高等学校新工科计算机类专业
“十三五”规划教材

信息科学类
专业英语

贾　晖　韩俊岗◎编　著

ENGLISH

FOR INFORMATION
SCIENCE

西安电子科技大学出版社
http://www.xduph.com

高等学校新工科计算机类专业
“十三五”规划教材

信息科学类专业英语

贾　晖　韩俊岗◎编　著

西安电子科技大学出版社
http://www.xduph.com

内 容 简 介

本书选取了信息科学前沿领域的相关内容，具有内容新颖、知识面广、趣味性强等特点。全书分为二十课，内容主要涉及云计算、物联网、三维打印、深度学习、5G 技术、数据挖掘、计算机视觉、大数据、嵌入式系统、人工智能、机器人、量子计算、GPU、生物信息学等领域的新技术、新进展，具有一定的广泛性和趣味性。读者通过阅读本书，可以大大拓展信息科学领域的知识面，提升学习兴趣。另外，本书每一课后都有针对考研和四、六级考试设计的长难句翻译和阅读理解题目，使读者在增长专业知识的同时提高英语的阅读和翻译技巧。

本书既可作为高等理工科院校信息科学与技术专业的科技英语课程教材，又可作为相关领域技术人员的工具书。

图书在版编目(CIP)数据

信息科学类专业英语 / 贾晖，韩俊岗编著. —西安：西安电子科技大学出版社，2019.9
ISBN 978-7-5606-5359-4

Ⅰ. ①信… Ⅱ. ①贾… ②韩… Ⅲ. ①信息技术—英语 Ⅳ. ①G202

中国版本图书馆 CIP 数据核字(2019)第 119591 号

策划编辑 陈 婷
责任编辑 张 梅 雷鸿俊
出版发行 西安电子科技大学出版社(西安市太白南路 2 号)
电 话 (029)88242885 88201467 邮 编 710071
网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com
经 销 新华书店
印刷单位 陕西天意印务有限责任公司
版 次 2019 年 9 月第 1 版 2019 年 9 月第 1 次印刷
开 本 787 毫米×1092 毫米 1/16 印张 14.5
字 数 266 千字
印 数 1～3000 册
定 价 35.00 元
ISBN 978-7-5606-5359-4 / G
XDUP 5661001-1

前　言

本书是在2011年出版的《信息科学类专业英语》(第二版)(西安电子科技大学出版社)的基础上，根据西安邮电大学教材建设的需要，经过全面的修改和补充完成的。增加了近年来信息科学领域的最新发展成果，包括人工智能、深度学习、三维打印、5G、数据挖掘、大数据分析、量子计算机、机器人、机器视觉等热门领域的相关文章。

科技文章的特点可以概括为客观性(objectivity)、准确性(accuracy)和精炼性(conciseness)。具体来说其语法特点如下：① 广泛使用被动语态；② 广泛使用名词形式；③ 省略句使用频繁；④ 形式主语、祈使句等句型使用频繁；⑤ 从句嵌套使用频繁，长难句较多；⑥ 后置定语使用较多等。这些特点导致本科生学了多年英语之后在参加工作或从事进一步科学研究时仍不能顺利地阅读专业文献和技术手册。而当今世界上科技资料的交流主要使用英语，特别是在软件工程、计算机科学与技术、网络工程以及电子、自动化等信息类专业领域，绝大部分专业资料都以英语的版本呈现。因此较高的专业英文资料的阅读能力能够推动高等院校学生的科研和就业。

本书的主要特点是内容新颖、知识面广、趣味性强，在扩大学生知识面的同时提高他们的英语阅读能力。本书不是其它已有专业课程所学内容的英文重现，而是围绕信息科学领域的前沿知识展开，适用于信息技术领域各个专业的学生学习和阅读。

专业英语涉及的知识面较广，教学难度较高。相关教师应具备较强的英语水平和丰富的信息科学领域的专业知识。教师可根据实际教学情况和课时安排选择本书的部分课程内容进行讲解。

本书中的所有文章均来自互联网。根据篇幅需要部分进行了节选，有些是多篇相关文章的综合。编者尽量保留了原作者的署名和文章出处，但仍有个别文章因查找不到信息而没有标注，在此向原作者致以诚挚的谢意。

