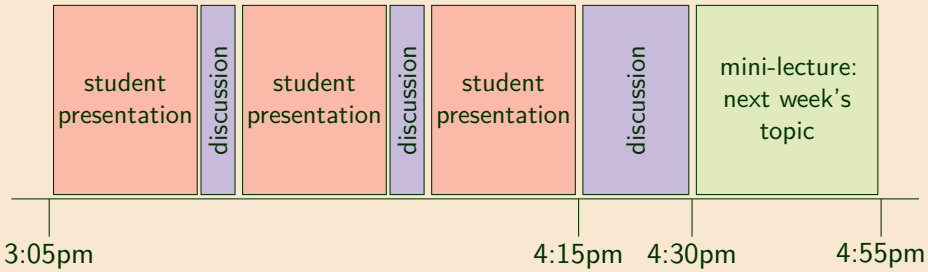
Introduction & overview

Jeremy Yallop

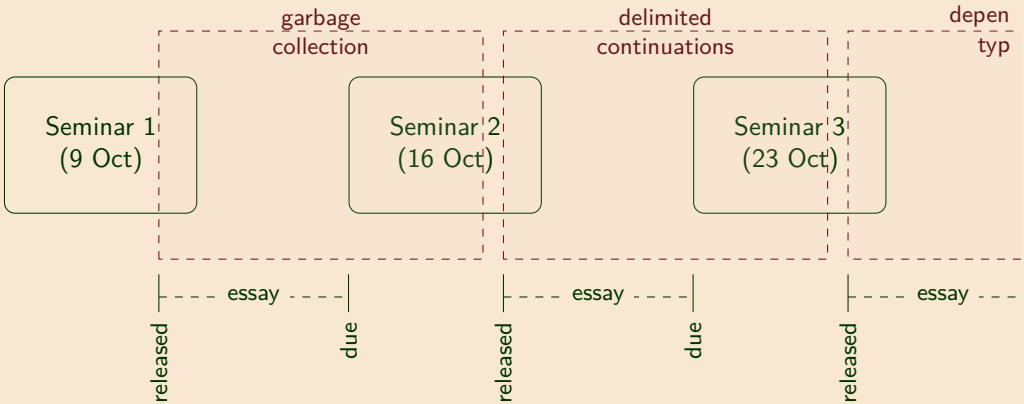
jeremy.yallop@cl.cam.ac.uk

About the class

Intraseminar structure



Interseminar structure



What you'll do each week

Introductory mini-lecture

Background reading

Garbage collection

Read papers

Optional: wider reading

A Real-time Garbage Collector with Low Overhead and Consistent Utilization

Write & submit essay

Take part in discussion

Date	Topic	Speaker 1	Speaker 2	Speaker 3
16 Oct	Garbage collection	-	-	-
23 Oct	Delimited continuations	-	-	-
30 Oct	Dependent types	-	-	-
6 Nov	Module systems	-	-	-
13 Nov	Abstract interpretation	-	-	-
20 Nov	Partial evaluation	-	-	-
27 Nov	Program synthesis	-	-	-

How well did allocation work? Everyone received his/her 1st or 2nd choice.

Opportunity: One remaining vacancy for *module systems*. Volunteers welcome!

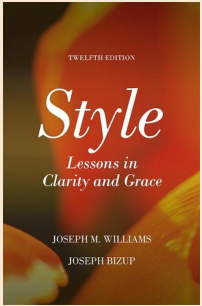
How to get high marks in this class

Essay marks are awarded for *understanding*,
for *insight and analysis*,
and for *writing quality*.

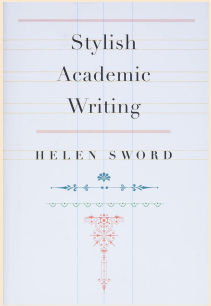
Essays should be around 1500 words.

