

Hideyuki Nakashima
Keiji Hirata *Editors*

Computational Thinking

Rethinking How We Think



Springer

Editors

Hideyuki Nakashima

Sapporo City University, Sapporo, Hokkaido, Japan

Keiji Hirata

Department of Complex and Intelligent Systems, Future University Hakodate, Hakodate, Hokkaido, Japan

ISBN 978-981-95-5961-9

e-ISBN 978-981-95-5962-6

<https://doi.org/10.1007/978-981-95-5962-6>

Translation from the Japanese language edition: “計算論的思考ってなに？” by Hideyuki Nakashima and Keiji Hirata, © Authors 2022. Published by Future University Hakodate Press & Kindaikagakusha Publisher. All Rights Reserved.

The original submitted manuscript has been translated into English. The translation was done using artificial intelligence. A subsequent revision was performed by the author(s) to further refine the work and to ensure that the translation is appropriate concerning content and scientific correctness. It may, however, read stylistically different from a conventional translation.

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2026

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd. The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

When I first read Jeannette Wing’s influential essay “Computational Thinking” published in 2006, I immediately recognized its importance. I wholeheartedly agreed with her assertion that computational thinking (CT) should be a fundamental component of university education for all disciplines, and that students majoring in computer science would be equipped to excel in a wide range of careers beyond their technical training. I translated Wing’s essay into Japanese, and it was published in the Journal of the Information Processing Society of Japan in 2015.

However, when Prof. Michael Vallance, our colleague at Future University Hakodate (hereafter referred to as FUN), used my translation as the teaching material, he found that students struggled to understand it. Wing’s essay includes many technical terms from computer science but offers few concrete examples—particularly ones that demonstrate how such concepts apply to other fields. While the essay was written for educators, it is not structured to help beginners grasp the essence of computational thinking.

This observation led us at FUN to launch a project aimed at publishing a book filled with accessible and practical examples of CT. It was not easy to find examples that made sense to people outside of computer science. We each tried various approaches and held frequent discussions. Eventually, we compiled our findings into the desired book in Japanese, which translates Wing’s abstract ideas into concrete terms that are easier to understand for non-specialists.

At the same time, international interest in teaching computational thinking was gaining momentum, with new academic societies and initiatives being launched. In Japan, however, government agencies such as the Ministry of Education had interpreted CT to mean programming and began to incorporate it into elementary school curricula. Even national media outlets like NHK produced TV programs based on this interpretation. Needless to say, programming knowledge and skills, along with the ability to effectively utilize modern AI technology, are certainly part of CT, but CT encompasses much more.

This English edition is based on the Japanese edition published by FUN Press in 2022. Chapters 1, 2, 4 and 5 of that Japanese edition have been revised to a diverse extent. This English edition was unable to include Chap. 3, authored by Michael Vallance. For his research, please see his paper titled “Computational thinking and interdisciplinary learning: time to focus on pedagogy” published in 2020.

The structure of this book is designed to be progressive, starting with an introduction to fundamental concepts in computer science, moving through methods for understanding computational thinking, and culminating in its practical application. However, it has been consciously written so that readers can begin reading from any chapter.

This book begins in Chap. [1](#) by serving as a primer for readers unfamiliar with computer science, defining computational thinking as “thinking like a computer scientist.” The chapter explains the origins of computer science and its unique worldview by contrasting it with fields like physics and mathematics, while also addressing its relationship with programming, deep learning, and generative AI.

Chapter [2](#) delves into the practical benefits of applying computational thinking in the real world. It introduces three of four essential concepts—abstraction, modeling, and virtualization—which are presented as fundamental operations for generating new information. The chapter uses

relatable examples to make these terms accessible and concludes by connecting computational thinking to the principles of information design.

Chapter [3](#) introduces the fourth essential concept, meta, and its related idea of self-reference. Acknowledging the complexity of the topic, the chapter explains how to think about an object and concept from a higher perspective, using real world examples to clarify how to apply this thinking both in human activities and within a computer.

Chapter [4](#) is an essay in which a computer scientist and education scholar shares her personal journey, explaining how her childhood experiences cultivated a computational mindset. She uses the unique analogy of cooking to explain computational thinking, breaking down a recipe into a series of logical steps, and discusses how this perspective can be taught effectively to children.

Finally, in Chap. [5](#), a cognitive scientist with no background in computer science reports on her personal struggle to understand computational thinking. She shares her initial discomfort and the turning point that allowed her to grasp the concept, concluding with a non-specialist's unique discovery of computational thinking's altruistic power.

Hideyuki Nakashima
Sapporo, Japan
September 2025

Contents

[1 What is Computational Thinking?—Think as Computer Scientists Do](#)

Hideyuki Nakashima and Keiji Hirata

[2 Practice of Computational Thinking](#)

Keiji Hirata and Hideyuki Nakashima

[3 What is Meta?](#)

Yasuhiro Katagiri

[4 Life and Computational Thinking](#)

Noyuri Mima

[5 Joy of Understanding Computational Thinking](#)

Misako Nambu

1. What is Computational Thinking?—Think as Computer Scientists Do

Hideyuki Nakashima¹  and Keiji Hirata¹
(¹) Sapporo City University, Sapporo, Hokkaido, Japan

 **Hideyuki Nakashima**
Email: hideyuki.nakashima@gmail.com

Abstract

This chapter is written to provide foundational knowledge about computers for readers unfamiliar with computer science, serving as a primer. Jeannette Wing, who proposed computational thinking, defined it as “thinking like a computer scientist.” Computer science is a discipline that seeks to understand the nature of the world based on the worldview of information and computation. Computer science is a relatively young field, having only begun to be recognized as an academic discipline around the mid-twentieth century. This chapter explains the origins and concepts of computer science, detailing unique ideas not found in other academic fields, while carefully guiding beginners to understanding. This chapter provides a clear definition of computational thinking by examining the thought processes of computer scientists. It includes a wealth of concrete examples to enhance understanding. A comprehensive perspective is offered by contrasting the meaning of computation with concepts from physics, logic, philosophy, and mathematics. It also addresses the relationship between programming and computational thinking, which is often confused, as well as its connections to deep learning and generative AI.

1.1 Thinking in the Computer Age

Without computers, we would not be able to use electricity, water, banks, railways, or airplanes. Understanding how computers accomplish those tasks also helps us think and plan our daily life tasks. Computational thinking, which this book deals with, is not just knowledge or technology, but something like a compass for living better in the information world.

1.2 Computation and Thinking

The term computational thinking contains two words that everyone knows. One is computation, and the other is thinking.

1.2.1 Computation

When you hear the word “computation”, you might think of arithmetic operations. However, computations in the information world are not limited to arithmetic operations. Thinking of tactics to win a soccer game, planning a 10-day vacation in Vienna, creating the best dish with the ingredients in the refrigerator, ensuring logistics when roads are cut off by an earthquake, and even writing novels or composing music are all computations. What these have in common is that they are about “finding the optimal solution” for a goal within the constraints of the real world.

Computation is a powerful method for generating new information. For example, finding the greatest common divisor, GCD, of two numbers is not an easy task in general. The GCD of 4 and 6 can be easily found (2), but finding the GCD of large numbers is difficult. There is a method of Euclid for finding the greatest common divisor GCD of two numbers (defined in Sect. [1.7.5](#)), which is generally referred to as “Euclidean algorithm”. This kind of computational method is called an “algorithm.”

You may ask what does finding GCD have something to do with our daily life. For instance, one textbook shows an example of a task of tile craftsmen.

You are trying to cover the floor of a room with short sides of 18.9 m and long sides of 24.5 m with square tiles. You are about to bake the tiles, but you want to use as few tiles as possible. How long should one side of the tile be?

The correct answer is 70 cm, as the GCD of 189 and 245 is 7. Note that the GCD is a concept that applies only to natural numbers, so we are multiplying by 10 for the calculation.

In actual daily life, you may rarely see situations where the GCD is used. However, the expression “greatest common divisor” is often used metaphorically in everyday life. For example, it means the largest common element of two or more opinions, which in this case means the compromise point of the discussion.

On the other hand, there is also the least common multiple. This means the smallest multiple of two natural numbers. There is a zodiac in the calendar, which consists of the combination of ten stems: 甲 (Jia), 乙 (Yi), 丙 (Bing), 丁 (Ding), 戊 (Wu), 己 (Ji), 庚 (Geng), 辛 (Xin), 壬 (Ren) and 癸 (Gui); and another twelve stems: 子 (rat), 丑 (ox), 寅 (tiger), 卯 (rabbit), 辰 (dragon), 巳 (snake), 午 (horse), 未 (goat), 申 (monkey), 酉 (rooster), 戌 (dog) and 亥 (pig). Each year is named by the combination of two stems. The least common multiple of 10 and 12 is 60, so the zodiac returns to its original state every 60 years. This is called the sexagenary cycle.

1.2.2 Thinking

When you hear the word thinking, you might think it’s all about mulling over ideas in your head. However, the thinking in computational thinking doesn’t stop there. What’s important is to actually try out what you’ve thought about in the real world, in other words, practice (more about practice will be discussed in Chap. [2](#)). Imagine yourself as the hardware and software of a computer, freely executing computations in the real world surrounding you. Jeannette Wing, who proposed computational thinking, has stated that computational thinking is thinking like a computer scientist [Wing 2006]. As you will see as you read this book, computer scientists use the tool of the computer, or the metaphor of the computer, to try to understand various phenomena and problems in the real world.

In the real world, everything exists, functions, and changes under the absolute constraints of physical laws. Specific constraints include time progresses, entropy increases, and energy and mass are conserved. Neither the human body, the earth, nor space rockets can escape these physical constraints. On the other hand, computations are generally abstract operations and are

not subject to physical constraints. In the physical world, there are physical law constraints to build a building above a certain height, and there is a limit to the height. However, there are no such constraints in the world of computation.

For example, no matter how long the number of digits, it is possible to perform arithmetic operations, and no matter how large the database, it can be created if desired. However, it is not possible to actually prepare an infinite-digit abacus (or computer memory) to hold the numbers to be calculated, nor is it possible to actually create a database with a record number exceeding the number of atoms in the entire universe. Therefore, at least in the real world, it is meaningless unless arithmetic operations can be performed, and databases can operate within the realistic range of costs, time, and space. Although it is not constrained by physical laws, there are constraints in the world of computation, such as the amount of memory and computation time (collectively referred to as “computational complexity”). This is an important concept in computational thinking.

1.2.3 Conditions for Good Computation

Furthermore, the computation in the real world inevitably involves the perspective of value. It is because of the physical constraints mentioned in the previous section that value becomes necessary. For example, one person may prioritize the speed of arithmetic operations over the size of the number of digits. Another person may prioritize the number of queries that can be processed at once over the response time for each query to the database. This means that with the introduction of value, computations are classified into desirable and undesirable ones, and if that value changes, the desirable computations will also change.

Column 1 The difficulty of value evaluation

You may often want to optimize the procedure of your work. It could be the procedure of cooking or cleaning, but “what to optimize?” is a difficult question. For example, it is difficult to create a measure for “taste.” It also depends on individual preferences (values).

Even if a measure can be defined, there is a problem of trade-offs between multiple measures. In cooking, for example, there are various measures such as cost of ingredients, cooking time, taste, etc., and they often contradict each other. If you try to optimize only the taste, the cost of ingredients will be higher and the cooking effort will increase. Humans are making compromises between these multiple conditions that cannot be optimized at the same time. For example, optimizing the taste within a cost of 1000 yen for ingredients.

When a JR (Japan Railways) express train runs late, it’s standard practice to hold local trains to keep the express on schedule. From JR’s point of view, the express train’s punctuality is a higher priority because delays beyond a specific duration often result in ticket refunds. As a passenger, you always want your own train to have the highest priority, but that’s not always the reality. One of the authors once learned this the hard way on the Airport train heading from Sapporo to New Chitose Airport. An announcement came on: the Airport train would stop temporarily to let a delayed express train pass. Despite having left ample time, the author missed their flight. In the moment, the author felt the missed connection was simply a consequence of JR’s scheduling priorities. Yet, when considering the severe disruption the flight connection caused—and not just for one person—the importance of airline connections should, arguably, be given higher weight. Of course, everyone’s situation is unique, and not all

priorities align. Ultimately, this scenario highlights a conflict between JR's internal operational values and the values of time-sensitive passengers.

Currently, corrections to schedule disruptions due to accidents or bad weather are mostly done manually. Optimization is something computers are good at, but it seems that it has not been programmed because it is not self-evident which conditions should be optimized (they change depending on the situation). At this point, it is unclear whether AI will be able to understand the values held by humans in the future.

A good computation in computational thinking gives the optimal solution from a certain value perspective to the goals set within such physical constraints. The result changes depending on which of the multiple values is chosen. Computational thinking is a way of thinking that provides shortcuts and hints to realize this good computation.

No matter how smart a robot is, or how well an autonomous vehicle drives compared to a human, or an AI that creates works of art, the computer is performing symbol manipulation within it. ¹ Through these manipulated symbols, the computer understands the real world and then influences it. What becomes important here is how to assign (correspond) symbols to the things and concepts of the real world bound by physical constraints. For example, the computation needed to answer the question "How many minutes are in a day?" is immediately clear. The meaning of the symbols "day", "hour", and "minute" that appear in that computation is obvious. However, symbol manipulation of "joy" or "sadness", symbols assigned to the movements of the mind, seems difficult or almost impossible. "Every time I listen to Bach's music, I discover something new", "I don't mind sacrificing myself for justice", is it possible to realize a computer that calculates such things? This is a fundamental problem of artificial intelligence. And it has been debated for a long time. Can intelligence be expressed with symbols alone? ² Let's leave such difficult questions for another occasion. In this book, we only want to consider how to represent phenomena, constraints, and problems of the real world as data structures and functions in a program. Through this, we want to change our view of the world. As for how to take such perspectives, set problems, and educate them, Chap. ⁴ will also address this.

The issue of representation is a historical challenge in computer science research, and it has been dealt with in the artificial intelligence research from the early stages, but it has not been given much importance in traditional education. The first condition for good computation is appropriately representing real-world problems within a computer, and the same way of thinking can also be useful in our daily lives. I would like to provide examples of this. The understanding of the real world by a computer through symbols within a computer is quite similar to human's understanding of the real world through metaphors and analogies. Using metaphors and analogies allows us to feel as if we understand "it" by using similar parts as clues, but it does not accurately represent all of "it", nor does it mean that we have come to understand all of "it." However, good metaphors and analogies greatly help humans' understanding and are very useful.

The second condition for good computation is to explore and find more efficient or effective ways to solve phenomena or problems that have been brought into the world of symbols. Algorithms and procedures are ways to show the path to solutions. They can be used not only for computer programs but also for humans. Reducing the size of the array needed for computation or reducing the number of iterations is involved in efficient and effective problem-solving. This way of thinking is effectively applied to human thought as well, probably because human memory

capacity, human recognition/computation speed and human error rates are also influenced by physical constraints.

Here we summarize the conditions of good computation; the first condition is to appropriately represent real-world problems within a computer, and the second one is to explore and find more efficient and effective ways to solve phenomena or problems that have been brought into the world of symbols. Let's illustrate the relationship between these conditions for good computation and computational thinking (Fig. 1.1).

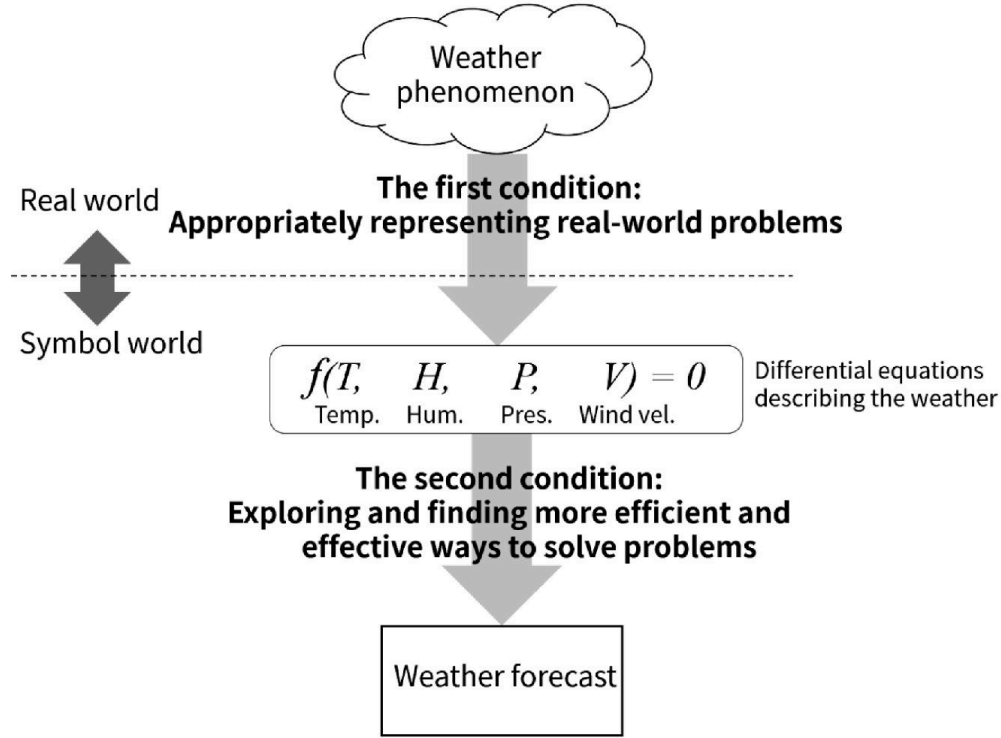


Fig. 1.1 Computational thinking that satisfies the two conditions for good computation

When facing phenomena and problems in the real world that seem ambiguous, massive, and sometimes even structureless, how can we effectively meet these two conditions of good computation? Let's consider this problem in detail below.

1.3 Exponential Explosion

When programming a computer for calculation, one must be careful of a phenomenon known as exponential explosion. This refers to the situation where the amount of data increases and the computation time and memory capacity required increase too much to handle. Such a situation is called the exponential increase as mentioned in the column, and "explosion" is a term that emphasizes the point of rapidly becoming intractable.

For instance, exponential increase can be reproduced by a process of adding up all the numbers that have appeared so far to create the next number as follows.

$$X_{n+1} = \sum_{i=1, \dots, n} X_i$$

In fact, let's see what happens if we start with $X_1 = 1$, and you obtain the sequence of 1, 1, 2, 4, 8, 16, 32, 64, ... As is evident, starting from the second term, numbers grow exponentially and can be expressed as 2^n . Why does the calculation produce such a beautiful sequence of numbers? You may understand it immediately if you think of the asymptotic formula to compute X_{n+1} as follows.

$$\begin{aligned} X_{n+1} &= \sum_{i=1, \dots, n} X_i \\ &= X_n + X_{n-1} + X_{n-2} + \dots + X_1 \\ &= X_n + X_n \\ &= 2X_n \end{aligned}$$

1.3.1 Friends of Friends of Friends of...

There are many examples in the world where numbers increase exponentially. You may have heard the fun phrase "a friend's friend is a friend." If each person has 10 friends, then a simple calculation tells us that friends of a friend would amount to 100 people. This situation is expressed as having 10 friends at the first level and 100 friends at the second level, and thus, the number of friends increases exponentially. Now, in sociology, there is a well-known rule called "six degrees of separation." This means that anyone can become friends with all the people on earth if he or she considers up to the sixth level of friends (assuming there is no overlap in friendships). The current global population is 7 billion, which is approximately 44^6 ; this implies that each person has an average of 44 friends. Intuitively, it seems reasonable that each of us has about 50 to 100 friends or acquaintances.³ What is surprising here is that by following the chain of friends-of-friend only 6 times, you can reach everyone on Earth. Amazingly every reader of this book can reach the President of the United States by tracing your friends-of-friend chain for 6 times! You might feel the power of exponential growth.

Computational complexity

In the world of programming, the effort required for (the memory and computation time required to execute the computation) is an issue. This is shown as a quantity relative to the size of the problem and is called the order, which is represented by a function O . For example, $O(N)$ means that resources proportional to the size of problem are needed. $O(k^N)$ is called exponential the computational complexity increases rapidly with the size of problem; it is usually not practical for computation. Conversely, $O(\log N)$ is an easy problem to handle.

Column 2 Exponential Acceleration of Evolution

Both the evolution of organisms and the evolution of science and technology have been developed so far through the combination of "parts." Evolution accelerates exponentially as the parts get larger. In terms of organisms, the evolution that was occurring at the molecular level extends to the evolution at the cellular level due to the creation of a part called a cell, and it became dramatically faster (incidentally, it is said that all existing organisms are descendants of a single cell that was created by chance). Furthermore, multicellular organisms were born from unicellular organisms, giving birth to a new unit of evolution. What was once a whole is divided

into parts, and these individual parts are differently combined to create a new whole, which becomes a new unit. It gives rise to the evolution at a higher speed.

Then humanity was born and gave birth to science and technology. The realm of the evolution shifted from the level of individual organisms' intelligence to the that of the intelligence of the entire society created by those organisms (humans). Necessarily, the laws governing evolution were also changed. Until cells were created, chemical reactions determining molecular structures were the dominant laws. When a cell became a unit, the dynamic non-equilibrium system as life became the governing principle. When organisms became a unit, the theory of evolution in the sense described by Darwin became the governing principle. Furthermore, when the knowledge created by humanity became a unit, the methodology of science and technology became the governing principle.

Science and technology have been developed on the sum of past knowledge. We learn only in a few hours the knowledge that old scientists like Galileo and Newton discovered over tens of years. Then we connect it to the next discovery. Thus, human knowledge is also increasing exponentially.

If we look at the evolution of evolution in a timeline, we should recognize that the evolution of evolution is being accelerated exponentially:

4.5 Billion years ago: Birth of the Earth

4 billion years ago: Birth of life.

2 billion years ago: Birth of cells.

1 billion years ago: Birth of multicellular organisms.

0.5 billion years ago: Birth of eyes (Cambrian explosion).

0.07 billion years ago: Birth of primates.

0.005 billion years ago: Birth of humanity.

0.00001 billion years ago: Birth of agriculture.

0.0000003 billion years ago: Industrial revolution.

0.00000007 billion years ago: Birth of computers.

1.3.2 Cell Division

Let's consider the number of cells as an example of a living thing that increases exponentially. You know that organisms grow by cell division. Imagine a fictional creature whose every cell doubles every day. If this organism was 10 cm tall one day, please guess how tall it would be the next day. No, it's not 20 cm. The correct answer is that the height the next day is the cube root of 2 multiplied by 10 cm. Doubling the height means doubling the width and depth, so the volume increases eightfold (Fig. [1.2](#)). Going back to the first question, if the number of cells doubles in a day, how many days will it take for the height to double? Yes, it's for 3 days. When the height doubles, the weight becomes 8 times. It takes 1 day for the weight to double, 2 days to quadruple, and 3 days to increase eightfold. If the fictional creature grows at this rate, it takes only 27 days for the organism that was 10 cm on the first day to become 50 m tall, which is the height of Godzilla. This is because 50 m is 500 times 10 cm (here, we use a commonly used approximation $500 \approx 512 = 2^9$), so the volume is approximately 2^{27} times. It is not surprising that an organism

of only 10 cm in height grows into Godzilla in 27 days by cell dividing once a day. This is the power of exponential increase.