由于编者水平有限，错误之处在所难免，希望广大师生和读者在使用中提出宝贵意见和建议(交流邮箱：jiahui@xupt.edu.cn)。

编　者

2019年4月

Contents

Lesson 1 Towards a Mathematical Science of Computation ····· 1

Lesson 2 A Brief Introduction of Cloud Computing ····· 10

Lesson 3 The Internet of Things ····· 23

Lesson 4 Real-World Applications of 3D Printing ····· 32

Lesson 5 Deep Learning: a Brief Guide for Practical Problem Solvers ····· 42

Lesson 6 Understanding 5G: Perspectives on Future Technological Advancements in Mobile ····· 57

Lesson 7 The Master Algorithm—Prologue ····· 68

Lesson 8 What Is Data Mining? ····· 82

Lesson 9 Smart Rooms ····· 92

Lesson 10 The Applications of Computer Vision ····· 104

Lesson 11 Big Data Analytics: Concepts, Technologies and Applications ····· 114

Lesson 12 Embedded Systems: A Primer ····· 124

Lesson 13 What Is Artificial Intelligence? ····· 132

Lesson 14 How to Make a Conscious Robot? ····· 143

Lesson 15 Quantum Computing ····· 152

Lesson 16 Utilisation of the GPU Architecture for HPC ····· 166

Lesson 17 Configurable Computing ····· 177

Lesson 18 Extreme Scale Computing ····· 198

Lesson 19 Introduction to Bioinformatics ····· 206

Lesson 20 An Introduction to MEMS ····· 215

Lesson 1 Towards a Mathematical Science of Computation

Text

Introduction

In this paper I shall discuss the prospects for a mathematical science of computation. In a mathematical science, it is possible to deduce from the basic assumptions, the important properties of the entities treated by the science. Thus, from Newton's law of gravitation and his laws of motion, one can deduce that the planetary orbits obey Kepler's laws.

What are the entities with which the science of computation deals?

What kinds of facts about these entities would we like to derive?

What are the basic assumptions from which we should start?

What important results have already been obtained?

How can the mathematical science help in the solution of practical problems?

I would like to propose some partial answers to these questions. These partial

---

The author of this article is John McCarthy (September 4, 1927 - October 24, 2011). He was an American computer scientist and cognitive scientist. McCarthy was one of the founders of the discipline of artificial intelligence. He coined the term "artificial intelligence" (AI), developed the Lisp programming language family, significantly influenced the design of the ALGOL programming language, popularized timesharing, and was very influential in the early development of AI.

McCarthy spent most his career at Stanford University. He received many accolades and honors, such as the 1971 Turing Award for his contributions to the topic of AI, the United States National Medal of Science, and the Kyoto Prize.

answers suggest some problems for future work. Firstly, I shall give some very sketchy general answers to the questions. Then I shall present some recent results on three specific questions. Finally, I shall try to draw some conclusions about practical applications and problems for future work.

What Are The Entities with Which Computer Science Deals?

These are problems, procedures, data spaces, programs representing procedures in particular programming languages and computers.

A problem is defined by the criterion which determines whether a proposed solution is accepted. One can understand a problem completely without having any method of solution. Procedures are usually built up from elementary procedures. What these elementary procedures may be and how more complex procedures are constructed from them, is one of the first topics in computer science. This subject is not difficult to understand since there is a precise notion of a computable function to guide us and computability relative to a given collection of initial functions is easy to define.

Procedures operate on members of certain data spaces and produce members of other data spaces, using in general still other data spaces as intermediates. A number of operations are known for constructing new data spaces from simpler ones, but there is as yet no general theory of representable data spaces comparable to the theory of computable functions. [1]

Programs are symbolic expressions representing procedures. The same procedure may be represented by different programs in different programming languages. We shall discuss the problem of defining a programming language semantically by stating what procedures the programs represent. [2] As for the syntax of programming languages, the rules which allow us to determine whether an expression belongs to the language have been formalized, but the parts of the syntax which relate closely to the semantics have not been so well studied. The problem of translating procedures from one programming language to another has been much studied, and we shall try to give a definition of the correctness of the translation.

Computers are finite automata. From our point of view, a computer is defined by the effect of executing a program with given input on the state of its memory and on its outputs. Computer science studies the various ways elements of data spaces are

represented in the memory of the computer and how procedures are represented by computer programs. From this point of view, most of current work on automata theory is beside the point.

What Kinds of Facts about Problems, Procedures, Data Spaces, Programs and Computers Would We Like to Derive?

Primarily, we would like to be able to prove that given procedures solve given problems. However, proving this may involve proving a whole host of other kinds of statement, such as:

1. Two procedures are equivalent, i. e. compute the same function.

2. A number of computable functions satisfy a certain relationship, such as an algebraic identity or a formula of the functional calculus.

