

CS 295B/CS 395B
Systems for Knowledge
Discovery

Industry: KDD at Scale



The University of Vermont

This Week

Two industrial papers

- Tensorflow (Google)
- TLA+ @ Amazon



Did you build the right thing?
(Validation)

Verification &
Validation
at scale

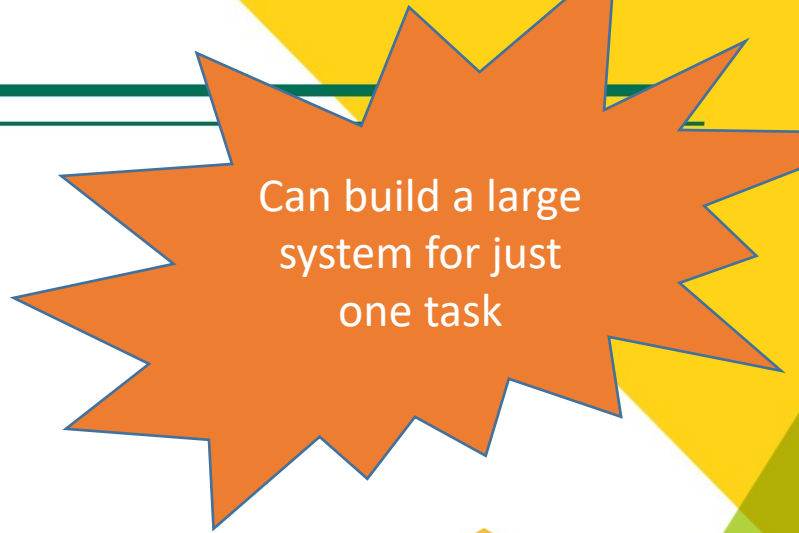
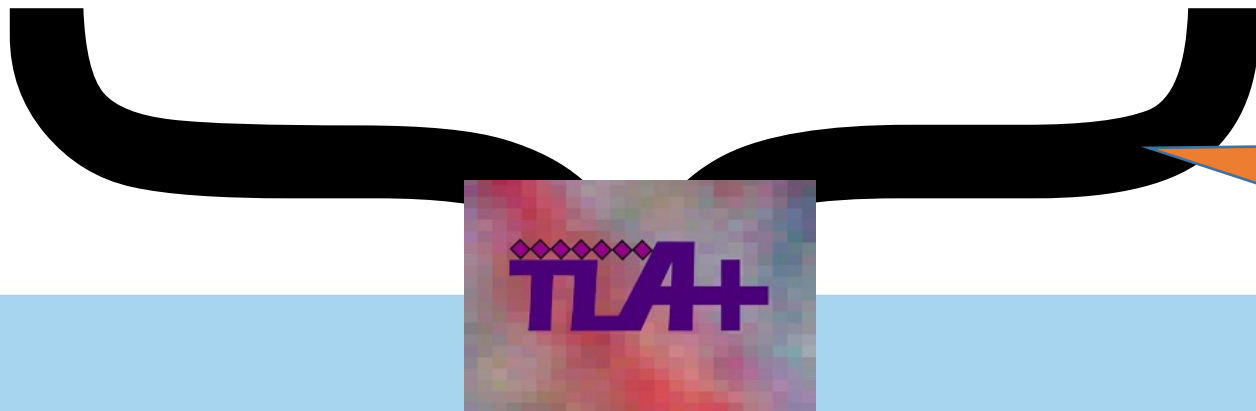
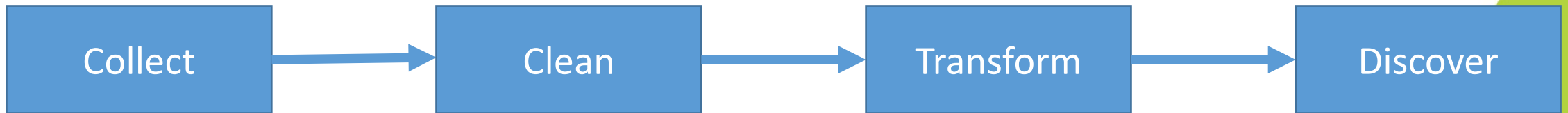
Did you build the thing
right?
(Verification)



System Design...for KDD?

Most systems-y papers we have seen so far

Where's the KDD?



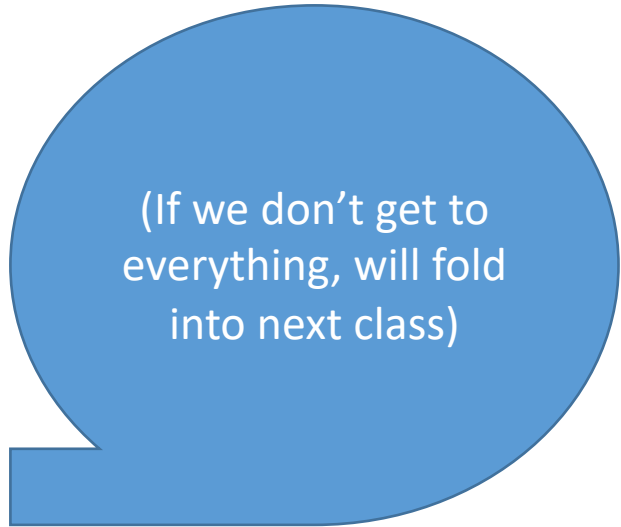
Topics for today

Technical Background

- System architecture (both, but mostly Tensorflow)
- Axiomatic semantics (TLA+)
- Temporal Logic (TLA+)

Context

Industrial systems: The Good, The Bad, and The Ugly



(If we don't get to everything, will fold into next class)

Technical Background

System architecture

Local architecture (local Tensorflow)

Cluster architecture (distributed Tensorflow)

Large-scale distributed systems (Amazon&TLA+)

Theme: KDD at scale

- AWS systems power KDD for both end-users and Amazon
- Tensorflow powers discovery

Classical compilation:
High level (e.g., C) → Assembly
(this diagram) → Bytes (gates)

