0

Context and Motivation

0.1. Learning in proper context

This book is about preparing people to learn about a computaitonal approach to interpret the world. As computing devices are becoming ever more pervasive, we started to realize that computation as a kind of process is actually taking place everywhere, all the time. This view has been explicitly stated by well-known proponent of physical computing, such as Seth Lloyd[Programming the Universe]. We want to use this book as a way to lead readers to build computing devices as also to create an operational discipline that helps people to use the knowledge of computing science to better construct systems. In other words, we believe that all systems are a part of a bigger computing system, and they all follow a similar mechanism to manage their complexity. Due to this self-similarity, we believe that all systems can learn from their environmentally generated data. This idea has also been stated explicitly by Leslie Valiant's Probably Approximately Correct (PAC) doctrine. Popular concepts, such as machine learning and artificial intelligence are becoming more relevant to everyday operations. This book will try to provide a foundational framework for people who are trying to keep up with the latest technologies.

To facilitate better learning in the context of fast-moving targets, we will present a number of stable principles that can help people keep their sanity and keep afloat. Therefore, we will present a relatively slow-changing framework, or the architecture of our learning approach. We will show that reliable architecture that inherently deals with learning and change, can be rather stable. This idea is partially derived from Turing Machine. The rationale should be self-evident when we get to that chapter.

0.1.1. The Hardest Question

As the book proceeds, we will see that all usefulness comes from its context. If one must answer a question starting with "why...", then, the most common way to answer this question is to point out the moving forces in the context of the question, then, the answer should often be self-evident.

To capture or to construct the motivating forces in context, this book choose a number of powerful forces to help readers identify their own context. However, context may be varying at all locations and time points. Therefore, we will provide a project to serve as an agency of all possible context. This agency is the Universal Computer, as originally proposed by Alan Turing. Then, readers can modify the project to suit their own need.

0.1.2. An Agent-based Architecture

This book presents a basic learning framework by treating every thing as learning agent. Here, we describe agents as organizations, human, and even man-made objects, such as computing devices, and computer programs as agents. Agents can be wrap around in other agents. In those cases, the containing agents can be considered as a kind of agency. An agency is also an operational context for other agents. As one might imagine, if a clever student ask: "What are the next thing within an agent?" The trivial answer would be: it will be agents all the way down. This will also become an implementation choice, when we start introducing programming langauges, such as NetLogo. It does use the old naming convention, by calling all agents: turtles.

0.2. Learning in a Crowd

To ensure that everyone who comes to this system could learn, we want to introduce the notion of Extreme Learning Process (XLP). XLP is a way to get a crowd of cooperative individuals to learn together as a crowd. This is particularly feasible in the modern internet age. There are many leanring activities that can take place in digital spaces. Clearly, we will also continue to conduct learning activities in physical places, but we want to make sure that students are constantly helping each other using computational and communication equipment. This idea can be summarized into a single diagram. We want the crowd to build a learning consciousness, so that they can explore a part of the solution space and content areas, then, use either synchronous or asynchronous workflows to combine their leanring results. So that people and machines can learn together. The followin diagram categorize the key motivating forces or regulatory constraints as the four modalities as Prof. Lawrence Lessig puts it:

Picture 1.1. The Four Fo.

0.2.1. The Instrumentation of XLP

We brand this kind of learning approach as XLP, because the learning capabilities of these wired communities can vary a lot by the communication tools they choose to employ. Moreover, whether they are aware of the functionalities of these communication systems is critical to their success.

XLP explicitly manages elements of crowd coordination into the learning orgnaization. This organization can also have their mathcing metaphorical objects in a computing system. For example, the source code is like the code of honor or code of behavior of an operating system, the computational resources can be considered as resources in the market place, the particular functionality or intelligence can be considered as a kind of technological architecture, and the popularized content can be linked to cultural norms. These matching elements should have their data structures and operational institutions to support these operations.

Table 1.1. Languages used in the course

Elements of Computing	*Metaphors*
RISC/CISC Computer Architecture	Technological Architecture
Operating System	Legal Code
Computational Resources/Memory/Bandwidth	Market Resources
Visual Reprsesentation/Formal Representation	Cultural Norms

Enter table note here.

