192 プログラム・シンセシス(自動合成)の理論的側面

後藤 滋樹
(日本電信電話公社 武蔵野電気通信研究所)

1. シンセシスの方法(原理)

プログラムのシンセシス(自動合成)を行う方法は、いろいろ考えられているが、大きく分ければフォーマル(論理的)なものと、インフォーマルなものがある。ここでは、フォーマルな方法、すなわち定理の自動証明(1)に基づくシンセシスについて考察する。

この方法は、①プログラムと論理式との間には密接な関係があること、②論理式の世界では自動証明が行えること、を利用している。実際の手順としては、まず(求める)プログラムが満たすべき性質を論理式で記述し、その論理式を自動的な方法で証明して、得られた証明からプログラムを引き出す処理を行う。

ここで大切なのは、「論理」としてどのような体系を選ぶべきかという点である。なぜなら、論理体系が異なれば、①プログラムと論理式との対応関係も、②自動証明のやり方も変わるからである。

2. 論理体系の重要性

一般に、単に論理と言った時には「一階古典述語論理」を指すことが多い。しかし上述のようにプログラムとの対応関係を考えた時には、古典論理よりは直観主義論理(intuitionistic logic)の方が望ましい[1]。例えば図1に示す二つの論理式とそれに対応するプログラムを考えてみよう。この二つの論理式は直観主義論理においては同値ではなく、また二つのプログラムもプログラムとして同値(2)ではない。にも拘らず、古典論理においてはこの二つの論理式は同値になってしまう[2]。

論理式1: $\forall x_1 \forall x_2 [x_2 \neq 0 \supset \exists z_1 \exists z_2 (x_1 = x_2 \cdot z_1 + z_2 \wedge z_2 < x_2)]$.

論理式2: $\forall x_1 \forall x_2 \exists z_1 \exists z_2 [x_2 \neq 0 \supset (x_1 = x_2 \cdot z_1 + z_2 \wedge z_2 < x_2)]$.

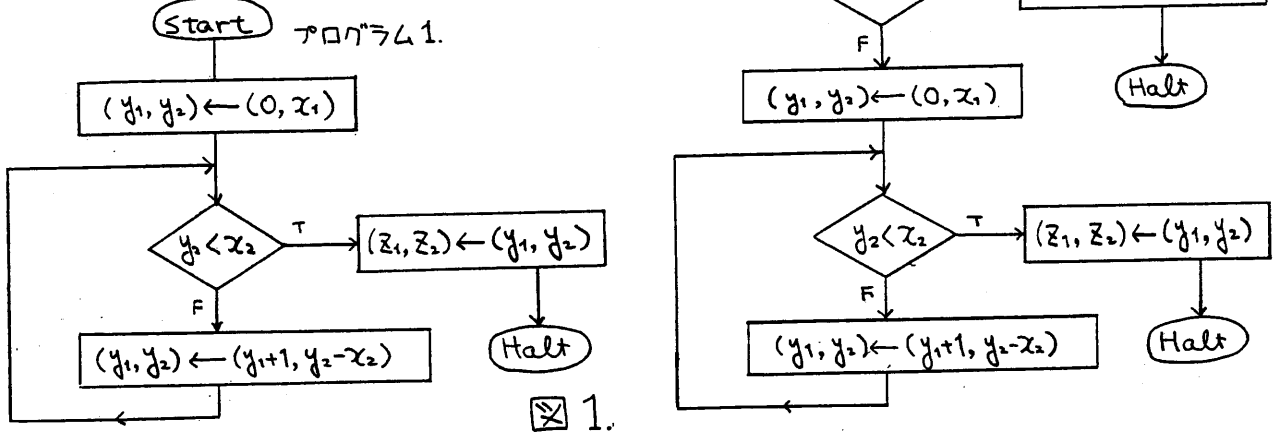


図1.

(1) 自動証明とresolution principleとを混同しないで頂きたい。resolution principleは自動証明法の一つではあるが、すべてではない。

(2) プログラムの同値の定義は、例えばManna[3].