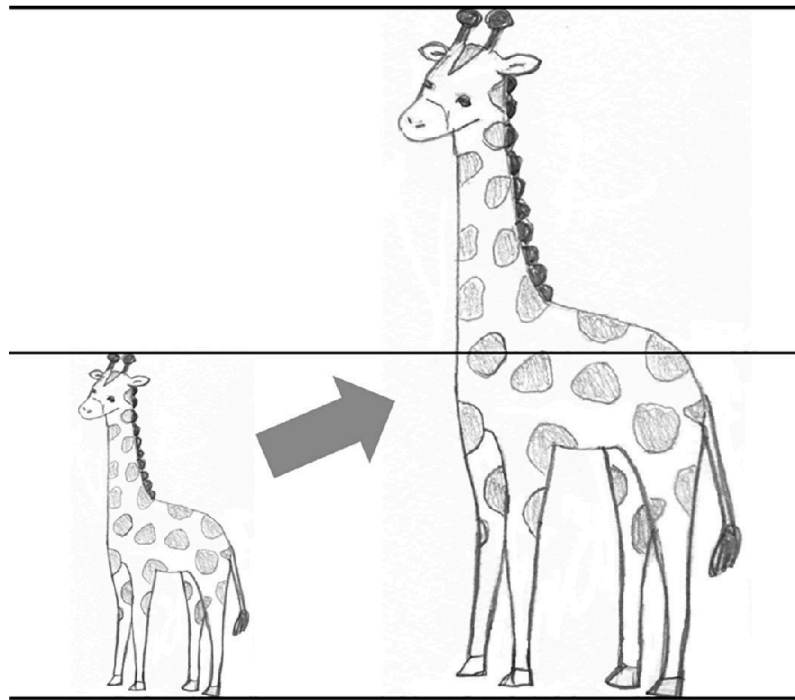


Fig. 1.2 Doubling the height results in an eightfold increase in volume

1.3.3 The Number of Your Ancestors

Every human has two parents, and four grandparents. That is, you have two ancestors one generation back, and four ancestors two generations back. In this way, the number of your ancestors increases exponentially. So, how many ancestors would you have if you go back 30 generations? It's about 1 billion. If we assume one generation is about 30 years, 30 generations is 900 years. So, you were born from about 1 billion ancestors who lived on this Earth 900 years ago.

But you may feel something is wrong here. The actual world population 900 years ago was just over 300 million. Generally, when a couple gets married, the husband has four grandparents, and the wife has four grandparents. But if cousins marry, the total number of grandparents is reduced to six. That is, if you trace your paternal and maternal ancestors, there must be some overlap. In other words, your father and mother can be traced back to one common ancestor at some point. In this sense, almost all people are related by blood somewhere.

1.3.4 Pyramid Scheme

Let's explain exponential growth with an example of a pyramid scheme or multi-level marketing. The job of a pyramid-scheme member is to recruit new members and to force them to purchase and sell products. If you recruit someone and they become a member, that person becomes your direct downline (child member). That is, you are the parent member of that new member. Then, a new member purchases a product from you and sells it to his/her child member with a certain amount of commission fee added. According to the rule of the multi-level marketing, a member

who earns a commission fee must pay a certain percentage (pay-up rate) of the commission fee to the parent member.

Each member has a mission to recruit more than N new members, and there is a quota for product procurement and sales. People who are invited to become a pyramid-scheme member by an acquaintance may hesitate whether to become a member or not, but then the parent member says something like “The earlier you become a member, the more income you will earn,” and urges them to join.

Is it true that “the earlier you become a member, the more income you will earn”? Let’s actually calculate it and check with concrete figures. Suppose that it is a rule that one member recruits at least two new members, purchases a product for 100 yen, sells it for 300 yen, and the quota is 100 items. Thus, the commission is 200 yen per item, for a total income of 20,000 yen. If you pay 50% (pay-up rate) of the commission fee to the parent member, you will have 10,000 yen left, and you will pay 10,000 yen to the parent member. Since that parent member has at least two child members, the total is two-fold 10,000 yen paid up. If there is a parent member above that parent member, you must pay up 10,000 yen, which is 50% of 20,000 yen, but if you think that the child member earned 10,000 yen while the parent member was sleeping, it’s a super easy business! Conversely, let’s consider the case where you have grandchild members. Suppose you have two grandchild members for each of the two child members, then a total of 40,000 yen is paid up from the four grandchild members to the two child members, and 20,000 yen is paid up from the two child members to you, and you pay up 10,000 yen to your parent member. In other words, you have indirectly received 10,000 yen from the four grandchild members. As you can see from a little calculation, as long as there are great-grandchildren and great-great-grandchildren beyond the grandchild members, you will receive 10,000 yen for each generation increase (Fig. 1.3).

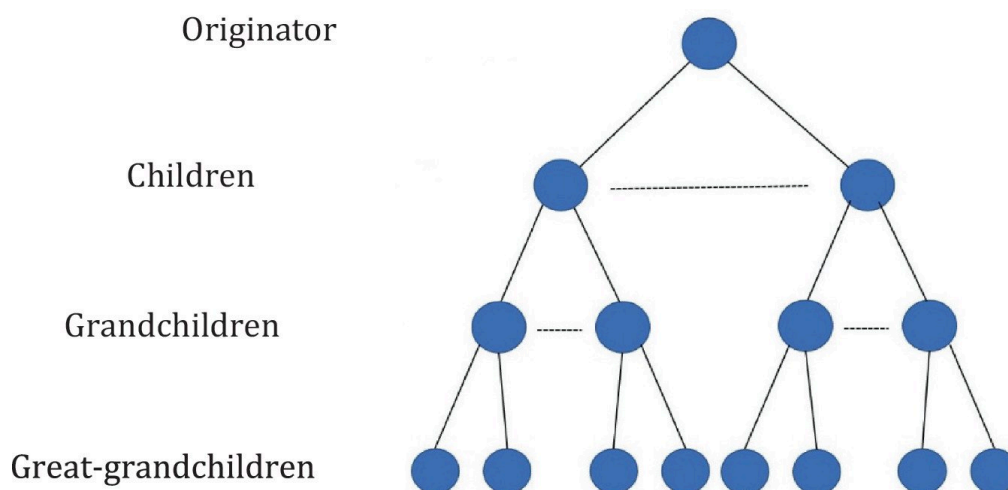


Fig. 1.3 Structure of generations in pyramid scheme

Now, let’s get to the point. Let’s reveal the answer to the question, “Is it true that the earlier you become a member, the more income you will earn?” The answer is qualitatively correct, but if you apply specific numbers and calculate, you will find that it breaks down immediately. The reason why this pyramid scheme works is the increase in members, but the speed of that increase is the problem. The mechanism of the pyramid scheme, “one member enrolls N or more new members,” leads to an exponential increase in the number of members. $N = 2$ is the so-called doubling game, which doubles in order: 1, 2, 4, 8, 16, 32. At first, these are small numbers, but at the 10th step, it’s

about one thousand, ⁴ at the 20th step, it's about 1 million, and at the 30th step, it's about 1 billion, rapidly inflating. In other words, this pyramid scheme reaches almost the entire population of the earth after 30 generations, and that's where it ends. Under these conditions, the "originator" who started this pyramid scheme can only obtain a mere 300,000 yen.

$N = 2$ shows that you can't generate much money. In a pyramid scheme, you must increase the proportion of the pay-up or increase the number of child members.

Figure 1.4 shows the increase in the number of child members in each generation. We increased it from 3 to 22. Although there is an increase approximately proportional to the number of child members, the maximum is only about 3.5 million yen, so it's not a big profit.

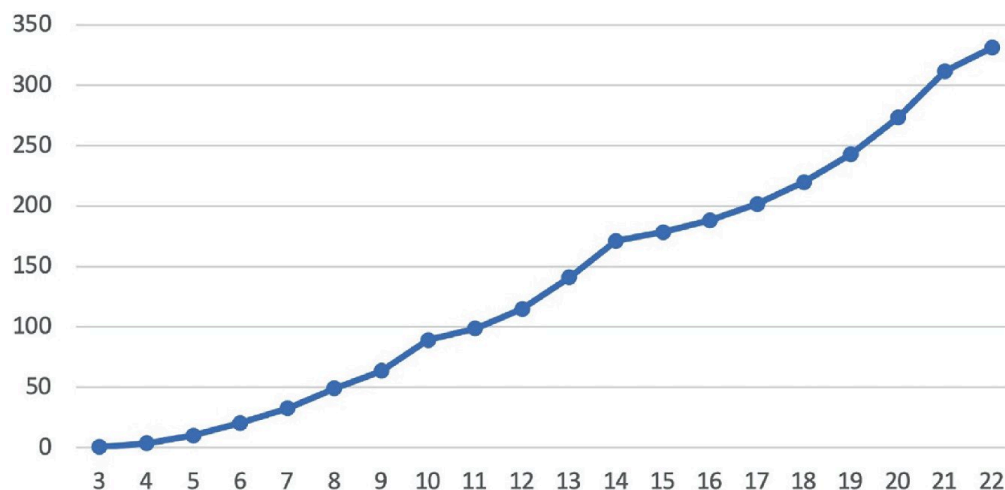


Fig. 1.4 Pyramid-scheme: number of child members and originator income (10 K yen) at pay-up rate of 50%

Figure 1.5 shows the change in the pay-up rate. We changed it from 0.1 to 1.0. Here, there is an exponential explosion of increase. A pay-up rate of 1.0 means that the parent takes all, so only the originator who started first gets the money, and there is no income for the other intermediate generations, so it does not work as a pyramid scheme. But as we saw above, at the pay-up rate of 0.5, it doesn't make much profit, so you might aim for 0.7 (income of 42 million yen) or 0.8 (income of 400 million yen). The closer the pay-up rate gets to 1.0, the more the originator's income increases. However, the rate of increase is difficult to discern in Fig. 1.5, so we introduce a logarithmic scale on the vertical axis. This is shown in Fig. 1.6. From 0.5 to the right, it's almost a straight line, so you can see that it shows an exponential increase.

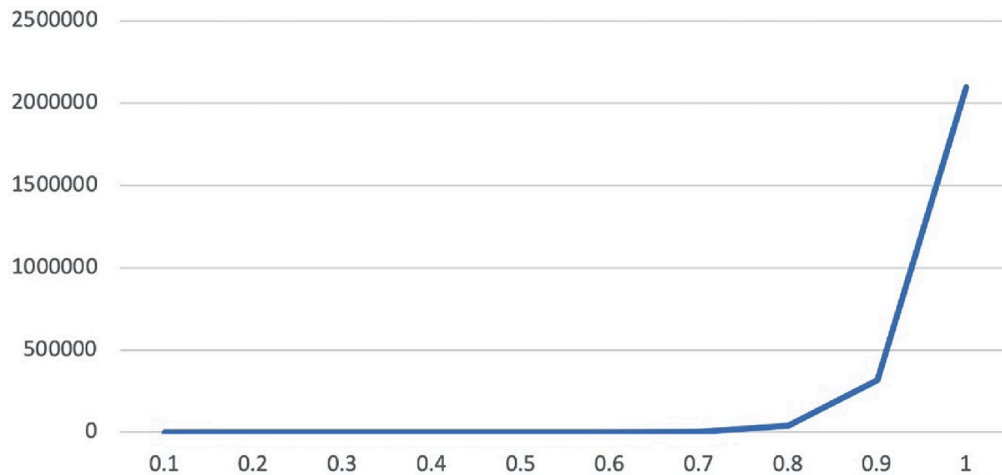


Fig. 1.5 Pyramid-scheme: pay-up rate and originator income (10 K yen)

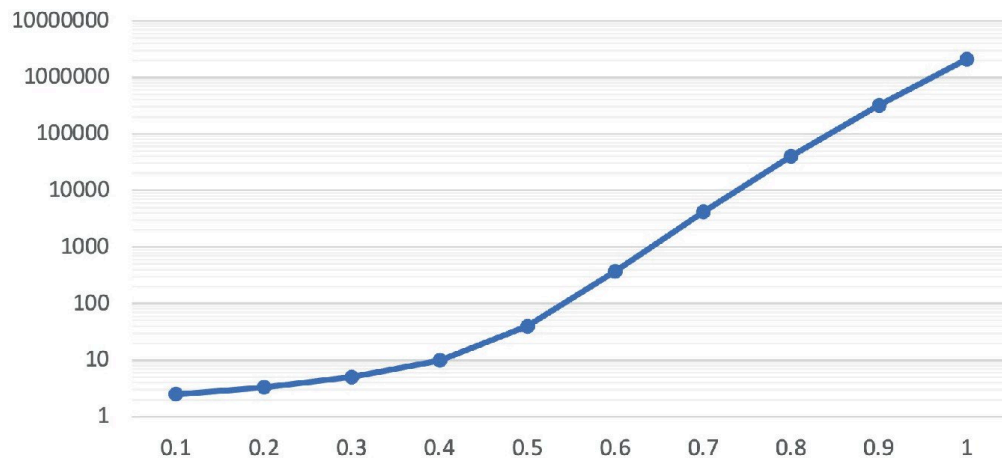


Fig. 1.6 Pyramid-scheme: pay-up rate and originator income (Log scale in 10 K yen)

Either way, when you calmly look at the figures relevant to the pyramid scheme shown here, it would seem odd that many people in the world are being duped by the pyramid scheme. There may not be anyone who profited a lot from the pyramid scheme. It is probably because people have a vague understanding of exponential increase and suppose that it is naively applied to their own profit in the pyramid scheme.

1.3.5 Traveling Salesman Problem

Let's consider a more realistic example, there is a problem called the traveling salesman problem. This is a problem in which a salesman finds an efficient stroke order to travel multiple cities for sales activities, i.e., making the plan of a visit, stopping only once in one city with saving time and transportation costs. It has been known as a typical example of exploration for a long time and one of the most difficult problems to find the optimal solution. It is because the computational complexity increases exponentially as the number of cities visited grows.

Here, we will not give a solution to this problem but just an important notice thinking of actual logistics. For example, when a courier delivers multiple packages to multiple homes, in what order is it most efficient? In reality, there are other conditions that we have to take into account such as the specified time of delivery and the total mileage, which make the problem more complicated.

Without a computer, the optimal solution cannot be obtained. From the standpoint of computational thinking, the important thing is not to solve this problem, but to know that this problem is difficult.

1.3.6 Avoiding Exponential Explosion

As we have seen so far, when an exponential explosion occurs, the amount of computation increases rapidly, making it impossible to compute with practical computational resources (time and memory that are available). There are some known ways to avoid this:

- Divide and conquer: This is a method of dividing the problem into independent units and solving each one (Sect. [1.4](#)).
- Settle for an approximate solution: This is a method of giving up on complete computation and accepting an approximate value. For example, the traveling salesman problem (Sect. [1.3.5](#)) inevitably causes an exponential explosion when calculated normally, so several methods have been developed to find approximate solutions.
- Record partial solutions: As mentioned at the beginning of this section, exponential explosions often have a structure where you repeatedly use what you have accumulated so far, so you record what you have calculated in a table at each time step and use the values in this table for the second time.

1.4 Divide and Conquer

Previously, we discussed how an exponential explosion in computations can become unmanageable. Essentially, this explosion occurs because we have to consider all possible scenarios or cases of the subject. Therefore, let's first consider properly dividing the entire subject into small parts that do not cause an exponential explosion. If there is no relationship between these parts, or if any relationship can be ignored, the processing of each divided part will be light. This might allow us to calculate the entire subject without causing an exponential explosion. In other words, if we can divide it well, we can govern it with a light effort. In the best-case scenario, the entire computation can be completed with effort proportional to the number of divided parts. In other words, the original computational complexity of

$$N_1 \times N_2 \times \cdots \times N_k$$

can be reduced to only

$$N_1 + N_2 + \cdots + N_k.$$

When you hear divide and conquer, you might think of the ancient Roman Empire, which governed vast territories, Western colonial management, or the Edo Shogunate. The key to divide and conquer here was to sever the solidarity and cooperation between regions, ethnic groups, and countries (sometimes, the rulers even incited mutual resentment and hatred). By severing these, they were able to govern each divided region, ethnic group, and country with minimal effort in total.

1.5 Recursion

According to the Super Daijirin, a Japanese dictionary,

Recursive

1. The result of one's actions returns to oneself. Feedback.
2. In mathematics, etc., the thing to be defined appears in the definition itself.

The second definition is our concern here.

"Recursive" is one of the important and powerful methods in programming, which allows many problems to be described concisely by including calls to itself in the program. First, I would like to start with an example of what recursion is.

This phenomenon can also be seen in everyday life. For example, in a pair of mirrors facing each other, image B is reflected in A, and A is reflected in the image of B, and this continues indefinitely.

Column 3 Mutual Exclusion

If multiple processes access a single resource at the same time, one result may overwrite the other, causing problems. Mutual exclusion is to ensure that only one process can access at a time.

When accessing a single resource, you first lock the resource to prevent other processes from accessing it. This is called a lock. You can safely access the resource after locking it and unlock it when you're done. If another process wants to use the resource while it's locked, it has to wait until it unlocked. This is how mutual exclusion is achieved. This book is also written by multiple authors, so we use mutual exclusion when modifying files.

The key to efficient mutual exclusion is to keep the interval between lock and unlock as short as possible. To achieve this, we carefully consider what should be shared resources and devise algorithms accordingly.

Mutual exclusion is also used in railways. Nowadays it is computer-controlled, but in the past, to prevent more than one train from entering a single track, only one tablet was prepared for that section (called a block section), and trains without it could not enter. This was featured in the Japanese movie "Railroad Man" (original title "Poppoya") starring Takakura Ken.

Speaking of which, it seems that there are many good hackers among model railway enthusiasts. (Or should I say, there are many model railway enthusiasts among good hackers?).

There is a Rakugo (Japanese traditional storytelling) called "Hell's Eight Scenic Spots of the Dead". A Rakugo storyteller who goes to hell will be able to go to paradise if he makes Enma (hell's guardian) laugh, so he tells Enma a Rakugo story.

Enma: Oh, a Rakugo storyteller, I love Rakugo. What kind of story is it? Rakugo storyteller: How about "Hell's Eight Scenic Spots of the Dead"? Enma: Oh, let's listen to it.

So, the same Rakugo starts in the Rakugo. Then, "Hell's Eight Scenic Spots of the Dead" starts again in that Rakugo, and it doesn't stop. In reality (?), however, Enma should stop it halfway through.

Describing the story in the order in which things happened like above, it continues to unfold indefinitely. However, if you think about it the other way around, you can describe repetitive tasks or computations in a short way. If you can capture various phenomena in everyday life recursively, you will be able to understand things in a simpler way. For example, the predicate “append” that connects two lists in the programming language Prolog is defined as follows:

```
append([], Z, Z).  
append([E|X], Y, [E|Z]) : - append(X, Y, Z).
```

The first line is the termination condition, and computation stops when the first argument becomes an empty list. If the first argument is not empty, the control goes to the second line; it is decomposed into the head element *E* and the rest *X*, and the predicate is to append *E* and *Z* at the place of the third argument. Here, *Z* is the result of append of *X* and *Y* obtained from the recursive computation of `append(X, Y, Z)`. It is enough to see that the second line is recursive. The LOGO language’s “word list” program (Sect. 4.4) is also written recursively.

This is a strategy of dividing a large task or computation into smaller ones (see also Sect. 1.4).

A famous example where recursive solutions are effective is the Tower of Hanoi puzzle (Fig. 1.7). This involves moving stacked discs one at a time to move the entire pile to another rod. You must not stack a larger disc on top of a smaller one. This is easy to think about recursively (Fig. 1.8).



Fig. 1.7 The Tower of Hanoi puzzle

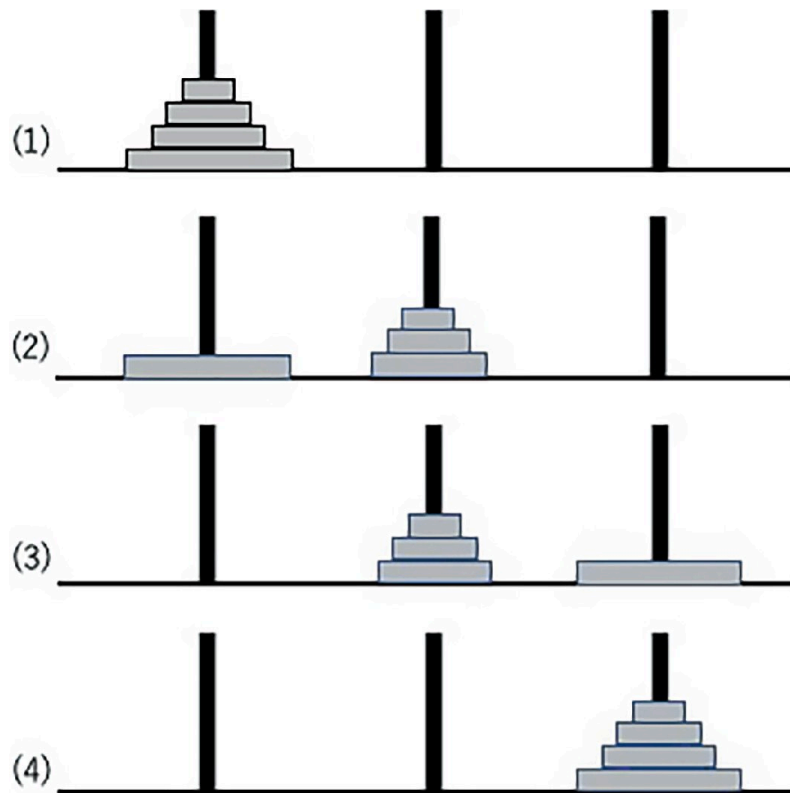


Fig. 1.8 How to move discs in the Tower of Hanoi

1. In order to move N discs to the target rod
2. Move $N - 1$ discs to a different rod named A (recursive procedure)
3. Move the remaining 1 disc to the target rod
4. Move the $N - 1$ discs stacked on rod A to the target rod (recursive procedure).

In other words, to move N discs, you need to use the operation of moving $N - 1$ discs twice, in which the slightly smaller move operation is called recursively.

1.6 Information as a Worldview

“Computational thinking” may be rephrased as “information theoretic thinking.” This emphasizes information as a worldview alongside matter and energy. In other words, it advocates viewing the world from the perspective of information and its operations.

Have you ever read the flash fiction written by a Japanese novelist Shinichi Hoshi titled “The Net of Voices”? One of the characters claims that information is energy:

In an era without knowledge of hydroelectric power, people probably didn't think of water as an energy source, and they certainly didn't dream that uranium-containing ore was a tremendous source of energy. Uranium ore was just a rock. Isn't it natural to think that the source of energy is the knowledge of nuclear power? In the early days of the nuclear age, information about atomic bombs was a high-level state secret, and there were scholars who were executed for giving it to foreign countries. Even then, picking up uranium ore was nothing special. The source of energy is information. (Translated by the authors)

Shinichi Hoshi, "The Net of Voices" (Kadokawa Bunko)

In other words, even if raw material exists, without the knowledge or information on how to extract energy from it, it is just a mere substance. This is why it is argued that information is a worldview. Whether you see the stone in front of you as just an ore, or as a valuable energy source, that difference is information.

What this example shows is that there are things that cannot be seen just by looking at the world from the perspective of matter and energy. Einstein's famous equation $E = mc^2$ describes the relationship between the mass of matter m and energy E . This equation explains the energy produced during nuclear fission and fusion, but it alone can neither build an atomic bomb nor unfortunately a fusion reactor. For this purpose, you need information on how to build an atomic bomb and a fusion reactor.

The general term "information" is used in a variety of contexts: Since around 1980 with the spread of personal computers, and since around 1990 with the spread of the Internet, the term "informationization" has become more common. This refers to the use of information technology when companies think about new products or services, or when they want to make individual processes in their operations more efficient and smarter. On a personal level, using information technology as a tool to make daily life more comfortable and effective is also included in informationization.

Unfortunately, there are still not many examples in Japan or around the world of using information technology to revamp traditional workflows or create new business models. Currently, attention is focused on GAFA (Google, Amazon, Facebook, Apple), which have become huge companies through information technology, but the world's information technology engineers intuitively understand that the real transformation brought about by information does not stop at the level of GAFA. They are convinced that information is just on the cusp of further fundamental changes to the world and society we live in.

Looking towards such a future, in recent years, the concept known as "Digital Transformation", or "DX" for short, has been proposed. It aims to launch innovative new businesses using information technology and/or digital technology and fundamentally improve people's attitudes and behaviors towards life. DX is about understanding the true value of information and practicing the worldview of information.

1.7 What Computational Thinking is Not

The discipline centered on the worldview of information is "Information Science." Computational thinking is based on the thinking methods created by researchers and engineers of information science who study information. Let's look at how this information science differs from existing disciplines such as physics and mathematics.