3. A certain procedure terminates for certain initial data or for all initial data.

4. A certain translation procedure correctly translates procedures between one programming language and another.

5. One procedure is more efficient than another equivalent procedure in the sense of taking fewer steps or requiring less memory.

6. A certain transformation of programs preserves the function expressed but increases the efficiency.

7. A certain class of problems is unsolvable by any procedure, or requires procedures of a certain type for its solution.

What Are The Axioms and Rules of Inference of a Mathematical Science of Computation?

Ideally, we would like a mathematical theory in which every true statement about procedures would have a proof, and preferably a proof that is easy to find, not too long and easy to check. In 1931, Gödel proved a result, one of whose immediate consequences is that there is no complete mathematical theory of computation. Given any mathematical theory of computation there are true statements expressible in it which do not have proofs.[3] Nevertheless, we can hope for a theory which is adequate for practical purposes, like proving that compilers work. The unprovable statements tend to be of a rather sweeping character, such as that the system itself is consistent.

It is almost possible to take over one of the systems of elementary number theory

such as that given in Mostowski's book "Sentences Undecidable in Formalized Arithmetic", since the content of a theory of computation is quite similar. Unfortunately, this and similar systems were designed to make it easy to prove meta-theorems about the system, rather than to prove theorems in the system. As a result, the integers are given such a special role that the proofs of quite easy statements about simple procedures would be extremely long.

Therefore, it is necessary to construct a new, though similar, theory in which neither the integers nor any other domain, (e. g. strings of symbols) are given a special role. Namely, an integer-free formalism for describing computations has been developed and shown to be adequate in the cases where it can be compared with other formalisms. Some methods of proof have been developed, but there is still a gap when it comes to methods of proving that a procedure terminates. The theory also requires extension in order to treat the properties of data spaces.

What Important Results have been Obtained Relevant to a Mathematical Science of Computation?

In 1936 the notion of a computable function was clarified by Turing, and he showed the existence of universal computers that, with an appropriate program, could compute anything computed by any other computers.[4] All our stored program computers, when provided with unlimited auxiliary storage, are universal in Turing's sense. In some subconscious sense even the sales departments of computer manufacturers are aware of this, and they do not advertise magic instructions that cannot be simulated on competitors machines, but only that their machines are faster, cheaper, have more memory or are easier to program.

The second major result was the existence of classes of unsolvable problems. This keeps all but the most ignorant of us out of certain Quixotic enterprises such as trying to invent a debugging procedure that can infallibly tell if a program being examined will get into a loop.[5] Later, we shall discuss the relevance of the results of mathematical logic on creative sets to the problem of whether it is possible for a machine to be as intelligent as a human. In my opinion, it is very important to build a firmer bridge between logic and recursive function theory on the one side, and the practice of computation on the other.

Much of the work on the theory of finite automata has been motivated by the hope of applying it to computation. I think this hope is mostly in vain because the fact of finiteness is used to show that the automation will eventually repeat a state. However, anyone who waits for an IBM 7090 to repeat a state, solely because it is a finite automation is in for a very long wait.

How Can a Mathematical Science of Computation Help in the Solution of Practical Problems?

Naturally, the most important applications of a science cannot be foreseen when it is just beginning. However, the following applications can be foreseen:

1. At present, programming languages are constructed in a very unsystematic way. A number of proposed features are invented, and then we argue about whether each feature is worth its cost. A better understanding of the structure of computations and of data spaces will make it easier to see what features are desirable.

2. It should be almost possible to eliminate debugging. Debugging is the testing of a program on cases one hopes are typical, until it seems to work. This hope is frequently vain. Instead of debugging a program, one should prove that it meets its specifications, and this proof should be checked by a computer program. For this to be possible, formal systems are required in which it is easy to write proofs. There is a good prospect of doing this, because we can require the computer to do much more work in checking each step than a human is willing to do. Therefore, the steps can be bigger than with present formal systems.