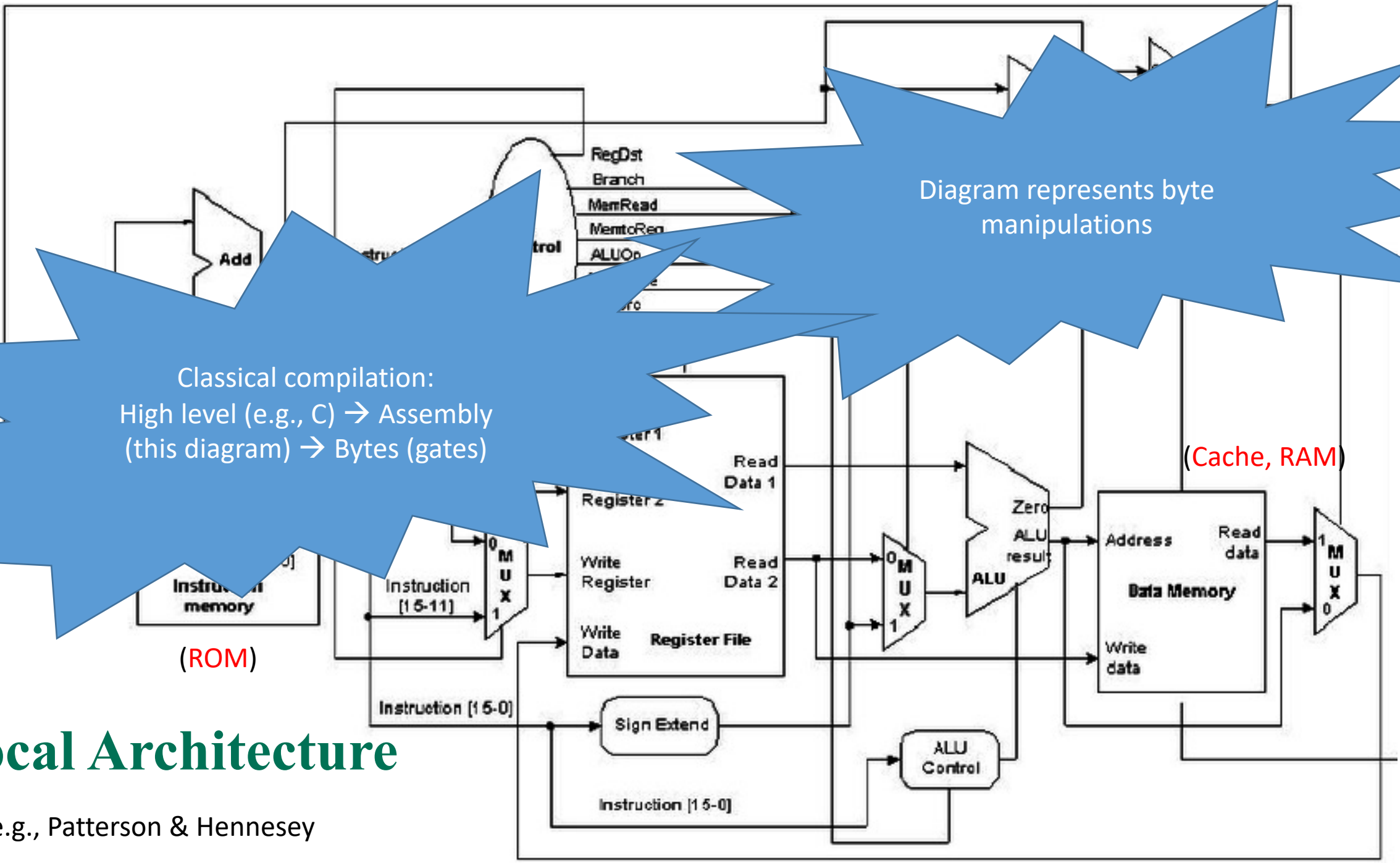
Diagram represents byte manipulations

(ROM)

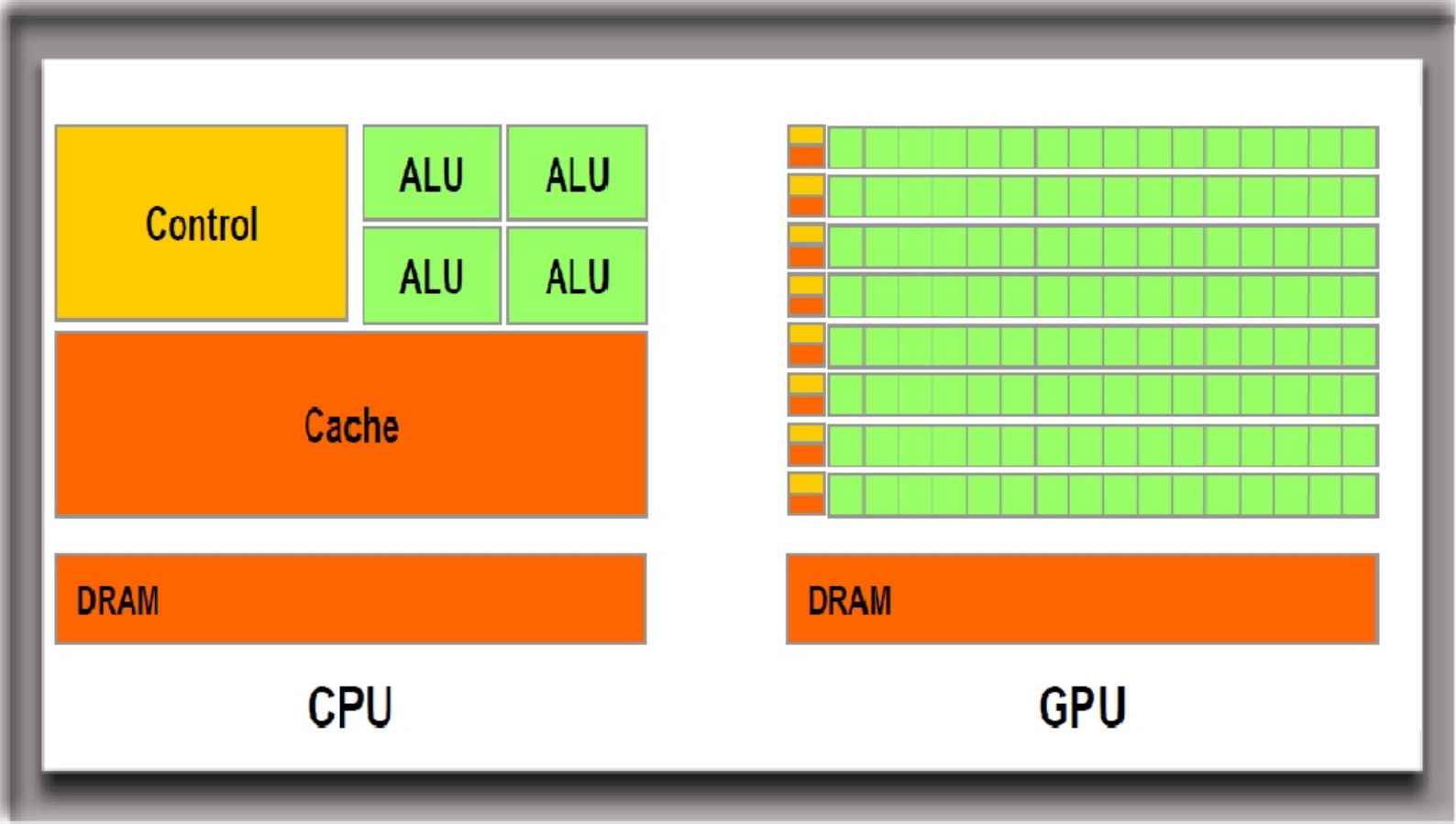
(Cache, RAM)