1. Contextualise *widely*
2. Analyse *deeply*
3. Appraise *thoughtfully*
4. Elucidate *carefully*
5. Describe *originally*
6. Synthesise *insightfully*
7. Expound *illustratively*
8. Write *stylishly*

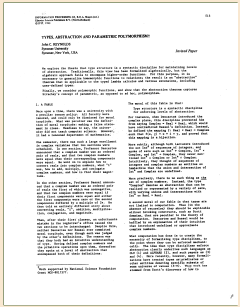
Read *a book*



Read *another book*



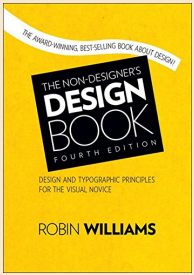
Read *some papers*



Presentation marks are awarded for *clarity*,
for *effective communication*,
and for *selection and organisation of topics*

1. *engage* with the audience
2. *empathize* with the audience
3. *bring people along*
4. explain the *problem*
5. bring out the *key idea*
6. have one *key example*

Read *a book*



Look at *some slides*

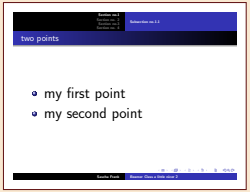
Watch *a presentation*

How to get low marks in this class

1. be exclusively *critical*
2. *quote* extensively
3. assert *without evidence*
4. stay *vague* and *noncommittal*

How to get a low mark in a presentation

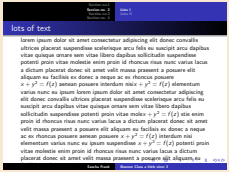
1. read your slides



1. my first point
2. my second point

2. overrun

3. stuff your slides



4. disregard structure

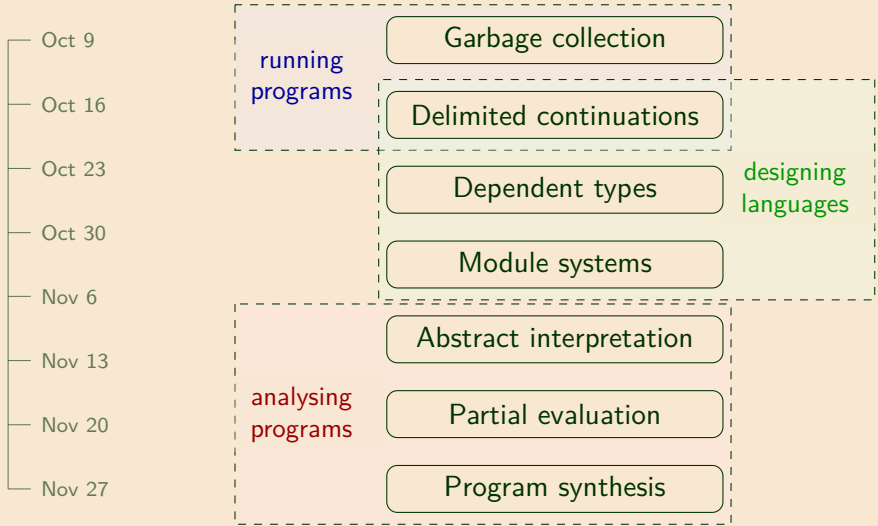
lorem ipsum dolor sit amet consectetur adipiscing elit donec convallis ultrices placerat suspendisse scelerisque arcu felis eu suscipit arcu dapibus vitae quisque ornare sem vitae libero dapibus sollicitudin suspendisse potenti proin vitae molestie enim proin id rhoncus risus nunc varius lacus a dictum placerat donec sit amet velit massa praesent a posuere elit aliquam eu facilisis ex donec a neque ac ex rhoncus posuere aenean posuere interdum nisi elementum varius nunc eu ipsum

Programming languages: themes

Q: what *is* a program?

Q: what *undecidable question*
are we approximating?

