Making communicating with computers more accessible:
easier, faster, and safer
Welcome!

- This is a graduate course; undergrads are welcome.
  - can have taken 152 or 179 and be just fine, not necessarily both
- You (as a student presenter) will present and lead discussion for at least one paper
- You (as a non-presenting student) will post questions and a summary of the design arguments by Friday of the previous week
- Key learning outcomes:
  - (279r) to look at scientific publications, identify the core design arguments, write new design arguments, and evaluate them
  - (252r) understand, design and implement language abstractions for solving a task
- Group projects will be composed of both “HCI folks” and “PL folks”
Why PL–HCI Matters
A case study in types

Medical infrastructure – Elena
Software engineering – Nada
Medical infrastructure
Hospitals
Suburban General Hospital
141-bed osteopathic hospital in Philadelphia

Opens in Dec 1976
Suburban General Hospital
141-bed osteopathic hospital in Philadelphia

ER opens in Dec 1976.

By July, months of patients have been treated.

Of those treated, 35 died.
Suburban General Hospital
141-bed osteopathic hospital in Philadelphia

ER opens in Dec 1976.
By July, months of patients have been treated.
A new patient comes in, who almost drowned.
Suburban General Hospital
141-bed osteopathic hospital in Philadelphia

ER opens in Dec 1976.
By July, months of patients have been treated.
A new patient comes in, who almost drowned.
Suburban General Hospital
141-bed osteopathic hospital in Philadelphia

ER opens in Dec 1976.
By July, months of patients have been treated.
A new patient comes in, who almost drowned.
Thirteen deaths—Here is a list of suits in mix-up

In September 1977, a three-man panel appointed by Acting District Attorney General Charles Thomas studied the hospital records of 39 patients who had died at tuberculosis hospitals in New York and New Jersey. The panel was headed by Doctor John P. A. Maguire, an expert in tuberculosis, who was also a former New York City Health commissioner. The panel’s findings were made public in a report published in the Philadelphia Inquirer on October 28, 1977.

The report found that 13 patients had been mistakenly treated for cancer instead of tuberculosis. The patients had been treated with radiation therapy, chemotherapy, and surgery, all of which are side effects of cancer treatment. The panel concluded that the patients had been mistreated and that the hospital had violated its rules and regulations.

The hospital was ordered to pay damages to the patients’ families, and the report was widely criticized for its handling of the patients.

The hospital denied the allegations and filed a lawsuit against the panel members. The case was settled out of court, and the hospital agreed to pay a settlement to the patients’ families.

The case highlighted the importance of accurate medical records and the need for better communication between doctors and patients. It also raised questions about the ethics of medical research and the role of government in regulating medical practices.

The report was widely criticized for its handling of the patients.

The hospital denied the allegations and filed a lawsuit against the panel members. The case was settled out of court, and the hospital agreed to pay a settlement to the patients’ families.

The case highlighted the importance of accurate medical records and the need for better communication between doctors and patients. It also raised questions about the ethics of medical research and the role of government in regulating medical practices.

How the gas mix-up was discovered

The gas mix-up was discovered by the New York City Department of Health in 1977. The department was investigating a series of mysterious deaths among patients at a tuberculosis hospital in New York City. The department found that the patients had been treated with a mixture of gases, including carbon monoxide and nitrogen.

It is not clear whether the patients received the gases because of a mistake or if they were deliberately exposed to them. The department concluded that the patients had been mistreated and that the hospital had violated its rules and regulations.

