研究(分野)紹介://環境双模倣.プログラミング言語理論.理論計算機科学.計算機科学

住井 英二郎（東北大学 大学院 情報科学研究科 教授, 日本学術会議 若手アカデミー 幹事）

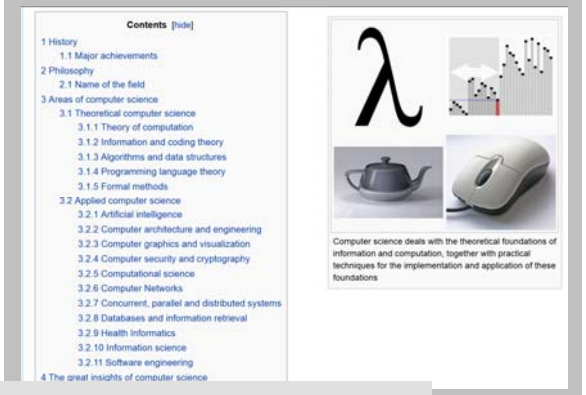
計算機科学ってどんな分野？

おなじみの「あの表」より：

系	分野	分科	****	細目名	分割	キーワード（記号）
総合系	情報学	情報学基礎	1001	情報学基礎理論		(1)計算理論、(2)オートマトン理論・形式言語理論、(3)プログラム理論、(4)計算量理論、(5)アルゴリズム理論、(6)符号系、(7)組織構造、(8)計算論的学習理論、(9)量子計算理論、(10)数理論理学、(11)情報理論、(12)符号理論
			1002	数理情報学		(1)最適化理論、(2)数理ファイナンス、(3)数理システム理論、(4)システム制御理論、(5)システム分析、(6)システム方法論、(7)システムモデリング、(8)システムシミュレーション、(9)組み合わせ最適化、(10)待ち行列理論
			1003	統計科学		(1)調査・実験計画、(2)多変量解析、(3)時系列解析、(4)統計的パターン認識、(5)統計的推測、(6)統計計算・コンピュータ支援統計、(7)統計的予測・制御、(8)モデル化・選択、(9)医薬生物・ゲノム統計解析、(10)行動計量分析、(11)官庁・環境統計、(12)統計教育、(13)統計的品質管理、(14)統計的学習理論、(15)社会調査の計画と解析、(16)データサイエンス、(17)保険数理
		計算基盤	1101	計算機システム		(1)計算機アーキテクチャ、(2)組込システム、(3)しなり制御技術、(4)リコンフィギュラブルシステム、(5)実環境アーキテクチャ、(6)超高速計算技術、(7)ハード・ソフト協調設計、(8)組み込みシステム
			1102	ソフトウェア		(1)プログラミング言語、(2)プログラミング方法論、(3)プログラミング実践技術論、(4)並列・分散処理、(5)オペレーティングシステム、(6)実環境システム、(7)保険生技術、(8)ソフトウェアセキュリティ、(9)クラウドコンピューティング基盤、(10)ソフトウエア工学、(11)仕様記述・検証、(12)開発理論、(13)開発管理
			1103	情報ネットワーク		(1)ネットワークアーキテクチャ、(2)ネットワークプロトコル、(3)インターネット、(4)モバイルネットワーク、(5)オーバレイネットワーク、(6)センサーネットワーク、(7)トラフィックエンジニアリング、(8)ネットワーク構成・運用・管理・計測技術、(9)ユビキタスコンピューティング、(10)サービス情報基盤技術、(11)情報家電システム
			1104	マルチメディア・データベース		(1)データモデル、(2)関係データベース、(3)データベースシステム、(4)マルチメディア情報基礎、(5)マルチメディア情報処理、(6)マルチメディア情報表現、(7)マルチメディア情報生成、(8)情報検索、(9)構造化文書、(10)コンテンツ流通・管理、(11)地理情報システム、(12)メタデータ、(13)ビッグデータ分析・活用
			1105	高性能計算		(1)並列処理、(2)分散処理、(3)グリッド・クラウドコンピューティング、(4)数値解析、(5)可視化、(6)コンピュータグラフィクス、(7)高性能計算アプリケーション
			1106	情報セキュリティ		(1)アクセス制御、(2)個人識別、(3)暗号、(4)認証、(5)セキュリティ評価・監査、(6)マルウェア対策、(7)ネットワークセキュリティ、(8)不正アクセス対策、(9)ソフトウェア保護、(10)プライバシー保護、(11)情報フィルタリング、(12)ディジタルフォレンジクス、(13)バイオメトリクス、(14)耐タンパー技術
		人間情報学	1201	認知科学		(1)感覚・知覚・学習、(2)記憶・記憶・創発、(3)思考・推論・問題解決、(4)感覚・知覚・感性、(5)感情・情動・行動、(6)認知心理学、(7)光認知心理学、(8)認知哲学、(9)脳認知科学、(10)認知言語学、(11)行動言語決定論、(12)認知工学、(13)認知考古学、(14)認知モデル、(15)社会性、(16)ふと心理