Vocabulary

1. mathematical science of computation 数学[化]的计算科学，指在计算科学中利用数学方法
2. sketchy adj. 粗略的
3. prospect n. 景象，景色，前景，前途视野；展望；眺望；预期；指望
   vt. ① 勘探；勘察；找矿 ② 对……进行仔细调查
4. deduce vt. 推论，演绎出
5. semantics n. 语义学

6. automata　n. 自动操作，自动控制
7. subconscious　adj. 下意识的；潜意识的
8. infallible　adj. ① 不会犯错误的，无过失的 ② 极准确的，极精确的 ③ 绝对可靠的，万无一失的，永远有效的
9. Quixotic　adj. 堂吉诃德式的，愚侠的，不切实际的
10. sweeping　adj. ① 包罗万象的，一扫而光的 ② 笼统的，泛泛的，一概而论的 ③ 影响广泛的；大范围的

Key Sentences

[1] A number of operations are known for constructing new data spaces from simpler ones, but there is as yet no general theory of representable data spaces comparable to the theory of computable functions.

人们知道有大量的方法可将简单数据空间构建成新的数据空间，但至今还没有任何描述数据空间的通用理论能够与可计算函数理论相媲美。

[2] We shall discuss the problem of defining a programming language semantically by stating what procedures the programs represent.

我们将讨论通过说明程序描述怎样的算法来从语义学的角度定义编程语言所面临的问题。

[3] In 1931, Gödel proved a result, one of whose immediate consequences is that there is no complete mathematical theory of computation. Given any mathematical theory of computation there are true statements expressible in it which do not have proofs.

在 1913 年，歌德尔证明了一个结论，该结论的一个直接推论就是完备的数学计算理论是不存在的。给定任何计算理论，在这个理论中必定存在可以表达为真的命题，但没有证据可证明。

[4] In 1936 the notion of a computable function was clarified by Turing, and he showed the existence of universal computers that, with an appropriate program, could compute anything computed by any other computer.

在 1936 年，图灵澄清了可计算函数的概念，并且证明通过适当的程序，任何一台计算机都可以完成任何其它计算机能够做的计算。

[5] The second major result was the existence of classes of unsolvable problems.

This keeps all but the most ignorant of us out of certain Quixotic enterprises such as trying to invent a debugging procedure that can infallibly tell if a program being examined will get into a loop.

第二个主要结论是存在一类不可解的问题。这个结论令我们大多数人，除了那些最无知的人之外，不再幻想那些堂吉诃德式的事业，例如试图发明能够绝对无误地检验一个程序是否会陷入死循环的查错过程。

Single Choice Questions

(1) According to the text, the entities that Computer Science deals with may not include( ).

A. problems

B. procedures

C. programming languages

D. computers

(2) According to the text, what is one of the first topics in computer science? ( )

A. What these elementary procedures may be?

B. How more complex procedures are constructed from elementary procedures?

C. How to understand a problem completely?

D. What elementary procedures may be and how more complex procedures are constructed from them?

(3) Which of the following statements describes the relationship between procedures and programming languages? ( )

A. The same procedure may be represented by different programs in different programming languages.

B. The problem of translating procedures from one programming language to another is easy.

C. The problem of defining a programming language semantically can be solved by stating what procedures the programs represent.

D. Computer science must study how procedures are represented by computer programs.

(4) Which of the following statements is wrong for describing the relationship of

procedure to data space and programming languages? ( )

A. Procedures operate on members of certain data spaces and produce members of other data spaces.

B. Programs are symbolic expressions representing procedures.

C. The same procedure may be represented by different programs in different programming languages.

D. Programming languages are constructed from different procedure.

(5) According to the text, which of the following statements is wrong about computer? ( )

A. Computers are finite automata.

B. A computer is defined by the effect of executing a program with given input on the state of its memory and on its outputs.

C. Much of the work on the theory of finite automata is fruitful when it is applied to computation.

D. Most of the current work on automata theory is beside the pointof computer science.

(6) According to the text of section 4, which of the following statements is wrong? ( )

A. We would like a mathematical theory in which every true statement about procedures would have an easy short proof.

B. There is no complete mathematical theory of computation.

C. We hope for a theory which is adequate for practical purposes, like proving that compilers work.

D. An integer-free formalism for describing computations has been developed and can be used to prove a procedure terminates.

(7) What important results have been obtained relevant to a mathematical science of computation? ( )

A. The notion of a computable function was clarified by Turing.

B. Much of the work on the theory of finite automata has been done.

C. Existence of classes of unsolvable problems.

D. A machine to be as intelligent as a human.

(8) What kind of work does the author suggest not to solve by mathematics science of computation? ( )

A. Confirming each feature of a programming language is worth its cost.

B. Proving a program meets its specifications, instead of debugging it.

C. Checking each step of program by human.

D. Better understanding of the structure of computations and of data spaces.

Questions

1. What is the object of computer science?
2. How can mathematics help computer science?

Lesson 2 A Brief Introduction of Cloud Computing

Text

Numerous surveys report that Cloud Computing will be a top 10 technology that enterprise business managers need to be aware of for 2010. Not that you can escape the marketing and information published about this latest super hyped topic. Much of the message focuses on Cloud Computing as a lower cost delivery model for IT services. This may or may not be true.

