture

Alex Simpson
John Longley

Reminder: pass criteria

You have completed your coursework. This provides your mark towards 25% of the course mark.

The remaining 75% of the course mark is provided by the exam.

For a pass in Inf2A, you need all of the following:

- At least 40% combined total mark.
- At least 35% in the exam.
- At least 25% on the assessed coursework.

For (safe) progression to Honours degree programmes, you need to pass *all* Inf2 courses with at least 50% *on the first attempt*.

The 2012 Inf2A Exam

December exam time and location:

NFR08008 - Informatics 2A
Location: Patersons Land
Date/Time: Friday 14/12/2012, 14:30:00-16:30:00 (02:00:00)

This is copied from the Registry exam timetable

`http://www.scripts.sasg.ed.ac.uk/registry/examinations/`

which is the official exam timetable. Make sure that you use this link to double-check all your exam times (including Inf2A).

A resit exam will be held in August 2013.

Exam structure

The exam is pen-and-paper, and lasts 2 hours.

It consists of:

- Part A: 5 compulsory short questions, worth 10% each.

  Guideline time per question: 10 minutes

- Part B: a choice of 2 out of 3 longer questions, worth 25% each.

  Guideline time per question: 30 minutes

The guideline times allow 10 minutes for reading and familiarizing yourself with the exam paper.

Part A questions

The 5 compulsory short questions are new this year and replace 20 multiple-choice questions in previous years.

The questions will be similar in style and length (but not necessarily in topic) to the questions on this week's Tutorial 9.

The multiple-choice questions of previous years still provide good revision material in terms of coverage of topics.

Revision office hours

Alex Simpson (IF 5.25):

11.30–12.30 on: Tue 4th, Fri 7th
Tue 11th, Thu 13th

John Longley (IF 5.12): TBA

Examinable material

Examinable material: formal language thread

Lectures 3–12

All of the material on regular and context-free languages (Lectures 3–12) is examinable unless explicitly flagged as non-examinable.

E.g., the general proof of Kleene's theorem (slides 20–23 of Lecture 5) is non-examinable.

Lectures 13, 27

The details of Micro-Haskell (syntax, type-checking and semantics), covered in Lectures 13 and 27, are not examinable.

However, the general principles of types, type-checking and abstract syntax, from Lecture 13, are examinable.

Examinable material: formal language thread (continued)

Lecture 28

The notions of context-sensitive, noncontracting and unrestricted grammar are examinable. As is the definition of the different levels in the Chomsky hierarchy.

The context-free pumping lemma is (sadly!) not examinable.

Lectures 29–30

Non-examinable: detailed definitions of Turing machines and linearly bounded automata, proof of undecidability of halting problem, examples of decidable and semidecidable but undecidable problems in general mathematics.

(Weakly) examinable: the general notions of decidable, semidecidable and undecidable problems.

Examinable material: natural language thread

The main thing being tested is your ability to apply *and understand* the methods for solving certain standard kinds of problems.

'Algorithmic' problems:

- POS tagging via bigrams or Viterbi algorithm (lecture 16).
- CYK and Earley parsing (lectures 18, 19).
- Tree probabilities; probabilistic CYK; inferring probabilities from a corpus; lexicalization of rules (lectures 20, 21).
- Computing semantics, including β-reduction (lecture 24).

Examinable material: natural language thread (continued)

‘Non-algorithmic’ problems **(simple examples only!)**

- Design of a transducer for some morphology parsing task (lecture 14).
- Design of context-free rules for some feature of English. (Includes parameterized rules for agreement — lecture 22.)
- Adding semantic clauses to a given context-free grammar (lectures 23, 24).
- Converting an English sentence to a formula of FOPL (lecture 23).

Examinable material: natural language thread (continued)

General topics

- The language processing pipeline (lecture 2).
- Kinds of ambiguity (lectures 2, 15, 17, 24).
- The Chomsky hierarchy, and where human languages sit (lectures 2, 25).
- The *general idea* of parts of speech (lecture 16).
- Word distribution and Zipf's law (lecture 16).

The ideas of recursive descent and shift-reduce parsing (lecture 17) are only weakly examinable.