1.7.1 Differences from Physics

Let's continue with the example of uranium from the aforementioned book "Voice of the Net" by Shinichi Hoshi. From a physical perspective, the atomic number of uranium is 92, and its atomic weight is 238. The atomic number is the number of protons in an atom, and the atomic weight is the weight of the atom, which is roughly the sum of the number of protons and neutrons.

Therefore, we know that uranium 238 is made up of 92 protons and 146 neutrons. As you may know, uranium is a radioactive substance that spontaneously undergoes nuclear fission to split into smaller atoms. This fission occurs slowly in nature, such as inside the Earth.

If we go back to a more fundamental level of matter, matter is made up of atoms, and those atoms are composed of a nucleus made up of protons and neutrons, and a cloud of electrons around it. It is known that atoms generally have an equal number of protons, which carry a positive charge, and electrons, which carry a negative charge, so atoms are electrically neutral. How matter changes, and how energy transforms through heat and motion, is the worldview of "physics", and the knowledge of physics.

Therefore, physics also deals with a lot of "hows", that is, information. But this information is generally descriptive information about the state and movement of matter and energy. This is why physics is said to be descriptive. It does not include information on how to manipulate matter and energy, or how to build machines like nuclear reactors that manipulate and control matter and energy. This kind of information and knowledge for creating something by human hand is dealt with in the field of "engineering", not physics.

The study of observing the state and movement of objects and discovering principles and laws therein is generally called "science." The definition of science is difficult, but let's consider it a method of objectively organizing and understanding things. Things exist first, and then humans understand them. Understanding the laws of nature is natural science, and physics, which deals with matter and energy in the natural world, is a branch of science. Nevertheless, physics is at the core of natural science. In fact, the methodology of natural science has developed based on physics.

Both science and engineering revolve around the loop of hypothesis formation and its verification. In science, to achieve their goals, hypothesis formation involves observing phenomena and inferring the underlying principles or laws. These inferred principles or laws are called "hypotheses." These hypotheses are then tested through experiments to verify their correctness. By going through this loop, principles or laws that explain phenomena more accurately and consistently are gradually being developed.

The situation in engineering is pretty much the same. In engineering, the hypothesis formation involves devising methods to operate the target as intended or to realize the desired function. The correctness of the hypothesis is verified by implementing the method. By going through this loop, the methods to operate the target as intended or to realize the desired function are gradually being developed.

Now, let's shift our focus from matter and energy to information. Information science is a discipline that centers on the worldview of information. In information science, we deal with methods to generate new information based on various information. This is called "inference" or "prediction." Let's introduce a simple example of inference. Let's consider three propositions A, B, and C as follows:

A: Socrates is human,

B: Nightingale is human,

C: If you are human, you get hungry.

From A and C, we can conclude that Socrates gets hungry. From B and C, we can conclude that Nightingale gets hungry. This type of inference is called “deductive reasoning.” Also, if we interpret the result of the inference as something that will happen in the future, it becomes a prediction that Socrates will get hungry.

Let’s introduce another simple example of inference. Again, let’s consider three propositions D, E, and F as follows:

D: Socrates and Nightingale are both human,

E: Socrates gets hungry,

F: Nightingale gets hungry.

From these three propositions, we can infer that “if you are human, you get hungry.” Of course, human beings are not just Socrates and Nightingale, but if Socrates and Nightingale get hungry, we might assume that other human beings will also get hungry. This type of inference is called “inductive reasoning.” What should be noted here is that the conclusion “if you are human, you get hungry” is not necessarily 100% correct. That’s what inductive reasoning is. It only says that it will probably be correct in most cases. If you think of it as a prediction, it’s natural that it can be right or wrong.

Some readers may be wondering by now, did not information originally exist in the natural world? Does the concept of information only make sense when there are human beings? Information has two aspects: information as inherently physical existence from the beginning and information as a concept that has meaning only when human beings exist. In fact, there are concepts shared between physics and information science. One of them is “entropy.” Entropy, simply put, is the “degree of disorder.” For example, a snowflake is a neatly arranged set of water molecules, a state of low entropy. When heat (energy) is added, the entropy increases (in everyday terms, it “melts”) and becomes water. Water is a more disordered state than ice. Whether this is disordered or not can be determined by how many possible states that can be taken. Even though there are various types of snowflakes, the number of states called crystals is far fewer than that of states called water.

If left alone, snow will turn into water, but the reverse does not happen. In other words, matter changes from a low entropy (ordered) state to a high entropy (disordered) state, but something needs to be done to cause the reverse. There is a saying, “Spilt water will not return to the tray”, which means that water spilt off from a tray will not return to a tray if left alone. In this way, entropy can only change in an increasing direction if left alone. This is called the “law of increasing entropy.”

From here, I will make a bit of a bombshell statement regarding the number of states. It is human beings who determine the number of states for molecules contained in crystals and water. In fact, comparing the numbers of states of snowflakes and water are physically meaningless. Under the condition that the number of snowflake molecules and that of water molecules are the same, the numbers of their states are just the same. Moreover, this is because it is impossible to distinguish each molecule in the first place. However, we human beings can recognize snowflakes as having specific patterns, right? Even for water, if you put water in a cup, water should maintain a specific shape. At that time, you human beings likely recognize it as one “state” even though there will be a huge number of states in a physical sense.

Let's explain entropy by using a shuffle of cards. Since it's tough to illustrate with 52 playing cards due to space constraints, consider we have a reduced sized set of playing cards: spades, clubs, hearts, and diamonds, each with a jack, queen, king, and ace. There are 16 playing cards in total. If these cards are separated by type and arranged in ascending order as follows:

(1) ♠J ♠Q ♠K ♠A ♦J ♦Q ♦K ♦A ♥J ♥Q ♥K ♥A ♠J ♠Q ♠K ♠A,

We can say that the entropy is low. If you shuffle this, the order may be changed and the order becomes more random, and the entropy increases.

Now, if separated by number and arranged for the four types in order as follows:

(2) ♠J ♦J ♥J ♠J ♠Q ♦Q ♥Q ♠Q ♠K ♦K ♥K ♠K ♠A ♦A ♥A ♠A,

This can also be said to be a state with low entropy.

What if we would decide that the following is a regular state.

(3) ♥K ♠A ♥J ♠J ♦Q ♥Q ♠A ♦K ♠J ♦A ♠Q ♠K ♥A ♦J ♠K ♠Q?

There are no simple rules. However, physically, there is no reason to distinguish (1) and (2) from each other. Why do we think (1) and (2) are different from (3), although (1), (2) and (3) are certain states of the playing cards, respectively? It's our cognitive mechanism that is creating the "difference." (1) and (2) can be described and recognized by simpler rules, but (3) cannot.

When you shuffle a well-ordered deck of 52 playing cards, it gradually becomes random. The probability of accidentally returning to the original state is very low, practically zero.⁵ However, the probability of reaching any specific state is equal and also practically zero.

There are two kinds of things in the world; one is the things that are orderly and understandable to human beings, and the other is the things that appear random. Therefore, "entropy cannot be defined without humans." This phrase was written in "The Order of time" by theoretical physicist Carlo Rovelli. This issue is interesting in itself, but if we delve deeper, we will go off on a tangent, so I will stop here. If you are interested, please refer to the book by Carlo Rovelli.⁶

1.7.2 Differences from Science and Engineering

There is a term "computer science." Can the study of computers be a science? Computers do not exist in nature. They are apparently human-made products of engineering. However, there are already fields called "computer science" or "computer engineering" in the world. Don't you readers feel a slight discomfort with the term "computer science"? I would like to solve this mystery for a moment.

In Japanese, there are expressions like "science and engineering" or "science and technology." In English, there are words like "science and art" as well. Interestingly, when we say art in Japanese, we think of fine arts such as painting, but art in English means technology and skill. Martial arts are techniques of fighting, and liberal arts are techniques (or knowledge) for being liberal. Organizing these terms in the following figure might make it click.

Column 4 The Law of Increasing Entropy and the Arrow of Time
--

Physical equations (such as Newton's equations of motion or the equations of relativity) contain a variable. If this is replaced with, in other words, if time is reversed, the equation still holds. This means that if you rewind a video capturing a physical phenomenon, you cannot distinguish it from playing it forward. However, there is one difference. friction. Without friction, you wouldn't be able to tell if a billiard ball's movement was being rewound, but with friction, the ball gradually slows down and eventually stops. If you rewind this, the stopped ball starts moving, which is strange.

Friction is an example of the law of increasing entropy, which means that the ball gradually turns into heat as it moves. The law of increasing entropy is the only thing in physics that determines the direction of time. The existence of time has long troubled physicists. Many books have been published on the "arrow of time".

Recently, a physicist has emerged who claims that this entropy is due to human cognitive function. His name is Carlo Rovelli. This is a unique perspective. He argues that the flow of time for humans exists because humans exist in this universe. The conclusion may still be far off, but I think this is a rare example of a physicist thinking computationally.

Now, I mentioned that physics is called "descriptive", and contrastingly informatics is called "operational", and specifically creates new information, it is called "constructive." Compared to physics, the worldview of information is well expressed by these words "operational" and "constructive."

Computational thinking as presented in this book is part of engineering thinking. In this book, we use the terms "computer science" and "computer scientist." According to Fig. 1.9, we Japanese should instead use "computer engineering."

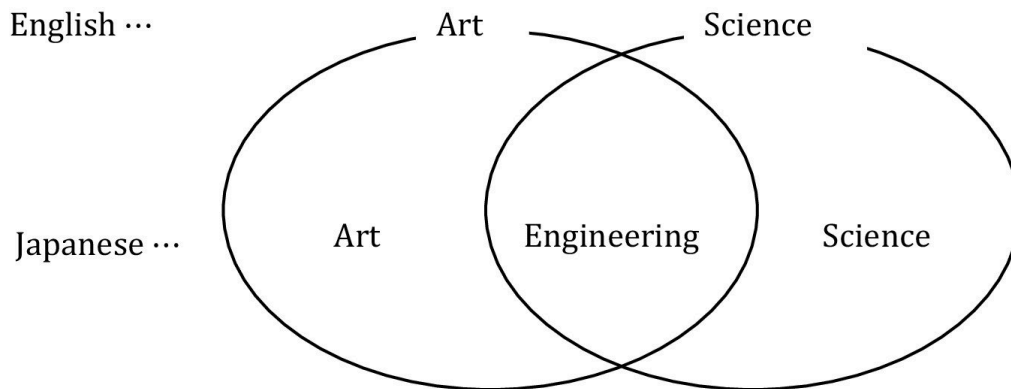


Fig. 1.9 The relationship between Japanese Art, Engineering and Science, and English Art and Science

1.7.3 Differences from Mathematics and Logic

Mathematics manipulates information differently from physics. Mathematics combines one piece of information with another to create new information. There is a divide among researchers as to whether mathematics is something created by humans or something that exists naturally.

There are those who say that every natural number such as 1, 2, 3 etc. is invented by humans, while there are those who believe that they exist in nature even without humans. Logic and mathematics are slightly different from natural sciences, but it wouldn't be unnatural to call them science collectively.

As an example of such mathematics, we will take up Euclidean geometry. Euclidean geometry is roughly the same as the geometry that we Japanese learn in elementary school. To explain Euclidean geometry precisely, it starts with the definition of a point and the definition of a line. A definition is a sentence that precisely determines what properties a concept has and how it is made. Then, properties called axioms are given using that definition. An axiom is the starting point that can be assumed to naturally hold without proof, such as the transitive law and reflexive law in logic.

Furthermore, there are properties called postulates, which are what would happen if we assumed they were correct (these are also sometimes called axioms); for example, “parallel lines do not intersect” and “there is only one line connecting two points.” These postulates cannot be proven correct.

Anyone can define anything freely and give axioms and postulates freely. Then, properties derived from axioms and postulates by deductive reasoning are called theorems; for example, the sum of the interior angles of a triangle is 180 degrees, Pythagoras’ theorem, etc. However, if contradictory axioms are given, contradictory conclusions will be derived, and if unrelated postulates are given, no meaningful conclusion will be derived. These meaningless conclusions cannot be called theorems.

As another example, let’s look at the definition of natural numbers. There are various ways to do this, but here we will introduce a definition using sets. Since they are natural numbers, we create them in the order of 0, 1, 2, etc. We use existing information to create new information. First, let’s assign the symbol ‘0’ (zero) to the empty set. Let’s write it as $0 = \{\}$. The empty set does not contain elements and is unique, that is, there is only one empty set in the world.

Since all numbers can be represented as sets, let’s define $n + 1$ as $n \cup \{n\}$. This term means the set that is the union of the set that represents n and the set that has only n as an element. In this way, the set of $n + 1$ has one more element than n , and the newly added element $\{n\}$ is obviously not included in n . Therefore, when you have made 0, 1, $\dots$, n in order, you can always create a set that is different from the sets you have created so far. Let’s see what kind of sets are actually being derived in order.

$$\begin{aligned} 1 &= 0 \cup \{0\} = \{\} \cup \{\{\}\} = \{\{\}\} \\ 2 &= 1 \cup \{1\} = \{\{\}\} \cup \{\{\{\}\}\} = \{\{\}, \{\{\}\}\} \\ 3 &= 2 \cup \{2\} = \{\{\}, \{\{\}\}\} \cup \{\{\{\}, \{\{\}\}\}\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\} \end{aligned}$$

It goes like this. However, there are too many brackets and it’s hard to understand what’s going on. Would it be easier to understand if we rewrite it as follows?

$$\begin{aligned} 1 &= 0 \cup \{0\} = \{\} \cup \{0\} = \{0\} \\ 2 &= 1 \cup \{1\} = \{0\} \cup \{1\} = \{0, 1\} \\ 3 &= 2 \cup \{2\} = \{0, 1\} \cup \{2\} = \{0, 1, 2\} \end{aligned}$$

In this way, it should be clear that the newly created set can be written as follows:

$$n + 1 = \{0, 1, \dots, n\}.$$

It is also apparent that $n + 1$ is different from any of $0, 1, \dots, n$. Here, $0 = \{\}$ and $n + 1 = n \cup \{n\}$ correspond to the axioms, and

$$\begin{aligned}1 &= \{0\} \\2 &= \{0, 1\} \\3 &= \{0, 1, 2\}\end{aligned}$$

correspond to the theorems.

In mathematics and logic, new knowledge and information called theorems are created from axioms and postulates. Above, I wrote that creating new information based on specific procedures is called computation, but since theorems are new information, the operation of creating theorems can also be considered computation. In this sense, part of mathematics and logic can be considered the equivalent of computational thinking or the worldview of information.

So, what are the points where mathematics and logic differ from computational thinking? In mathematics and logic, there are cases where “you don’t have to know the specific procedure of computation.” If you are good at mathematics, you may have encountered cases where you can think of many ways to solve a problem. That’s right, there are generally multiple ways from axioms and postulates to theorems. Therefore, “a method based on a specific procedure” usually means that you know at least one of the several specific procedures. However, there are also cases where you only know that there exists a path consisting of specific procedures, but you don’t know what it is.

Mathematicians sometimes use the phrase “principally possible.” What it means to be computationally possible in principle is that it doesn’t matter how much time it takes, or how much memory or disk space is consumed. There is a big difference between what can be computed in principle and what can be computed in reality. Even if it is computationally possible in principle, if the computation time takes hundreds of years, or if the required memory size is larger than the number of atoms in the universe, the execution of the computation is practically impossible. In computational thinking, we are aware of whether the computation can be completed within a realistically acceptable time range, or whether the computation can be executed with the memory capacity and disk capacity that are realistically available. Memory capacity and disk capacity correspond to the maximum amount of data that must be recorded during the computation. Mathematics covers ideal and limit states, but computational thinking is limited to the range that can be calculated in practice.

In logic and mathematics, there is a proof method without calculating a certain value known as “reductio ad absurdum.” In reductio ad absurdum, we take the approach of starting the proof without touching the specific computation procedure, just by assuming the existence of a certain mathematical concept. For example, when trying to prove a certain proposition to be correct, if it is difficult to prove it directly, we consider what conclusion would be under the assumption that the proposition is wrong. Then, if a contradiction is derived by assuming that a certain proposition is wrong, it is concluded that the proposition was correct. Reductio ad absurdum is more straightforwardly called proof by contradiction. Here, a contradiction refers to the state that for a proposition P , “ P ” and “not P ” are held at the same time.

Reductio ad absurdum (proof by contradiction)
--

To prove a proposition P , instead of directly showing its correctness, it is a method of proving that P is true indirectly by showing that it contradicts when assuming “not P ”.

Now, let's say we want to prove the proposition “There are infinitely many prime numbers.” However, even using the fastest supercomputer in the universe, you cannot exhaustively generate an infinite number of primes. It takes an infinite amount of time to generate an infinite number of primes. So, it's hard to prove directly. Therefore, we assume that “there are only a finite number of primes, and all of these finite primes are $p_1, p_2, \dots, p_n$.” That is, the negation of the proposition to be proved has been assumed. The point here is that it does not provide any specific method or value for calculating $p_1, p_2, \dots, p_n$, it only assumes that such finite primes exist. Next, consider the new number $p = p_1 \times p_2 \times \dots \times p_n + 1$. Then, for all $i = 1, \dots, n$, it is clear that p is larger than p_i , and p cannot be divided by p_i because it always leaves a remainder of 1. The new number p actually meets the conditions of a prime number. Therefore, it contradicts the assumption that “there are only a finite number of primes, and the entire finite number of primes are $p_1, p_2, \dots, p_n$.” Therefore, it was concluded that “there are an infinite number of primes.” This is a famous proof by Euclid for the proposition that “there are an infinite number of primes.”

Not only *reductio ad absurdum* is often used in mathematics, but also it appears in detective novels. For example, suppose there is information that only the culprit would know in a murder case and the murder weapon has not been announced. If the suspect is assumed to be “not the culprit”, the suspect should not know about the weapon. Let's say the detective cornered the suspect and made him/her say “I couldn't possibly have a gun”. The weapon could have been a knife, and the only one who could specify a gun would be the culprit, so such reasoning becomes valid. In other words, if the suspect is assumed to be “not the culprit”, it contradicts the fact that the suspect cannot know that the weapon is a gun. Therefore, it is reasoned that the suspect must be the culprit.

In mathematics, computation was a powerful method to generate new information, but logic, represented by *reductio ad absurdum*, is also a powerful method to derive new conclusions and generate new information. Computation and logic, both powerful methods to generate new information, differ greatly in the point of whether computation calculates specific values in order according to specific procedures and reaches a conclusion. Then, the message conveyed in the book is that thinking and executing these specific procedures would be very useful in everyday life.

1.7.4 Difference from Philosophy

Philosophers often talk about things “in principle.” It's similar to mathematics.

We are going to discuss the thought experiment of “Chinese Room” by John Searle,² a typical argument of philosophers. This is a fictional experiment to argue that machines never “understand.” The setting of the Chinese Room is as follows.

- There is an American person in the room who does not understand Chinese.
- A piece of paper with a question written in Chinese is thrown into the room.
- The person does not understand Chinese but has a thick English manual written in English about Q&A in Chinese.
- The person follows the Chinese characters on the paper with a question in order (it's hard to find unreadable characters, but let's ignore that), works intensively according to the

instructions in the manual, and eventually gets a response in Chinese.

- The person writes down the response in Chinese on a piece of paper and returns it out of the room.

From the perspective of a person outside the Chinese Room, if you put in a question in Chinese, you get a response in Chinese. So, the room seems to understand Chinese. However, when you peek into the room, there is the person inside who does not understand Chinese at all, and of course, does not understand the content of the question or answer. Searle argues that AI programs are the same as the person inside, even if they appear to behave intelligently, they do not have intelligence.

There are various counterarguments from the AI side, one of which is about computational complexity and descriptive quantities. I don't think a manual that anticipates all questions can be created in the first place, but even if it could, how thick would it be? One AI researcher wrote a paper trying to estimate that amount and proved that even simple addition exceeds the total number of atoms in the universe, as it is difficult to measure the description of a general manual.

Searle does not have an estimate of the amount of work. He describes the thought experiment "in principle." The problem is discussed as if an answer can be created in a few minutes or at most a few hours, but it turns out that this is not the case.

In other words, the difference between philosophy and computational thinking is whether to consider the resources (memory and computation time) required for computation. Estimating this computational quantity is quite important for us to solve various problems in our daily lives.

The "Frame Problem" is also famous as a philosophical problem discovered by AI researchers, John McCarthy and Patrick Hayes. Originally, it was a problem that it is difficult to describe how the situation is changed by performing a certain action. Despite only a part of it being changed, a large amount of unchanged framework (frame) must be described, hence it was named as such. In other words, the Frame Problem means that when trying to describe an action, the effort to explicitly describe both what is changing and what is not changing, and to distinguish between them, may increase exponentially in description and inference.

If we don't think about specific computations, we don't feel the frame problem in our daily lives. That was why philosophers didn't notice it. However, when McCarthy tried to write the specific computation down as an AI program, he faced this problem. Even in the simple problem called "block world", where they were just rearranging blocks in early AI research, they couldn't write an AI program that realized specific computation rules.

One of the important things in computational thinking is to know that there are limits to computation. Even if the problem is describable in principle, there will be no end to the list of specific conditions. This may not have been a problem in mathematics or philosophy where it is finished with "it is possible in principle." This problem cannot be given a solution even by computational thinking, but it is important to recognize that there is such a hard problem. When talking with friends or family in daily life, you may find your understanding of "it is possible in principle" is at odds with theirs and you may not even realize it.

1.7.5 Differences from Programming

It seems that there are many people in the world who understand that computational thinking is the ability to write computer programs. But that's a mistake. There are many people who can program but cannot think computationally. Conversely, there are many people who can think computationally but don't need to write a program. So, what is the difference between being able to program and doing computational thinking?

First, a “program” is a blueprint for realizing the computation you want to achieve, which is a procedure that describes the specific steps in order. “Programming” refers to the task of creating a program. The machine that executes the program is called a “computer.” The languages (natural languages) used for human communication include English, Arabic, and Japanese. Similarly, writing programs, “programming languages” such as Python, Java, and C are used. In these programming languages, like mathematics, a very small number of well-defined elements are given at first, and from there, elements are combined according to strictly defined rules to gradually construct complex computations. Strict rules for executing a program are also set in detail, and new information is generated one after another as the execution proceeds. This is what it means to compute in real life.

Assuming we have a Python program here, a program that computes and generates the same information as the Python program could be written in Java or C. For example, a program that makes an LED blink at a frequency of once per second can be written not only in Python, but also in Java, or C. This is the same as being able to greet or express gratitude in English, Arabic, or Japanese. The reason why programs having the same function can be written in different languages is because in the brain of a programmer, there is a procedure at the abstract level computation that does not depend on the existing programming languages used including Python, Java, and C. Such a procedure at the abstract level computation is called an “algorithm.”

As an example of an algorithm, let’s introduce Euclid’s algorithm, which is also known as the oldest one in the world. This is to find the greatest common divisor (GCD) of two natural numbers (positive integers) as follows.

For two natural numbers $a, b (a \geq b)$, if the remainder of a by b is r , the greatest common divisor of a and b is equal to the greatest common divisor of b and r . Using this property, if the calculated value of the remainder of a by b is x , then b is set to the new a and the remainder x is set to the new b . Such that, the calculation of the remainder of a by b is repeated until the remainder becomes 0. Then, b when the remainder becomes 0 is the greatest common divisor of original a and b .

This is probably one of the things that everyone has learned in school. What I want you to pay attention to here is the method for explanation of the computation procedure. In the explanation, there is no mention of specific programming languages such as Python, Java, and C. However, based on the explanation, everyone can implement Euclid’s algorithm in Python, Java, or C. This is because the computation procedure is described at an abstract level.

The process of thinking about an algorithm and implementing the algorithm in a specific programming language is called “programming.” The first half, where you think about the algorithm, is sometimes called “design” or “planning”, and the second half, where you implement it in a specific programming language, is sometimes called “programming” or “coding.” In the latter part, whether you are programming alone or with multiple people, various programming methodologies and practices have been developed. Those methodologies and practices lead to the field of study known as software engineering.