To help student better understand the meaning of these systems, we also recommend students to learn about historyof technology, history of constitutions, and history of personalities. These historical contenxt will better ground students understanding to a system of inter-relationships, so that they can draw analogies to solidify their understanding and applications of their knowledge.

Table 1.2. Referenced Text

Subjects	Book	*Author*
History of Mathematics	Form and Function of Mathematics	Saunder McLane
History of Video Games	Enter data here	Enter data here
History of Constitutional Laws	Constitution for Dummies	Enter data here
History of Computing Scientists	Enter data here	Enter data here

Enter table note here.

0.3. Why Start with Computers?

Using the crowd as a knowledge management device, we might be able to get the surrounding people to help individuals to learn about many subjects. To make this course having a specific focus, we decided to focus on one subject matter: building a physical computing system from scratch. This subject is meaningful to many people. The first reason is due to the pervasiveness of computing devices. The second reason is that we computing systems are built for encoding and decoding information. We need to use this extensible (programmable) devices as both a physical device to help us manage information, as well as using the vocabulary and technical principles of computing industry to help us better understand the functions and application forms of modern information processing. Based on our experience, computation and mathematics are pervasive in our thought processes, and they have been formally linked at the birth of the computing science disicipline. Knowing the roots of computing sciences, can help us better understand the other branches of computing, and many parallel disciplines.

We also recommend readers to go through the process of “building” a physical computer by following our tutorial material accomanies with this book. We believe that the engineering process of constructing a computer is also a rich context for the participants to ground their understanding of the abstract ideas. The field of computing science is about abstraction, but different layers of a physical computing usually represent

a unique style or approach of abstraction. Having direct experience in implementing some of these abstractions in the concrete form, can also solidify the depth and breadth of abstration skills.

0.3.1. Stack of Abstractions

Our approach to build this book, is to incrementally add tools and invite readers to help construct the technology stack of this book. As the very first version of this book is being written in the Mathematica Notebook environment, we intend to slowly switch it toward some more open, more network-centric environment. As one might already know, the existence of IPython, and many web-based authoring tool is becoming very attractive to serve as an alteranive to Mathematica Notebook authoring. Since IPython also offer a Mathematica Notebook-inspired user interface. Temporarily, we will proceed with Mathematica, and the content will be transferred to other system.

0.3.2. Design by Contract

A critical discipline that must be upheld is the concept of writing test cases before we write implementation. This will be a preferred approach throughout any project. In other words, we will write the design contract before we implement the software. This will show in the book by starting the whole book with Design by Contract as its first chapter. In the natural language world, we will probably shuffle the content of this book through a natural language processing tool as a starting point to demonstrate everything, including natural language text could be tested.

0.3.3. Natural Langauge Processing

Human brains are a kind of computer. It may run on a different architecture than the silicon-based computers that we are seeing everyday. Like computers, human brains can be programmed by many kinds of languages, including the sights and sounds in the physical environments as well as natural langauges. In this course, natural language processing will also be discussed, since that some of the logical structures of natural language happens to be the corner stone of modern computer programming languages. This is particularly made known by the Chomsky assumption, and parsing formal languages often follow the “Chomsky Normal Form”. As the course proceed, we will show that natural language processing techniques shows up on many levels of complex systems. We will also incorporate concepts and ideas throught the weeks. At the time of this writing, we plan to use NLTK3.0, the natural langauge tool kit written in Python to perform this task.

0.4. From Nand to Tetris

This book is written under the strong influence of the work of Noam Nisan and Shimon Sckocken’s The Elements of Computing Systems. This course is also known as From Nand to Tetris, as it starts instructing a complete novice of computing systems to construct both hardware and software systems of a computing device, from simple nand gate to a full-blown application such as a Tetris game. This course is very inspiring to us, because it showed that there is a way to integrate many elements of computing systems to a continuous set of learning activities. These activities can be strongly connected with one another, therefore strengthening the understanding of a very complex system with sufficient overlap and mutual support. Students can learn many new ideas, while using older, more familiar concepts or exercises to slowly acquire a better comprehension of new abstractions. To better learn how this can be accomplished, please read Nand2Tetris’ authors’ original paper on how to teach students to build a complex engineering system.

