Kyle Harrity
COSC 460-001
September 1, 2013

10 Domains in Computer Science

1. Computer Theory

Computer Theory, according to our lecture notes, is the branch of computer science concerned with finding out how efficiently a given problem can be solved on a model computer using a given algorithm. Within Computer Theory are three main focuses: Computability Theory, Complexity Theory, and Formal Languages.

Computability theory is concerned with finding out whether or not a problem is solvable with a finite number of computations. This is an important field in computer science because it saves time by determining if a problem is solvable.

Complexity Theory deals with finding out how much time an algorithm will take to solve a problem based on how many elements are involved with the computation. The answer is usually written in “Big O” notation. There are generally two speeds at which an algorithm works: polynomial time and exponential time. Polynomial time is good for most algorithms, exponential is completely impractical for any useful dataset. The details of polynomial and exponential algorithms are discussed in more detail in the section “Tractable vs. Intractable Problem.”

Formal languages are a way of talking about languages in the abstract. They are called “formal” because “...all the rules for the language are explicitly stated in terms of what strings of symbols can occur” (1 p.7). Formal languages are defined using set theory. There is a fundamental set containing the acceptable characters in the language. Up from the basic set is another set containing all allowable combinations of characters, and beyond this is a set of all allowable sentences. This continues to the largest constructs in a given language.
\

2. Algorithms

Algorithms are an explicitly described method for solving a problem. A great deal of work is done by computer scientists to find the most efficient algorithm to solve a problem. For example, the problem of sorting a list of integers can be solved in many ways. The most obvious one would be to compare each element to its neighbor and shift positions if they are out of the desired order. This routine, called Bubble Sort is terribly inefficient for all but the smallest of lists. A more appropriate algorithm for practical use is Merge Sort, which uses recursion and the Divide and Conquer method to efficiently sort long lists.

Implementations of algorithms are the most common applications of computers. Computers excel at executing algorithms. Almost any simple task can be defined by an algorithm. When put together, a collection of algorithms creates a program.

There are limitations to algorithms though. Many algorithms are very rigid and only correctly on a narrow range of inputs. A good example of this is Dijkstra's Algorithm (a shortest path graph algorithm). According to Weiss, this algorithm performs correctly only if there are no negative cost edges (2). An alternative to algorithms is neural networks, discussed in section 9.

3. Cryptography

Cryptography is an ancient science concerned with secret writing. That is hiding a message in what appears to be an unintelligible sequence of symbols but which can through some transposition or substitution algorithm be converted back into a meaningful message. Cryptography is still a very important discipline, perhaps more than it ever was as a result of people and businesses storing sensitive information on remotely accessible computers.

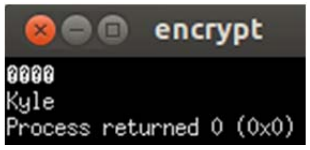
In the modern context of computers, cryptography is the science of encrypting data. Data encryption is necessary to keep information such as bank account numbers hidden from those with malicious intent. The importance of cryptography has increased greatly since the ubiquity of the Internet. In fact, one of the most common applications of cryptography is to protect data transmitted across networks. Older, more naïve, protocols such as telnet send unencrypted plain text across networks allowing user credentials to be easily intercepted. SSH is a secure substitute for telnet; It encrypts data before transmitting it. Encrypting data at rest on a disk is just as important as encrypting transmitted data. If a malicious user obtains access to a disk either physically or electronically, file system and database encryption are the last line of defense.

A simple method for encrypting data is to use the XOR bitwise operator on a key string and the message to be encrypted. The result is an encrypted message. The beauty of the XOR operator is that if the key string is then XORed against the encrypted data it will return the original message. This example could be considered a substitution encryption because it is only masking the bits of the message by XORing them with the key, nothing is moved around. Below is a simple (and very weak) implementation of the above encryption algorithm, output is on the right:

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char word[] = "Kyle";
  int i;
   int key = 0xff;

   for( i = 0; i < 4; i++) //encrypt
   {
      word[i] = word[i] ^ key;
   }
   printf("%s\n", word); //print encrypted data

   for( i = 0; i < 4; i++) //decrypt
   {
      word[i] = word[i] ^ key;
   }
   printf("%s", word); //print decrypted data
   return 0;
}
```