Local Architecture

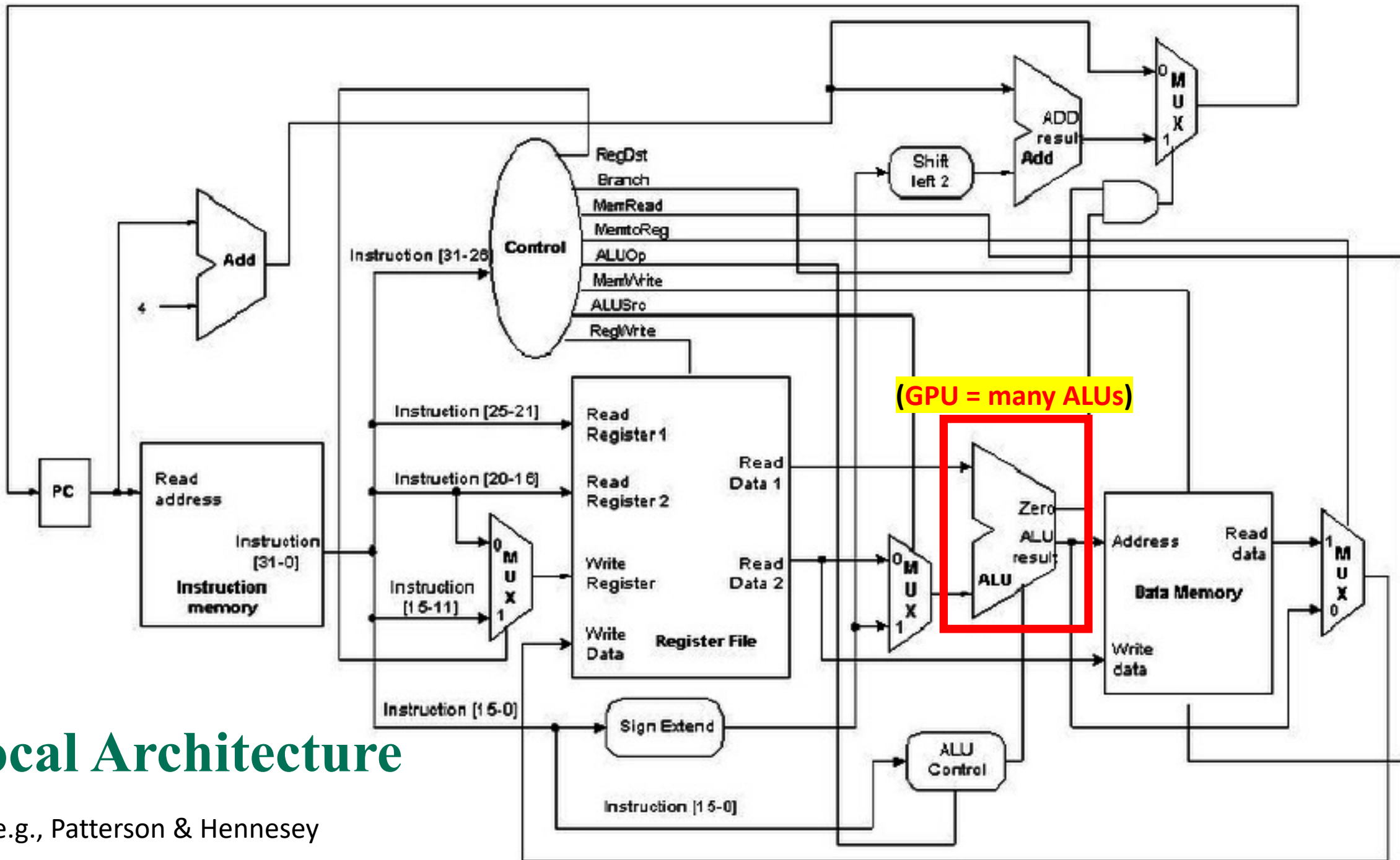
e.g., Patterson & Hennesey



Local with GPU

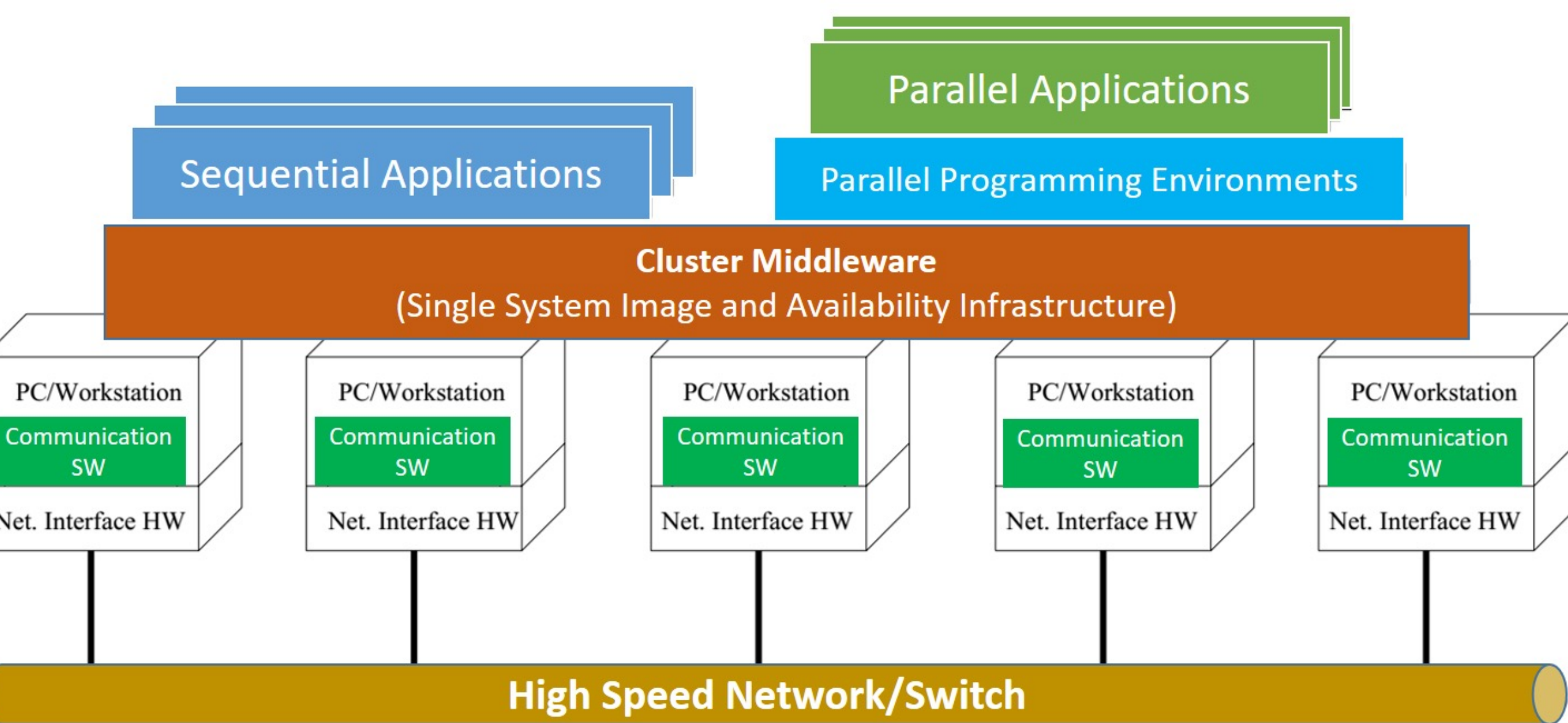


<https://www.omnisci.com/technical-glossary/cpu-vs-gpu>



Local Architecture

e.g., Patterson & Hennesey



<https://www.oreilly.com/library/view/distributed-computing-in/9781787126992/c8b8695b-fce0-4f1f-8c37-2a211fa314fa.xhtml>

Computer Cluster Architecture

Distributed Architecture

- Similar to cluster, but even more loosely connected
- Coordination a bigger problem due to fewer guarantees
 - Biggest challenge: locality
 - Think: MapReduce

Machine Learning Tasks as Architecture (TensorFlow)

Most ML focuses on math, proofs, etc.

Goal: design a system to execute math

Need to think in terms of components:

- What they do, how they connect (language)
- How to optimally compute them so people will actually use them
 - What pieces do they operate over?
 - How do these logical, mathematical components map to physical/operational components?



TF discussion
of parameter
store & global
state as
coordination

Reasoning over Distributed Systems (TLA+)

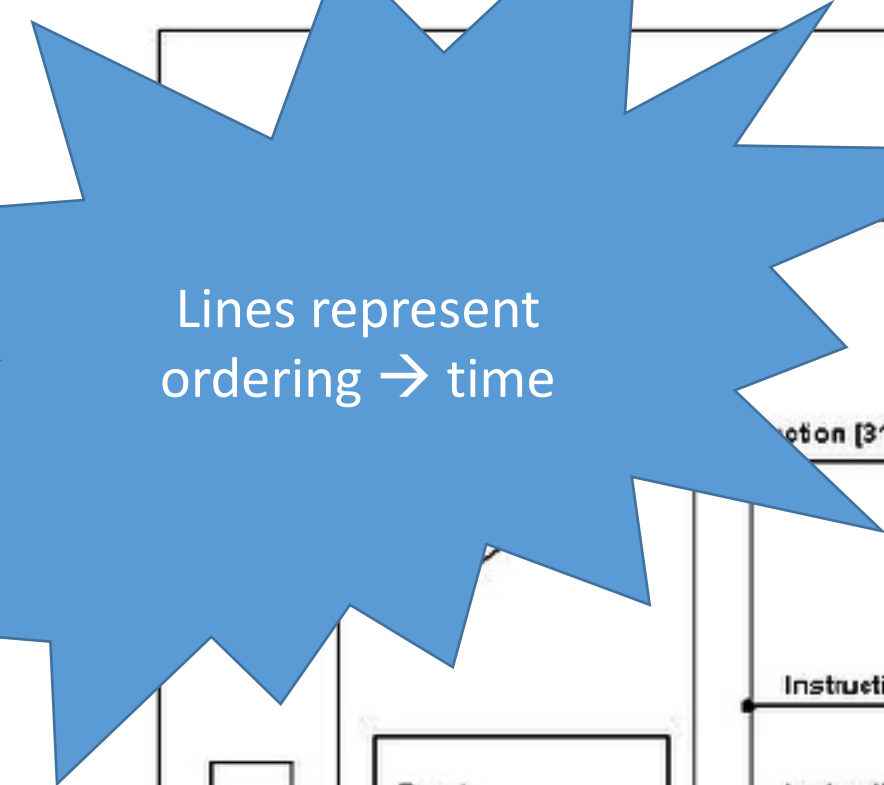
How do we reason at this level of abstraction?

