Newspeak & its Children: Avarice and Sloth

Gilad Bracha
Ministry of Truth

Functional Object-Oriented Programming

- No contradiction
  - FP is computational paradigm
  - OO is an organizational one

FOOP is Spreading

- OCaml/F#
- Scala

Pure FOOP

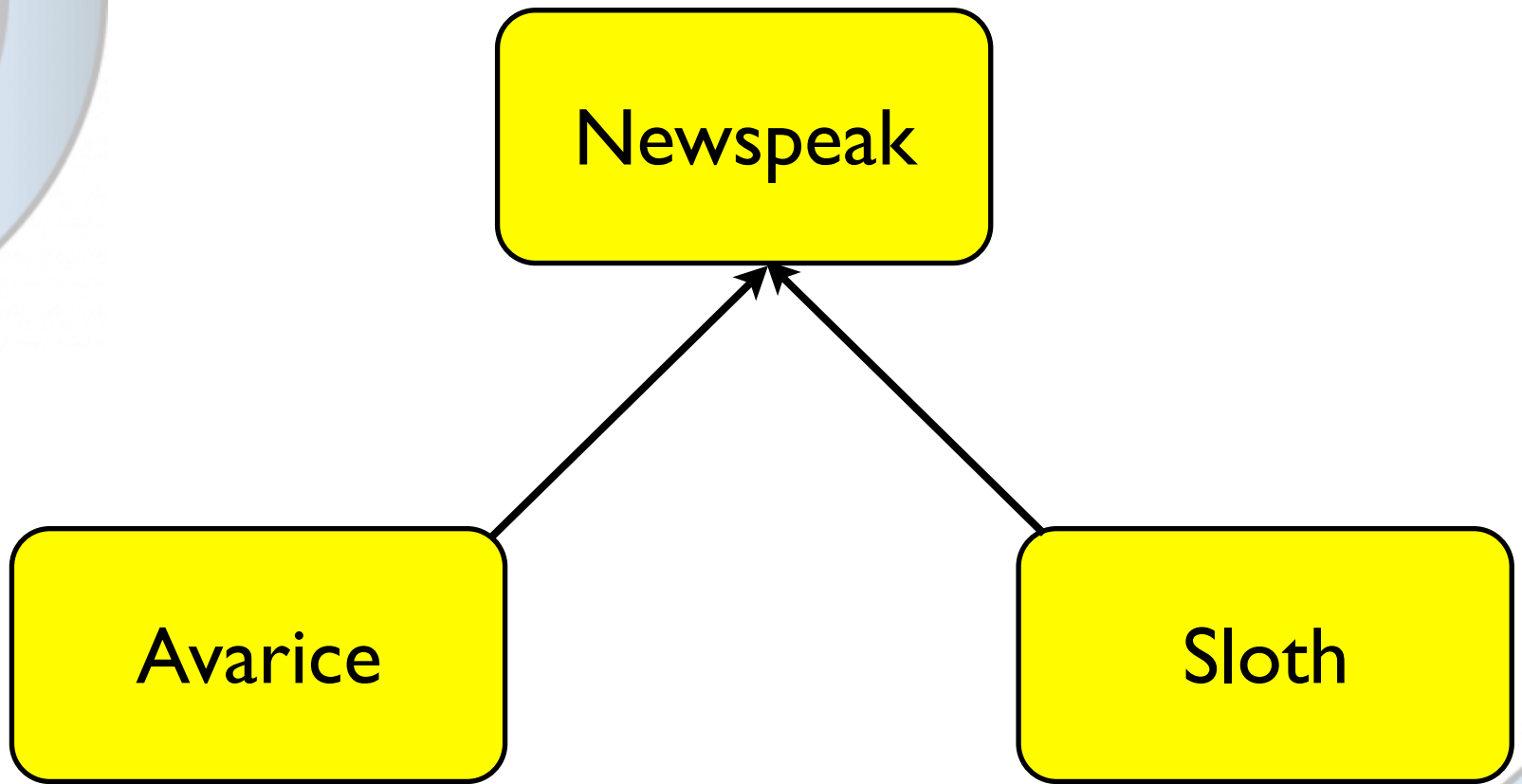
- Purely Object-Oriented
- Purely Functional

Purely Object-Oriented

- Everything is an Object

Purely Functional

- HOFs
- Referentially transparent
- Lazy?