この例を一般化して述べると，一階直観主義述語論理においては下の関係が証明できるが，逆は成立しない。

$$\forall x \exists z [A(x) \supset B(x,z)] \supset \forall x [A(x) \supset \exists z B(x,z)] \quad \cdots\cdots\cdots (1)$$

また論理式と recursive function[3] との関係を用いると，$\forall x \exists z [A(x) \supset B(x,z)]$ には total recursive function が，また $\forall x [A(x) \supset \exists z B(x,z)]$ には partial recursive function が対応するのである。これはちょうど図1.の場合のプログラムの totality/partiality に一致する。さらに，プログラムの拡張(extension)関係と式(1)との対応も良い。

ただ残念なことは，一階直観主義述語論理に対しては，古典論理における resolution principle のような強力な自動証明法が開発されていないことである。実際のシンセシスにおいては自動証明がひとつの重要なファクターであるから，この方面の今後の発展が待たれる。

3. Resolution principle との関係

上の結果を用いて，従来の方法(resolution)を新たな角度から見直してみよう。resolution principle における肝要な部分 $(P \vee Q) \wedge (R \vee \neg Q) \supset (P \vee R)$ は直観主義においても成り立つ。このルールを適用する以前に論理式は clause (節) の形に変形されるが，このうち興味深い部分を図2.に例示した。図2によれば，total/partial の区別がなくなって一律に total 側に変形されてしまうことがわかるであろう。このことから従来の方法は total/partial の微妙な区別を保存しないとも言えるし，また安全側(total)へ倒れるとも言える。

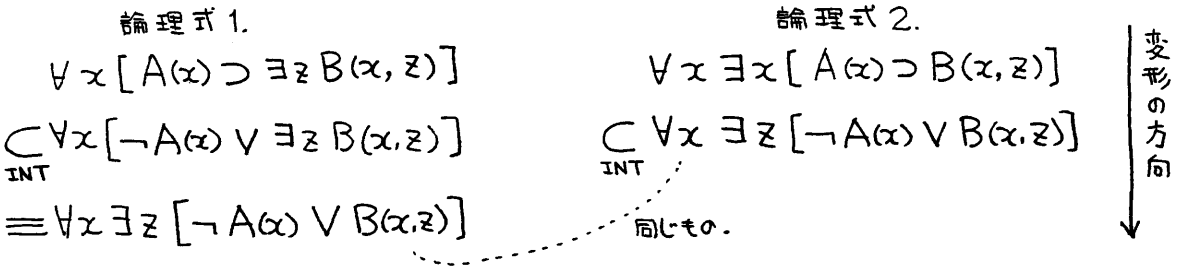


図2. clause形への変形(特に prenex normal form への変形)の例.
$\subseteq_{INT}$ は直観主義論理における関係であることを強調したもの。

最後に，本研究について日頃御討論頂く武蔵野通研・基礎研究部オー研究室ならびに池野特別研究室の諸氏に感謝いたします。

文献

[1] S. C. Kleene, Realizability: a retrospective survey, Cambridge Summer School in Math. Logic, pp 95-112, Springer Lecture Notes in Math. 337 (1973).
[2] 後藤 滋樹, Recursive realizability とプログラム・シンセシス, 信学技報 AL75-54 (1975).
[3] Z. Manna, 五十嵐 滋(訳), プログラムの理論, 日本コンピュータ協会 (1975).
[4] 松本 和夫, 数理論理学・増補版, 共立出版 (1971).
[5] M. C. Fitting, Intuitionistic Logic Model Theory and Forcing, North-Holland (1969).

