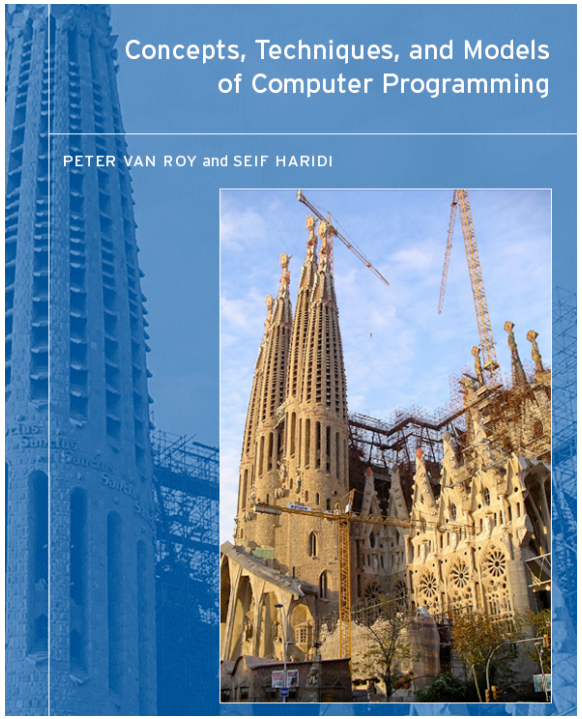
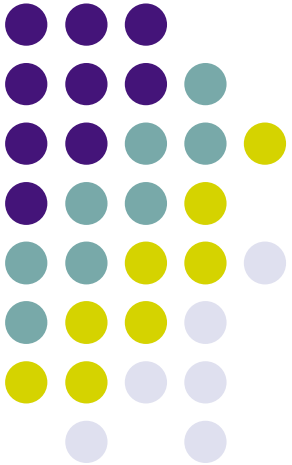
Concepts, Techniques, and Models of Computer Programming



Seif Haridi

& Peter Van Roy

haridi@comp.nus.edu.sg



The Problems of Teaching Programming

- For our purposes, let us define “programming” broadly as the activity that starts with a specification and leads to its solution on a computer

- This includes designing a program and coding it in a language

The Problems of Teaching Programming

- How can we teach programming without being tied down by the limitations of existing tools and languages?

- Example: concurrency
  - is both complicated and expensive in Java, so Java-taught programmers get the mistaken impression that it is always so

The Problems of Teaching Programming

- How can we teach programming without being tied down by the limitations of existing tools and languages?

- Example: data abstraction
  - is limited in pure object-oriented languages to a single style, the “object style”,
  - Programmers don’t realize that there are many other styles, e.g., the “abstract data type” style, each with its own trade-offs.

The Problems of Teaching Programming

- How can we teach programming as a unified discipline with a scientific foundation?
- Not as a set of disjoint paradigms

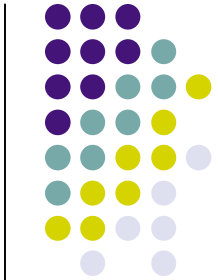
Our Solution
A Concepts-based Approach

- We start with a small language containing just a few programming concepts
- We show how to program and reason in this language
- We then add concepts one by one to remove limitations in expressiveness
- In this way we cover all major programming paradigms
- We show how they are related and how and when to use them together

Our Solution
A Concepts-based Approach

- Similar approaches have been used before, notably by Abelson & Sussman in SICP
- We apply it both broader and deeper: we cover more paradigms and we have a simple formal semantics for all concepts
- We have especially good coverage of concurrent programming

Realizing the Approach



- We draw on more than a decade of research in language design and implementation by an international group, the Mozart Consortium
- We have a software system, Mozart, that can run all the examples
- We have a simple formal semantics for all the paradigms

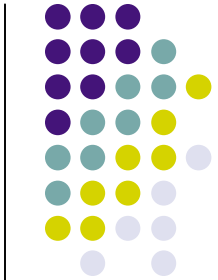
Realizing the Approach

- We have been writing the textbook for four years and teaching with a draft for three and a half years
  - The draft has been used in ten universities around the world
  - The textbook is now available for the first time at SIGCSE 2004 from MIT Press: “Concepts, Techniques, and Models of Computer Programming”, by Peter Van Roy and Seif Haridi
- We are making available for free complete course materials for several courses based on the approach