Background: Axiomatic Semantics

Three types of semantics:

- Denotational (what does this mean?)
- Operational (what does this this do?)
- Axiomatic (what needs to be true?)

Do stuff on the board

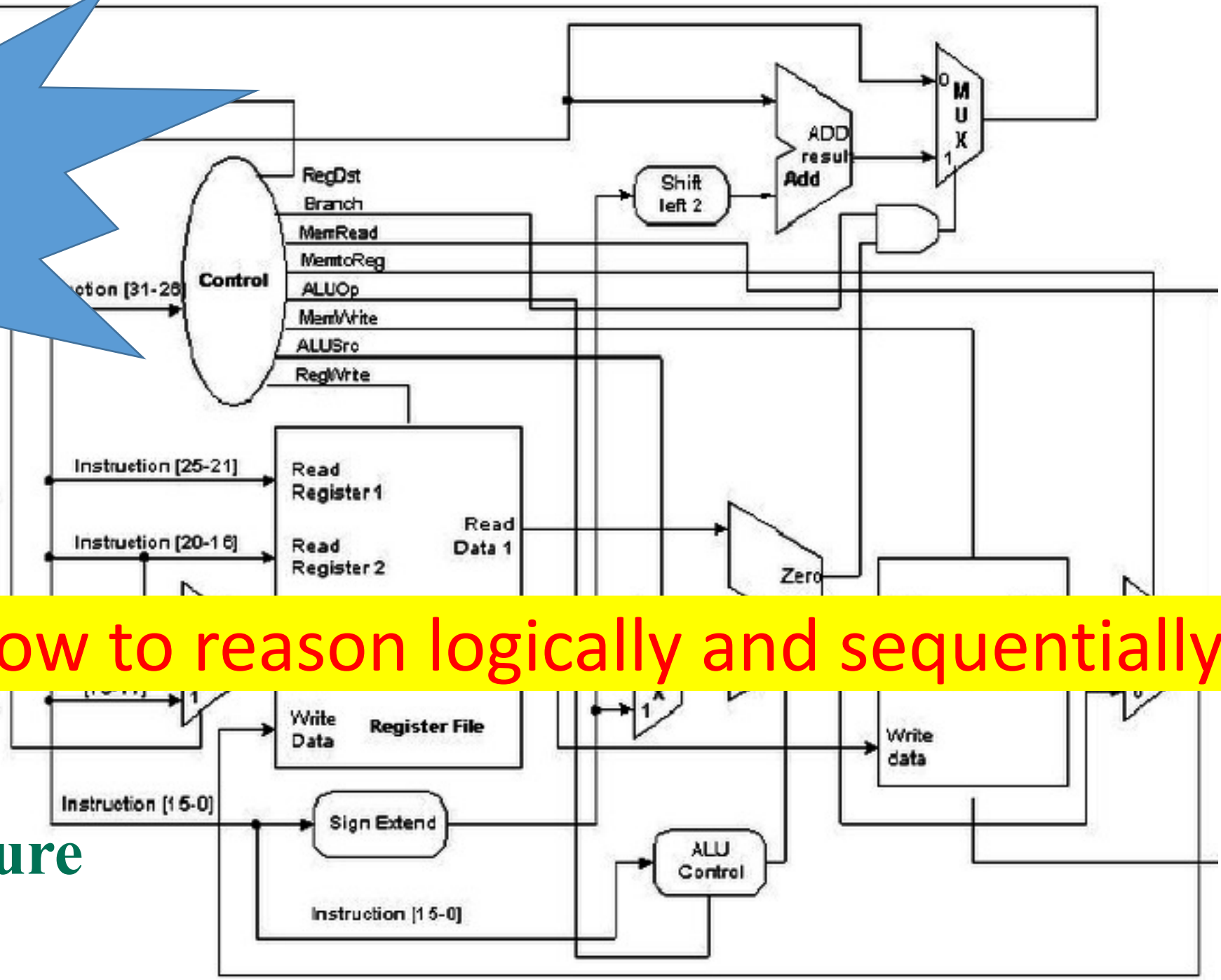


Lines represent ordering → time

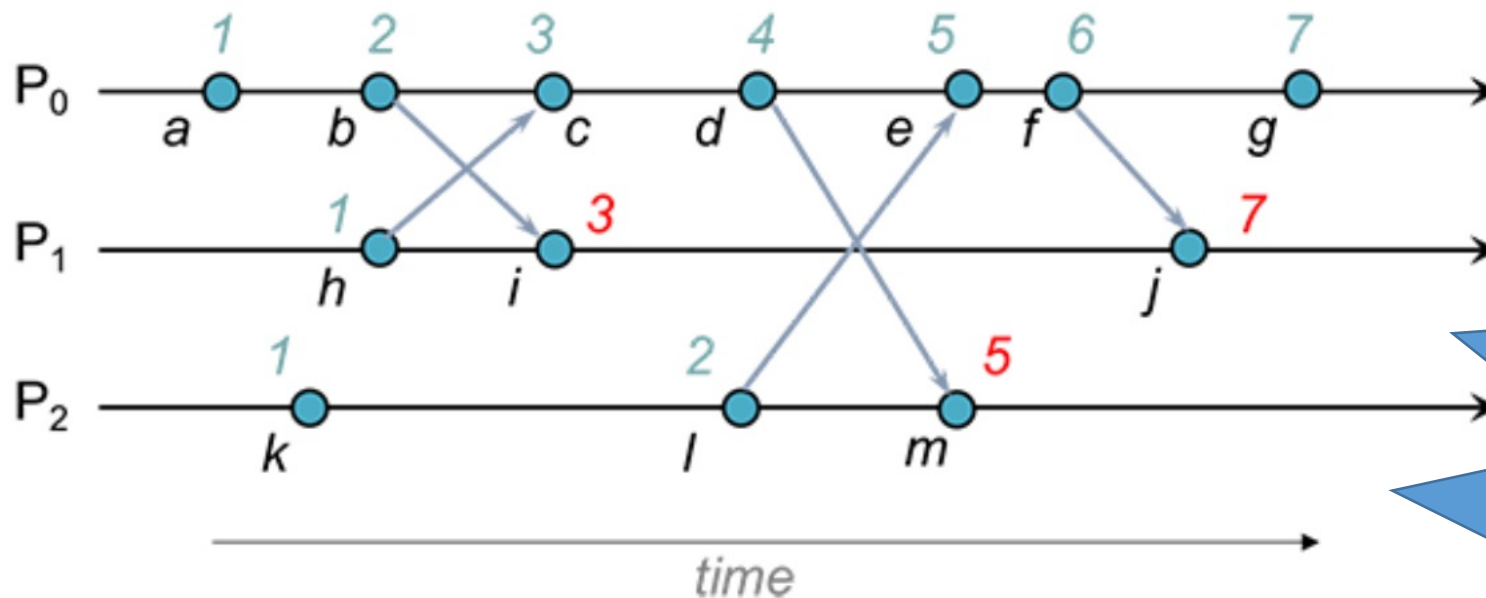
How to reason logically and sequentially?