What Is Cloud Computing?

We see Cloud Computing as a computing model, not a technology. In this model "customers" plug into the "cloud" to access IT resources which are priced and provided "on-demand". Essentially, IT resources are rented and shared among multiple tenants much as office space, apartments, or storage spaces are used by tenants. Delivered over an Internet connection, the "cloud" replaces the company data center or server providing the same service. Thus, Cloud Computing is simply IT services sold and delivered over the Internet. Refer to section of Types of Cloud Computing.

Cloud Computing vendors combine virtualization (one computer hosting several "virtual" servers), automated provisioning (servers have software installed automatically) and Internet connectivity technologies to provide the service.[1] These are not new technologies but a new name applied to a collection of older (albeit updated) technologies that are packaged, sold and delivered in a new way.

A key point to remember is that, at the most basic level, your data resides on someone else's server(s). This means that most concerns (and there are potentially hundreds) really come down to trust and control issues. Do you trust them with your data?

The Economics

Economies of scale and skill drive Cloud Computing economics. As with rented Real Estate, the costs of ownership are pooled and spread among all tenants of the multi-tenant Cloud Computing solution. Consequentially, acquisition costs are low but tenants never own the technology asset and might face challenges if they need to "move" or end the service for any reason. [2]

Something that is often overlooked when evaluating Cloud Computing costs is the continued need to provide LAN services that are robust enough to support the Cloud solution. These costs are not always small, for example, if you have 6 or more workstation computers, you will probably need to continue to maintain a server in a domain controller role (to ensure name resolution), at least one switch (to connect all of the computers to each other and the router), one or more networked printers and the router for the Internet connection.

What Do I Need to Use Cloud Computing?

All that is really needed to acquire and use Cloud Computing solutions is a credit card (or other payment method) and a LAN with an Internet connection robust enough to support the Cloud delivered service. These two requirements are deceptively simple.

From a technical point of view the biggest challenge for businesses, particularly SMBs, may be the need for an appropriately robust LAN infrastructure and Internet connection. Typically, Internet access is provided by a single commercial service ISP provider through a single port on a router. A characteristic of this type installation is that all of the computers connecting through the LAN share the Internet bandwidth equally. This can quickly become an issue.

For example, Verizon FiOS Internet 15/2 (down/up) service might have a measured speed of 14420/1867 Kbps. This would seem to be plenty of speed. However, suppose a business had 5 computers using a Cloud solution and sending data to the cloud for processing. The bandwidth available to each computer would be 373Kbps (up 1867/5). That is about 46 (8 bit) characters per second to the cloud application and does not include any communication or application data. The cloud

solution might not work or responses so slow as to be unacceptable. It isn't the download speed that becomes a limit, but the upload speed.

The on-demand nature of Cloud Computing presents a dilemma: the on-demand model includes a self-service interface that allows users to self-provision services (for example storage).[3] This empowers users but can make services too easy to acquire and consume. To quote an IT administrator "People could care less about policies. They want what they want when they want it. They don't involve IT."

Consider the faculty member at the University of Massachusetts who quietly (without anyone's knowledge) used a cloud service to back up 20 GB of data each night over the Internet bringing the school LAN to its knees. How management controls Cloud Computing is unique to each organization and is an IT Governance issue.

Conclusion

We are often told particularly by vendors and evangelists, You don't like Outsourced or Cloud Computing solutions. This is simply not true. Outsourcing and using third parties for service can be very helpful to clients. However, we do not think that these solutions are appropriate or effective in every situation or for every organization.

Types of Cloud Computing

SaaS (Software as a Service)

It is the most widely known and widely used form of cloud computing. It provides all the functions of a sophisticated traditional application to many customers and often thousands of users, but through a Web browser, not a "locally-installed" application.[4] Little or no code is running on the users, local computer and the applications are usually tailored to fulfill specific functions. SaaS eliminates customer worries about application servers, storage, application development and related, common concerns of IT.

Highest-profile examples are Salesforce. com, Google's Gmail and Apps, instant messaging from AOL, Yahoo, Google, VoIP from Vonage and Skype.

PaaS (Platform as a Service)

It delivers virtualized servers on which customers can run existing applications or