According to Wing, “thinking like a computer scientist” is the definition of computational thinking. In other words, it is the application of the thinking methods developed by researchers and engineers in informatics, who study information, to practical tasks in real daily life. It is the computer scientists who have come up with many efficient and readable algorithms and have

devised programming techniques or methods. Furthermore, computer scientists have not only pursued programs, but also the hardware that runs the programs.

The machine called a computer that executes a program is a physical entity wired with electronic components made of semiconductors like silicon and metals like copper or aluminum. Since a computer can only operate within such physical constraints, it must consider hardware, fast and power-saving circuits, technology to pack as many components as possible into a small space, easy-to-control configurations (good compatibility with software), and suitability for mass production.

Therefore, “thinking like a computer scientist” includes this hardware part, and that’s why the thinking method of computer scientists can be applied to our daily life surrounded by physical constraints, and that’s where the value of computational thinking lies. Computational thinking is not just about programming.

Of course, there are commonalities between computational thinking and programming. To create a good program, you have to think about various things. Efficiency (fast operation, less memory required) and readability (easy to understand, easy to maintain) are important, and in particular, efficiency is closely related to computational thinking.

What you need to be careful about when programming is the phenomenon of exponential explosion mentioned in Sect. 1.3. When you are writing a program, have you ever thought that you should let the computer do simple repetitive tasks because that’s what computers are good at? I think it’s a common occurrence in programming. This whisper of the heart is basically correct, but if you let the computer do simple repetitive tasks without thinking deeply, you will fall into the phenomenon of exponential explosion, and no matter how fast the computer is, it will take so long to finish the computation that you will lose your motivation. Thus, when writing a program that lets the computer do simple repetitive tasks, think carefully to avoid exponential explosion. Also in daily life, if you are aware of actions or situations that could lead to an exponential explosion, you have to avoid inefficiency and irrationality.

Column 5 “Computer” Once Meant a Person Who Computes

The word “computer” originally referred not only to machines but also to people who performed calculations. Renaissance astronomers such as Johannes Kepler relied heavily on numerical work to calculate planetary positions. At that time, assistants known as “computers,” meaning people who carried out calculations, helped perform these tasks.

Until the advent of commercial electronic computers, the term “computer” referred to people who performed mathematical calculations. From the late nineteenth century, particularly in the United States, human computers worked on calculations for astronomy, meteorology, and ballistics. They organized into teams ranging from a few members to several hundred, dividing tasks to carry out large and complex calculations in parallel. During World War II, as many male researchers were drafted, the occupation of “computer” became predominantly female. In Japan, female students also played key roles in computing altitude and azimuth tables used for aerial navigation of the Ginga bomber.

These computers used mechanical and electromechanical calculators, and later early electronic computers, to support their work. As fully electronic computers became available, many of these human computers transitioned into the emerging profession of computer programmer. Notably, the first six programmers of ENIAC, the Electronic Numerical Integrator and Computer, were women who had been recruited from ballistic computing teams.

Since the late twentieth century, individuals with extraordinary mental-arithmetic skills have sometimes been called human computers. Over time, the meanings of compute, computation, and computer have evolved.

This evolution, from computer as a human occupation to computer as a machine, reflects how advances in technology have continually reshaped both language and society.

Column 6 Women's Pioneering Role in NASA's Computing History

The Calculating Stars, a celebrated science fiction novel, received the Hugo, Nebula, and Locus Awards. Set in an alternate history where a massive meteorite strikes Earth and triggers a space colonization program, the novel accelerates the space program's timeline. It follows a woman working as a 'computer girl' in NASA's computing division, tracing her remarkable journey from performing orbital calculations to becoming an astronaut who travels to the Moon. In this context, 'computer girls' refers to women who carried out critical trajectory and orbital computations. The story unfolds in the 1950s, before the real U.S. space program, depicting a time when humans, not machines, were the primary 'computers.'

The novel's themes resonate strongly with those in the film *Hidden Figures*, which is based on a true story. This film recounts the challenges and triumphs of three exceptional African American women at NASA during the segregation era. As human computers, they played vital roles in the space race against the Soviet Union, producing groundbreaking calculations. When NASA began adopting electronic computers, one of them mastered FORTRAN and taught programming to her colleagues. She eventually led a team of thirty women, including herself, in the new programming division and was promoted to its supervisor. On a crucial launch day, when astronauts doubted the machine's calculations, her precise manual calculations confirmed the accuracy of the mission's trajectory. In 2016, NASA honored her by naming the Katherine Johnson Computational Research Facility at the Langley Research Center, commemorating a true pioneer in both space exploration and computing.

1.8 Deep Learning and Generative AI

One of the reasons for the recent attention to AI technology, which has had a great impact on society, was the announcement of ChatGPT 3.5 by OpenAI in November 2022. Since the Japanese edition of this book was published in January 2022, it did not include descriptions of ChatGPT, generative AI, and large-scale language models (LLM). We would like to discuss this topic in the English edition of the book.

Computational thinking in this book means thinking like a computer scientist or applying methodologies for generating and analyzing information by computer to everyday life in general. Efficiency at the symbolic processing level or information processing literacy is at its root. In contrast, deep learning is the process of collecting a large amount of training data or examples to train a huge neural network. In the Transformer framework for generative AI, optimization of the functions with hyper-multi-dimensional vectors as input and output is the underlying principle.

Thus, whereas traditional information processing is dominated by methodologies in which humans write down their knowledge and experience as rules and algorithms in order to achieve the desired function, deep learning and generative AI are now able to achieve the same or more complex functions through learning with a large amount of data. In other words, the cost and

accuracy of collecting and learning a large amount of data has come to exceed the cost and accuracy of describing knowledge as rules, due to the spread of the Internet, advances in machine learning technologies, and improvements in hardware performance.

How will computational thinking change and evolve in the current and future era of deep learning and generative AI?

First, since deep learning makes inference from some examples, it is impossible in principle to learn and infer 100% correctly. It is known that two types of improper learning can occur (see Column 7). One is called overfitting, in which the generalization ability becomes low due to over-adaptation to the given data (fitting too restricted examples), resulting in misjudgment even though the data is correct. The other is the opposite of overfitting, and is called overgeneralization, in which the system generalizes too much from the given data (generalizes beyond the range of correct answers), and even incorrect data are accepted as correct. In reality, learning is performed with a mixture of overfitting and overgeneralization.

By intentionally creating data that falls within the small difference between the range that a neural network that has completed training judges to be correct and the true correct answer, it is possible to deceive the network. In fact, many examples have been reported of slightly modifying an image of a school bus to make it misjudged as a camel or adding noise to a road sign to make it misidentified as a different sign.

In addition, neural networks are subject to the biases of the training data used for learning. For example, there is gender bias in that pilots are male, and cabin attendants are female, or medical doctors are male. There are also statements that are statistically correct but should not be expressed humanely, including racial bias in certain professions. In order to deal with such principles related to ethics, morality, justice, and religion without offending public order and morality, it is necessary to properly treat meta-level evaluations and reasoning such as values and ethics. It would be difficult just to describe and learn them as a language, but there are many implicit parts that are difficult to describe as a language in the first place, and it would be extremely difficult to capture them in machine learning.

This is why it is necessary to collect a large amount of good-quality training data and examples. However, at the data collection stage, it is not clear how the data will be used and processed for learning and inference. For a computer, it is sufficient to collect as much data as is relevant to its purpose, but this is not obvious beforehand. Therefore, it would be better to collect all the good data in the world in advance, but this is not possible. This can be called the frame problem in the era of deep learning and generative AI, which has not been solved so far.

Since human beings share embodiment and living environment and have learned various constraints from years of living together, this real-world experience and learning provide the criteria for deciding what range of data to consider, and the frame problem is rarely apparent (but not completely gone). Since generative AIs do not live their social lives as living people, it is difficult for them to share experiences and learnings as humans do. For example, learning about love from a romance novel is different from actually experiencing love, and understanding a dish by reading a cooking recipe is different from actually cooking it and tasting it. The same applies to the appreciation of music and paintings.

Therefore, in front of a neural network, a computer scientist who is trying to achieve the desired deep learning should pay attention to the following in order to make the neural network learn and infer as correctly as possible.

What kind of data do you want the neural network to output or infer?

What kind of data and conditions should be input for that purpose?

How and where should such data be collected?

What kind of neural network configuration should we construct?

What kind of data should be given to the neural network, and how?

Until now, computational thinking has focused on efficient handling of computational resources. Computational thinking in the era of deep learning and generative AI requires us to be aware of how to correctly handle the meaning of data. We believe that such new computational thinking will become important in the way we deal with deep learning and generative AI.

In symbolic processing, which has been the premise of computational thinking up to now, the issue of data representation (i.e., what is a symbol and what is a feature) was decided by humans in advance, so the inference process and the meaning of symbols are understandable to humans. On the other hand, deep learning can automatically acquire data representations or an embedding space consisting of super-multi-dimensional feature vectors. Therefore, while it is possible to acquire tacit knowledge that is difficult to verbalize, the inference process and the meaning of feature vectors are difficult for humans to understand and become a black box. The object of computation in traditional computer science is a symbol, and the hyper-multi-dimensional feature vectors that are the object of computation in deep learning may correspond to signals. Because of signals, to some extent, we must accept that the contents of a neural network are a black box.

Novel computational thinking may be to understand not only the strengths of deep learning and generative AI, but also its weaknesses and shortcomings, and to cope with them.

Column 7 Overfitting and Overgeneralization

Consider learning to recognize images of cats. By presenting various images, the goal is to have the system identify only the images of cats as “cats.” Suppose the target range of correct answers is within the circular range shown by horizontal lines in Fig. [1.10](#) . Since the system cannot know this correct range, it must infer (learn) from the given examples. But precise learning is difficult when relying solely on positive examples. For instance, although the provided examples (indicated by the small squares in Fig. [1.11](#)) are covered, they might be too close to the examples, resulting in low generalization (this is called overfitting). In overfitting, even correct examples might be marked incorrect. Conversely, aiming too much for generalization may exceed the range of correct answers, leading to incorrect data being judged as correct (Fig. [1.12](#)). This is called overgeneralization.



Fig. 1.10 Target: Any point within the circle is the correct instance of the category which is the target of learning. However, the correct target is not known to the system



Fig. 1.11 Overfitting: Small squares represent correct instances given to the system. If the system tries to be precise only for those instances, the result of learning is too narrowly focused on the target area

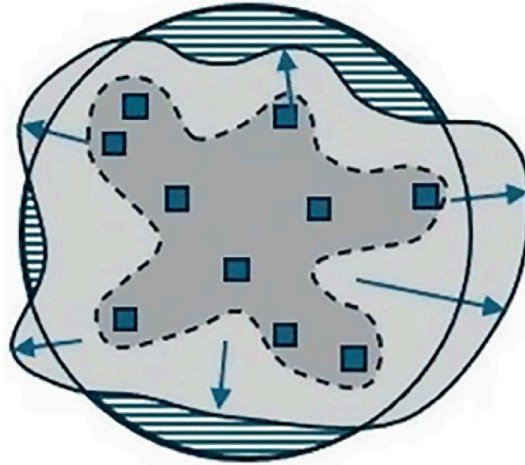


Fig. 1.12 Overgeneralization: When the system tries to generalize from the given instances too much, it leads to overgeneralization, which should be avoided

To deceive the results of deep learning, one can create examples that differ from the correct answers. Especially in areas where there is a lack of generalization (indicated by the horizontal line in Fig. 1.11), even though it is an image of a cat, it might be judged as something else. On the other hand, if one provides examples where overgeneralization has occurred (areas exceeding the correct range in Fig. 1.12), it might answer “cat” even though it is an image of something else.

There are many studies conducted to deceive the results of deep learning. Examples include making a “school bus” be misrecognized as an “ostrich”, or a “stop sign” being misidentified as a “speed limit sign.” These methods are more elaborate than the misrecognitions discussed here and will not be detailed as they exceed the scope of this book.

Footnotes

¹ When we were writing the original Japanese version of this book in 2021, this statement was true. However, ChatGPT released in late 2022 changed the scenery.

² Deep learning and generative AI do not use symbol manipulation in the sense written here; they operate using high-dimensional vectors instead.

³ The cognitive limit for the number of stable, meaningful social relationships an individual can maintain is estimated to be 150 people, known as Dunbar’s number.

⁴ To be precise, 2^{10} is 1024, but remembering that it is about 1000 is useful for various calculations. Even larger numbers such as one million and one billion can also be easily calculated in your head.

⁵ The exact probability can be calculated using permutation as $1/P(16,16)$, which is $1/20922789888000$.

[6](#) Carlo Rovelli: The Order of Time, Penguin Books (2018).

[7](#) American philosopher (1932–2025).

2. Practice of Computational Thinking

Keiji Hirata¹  and Hideyuki Nakashima¹

(¹) Mirai Share Co., Ltd. and Future University Hakodate, Hakodate, Hokkaido, Japan

 **Keiji Hirata**

Email: hirata@miraishare.jp

URL: <https://www.miraishare.co.jp/>

Abstract

Applying computational thinking in daily life and work means understanding the things and events around readers by using the way computers work as an analogy and engaging with them actively. As a result, readers gain an incredibly clear understanding of these things and events and can compute very efficiently. To achieve this, four essential concepts are indispensable. This chapter introduces three of them: abstraction, modeling, and virtualization (the remaining one will be introduced in the next Chap. 3). Computational thinking encompasses not only arithmetic operations like addition and multiplication, but also the broader process of generating new information from existing information. These three (four) key concepts can be seen as the fundamental operations for creating new information. Some may not be familiar with the terms abstraction, modeling, and virtualization in everyday conversation, but here are examples showing how these concepts function right around us. At the end of this chapter, we introduce the field of information design to reaffirm the characteristics of computational thinking. Readers will see that the goal of information design—to make the representation and communication of information and knowledge more flexible and richer—is closely connected to computational thinking as a methodology for generating new information.

2.1 What It Means to Be a Computer Scientist

I would like to explain the thought required to practice computational thinking in everyday life.

Computational thinking means thinking like a computer scientist. However, this does not mean being familiar with computer hardware, software, and systems.

Computer scientists, who have a deep and broad understanding of computers and feel familiar with them, often anthropomorphize the mechanisms and operations of computers. For example, they may say there are relationships called parent processes and child processes, and a program is ‘running’. It is even common to say things like ‘killing’ a process. Conversely, computer scientists express the state of the real-world using computer terms. Saying, for example, “this recipe should be refactored”, and “we want to give this whole business a reboot”, etc.

Thinking like a computer scientist includes creating metaphors for real-world things and events using the mechanisms and operations of computers. There is an English proverb,

If all you have is a hammer, everything looks like a nail.

This means that if you only have one means, you will try to solve all problems with that one means, which may be inefficient and ineffective. Computer scientists, in a good sense, try to understand everything in the real world by metaphors using the mechanisms and operations of computers.

What I want you to note here is the difference between a hammer and a computer. In common sense, a hammer only has the function of hitting a nail, but a computer is versatile. In principle, a computer can perform any calculation and realize any function. Therefore, computer scientists, who have a deep understanding of the mechanisms and operations of computers, can create a metaphor for any real-world things and events using the appropriate mechanisms and operations of computers without compromising their essential properties and relationships.

2.2 Abstraction

2.2.1 Meaning of Abstraction

What does it mean to create a metaphor for something using the mechanisms and operations of a computer? Here are three concepts that become important. They are abstraction, modeling, and virtualization, all of which involve applying something to the original thing and creating something new.

By properly abstracting, modeling, and virtualizing real-world things and events, you can correctly understand and imitate (simulate) the real world through symbols on a computer. Here, the symbols on a computer refer to variables and numbers that appear in a program, or their arrangements (arrays or structures), but they are always required to correctly correspond to real-world things and events throughout the execution of the program.

To accurately forecast the weather, you need to accurately observe the temperature, humidity, atmospheric pressure, wind speed, etc. in various places on the earth, and throw them into a huge set of differential equations. Prior to this, there is the task of expressing the physical phenomenon of weather as a differential equation with parameters (or intermediate variables). In the real world, there are a wide variety of physical parameters, as well as factors such as sounds heard on the spot (soundscapes), the body temperature of the people living in the relevant area, the number of dogs kept, the proportion of people who like Mozart, etc., but common sense tells us that these pieces of information do not have a significant impact on the weather, so it is safe to assume that there is no need to incorporate them as parameters in the differential equation for weather forecasting. Thus, meteorologists have simply selected those parameters that are most heavily involved in the weather, such as temperature, humidity, atmospheric pressure, and wind speed.

According to the sixth edition of 'Kojien', one popular Japanese dictionary, abstraction is defined as 'a mental action that grasps a certain aspect or property of a thing or representation apart from reality'. In the above example, abstraction refers to the process of selecting parameters that are considered important, by setting a certain perspective, and selecting parameters that are thought to have a large effect on the phenomenon being observed. In other words, abstraction is defined as focusing on important information and ignoring complex details. When creating a differential equation, we simply ignore parameters that are not important or that are unrelated to the phenomenon being observed. This is called omission.

Furthermore, the word abstraction is used in a sense other than selection of parameters. That is, when parameters that are not independent of each other are found, they are combined into one to reduce the number of parameters. For example, consider the course of a marathon race. The

distance from the start to the goal is fixed at 42.195 km, but the shape and elevation of the course vary from city to city. As a parameter to represent where the runner is running on the course, a point in a three-dimensional Cartesian coordinate space can be used. In other words, we need three variables, X, Y, and Z, to which values representing the runner's location are assigned. However, the course the runner runs is a single path, and the runner cannot freely exist at any points in the three-dimensional Cartesian coordinate space. That is, parameters X, Y, and Z are not independent of each other, and there are certain constraints on them. Considering these constraints, the runner's position can be uniquely determined by giving only one variable: how far from the start the runner is on a single path. As a result, the runner's position has been abstracted as a single parameter, how many kilometers from the start. Information about the shape and elevation of the course, which varies from city to city, has been discarded.

Column 8 Definition of “Abstraction” in a Dictionary

‘Super Daijirin’, a commonly used Japanese dictionary, defines abstraction as follows: The act of focusing on certain properties, commonalities, or essences of things or representations, and understanding them by extracting them. This process also involves the action of excluding other unnecessary properties (= omission), so abstraction and omission form two aspects of the same action.

Example: “what appears in the meaning or judgment is a part abstracted from the original experience.”

Opposite: concrete. See also ‘omission’.

2.2.2 Difficulty of Appropriate Abstraction

Regarding the parameters selected when abstracted in this way, the physical variable of temperature, for example, has a strictly consistent definition and is implemented as a corresponding variable in the program. As long as the calculation is carried out according to the differential equation, the variable corresponding to the temperature always correctly represents the temperature throughout the execution of the program.

To make a more accurate weather forecast, it might be effective to consider the effects of clouds, aerosols, cosmic rays, terrain, buildings, trees, and traffic volume. However, how should we express information about clouds and terrain in differential equations? It does not seem possible to simply represent them with real number variables in the same way as temperature and humidity. Even if we express information about clouds with specific terminology like “cumulonimbus” or “streaky clouds”, we cannot incorporate it well into differential equations or forecast calculations. We would probably have to use vectors and matrices and set complex boundary conditions. However, it is generally very difficult to give a physically strictly consistent definition to the parameters representing the clouds and implement them as arrays or structures in the corresponding program.

2.2.3 Good Thing About Abstraction

So, what is the benefit of the abstraction that computer scientists are proficient in? It is that when you abstract, different things look the same. This ability to see the same is inevitable to create a metaphor for real-world things and events using the mechanisms and operations of computers. For example, if you add 8 GB of memory to 4 GB of memory, the total is 12 GB. If you load 4 kg of

rice and 8 kg of water onto a truck, the total is 12 kg. If you buy a \$400 ring and an \$800 necklace, the total is \$1,200. If 8 people get on a bus with 4 passengers, the total is 12 people. All these real-world calculations are achieved by the same process, which involves first abstracting the relevant factors into the numbers 4 and 8, calculating using the addition formula $4 + 8 = 12$, and then returning the value 12 to the real world again (called concretization or instantiation). By selecting only the property of value, that is, setting the perspective of quantity and focusing only on that value, various phenomena in different domains such as memory expansion, total weight, total amount, and total number of passengers appear the same.

2.3 Modeling

2.3.1 Meaning of Modeling

By abstracting things and events in the real world, what we get next is a model. We call the process of obtaining a model from the real world ‘modeling’. Normally, when we say “model”, many people likely think of human models appearing in fashion shows, but, in this book, let us assign to the word model the meaning of a substitute for the real thing with respect to functions and features, which works for understanding and explanation in a broad sense.

Even if we assign such a meaning to the word model, the concept may be still unclear, so let us consider Newtonian physics, one of the most successful models in human history, as an example. Long before Newtonian physics, around 350 BC, Aristotle claimed that all matter in the world was made up of four elements: fire, air, water, and earth. In China, it was thought that the world was made up of fire, water, wood, gold, and earth, according to the theory of five elements. However, these theories could not explain certain mysteries, such as why a small iron ball and a large iron ball and a feather all fall at different speeds, and why the moon does not fall to the ground. Then, in the 1660s, Newton discovered (or “extracted”) the concepts of mass and force. Until then, people had tried to understand the world through visible things, but Newton tried to understand the world with essential concepts that are invisible. Newton proposed the famous formula $F = ma$ (F is force, m is mass, and a is acceleration). Furthermore, he also discovered (or “invented”) the concept of calculus to make it possible not just to understand the relationship between objects, but also to predict movement. This calculus can be seen as a technology to predict the temporal change of force, mass, and acceleration using a model.

2.3.2 Three Types of Models

The models actually used in the world are broadly classified into three types: mathematical (or representational) model, numerical model, and concrete model. A mathematical model is one that expresses the behavior or state of the target object in mathematical formulas, such as Newtonian physics (classical mechanics) and linear regression used in data analysis.

A numerical model expresses the behavior or state of the target object with an algorithm. Describing by an algorithm means using computational means such as sequential execution, iteration, conditional branching, recursive call, probabilistic state transition, and multi-agency. An earth simulator and cellular automaton are examples of this.

A concrete model is one that substitutes the target object with another physical object. For example, model organisms like guinea pigs; miniature airplanes or rockets used in wind tunnel experiments to test aerodynamic performance; and celestial globes that faithfully reproduce the movements of the sun, moon, and stars visible in the sky (such as Ptolemy's famous celestial

globe). The human models that appear in fashion shows can also be considered concrete models, as they demonstrate and simulate what it would be like for someone to wear certain clothes.

The purpose of obtaining and considering models is to understand, explain, and predict the phenomena of the real world from a certain perspective, which is something that humans have an essential desire to do. In Section 1.7.2, we stated that “(the reason computational thinking is important is because) computation creates new information based on specific procedures.” Prediction is a type of information generation, and it is an important activity for humans. However, it is generally very difficult to invent a model for generating new information or making predictions. For example, compare the task of judging which of two dishes is more delicious with the task of actually cooking a dish that is more delicious than another. It is immediately clear that the latter task is harder, as it is much more difficult to actually cook food than to judge it. The same is true for earthquakes and economics, as it is well known that it is much more difficult to predict what will happen than to analyze what has happened. Thus, practicing computational thinking and generating new information is not an easy task.

Then, let us think a little more about prediction in mathematical models. First, by abstracting things in the real world, we select parameters, and relationships between parameters, that represent the state and changes we are focusing on. In the case of Newtonian physics, the parameters are force, mass, and acceleration, and the relationship between parameters is $F = ma$. The whole of these parameters and relationships is called a structure. Since the real-world changes over time, or in response to certain action, it is also necessary to consider the operations that change the corresponding structure. In Newtonian physics, computing calculus corresponds to performing the operations. The structure and the operations that alter the structure are combined to form a predictive model. With the use of calculus, you can predict how an object changes over time. The correspondence between the structure and operations included in a model and the elements and changes in the real world is called interpretation. Thus, a mathematical model is also said to be an interpreted mathematical structure.