Local Architecture

e.g., Patterson & Hennesey



Background: Vector Clocks and Partial Orders



Lamport Clock Assignment

<https://people.cs.rutgers.edu/~pxk/417/notes/logical-clocks.html>

“happens-before”
relation

Background: Temporal logic

“TLA” == “Temporal Logic of Actions”

Recall: axiomatic semantics → reasoning over predicates

Idea: express those predicates in terms of *logical temporal expressions*

Two major families:

- linear (LTL: linear temporal logic)
- branching (CTL: computation tree logic)



Path-based; most common

Do stuff on the board

LTL: Example (Until)

$P \text{ U } Q \iff$ "P is true until Q is true"

- $P \iff$ "file F is not writable"
- $Q \iff$ "file F is closed"
- $P \text{ U } Q \iff$ "file F is not writable until file F is closed"

Can map to database transactions

Big Idea

Axiomatic Semantics + Temporal Logic
are the appropriate level of **granularity**
for large-scale systems

Context

Tensorflow

Simultaneous discovery as a scientific phenomenon

- 2009-2012 – low-level languages for NN
- OSDI 2016 – presented two weeks before the PLDI 2016 deadline
- Several submitted NN compilers (one accepted)
- Why did Tensorflow win?
 - Great documentation from Google
 - Now: Torch (supported by Facebook)

Latte: A Language, Compiler, and Runtime for Elegant and Efficient Deep Neural Networks

Leonard Truong
Intel Labs / UC Berkeley, USA
leonard.truong@intel.com

Rajkishore Barik
Intel Labs, USA
rajkishore.barik@intel.com

Ehsan Toton
Intel Labs, USA
ehsan.totoni@intel.com

Hai Liu
Intel Labs, USA
hai.liu@intel.com

Chick Markley
UC Berkeley, USA
chick@berkeley.edu

Armando Fox
UC Berkeley, USA
fox@cs.berkeley.edu

Tatiana Shpeisman
Intel Labs, USA
tatiana.shpeisman@intel.com

Abstract

Deep neural networks (DNNs) have undergone a surge in popularity with consistent advances in the state of the art for tasks including image recognition, natural language processing, and speech recognition. The computationally expensive nature of these networks has led to the proliferation of implementations that sacrifice abstraction for high performance. In this paper, we present Latte, a domain-specific language for DNNs that provides a natural abstraction for specifying new layers without sacrificing performance. Users of Latte express DNNs as *ensembles of neurons with connections* between them. The Latte compiler synthesizes a program based on the user specification, applies a suite of domain-specific and general optimizations, and emits efficient machine code for heterogeneous architectures. Latte also includes a communication runtime for distributed memory data-parallelism. Using networks described using Latte, we demonstrate 3-6 \times speedup over Caffe (C++/MKL) on the three state-of-the-art ImageNet models executing on an Intel Xeon E5-2699 v3 x86 CPU.

Categories and Subject Descriptors D.3.4 [Programming Languages]: Processors—Code generation, Compilers, Optimization

Keywords Deep Learning, Neural Networks, Domain Specific Language, Compiler, Optimization

1. Introduction

Applications such as image processing, video processing, and speech recognition have become ubiquitous in modern computing. This has driven the development of algorithms that can automatically analyze their content to perform tasks such as online web search and contextual advertising. Recently, neural networks have demonstrated state-of-the-art results for the tasks of understanding images, videos, and speech. Apple Siri, Google Now, Microsoft Cortana, Microsoft Bing, and Amazon Echo all use deep neural networks (DNNs) under-the-hood.

Deep neural networks are a class of models that use a system of interconnected neurons to estimate or approximate functions. In practice these networks are built with groups of neurons called *layers* with an input layer and an output layer corresponding to inputs and outputs of the function being estimated. Between the input and output layers are *hidden layers* that perform intermediate computation. A neural network *architecture* describes a neural network configuration with a specific set of layers.

Advances in deep learning research are largely driven by the introduction of novel hidden layers and architectures. Examples of recent advancements in the state of the art that stem from novel layers and architectures are the Inception Architecture [44], the PReLU layer [26], and Batch Normal-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
PLDI'16, June 13-17, 2016, Santa Barbara, CA, USA
© 2016 ACM. 978-1-4503-4261-2/16/06...\$15.00
<http://dx.doi.org/10.1145/2908080.2908115>

TLA+ for Amazon

- Several prior whitepapers on formal methods
- Iterations through tools (initially, Alloy?)
- This paper: most accessible, but elides much detail
- Sparked interest in formal methods more broadly

TLA+ for startups (part 1)

Why formal specifications are not a completely insane idea for startups and small teams



Neil O'Connor [Follow](#) [✉](#)

Nov 25, 2019 · 9 min read



This is the start of my personal and professional journey into the formal specification language TLA+. Before we dive into the details, I should probably begin by explaining my motivation for creating this series of articles.

Startup life: right thing, right time, done right

In my professional life, I build and run software development teams that create business systems using some of the latest technologies and methodologies. The systems we build are complex, but not Apollo programme or nuclear power plant complex. Thousands of lives may not depend on us writing robust, reliable and correct code; but if we write brittle, unreliable and faulty code, the company's reputation will suffer, our clients' reputation will suffer, and we will lose money.

The quality of the systems we build could make the difference between us winning in our chosen business domain and failing, badly. As I work in startups, marginal losses from poorly thought-out business strategies, poor tactical choices, or badly written code can take you very quickly indeed to the end of the road.

Current Role of Modern Industrial Systems Research

What industry adds

Scale – simply more users/machines

Applications/Applicability – ability to evaluate over more data/scenarios

New problems – scale + real-life use can highlight problems not seen before

Engineering clout – i.e., why these systems are actually used

Money – both internally (to fund this work) and externally (academic grants)

