識を発想する方法も含めて高速化する知識ベースのコンパイル法を開発した．これによって推論をメモリ探索に置き換えることができる．制約充足問題 (CSP) における k-consistency アルゴリズムと深く関係した技術である．知識ベースすべてをコンパイルするのはメモリ空間が膨大となり実用的でないので，部分コンパイル法も開発した．類推，学習の利用による高速化が人工知能的観点からのアプローチであるのに対し，知識コンパイルはコンピュータ技術寄りの観点からのアプローチといえ，両者の交流はますます重要となろう．

発想やひらめきのメカニズムなどは難しいように思えるが，このアプローチの延長で10年ほどで利用の目途がついてくると思う．発想までいったからといって人間の知能機能を実現できたというわけではなく，このような知能機能は人間の脳でいえば左脳(理性の脳)の部分である．右脳（感性の脳）に関係する直観（常識も深く関係する）などは残される．ニューラルネットワークは右脳の機能への一つのアプローチといえるが，どこまでのものか筆者にはよく分からない．10年，15年のスパンで考えれば，左脳の部分はコンピュータですべて実現し，右脳の部分は人間が担当し，その間に良いヒューマンインタフェースを実現することが工学的なアプローチといえよう．並列コンピュータなどによるコンピュータパワー，分散協調推論などを，このような情報システムを可能にするために導入していくことも重要な視点である．30年後のこととなると（筆者はもう研究をしていないであろうから）よく考えていない．

以上，筆者の個人的な考えを記した．大きな人工知能という目標に対して，研究者は個々に明確な視点，サブゴールを設定して研究を進めることが重要だと思う．M. ミンスキー先生の「心の社会」の本[2]は，今後10年，20年，30年先の人工知能技術を考える上でいろいろと示唆に富んでいると思う．

参考文献

1) 石塚　満：不完全な知識の操作による次世代知識ベースシステムへのアプローチ，人工知能学会誌，Vol. 3, No. 5, pp. 552-562 (1988).
2) M. ミンスキー（著），安西（訳）：心の社会，産業図書 (1990).

石塚　満（正会員）

昭和46年東京大学工学部電子工学科卒業．昭和51年同大学院博士課程修了．工学博士．同年NTT入社，横須賀研究所勤務．昭和53年より東京大学生産技術研究所・助教授，現在に至る．人工知能，知識システム，画像理解，ヒューマンインタフェースの研究を行っている．現在，本学会人工知能研究会主査．

当たらぬも八卦

竹内　郁雄†

私は4～5年前のこと，ソフトウェア工学研究会主催のシンポジウムのパネル「明日のソフトウェア工学」で，「明後日（あさって）のソフトウェア工学」と題して，あることないこと，まるでアサッテの話をベラベラやったあと，そのうち明治生命の保険屋さんがソフトウェアの信頼性について画期的な貢献をする，1992年でソフトウェアの進歩は止まる，2016年12月2日に天才的学者がプログラムの最適冗長性に関して画期的な論文を発表する，などとデタラメな予言をしたことがあ

記号処理研究会主査
† NTT 基礎研究所

る[1]．で，またもいい加減なことを書けとのお達しである．

30年先の予測というのは情報処理のように急激な進歩をとげている分野ではいかなる意味でも視野の外という感があるが，冷静に考えると，情報処理の進歩とはなんだろうという疑問も起こる．ハードウェアは，速度にしろ容量にしろ過去30年間で約100万倍の性能向上があった．これは数値として見えるので分かりやすい．ソフトウェアの進歩は？というと，うまく数値化できないのでどうも説得力のある説明ができない．しかし，そういえば1960年代，いまをときめく大先生たちの若かりしころ，日立中研で数人がかりで1年以上かけてFortranのコンパイラを作っていたという伝説がある．これはちょっとできる学部学生の練習問題程度だといまは思われている．量ではうまく説明できないが，これはだれにでも納得できる進歩の顕れだろう．

ここで私の基本定理を発表しよう．例によって，証明は余白に書くには長すぎる．

定理：ソフトウェアの進歩とは人間の慣れのことである．

要するにだれかが一度やって，ハハーン，なんだ，そういうことだったのか，とみんなが納得してみんなもそれをできるようになることがソフトウェアの進歩なのである．コロンブスの卵や，一輪車の乗りこなしと異なり，ソフトウェアは運動感覚の問題ではないから，みんなが納得するためには，みんなが共有できるような概念または語彙が必要である．少々古いがスタックという語彙はその端的な例だろう．こういう事情はまさに言葉の変化・進化と同じである．

