# Dare to Teach APL!

Aaron W. Hsu

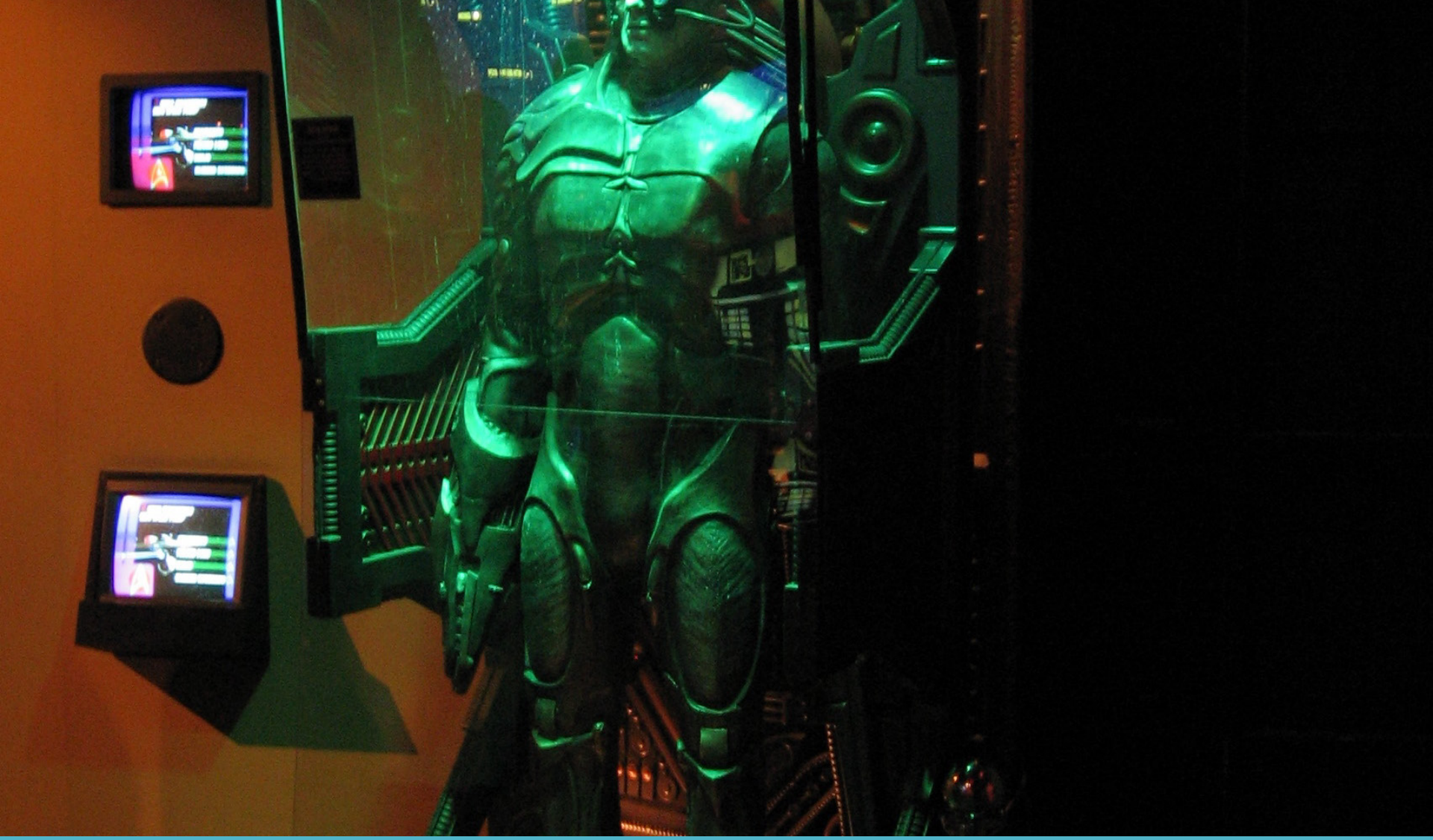
Indiana University

# Warning!

All results, statistics, and discussion of this project and in this presentation are extremely preliminary and should be understood in this context.