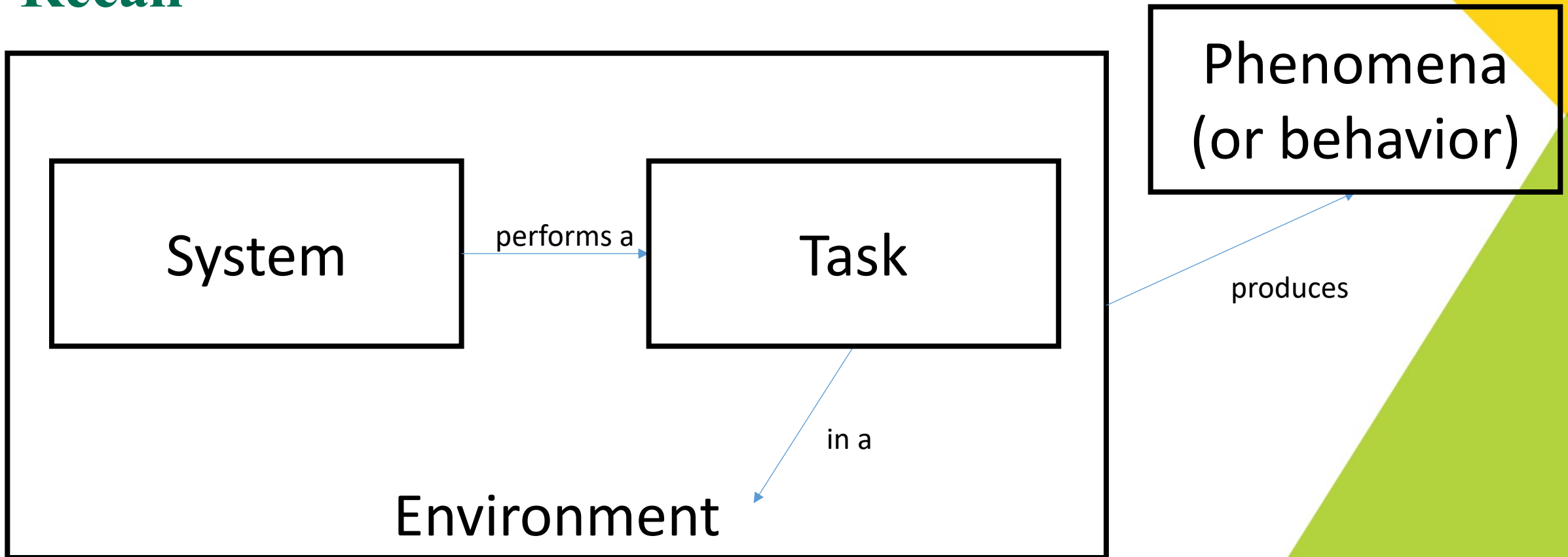
Industrial complications

Money – a blessing and a curse

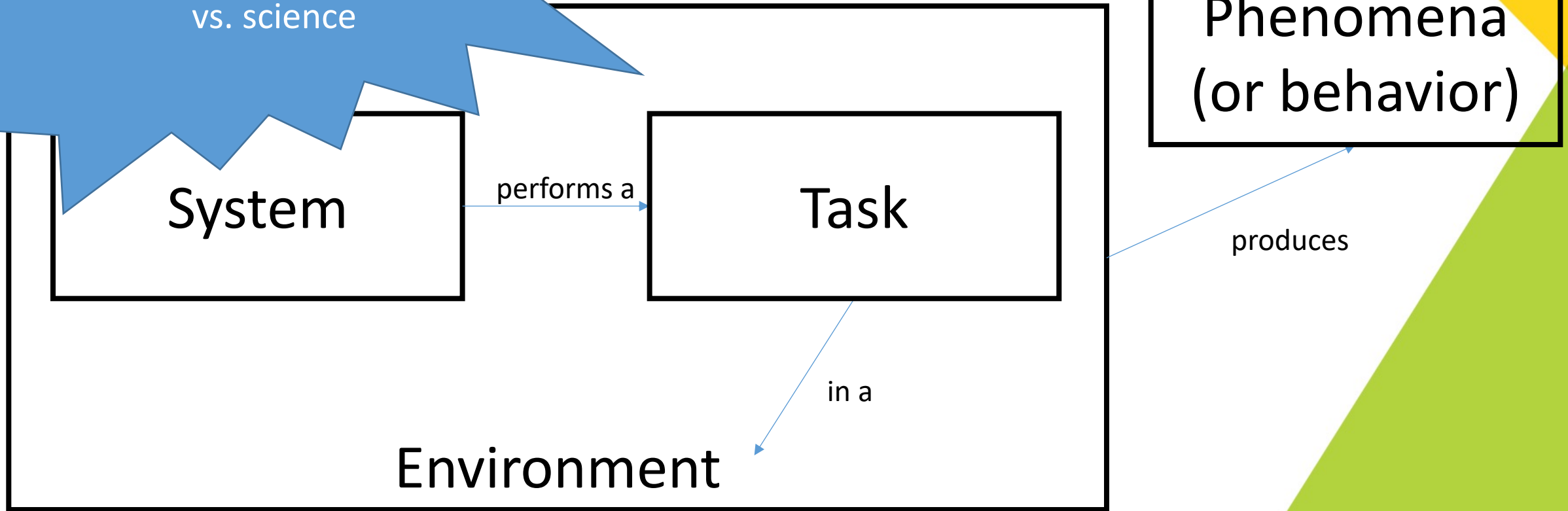
- Resource that powers all research (unless you are independently wealthy)
- Can drive interest away from basic research toward applied

Boards of directors, public relations can influence study of phenomena

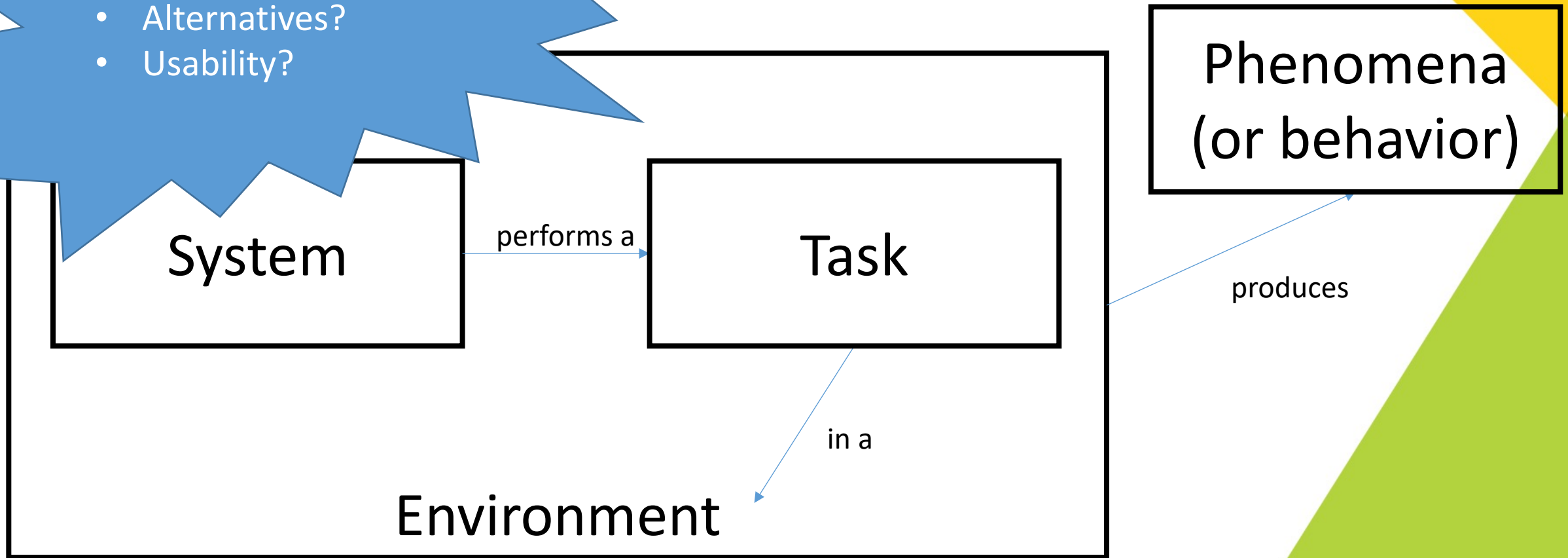
Recall



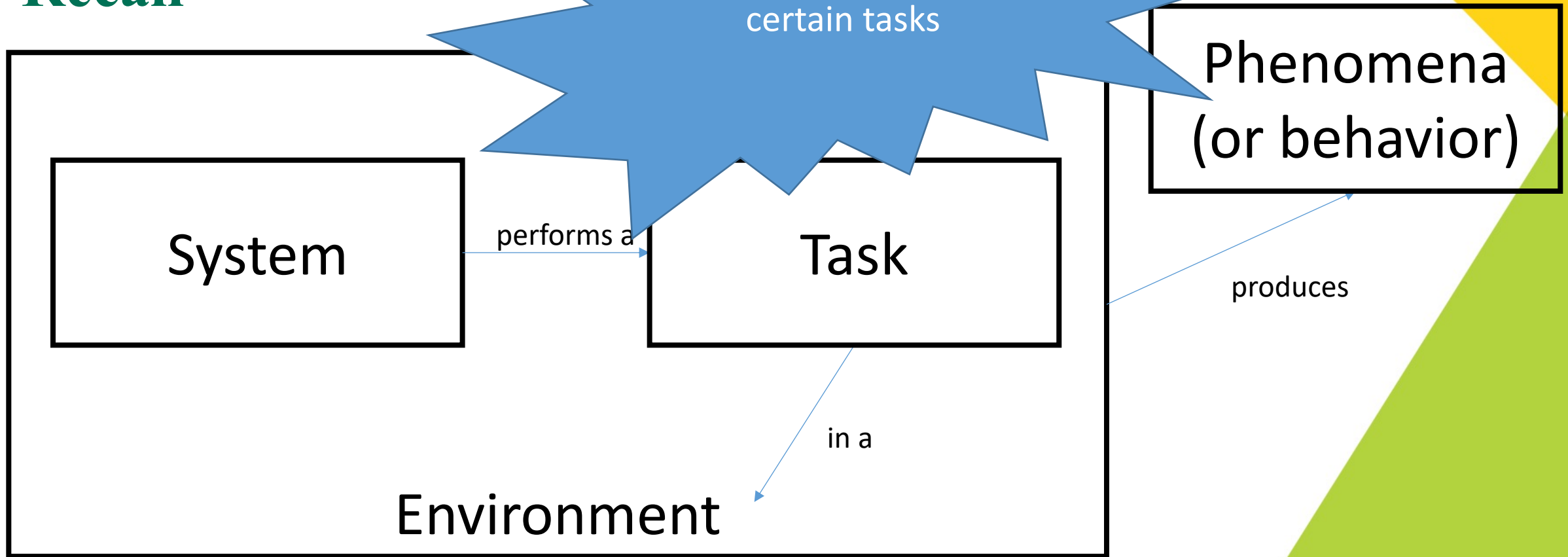
Funding to build
systems: engineering
vs. science