英語版Wikipedia※1より：

※1 英語版は一般的に日本語版よりは信頼できます



Contents [hide]

1 History
1.1 Major achievements
2 Philosophy
2.1 Name of the field
3 Areas of computer science
3.1 Theoretical computer science
3.1.1 Theory of computation
3.1.2 Information and coding theory
3.1.3 Algorithms and data structures
3.1.4 Programming language theory
3.1.5 Formal methods
3.2 Applied computer science
3.2.1 Artificial intelligence
3.2.2 Computer architecture and engineering
3.2.3 Computer graphics and visualization
3.2.4 Computer security and cryptography
3.2.5 Computational science
3.2.6 Computer Networks
3.2.7 Concurrent, parallel and distributed systems
3.2.8 Databases and information retrieval
3.2.9 Health Informatics
3.2.10 Information science
3.2.11 Software engineering
4 The great insights of computer science

Computer science deals with the theoretical foundations of information and computation, together with practical techniques for the implementation and application of these foundations

"Computer Science is no more about computers than astronomy is about telescopes."
- attributed to E. W. Dijkstra

× 「WordやExcelの研究ですか？」（医者の知人）
× 「テレビ壊れたから直して」（ある先生のお父上）

- 分野も人間（研究者）も若い（20代准教授、30代教授、高校生等）
- 国際会議予稿集論文中心（2段組10〜30頁、査読採択率7〜30%）
- インフォーマル（スーツ・ネクタイ皆無）
- 残念ながら日本は一部を除き弱い（Journal of the ACMの日本からの論文は1954年の創刊より30件のみ）

理論計算機科学：計算（情報処理）に関する数学・論理学に基づく研究分野（計算理論、アルゴリズム、**プログラミング言語理論**等）

プログラミング言語理論：計算を記述する記号的体系に関する研究分野（CやJava等に限らず「**λ計算**」「**π計算**※2」等）

※2 円周率とは無関係です

λ計算[Church 1936]：関数（適用）に基づく計算体系

$(\lambda x.\ M)\ N \rightarrow [N/x]M$　xを受け取りMを返す関数λx. Mを、引数Nに適用

π計算[Milner et al. 1992]：プロセス（通信）に基づく計算体系

$c!(M).P \mid c?(x).Q \rightarrow P \mid [M/x]Q$　チャンネルcにメッセージMを送信してPに遷移するプロセスc!(M).Pが、cからxを受信してQに遷移するプロセスc?(x).Qと通信

自分の研究

プログラム等価性（二つのプログラムの振る舞いが等しいこと）を証明するための理論（**環境双模倣**）

- 「式の等しさ」はあらゆる数理科学の基本($x^2+y^2=z^2$, $E=mc^2$, etc.)
- 古典的理論（表示的意味論[Scott 1970]や双模倣[Park 1981]）は高度な機能（ポインタやクロージャ等）に対し不完全ないし不健全
- 環境双模倣は**初の健全・完全かつ初等的**な理論

（JACM採録、MSR賞・IBM科学賞・学振賞等受賞）

定義の概略（π計算の場合）：Xが環境双模倣であるとは、すべての(P,Q,E)∈Xに対して以下が成り立つことである。

(1) PがメッセージMをチャンネルcに送信してP'に遷移するならば、QもメッセージNをcに送信してあるQ'に遷移でき、Eに(M,N)を加えた環境E'に対して(P',Q',E')∈X

(2) 環境Eから合成できる任意のメッセージの組(M,N)に対し、PがMをチャンネルcから受信してP'に遷移するならば、QもNをcから受信してあるQ'に遷移でき、(P',Q',E)∈X