# What happens when…

You let them pick what they want to do, and you only teach them how to program in parallel, purely functionally, with none other than APL dfns?



Resistance is futile

# What is FSM?

Foundations in Science and Mathematics

AKA: Flying Spaghetti Monster, Finite State Machine

The FSM was an opportunity to create a completely original, ground-up re-envisioning of Computer Science Education:

How does syntax/semantics affect student learning?

What does it look like for students to know naught but parallel programming?

APL (dfns) uniquely suited to such a study.

# Can we funnel new talent into the University System?

We emphasized Women in Computing initiatives and targeted IU's Scheme-based, functional style curriculum for "future looking" plans.

# Major People Involved

Aaron W. Hsu (APLer, Schemer, masochist willing to mention APL to the faculty)

Jason Hemann (Apprenticed to Dan Friedman)

Jennie Lipson (IU Undergraduate Instructor for entry-level courses)

Anna Eilering (Pre-course preparation, information, and Robot obsessed)

# How do you handle enrollment?

Students are self selected, but the Computer Science course was opt-out rather than opt-in.

Enrollment strategies designed to compensate for traditional selection bias.

Our educational framework is clumsily and sloppily based on concepts appearing in Educational Research, including situated learning and constructivist techniques.

Put in NP speak, this means that we tried to make the work they did relevant and self-motivated based on their own interests and “context.”

We used little formal lecture and no linear curriculum for the design.

# Course Structure

DOMAINS

- Math
- Physics
- Biology
- Chemistry
- Image Manipulation
- Whatever ya want, kid…

APL, REFERENCE MATERIALS

- Mastering Dyalog APL (sort of)
- Custom “Guide”
  - Glossary of terms
  - Emphasized declarative programming
- Dyalog APL on RHEL Workstations
- Full environment, no special aids

# Do what you like

# Work on real problems

# Just code, socially

Declarative

Functional

Parallel

# We achieved significant minority enrollment

More than 1/3 were women

Enrollment process actively contributed to these #’s

# What we were worried about…

Symbols, symbols, symbols…

Console programming in the Linux environment

Slow, boring progress, lack of demonstrated interest in domains

Symbols, symbols, symbols…

# Students had zero tangible difficulties with APL symbols!

This was one of the most unexpected anecdotal experiences we had.

Every class had at least one student in each domain

# There appears to be little to no additional difficulty for parallel programming

While anecdotal, students were able to successfully accomplish a significant amount of work using nothing but pure functional, parallel programming.

While unsurprising, it is still disappointing to see that students appear to completely forget everything they ever learned, **ever**, when they enter the APL Session.

# This was an “unexpected” success

We encountered significant opposition to the use of APL

A few of the students are regular participants in the IU computing groups

A surprising number of students were able to complete more than one domain’s worth of work

Students by and large had fun, were engaged, and participated successfully in traditionally valuable Computer Science “norms”

# Write your own code

Unlike many approaches to high school C.S. we did not give them much code, and they wrote all of the code for their solutions, with almost zero scaffolding.

# Example Problems

Fractals

Blurring an animation repeatedly as an animation

Plotting the trajectory of a cannon ball

Developing a colorized version of the Game of Life

Writing a sprite oriented RPG game engine (Rogue-like)

A host of custom image manipulations designed by students

# Parallel programming has been mistaught

APL allowed us to expose students to a surprisingly wide variety of relatively advanced Computer Science concepts in the time it takes most classes to get to defining your first helper function.

The exposure was intentionally shallow but broad.

Students actually enjoyed themselves, usually.

# Future Work

Do these same techniques work differently if we use a different language?

Does this scale to a longer course?

Can we obtain any quantitative results?

How well can students transition into a traditional program?

Analysis of recorded screen sessions

# Thank you

Questions eagerly awaited….