この定理のもう一つの含意は，ソフトウェアの進歩に絶対基準はないということである．それは，社会制度といった数千年の歴史をもつ「ソフトウェア」のことを考えてみれば容易に了解されよう．世の中すべてUnixになることが進歩なのか，10年ごとに様変りしたOSが出てくることが進歩なのか，にわかには判定し難いものがある．

さて，この定理を念頭において30年後を占ってみよう．いくつかの予測を箇条書にするが，おたがいに矛盾したものがある．複数個書けば，どれかは運よく当たることもあろうという算段である．

1. 蓄積型プログラムの終焉

ハードウェア技術の進歩により，フォン・ノイマンの蓄積型プログラムの原理は捨て去られる．つまり，プログラムを書くということは即座に専用のプロセッサが作られるということになる．これはその都度LSI（そのころはULSI, Super LSIの上のExtremely LSI, すなわちELSIになっている）を焼くというのではなく，可塑性をもったELSIを瞬時に変更するという操作で行われる．昔，アナログコンピュータのパッチボードの配線を手でつなぎ変えていたのが微細素子のレベルで実現するわけである．つながれる機能回路要素は，$n \times n$ 行列の浮動小数点掛け算器（n でパラメータ化された最適の回路）とか，文字列のソータ，3次元パターンの山崎マッチャー（2015年ごろに山崎雄一郎という人が発明するはずのマッチャー）とかいった現在の目で見ればかなり高等なものであり，ソフトウェアの常用語彙とほぼ一致したものになる．

2. キーボードの終焉

人間の慣れを追求した結果，とうとう人間はコンピュータに特化した発声ができるようになる．話言葉ではせいぜい150ボーしかいかないが，モデムのような発声により，1200ボーぐらいの入力が平気でできるようになる．頭の回転も当然それに見合って速くなる．音を使えないときは，現在の手話をさらに高速化した「指話」を使う．若い人同士の会話もこれらを使うようになる．しかし，われわれのような老人（そのときまで生きていればの話だが）のために，蒔絵付き漆塗りのキーボードなども一応生き残る．

3. プログラミング言語の自然言語化

自然言語とプログラミング言語がおたがいに歩み寄ることにより，プログラミング言語と自然言語の区別がはっきりしなくなる．当然，プログラム自動合成といったものは死語となる．また，プログラミング言語で詩歌や文学が書かれるようになる．つまり，executable poemといった感じのものが登場する．

4. コンピュータソフトウェアが社会制度になる

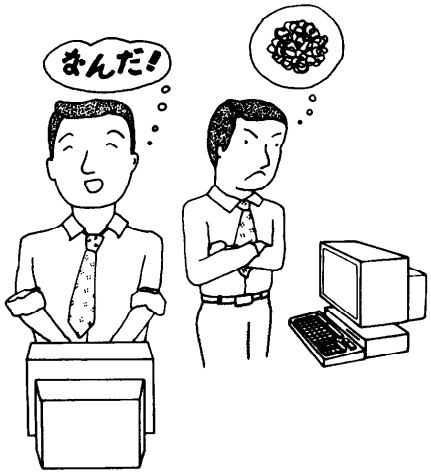
人間の扱える情報の総量はそれほど伸びない．だから，コンピュータソフトウェアの質・量が拡大するにつれ，基本ソフトウェアから順にアプリケーションソフトウェアまでどんどん（中身を知る必要のない）ブラックボックス化が進んでいく．そして基本ソフトウェアになるほど，新しいものは作られなくなっていく．これはソフトウェアのある意味での硬直化である．コンピュータはすでに社会のインフラストラクチャになっているから，これに合わせて社会制度自身も硬直化してしまう．つまり，社会制度を既存のソフトウェアに合わせてしまうようになる．

5. 完全制御から遺伝子型制御へ

現在のところ，プログラムとは究極的には計算機の論理回路の動作すべてを制御するという発想で作られているが，30年後には超並列非同期計算機が当たり前になるので，プログラムであらかじめ計算機システムの動作を指定するということが不可能になる．すなわち，従来の意味でのソリッドな制御をあきらめなければならない．それに代わるのは，ちょうどシミュレーションのようなプログラミングである．あるいは統計力学的なプログラミングである．超並列非同期計算機の個々の要素は制御できても，全体の動作は決定論的には制御（あるいは予測）しえない．しかし，それでも何かを合目的に行わせるために局所的で間接的な制御，遺伝学のアナロジーでいうと遺伝子型制御の概念が登場する．これに関してはある座談会で，「薬品調合型プログラミング」というタイトルで放談をしたので参照願いたい[2]．

