コンピュータに証明できること・できないこと

照井一成

◎京都大学数理解析研究所

コンピュータ将棋からコンピュータ証明へ

今、人工知能がブームである。機械翻訳や画像認識における成功はもとより、チェスや将棋においてもコンピュータ棋士の活躍には目覚ましいものがある。人工知能は、やがていつか数学にも進出してくるのだろうか？人間数学者とコンピュータ数学者が競い合う時代がやってくるのだろうか？そんな可能性について考えてみたい。あまり大したことは言えないが、数理論理学の立場からの一意見として参考にしていただければ幸いである。

まず、もっとも一般的なのは次のような意見であろう。

「コンピュータが数学者になんてなれるわけがない。なぜなら機械的計算しかできないコンピュータには、数学にとってもっとも大切な創造力や構想力を持ちえないからだ」

だがどうして「創造力」や「構想力」が機械的計算によっては実装しえないと、はなから決めつけることができるのだろうか？実際将棋でいえば、現代のコンピュータ棋士は、「大局観」なる神秘的特性を

備えた人間棋士に十分太刀打ちできる。この事実を忘れてはならない。

「いや将棋と数学では全然違うだろう。一方はルールにのっとったゲームにすぎないが、他方は無限の展開可能性を秘めた創造的行為なのだから」

そんなことはない。数学だって論理というルールに縛られている。どんなにすごい定理を“証明”したって、それが論理法則に従っていなければ、反則負けだ。確かに新たな概念を定義したり、新しい理論を構築したりといった創造的側面が数学にとって本質的なことは否定しないが、まずはもう少し非創造的な部分に問題を限定することもできるだろう。すなわち、既存理論に限って、一定の公理群から定理を証明すること。そういった**コンピュータ証明**の可能性に思いを馳せることは十分に意義がある。実際、人間数学者には証明しえなかった未解決問題をコンピュータが解決した事例もある（ブール代数における**ロビンス予想**の解決 [1]。ただしこの一例をもってコンピュータ証明の現状を過大評価すべきではない）。

コンピュータ証明の可能性を計る 1 つの目安は、「定理を証明する」ことと「将棋を指す」ことがどの程度近いか、である。もしも十分に近いならば、コンピュータ証明はコンピュータ将棋と同じくらい躍進する潜在力を秘めているといってよいだろう。

実数だけなら大丈夫、自然数や集合は難しい

では、どんなことならコンピュータに証明できるのか？具体例を与えるために、次のような言葉（記号）だけを使って作られる文を考えよう。

- 有理係数の多変数多項式

- 等号、不等号
- $\wedge$（かつ）、$\rightarrow$（ならば）、$\forall$（すべて）、$\exists$（存在）などの論理記号。

ただし変数は実数上を動くものとする。たとえば「$\forall x.x^2 \geqq 0$」と書いたら、「すべての実数 x について $x^2 \geqq 0$」を意味する。

このような言語で書ける文を仮に **R 文**と呼ぶことにしよう。たとえば「$a > 0$ ならば関数 $f(x) = ax^2 + bx$ は最小値を持つ」は R 文により

$$\forall a, b.(a > 0 \rightarrow \exists x.\forall y.\, ax^2 + bx \leqq ay^2 + by)$$

と書ける。

どんな R 文 φ についても、次のことが成り立つ。

- φ は量化子 $\forall, \exists$ を含まない文に変形できる（**量化子除去**）
- φ は真である $\Longleftrightarrow$ φ は実閉体の公理系 RCF から証明できる（**完全性**）
- φ が真かどうかは機械的に判定できる（**決定可能性**）

そんなわけで、R 文についてはコンピュータ証明がある程度うまくいく（ただし変数の個数が増えるに連れて計算時間は二重指数関数的に増加してしまう）。だが実数だけで数学はできない。自然数（整数）や関数、集合が自由に扱えて初めて数学である。一方で、R 文の言語には個々の自然数 $0, 1, 2, \ldots$ は含まれているものの、それらをまとめて「すべての自然数について」と言うことはできない（注：数学基礎論やコンピュータ科学では 0 も自然数に含める）。

また、「関数」や「集合」一般について語ることもできない。よく知られているように、関数は集合を用いて表せるし、自然数全体も集合を用いればうまく定義できる。結局のところ、R 文では**集合**について十全に語れない、それが問題の核心である。だからといって、自然数や集合について語れるように言語を拡張すると、よい理論的性質（量化子除去・完全性・決定可能性）は途端に失われてしまう。思い通りにはいかないものだ。もっとも、よい理論的性質が失われたからといってただちにコンピュータ証明が不可能になるわけではないから、諦めるのはまだ早い。