Picture 1.2. Picture caption.

0.4.1. Physical Devices

Nand2Tetris course is really a software engineering course. Every device has a software component matching its functions. Therefore, students can “simulate” a physical computer using software. However, to best understanding the “platonistic” view of computing science, it would be nice to have direct experience in building these devices as the course proceeds. We recommend the following steps.

Table 1.3. Referenced Text

Devices	Resource	*Software*
Logic Gates and 4−bit Adder	Breadborad	Fritzing
FPGA implemented CPU and Memory	Altera	Altera Software

Secure Moible Devices	ARM Secure CPU	ARM−based
Distributed OS and IoT−based Network Services	Jini/River/SORCER	SORCER/NetLogo

Enter table note here.

We know for sure that younger children, age as low as 5 to 8, can take the first half of this course. With physical device construction exercise, we can stimulate their interest, and further test their understanding.

0.4.2. The Language Game

A fundamental reason for us to adopt Nand2Tetris course as a pedagogical model is to leverage its multi-language approach throught the course. To get silicon devices to perform computation, one must translate human symbolism into electronic signals or to certiain mathematical expressions, so that translation can be grounded to a reliable system of metaphors. Even the Nand2Tetris course didn't explicitly emphasize this idea, the exercises and the way problems are solved in every chapter, are all exhibiting this philosophy.

To further amplify this notion, that problem solving can be translated to adequate langauge system, and then, translate back to certain implementation strategy, we will get all the students to learn more about formal languages. They must be consciously aware that they are learning languages. Each language would be suited to solve a particular kind of problem. Moreover, formal languages have strict language syntax and semantics, so that formalized translation can reliably take place between different languages.

For certain people, one language may be more convenient than the others. Therefore, we will introduce a number of automatic tools to demonstrate that certain languages are easier in expressing questions, and certain languages are easier in presenting solutions. Some forms of languages are easier in visualizing the nature of the problems.

Particularly, we plan to use Mathematica and IPython to help students write down their ideas and problem formulations at every stage. Once they build up this habit, they can learn more about how to incrementally solve problems at different scales.

Table 1.4. Languages used in the course

Language	*Enter column head here*
Mathematica	Enter data here
Python	Enter data here
Hardware Description Language	Enter data here
Jack/Hack Assembly	Enter data here
	NetLogo

Enter table note here.

0.4.3. Language-Oriented Methods

In computing science, the term "machine" is often marked as being exchangeable with "language". Therefore, we hope that students who study with us would try to realize that a language is often designed to help compress a domain of knowledge into a set of basic vocabulary, a convenient syntax, and a set of relevant semantic constructs. Then, many information can be expressed using these linguistic tools. This is also called: Language-Oriented Engineering. In other words, langauges are a malleable form of abstraction. They are the concrete instrumentation to manage abstraction.

Table 1.5. Referenced Text

Concepts	Key Ideas	*Language*
Domain Specific Langauges	DSL	Meta−Language
English Language	Dictionary/Grammar/Usage Context	ML
Functional Programming	Lambda Expression	Haskell
Programming Languages	Declarative/Procedural/Object − based	ANTLR/YACC

Enter table note here.

0.4.4. Knowledge Management

We hope that students can eventually learn about learning as a concrete way to managing layers of abstractions. One should be aware of the instrumentation in collecting and compressing information, as well as the way to present information in different languages. We believe that by carefully studying with this book, you will gain an over-arching view of how knowledge can be managed.

A well known theory called Byzantine General's Problem, and its famouns application, Bitcoin/Blockchain is a mathematical approach to ensure the trust-worthiness of knowledge content.

Table 1.6. Branches of Knowledge Management Practices

Branches	Key Publications	*Author*
Corporate Knowledge Management	Theory *U*/The Fifth Discipline	Otto Scharmer/Peter Senge

Engineering Knowledge Management	Wikinomics/Management 3.0	TBD
Scientific Knowledge Management	Science of Artificials/Artificial Intelligence/PAC	Herbert Simon/Stuart Kuafman/Leslie Valiant
Societal Knowledge Management	Byzantine General's Problem/Social Physics/Code v2	Pentland/Lessig

Enter table note here.