2.3.3 From Modeling to Science

By improving a model, it can be made to function more appropriately, enabling humans to explain the phenomena of the real world more skillfully. This joy of being able to explain invokes a kind of aesthetic sense, and it is said that this led to the discovery of science in 16th century Europe. To facilitate the joy of being able to explain, the following three methodologies, each containing a “not” have been adopted among scientists.

- Not being concerned with the purpose (though science may originally have been used for agriculture or war, the ability to explain itself became the purpose),
- Not pursuing preciseness (the simplicity of the model was prioritized even if the prediction was slightly inaccurate), and
- Not caring that the conditions set are artificial (e.g., experiments on dropping iron balls in a vacuum, experiments using miniatures, or introduction of mass points).

By developing such methodologies, science has been proposing better models, that is, more concise models that can provide a more thorough explanation. Gradually, as the concept of science was formed, a certain consensus was also formed about the attitude of scientists in pursuing science. Definitions regarding the scientific attitude were given from two perspectives, empirical and procedural. Empirically, it was thought that what is beautiful is scientifically correct; a correct

model does not have to be accurate, and an artificial one might be the correct model. Procedurally, the following were considered important.

- Generalizations beyond human subjectivity are made,
- Evidence (rationale) to demonstrate these generalizations is always prepared,
- Reproducibility is guaranteed, and
- There is refutability against generalizations that lead to saying things are true that are not true.

It is clear that these scientific attitudes underline modern science. The effort to produce better models has had an immeasurable impact on the development of science.

Now that you have understood what a model is and the benefits of thinking about models, let us return to the discussion of computational thinking. Modeling is included in the act of behaving like a computer scientist creating a metaphor for the real world using the mechanisms and operations of a computer. Moreover, the models that generate new information or make predictions are useful to humans.

The major premise of physics is that there is only one correct standard model in this world (or at least that there should be fewer models than there are scientists), but computer scientists consider that scientists, when observing the real world, create models from different perspectives, so there can be as many *correct* models as there are scientists. Undoubtedly, computer science is a rare discipline in which multiple, possibly conflicting models may be allowed. It is the versatility of computers that allows for flexible proper modeling. Therefore, when people model various things and events in the real world from their own perspectives, it is computer science that can provide useful insights, and this is computational thinking.

2.4 Virtualization

2.4.1 Meaning of Virtualization

Finally, let us discuss another important concept, virtualization. In Section 1.2.3, we mentioned that the second condition of good computation is to find a more efficient or effective solution to the problem represented by the model after bringing the things and events of the real world into the world of the computational model. This solution is called a procedure or algorithm, and the explicit representation of the relationship between data is called a data structure. Rather than processing related elements separately, it is desirable to group them into a vector or matrix and process them all at once to achieve a readable program or algorithm for easy improvement.

Thanks to long-term research on algorithms and data structures by computer scientists, people can easily access a generic catalog for excellent solutions. If you can model a problem of the real world in such a way that the cataloged algorithms and data structures can be applied smoothly, you can easily obtain lean processing. This is a good computation, and good modeling is the key to good computation. Then, the key to good modeling is virtualization, which is the topic of this section.

2.4.2 Does Virtual in Virtual Reality Really Mean Virtual?

First, we have to clarify the meaning of the word “virtual” because it is used in two major senses. Here, let us ask ChatGPT what “virtual” means. The answer is as follows:

“Virtual” generally refers to something that exists or occurs in a digital or simulated environment, rather than in the physical world. It can also denote the imitation or

representation of something, often created or experienced through electronic or computer technology.

This answer is ambiguous as to whether the original object being simulated or imitated is real or not. In other words, virtual has two major meanings. One is to extract or focus only on the essential or substantial parts of something that already exists in the world, and the other is to make something appear as if it actually exists, including things that do not exist in the world. Thus, the latter is more broadly defined and includes the nuances of fictitious and hypothetical. The former does not include such nuances and only looks at things based on reality.

The term “virtual reality” is sometimes used in the mass media to mean making you perceive and feel as if the fictitious space created by a computer is real, bringing experiences that are impossible in reality. In contrast, the “virtual reality” commonly used in the academic community is defined, for instance, as follows:

A technology or a general term for a system that creates the essential functions (the essence of reality) of events and phenomena in the real world, even if their appearance and form are quite different from those in the real world, by presenting information generated by a computer to various human senses.

Incidentally, the Japanese Society for Artificial Intelligence suggests that “artificial reality” might be a less misleading term than virtual reality.

Hereafter, the word “virtual” in this book will be used in the sense of virtualization as the reproduction of a situation that is essentially the same in function as things and events in the real world. However, simply reproducing a situation does not benefit anyone, so it is reproduced in a way that is easy for users to handle. In focusing only on logical existence and objects, by removing the physical structure and constraints of the real world, virtualization means the same as abstraction. However, the key point of virtualization is to “reproduce in a way that is easy for users to handle.” In reproduction, it absorbs the minor physical differences that realize its function and state and provides only the substantial function. This leads to “ease of use for people.”

Column 9 the Virtual President

The English word “virtual” can mean “in effect, if not in name.” It is said that Bill Clinton, the 42nd President of the United States, once said “*Hillary is the virtual President.*” This means that Hillary is the president in effect, and it may be implied that Bill is the nominal President. Hillary is the wife of former President Clinton, and she is said to be a more capable politician than her husband. As you all know, she later ran for president against Mr. Trump and was narrowly defeated.

2.4.3 Virtual Memory

The quickest way to understand virtualization is to look at a concrete example. There is a technique called virtual memory that virtualizes memory (Fig. [2.1](#)). Physical memory is realized by semiconductors and has a fixed capacity such as 512 KB or 1 GB, but the logical address space expressed in the program is 32 or 64 bits wide, the memory space of which is 16 GB or 16 EB, respectively. It is difficult, both in terms of cost and speed, to allocate real memory for the entire logical address space. Accordingly, a technique has been developed that realizes an ideal memory, in which physical memory exists throughout the entire logical address space when viewed from a

program, despite using only a small amount of physical memory that covers only a fraction of the logical address space in reality. This is a virtual memory. Specifically, during the execution of a program, the requested logical addresses are sequentially associated with the physical addresses where the actual data exists. Logical addresses that are no longer used are automatically disassociated with their physical addresses. The substantial function that was focused on here is the memory address. From the program's perspective, it can freely be used anywhere in the logical address space, and it can dispose of it without being aware of the reuse of the memory area. This alone makes it “easy for the user (in this case, the programmer) to handle”, yet what is even more amazing about virtual memory is that multiple programs (processes) can use the same logical address at the same time without any problems. In other words, you can write a program without worrying about which part of the logical address other processes are using. Virtualizing the physical memory address and presenting it to the programmer as a logical address makes memory extremely easy to handle.

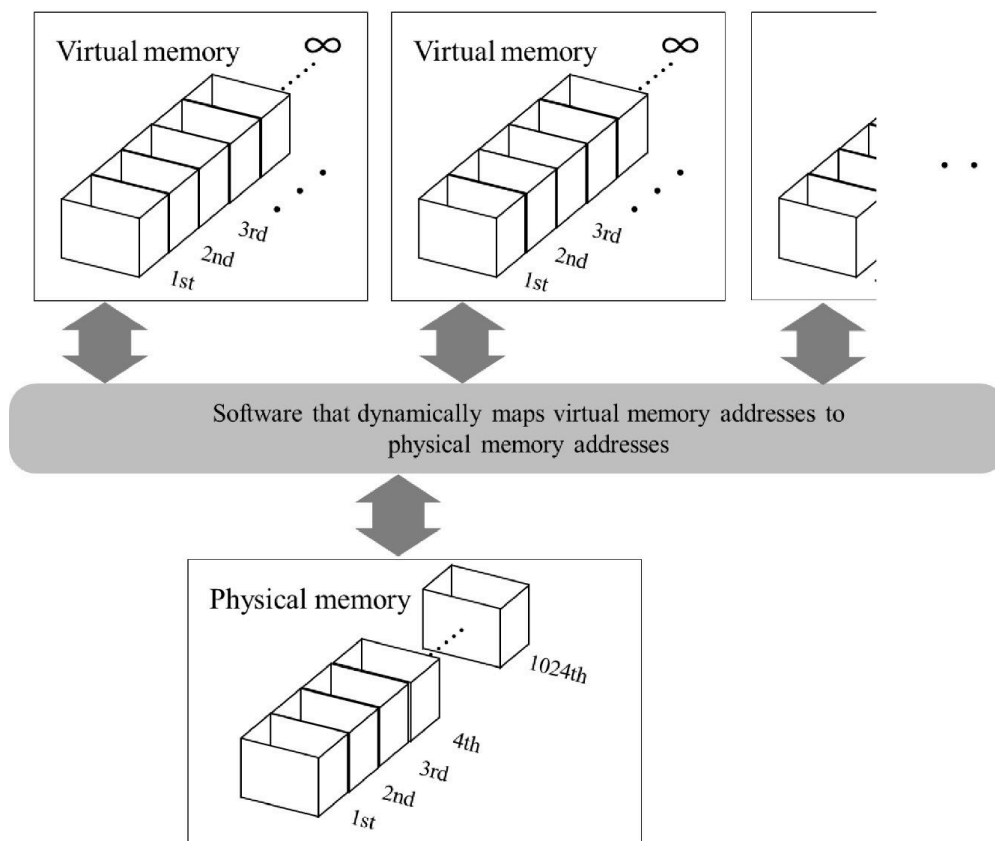


Fig. 2.1 Mechanism of virtual memory

2.4.4 Virtual Machine

Another example is the technique of virtualizing the CPU, which is called a virtual machine (Fig. [2.2](#)). There are companies like Intel and AMD that manufacture and sell physical CPUs, and they have a variety of products lined up. For embedded use, Raspberry Pi and Arduino are popular. These different CPUs operate in different machine languages and have different functions, so if you want to implement a Python language processing system (compiler, runtime system, etc.), an OS or applications that work the same way on any CPU, you have to create a program in a different

machine language for each different CPU. This is a very labor-intensive task. Therefore, it is preferable to set up only one logical CPU and machine language. That is the virtual machine. The virtual machine is implemented in the machine language of different real CPUs, and the Python language processing system, an OS and applications are implemented in the machine language of the virtual machine. The machine language of the virtual machine is also called the intermediate language. While different CPUs may be used, the components running under the virtual machine's machine language can be commonly used. The substantial functions that are focused on here are the functions and machine language of the CPU. This alone makes it “easy for the user to handle”, but what is even more amazing about virtual machines is that multiple independent virtual machines can be created on a single physical computer (CPU). In other words, if you prepare a CPU with powerful computing power and launch multiple virtual machines on it, the computing power, memory capacity, and communication capabilities of that CPU can be shared among multiple virtual machines, resulting in a flexible computer environment with little waste. The provision of these virtual machines to users over a network is what is known as the cloud.

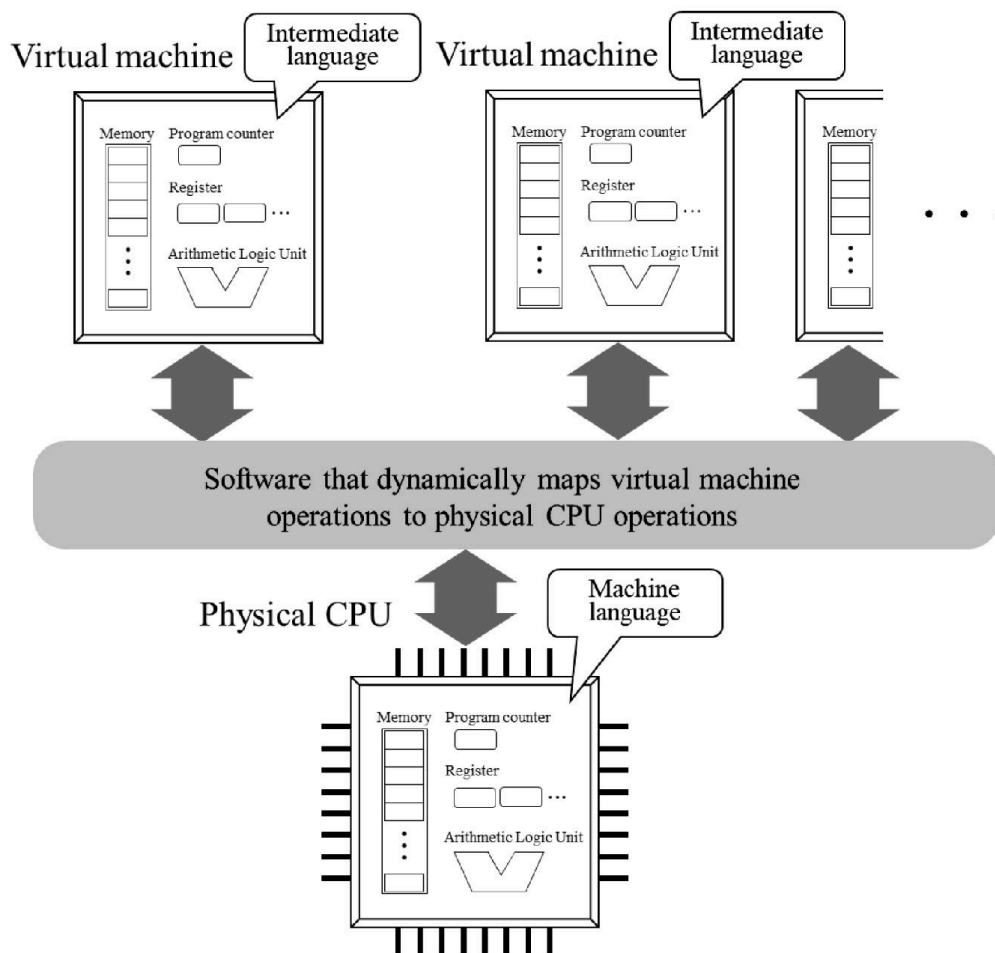


Fig. 2.2 Mechanism of virtual machine

Thus, the direct benefits of virtualization are the ability to flexibly divide, copy, integrate, and transform computational resources of a real computer. Generally, resources mean things that are necessary for activities and that decrease when consumed. Computational resources refer to the resources necessary for executing and realizing computations, specifically, time, memory space,

communication lines, etc. By virtualizing computational resources, users can adjust the quality and quantity of computational resources at their disposal and freely process them (adjust them for their own purposes).

2.4.5 Various Names for Virtualization

In the various fields and application areas of computer science, the concepts equivalent to virtualization are called by different names. First, in the field of information systems, a group of services further developed from virtual machines is called the cloud (Fig. 2.3). In addition to virtual machines, services that provide users with the functions of specific software or applications are called SaaS (Software as a Service). Furthermore, services that provide users with not only applications but also their development environment are called PaaS (Platform as a Service). And services that virtualize the entire computational environment, including memory and network, and present it to users as infrastructure, are called IaaS (Infrastructure as a Service). In the field of networks, the extraction and commonalization of only the substantial functions of the equipment or processes to be controlled are called standardization and normalization. Standardization and normalization improve interoperability. In the field of software engineering, appropriately dividing a program into independent parts, focusing only on its functions and interfaces, and abstracting (hiding) its implementation is called modularization, componentization, object orientation, etc., which correspond to the virtualization of programs. The virtualization of programs both enhances the reusability of the programs and contributes significantly to software design theory, by which large-scale software is accurately and efficiently created by a small number of people.

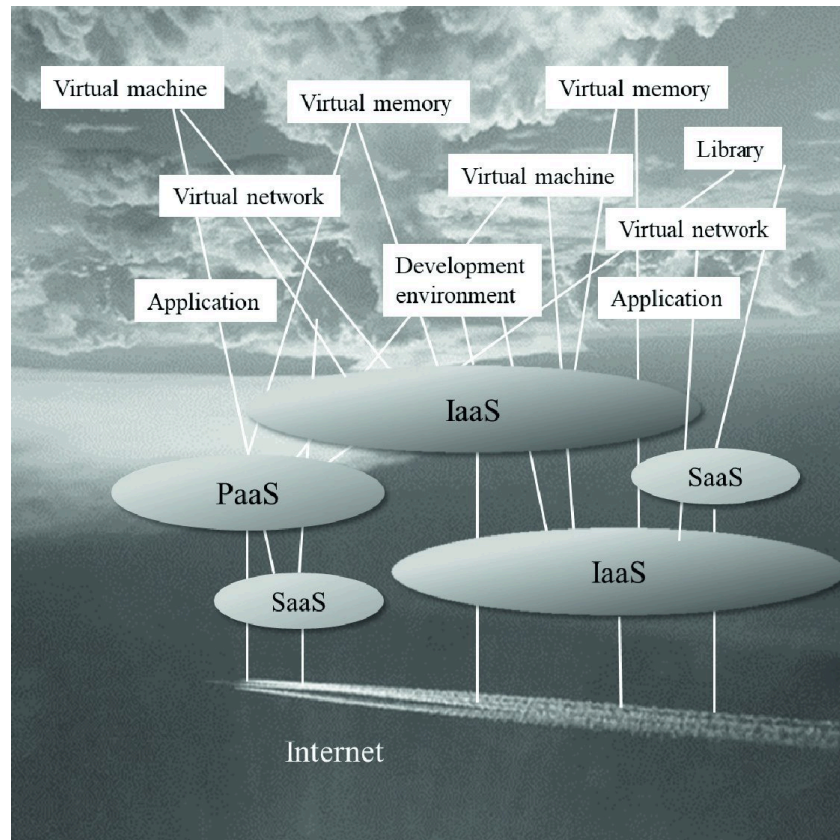


Fig. 2.3 Cloud = Virtualization of entire computing environment

2.4.6 Other Virtualizations

Not only in the world of computer science, but also in the real world, you will be able to find many examples of virtualization. In the past, physical entities and the functions, services, and contents provided there were inseparably linked, but by virtualizing them, we produce the trend of separating functions, services, and contents from physical entities and providing users with more essential things, a trend which is accelerating nowadays. Virtualized functions, services, and contents are freely divided, copied, integrated, and transformed according to the purpose and environment on the provider and/or recipient sides, and are used more appropriately and conveniently. For example, when listening to music, in the past, the physical entity (a vinyl record) and the content (music) were inseparably bound together, and even the playback method was determined and limited. The advent of CDs enabled music to be handled as digital data, freeing it from the constraints of sound quality and the limited playback methods of vinyl records. After that, the sound of music has become digital data, independent of the physical form of a CD, allowing users to download it via the Internet without having to acquire a physical medium. Now, with streaming, only the digital data necessary for playback is transferred on demand.

Human activities are also being virtualized. In society, humans can be seen as integrated entities possessing knowledge, experience, and skills, and digital technology has been separating these attributes from the physical existence of humans. For example, when reading an email, you can understand the sender's message, but you usually do not know where the sender was when they wrote the email. There is also the gig economy, a form of labor market that separates and extracts specific abilities of a human, integrates them, and serves them in a cloud-like manner, which can be said to be a virtualization of human activity.

The wave of virtualization is also affecting the way corporate organizations operate. Companies have emerged that provide an outsourcing service, taking on only the common general affairs and personnel tasks of other companies (for example, ADP (Automatic Data Processing) and Insperity). This can be considered a virtualization of corporate activities. Another example is in the field of design, in which there is a focus on UX (User Experience). Companies and designers have emerged who specialize in thinking about how design can produce a more comfortable and richer experience for users through the use of web services and establishing a comprehensive and extended design methodology. This can be considered a type of virtualization that separates the design activity from the context and standardizes it, while traditional design was often inseparable from context.

Furthermore, virtualization is occurring in the world of transportation, such as taxis, city buses, and shuttle buses. The mobility services provided by traditional taxis, city buses, and shuttle buses were bound by parameters such as route limitations, timetable limitations, and the possibility of sharing, which were tightly restricted by law. In addition, these mobility services were inseparable from their hardware (vehicles, stops, operation/reservation systems, etc.), and as a result, offered very little freedom. If these mobility services were freed from the constraints of the law and separated from the hardware by virtualization, it would be possible to consider a generic mobility service where each parameter can be freely adjusted (Fig. 2.4). As mentioned earlier, the benefits of virtualization include the ability to flexibly divide, copy, integrate, and transform resources. If these general mobility services can be divided, copied, integrated, and transformed, they could provide optimal mobility services according to passenger demand. This

corresponds to the concept of MaaS (Mobility as a Service), which has become increasingly common in recent years. It can also be rephrased as the cloudification of mobility services.

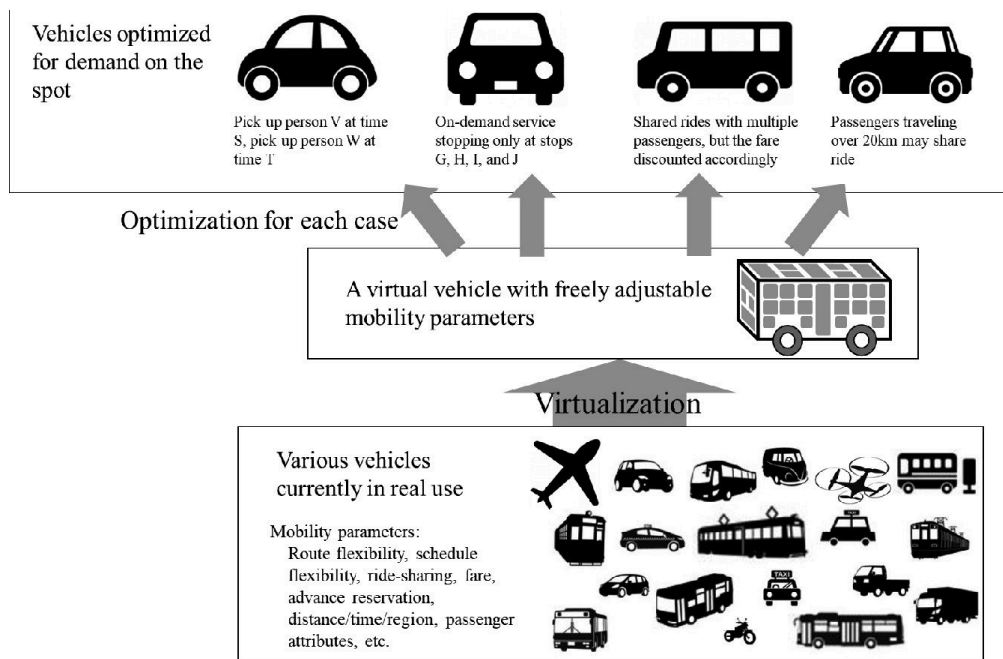


Fig. 2.4 MaaS as virtualization of entire transportation services

The final example of virtualization is the 3D printer, which has revolutionized manufacturing. Traditional manufacturing required tools, materials, blueprints, and a maker with skills, knowledge, and experience, all of which needed to be gathered in one place at the same time. However, the advent of 3D printers has dramatically relaxed these constraints. As a result, anyone can now make certain things that were previously only made by craftsmen or experts. Manufacturing skills used to be scarce, but it will no longer be so. Traditionally, goods had to be mass-produced and consumed in large quantities due to cost and time constraints. However, it is now possible to produce a small variety of goods and consume locally. Dr. Hiroshi Ishii, the associate director of the MIT (Massachusetts Institute of Technology) Media Lab, has proposed the concept of “tangible bits and radical atoms”, and the 3D printer could be seen as a universal machine that realizes this concept.

Please keep in mind the concepts of abstraction, modeling, and virtualization that we have explained so far, and try to create metaphors for the things and events in the real world using the mechanisms and operations of computers, as computer scientists do. No doubt you will find that there are many opportunities to enjoy practicing computational thinking in everyday life.

2.5 Design

Computational thinking is a way of thinking about how to create new information. The “design” we often hear about is also an activity and field that creates new things, mechanisms, and experiences. So, what is the relationship between design and computational thinking?

Figure 2.5 shows the relationships between various academic and practical fields. The important point here is that computer science and design are orthogonal to other fields. Computer science is now the foundation of all fields. Bioinformatics is a typical example, which is a field that

studies biology from the perspective of informatics. Since organisms are made from genes, and these genes hold the blueprint (information) of the body, it is quite natural that these two fields should be combined.