Newspeak

- The “Mother of all sins”
- Mother and Sin share common OO structure

Newspeak

- Message based
- No global scope

Message-based

- All operations are “message sends”
  - All names are late bound, depending on their receiver

Program to an Interface not an Implementation

No References to Variables

Representation Independence

- Always use slots via accessors

id = letter, (letter | digit) star.

No References to Classes

- Always use accessors
- Classes are first class objects
  - Concepts are phenomena

No Global Scope

- As in lambda calculus
- Newspeak is almost as simple as a class calculus

Ban Imports!

- No packages, assemblies, modules, ...
- No imports, using clauses, ...
- Top level classes define modules

How does it work?

- Tools provide any required top level namespace
- Aggregation makes parameterization manageable at top level
- Nesting handles the rest

Nested Classes

- Nested as in Beta, not as in Java
  - Great for Modeling
  - Natural Modularity Solution

Message-based Nested Classes

- Classes are always virtual
- Classes are always mixins
- Class hierarchy inheritance

Practical Benefits

- Side-by-side deployment
- Polymorphism over entire libraries
- Security

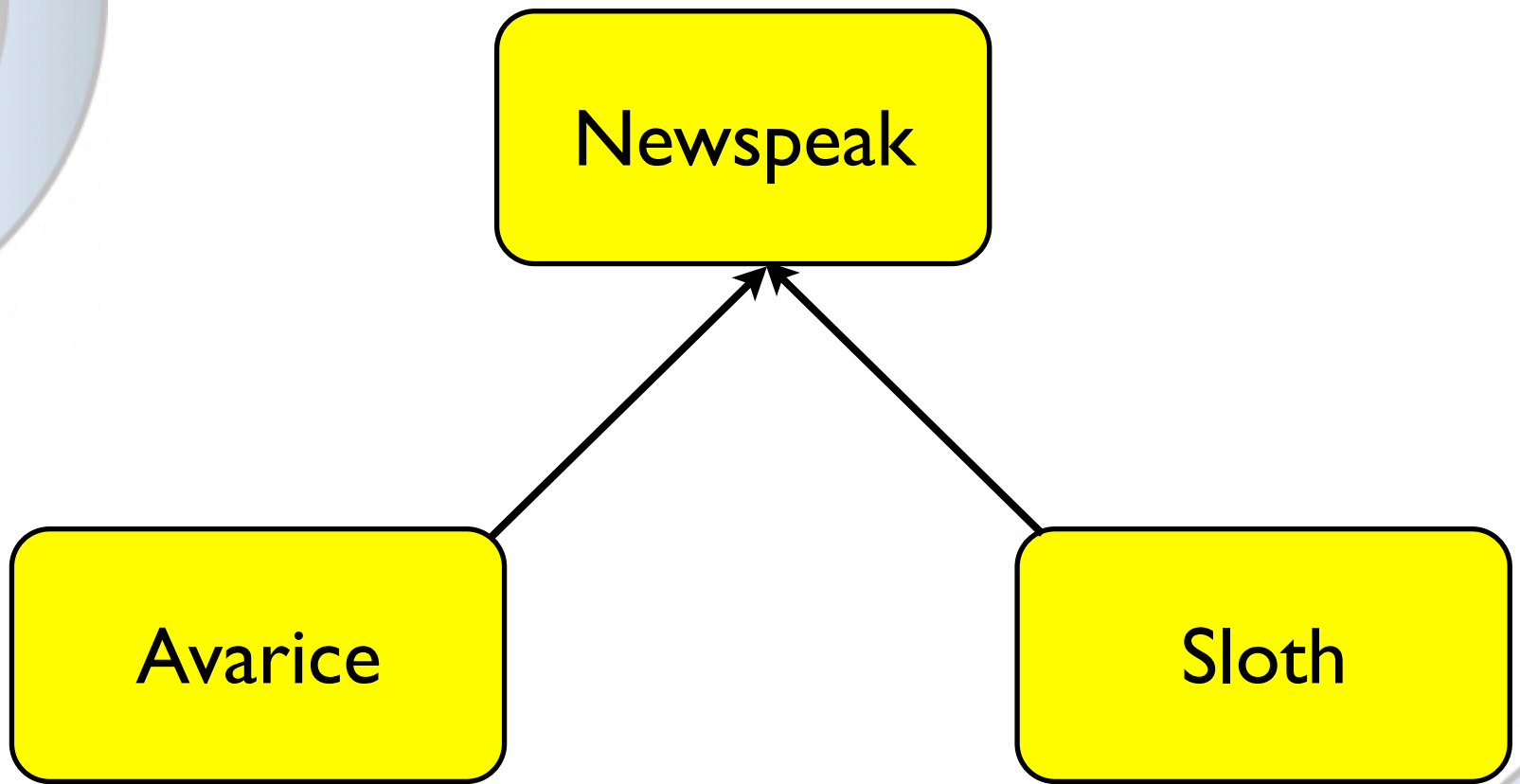
Security Implications

- *No Ambient Authority*
- Object capability model
  - Object reachability defines authority

Beyond Baked

Concurrency?

- Actors
- Value types
- A slippery slope into Avarice and Sloth



Newspeak vs. Avarice

```
class Foo {
        | x  y ::= 2 * x.  z = y + 3. |
} ...
```

Newspeak vs. Avarice

```
class Foo {
