CS 395T (Fall 2003)
# Integrating Programming Languages & Databases

William Cook
UT Austin CS

## About Me

- PhD Computer Science, 1989
  - Brown University w/Peter Wegner
  - "A Denotational Semantics of Inheritance"
- HPLabs researcher
  - Type theory for OOPL
- Apple Computer
  - Designed and implemented AppleScript
- Startups
  - BAM! Software: Multimedia publishing engine
  - Net-It Software: Intranet/Java
  - Allegis: Enterprise software
    - Partner Relationship Management (PRM)



## Past Research

1. Denotational Semantics of Inheritance
   Foundation for most formal study of inheritance
2. Mixin-Based Programming
   Applied to modules, C++ templates, product lines, etc.
3. ADTs versus OOP
   Objects are not abstract data types
4. A proposal for making Eiffel type-safe
   Canonical example of importance of theory
5. F-Bounded Polymorphism
   Fundamental to theory of typed object systems
6. Inheritance is not Subtyping
   Separation of `extends` and `implements` in Java
7. Interfaces/specifications for collection classes
   Galois lattices for fine-grained interfaces in class libraries

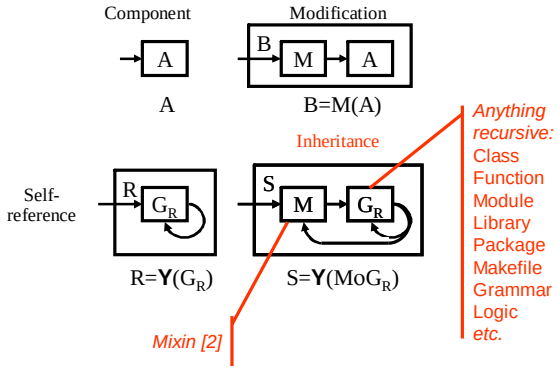


## Impact

- Showed Eiffel type system to be unsound
- Influenced separation of *interfaces* and *classes* in Java
- Developed theory of *mixins*
- Created F-Bounded polymorphism, which is being used in generics for Java



## Inheritance [1]

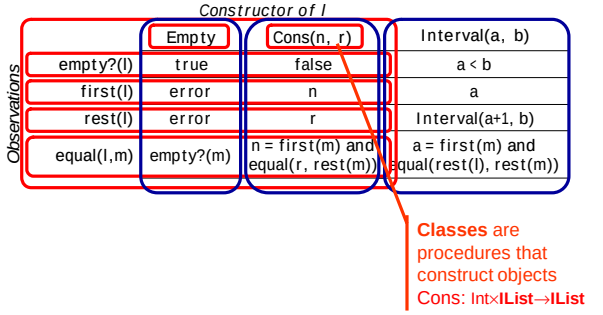





## ADT versus OOP [3]

**Abstract Data Type**
Operations on opaque type

**Object-Oriented Programming**
Procedural data values




**Classes** are procedures that construct objects
Cons: Int×**IList**→**IList**



## Typing Object Interfaces

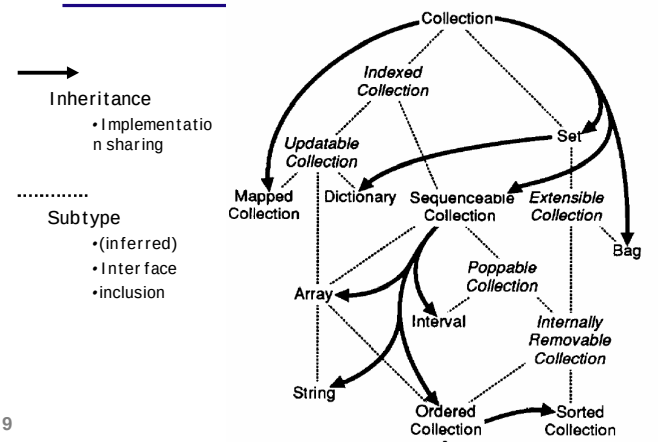
- Objects may have recursive interface types
  - IList = { empty?: Bool,
  - first: Int,
  - rest: IList,
    equal: IList→Bool }
- Consider adding an operation
  - IListEx = { empty?: Bool,
  - first: Int,
  - rest: IListEx,
    equal: IListEx→Bool,
    size: Int }
- Now IListEx is not a subtype of IList!
  - This is because of contravariance in the equal method
  - Normal bounded quantification ∀ t ≤ IList … does not work



## F-Bounded Polymorphism

- Define generator of recursive type
  - $G_{IList}[t]$ = { empty?: Bool,
  - first: Int,
  - rest: t,
    equal: t→Bool }
- IListEx inherits (at the type level) from IList
  - $G_{IListEx}[t] = G_{IList}[t]$ + { size: Int }
- Use F(unction)-Bounded Polymorphism
  - $\forall\ t \le G_{IList}[t]$
  - This allows IListEx because $IListEx \le G_{IList}[IListEx]$
- Conclusions
  - Subtype polymorphism cannot define functions over families of types with similar pattern of recursion
  - Inheritance does not always define subtypes [6]

## Smalltalk Collections [7]






## Authorization Policy

- Domain-specific language for expressing authorization policies
- Integrated with database interface in SQL



## Integrating Programming Languages & Databases

Course Introduction

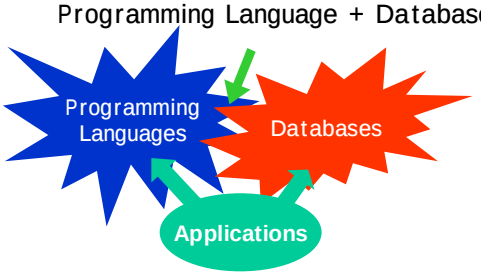
## Introduction

- System = Computation + Persistence
- Computation
  - Many paradigms, we will focus on
    - Object-oriented, some functional
- Persistence
  - Many paradigms, we will focus on
    - Databases (relational, OO, etc.)
- Focus on the *interface* between the two



## Introduction

- System = Programming Language + Database




**Cross-disciplinary research**
**Applications are point of integration**



## Examples

- Mail server
- News server
- Ecommerce application
- Spreadsheet
- Multi-user games
- Web applications
- Business (ERP, CRM, PRM, HRM, SCM)
- Source code control
- Bibliography DB
- File server
- *Just about any system you can thing of…*



## Approaches

- Lots of solutions
  - Embedded SQL
  - Persistent programming language (PPL)
  - Database programming languages (DBPL)
  - Object-oriented database (OODB)
  - Transaction middleware (EJB, COM+)
  - Object-relational mapping (O/R)
- Lots of partial success
  - Some might say "failure"
- What is the problem?



## "Impedance Mismatch"

- Connecting PL and DB is hard because
  - Models don't match
    - Flat tables *versus* Complex objects
    - Declarative queries *versus* Procedural programs
    - Transactions *versus* Semaphores
  - Cultural mismatch
    - DP people don't understand PL research
      - "everything is a database"
    - PL people don't understand DB research
      - "lambda calculus is computationally complete"
  - And anyway, it's not our problem…
    - Industry will figure it out

## What Are Databases For?

- Search algorithm compiler
  - Queries specify what to find, not how
  - Optimizations
    - Content heuristics
    - Physical characteristics (e.g. page size)
    - Indexes, Etc…
  - Runtime compiler
- Concurrency control
  - Manage concurrent read/write
  - Transactions
  - ACID: Atomic, Consistent, Isolated, Durable



## Course Rules

- Listen
- Make mistakes
- Speak clearly
- Show respect
- Think deeply
- Be creative
- Have fun