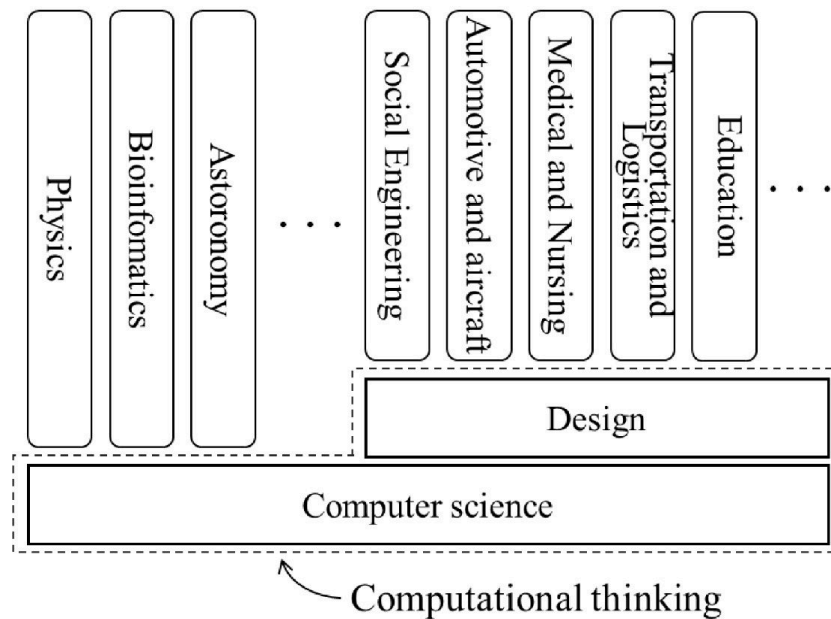


Fig. 2.5 Relationship between computational thinking and design

In astronomy, recently, non-optical telescopes such as radio telescopes are used to observe distant celestial bodies. The “photo” of the black hole that made the news in 2019 was taken by the Event Horizon Telescope, which combines the data from eight radio telescopes on the earth and processes it with computers. This is representative of how the data analysis of certain experiments, though the experiments may be physical, cannot be conceived without computers.

Computer software can basically program anything you can think of. However, when applied to fields such as transportation and healthcare, it is necessary to carefully consider human preferences and requirements. That's where design comes in, introducing values of what is desirable and creating mechanisms that are appropriate for it. Design is the foundation of all engineering. We believe that computational thinking lies in the combination of insights from both computer science and design.

Editors

Hideyuki Nakashima

Sapporo City University, Sapporo, Hokkaido, Japan

Keiji Hirata

Department of Complex and Intelligent Systems, Future University Hakodate, Hakodate, Hokkaido, Japan

ISBN 978-981-95-5961-9 e-ISBN 978-981-95-5962-6

<https://doi.org/10.1007/978-981-95-5962-6>

Translation from the Japanese language edition: “計算論的思考ってなに？” by Hideyuki Nakashima and Keiji Hirata, © Authors 2022. Published by Future University Hakodate Press & Kindaikagakusha Publisher. All Rights Reserved.

The original submitted manuscript has been translated into English. The translation was done using artificial intelligence. A subsequent revision was performed by the author(s) to further refine the work and to ensure that the translation is appropriate concerning content and scientific correctness. It may, however, read stylistically different from a conventional translation.

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2026

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd. The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

When I first read Jeannette Wing’s influential essay “Computational Thinking” published in 2006, I immediately recognized its importance. I wholeheartedly agreed with her assertion that computational thinking (CT) should be a fundamental component of university education for all disciplines, and that students majoring in computer science would be equipped to excel in a wide range of careers beyond their technical training. I translated Wing’s essay into Japanese, and it was published in the Journal of the Information Processing Society of Japan in 2015.

However, when Prof. Michael Vallance, our colleague at Future University Hakodate (hereafter referred to as FUN), used my translation as the teaching material, he found that students struggled to understand it. Wing’s essay includes many technical terms from computer science but offers few concrete examples—particularly ones that demonstrate how such concepts apply to other fields. While the essay was written for educators, it is not structured to help beginners grasp the essence of computational thinking.

This observation led us at FUN to launch a project aimed at publishing a book filled with accessible and practical examples of CT. It was not easy to find examples that made sense to people outside of computer science. We each tried various approaches and held frequent discussions. Eventually, we compiled our findings into the desired book in Japanese, which translates Wing’s abstract ideas into concrete terms that are easier to understand for non-specialists.

At the same time, international interest in teaching computational thinking was gaining momentum, with new academic societies and initiatives being launched. In Japan, however, government agencies such as the Ministry of Education had interpreted CT to mean programming and began to incorporate it into elementary school curricula. Even national media outlets like NHK produced TV programs based on this interpretation. Needless to say, programming knowledge and skills, along with the ability to effectively utilize modern AI technology, are certainly part of CT, but CT encompasses much more.

This English edition is based on the Japanese edition published by FUN Press in 2022. Chapters 1, 2, 4 and 5 of that Japanese edition have been revised to a diverse extent. This English edition was unable to include Chap. 3, authored by Michael Vallance. For his research, please see his paper titled “Computational thinking and interdisciplinary learning: time to focus on pedagogy” published in 2020.

The structure of this book is designed to be progressive, starting with an introduction to fundamental concepts in computer science, moving through methods for understanding computational thinking, and culminating in its practical application. However, it has been consciously written so that readers can begin reading from any chapter.

This book begins in Chap. [1](#) by serving as a primer for readers unfamiliar with computer science, defining computational thinking as “thinking like a computer scientist.” The chapter explains the origins of computer science and its unique worldview by contrasting it with fields like physics and mathematics, while also addressing its relationship with programming, deep learning, and generative AI.

Chapter [2](#) delves into the practical benefits of applying computational thinking in the real world. It introduces three of four essential concepts—abstraction, modeling, and virtualization—which are presented as fundamental operations for generating new information. The chapter uses

relatable examples to make these terms accessible and concludes by connecting computational thinking to the principles of information design.

Chapter [3](#) introduces the fourth essential concept, meta, and its related idea of self-reference. Acknowledging the complexity of the topic, the chapter explains how to think about an object and concept from a higher perspective, using real world examples to clarify how to apply this thinking both in human activities and within a computer.

Chapter [4](#) is an essay in which a computer scientist and education scholar shares her personal journey, explaining how her childhood experiences cultivated a computational mindset. She uses the unique analogy of cooking to explain computational thinking, breaking down a recipe into a series of logical steps, and discusses how this perspective can be taught effectively to children.

Finally, in Chap. [5](#), a cognitive scientist with no background in computer science reports on her personal struggle to understand computational thinking. She shares her initial discomfort and the turning point that allowed her to grasp the concept, concluding with a non-specialist's unique discovery of computational thinking's altruistic power.

Hideyuki Nakashima
Sapporo, Japan
September 2025

Contents

[1 What is Computational Thinking?—Think as Computer Scientists Do](#)

Hideyuki Nakashima and Keiji Hirata

[2 Practice of Computational Thinking](#)

Keiji Hirata and Hideyuki Nakashima

[3 What is Meta?](#)

Yasuhiro Katagiri

[4 Life and Computational Thinking](#)

Noyuri Mima

[5 Joy of Understanding Computational Thinking](#)

Misako Nambu

1. What is Computational Thinking?—Think as Computer Scientists Do

Hideyuki Nakashima¹  and Keiji Hirata¹
(¹) Sapporo City University, Sapporo, Hokkaido, Japan

 **Hideyuki Nakashima**
Email: hideyuki.nakashima@gmail.com

Abstract

This chapter is written to provide foundational knowledge about computers for readers unfamiliar with computer science, serving as a primer. Jeannette Wing, who proposed computational thinking, defined it as “thinking like a computer scientist.” Computer science is a discipline that seeks to understand the nature of the world based on the worldview of information and computation. Computer science is a relatively young field, having only begun to be recognized as an academic discipline around the mid-twentieth century. This chapter explains the origins and concepts of computer science, detailing unique ideas not found in other academic fields, while carefully guiding beginners to understanding. This chapter provides a clear definition of computational thinking by examining the thought processes of computer scientists. It includes a wealth of concrete examples to enhance understanding. A comprehensive perspective is offered by contrasting the meaning of computation with concepts from physics, logic, philosophy, and mathematics. It also addresses the relationship between programming and computational thinking, which is often confused, as well as its connections to deep learning and generative AI.

1.1 Thinking in the Computer Age

Without computers, we would not be able to use electricity, water, banks, railways, or airplanes. Understanding how computers accomplish those tasks also helps us think and plan our daily life tasks. Computational thinking, which this book deals with, is not just knowledge or technology, but something like a compass for living better in the information world.

1.2 Computation and Thinking

The term computational thinking contains two words that everyone knows. One is computation, and the other is thinking.

1.2.1 Computation

When you hear the word “computation”, you might think of arithmetic operations. However, computations in the information world are not limited to arithmetic operations. Thinking of tactics to win a soccer game, planning a 10-day vacation in Vienna, creating the best dish with the ingredients in the refrigerator, ensuring logistics when roads are cut off by an earthquake, and even writing novels or composing music are all computations. What these have in common is that they are about “finding the optimal solution” for a goal within the constraints of the real world.

Computation is a powerful method for generating new information. For example, finding the greatest common divisor, GCD, of two numbers is not an easy task in general. The GCD of 4 and 6 can be easily found (2), but finding the GCD of large numbers is difficult. There is a method of Euclid for finding the greatest common divisor GCD of two numbers (defined in Sect. 1.7.5), which is generally referred to as “Euclidean algorithm”. This kind of computational method is called an “algorithm.”

You may ask what does finding GCD have something to do with our daily life. For instance, one textbook shows an example of a task of tile craftsmen.

You are trying to cover the floor of a room with short sides of 18.9 m and long sides of 24.5 m with square tiles. You are about to bake the tiles, but you want to use as few tiles as possible. How long should one side of the tile be?

The correct answer is 70 cm, as the GCD of 189 and 245 is 7. Note that the GCD is a concept that applies only to natural numbers, so we are multiplying by 10 for the calculation.

In actual daily life, you may rarely see situations where the GCD is used. However, the expression “greatest common divisor” is often used metaphorically in everyday life. For example, it means the largest common element of two or more opinions, which in this case means the compromise point of the discussion.

On the other hand, there is also the least common multiple. This means the smallest multiple of two natural numbers. There is a zodiac in the calendar, which consists of the combination of ten stems: 甲 (Jia), 乙 (Yi), 丙 (Bing), 丁 (Ding), 戊 (Wu), 己 (Ji), 庚 (Geng), 辛 (Xin), 壬 (Ren) and 癸 (Gui); and another twelve stems: 子 (rat), 丑 (ox), 寅 (tiger), 卯 (rabbit), 辰 (dragon), 巳 (snake), 午 (horse), 未 (goat), 申 (monkey), 酉 (rooster), 戌 (dog) and 亥 (pig). Each year is named by the combination of two stems. The least common multiple of 10 and 12 is 60, so the zodiac returns to its original state every 60 years. This is called the sexagenary cycle.

1.2.2 Thinking

When you hear the word thinking, you might think it’s all about mulling over ideas in your head. However, the thinking in computational thinking doesn’t stop there. What’s important is to actually try out what you’ve thought about in the real world, in other words, practice (more about practice will be discussed in Chap. 2). Imagine yourself as the hardware and software of a computer, freely executing computations in the real world surrounding you. Jeannette Wing, who proposed computational thinking, has stated that computational thinking is thinking like a computer scientist [Wing 2006]. As you will see as you read this book, computer scientists use the tool of the computer, or the metaphor of the computer, to try to understand various phenomena and problems in the real world.

In the real world, everything exists, functions, and changes under the absolute constraints of physical laws. Specific constraints include time progresses, entropy increases, and energy and mass are conserved. Neither the human body, the earth, nor space rockets can escape these physical constraints. On the other hand, computations are generally abstract operations and are

not subject to physical constraints. In the physical world, there are physical law constraints to build a building above a certain height, and there is a limit to the height. However, there are no such constraints in the world of computation.

For example, no matter how long the number of digits, it is possible to perform arithmetic operations, and no matter how large the database, it can be created if desired. However, it is not possible to actually prepare an infinite-digit abacus (or computer memory) to hold the numbers to be calculated, nor is it possible to actually create a database with a record number exceeding the number of atoms in the entire universe. Therefore, at least in the real world, it is meaningless unless arithmetic operations can be performed, and databases can operate within the realistic range of costs, time, and space. Although it is not constrained by physical laws, there are constraints in the world of computation, such as the amount of memory and computation time (collectively referred to as “computational complexity”). This is an important concept in computational thinking.

1.2.3 Conditions for Good Computation

Furthermore, the computation in the real world inevitably involves the perspective of value. It is because of the physical constraints mentioned in the previous section that value becomes necessary. For example, one person may prioritize the speed of arithmetic operations over the size of the number of digits. Another person may prioritize the number of queries that can be processed at once over the response time for each query to the database. This means that with the introduction of value, computations are classified into desirable and undesirable ones, and if that value changes, the desirable computations will also change.

Column 1 The difficulty of value evaluation

You may often want to optimize the procedure of your work. It could be the procedure of cooking or cleaning, but “what to optimize?” is a difficult question. For example, it is difficult to create a measure for “taste.” It also depends on individual preferences (values).

Even if a measure can be defined, there is a problem of trade-offs between multiple measures. In cooking, for example, there are various measures such as cost of ingredients, cooking time, taste, etc., and they often contradict each other. If you try to optimize only the taste, the cost of ingredients will be higher and the cooking effort will increase. Humans are making compromises between these multiple conditions that cannot be optimized at the same time. For example, optimizing the taste within a cost of 1000 yen for ingredients.

When a JR (Japan Railways) express train runs late, it’s standard practice to hold local trains to keep the express on schedule. From JR’s point of view, the express train’s punctuality is a higher priority because delays beyond a specific duration often result in ticket refunds. As a passenger, you always want your own train to have the highest priority, but that’s not always the reality. One of the authors once learned this the hard way on the Airport train heading from Sapporo to New Chitose Airport. An announcement came on: the Airport train would stop temporarily to let a delayed express train pass. Despite having left ample time, the author missed their flight. In the moment, the author felt the missed connection was simply a consequence of JR’s scheduling priorities. Yet, when considering the severe disruption the flight connection caused—and not just for one person—the importance of airline connections should, arguably, be given higher weight. Of course, everyone’s situation is unique, and not all

priorities align. Ultimately, this scenario highlights a conflict between JR's internal operational values and the values of time-sensitive passengers.

Currently, corrections to schedule disruptions due to accidents or bad weather are mostly done manually. Optimization is something computers are good at, but it seems that it has not been programmed because it is not self-evident which conditions should be optimized (they change depending on the situation). At this point, it is unclear whether AI will be able to understand the values held by humans in the future.

A good computation in computational thinking gives the optimal solution from a certain value perspective to the goals set within such physical constraints. The result changes depending on which of the multiple values is chosen. Computational thinking is a way of thinking that provides shortcuts and hints to realize this good computation.

No matter how smart a robot is, or how well an autonomous vehicle drives compared to a human, or an AI that creates works of art, the computer is performing symbol manipulation within it. ¹ Through these manipulated symbols, the computer understands the real world and then influences it. What becomes important here is how to assign (correspond) symbols to the things and concepts of the real world bound by physical constraints. For example, the computation needed to answer the question "How many minutes are in a day?" is immediately clear. The meaning of the symbols "day", "hour", and "minute" that appear in that computation is obvious. However, symbol manipulation of "joy" or "sadness", symbols assigned to the movements of the mind, seems difficult or almost impossible. "Every time I listen to Bach's music, I discover something new", "I don't mind sacrificing myself for justice", is it possible to realize a computer that calculates such things? This is a fundamental problem of artificial intelligence. And it has been debated for a long time. Can intelligence be expressed with symbols alone? ² Let's leave such difficult questions for another occasion. In this book, we only want to consider how to represent phenomena, constraints, and problems of the real world as data structures and functions in a program. Through this, we want to change our view of the world. As for how to take such perspectives, set problems, and educate them, Chap. ⁴ will also address this.

The issue of representation is a historical challenge in computer science research, and it has been dealt with in the artificial intelligence research from the early stages, but it has not been given much importance in traditional education. The first condition for good computation is appropriately representing real-world problems within a computer, and the same way of thinking can also be useful in our daily lives. I would like to provide examples of this. The understanding of the real world by a computer through symbols within a computer is quite similar to human's understanding of the real world through metaphors and analogies. Using metaphors and analogies allows us to feel as if we understand "it" by using similar parts as clues, but it does not accurately represent all of "it", nor does it mean that we have come to understand all of "it." However, good metaphors and analogies greatly help humans' understanding and are very useful.

The second condition for good computation is to explore and find more efficient or effective ways to solve phenomena or problems that have been brought into the world of symbols. Algorithms and procedures are ways to show the path to solutions. They can be used not only for computer programs but also for humans. Reducing the size of the array needed for computation or reducing the number of iterations is involved in efficient and effective problem-solving. This way of thinking is effectively applied to human thought as well, probably because human memory

capacity, human recognition/computation speed and human error rates are also influenced by physical constraints.

Here we summarize the conditions of good computation; the first condition is to appropriately represent real-world problems within a computer, and the second one is to explore and find more efficient and effective ways to solve phenomena or problems that have been brought into the world of symbols. Let's illustrate the relationship between these conditions for good computation and computational thinking (Fig. 1.1).

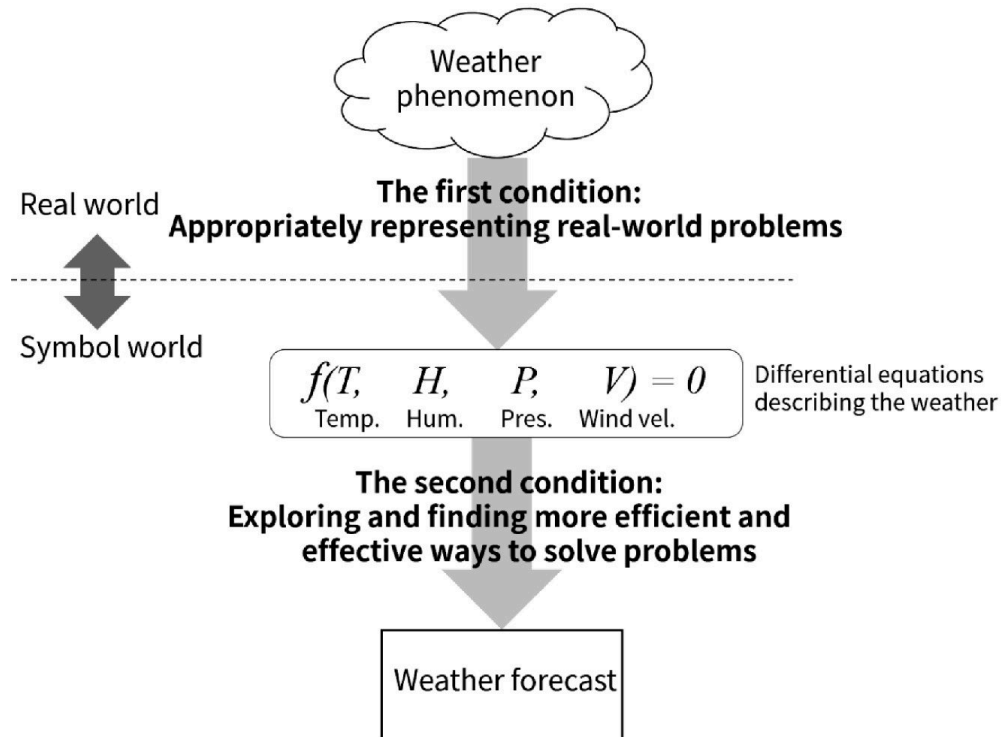


Fig. 1.1 Computational thinking that satisfies the two conditions for good computation

When facing phenomena and problems in the real world that seem ambiguous, massive, and sometimes even structureless, how can we effectively meet these two conditions of good computation? Let's consider this problem in detail below.

1.3 Exponential Explosion

When programming a computer for calculation, one must be careful of a phenomenon known as exponential explosion. This refers to the situation where the amount of data increases and the computation time and memory capacity required increase too much to handle. Such a situation is called the exponential increase as mentioned in the column, and "explosion" is a term that emphasizes the point of rapidly becoming intractable.

For instance, exponential increase can be reproduced by a process of adding up all the numbers that have appeared so far to create the next number as follows.

$$X_{n+1} = \sum_{i=1, \dots, n} X_i$$

In fact, let's see what happens if we start with $X_1 = 1$, and you obtain the sequence of 1, 1, 2, 4, 8, 16, 32, 64, ... As is evident, starting from the second term, numbers grow exponentially and can be expressed as 2^n . Why does the calculation produce such a beautiful sequence of numbers? You may understand it immediately if you think of the asymptotic formula to compute X_{n+1} as follows.

$$\begin{aligned} X_{n+1} &= \sum_{i=1, \dots, n} X_i \\ &= X_n + X_{n-1} + X_{n-2} + \dots + X_1 \\ &= X_n + X_n \\ &= 2X_n \end{aligned}$$

1.3.1 Friends of Friends of Friends of...

There are many examples in the world where numbers increase exponentially. You may have heard the fun phrase "a friend's friend is a friend." If each person has 10 friends, then a simple calculation tells us that friends of a friend would amount to 100 people. This situation is expressed as having 10 friends at the first level and 100 friends at the second level, and thus, the number of friends increases exponentially. Now, in sociology, there is a well-known rule called "six degrees of separation." This means that anyone can become friends with all the people on earth if he or she considers up to the sixth level of friends (assuming there is no overlap in friendships). The current global population is 7 billion, which is approximately 44^6 ; this implies that each person has an average of 44 friends. Intuitively, it seems reasonable that each of us has about 50 to 100 friends or acquaintances.³ What is surprising here is that by following the chain of friends-of-friend only 6 times, you can reach everyone on Earth. Amazingly every reader of this book can reach the President of the United States by tracing your friends-of-friend chain for 6 times! You might feel the power of exponential growth.

Computational complexity

In the world of programming, the effort required for (the memory and computation time required to execute the computation) is an issue. This is shown as a quantity relative to the size of the problem and is called the order, which is represented by a function O . For example, $O(N)$ means that resources proportional to the size of problem are needed. $O(k^N)$ is called exponential the computational complexity increases rapidly with the size of problem; it is usually not practical for computation. Conversely, $O(\log N)$ is an easy problem to handle.

Column 2 Exponential Acceleration of Evolution

Both the evolution of organisms and the evolution of science and technology have been developed so far through the combination of "parts." Evolution accelerates exponentially as the parts get larger. In terms of organisms, the evolution that was occurring at the molecular level extends to the evolution at the cellular level due to the creation of a part called a cell, and it became dramatically faster (incidentally, it is said that all existing organisms are descendants of a single cell that was created by chance). Furthermore, multicellular organisms were born from unicellular organisms, giving birth to a new unit of evolution. What was once a whole is divided

into parts, and these individual parts are differently combined to create a new whole, which becomes a new unit. It gives rise to the evolution at a higher speed.

Then humanity was born and gave birth to science and technology. The realm of the evolution shifted from the level of individual organisms' intelligence to the that of the intelligence of the entire society created by those organisms (humans). Necessarily, the laws governing evolution were also changed. Until cells were created, chemical reactions determining molecular structures were the dominant laws. When a cell became a unit, the dynamic non-equilibrium system as life became the governing principle. When organisms became a unit, the theory of evolution in the sense described by Darwin became the governing principle. Furthermore, when the knowledge created by humanity became a unit, the methodology of science and technology became the governing principle.

Science and technology have been developed on the sum of past knowledge. We learn only in a few hours the knowledge that old scientists like Galileo and Newton discovered over tens of years. Then we connect it to the next discovery. Thus, human knowledge is also increasing exponentially.