The hospital denied the allegations and filed a lawsuit against the department. The case was settled out of court, and the hospital agreed to pay a settlement to the patients’ families. The case highlighted the importance of accurate medical records and the need for better communication between doctors and patients. It also raised questions about the ethics of medical research and the role of government in regulating medical practices.
Medical Gas Systems Installation
You Need to Know for Safety

<table>
<thead>
<tr>
<th>Piping System Controls</th>
<th>Band</th>
<th>Tape Colors</th>
<th>Stenciled Legends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oxygen</td>
<td>Green</td>
<td>White w/ Green Letters</td>
<td>Oxygen or O2</td>
</tr>
<tr>
<td>Nitrous Oxide</td>
<td>Blue</td>
<td>Blue w/ White Letters</td>
<td>Nitrous Oxide or N2O</td>
</tr>
<tr>
<td>Medical Vacuum</td>
<td>White</td>
<td>Yellow w/ Black Letters</td>
<td>Medical Vacuum or VAC</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>Black</td>
<td>Black w/ White Letters</td>
<td>Nitrogen or NIT</td>
</tr>
<tr>
<td>Carbon Dioxide</td>
<td>Gray</td>
<td>Gray w/ Black Letters</td>
<td>Carbon Dioxide or CO2</td>
</tr>
<tr>
<td>Medical Air</td>
<td>Yellow</td>
<td>White &amp; Black w/ Contrast Letters</td>
<td>Medical Air or Med Air</td>
</tr>
<tr>
<td>Medical Anesthetic Gas Disposal</td>
<td>Violet</td>
<td>Violet w/ White Letters</td>
<td>WAGD</td>
</tr>
</tbody>
</table>
Mix-Ups

- **CO2 and O2:** A grayish green cylinder was confused for a greenish gray cylinder that resulted in a patient inhaling CO2 during transport instead of oxygen. The use of universal adaptors (universal adaptors override the pin indexing system on the cylinder) contributed to this event.

- **CO2/O2 and CO2:** Insufflation of the body cavity for arthroscopy is done with CO2 as the gas will not sustain combustion and is easily absorbed by the body. A gray and green CO2/O2 cylinder was confused with the gray CO2 cylinder that resulted in an internal body cavity fire when a surgical laser was used. The CO2/O2 gas will support combustion.

Oxygen Not Available

- It isn’t always apparent whether an oxygen cylinder is full, partially full, or empty. In cases where the cylinder valve is closed and the regulator valve is open (see photograph; click on photo for larger view), no pressure will register on the pressure gauge. Staff in a hurry has assumed the cylinder is empty when in fact it is full.

- In some cases the O2 cylinder is believed to be empty when trapped pressure in regulator is bled off by opening the flow meter/regulator valve when the cylinder valve is in the closed position.

- You can’t always tell by just looking at the valve if it’s open or closed. Valves controlling the oxygen flow are not indicating type valves. What’s an indicating valve? See: A Brief History of Indicating Valves for Fire Protection ↑↑ for further information on this topic.

Cylinder goes ballistic

- Ferromagnetic O2 cylinders introduced into the MRI environment can inadvertently be turned into missiles when they are drawn into the magnet. For more information on projectile hazards in and around MRIs see our MRI hazards page.

- A second way a cylinder can be turned into a missile is to fracture the cylinder. Escaping gas will propel the cylinder with enough force to penetrate cinder block walls.

Case Studies

From FDA Manufacturer and User Facility Device Experience Database (MAUDE), ECRI, and VA databases

- An "E" cylinder containing CO2 was mistaken for O2 and was used during patient transport - he died. The modified O2 regulator had been modified to fit “grayish” O2 cylinder that really contained CO2. This event emphasizes the point that cylinder color alone cannot be used to confirm the content of the cylinder.
Pin Index Safety System

From Wikipedia, the free encyclopedia

"PISS" redirects here. For other uses, see Piss (disambiguation).

The Pin Index Safety System, or PISS, is a means of connecting high pressure cylinders containing medical gases to a regulator or other utilization equipment. It uses geometric features on the valve and yoke to prevent mistaken use of the wrong gas. This system is widely used worldwide for anesthesia machines, portable oxygen administration sets, and inflation gases used in surgery.