Some Courses

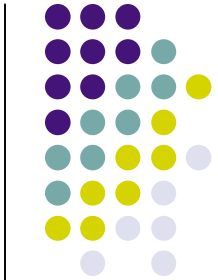
- Here are two ways we have taught with the approach
- Single course (Datalogi II at KTH, CS2104 at NUS, second year)
  - Start with functional programming
  - Give declarative techniques and higher-order programming
  - Add concurrency: gives dataflow programming
  - Add communication channel: gives multi-agent programming

Some Courses



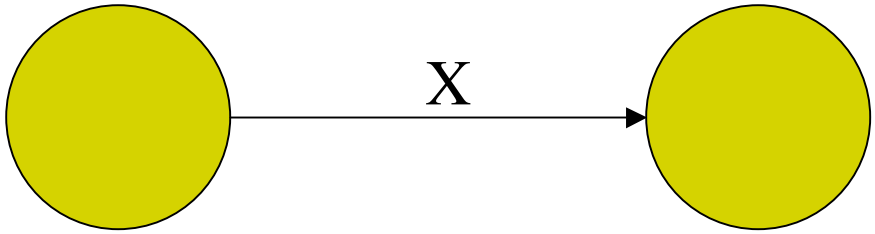
- Two course sequence
  - (at UCL, second and third years)
- First course: similar to the SICP approach (LINF1251)
  - Start with functional programming
  - Give declarative techniques and higher-order programming
  - Add state: lets us cover techniques for data abstraction, such as OOP
  - Explain components and objects

Some Courses



- Two course sequence
  - (at UCL, second and third years)
- Second course: focus on concurrency (INGI2131)
  - Give refresher on functional programming
  - Add concurrency: dataflow programming)
  - Add communication channel: multi-agent programming
  - Add state: gives locks, monitors, and transactions

Stream Communication with Dataflow Concurrency

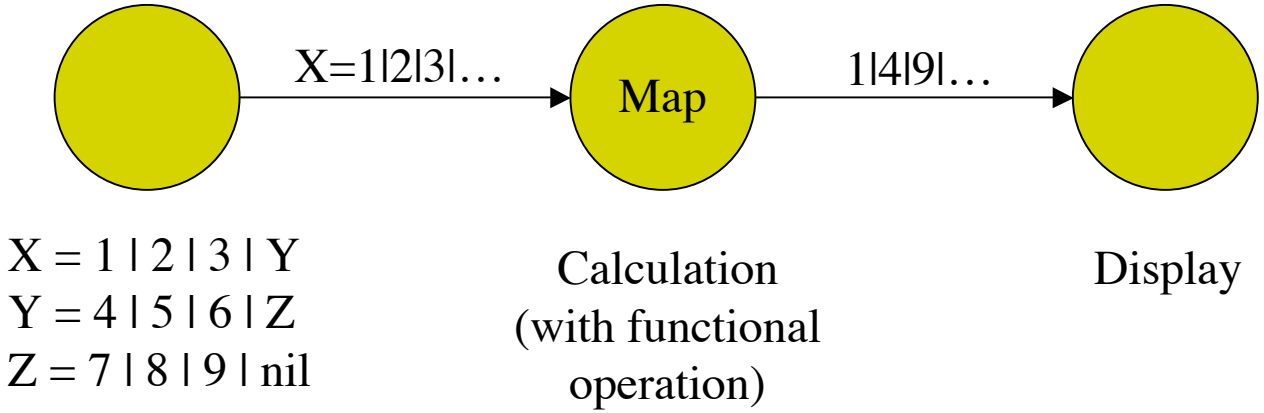


X = all | roads | Y
Y = lead | to | Z
Z = alexandria | nil

Display
(with Browse tool)