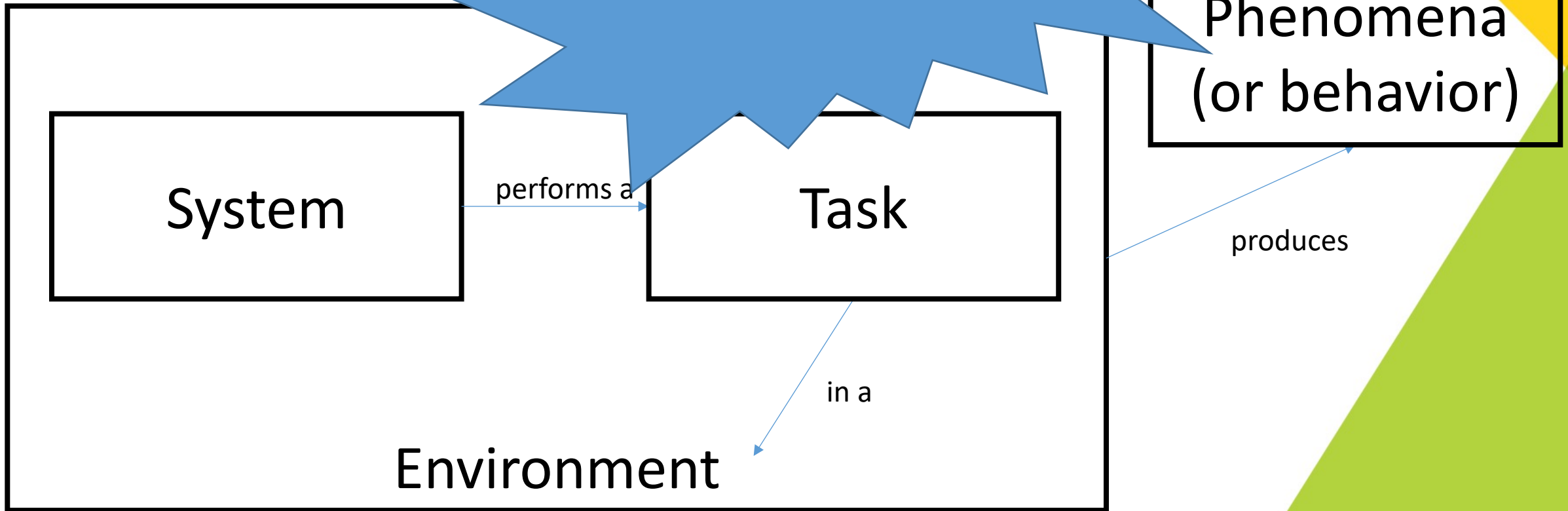
- Open source?
- Alternatives?
- Usability?