Contents

- Concept
- Pin index configurations
- International Standards
- Limitations
- Alternative systems
- Blanking Plugs
- See also
- References

Concept

The pin index safety system uses a face seal between the cylinder valve and the associated yoke clamp. There are two holes in specific positions on the cylinder valve body below the outlet port, in positions associated with the gas mixture, which prevent connection of the cylinder to a yoke or pressure regulator with a mis-matched set of pins. The holes accept pins 4 mm diameter by 6 mm long which are correctly aligned with the holes.\(^1\)

Pin index configurations

Each gas cylinder has a pin configuration to fit its respective gas yoke. Refer to the diagram for pin numbers; dimensions are in millimeters.

- O_2: 2,5\(^2\) [mm]\(^1\)
- N_2O: 3,5\(^2\) [mm]\(^1\)
Mix-Ups

- CO₂ and O₂: A grayish green cylinder was confused for a greenish gray cylinder that resulted in a patient inhaling CO₂ during transport instead of oxygen. The use of universal adaptors (universal adaptors override the pin indexing system on the cylinder) contributed to this event.
- CO₂/O₂ and CO₂: Insufflation of the body cavity for arthroscopy is done with CO₂ as the gas will not sustain combustion and is easily absorbed by the body. A gray and green CO₂/O₂ cylinder was confused with the gray CO₂ cylinder that resulted in an internal body cavity fire when a surgical laser was used. The CO₂/O₂ gas will support combustion.

Oxygen Not Available

- It isn’t always apparent whether an oxygen cylinder is full, partially full, or empty. In cases where the cylinder valve is closed and the regulator valve is open (see photograph; click on photo for larger view), no pressure will register on the pressure gauge. Staff in a hurry has assumed the cylinder is empty when in fact it is full.
- In some cases the O₂ cylinder is believed to be empty when trapped pressure in regulator is bled off by opening the flow meter/regulator valve when the cylinder valve is in the closed position.
- You can’t always tell by just looking at the valve if it’s open or closed. Valves controlling the oxygen flow are not indicating type valves. What’s an indicating valve? See: A Brief History of Indicating Valves for Fire Protection *† for further information on this topic.

Cylinder goes ballistic

- Ferromagnetic O₂ cylinders introduced into the MRI environment can inadvertently be turned into missiles when they are drawn into the magnet. For more information on projectile hazards in and around MRIs see our MRI hazards page.
- A second way a cylinder can be turned into a missile is to fracture the cylinder. Escaping gas will propel the cylinder with enough force to penetrate cinder block walls.

Case Studies

From FDA Manufacturer and User Facility Device Experience Database (MAUDE), ECRI, and VA databases

- An "E" cylinder containing CO₂ was mistaken for O₂ and was used during patient transport - he died. The modified O₂ regulator had been modified to fit "grayish" O₂ cylinder that really contained CO₂. This event emphasizes the point that cylinder color alone cannot be used to confirm the content of the cylinder.
Root Causes and Other Factors

- The most obvious cue for staff on the contents of a gas cylinder is the cylinder color, however, the color may be misleading or misunderstood. The cylinder label is the primary means of identifying the cylinder content. Follow directions and labels first - not color or other cues (e.g., storage location)… To be truly effective, CGA/ECRI recommends that the label be overwhelming in size.

- If there is a mismatch between the color of the cylinder and the cylinder content, for example a gray cylinder that contains oxygen or a green cylinder that contains nitrogen, this is a guarantee that future problems will occur. For more information on color mismatch confusion and how humans cannot ignored conflicting inputs see the “Stroop Effect” exercises.

- Text labels on cylinder can be damaged and difficult to read; if the label cannot be reliably read the cylinder should not be used.

Next Steps

- Color-coding may be misleading and is not ideal for use in identifying medical gases., NCPS will work with CGA and FDA to determine optimal labeling without use of color for coding on cylinders.
Medical Record Mix-Ups a Common Problem, Study Finds

The opportunities for the mistakes, which can be deadly, are increasing as health care becomes more complex.