If we look at the evolution of evolution in a timeline, we should recognize that the evolution of evolution is being accelerated exponentially:

4.5 Billion years ago: Birth of the Earth
4 billion years ago: Birth of life.
2 billion years ago: Birth of cells.
1 billion years ago: Birth of multicellular organisms.
0.5 billion years ago: Birth of eyes (Cambrian explosion).
0.07 billion years ago: Birth of primates.
0.005 billion years ago: Birth of humanity.
0.00001 billion years ago: Birth of agriculture.
0.0000003 billion years ago: Industrial revolution.
0.00000007 billion years ago: Birth of computers.

1.3.2 Cell Division

Let's consider the number of cells as an example of a living thing that increases exponentially. You know that organisms grow by cell division. Imagine a fictional creature whose every cell doubles every day. If this organism was 10 cm tall one day, please guess how tall it would be the next day. No, it's not 20 cm. The correct answer is that the height the next day is the cube root of 2 multiplied by 10 cm. Doubling the height means doubling the width and depth, so the volume increases eightfold (Fig. [1.2](#)). Going back to the first question, if the number of cells doubles in a day, how many days will it take for the height to double? Yes, it's for 3 days. When the height doubles, the weight becomes 8 times. It takes 1 day for the weight to double, 2 days to quadruple, and 3 days to increase eightfold. If the fictional creature grows at this rate, it takes only 27 days for the organism that was 10 cm on the first day to become 50 m tall, which is the height of Godzilla. This is because 50 m is 500 times 10 cm (here, we use a commonly used approximation $500 \approx 512 = 2^9$), so the volume is approximately 2^{27} times. It is not surprising that an organism

of only 10 cm in height grows into Godzilla in 27 days by cell dividing once a day. This is the power of exponential increase.

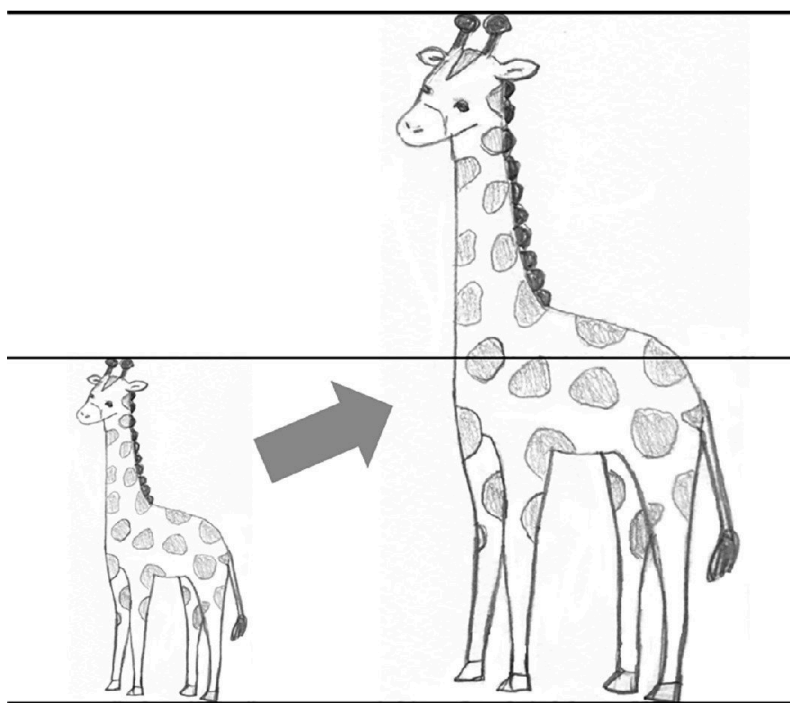


Fig. 1.2 Doubling the height results in an eightfold increase in volume

1.3.3 The Number of Your Ancestors

Every human has two parents, and four grandparents. That is, you have two ancestors one generation back, and four ancestors two generations back. In this way, the number of your ancestors increases exponentially. So, how many ancestors would you have if you go back 30 generations? It's about 1 billion. If we assume one generation is about 30 years, 30 generations is 900 years. So, you were born from about 1 billion ancestors who lived on this Earth 900 years ago.

But you may feel something is wrong here. The actual world population 900 years ago was just over 300 million. Generally, when a couple gets married, the husband has four grandparents, and the wife has four grandparents. But if cousins marry, the total number of grandparents is reduced to six. That is, if you trace your paternal and maternal ancestors, there must be some overlap. In other words, your father and mother can be traced back to one common ancestor at some point. In this sense, almost all people are related by blood somewhere.

1.3.4 Pyramid Scheme

Let's explain exponential growth with an example of a pyramid scheme or multi-level marketing. The job of a pyramid-scheme member is to recruit new members and to force them to purchase and sell products. If you recruit someone and they become a member, that person becomes your direct downline (child member). That is, you are the parent member of that new member. Then, a new member purchases a product from you and sells it to his/her child member with a certain amount of commission fee added. According to the rule of the multi-level marketing, a member

who earns a commission fee must pay a certain percentage (pay-up rate) of the commission fee to the parent member.

Each member has a mission to recruit more than N new members, and there is a quota for product procurement and sales. People who are invited to become a pyramid-scheme member by an acquaintance may hesitate whether to become a member or not, but then the parent member says something like “The earlier you become a member, the more income you will earn,” and urges them to join.

Is it true that “the earlier you become a member, the more income you will earn”? Let’s actually calculate it and check with concrete figures. Suppose that it is a rule that one member recruits at least two new members, purchases a product for 100 yen, sells it for 300 yen, and the quota is 100 items. Thus, the commission is 200 yen per item, for a total income of 20,000 yen. If you pay 50% (pay-up rate) of the commission fee to the parent member, you will have 10,000 yen left, and you will pay 10,000 yen to the parent member. Since that parent member has at least two child members, the total is two-fold 10,000 yen paid up. If there is a parent member above that parent member, you must pay up 10,000 yen, which is 50% of 20,000 yen, but if you think that the child member earned 10,000 yen while the parent member was sleeping, it’s a super easy business! Conversely, let’s consider the case where you have grandchild members. Suppose you have two grandchild members for each of the two child members, then a total of 40,000 yen is paid up from the four grandchild members to the two child members, and 20,000 yen is paid up from the two child members to you, and you pay up 10,000 yen to your parent member. In other words, you have indirectly received 10,000 yen from the four grandchild members. As you can see from a little calculation, as long as there are great-grandchildren and great-great-grandchildren beyond the grandchild members, you will receive 10,000 yen for each generation increase (Fig. 1.3).

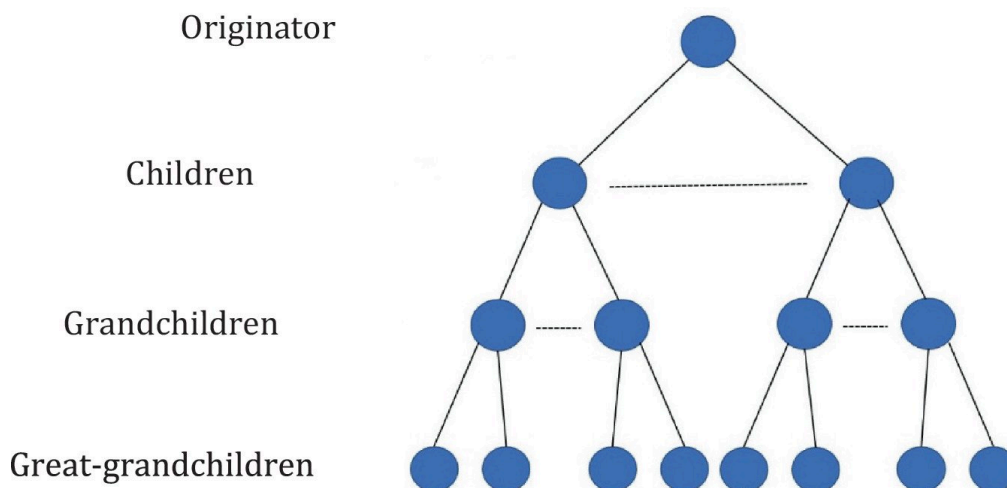


Fig. 1.3 Structure of generations in pyramid scheme

Now, let’s get to the point. Let’s reveal the answer to the question, “Is it true that the earlier you become a member, the more income you will earn?” The answer is qualitatively correct, but if you apply specific numbers and calculate, you will find that it breaks down immediately. The reason why this pyramid scheme works is the increase in members, but the speed of that increase is the problem. The mechanism of the pyramid scheme, “one member enrolls N or more new members,” leads to an exponential increase in the number of members. $N = 2$ is the so-called doubling game, which doubles in order: 1, 2, 4, 8, 16, 32. At first, these are small numbers, but at the 10th step, it’s

about one thousand, ⁴ at the 20th step, it's about 1 million, and at the 30th step, it's about 1 billion, rapidly inflating. In other words, this pyramid scheme reaches almost the entire population of the earth after 30 generations, and that's where it ends. Under these conditions, the "originator" who started this pyramid scheme can only obtain a mere 300,000 yen.

$N = 2$ shows that you can't generate much money. In a pyramid scheme, you must increase the proportion of the pay-up or increase the number of child members.

Figure 1.4 shows the increase in the number of child members in each generation. We increased it from 3 to 22. Although there is an increase approximately proportional to the number of child members, the maximum is only about 3.5 million yen, so it's not a big profit.

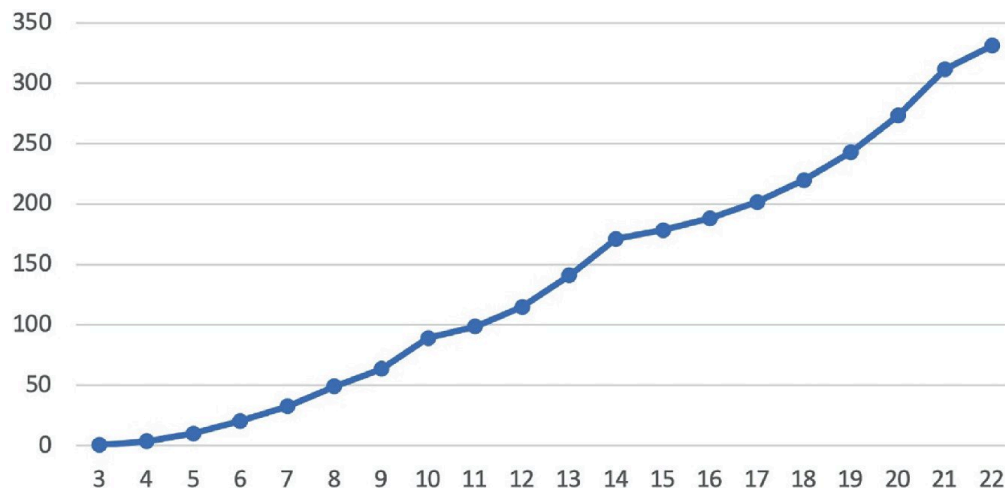


Fig. 1.4 Pyramid-scheme: number of child members and originator income (10 K yen) at pay-up rate of 50%

Figure 1.5 shows the change in the pay-up rate. We changed it from 0.1 to 1.0. Here, there is an exponential explosion of increase. A pay-up rate of 1.0 means that the parent takes all, so only the originator who started first gets the money, and there is no income for the other intermediate generations, so it does not work as a pyramid scheme. But as we saw above, at the pay-up rate of 0.5, it doesn't make much profit, so you might aim for 0.7 (income of 42 million yen) or 0.8 (income of 400 million yen). The closer the pay-up rate gets to 1.0, the more the originator's income increases. However, the rate of increase is difficult to discern in Fig. 1.5, so we introduce a logarithmic scale on the vertical axis. This is shown in Fig. 1.6. From 0.5 to the right, it's almost a straight line, so you can see that it shows an exponential increase.

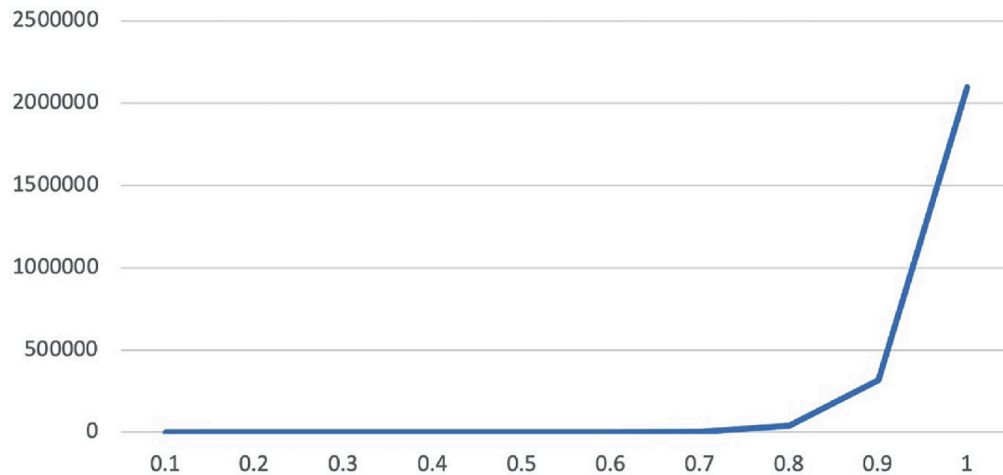


Fig. 1.5 Pyramid-scheme: pay-up rate and originator income (10 K yen)

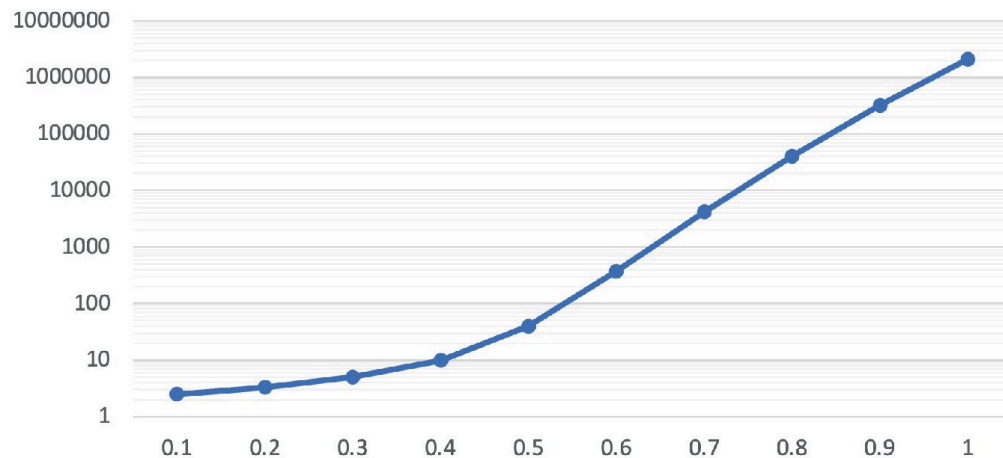


Fig. 1.6 Pyramid-scheme: pay-up rate and originator income (Log scale in 10 K yen)

Either way, when you calmly look at the figures relevant to the pyramid scheme shown here, it would seem odd that many people in the world are being duped by the pyramid scheme. There may not be anyone who profited a lot from the pyramid scheme. It is probably because people have a vague understanding of exponential increase and suppose that it is naively applied to their own profit in the pyramid scheme.

1.3.5 Traveling Salesman Problem

Let's consider a more realistic example, there is a problem called the traveling salesman problem. This is a problem in which a salesman finds an efficient stroke order to travel multiple cities for sales activities, i.e., making the plan of a visit, stopping only once in one city with saving time and transportation costs. It has been known as a typical example of exploration for a long time and one of the most difficult problems to find the optimal solution. It is because the computational complexity increases exponentially as the number of cities visited grows.

Here, we will not give a solution to this problem but just an important notice thinking of actual logistics. For example, when a courier delivers multiple packages to multiple homes, in what order is it most efficient? In reality, there are other conditions that we have to take into account such as the specified time of delivery and the total mileage, which make the problem more complicated.

Without a computer, the optimal solution cannot be obtained. From the standpoint of computational thinking, the important thing is not to solve this problem, but to know that this problem is difficult.

1.3.6 Avoiding Exponential Explosion

As we have seen so far, when an exponential explosion occurs, the amount of computation increases rapidly, making it impossible to compute with practical computational resources (time and memory that are available). There are some known ways to avoid this:

- Divide and conquer: This is a method of dividing the problem into independent units and solving each one (Sect. [1.4](#)).
- Settle for an approximate solution: This is a method of giving up on complete computation and accepting an approximate value. For example, the traveling salesman problem (Sect. [1.3.5](#)) inevitably causes an exponential explosion when calculated normally, so several methods have been developed to find approximate solutions.
- Record partial solutions: As mentioned at the beginning of this section, exponential explosions often have a structure where you repeatedly use what you have accumulated so far, so you record what you have calculated in a table at each time step and use the values in this table for the second time.

1.4 Divide and Conquer

Previously, we discussed how an exponential explosion in computations can become unmanageable. Essentially, this explosion occurs because we have to consider all possible scenarios or cases of the subject. Therefore, let's first consider properly dividing the entire subject into small parts that do not cause an exponential explosion. If there is no relationship between these parts, or if any relationship can be ignored, the processing of each divided part will be light. This might allow us to calculate the entire subject without causing an exponential explosion. In other words, if we can divide it well, we can govern it with a light effort. In the best-case scenario, the entire computation can be completed with effort proportional to the number of divided parts. In other words, the original computational complexity of

$$N_1 \times N_2 \times \cdots \times N_k$$

can be reduced to only

$$N_1 + N_2 + \cdots + N_k.$$

When you hear divide and conquer, you might think of the ancient Roman Empire, which governed vast territories, Western colonial management, or the Edo Shogunate. The key to divide and conquer here was to sever the solidarity and cooperation between regions, ethnic groups, and countries (sometimes, the rulers even incited mutual resentment and hatred). By severing these, they were able to govern each divided region, ethnic group, and country with minimal effort in total.

1.5 Recursion

According to the Super Daijirin, a Japanese dictionary,

Recursive

1. The result of one's actions returns to oneself. Feedback.
2. In mathematics, etc., the thing to be defined appears in the definition itself.

The second definition is our concern here.

"Recursive" is one of the important and powerful methods in programming, which allows many problems to be described concisely by including calls to itself in the program. First, I would like to start with an example of what recursion is.

This phenomenon can also be seen in everyday life. For example, in a pair of mirrors facing each other, image B is reflected in A, and A is reflected in the image of B, and this continues indefinitely.

Column 3 Mutual Exclusion

If multiple processes access a single resource at the same time, one result may overwrite the other, causing problems. Mutual exclusion is to ensure that only one process can access at a time.

When accessing a single resource, you first lock the resource to prevent other processes from accessing it. This is called a lock. You can safely access the resource after locking it and unlock it when you're done. If another process wants to use the resource while it's locked, it has to wait until it unlocked. This is how mutual exclusion is achieved. This book is also written by multiple authors, so we use mutual exclusion when modifying files.

The key to efficient mutual exclusion is to keep the interval between lock and unlock as short as possible. To achieve this, we carefully consider what should be shared resources and devise algorithms accordingly.

Mutual exclusion is also used in railways. Nowadays it is computer-controlled, but in the past, to prevent more than one train from entering a single track, only one tablet was prepared for that section (called a block section), and trains without it could not enter. This was featured in the Japanese movie "Railroad Man" (original title "Poppoya") starring Takakura Ken.

Speaking of which, it seems that there are many good hackers among model railway enthusiasts. (Or should I say, there are many model railway enthusiasts among good hackers?).

There is a Rakugo (Japanese traditional storytelling) called "Hell's Eight Scenic Spots of the Dead". A Rakugo storyteller who goes to hell will be able to go to paradise if he makes Enma (hell's guardian) laugh, so he tells Enma a Rakugo story.

Enma: Oh, a Rakugo storyteller, I love Rakugo. What kind of story is it? Rakugo storyteller: How about "Hell's Eight Scenic Spots of the Dead"? Enma: Oh, let's listen to it.

So, the same Rakugo starts in the Rakugo. Then, "Hell's Eight Scenic Spots of the Dead" starts again in that Rakugo, and it doesn't stop. In reality (?), however, Enma should stop it halfway through.

Describing the story in the order in which things happened like above, it continues to unfold indefinitely. However, if you think about it the other way around, you can describe repetitive tasks or computations in a short way. If you can capture various phenomena in everyday life recursively, you will be able to understand things in a simpler way. For example, the predicate “append” that connects two lists in the programming language Prolog is defined as follows:

```
append([], Z, Z).
append([E|X], Y, [E|Z]) : - append(X, Y, Z).
```

The first line is the termination condition, and computation stops when the first argument becomes an empty list. If the first argument is not empty, the control goes to the second line; it is decomposed into the head element E and the rest X , and the predicate is to append E and Z at the place of the third argument. Here, Z is the result of append of X and Y obtained from the recursive computation of $\text{append}(X, Y, Z)$. It is enough to see that the second line is recursive. The LOGO language’s “word list” program (Sect. 4.4) is also written recursively.

This is a strategy of dividing a large task or computation into smaller ones (see also Sect. 1.4).

A famous example where recursive solutions are effective is the Tower of Hanoi puzzle (Fig. 1.7). This involves moving stacked discs one at a time to move the entire pile to another rod. You must not stack a larger disc on top of a smaller one. This is easy to think about recursively (Fig. 1.8).



Fig. 1.7 The Tower of Hanoi puzzle

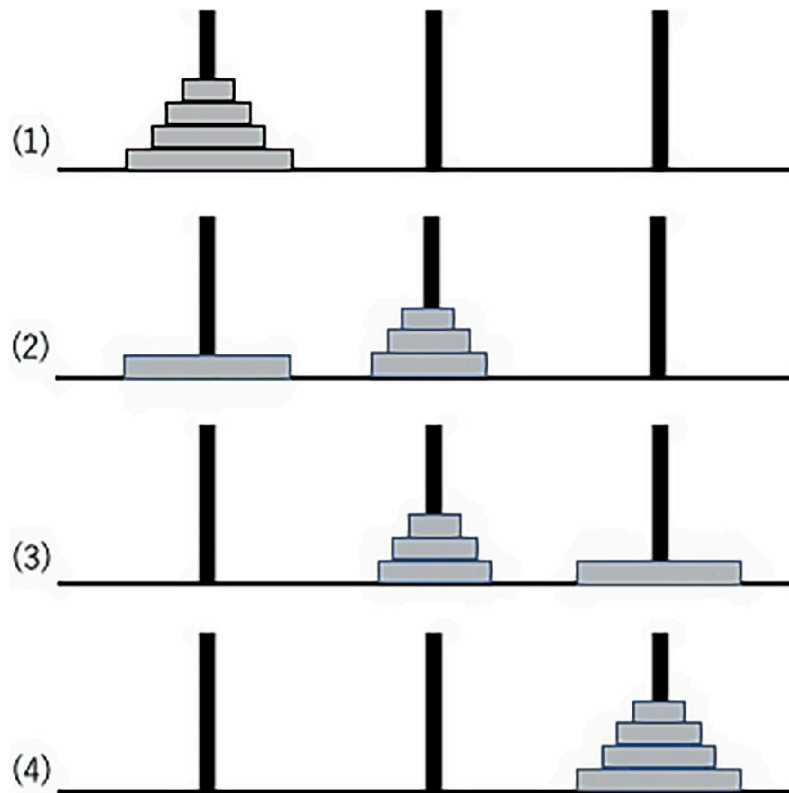


Fig. 1.8 How to move discs in the Tower of Hanoi

1. In order to move N discs to the target rod
2. Move $N - 1$ discs to a different rod named A (recursive procedure)
3. Move the remaining 1 disc to the target rod
4. Move the $N - 1$ discs stacked on rod A to the target rod (recursive procedure).

In other words, to move N discs, you need to use the operation of moving $N - 1$ discs twice, in which the slightly smaller move operation is called recursively.

1.6 Information as a Worldview

“Computational thinking” may be rephrased as “information theoretic thinking.” This emphasizes information as a worldview alongside matter and energy. In other words, it advocates viewing the world from the perspective of information and its operations.

