Compilation 2018

What will you learn?

Aslan Askarov
aslan@cs.au.dk

Revised from slides by E. Ernst, M.I. Schwartzbach, and J.Midtgaard

Administrativia

Administrativia

- Lectures are video recorded and live streamed
- Generic questions — use the web forum
- More specific questions – ask your TAs
- The project is done in groups of size 3
  - Use web forum to find group-mates
- PeerWise
  - Use this while studying the material to test yourself and others
- Group registration deadline: **Monday, Sep 03, 14:00**

ACTION: send an email to Benjamin Barslev Nielsen **barslev@cs.au.dk**. Include the following information for all of your group members: 1) Full name 2) AUid

Example email to Benjamin

From: aaabbb@au.dk
To: barslev@cs.au.dk
Sent: Monday, Sep 03 13:58

Dear Benjamin,

Please find below information about our group

1) Aaa Bbb, AUid123
2) Ccc Ddd, AUid456
3) Eee Fff, AUid789

Best regards,
Aaa, Ccc, and Eee

What will you learn?

What can you learn in this course?

- Concrete skills
  - compiler technology
  - insights into programming language design and semantics
- Technical skills
  - a new programming language: ML (SML/NJ)
  - building and testing software on a larger scale
  - reading complex specifications
- Organizational skills
  - handling stress and deadlines
  - finding relevant help and information
  - managing as a group

Compiler technology

- Architecture of a compiler
- Basic compiler phases
- Scanners and parsers
- Scope resolvers
- Type checking
- Translation to intermediate representation
- Normalization
- Code generation

- **You can build your own compiler**

Programming language design

- The interplay between languages and compilers
- Language features that are difficult to implement
- Limitations of current technology
- Scope rules
- Type rules
- Consequences of undecidability

- **You can discuss language design intelligently**
- **You can design another language**

A new programming language: ML

- ML is a fine tool for the job:
  it is made for tree processing
  - Algebraic datatypes
  - Pattern matching
  - ML is a functional language (like Scheme)
  - ML is strongly typed (more so than Java)
  - Discourages but supports assignments
  - Powerful module system

- **One more language tool in your CS belt**

Building and testing software

- 4K-10K LOC of SML to hand in
- Code sharing and versioning
- Systematic observation and testing
- Integrating auto-generated code
- Debugging

- **You can handle a complex piece of software**

Using complex specifications

- The SML language specification
- The Compilation Manager
- The ml-lex and ml-yacc specifications
- IR specification
- x86 assembly

- 7 complex and subtle hand-in specifications

- **You can read complex specifications**

Handling stress and deadlines

- You’ve got 7 deadlines to meet
  - Deadlines are a way of life, not a sudden crisis

- Must balance ambitions with resources
  - Be realistic when estimating the work load
  - Get started in time

- **You are better at facing stress and deadlines**

Finding help and Information

- Specifications may be missing or unclear
- Don’t panic
- Help is available from many sources
  - RTFM
  - Help each other
  - Use weekly consulting productively
  - Ask useful questions on the web board

- **You are better at finding help and information**

Working in a group

- Group work is a challenge
  - Ensure you agree on the ambition level
  - Anticipate mixed skill level and expertise
  - Try to minimize personal drama
- Having people to discussing is as important as the actual work
- Respect internal deadlines
- Ensure your own outcome

- **You will have more experience in group work**

Q: How much do you ***plan*** to work?

My *planned* work load in this course per week is:

A) < 8 hours
B) 8 - 12 hours
C) 12 - 15 hours
D) 15 - 18 hours
E) > 18 hours

Report from 2014 student evaluation:

How many hours did you spend overall (teaching + preparation) on this course per week?