        | x  y ::= 2 * x.  z = y + 3. |
} ...
```

Newspeak vs. Avarice

```
class Foo {
        | x = 0.  y = 2 * x.  z = y + 3. |
} ...
```

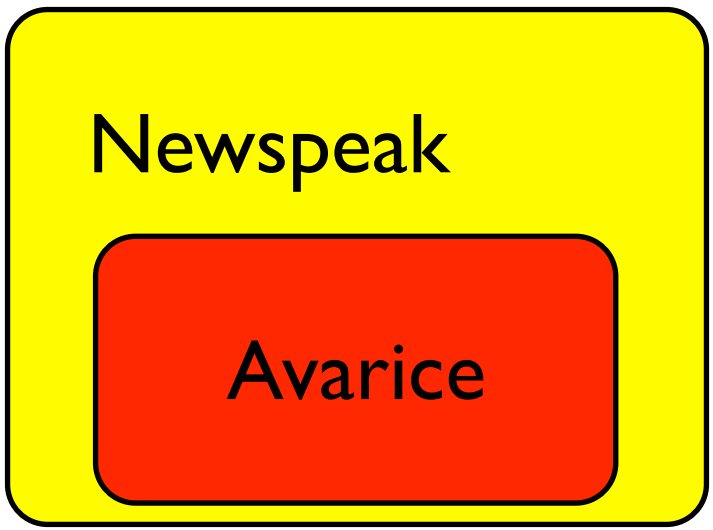
Newspeak vs. Avarice

```
class Foo {
        || x = 0.  y = 2 * x.  z = y + 3. ||
} ...
```

Newspeak vs. Avarice

- Eliminate mutable slot declarations
- Hide order of slot initialization
- Change libraries:
  - Object identity
  - Reflective modification
  - Application libraries

Avarice is a proper subset of Newspeak



Avarice and Sloth:

Same syntax, different semantics

Avarice	Sloth
Applicative order	Normal order

Pattern Matching?

- Only if we preserve data abstraction
- Everything is an object
  - First class patterns and queries

Typechecking?

- Pluggable type system planned
  - Type system is a priority - the lowest one
- Non-trivial
  - Classes, superclasses all dynamically bound

Status

- Available at *http://newpeaklanguage.org*
  - open source under Apache 2.0 license
- Work in Progress
  - Expect some tweaks to syntax and semantics
  - Implementation still not complete - especially libraries

Connections

- Mirrors
  - Self
  - Strongtalk, JDI, APT ... See OOPSLA 04
- No static
  - Scala
  - E

Connections

- Security
  - E (Miller 06)
  - Java
- Modules
  - Jigsaw, 1991
  - Units
  - ML
  - Fortress

Connections

- Message-based programming
  - Emerald, Trellis/Owl
  - Smalltalk
  - Self
- Virtual Types
  - Beta, gBeta, Scala, CaesarJ, Tribe...
- Hierarchy inheritance: Ossher & Harrisson 92, Cook 89

Credits

- Peter Ahe
- Vassili Bykov
- Yaron Kashai
- Bill Maddox
- Eliot Miranda