(3) PがP'に内部遷移するならば、QもあるQ'に内部遷移でき、(P',Q',E)∈X

(4) 1〜3の逆も成り立つ

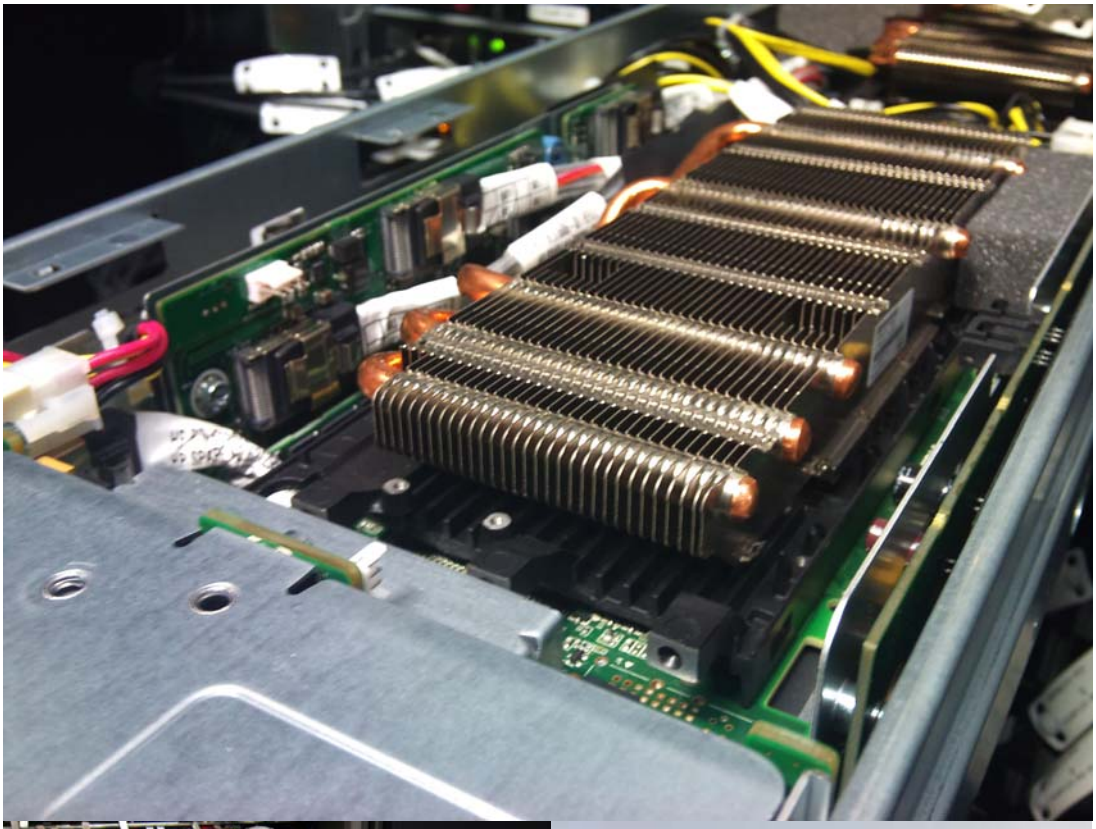
定理（健全性・完全性）：ある環境双模倣Xに対して(P,Q,∅)∈Xならば、PとQは等価である。逆も成り立つ。

何の「役に立つ」のか？

- 情報処理システムの**不具合を未然に防止**（テストやレビューより確実、自分の研究ではないが分野として**実用ソフトウェアで複数の実績**）
- しかし、果たして短期的に「役に立つ」ことだけが重要なのか？（**λ計算は約80年、数理論理学は数千年の歴史**）
- 本当に「役に立つ」ためには**心理学や社会科学との連携**が必要では

ユークリッドからインターネットそしてスーパーコンピュータへ

住井 英二郎
東北大学大学院情報科学研究科 教授



上と左下：省エネ世界１位（2013＆14年GREEN500）、速度世界５位（2011年TOP500）等を獲得したスーパーコンピュータ「TSUBAME」シリーズ（東工大）／右下：速度世界１位（2011年TOP500）獲得のスーパーコンピュータ「京」（理研）

パソコン少年

小さいころから算数や数学が好きでした。兄がやっていた公〇式を見て、親にせがんで私もやらせてもらいました（小学四年生ぐらいで飽きてやめてしまいましたが）。

小学一年生のとき、兄の中学入学祝いにパソコンを買うというので、一緒にN社のショールームやパソコンショップに連れていってもらいました。今と違って、当時のパソコンは自分でプログラムを書いて使うのが普通でした。子供なので大したことはありませんでしたが、ショールームに通い詰めて、ごく簡単なプログラムは書けるようになりました。

家にパソコンが来てからも、親に「パソコンは一日30分まで」と制限されてしまったので、紙の上にプログラムを書いて、（紙の上で）実行したりしていました。

当時の一般的な家庭用パソコンはCPUが8ビット・4メガヘルツ程度、メモリは16キロバイト程度、グラフィックス画面は256×192ドット程度でした。

現在のPCはCPUが64ビット・2ギガヘルツ（＝2千メガヘルツ）、メモリは2ギガバイト（＝2百万キロバイト）、画面は1920×1080ドットぐらいでしょうか。それぞれ何倍に増えたか計算してみてください。