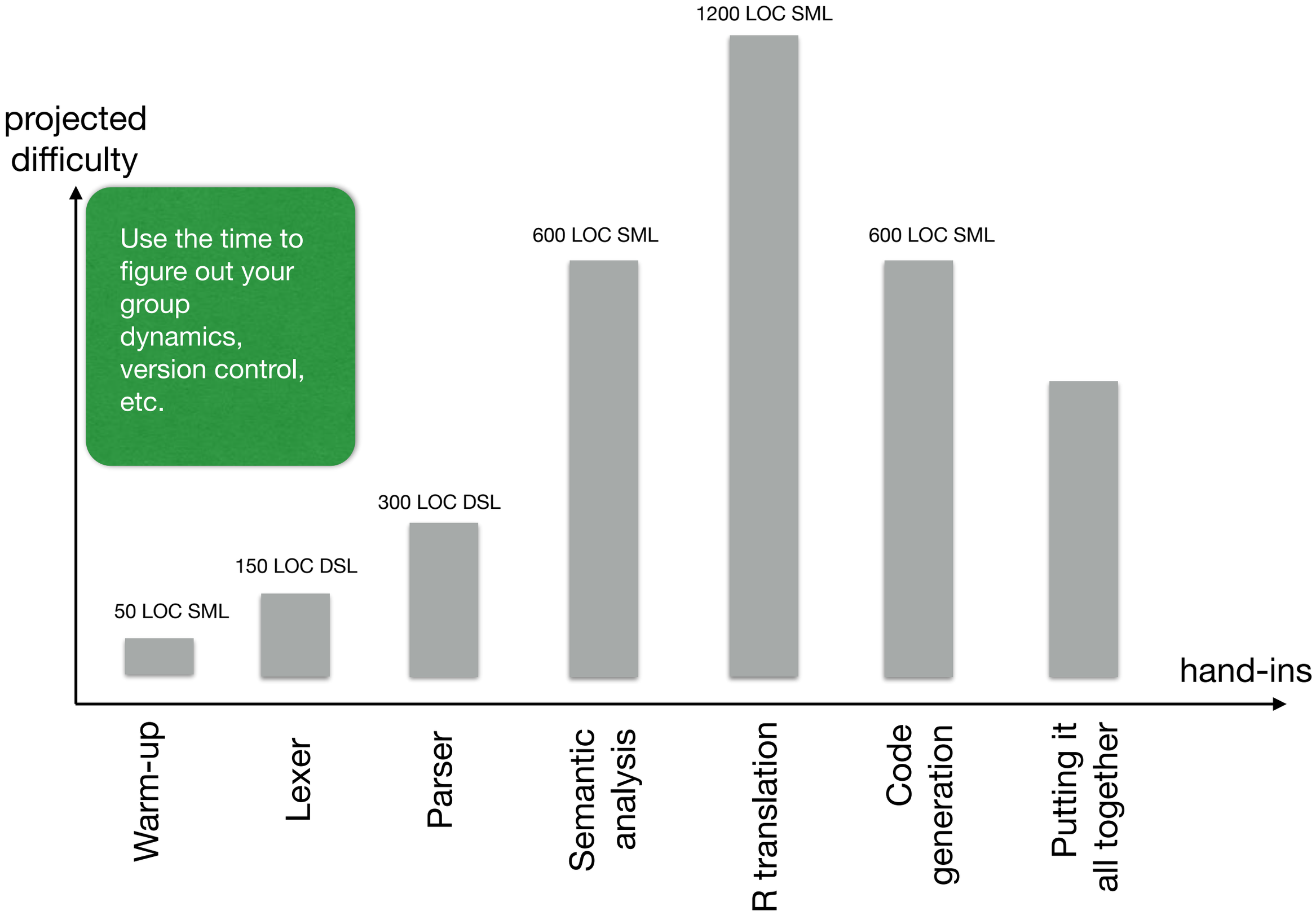
< 8 h	0 %
8 - 12 h	0 %
12 - 15 h	12 %
15 - 18 h	35 %
> 18 h	54 %

Report from 2015 student evaluation:

How many hours did you spend overall (teaching + preparation) on this course per week?

< 8 h	0 %
8 - 12 h	3,1 %
12 - 15 h	9,4 %
15 - 18 h	28,1 %
> 18 h	59,4 %

Hand-in difficulty (estimate)



Why is it *so much* work?

1. Implementing something rather complex and novel for the first time
   - give yourselves time to understand what it is that you need to do (avoid mindless hacking)
2. Using a new programming language to do that
   - think about how you are going to approach the problem
3. Once one gets the grasp of both what to do and how to do it, it's still a lot of programming

Surviving as a dOvs student

- Start working on your assignments early
- Attend the lectures
- Ask questions
- Make sure **you** understand the material
- Claim **your** share of the lecturer/TA's time
- Ensure your own outcome

Background literature recommendations

- Basics of Compiler Design by Torben Æ. Mogensen, available at: http://www.diku.dk/hjemmesider/ansatte/torbenm/Basics/

- Compilers: Principles, Techniques, and Tools (2nd Edition) by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman

- Advanced Compiler Design and Implementation by Steven S. Muchnick (advanced material)