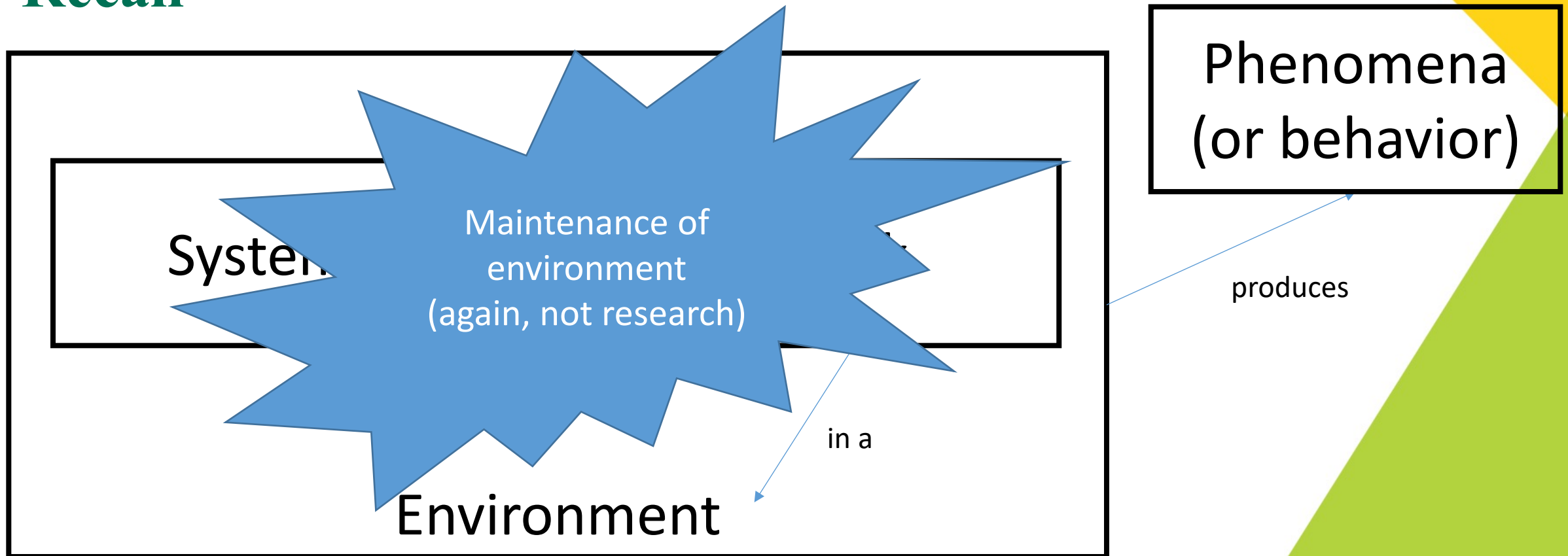
Recall



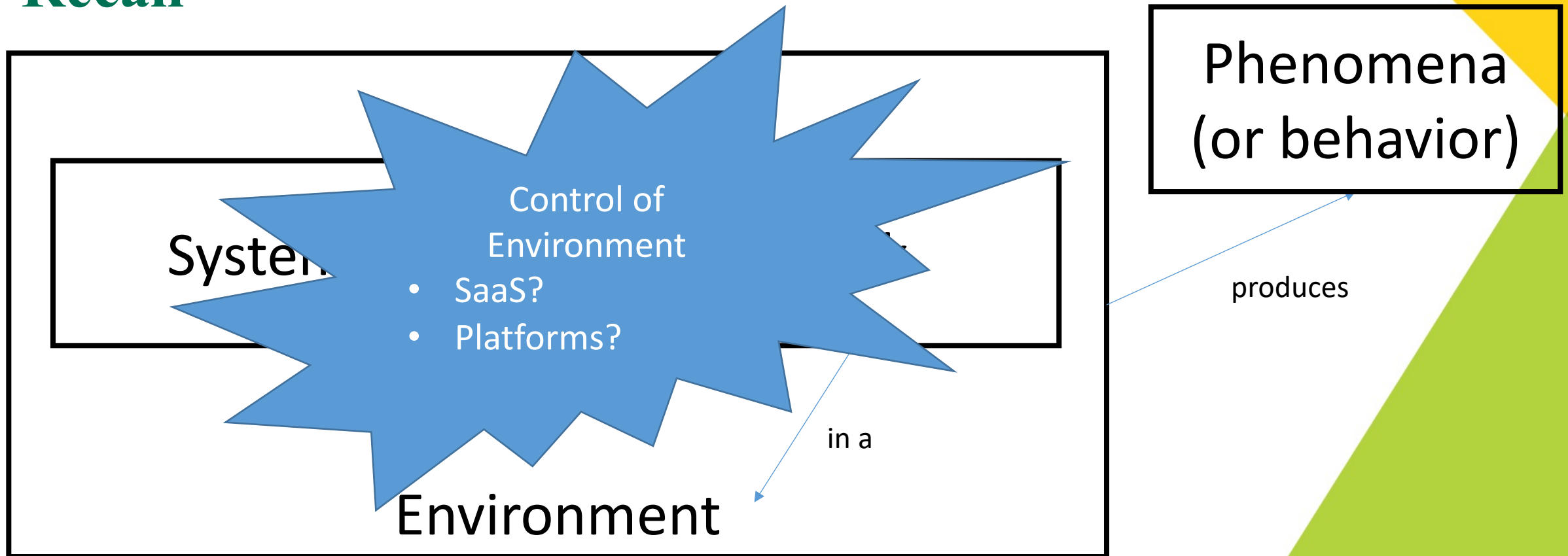
Recall



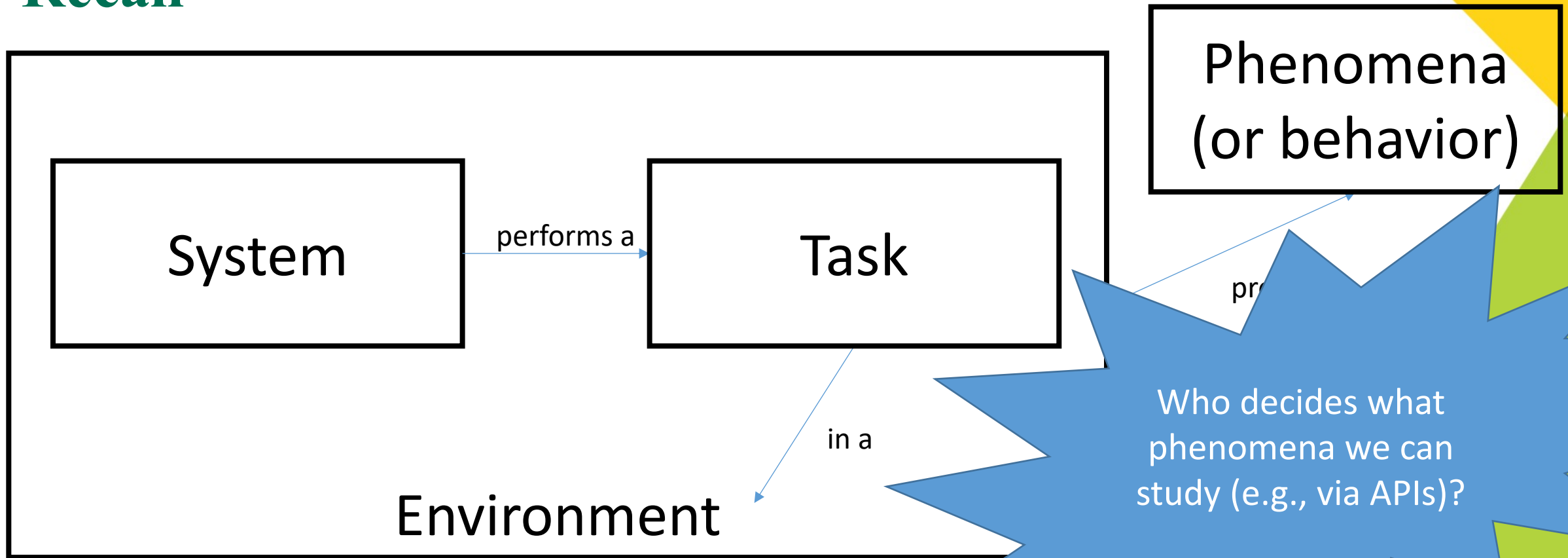
Recall



Recall



Recall

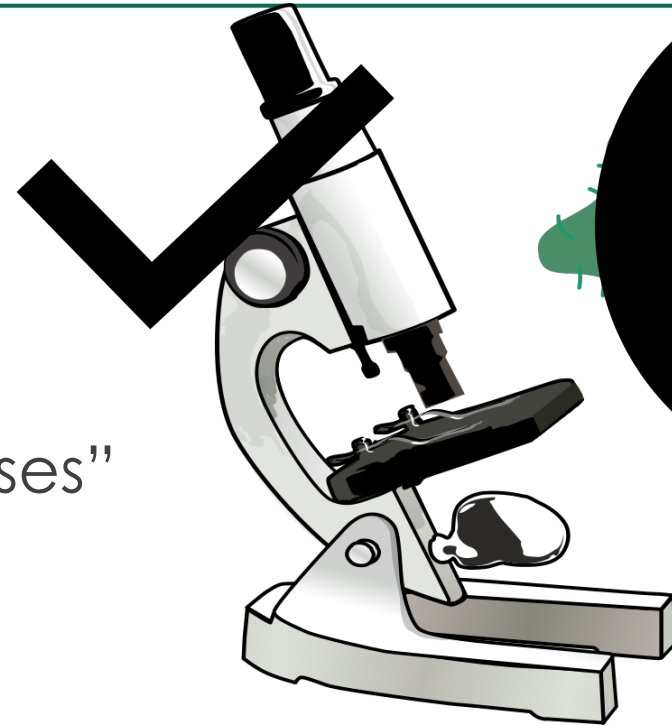


Threats

- Limits on questions (e.g., phenomena)
- Non-reproducible methods (who controls the software/platforms?)
 - Is this actually bad? (see: CERN)
- No reason to register hypotheses
- Publicity machines
 - Undermines double-blind
 - Early press releases set unrealistic expectations
 - Problems with communicating with the public

Reminder: about studying tools

- KDD = “Knowledge Discovery in Databases”
- KDD subsumed by “data science”
- *I will **not** be teaching data analytics, **data science**, nor data mining in this course. **Instead** we will focus on **tool support** for these tasks and discuss how to design and augment existing systems, specifically for **data collection** tasks.*



Reminder: about studying tools

Current discourse around industrial research focuses on phenomena.

- KDD = “Knowledge Discovery in Databases”
- KDD subsumed by “data science”

• I will **not** be teaching data analytics, **data science**, nor data mining in

Less discourse around tools.

this course. I will be discussing how to design and augment existing systems, specifically for

data collection tasks.



Models of Science in Industry: How much of a wall?

- **Bell Labs** – fairly strong division between basic and applied research
 - Today: similar to government contractors (MITRE, Leidos, BBN, SRI, etc.)
- **MSR** – pre-2014ish: strong division (esp. for RiSE)
 - Today: MSRNext more integrated
- **Google** – pre-2014ish: strongly integrated with applications
 - Today: Google Brain, DeepMind less integrated
- **Facebook** – tends to mimic Google

Reflect

Can you do good science in an industrial setting?

- What safeguards exist?

Can you do good science in a for-profit context?

- What are the incentives?

Next class: future of industry-academia-public/government research

Requests

Please submit your reviews by midnight tomorrow!

- I read these at 7am and it makes me very sad when there are max 3 entered at that time. 😭

Leviathan Falls is released Tuesday

- I will be off social media, not reading news
- **Spoilers will be met with proportionate response**