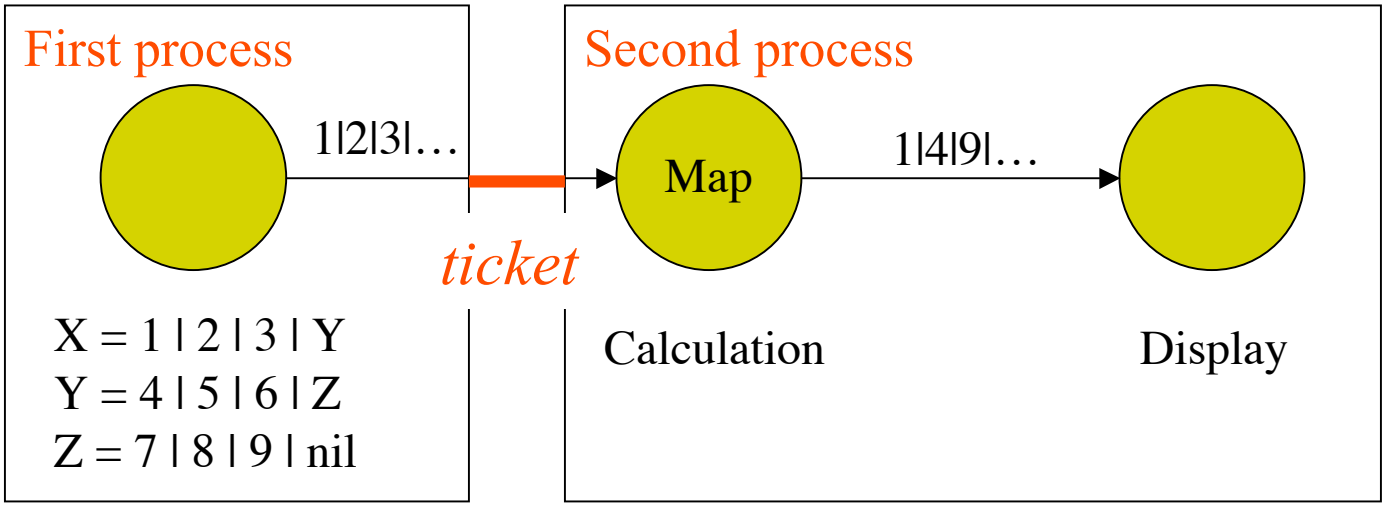
- There are two threads
- The first thread creates the stream X incrementally
- The second thread displays it using dataflow
- Transmission is asynchronous (like a pipe)

Stream Communication with Dataflow Concurrency



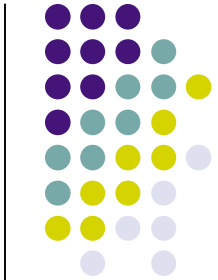
- There are three threads
- The first thread creates a stream of data
- The second thread does a calculation
- The third thread displays the results

Stream Communication with Dataflow Concurrency



- Exactly the same thing, but *distributed*
- The processes connect through a *ticket*
  - A ticket is a reference that can exist outside of a process (since it is coded as an Ascii string)
- Except for the ticket, the program is unchanged

Other Courses

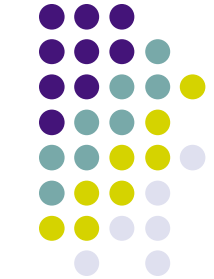


- We also cover these other paradigms
  - Distributed programming (see dataflow example)
  - Lazy (demand-driven) programming
  - Relational programming
  - Constraint programming
  - Logic programming (deterministic and nondeterministic)
  - Concurrent logic programming
  - Graphical user interface programming
- All of these paradigms fit naturally with the rest
  - They are all covered in the textbook

The Exaggerated Importance of Object-oriented Programming

- Consider for example the task of building robust telecommunications systems
- Ericsson has developed an extremely reliable ATM switch (the AXD 301) using a message-passing architecture
- The important concepts are isolation, concurrency, and higher-order programming
- Not used: inheritance, classes and methods, UML diagrams, and monitors

The Exaggerated Importance of Object-oriented Programming



- We find that inheritance especially is overused with respect to other techniques such as composition
- Our approach is agnostic with respect to object-oriented programming
- We place it in the wider context of data abstraction and concurrent programming

Semantics

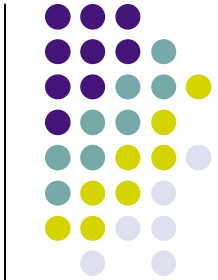
- It's important to put programming on a solid foundation. Otherwise, students will have muddled thinking for the rest of their careers.

- We propose a flexible approach, where more or less semantics can be given depending on taste and the course goals

Semantics can be Taught at Three Levels

- Informal presentation of the formal semantics
- Give an outline of an abstract machine. Explain the concepts of execution stack and environment.
- This can explain last call optimization and memory management (including garbage collection)

Semantics can be Taught at Three Levels



- Complete formal semantics using an abstract machine
- The semantics is at the service of programming: it is as simple as possible without sacrificing rigor or coverage
- Simple reasoning techniques such as invariant assertions can be used in both declarative and procedural programming