Overview

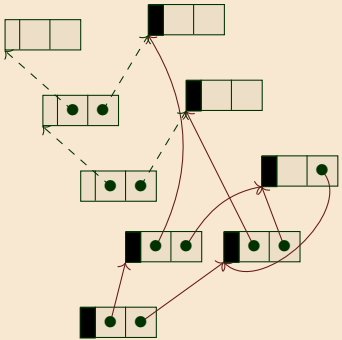


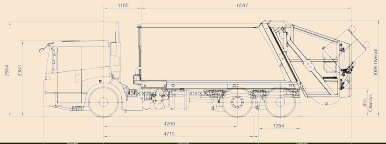
Running programs

Question: How can we efficiently automatically reclaim storage that is no longer needed by a program?

A program is a process that mutates memory by allocating, freeing, reading and writing blocks of memory

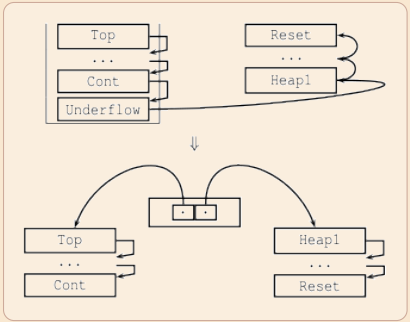
What's undecidable? *Liveness*: it is not possible to determine whether each value can be used by the program in future





Question: How can we extend programming languages with operators that allow powerful manipulation of control flow?

A program is a calculation that may interact with its context



More questions:

- How can we give types to delimited control operators?
- How can we elaborate programs with delimited control?
- What is the connection with algebraic effects?

Designing programming languages

Question: How can we build a powerful, usable, and efficient programming language out of type theory?

A program is a blend of logic and computation.

$$m < n \Rightarrow n \neq 0 : m < n \to n \neq 0$$
$$m < n \Rightarrow n \neq 0 \;\; (s \leq s \;\; m \leq n) \;\; ()$$

What's undecidable? *Type* equivalence is undecidable in general

More questions:

- How should we handle equality?
- How might we write programs in a dependently-typed language?
- How might we compile programs effectively?

Question: How can we construct a language that allows us to assemble large systems from well-specified components?

A program is a large modular system assembled from separately-defined components.

```
module type SET =
sig
  type t
  type elem
  val empty : t
  val add : elem → t → t
  val mem : elem → t → bool
end

module MakeSet (Elem: ORDERED) :
  SET with type elem = Elem.t
```

More questions:

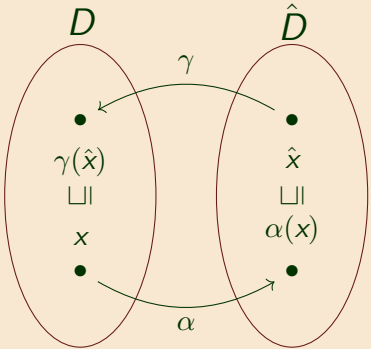
- How can we support abstraction and flexible composition?
- What might a core language of modules look like?
- How might we add support for recursion, higher-order modules, and first-class modules?
- What problems might arise in sophisticated module systems?

Analysing programs

Question: How can we analyse a program to obtain information about it?

A program is an object that can be given a variety of semantics of varying levels of precision

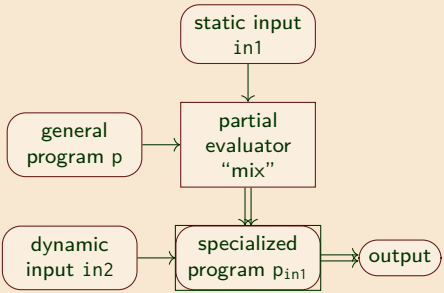
What's undecidable? Most questions. Instead, deal with sound overapproximations.



Question: How might we perform as much computation as possible in advance?

A program is an *open term* that can be simplified using reductions.

What's undecidable? Whether a program is optimally partially evaluated is undecidable.



More questions:

- How can we transform a program to improve its partial evaluation?
- Is partial evaluation useful in practice?
- How can we incorporate equations other than β?

Question: How can we generate programs from specifications?

A program is an object in a very large search space.

What's undecidable? Whether a program meets a specification is undecidable in general.