Have you ever read the flash fiction written by a Japanese novelist Shinichi Hoshi titled “The Net of Voices”? One of the characters claims that information is energy:

In an era without knowledge of hydroelectric power, people probably didn't think of water as an energy source, and they certainly didn't dream that uranium-containing ore was a tremendous source of energy. Uranium ore was just a rock. Isn't it natural to think that the source of energy is the knowledge of nuclear power? In the early days of the nuclear age, information about atomic bombs was a high-level state secret, and there were scholars who were executed for giving it to foreign countries. Even then, picking up uranium ore was nothing special. The source of energy is information. (Translated by the authors)

Shinichi Hoshi, "The Net of Voices" (Kadokawa Bunko)

In other words, even if raw material exists, without the knowledge or information on how to extract energy from it, it is just a mere substance. This is why it is argued that information is a worldview. Whether you see the stone in front of you as just an ore, or as a valuable energy source, that difference is information.

What this example shows is that there are things that cannot be seen just by looking at the world from the perspective of matter and energy. Einstein's famous equation $E = mc^2$ describes the relationship between the mass of matter m and energy E . This equation explains the energy produced during nuclear fission and fusion, but it alone can neither build an atomic bomb nor unfortunately a fusion reactor. For this purpose, you need information on how to build an atomic bomb and a fusion reactor.

The general term "information" is used in a variety of contexts: Since around 1980 with the spread of personal computers, and since around 1990 with the spread of the Internet, the term "informationization" has become more common. This refers to the use of information technology when companies think about new products or services, or when they want to make individual processes in their operations more efficient and smarter. On a personal level, using information technology as a tool to make daily life more comfortable and effective is also included in informationization.

Unfortunately, there are still not many examples in Japan or around the world of using information technology to revamp traditional workflows or create new business models. Currently, attention is focused on GAFA (Google, Amazon, Facebook, Apple), which have become huge companies through information technology, but the world's information technology engineers intuitively understand that the real transformation brought about by information does not stop at the level of GAFA. They are convinced that information is just on the cusp of further fundamental changes to the world and society we live in.

Looking towards such a future, in recent years, the concept known as "Digital Transformation", or "DX" for short, has been proposed. It aims to launch innovative new businesses using information technology and/or digital technology and fundamentally improve people's attitudes and behaviors towards life. DX is about understanding the true value of information and practicing the worldview of information.

1.7 What Computational Thinking is Not

The discipline centered on the worldview of information is "Information Science." Computational thinking is based on the thinking methods created by researchers and engineers of information science who study information. Let's look at how this information science differs from existing disciplines such as physics and mathematics.

1.7.1 Differences from Physics

Let's continue with the example of uranium from the aforementioned book "Voice of the Net" by Shinichi Hoshi. From a physical perspective, the atomic number of uranium is 92, and its atomic weight is 238. The atomic number is the number of protons in an atom, and the atomic weight is the weight of the atom, which is roughly the sum of the number of protons and neutrons.

Therefore, we know that uranium 238 is made up of 92 protons and 146 neutrons. As you may know, uranium is a radioactive substance that spontaneously undergoes nuclear fission to split into smaller atoms. This fission occurs slowly in nature, such as inside the Earth.

If we go back to a more fundamental level of matter, matter is made up of atoms, and those atoms are composed of a nucleus made up of protons and neutrons, and a cloud of electrons around it. It is known that atoms generally have an equal number of protons, which carry a positive charge, and electrons, which carry a negative charge, so atoms are electrically neutral. How matter changes, and how energy transforms through heat and motion, is the worldview of "physics", and the knowledge of physics.

Therefore, physics also deals with a lot of "hows", that is, information. But this information is generally descriptive information about the state and movement of matter and energy. This is why physics is said to be descriptive. It does not include information on how to manipulate matter and energy, or how to build machines like nuclear reactors that manipulate and control matter and energy. This kind of information and knowledge for creating something by human hand is dealt with in the field of "engineering", not physics.

The study of observing the state and movement of objects and discovering principles and laws therein is generally called "science." The definition of science is difficult, but let's consider it a method of objectively organizing and understanding things. Things exist first, and then humans understand them. Understanding the laws of nature is natural science, and physics, which deals with matter and energy in the natural world, is a branch of science. Nevertheless, physics is at the core of natural science. In fact, the methodology of natural science has developed based on physics.

Both science and engineering revolve around the loop of hypothesis formation and its verification. In science, to achieve their goals, hypothesis formation involves observing phenomena and inferring the underlying principles or laws. These inferred principles or laws are called "hypotheses." These hypotheses are then tested through experiments to verify their correctness. By going through this loop, principles or laws that explain phenomena more accurately and consistently are gradually being developed.

The situation in engineering is pretty much the same. In engineering, the hypothesis formation involves devising methods to operate the target as intended or to realize the desired function. The correctness of the hypothesis is verified by implementing the method. By going through this loop, the methods to operate the target as intended or to realize the desired function are gradually being developed.

Now, let's shift our focus from matter and energy to information. Information science is a discipline that centers on the worldview of information. In information science, we deal with methods to generate new information based on various information. This is called "inference" or "prediction." Let's introduce a simple example of inference. Let's consider three propositions A, B, and C as follows:

A: Socrates is human,

B: Nightingale is human,

C: If you are human, you get hungry.

From A and C, we can conclude that Socrates gets hungry. From B and C, we can conclude that Nightingale gets hungry. This type of inference is called “deductive reasoning.” Also, if we interpret the result of the inference as something that will happen in the future, it becomes a prediction that Socrates will get hungry.

Let’s introduce another simple example of inference. Again, let’s consider three propositions D, E, and F as follows:

D: Socrates and Nightingale are both human,

E: Socrates gets hungry,

F: Nightingale gets hungry.

From these three propositions, we can infer that “if you are human, you get hungry.” Of course, human beings are not just Socrates and Nightingale, but if Socrates and Nightingale get hungry, we might assume that other human beings will also get hungry. This type of inference is called “inductive reasoning.” What should be noted here is that the conclusion “if you are human, you get hungry” is not necessarily 100% correct. That’s what inductive reasoning is. It only says that it will probably be correct in most cases. If you think of it as a prediction, it’s natural that it can be right or wrong.

Some readers may be wondering by now, did not information originally exist in the natural world? Does the concept of information only make sense when there are human beings? Information has two aspects: information as inherently physical existence from the beginning and information as a concept that has meaning only when human beings exist. In fact, there are concepts shared between physics and information science. One of them is “entropy.” Entropy, simply put, is the “degree of disorder.” For example, a snowflake is a neatly arranged set of water molecules, a state of low entropy. When heat (energy) is added, the entropy increases (in everyday terms, it “melts”) and becomes water. Water is a more disordered state than ice. Whether this is disordered or not can be determined by how many possible states that can be taken. Even though there are various types of snowflakes, the number of states called crystals is far fewer than that of states called water.

If left alone, snow will turn into water, but the reverse does not happen. In other words, matter changes from a low entropy (ordered) state to a high entropy (disordered) state, but something needs to be done to cause the reverse. There is a saying, “Spilt water will not return to the tray”, which means that water spilt off from a tray will not return to a tray if left alone. In this way, entropy can only change in an increasing direction if left alone. This is called the “law of increasing entropy.”

From here, I will make a bit of a bombshell statement regarding the number of states. It is human beings who determine the number of states for molecules contained in crystals and water. In fact, comparing the numbers of states of snowflakes and water are physically meaningless. Under the condition that the number of snowflake molecules and that of water molecules are the same, the numbers of their states are just the same. Moreover, this is because it is impossible to distinguish each molecule in the first place. However, we human beings can recognize snowflakes as having specific patterns, right? Even for water, if you put water in a cup, water should maintain a specific shape. At that time, you human beings likely recognize it as one “state” even though there will be a huge number of states in a physical sense.

Let's explain entropy by using a shuffle of cards. Since it's tough to illustrate with 52 playing cards due to space constraints, consider we have a reduced sized set of playing cards: spades, clubs, hearts, and diamonds, each with a jack, queen, king, and ace. There are 16 playing cards in total. If these cards are separated by type and arranged in ascending order as follows:

(1) ♠J ♠Q ♠K ♠A ♦J ♦Q ♦K ♦A ♥J ♥Q ♥K ♥A ♠J ♠Q ♠K ♠A,

We can say that the entropy is low. If you shuffle this, the order may be changed and the order becomes more random, and the entropy increases.

Now, if separated by number and arranged for the four types in order as follows:

(2) ♠J ♦J ♥J ♠J ♠Q ♦Q ♥Q ♠Q ♠K ♦K ♥K ♠K ♠A ♦A ♥A ♠A,

This can also be said to be a state with low entropy.

What if we would decide that the following is a regular state.

(3) ♥K ♠A ♥J ♠J ♦Q ♥Q ♠A ♦K ♠J ♦A ♠Q ♠K ♥A ♦J ♠K ♠Q?

There are no simple rules. However, physically, there is no reason to distinguish (1) and (2) from each other. Why do we think (1) and (2) are different from (3), although (1), (2) and (3) are certain states of the playing cards, respectively? It's our cognitive mechanism that is creating the "difference." (1) and (2) can be described and recognized by simpler rules, but (3) cannot.

When you shuffle a well-ordered deck of 52 playing cards, it gradually becomes random. The probability of accidentally returning to the original state is very low, practically zero.⁵ However, the probability of reaching any specific state is equal and also practically zero.

There are two kinds of things in the world; one is the things that are orderly and understandable to human beings, and the other is the things that appear random. Therefore, "entropy cannot be defined without humans." This phrase was written in "The Order of time" by theoretical physicist Carlo Rovelli. This issue is interesting in itself, but if we delve deeper, we will go off on a tangent, so I will stop here. If you are interested, please refer to the book by Carlo Rovelli.⁶

1.7.2 Differences from Science and Engineering

There is a term "computer science." Can the study of computers be a science? Computers do not exist in nature. They are apparently human-made products of engineering. However, there are already fields called "computer science" or "computer engineering" in the world. Don't you readers feel a slight discomfort with the term "computer science"? I would like to solve this mystery for a moment.

In Japanese, there are expressions like "science and engineering" or "science and technology." In English, there are words like "science and art" as well. Interestingly, when we say art in Japanese, we think of fine arts such as painting, but art in English means technology and skill. Martial arts are techniques of fighting, and liberal arts are techniques (or knowledge) for being liberal. Organizing these terms in the following figure might make it click.

Column 4 The Law of Increasing Entropy and the Arrow of Time

Physical equations (such as Newton's equations of motion or the equations of relativity) contain a variable. If this is replaced with, in other words, if time is reversed, the equation still holds. This means that if you rewind a video capturing a physical phenomenon, you cannot distinguish it from playing it forward. However, there is one difference. friction. Without friction, you wouldn't be able to tell if a billiard ball's movement was being rewound, but with friction, the ball gradually slows down and eventually stops. If you rewind this, the stopped ball starts moving, which is strange.

Friction is an example of the law of increasing entropy, which means that the ball gradually turns into heat as it moves. The law of increasing entropy is the only thing in physics that determines the direction of time. The existence of time has long troubled physicists. Many books have been published on the "arrow of time".

Recently, a physicist has emerged who claims that this entropy is due to human cognitive function. His name is Carlo Rovelli. This is a unique perspective. He argues that the flow of time for humans exists because humans exist in this universe. The conclusion may still be far off, but I think this is a rare example of a physicist thinking computationally.

Now, I mentioned that physics is called "descriptive", and contrastingly informatics is called "operational", and specifically creates new information, it is called "constructive." Compared to physics, the worldview of information is well expressed by these words "operational" and "constructive."

Computational thinking as presented in this book is part of engineering thinking. In this book, we use the terms "computer science" and "computer scientist." According to Fig. 1.9, we Japanese should instead use "computer engineering."

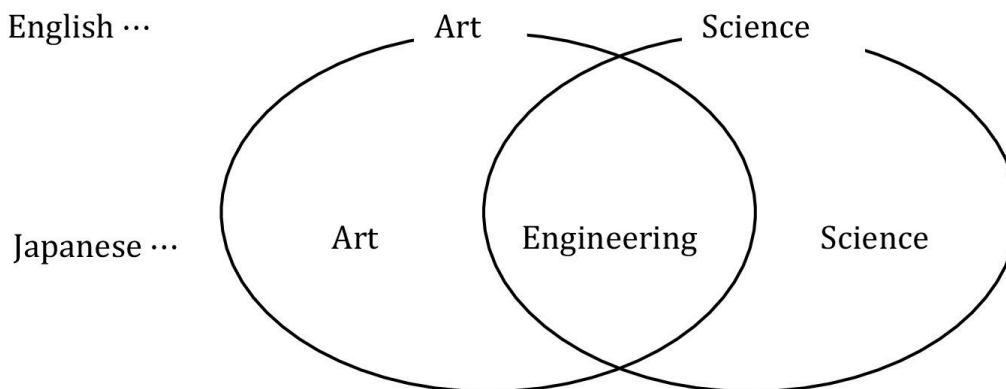


Fig. 1.9 The relationship between Japanese Art, Engineering and Science, and English Art and Science

1.7.3 Differences from Mathematics and Logic

Mathematics manipulates information differently from physics. Mathematics combines one piece of information with another to create new information. There is a divide among researchers as to whether mathematics is something created by humans or something that exists naturally.

There are those who say that every natural number such as 1, 2, 3 etc. is invented by humans, while there are those who believe that they exist in nature even without humans. Logic and mathematics are slightly different from natural sciences, but it wouldn't be unnatural to call them science collectively.

As an example of such mathematics, we will take up Euclidean geometry. Euclidean geometry is roughly the same as the geometry that we Japanese learn in elementary school. To explain Euclidean geometry precisely, it starts with the definition of a point and the definition of a line. A definition is a sentence that precisely determines what properties a concept has and how it is made. Then, properties called axioms are given using that definition. An axiom is the starting point that can be assumed to naturally hold without proof, such as the transitive law and reflexive law in logic.

Furthermore, there are properties called postulates, which are what would happen if we assumed they were correct (these are also sometimes called axioms); for example, “parallel lines do not intersect” and “there is only one line connecting two points.” These postulates cannot be proven correct.

Anyone can define anything freely and give axioms and postulates freely. Then, properties derived from axioms and postulates by deductive reasoning are called theorems; for example, the sum of the interior angles of a triangle is 180 degrees, Pythagoras’ theorem, etc. However, if contradictory axioms are given, contradictory conclusions will be derived, and if unrelated postulates are given, no meaningful conclusion will be derived. These meaningless conclusions cannot be called theorems.

As another example, let’s look at the definition of natural numbers. There are various ways to do this, but here we will introduce a definition using sets. Since they are natural numbers, we create them in the order of 0, 1, 2, etc. We use existing information to create new information. First, let’s assign the symbol ‘0’ (zero) to the empty set. Let’s write it as $0 = \{\}$. The empty set does not contain elements and is unique, that is, there is only one empty set in the world.

Since all numbers can be represented as sets, let’s define $n + 1$ as $n \cup \{n\}$. This term means the set that is the union of the set that represents n and the set that has only n as an element. In this way, the set of $n + 1$ has one more element than n , and the newly added element $\{n\}$ is obviously not included in n . Therefore, when you have made 0, 1, $\dots$, n in order, you can always create a set that is different from the sets you have created so far. Let’s see what kind of sets are actually being derived in order.

$$\begin{aligned} 1 &= 0 \cup \{0\} = \{\} \cup \{\{\}\} = \{\{\}\} \\ 2 &= 1 \cup \{1\} = \{\{\}\} \cup \{\{\{\}\}\} = \{\{\}, \{\{\}\}\} \\ 3 &= 2 \cup \{2\} = \{\{\}, \{\{\}\}\} \cup \{\{\{\}, \{\{\}\}\}\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\} \end{aligned}$$

It goes like this. However, there are too many brackets and it’s hard to understand what’s going on. Would it be easier to understand if we rewrite it as follows?

$$\begin{aligned} 1 &= 0 \cup \{0\} = \{\} \cup \{0\} = \{0\} \\ 2 &= 1 \cup \{1\} = \{0\} \cup \{1\} = \{0, 1\} \\ 3 &= 2 \cup \{2\} = \{0, 1\} \cup \{2\} = \{0, 1, 2\} \end{aligned}$$

In this way, it should be clear that the newly created set can be written as follows:

$$n + 1 = \{0, 1, \dots, n\}.$$

It is also apparent that $n + 1$ is different from any of $0, 1, \dots, n$. Here, $0 = \{\}$ and $n + 1 = n \cup \{n\}$ correspond to the axioms, and

$$\begin{aligned} 1 &= \{0\} \\ 2 &= \{0, 1\} \\ 3 &= \{0, 1, 2\} \end{aligned}$$

correspond to the theorems.

In mathematics and logic, new knowledge and information called theorems are created from axioms and postulates. Above, I wrote that creating new information based on specific procedures is called computation, but since theorems are new information, the operation of creating theorems can also be considered computation. In this sense, part of mathematics and logic can be considered the equivalent of computational thinking or the worldview of information.

So, what are the points where mathematics and logic differ from computational thinking? In mathematics and logic, there are cases where “you don’t have to know the specific procedure of computation.” If you are good at mathematics, you may have encountered cases where you can think of many ways to solve a problem. That’s right, there are generally multiple ways from axioms and postulates to theorems. Therefore, “a method based on a specific procedure” usually means that you know at least one of the several specific procedures. However, there are also cases where you only know that there exists a path consisting of specific procedures, but you don’t know what it is.

Mathematicians sometimes use the phrase “principally possible.” What it means to be computationally possible in principle is that it doesn’t matter how much time it takes, or how much memory or disk space is consumed. There is a big difference between what can be computed in principle and what can be computed in reality. Even if it is computationally possible in principle, if the computation time takes hundreds of years, or if the required memory size is larger than the number of atoms in the universe, the execution of the computation is practically impossible. In computational thinking, we are aware of whether the computation can be completed within a realistically acceptable time range, or whether the computation can be executed with the memory capacity and disk capacity that are realistically available. Memory capacity and disk capacity correspond to the maximum amount of data that must be recorded during the computation. Mathematics covers ideal and limit states, but computational thinking is limited to the range that can be calculated in practice.

In logic and mathematics, there is a proof method without calculating a certain value known as “reductio ad absurdum.” In reductio ad absurdum, we take the approach of starting the proof without touching the specific computation procedure, just by assuming the existence of a certain mathematical concept. For example, when trying to prove a certain proposition to be correct, if it is difficult to prove it directly, we consider what conclusion would be under the assumption that the proposition is wrong. Then, if a contradiction is derived by assuming that a certain proposition is wrong, it is concluded that the proposition was correct. Reductio ad absurdum is more straightforwardly called proof by contradiction. Here, a contradiction refers to the state that for a proposition P , “ P ” and “not P ” are held at the same time.

Reductio ad absurdum (proof by contradiction)

To prove a proposition P , instead of directly showing its correctness, it is a method of proving that P is true indirectly by showing that it contradicts when assuming “not P ”.

Now, let's say we want to prove the proposition “There are infinitely many prime numbers.” However, even using the fastest supercomputer in the universe, you cannot exhaustively generate an infinite number of primes. It takes an infinite amount of time to generate an infinite number of primes. So, it's hard to prove directly. Therefore, we assume that “there are only a finite number of primes, and all of these finite primes are $p_1, p_2, \dots, p_n$.” That is, the negation of the proposition to be proved has been assumed. The point here is that it does not provide any specific method or value for calculating $p_1, p_2, \dots, p_n$, it only assumes that such finite primes exist. Next, consider the new number $p = p_1 \times p_2 \times \dots \times p_n + 1$. Then, for all $i = 1, \dots, n$, it is clear that p is larger than p_i , and p cannot be divided by p_i because it always leaves a remainder of 1. The new number p actually meets the conditions of a prime number. Therefore, it contradicts the assumption that “there are only a finite number of primes, and the entire finite number of primes are $p_1, p_2, \dots, p_n$.” Therefore, it was concluded that “there are an infinite number of primes.” This is a famous proof by Euclid for the proposition that “there are an infinite number of primes.”

Not only *reductio ad absurdum* is often used in mathematics, but also it appears in detective novels. For example, suppose there is information that only the culprit would know in a murder case and the murder weapon has not been announced. If the suspect is assumed to be “not the culprit”, the suspect should not know about the weapon. Let's say the detective cornered the suspect and made him/her say “I couldn't possibly have a gun”. The weapon could have been a knife, and the only one who could specify a gun would be the culprit, so such reasoning becomes valid. In other words, if the suspect is assumed to be “not the culprit”, it contradicts the fact that the suspect cannot know that the weapon is a gun. Therefore, it is reasoned that the suspect must be the culprit.

In mathematics, computation was a powerful method to generate new information, but logic, represented by *reductio ad absurdum*, is also a powerful method to derive new conclusions and generate new information. Computation and logic, both powerful methods to generate new information, differ greatly in the point of whether computation calculates specific values in order according to specific procedures and reaches a conclusion. Then, the message conveyed in the book is that thinking and executing these specific procedures would be very useful in everyday life.

1.7.4 Difference from Philosophy

Philosophers often talk about things “in principle.” It's similar to mathematics.

We are going to discuss the thought experiment of “Chinese Room” by John Searle,² a typical argument of philosophers. This is a fictional experiment to argue that machines never “understand.” The setting of the Chinese Room is as follows.

- There is an American person in the room who does not understand Chinese.
- A piece of paper with a question written in Chinese is thrown into the room.
- The person does not understand Chinese but has a thick English manual written in English about Q&A in Chinese.
- The person follows the Chinese characters on the paper with a question in order (it's hard to find unreadable characters, but let's ignore that), works intensively according to the

instructions in the manual, and eventually gets a response in Chinese.

- The person writes down the response in Chinese on a piece of paper and returns it out of the room.

From the perspective of a person outside the Chinese Room, if you put in a question in Chinese, you get a response in Chinese. So, the room seems to understand Chinese. However, when you peek into the room, there is the person inside who does not understand Chinese at all, and of course, does not understand the content of the question or answer. Searle argues that AI programs are the same as the person inside, even if they appear to behave intelligently, they do not have intelligence.

There are various counterarguments from the AI side, one of which is about computational complexity and descriptive quantities. I don't think a manual that anticipates all questions can be created in the first place, but even if it could, how thick would it be? One AI researcher wrote a paper trying to estimate that amount and proved that even simple addition exceeds the total number of atoms in the universe, as it is difficult to measure the description of a general manual.

Searle does not have an estimate of the amount of work. He describes the thought experiment "in principle." The problem is discussed as if an answer can be created in a few minutes or at most a few hours, but it turns out that this is not the case.

In other words, the difference between philosophy and computational thinking is whether to consider the resources (memory and computation time) required for computation. Estimating this computational quantity is quite important for us to solve various problems in our daily lives.

The "Frame Problem" is also famous as a philosophical problem discovered by AI researchers, John McCarthy and Patrick Hayes. Originally, it was a problem that it is difficult to describe how the situation is changed by performing a certain action. Despite only a part of it being changed, a large amount of unchanged framework (frame) must be described, hence it was named as such. In other words, the Frame Problem means that when trying to describe an action, the effort to explicitly describe both what is changing and what is not changing, and to distinguish between them, may increase exponentially in description and inference.

If we don't think about specific computations, we don't feel the frame problem in our daily lives. That was why philosophers didn't notice it. However, when McCarthy tried to write the specific computation down as an AI program, he faced this problem. Even in the simple problem called "block world", where they were just rearranging blocks in early AI research, they couldn't write an AI program that realized specific computation rules.

One of the important things in computational thinking is to know that there are limits to computation. Even if the problem is describable in principle, there will be no end to the list of specific conditions. This may not have been a problem in mathematics or philosophy where it is finished with "it is possible in principle." This problem cannot be given a solution even by computational thinking, but it is important to recognize that there is such a hard problem. When talking with friends or family in daily life, you may find your understanding of "it is possible in principle" is at odds with theirs and you may not even realize it.

1.7.5 Differences from Programming

It seems that there are many people in the world who understand that computational thinking is the ability to write computer programs. But that's a mistake. There are many people who can program but cannot think computationally. Conversely, there are many people who can think computationally but don't need to write a program. So, what is the difference between being able to program and doing computational thinking?