6. 自律的な計算機

上に述べたことにも関連するが，自己モニタ能力（リフレクション能力）をもつ計算機が登場する．つまり，疲れを知らない計算機ではなく，同じことをやっていると飽きてくるような計算機である．自己モニタが可能になることによって，ホメオスタシス，自己変革，自己学習などが可能になる．このためには，現在の計算機のように記号論理やデジタル回路に基づいているだけではダメ



である．誤解を恐れずにいえば，記号に物理量が付随したようなデジタル・アナログのハイブリッド計算機でそれが実現される．当然，記号論理学は廃れ，記号物理学や統計記号学が興る．ここでも決定論的な理論基盤は敗北する．

7. Fortran は不滅です

過去30年間，Fortran は不滅であった．だとすれば，C も Unix も今後 30 年間は不滅かもしれない．これは 4. で述べたことと関連するが，ソフトウェアというのは一つの文化であるから，目まぐるしく変化するというより，蓄積していく性質のものである．従来の文化と異なり，情報流通が瞬時かつワイドに起こる時代だから，地域に文化が付随するという図式は成り立たない．しかし，今日のいわゆる EDP 屋と AI 遊牧民の間の文化ギャップにみられるような種族(?)間文化格差は 30 年後にはますます固定化していく．そのときにはもう情報処理というキーワードでは種族間で共通の話題がなくなる．情報処理学会は，各種族の学会（Fortran 保存会，C 保守学会，Unix 検定学会，…，記号物理学会等など）の連合として存続することになるだろう．

ほんまかいな．

参　考　文　献

1) パネル討論「明日のソフトウェア工学」，情報処理，Vol. 28, No. 7, pp. 922-939 (1987).
2) 座談会「未来のプログラミング」，月刊アスキー，No. 160, pp. 294-300 (1990. 10).

竹内　郁雄（正会員）

1946年生．1969年東京大学理学部数学科卒業．1971年同大学院理学系研究科修士課程修了．同年，電電公社電気通信研究所入社，現在，NTT基礎研究所情報科学研究部主幹研究員．プログラミングパラダイム，AIマシン・言語などの研究を行っている．ACM，日本ソフトウェア科学会各会員．

ソフトウェア作成技術

大　蒔　和　仁†

1. ソフトウェア作成技術の発展はハードウェアのそれと比べて遅いか

30年後について思いをめぐらそうとするときに真っ先に思いつく方法は，30年前から現在に至るまでの状況を思い出し，その発展過程を外挿することである．一般にソフトウェア作成技術はハードウェアのそれと比べて発展が非常に遅いといわれる．本当にそうだろうか．この章では過去を振り返ってみて，ソフトウェア作成技術は非常に進歩してきたと主張したい．

30年前（1960年代前半）といえば，筆者は小学校へ入学したころである．ちょうどテレビ（白黒）が普及しかけたときである．そして，テレビをもっている家に近所の人たちが集まり，大鵬と柏戸の相撲などを見たものである．計算機の歴史の本をひもとけば（私が体験したわけではないが），当時の計算機ソフトウェアの作成技術の状況を想像できる．今の状況と比べてみると，CRTの端末もなければTSSもないという，想像もできないようなプログラム開発環境であった（らしい）．

次に20年ぐらい前（1970年代前半）を考えてみたい．そのころインテルの4ビットワンチップマイコン4004ができて，すぐに8008そして8080となっていく．筆者は卒業研究で8008に初めて触った．つづいてモトローラの6800などができたりした．このころはラジオ少年たちにとっては，まだ素人でも手作りで計算機ができるという，夢多き時代であった．また当時は，OKITAC 4300やHITAC 10などの，ミニコンと呼ばれる16ビットで磁気コアメモリの計算機を手元において触れる時代であった．FORTRANのコンパイラもあったが，紙テープで入力し，出力は機械的なプリンタが主流であった．一方，大型機はIBM系統のジョブ制御言語を覚えて，パンチカードで入力するシステムが主流であった．

1970年代後半になると，それまでのスパゲッティプログラムを廃して構造化プログラミングすべきであるとの主張からAlgol系のPascalなどが盛んに推奨された．このころは，特に文脈自由文法などの形式言語に関する研究が精力的に行われ，この理論がコンパイラ作成技術の確立へとつながっていった．このころにはTSSのシステムも一般的に使われはじめ，端末もCRTつきのものへと変わっていった．これが今からほんの10数年前のことである．

1980年代前半にはMacintoshが出て，それまで高嶺の花であったマウスとかアイコンとかの道具が身近になった．

現在ではビットマップやマウスを備えたワークステーションを使ってプログラムを開発しないと

ソフトウェア工学研究会幹事
†† 電子技術総合研究所情報アーキテクチャ部言語システム研究室