Meta-Implementation Protocol

Semantics + Reflection = First-Class Implementations

Turn your Lisp into a Meta-Platform

François-René Rideau, *TUNES Project*

Lightning Talk at the European Lisp Symposium, 2017-04-03

http://fare.tunes.org/files/cs/fci-els2017.pdf

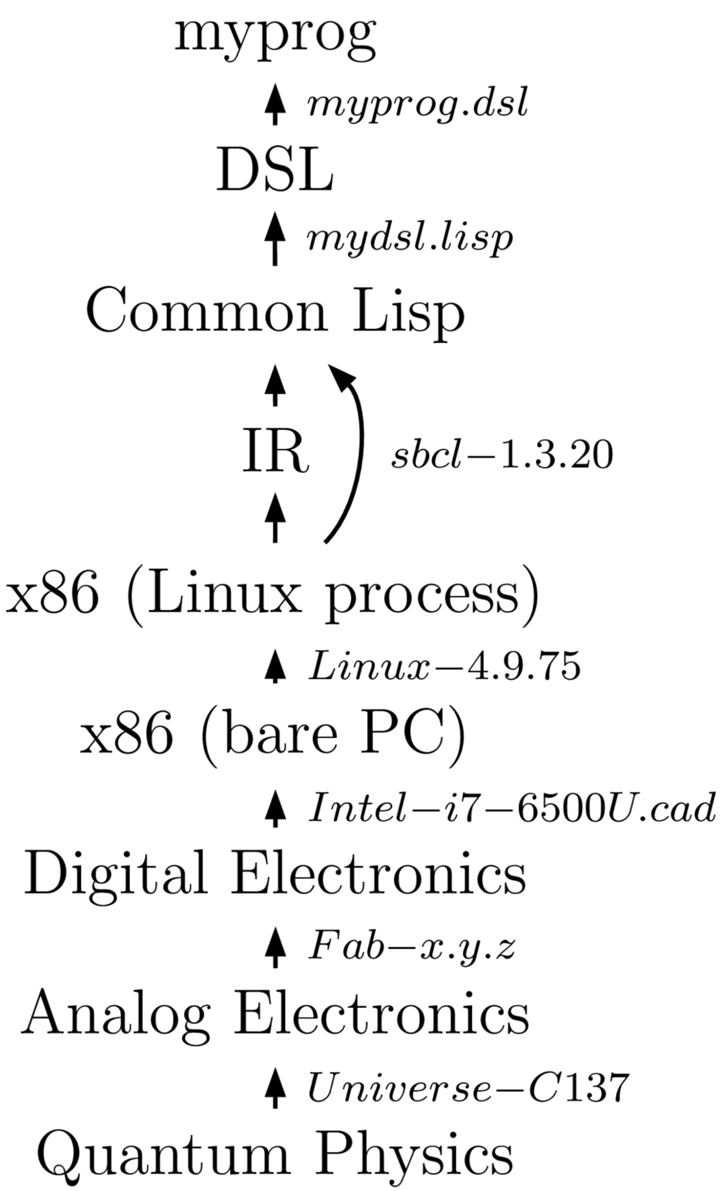
Basic Intuitions

Good programmers can mentally zoom in and out
of levels of abstraction

Interesting theorems allow you to change
your perspective on existing objects

What if these were not just think-time activities
but runtime capabilities of your system?

Semantic Tower



myprog

$myprog.dsl$

DSL

$mydsl.lisp$

Common Lisp

IR

$sbcl-1.3.20$

x86 (Linux process)

$Linux-4.9.75$

x86 (bare PC)