By Melinda Beck
Sept. 25, 2016 7:00 p.m. ET

A patient in cardiac arrest was mistakenly not resuscitated because clinicians confused him with a patient who had a do-not-resuscitate order on file.

Another patient was given an okay to undergo surgery based on a different patient’s records and was found dead in his hospital room the next day.

Such patient-identification mix-ups are common...
Medical Record Mix-Ups a Common Problem, Study Finds

The opportunities for the mistakes, which can be deadly, are increasing as health care becomes more complex.

By Melinda Beck
Sept. 25, 2016 7:00 p.m. ET

A patient in cardiac arrest was mistakenly not resuscitated because clinicians confused him with a patient who had a do-not-resuscitate order on file.

Another patient was given an okay to undergo surgery based on a different patient’s records and was found dead in his hospital room the next day.

Such patient-identification mix-ups are common...
Polarized connectors — beyond medical infrastructure

Standard header pins are easy to plug something in the wrong way:

Keyed, polarized plugs — only go in one way:
Types
design & evaluation
Iterative Process

- Design
- Implement
- Evaluate
Types

• “a program p has type T” means that we can tell that p evaluates to an appropriate value, “obviously” — without evaluating.

• 1 has type Int, 1+2 has type Int, 100! has type Int.

• This pipe is for oxygen.
Case Study Today

- **Java** is a mainstream and widely studied OO language.
- **Scala** is a language that fuses OO and FP.
- **DOT** is a type-theoretic foundation for Scala:
  - simplify Scala’s type system by desugaring into DOT
  - simplify type inference by relying on DOT
Some Papers for Today

• WadlerFest’16 The Essence of DOT
  by Amin & Grütter & Odersky & Stucki & Rompf

• OOPSLA’16 Type Soundness for DOT by Rompf & Amin

• OOPSLA’16 Java and Scala’s Type Systems are Unsound:
  The Existential Crisis of Null Pointers by Amin & Tate
in Java

class Pipe<T> {}
class Source<T> {}
class Assembly<T> {
    public Pipe<T> pipe;
    public Source<T> source;
}
class O2 {}
class NO2 {}
public class pipes {
    public static void main(String[] args) {
        Assembly<O2> o2 = new Assembly<O2>();
        o2.pipe = new Pipe<O2>();
        o2.source = new Source<O2>();
        o2.source = new Source<NO2>();
        // error: incompatible types:
        // Source<NO2> cannot be converted to Source<O2>
    }
}
class Pipe { type T }
class Source { type T }
class Assembly { self =>
    type T;
    var pipe: Pipe { type T = self.T };
    var source: Source { type T = self.T } }

class O2

class N20

object pipes extends App {
  val o2 = new Assembly { type T = O2; }
  o2.pipe = new Pipe { type T = O2; }
  o2.source = new Source { type T = O2; }
  o2.source = new Source { type T = N20; }
}
in Scala

```scala
o2.source = new Source { type T = N2O; }
// error: type mismatch
// found   : Source{type T = N2O}
// required: Source{type T = o2.T}
// (which expands to) Source{type T = O2}
```
Soundness

• “Well-typed programs can’t go wrong.”

• “hello”–1, “hello”.foo(1) goes wrong.

• If we have a pipe for oxygen, then the pipe can only be assembled with a source of oxygen.
Unsoundness in Scala

// fits in a tweet

trait A { type L <: Any
def id1(a: A, x: Any): a.L = x
val p: A { type L <: Nothing } = null
def id2(x: Any): Nothing = id1(p, x)
id2("oh")
class Unsound {
    static class Constrain<A, B extends A> {}
    static class Bind<A> {
        <B extends A>
        A upcast(Constrain<A,B> constrain, B b) {
            return b;
        }
    }
    static <T,U> U coerce(T t) {
        Constrain<U,? super T> constrain = null;
        Bind<U> bind = new Bind<U>();
        return bind.upcast(constrain, t);
    }
    public static void main(String[] args) {
        String zero = Unsound.<Integer,String>coerce(0);
    }
}
Unsoundness