```
����
Kyle
Process returned 0 (0x0)
```

The above implementation uses only a single 8-bit binary sequence as the key, making it especially vulnerable to brute forcing. My computer was able to print out the result of all 256 possible keys in just 0.003 seconds. To compare, many modern encryption keys are as large as 1024-bits and are not expected to be factored by a computer until 2040 (3) . The danger in using my implementation or any other homegrown encryption algorithm is well documented.

Stephen Farrell's article, “Applications Directly using Cryptography”, published in IEEE Internet Computing, notes that in designing one's own encryption algorithm or by creating an implementation of a standard encryption algorithm many mistakes can be made. Chief among these is accidentally including bugs. Some of these bugs could even make it possible to subvert the security of the application. Farrell also warns that using another person's homegrown implementation leaves an application open to the same risks; its even possible the designer of the homegrown implementation intended for the bugs to be there for malicious purposes. To avoid these pitfalls, Farrell advises using only stable implementations of standard encryption algorithms. The article goes on to list many encryption options available to a developer.(4)

The wisdom in these words cannot be understated. Handmade encryption algorithms often fall prey to a variety of exploits that their better designed standard counterparts are built to avoid. In fact modern cryptography is so powerful that few systems are ever compromised by cryptanalysis, instead other more easily exploited flaws are the downfall of a system's security.

Cryptography is an important field in computer science. It has applications everywhere because of the prevalence of private data stored on multi-user systems. Much research is done in trying to build better encryption keys and algorithms. Cryptography will continue to be a part of system security for as long as computers exist.

4. Distributed Computing

Distributed computing is a means of breaking up a complex problem, distributing the pieces to various computers across a network (usually the Internet), and having them work on the problem concurrently until the problem is solved. The difference between distributed and parallel computing is that parallel computing is done on a single architecture with multiple processor cores and distributed computing is done over a network.

According to Bill Godfrey, distributed computing is performed with a client-server model where the server sends out work packages and constructs the answer from the responses returned by the clients (5). The networked nature of distributed computing means that it is not suitable for all problems. For example an algorithm that requires all steps be completed in sequence cannot be distributed merely duplicated. Searches, encryption breaking, and other algorithms where each step does not rely on the previous one are ideal for distributed computing because they could take much longer on a single computer. Another important feature of distributed computing is that it client machines only process data when the computer is idle. This minimizes the impact on the user and makes distributed computing more appealing to clients who may not directly benefit from their donated computing power.

Distributed computing is most commonly applied to scientific and mathematical research projects with limited hardware resources. folding@home is a well known distributed computing project that explores protein structures and helps researches understand cancer cells. Distributed computing is an excellent alternative to a super computer when it is not possible for researchers to use one.

5. Cloud Computing

Cloud Computing is a popular and often misused buzz word in the IT industry. It is often used in conjunction with any Internet service or application regardless of whether or not that software actually implement a cloud infrastructure.

Cloud Computing is more precisely defined than its colloquial usage would lead one to believe. According to Dikaiakos and others, "Cloud computing is a recent trend in IT that moves computing and data away from desktop and portable PCs into large data centers" (6 p.10). The most important part of this statement is that both data and computing power are being removed from end-user devices. Having software stored and run remotely allows for user devices to be ultra portable, thus cloud computing is a very popular model for phones and tablets.

There are three main ways in which a cloud is implemented: infrastructure as a service, platform as a service, and software as a service. The infrastructure level provides storage and computing power across the network. Further up is the platform level which allows higher abstractions usually for the purpose of software development to be shared across a network. The highest level, software provides an entire application to users over a network.

Though cloud computing is promoted as a new idea, it is actually quite similar to an old one: the client-server model. In the client-server model a central server provides computing power and applications to multiple users connected to the server via terminals. Cloud computing is an update of this concept: instead of being limited to a terminal on the same network as a server, any device with an Internet connection can take advantage of the software and hardware of a remote server.

6. Computational Learning

Computational learning, more commonly referred to as machine learning is a sub-field of artificial intelligence. More specifically, "Machine learning is the field that is dedicated to the design and development of algorithms and techniques that allow computers to 'learn'" (7 p.307). Learning, in the sense of computers is the ability to adapt an algorithm based on inputs. This adaptability is often implemented with the use of neural networks. Neural networks will be discussed in more detail later in this paper.

There are several machine learning strategies. The ones described here are supervised, semi-supervised, unsupervised, and reinforcement learning. In supervised learning, the computer is given the correct outputs for many different inputs. Then it is expected to generalize the relationship and produce correct output for new, previously unseen inputs. Unsupervised learning machines are only given the input and then expected to "see" a structure in the inputs. This is useful when the desired output is not known to the user. Semi-supervised learning combines supervised and unsupervised techniques together. Finally, reinforcement learning is a broadly defined branch of learning concerned with how an intelligent agent ought to act in an environment to get the highest reward. Unlike supervised or unsupervised learning methods, reinforcement learning does not take any inputs it simply adapts to its environment. From searches on the IEEE Computer Society article databases it would seem that reinforcement learning is a popular framework for video game A.I.