フラクタル図形

中学校入学時に私も最新の16ビットパソコンを買ってもらい、フロッピーディスクが使えるようになったので（それまでは使わせてもらえず、カセットテープにセーブしていました）、それまでより難しいプログラムも書けるようになってきました（依然として一日30分の制限はありましたが）。

特に、パソコン雑誌で見かけた「フラクタル図形」（自己相似形）を描くプログラムに感動し、自分も書きたくなりました。

しかし、フラクタル図形を描くプログラムは「再帰」という仕組みを使って書かれることが多く、当時の「BASIC」というプログラミング言語では書くことができませんでした（他の言語のコンパイラやインタプリタは高価で買えませんでした）。

仕方がないので「再帰」のかわりに「スタック」と呼ばれる仕組みをBASICの上に作り、それを用いて「再帰」をシミュレートすることにより、何とかフラクタル図形を描くプログラムを書くことに成功しました。

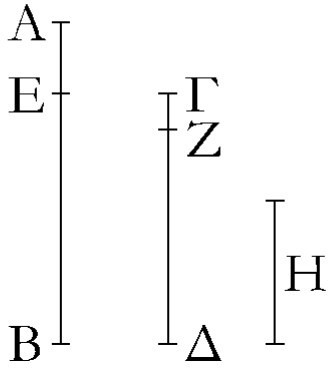
ユークリッドの「原論」

それからしばらく、高校生の間はパソコンよりも純粋数学に興味

Eijiro Sumii
●1975年東京都生まれ。東京大学理学部情報科学科卒業。同大学院情報理工学研究科コンピュータ科学専攻、ペンシルバニア大学を経て2005年東北大学へ。

β′.

Δύο ἀριθμῶν δοθέντων μὴ πρώτων πρὸς ἀλλήλους τὸ μέγιστον αὐτῶν κοινὸν μέτρον εὑρεῖν.



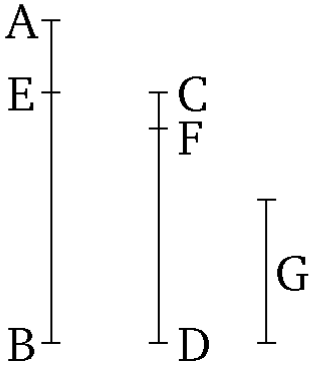
Ἔστωσαν οἱ δοθέντες δύο ἀριθμοὶ μὴ πρῶτοι πρὸς ἀλλήλους οἱ AB, ΓΔ. δεῖ δὴ τῶν AB, ΓΔ τὸ μέγιστον κοινὸν μέτρον εὑρεῖν.

Εἰ μὲν οὖν ὁ ΓΔ τὸν AB μετρεῖ, μετρεῖ δὲ καὶ ἑαυτόν, ὁ ΓΔ ἄρα τῶν ΓΔ, AB κοινὸν μέτρον ἐστίν. καὶ φανερόν, ὅτι καὶ μέγιστον· οὐδεὶς γὰρ μείζων τοῦ ΓΔ τὸν ΓΔ μετρήσει.

Εἰ δὲ οὐ μετρεῖ ὁ ΓΔ τὸν AB, τῶν AB, ΓΔ ἀνθυφαιρουμένου ἀεὶ τοῦ ἐλάσσονος ἀπὸ τοῦ μείζονος λειφθήσεταί τις ἀριθμός, ὃς μετρήσει τὸν πρὸ ἑαυτοῦ. μονὰς μὲν γὰρ οὐ λειφθήσεται· εἰ δὲ μή, ἔσονται οἱ AB, ΓΔ πρῶτοι πρὸς ἀλλήλους· ὅπερ οὐχ ὑπόκειται. λειφήσεταί τις ἄρα ἀριθμὸς, ὃς μετρήσει τὸν πρὸ ἑαυτοῦ. καὶ ὁ μὲν ΓΔ τὸν BE μετρῶν λειπέτω ἑαυτοῦ ἐλάσσονα

Proposition 2

To find the greatest common measure of two given numbers (which are) not prime to one another.



Let AB and CD be the two given numbers (which are) not prime to one another. So it is required to find the greatest common measure of AB and CD.

In fact, if CD measures AB, CD is thus a common measure of CD and AB, (since CD) also measures itself. And (it is) manifest that (it is) also the greatest (common measure). For nothing greater than CD can measure CD.