- Exception in thread "main"
  java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.String

- http://io.livecode.ch/learn/namin/unsound
Unsound pipes

```java
Assembly<O2> o2 = new Assembly<O2>();
o2.pipe = new Pipe<O2>();
o2.source = Unsound.<Source<N2O>, Source<O2>>coerce(
    new Source<N2O>());
```
Dependent Object Types

- Minimal theory that can model
  - type parameterization
  - modules
  - objects and classes
  - subtyping
DOT Types

- $T$
- $\bot$
- $T \land T$
- $T \lor T$
- $L : S .. U$
- $m(x : S) : U$
- $x.L$
- $\{z \Rightarrow T\}$
DOT Types

- $\top$ (top)
- $\bot$ (bottom)
- $T \land T$ (intersection)
- $T \lor T$ (union)
- $L : S..U$ (type member)
- $m(x : S) : U$ (method member)
- $x.L$ (type selection)
- $\{z \Rightarrow T\}$ (recursive type)
DOT Types

- $\top$ (top) (subtyping lattice)
- $\bot$ (bottom) ...
- $T \land T$ (intersection) ...
- $T \lor T$ (union) ...
- $L : S..U$ (type member) (structural member types)
- $m(x : S) : U$ (method member) ...
- $x.L$ (type selection) (path-dependent types)
- $\{z \Rightarrow T\}$ (recursive type) (object types)
Subtyping Lattice

- A lattice has a greatest and lowest type, and for any two types, a least upper bound and greatest lower bound.
- Type inference needs GLB and LUB. e.g. if expression.
- In DOT, GLB = intersection, LUB = union. Well-formed by construction.
Structural Member Types

- Method members.
- By analogy, type members.
- Via intersection, DOT has structural record types. 
  \[(m1(x : S1) : U1) \& (m2(x : S2) : U2)\]
Path-Dependent Types

• Because DOT has type members, it also has an “elimination rule” for them, type selection. If an object o has a type member L, then the type selection o.L is a type, a path-dependent type.

• Type members are defined by a lower bound and an upper bound. Enables translucency.
Object Types

- Recursive self type.

- Objects with recursive self
  \( \{ z \Rightarrow (L: \bot..\top & (m1(x : z.L):z.L) & (m2(x:\top):z.L) \} \)

- Nominality, through a lower bound of bottom.
a DOT type

{ p =>
  type Pipe : ⊥ .. { type T: ⊥ .. ⊤ }
  type Source : ⊥ .. { type T: ⊥ .. ⊤ }
  type Assembly : ⊥ .. { z ⇒
    type T: ⊥ .. ⊤
    setPipe (x: p.Pipe ∧ { type T: z.T .. z.T }): ⊤
    setSource(x: p.Source ∧ { type T: z.T .. z.T }): ⊤
  }
  type O2: ⊥ .. ⊤
  type N2O: ⊥ .. ⊤
  newAssembly(x: {type T: ⊥..Τ}): z.Assembly ∧ {type T = x.T}
  newPipe    (x: {type T: ⊥..Τ}): z.Pipe     ∧ {type T = x.T}
  newSource  (x: {type T: ⊥..Τ}): z.Source   ∧ {type T = x.T}
}


In DOT terms

```javascript
let o2 = p.newAssembly(new {type = p.O2}) in
o2.setPipe (p.newPipe(new {type = p.O2}));
o2.setSource(p.newPipe(new {type = p.O2}));
o2.setSource(p.newPipe(new {type = p.N2O}));
```
Semantic Intuition

\[
\Gamma \vdash x : (L : S..U) \\
\frac{}{\Gamma \vdash S <: x.L <: U}
\]
Trouble 1
User–Definable Subtyping Theories

In DOT, the subtyping relation is given in part by user–definable definitions

type T <: S <= U
This makes T a supertype of S and a subtype of U. By transitivity, S <= U.