7. Computer Vision

Computer vision is the field of computer science in which the goal is to process video or still images and produce from them a representation of the world. This entails things such as recognition of objects, gestures, and motion. The purpose of recognizing these things is for decision making. For example, an automated security system equipped with computer vision software could alert the police if it recognizes humans in a restricted area. Because of computer vision's use in decision making as well as the difficulty of creating a computer vision system by conventional means, it overlaps with artificial intelligence.

Many techniques for implementing computer vision use one of the above learning methods. Just like in video games, reinforcement learning is popular for computer vision because of its focus on environment. Computer vision presents a challenging problem to computer scientists because there are so many different objects in a given image or video and it can be very hard to identify partly hidden ones or to recognize gestures correctly. Despite these problems, progress continues in part because of the proliferation of more powerful embedded systems such as the ARM multi-core processors found in many mobile phones and tablets.

Future applications of computer vision may include helping visually impaired people, gesture based user interfaces, advanced security systems, and better robot navigation systems.

8. Big Data

The idea of big data is fairly self explanatory, it is the collection and analysis of large amounts of data. It is usually associated with large Internet companies such as Facebook and Google. These companies collect vast amounts of data about users on their sites. They then organize it and use it as a means for targeted advertising. Since the proliferation of cloud computing platforms, big data has become very attractive to companies that previously did not see the benefit of storing and organizing huge amounts of user data. It is even creating new ways of providing services: instead of waiting for users to enter information, some businesses now just use web crawlers to aggregate data and then organize and build a site from that. In fact, some startups such as TalentBin and Gild use big data to build meta-resume sites where they collect information on job candidates via various social media sites.

Big data has other applications as well. Recently, the NSA got exposed for tracking people based on information obtained from companies using big data. According to Lesk, there is almost no regulation in the US for personal data collection or usage (8). Unfortunately, due to the power of the companies and government agencies behind big data, the lack of regulation is unlikely to cahnge in the near term.

Not all applications of big data are bad, however. Big data can be used by cooperating hospitals to keep medical records more current and accurate. Additionally it could be used to track the spread of disease through medical records. There are many applications for big data that benefit people. The availability of vast amounts of data is not always a bad thing.

9. Neural Networks

Neural networks are artificial intelligence agents that simulate biological neuron structures to process inputs and produce outputs. Earlier, this paper mentioned machine learning. Neural networks require machine learning to function usefully. Neural networks are trained on inputs until they recognize patterns and produce useful outputs.

Neural networks differ from normal algorithmic programs in that they do not follow a sequence of steps. They are not even programmable. Instead they learn from example inputs. However, as the article, “Neural Networks”, notes, “The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly” (9). Because neural networks cannot be programmed to perform tasks they are very hard to troubleshoot when malfunctioning. The operator of the network has no means by which to “fix a bug.” Despite this drawback of neural networks, they are still better suited than algorithms for some tasks, especially those where the task is not clearly defined or understood.

The applications of neural networks are far reaching. They can perform tasks such as making predictions based on available data, interpreting visual data for face recognition, and other computer vision tasks, and even serve as expert systems such and perform medical diagnoses.

10. Tractable Problem vs. Intractable Problem

A tractable problem are problems that can be solved in polynomial time (ex. O(n), O(n^k), O(nlog(n))). An intractable problem can currently only be solved by an algorithm taking exponential time (ex O(2^n), O(n!)). The tractable problems can be solved in a reasonable amount of time for most values of n while the intractable problems can only be solved in our life time if n is too small to be of any practical use. Because of the vast time difference between tractable and intractable problems, much work is done in trying to find tractable solutions to intractable problems.

A very pressing question to computer scientists looking for tractable solutions to difficult problems is the P vs. NP problem. P is the set of problems whose solutions can be *found* in polynomial time and NP is the set of problems whose solutions can be *verified* in polynomial time. The distinction between P and NP is small but important, NP problems are intractable but the verification of their solution is tractable. If P =NP then all NP problems have tractable solutions and all that remains is to find the algorithm for each NP problem that works in polynomial time. If P≠NP or the answer remains unknown, a lot of time may be wasted in looking for tractable solutions that do not exist. The P vs. NP problem has been open for quite sometime and still no one has proved or disproved the assertion P=NP.

References

1. D.I.A. Cohen, *Introduction to Computer Theory*. R. Brooks, p. 7, John Wiley & Sons Inc., (1997).

2. M.A. Weiss, *Data Structures and Algorithm Analysis in Java*. M. Hirsch, p. 400, Pearson, Essex, England, (2012).

3. I. Grigg, P Gutmann, “The Curse of Cryptographic Numerology”, *IEEE Security and Privacy*. p. 70, (2011).

4. S. Farrell, “Applications directly Using Cryptography”, *IEEE Internet Computing*. p. 85, (2010).

5. B. Godfrey, “A Primer on Distributed Computing”, [http://www.bacchae.co.uk/docs/dist.html]. 2006.

6. M.D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, A. Vakali, “Cloud Computing: Distributed Internet Computing for IT and Scientific Research”, *IEEE Internet Computing*. p. 10, (2009).

7. D. Bui, D.K. Nguyen, T.D. Ngo, “Supervising an Unsupervised Neural Network”. *Asian Conference on Intelligent Information and Database Systems*. p. 307, (2009).

8. M. Lesk, “Big Data, Big Brother, Big Money”. *IEEE Security and Privacy*. p. 87, (2013).

9. C. Sterigou, D. Siganos, “Neural Networks” [http://www.doc.ic.ac.uk/~nd/ - surprise_96/journal/vol4/cs11/report.html]. retrieved 2013.