$Intel-i7-6500U.cad$

Digital Electronics

$Fab-x.y.z$

Analog Electronics

$Universe-C137$

Quantum Physics

Navigating, not mere debugging

Debugging

Local program state only

Only recover one level of abstraction

One way fixed magic operation

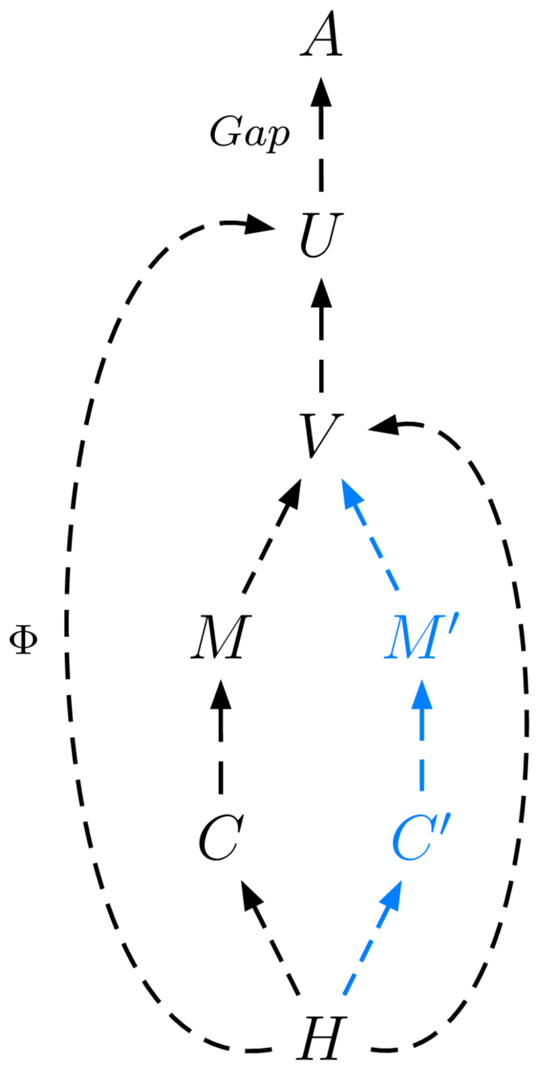
Navigating

Recurse to complete program state

Compose to recover any level you like

First-class operation both ways

Migration



When your hammer is Migration...

Process Migration

Garbage Collection

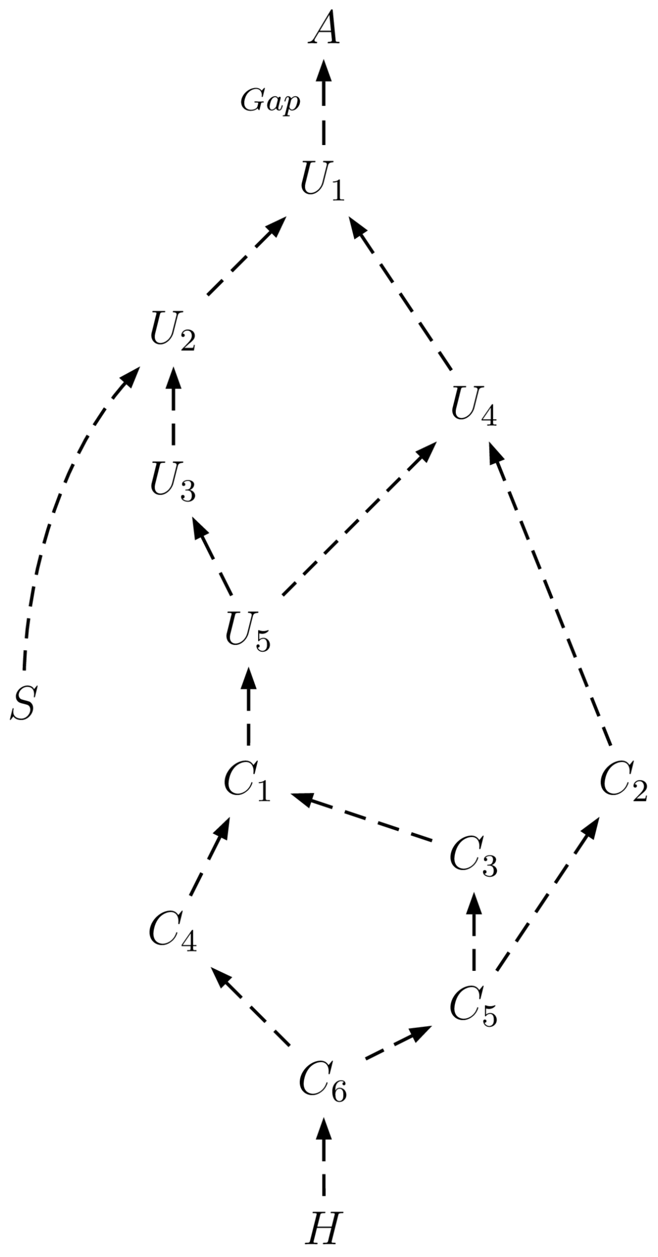
Zero Copy Routing

Dynamic Configuration

JIT Compilation

etc.

Semantic Towers need not be linear!



New Insights on...

Computation Semantics

Compilation

Semantics-preserving transformations

Aspect-Oriented Programming

Code Instrumentation

Virtualization

Computational Reflection

Software Architecture

Security

First-Class Implementations

Formalizing Implementations: Categories!

Observability: Neglected key concept — safe points

First-Class Implementations via Protocol Extraction

Explore the Semantic Tower — at runtime!

Principled Reflection: Migration

Natural Transformations generalize Instrumentation

Reflective Architecture: 3D Towers

Challenge

Put the "MIP" in your Lisp

Let’s change software architecture!

Thank you

My blog: *Houyhnhnm Computing*

http://ngnghm.github.io/