So the type definition above proves a subtype relationship which was potentially not provable before.
Bad Bounds

What if the bounds are non-sensical?

type T >: \top <: \bot
By the same argument as before, this implies that

\top <: \bot
Any <: Nothing // in Scala

Once we have that, again by transitivity we get S <: T for arbitrary S and T.
That is the subtyping relation collapses to a point.
Resolution
Evaluation

- PL evaluation of type systems:
  - soundness
  - decidability
  - optional? prescriptive or descriptive?
  - verbosity? type inference.
  - expressivity? powerful orthogonal concepts.
  - predictability? uniqueness of typing.
Evaluation

• HCI evaluation of type systems:
  • What is the user experience of types?
  • How learnable is the type system?
  • Once learned, how powerful is the language?
  • How consistent is it with expectations based on other programming and/or life experience?
Languages vs Users
Group Projects

- Systems HCI requiring heavy-duty PL
  - Humans modifying DSLs for PBD (programming by demonstration)
  - Examplore with interactively defined templates
- Generic human-centered PL
  - Pick language feature, design it in a human-friendly way
  - Pick a language, describe how—and to what extent—its features are being used in the wild
- Usable + X (PL technique)
  - Usable Generative Programming
  - Usable Probabilistic Programming
  - Usable Type System / Verification
  - Usable Synthesis
    - inductive bias alignment between human and machine
    - ranking function improvements
    - DSL improvements
    - expressing constraints on intermediate states, i.e., equivalence values or types
Thank you!
Evaluation

• PL evaluation of type systems
  additional mentioned in discussions
  
  • complexity
  
  • completeness (of algorithm wrt to formalism)
  
  • error handling
Modularity in Java?

• Define an interface, which says: we have some type ‘Key’ and some method ‘make’, which takes a ‘String’ and returns a ‘Key’.

• Implement this interface by a class, where type ‘Key’ is type ‘int’ and ‘make’ returns the ‘hashCode’ of the argument.

• Implement as a separate library, a method to ‘map’ the ‘make’ over a list of ‘String’s.
Modularity in OCaml (1)

module type GEN = sig
  type key
  val make : string -> key
end;;

module HashGen : GEN = struct
  type key = int
  let make = Hashtbl.hash
end;;
module MakeGenLib (Gen: GEN) = struct
  let mapKeys = List.map Gen.make
end;;

module HashGenLib = MakeGenLib(HashGen);;

let ex1 = HashGenLib.mapKeys ["Hello"; "World"];;
(* ex1 : HashGen.key list *)
Modularity in Scala

trait Gen {
  type Key
  def make(data: String): Key
}

object hashGen extends Gen {
  type Key = Int
  def make(s: String) = s.hashCode
}

def mapKeys(g: Gen, ss: List[String]): List[g.Key] =
  ss.map(g.make)
Translucency

\[ \Gamma \vdash x : (L : S..U) \quad \Gamma \vdash S <: x.L <: U \]

```scala
val abstracted: Gen { type Key } = hashGen
val transparent: Gen { type Key = Int } = hashGen
val upperBounded: Gen { type Key <: Int } = hashGen
val lowerBounded: Gen { type Key >: Int } = hashGen
(1: lowerBounded.Key)
(upperBounded.make("a"): Int)
```
trait Brand {
    type Hidden
    def pack(x: Int): Hidden
    def unpack(x: Hidden): Int
}

val brand: Brand = new Brand {
    type Hidden = Int
    def pack(x: Int): Hidden = x
    def unpack(x: Hidden): Int = x
}

brand.unpack(brand.pack(7)) // ok
brand.unpack(7) // not ok – but occurs during red.!