一階論理 vs 二階論理

そんなわけで、集合は取扱いが難しい。実閉体などの公理系の話は一端忘れて純粋に論理の話をすれば、**一階論理**から**二階論理**への移行が問題だということである。一階論理では、対象を表す変数 $x, y, z, \ldots$ しか用いないが、二階論理では、集合を表す変数 $X, Y, Z, \ldots$ も使うことができて、$x \in X$ のような表現が許される。それゆえ最低限の集合概念を用いることができる。そうすると、「n が自然数である」ことは

$$\forall X.(0 \in X \wedge \forall x.(x \in X \to x+1 \in X) \ \to \ n \in X) \tag{1}$$

と書ける。これが意味するのは、「0 を含み +1 について閉じているどんな集合 X にも n は属する」ということで、平たく言えば「数学的帰納法を使えるのが自然数だ」ということである。こんな風に、様々な概念を自前で定義できるのが、二階論理の特徴である。

一階論理と二階論理では性質が全く異なる。もっとも顕著なのは完全性である。

- すべての妥当な一階の文を導出できる完全な証明系が存在する（**完全性**）
- すべての妥当な二階の文を導出できる完全な証明系は存在しない（**不完全性**）

ただし上で「妥当な」というのは「標準意味論で妥当な」ということであり、「証明系」というのは「再帰的な証明系」のことである。不完全性が成り立つのは、(1) を用いれば二階論理（の標準モデル）の中で自然数の集合が定義できるからである。数理論理学の入門書では、ゲーデルの完全性定理（1930 年）と不完全性定理（1931 年）に言及するとき、二つの「完全性」の意味合いは異なると説明されることが多い。だがもちろん両者には深い関係があるのであり、（公理系ではなく）純粋論理で言えば、「一階論理は完全だが二階論理は不完全だ」ということなのである。

二階の壁

先に述べたとおり、不完全だったり決定不能だったりするからといって、コンピュータ証明が全くできないというわけではない。当面の目標は、人間数学者に匹敵するコンピュータ数学者を生み出すことである。だとしたら別にすべての真理を証明できなくてもよいし、すべての文の真偽を判定できなくてもよい。しかしそのように実利的な立場にたっても、やはり一階と二階の間には越えられない壁があるように思われる。このことを少しだけ詳しく説明しよう。

形式的にいえば、「定理を証明する」とは「公理から出発して推論規則を用いることにより目標定理に到達する」ことを意味する。これをコンピュータにやらせるときには、同じことだが「目標定理の否定を仮定して公理と矛盾することを示す」のがしばしば有効である。これを**反証手続**という。将棋とのアナロジーでいえば、目標定理の否定（および公理）という「初期盤面」から出発して、推論規則という「手を指す」ことにより、矛盾という「詰み」の状態に持っていくことである。

さて反証手続を進めるときに問題になるのが、量化子の取り扱いである。次のような推論規則がある。

「前提 $\forall x.\varphi(x)$ から $\varphi(t)$ を結論してよい」

すべての x について $\varphi(x)$ が成り立つのなら、当然具体的な値 t を x に代入しても $\varphi(x)$ が成り立つ。だからこの規則は当たり前のことを述べている。しかしどのような t を選べば矛盾に到達できるのかは、まったく定かではない。しかも t の取りうる値は無限にあるから、この「局面」において「次の一手」は無限通りあることになる。これは将棋のアナロジーからの大きな逸脱だ。

そこで自動証明をするときには、**代入例** t の決定を何らかの形で遅延する必要がある。そのために、**スコーレム化**と**単一化**というトリックがよく用いられる。たとえば、反証の過程で公理 $\forall x.\varphi(x)$ を使う必要があるときには、とりあえず新しい変数 z を使って $\varphi(z)$ と置き換える。また（固有変数条件を巡るいざこざを避けるために）$\exists x.\varphi(x)$ を使うときには、スコーレム関数記号 f を使って $\varphi(f(y_1,\ldots,y_n))$ で置き換える。ただし $y_1,\ldots,y_n$ は環境に現れる適当な

自由変数である（実行時スコーレム化）。具体的な z の値は単一化によって後で適切な時点で求める。ちなみに単一化とは、$f(x, g(x)) = f(c, y)$ のような “方程式” が与えられたとき $x := c,\ y := g(c)$ のような代入を施して両辺を等しくする手続きのことである。これは高速に実行可能である。