Non-examinable material: natural language thread

- Specific knowledge of linguistics (everything you need will be given in the question).
- Details of particular POS tagsets; ability to do POS tagging by hand (lecture 15).
- Fine-grained typing, e.g. selectional restrictions on verbs (lecture 22).
- Linear indexed grammars (lecture 25).
- Human parsing; animal language (lectures 25, 26)
- Knowledge of Python.

All natural language examples will be taken from English!

Examinable material: formal language thread (revisited)

The same distinction between algorithmic and non-algorithmic problems applies to the formal language thread.

Algorithmic problems: Minimizing a DFA, converting NFA to DFA, executing a PDA, LL(1) parsing using parse table, generating parse table from LL(1) grammar, . . .

When the algorithm is very complex (e.g., minimization, calculating first and follow sets), it may be easier to work directly with the definitions rather than following the algorithm strictly.

Non-algorithmic problems: Converting DFA to regular expression, designing regular expression patterns, applying pumping lemma, designing CFGs, converting CFG to LL(1), parsing using CSG or noncontracting grammar, . . .

Follow-on Informatics courses

Compiling techniques (UG3)

Covers the entire language-processing pipeline for programming languages, aiming at effective compilation: translating code in a high-level source language (Java, C, Haskell, . . .) to equivalent code in a low-level target language (machine code, bytecode)

Syllabus includes lexing and parsing from a more practical perspective than in Inf2A.

Majority of course focused on latter stages of language-processing pipeline. Converting lexed and parsed source-language code into equivalent target-language code.

Language Semantics and Implementation (UG3)

A course providing detailed coverage of the technique of operational semantics, outlined in lecture 27, for different kinds of programming language.

Programs are assumed given as abstract syntax trees (so lexing and parsing not involved).

A variety of techniques needed to cover different programming styles and features: imperative, functional, local variable declarations, . . .

Computability and Intractability (UG3)

Develops the understanding of what is and is not computable in principle, using Turing machines (Lectures 29–30) as the basic model.

This gives detailed coverage of the notions of decidability and undecidability we met in Lecture 30, including many interesting examples.

A second half looks at classifying decidable problems according to whether their solutions can be computed efficiently, or whether it is seemingly impossible to find an efficient algorithm for them even though inefficient ones exist (so-called intractable problems).

Natural Languages: what we've done, what we haven't.

NLs are endlessly complex and fascinating. In this course, we have barely scratched the surface.

There's a world of difference between doing NLP with small toy grammars (as in this course) and wide-coverage grammars intended to cope with real-world speech/text.

- Ambiguity is the norm rather than the exception.
- Empirical and statistical techniques (involving text corpora) come to the fore, as distinct from logical and symbolic ones.

Coping with the richness and complexity of real-world language is still a largely unsolved problem!

Discourse structure.

In this course, we haven't considered any structure above the level of sentences. In practice, higher level discourse structure is crucial. E.g.

> The Tin Man went to the Emerald City to see the Wizard of Oz and ask for a heart. Then he waited to see whether he would give it to him.

Or compare:

- John hid Bill's car keys. He was drunk.
- John hid Bill's car keys. He likes spinach. (??)

Deep vs. shallow processing.

Roughly, the further we go along the NLP pipeline, the deeper our analysis.

- Many apparently ‘shallow’ NLP tasks (e.g. spell checking; speech transcription) can benefit from the use of ‘deeper’ techniques such as parsing.

- On the other hand, for many seemingly ‘deep’ tasks (e.g. machine translation), current state-of-the-art techniques are surprisingly ‘shallow’ (e.g. use of N-gram techniques with massive corpora).

Follow-on courses in NLP

- **Foundations of Natural Language Processing** [UG3]. Empirical rather than theoretical in focus. Material on text corpora, N-grams, the 'noisy channel' model. A bit on the discourse level.
- **Machine Translation** [UG4]. Mainly on shallow techniques for MT: e.g. phrase-based models. Find out how Google Translate works!
- **Natural Language Generation** [UG4]. Typical problem: generating an English weather report from a weather map. More here at the deep end: e.g. discourse planning and coherence.
- **Natural Language Understanding** [UG4]. Not run this year.

Thank you!

Hope you've enjoyed Inf2A,

and good luck with the exam.