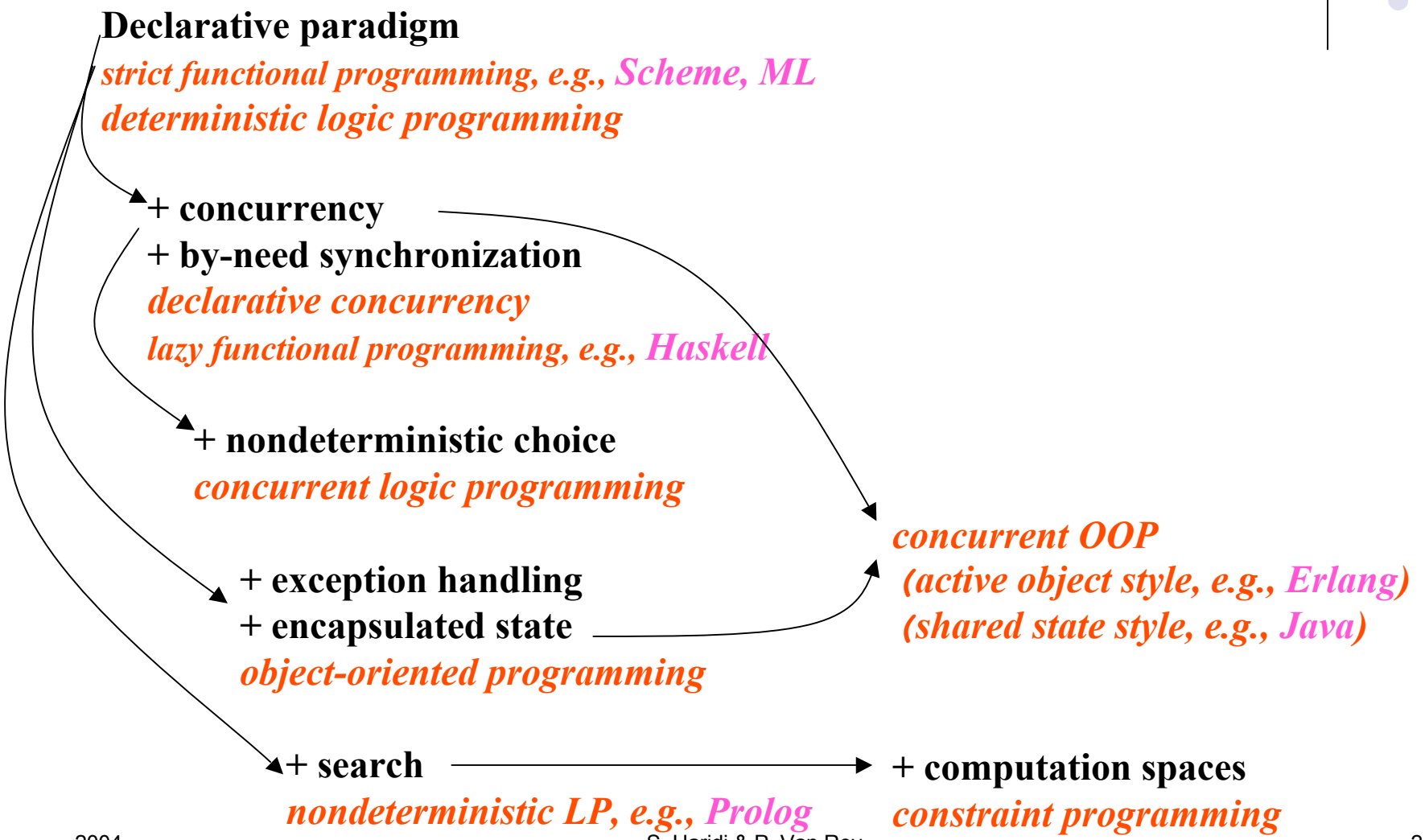
Semantics can be Taught at Three Levels

- Structural operational semantics
- This is the most concise way to give the semantics of a practical language
- Other approaches (axiomatic, denotational, and logical) are introduced for the paradigms in which they work the best

Programming Languages and Paradigms

- We show the relationships between the different paradigms
- Each paradigm has its own kernel language, its own reasoning techniques, and its own programming techniques
- The kernel languages are closely related, e.g., the declarative paradigm is a subset of all of them

Programming languages and paradigms



Conclusions

- We have presented an approach for teaching programming that is based on programming language concepts
  - This covers all major programming paradigms; they are placed in a wider framework and we show why and how to use them together

- We have been teaching with this approach for more than three years and we have written a textbook now published by MIT Press
  - If you are interested in trying out the approach, we will be happy to help
  - See http://www.info.ucl.ac.be/people/PVR/book.html