もちろん一階論理は決定不能なので、反証手続が必ず停止するという保証はないが、上の方針を用いれば、公理 $\forall x.\varphi(x)$ を使うときに次の一手が無限通りあるという困った事態は避けることができる。

一階論理をベースとした自動証明の研究は随分と進められてきた。数多くの**自動定理証明システム**が開発されており、コンピュータの性能向上と相まって、今や何百何千もの公理からなる公理系で定理を高速に証明することができる。CASC などの国際競技会が毎年開催され、多数の自動定理証明システムが、より多くの定理をより速く証明することを求めて競い合っている [2]。システムごとにデザインコンセプトは異なるが、一階自動証明がこれほど成功したのは、スコーレム化と単一化（あるいはそれに相当するもの）がうまく働いたおかげであるといってよいだろう。

ただし（汎用的な）自動定理証明システムが有効なのは、基本的に**一階論理**上**有限個**の公理で比較的素直に公理化ができる理論に限る。先ほど触れたロビンス予想はこの範疇に含まれる。一方で無限個の置換公理を伴う公理的集合論や、無限個の帰納法公理を伴う初等数論、それに二階論理ベースの公理系となると、随分様子が異なってくる。実際、**二階単一化**は決定不能であり、遅延トリックも一階の場合ほ

どうまくいかない。たとえば自然数の定義 (1) —— これは数学的帰納法の公理とも見なせる—— を証明中で用いるには、X に何らかの述語 $\psi(x)$ を代入して

$$\psi(0) \wedge \forall x.(\psi(x) \to \psi(x+1)) \ \to \ \psi(n)$$

としなければならないが、矛盾を導くようなうまい代入例 $\psi(x)$ を見つけるには、たとえ遅延トリックを用いても相当な試行錯誤が必要である。人間数学者はしばしば「経験と勘」でもってうまい $\psi(x)$ を見つけてくるが、そういった「経験と勘」に現在のコンピュータ証明はまったく太刀打ちできない。

一階に限らず、二階ないし**高階**の自動定理証明システムも多く開発されており、CASC 競技会の高階論理部門で互いに競い合っている。それでも一階と二階の間には本質的な壁が立ちはだかっているように筆者には思われる。そしてコンピュータ証明で人間に対抗するには、この**二階の壁**を何とかして克服しなければならないのである。

A New Hope: 人間からの学習

では一体どうしたらよいのだろうか？ここで注目したいのが**証明支援系**である。**対話的定理証明システム**ともいう。これが将来的にコンピュータ証明に大きな恩恵をもたらす可能性がある。証明支援系とは、人間が厳密な証明をするのを支援するツールである。人は誤りを犯しやすい。出版された論文でも、間違いが随分ある。コンピュータの支援により、間違いをゼロにしよう、また証明中の自明なステップはある程度コンピュータにまかせようというのが、その眼目である。産業界やコンピュータ科学界ではずい

ぶん使われるようになったが、数学での利用はまだまだこれからであろう。それでも、ほとんどの証明支援系は高階論理に基づいているか集合論を含んでおり、数学への応用を十分視野に入れている。たとえば、最近の注目すべき成果の 1 つに**ケプラー予想の厳密証明**が挙げられる。

1998 年、トーマス・ヘイルズらは 380 年以上にわたり未解決であったケプラー予想を解決したのだが、査読委員会の 4 年間にわたる努力にもかかわらず、完全な検証はなされなかった。そこでヘイルズが 2003 年に立ち上げたのが、**Flyspeck 計画**である。これは証明支援系を用いてケプラー予想を再度厳密に証明し直すことを目的としたものである。ちなみに Flyspeck とは「Formal Proof of Kepler」の頭文字 F,P,K をうまくつないだものである。計画は、2014 年 8 月 10 日に終了がアナウンスされた。ケプラー予想の厳密証明がついに得られたのである [3,4]。

しかし話はそれで終わりではない。検証の結果、大量の定義や補題からなる **Flyspeck ライブラリ**が残された。これは人間が証明支援系を用いて作成したものであり、それゆえ人間的な「経験と勘」が詰まった大変貴重なデータであるといってよい。これを用いてコンピュータに証明を学習させる実験がアーバンらにより行われている [5]。あたかも人間棋士が残した大量の棋譜をもとにコンピュータ棋士に「大局観」を学習させるかのように。以下、実験の一部を思いっ切り簡略化して説明する。