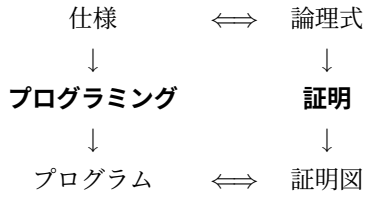
(3) ここでは，recursive function とプログラムとをほぼ同じものと考えておく。

情報技術は電子文具を超えるか

後藤滋樹
早稲田大学 理工学部

1　ソフトウェアの危機

ソフトウェアが大量に必要になるのに、プログラムを書くことのできるプログラマの人数が絶対に不足する。これが**ソフトウェアの危機**の意味であった。これを解決するにはプログラムを自動的に合成 (シンセシス) できるようになれば良い。



上の図は幾つかの事例において正しく動く。ただし一般的にプログラムを自動的に合成できるかというと、それは難しい。結論を簡単に言うと、**外部**の仕様 (specification, 入力と出力との関係で表す) からプログラムの**内部**を決めることが出来ない場合がある。

2　温故知新

最初から理論的な限界が明らかであっても、実際には問題を分類したり、工夫を凝らすことができる。上の枠組を研究する中では、研究の旺盛であった欧州の研究者と数ヶ月違いで論文での発表を争ったことがある。国内では研究仲間との親交を深めた。さらに畏友佐藤雅彦氏 (現在京都大学教授) の紹介により、私は 1984 年夏から米国スタンフォード大学に客員研究員として滞在することになった (1984 〜 85)。1 年間の短い期間であったが、米国には日本には無いものが存在した。これは面白い経験になった。

当時の ARPAnet が大学での生活の一部を成していたことは印象的である。日本ではインターネットに取組む人が少なく、私は帰国後はネットワークにも関わることになる。この部分が温故知新としての話としては役に立つかもしれない。ただし他の機会に講演をした記録があるので、そちらを参照していただければありがたい [1]。

スタンフォード大学に設置されていた CISCO の製品を日本で初めて輸入して使用したり、スタンフォード大学で良く使われていた TeX を導入して、斉藤廉己氏、磯崎秀樹氏、桜井貴文氏らの尽力で NTT 版 jTeX/jLaTeX が誕生したことも楽しいな思い出になっている。私自身は大日本印刷にフォントの交渉に出掛けたくらいで、 jTeX のプログラムには貢献していない。

3　コンピュータ・サイエンスの危機

自分の研究は棚に上げて、この分野の概観をする。情報分野には多くの研究成果があるにも拘わらず、世の中にあまり恩恵をもたらしていない。

オペレーティング・システムは人間 (オペレータ) の手間を省き、コンピュータの運用するために資源 (リソース) の管理をする。いわば自動化を目的とする。その結果、**自動車**のような立派な技術が完成したが、世の中の人は自転車のようなオペレーティング・システムを使っている。

- 自転車は乗る人によって速度が違う。
- 自転車は、すぐに転ぶが、起き上がるのも早い。
- 自転車は、運転手とお客の区別が曖昧。

データベースの研究は、物理的な媒体から論理的に独立するという動きである。折角リレーショナル・データベースまで実現したのに、世の中の大部分のデータは「べた詰め」の文書であり、全文検索がまかり通っている。

プログラミング言語は、マシン語を離れて、いわば紙の上でもプログラムが理解できるという高級言語の路線を進んできた筈である。ところが世の中では Java という昔ながらのスタイルが主流である。

4　もっとアプリケーションを！

多くの人が**パソコンを使える人**になった。それは世の中の進歩であるとしても、実際にパソコンをどのように使っているのだろうか。ワープロ、表計算、プレゼンテーション、という三点セットであれば、**電子文具**の範疇である。インターネットを使う場合でも、電子メールは遠隔地にファイルをプリントするようなものである。 Web は電子辞書の拡張版である。これがコンピュータの使い途だとするならば、ごく少量のプログラムがあれば十分だ。

むしろコンピュータを使っていると悪者に狙われる弱点を持つ。ウィルス騒動や不正侵入の事例が日常的に起こる。それが普通の新聞やテレビのニュースの題材になる時代になった。皮肉なことに、ウィルスや不正侵入は高度のプログラミング技法を応用している。

現代社会に閉塞感があるとすれば、それは社会基盤としての情報処理技術の閉塞状況でもある。今の時代には、情報処理の側から積極的に**アプリケーション**を探しに行くべきであろう。

参考文献

[1] 後藤滋樹「日本のインターネットの歴史と教訓」 http://www.k12.gr.jp/OLD/page4-5-goto-profile.pdf