But if CD does not measure AB then some number will remain from AB and CD, the lesser being continually subtracted, in turn, from the greater, which will measure the (number) preceding it. For a unit will not be left. But if not, AB and CD will be prime to one another [Prop. 7.1]. The very opposite thing was assumed. Thus,

図１：ユークリッド「原論」（紀元前 300 年頃）第 7 巻命題 2 より。「世界最古のアルゴリズム」と言われる「ユークリッドの互除法」。現代的に述べると「二つの正整数 m, n の最大公約数を求めるためには、大きいほうの整数を、小さいほうの整数で割っていく。これを互いに繰り返し、小さいほうの整数が大きいほうの整数を割り切ったならば、それが元の m, n の最大公約数となる。」　ただし m, n といった「変数」は 16 世紀末に考えられたため、「原論」では変数ではなく線分を用いて整数を表している。プログラム（例えば C 言語の関数）で書くと gcd(m,n){return(n==0?m:gcd(n,m%n));}のようになる。

```
> cd openssl-1.0.0e/crypto/
> grep euclid bn/bn_gcd.c
static BIGNUM *euclid(BIGNUM *a, BIGNUM *
b);
t=euclid(a,b);
static BIGNUM *euclid(BIGNUM *a, BIGNUM *
b)
> grep gcd rsa/*.c
rsa/rsa_chk.c:
r = BN_gcd(m, i, j, ctx);
rsa/rsa_gen.c:
if (!BN_gcd(r1,r2,rsa->e,ctx)) goto err;
rsa/rsa_gen.c:
if (!BN_gcd(r1,r2,rsa->e,ctx)) goto err;
```

図２：HTTPS など、インターネット上の安全な暗号化通信を実現するプログラムの一部。実際に「ユークリッド(Euclid)の互除法」により、巨大な整数(big num)の最大公約数(greatest common divisor)を計算している。

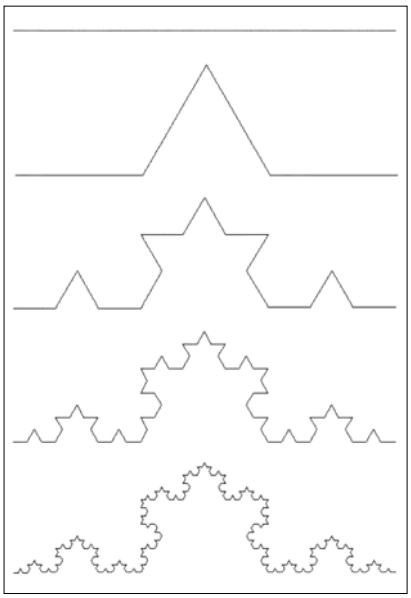


図 3：「フラクタル図形」の一種「コッホ曲線」の生成過程。一つの線分（一番上）を三等分し、真ん中の線分を、それを一辺とする正三角形の他の二辺で置き換える（二番目）。この「三等分して真ん中を置き換える」操作を、各線分に対して繰り返す（三番目以降）。プログラムで書くと

```
koch(x) {
  if (x<10) forward(x);
  else {
    koch(x/3);
    left(60); koch(x/3);
    right(120); koch(x/3);
    left(60); koch(x/3);
  }
}
```

ただし forward(x)は「前に長さ x の線分を描いて進む」、left(60)は「左に 60°向きを変える」、right(120)は「右に 120°向きを変える」動作を表す。

スパコン世界１位

時代は一気に下って２００９年１１月、「２位じゃダメなんでしょうか」で有名な「事業仕分け」により、皮肉にも「スーパーコンピュータ」などという用語が世間やマスコミをにぎわせる状況となりました。

その後、まさにその事業仕分けの対象となった「京」がスーパーコンピュータの世界ランキング「ＴＯＰ５００」でダントツ１位を獲得、次のＴＯＰ５００でも２回連続で１位を獲得しました。

また、同じく日本のスーパーコンピュータ「ＴＳＵＢＡＭＥ２」も２回連続で５位を獲得、両者は他にもスーパーコンピュータ関連の賞を総なめしています。

が移り、「ブルバキ」という２０世紀フランスの数学者グループが書いた「数学原論」という本を読もうとして最初の１章で挫折したり、数学オリンピックの一次予選に落ちたりして（東京の国立高校だったので、周りには本選金メダリストも何人かいました）、自分には数学の才能がない（笑）と考えるようになりました。

ちなみにブルバキの「数学原論」は２０世紀の本ですが、紀元前古代ギリシャの数学者ユークリッドが編纂した「原論」という本に因んだタイトルです。ユークリッドの「原論」には「世界最初のアルゴリズム」と呼ばれる「ユークリッドの互除法」が記されており（図１）、現代のインターネットにおける安全な暗号化通信に欠かせないアルゴリズムとなっています（図２）。