2012 年 6 月の時点で、Flyspeck ライブラリには少なくとも 1897 個の定義・公理と 14185 個の補題が含まれていた。定義・公理の集合を Γ とし、補題の集

合を $\Delta = \{L_1, \ldots, L_{14185}\}$ とする。証明とは定義・公理から出発して補題の積み重ねにより目標定理に到達する過程である。それゆえ各補題 L_n は、定義・公理 Γ や先行する補題 $\{L_1, \ldots, L_{n-1}\}$ から証明可能である。すなわち各 $1 \leqq n \leqq 14185$ について、

$$\bigwedge \Gamma \cup \{L_1, \ldots, L_{n-1}\} \to L_n$$

が成り立つはずである。これら 14185 個の文のうち、一体どの程度が自動証明可能だろうか？問題なのは n が大きくなるに連れ、先行補題の数がどんどん多くなっていくことだ。そうなると自動証明の効率がはなはだしく落ちる。

そこで次のようなアイデアが考案された。確かに L_n は先行する $\Gamma \cup \{L_1, \ldots, L_{n-1}\}$ に依存するが、そのすべてに依存するわけではない。事前に L_n に関連しそうな定義・公理・補題をうまく選択することができれば、自動証明の効率を上げられるだろう（**前提選択**）。そして適切な前提を選択する能力は、**機械学習**により強化することができる。

以上のようなシナリオ（本当はもっと複雑だが）の下で実験を行った結果、実に 39 %の補題について自動証明が成功したとのことである。

もちろんこの結果は、Flyspeck ライブラリの 39 %をコンピュータが自力で構築できるということを意味しない。定義や補題を「道標」として人間が与えてやれば、道標間のギャップのうち 39 %を自動的に埋めることができるというに過ぎない。それに、残りの 61 %の中にケプラー予想のエッセンスに関わる難しい補題が含まれている可能性が大いにある。それゆえこの実験の結果は、過大評価されるべきでは

ない。

それでも、ここには証明支援系、自動定理証明、機械学習の見事な協調が見られる。上の実験では機械学習の対象は前提選択に限られているが、その適用範囲を拡大しようとする後続研究もある。機械学習をさらに進めて、人間が残した大規模証明ライブラリからコンピュータが「経験と勘」を学ぶことができるようになれば、それが二階の壁を超える突破口になるかもしれない。たとえば数学的帰納法の公理 (1) を用いるときに、目標定理に応じて代入例 $\psi(x)$ にあたりをつける方法を学習できれば、事態は大きく変わることだろう。もちろん言うは易し、行うは難しである。今後どうなるかはわからない。

おわりに

最近「**コンピュータは数学者になれるのか**」という本を書いた [6]。世間の評判が気になったのでネットで検索してみたところ、「**数学者はコンピュータに慣れるのか**」というツイートが引っかかった。世の中には、物事の本質をたった一言で言い表してしまう頭のよい人がいるものである。実際、この 2 つの問いが**等価**であること、それが本質なのだと思う。現在の証明支援系は、はなはだ使い勝手が悪く、普通の数学者が快適に使えるような代物では全然ない。だがコンピュータ証明が発展すれば、その成果は証明支援系に反映されるはずだ。実際多くの証明支援系が、部分的に自動証明するメカニズムを備えている。そうなると人間側の負担が減り、人間数学者がいっそう証明支援系を使うようになるだろう。そうなると人間による証明データがますます蓄積され、それ

が機械学習に活かされ、コンピュータ証明はさらに発展するはずだ。この**ウインウインの循環**を継続させること、それがコンピュータ証明を成功に導く唯一の道なのではないか、そんな風に空想する次第である。

参考文献

[1] W. McCune. Solution of the Robbins Problem. *Journal of Automated Reasoning*, 19(3):263-276, 1997.

[2] http://www.cs.miami.edu/~tptp/CASC/

[3] T. Hales, et. al. A formal proof of the Kepler conjecture. arxiv.org/abs/1501.02155, 2015.

[4] 溝口佳寛, 田上真. ケプラー予想の計算機による証明と検証について. 数学セミナー 2014 年 12 月号, pp. 48-55, 2014.

[5] C. Kaliszyk and J. Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53: 173-213, 2014.

[6] 照井一成. コンピュータは数学者になれるのか？数学基礎論から証明とプログラムの理論へ. 青土社, 2015.