0.4.5. The Ongoing Frontiers

We live in a world that are filled with exponential growth patterns. There will be many more topics and tools that are relevant to this finite book. To capture and manage those ideas, we will list a number of fields that are going to be relevant to our development effort. We plan to incorporate life sciences, neural network, probablistic theories and other foundational models of computation as a part of this book. We pick these topics, since these topics are relevant to the most basic models of computation. Moreover, these areas are also experience the highest degree of growth rate, due to their mutal relevance. Therefore, we will explicitly draw references and tools while proceeding with each chapter.

Table 1.7. Branches of Knowledge Management Practices

Topics	Key Publications	*Author*
Life Sciences	What is Life?	Erwin Schrodinger
Neural Network	TBD	TBD
Statistical/Probablistic Models	Logic of Science/Causality	Judea Pearl
Category Theory	Category Theory for Working Mathematicians	Saunder McLane

Enter table note here.

0.5. Closing the Loop

A useful pedagogical approach should always come with a set of complimentary feedback mechanism. Due to the computational nature and the data-intensive nature of this course, we will also instruct all our students to use a set of popular, yet extensible set of data collection and sharing tools. We can use the process data collected by these tools to help us assess the results of the overal system. Our approach includes the following elements:

Table 1.8. Learning Outcome Feedback

Area	Key Data Collector	*Executors*
In−Presence Learning	On Site Videos/Interviews	Instructors/Asssitants
Off−site Learning	Git/Wiki/Teambition	Project managers
Statistical Data	Learning Analytics	Software Packages
Simulation	NetLogo Simulation	Agent − based Modelers

Enter table note here.

In theory, as we run many courses, we will reveal our learning experience using the data collected and computational tools that we taught our students. Ideally, students should be authoring their personal websites, and publish their content as they wish.

0.5.1. For Your Eyes Only

Over time, we wish that we may use the whole stack of Nand2Tetris-based computing system to implement a secure Knowledge Management platform on a Internet scale. We hope that, collectively, we may include the notion of Byzantine General's Problem as a way to mutally witness the transparency of data, while keeping the identity of participating individuals private. Moreover, we also wish that we can show people that using agent-based modeling language, we can allow students to build secure, yet flexible/recombinable systems in relatively short amount of time.

References

A. Authorlast, "Article Title," *Journal Title*, **Volume**(Issue), 2005 pp. #–#.

B. Authorlast1 and B. Authorlast2, "Article Title," *Journal Title*, **Volume**(Issue), 2005 pp. #–#.

C. Authorlast1, B. Authorlast2, and C. Authorlast3, "Article Title," *Journal Title*, **Volume**(Issue), 2005 pp. #–#.

D. Authorlast, *Book Title*, *n*th ed., Publisher Location: Publisher Name, 2005.

E. Authorlast1 and B. Authorlast2, *Book Title*, *n*th ed., Publisher Location: Publisher Name, 2005.

F. Authorlast1, B. Authorlast2, and C. Authorlast3, *Book Title*, *n*th ed., Publisher Location: Publisher Name, 2005.

G. Authorlast, "Paper Title," in *Conference Proceedings Title (Conference Acronym and Year)*, Conference Location (A. Authorlast, ed.), Publisher Location: Publisher Name, Publication Date pp. #–#.

H. Authorlast1, B. Authorlast2, and C. Authorlast3, "Paper Title," in *Conference Proceedings Title (Conference Acronym and Year)*, Conference Location (A. Authorlast, ed.), Publisher Location: Publisher Name, Publication Date pp. #–#.

I. Authorlast1, B. Authorlast2, and C. Authorlast3, "Paper Title," in *Conference Proceedings Title (Conference Acronym and Year)*, Conference Location (A. Authorlast, ed.), Publisher Location: Publisher Name, Publication Date pp. #–#.

J. Authorlast. "Website Title." (Last updated date or date visited in three-character Month Day, Year format) URL.

K. B. Authorlast. "Entry Title" from CompanyN—A CompanyN Web Resource. URL.