Automatic Coding について（Ⅱ）

— Boolean Expression の翻訳 —

穂鷹良介

筆者は先に本誌上（第巻15第1号）で論じたように，Automatic Codingの一種である Assembler HAP の開発に成功したが，より高級な Automatic Coding を作成するための一段階として，ALGOL に含まれている形式の Boolean Expression の翻訳プログラムの作成を手がけて見た。本論文ではその際筆者によって案出された多少の翻訳テクニックを紹介し，又，その結果を利用した計算例を述べる。ALGOL に含まれている Boolean Expression の形成する論理はいわゆる古典述語論理の一部分体系をなす NKM で，その論理命題式の真偽は有限回の手続きで確かめることが出来ることが証明されている。（吉田夏彦著「論理学」培風館 新数学シリーズ第10巻参照）従って差し当っての応用例としては，このプログラムを利用して任意の Boolean Expression の Truth Table を作ることが考えられるが，この例が後に述べられる。今回使用した計算機は OKITAC 5090A である。なお，計算機使用に際しては小樽商科大学教授　古瀬大六氏に一方ならぬ御便宜をはかっていただいたことに対し，ここで感謝の意を表したい。

1. Boolean Expression

本論文でいうところの Boolean Expression の意味を明確にしておこう。

その定義は ALGOL で定めているものと殆んど同じであるが，その構成要素は次の二種のものである。

i) variable これは英数字の組合せとする。

ii) End Mark（以後 E. M. と略称する）これは演算 operator ¬, ∧,

∨, ⊃, ≡ の5種とカッコ（, ）2種及びピリオド．空記号 ϕ の計9種とする。演算 operator は上にあげた順に夫々，negation, logical and, logical or, implication, equivalence と呼ばれているもので，operator 間の演算順位もこの順序である。

Boolean Expression は variable を演算 operator に関して意味のあるように配列して最後にピリオドを置いたものと定義する。

例えば

$$\neg\neg A \vee B \wedge C \supset D.$$

はここでいう Boolean Expression で演算順序を考慮するとその意味は

$$((\neg(\neg A)) \vee (B \wedge C)) \supset D.$$

と同じである。演算 operator の真理表を表1にかかげる。

表　1

B1	B2	¬B1	B1∧B2	B1∨B2	B1⊃B2	B1≡B2
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

T は True　　F は False を示す。

2. Interpretation Beginning End Mark (I. B. E. M.)

E. M. 間に次のような order ＞ を導入する。

$$\neg > \wedge > \vee > \supset > \equiv > (>) > . > \phi$$

計算機内部ではこのような order はその order を整数の大小として比較するのが便利であるので次のような mapping φ : E. M. → 整数を考える。$\varphi[\neg]=8$, $\varphi[\wedge]=7$, $\varphi[\vee]=6$, $\varphi[\supset]=5$, $\varphi[\equiv]=4$, $\varphi[(]=3$, $\varphi[)]=2$, $\varphi[.]=1$, $\varphi[\phi]=0$.

この mapping は E. M. 間の order ＞ を実数間の order ＞ として保存する。この order は演算 operator に関しては演算の優先順位と一致してい

るが，変っているのは，右カッコ），左カッコ（，ピリオド．，空記号 ϕ にも operator と同様にその間に order が定義されていることである。このようにカッコ，ピリオド等も或程度 operator と同列に扱って翻訳の論理的一貫性，規則性を持たせたのがこの翻訳プログラムの特徴である。

ある任意の Boolean Expression があるとしよう。このときBの前に ϕ を置いたものを $B'=\phi B$ とする。B' に存在する E. M. だけをとりだして E. M. の列を作ったとき，ϕ から始めて相隣る E. M. を pairwise に比較していき，前の E. M. に比較して order の低いもの又は同じものが現われたとき（Bに演算 operator が少くとも一つ含まれていれば，Bの終りはピリオドであるから，このような E. M. は必らず存在する）その新たな E. M. を I. B. E. M. と定義する。

3. 翻　訳　方　法

一般的な翻訳の手続きを言葉で表現することは不適当と思われるので，ここでは一般的な rule を Fig. 1 にかかげた flow Chart で示しておいて，後は具体的な例についてこの流れをどうたどって翻訳が進行するかを見る。そのために Fig. 1 の中の 2，と 9，とについてあらかじめ説明する。

読込み Sub-Routine　box 2.

この操作 box では variable のある場合には variable とその後に続く E. M.，ない場合には E. M. のみを入力より読みとる。（これ以外の場合，例えば variable の後に E. M. が来ないで再び variable があるような場合は文法違反である。）例えば以下の例 1 のように A∧B. とあれば始めの読込みで A∧；2 回目に B. 迄を読む。又 ((A∨B)∨A)∨B. のような場合には，(；(；A∨；B)；∨；A)；∨；B.； のように；の区切りごとに読んでいくものとする。

翻訳 Routine　box 9.

この操作 box では I. B. E. M. が検知されたときにその段階で関与している operator 一個に関しての object プログラムを作成する所である。なお，

その際に翻訳し終ったものを順に Table から取り去ってしまい，且つ Table に残して置く必要のある情報を Table にいれることもする。

なお object program に現われる WS 1, WS 2,……は temporary な作業番地で，この記憶容量の節約の方法については第 4 節に述べる。

例 1　　A∧B.

操作又は判定	Table	翻訳結果
1. Table Clear	ϕ	
2. A ∧ をよむ		
3. 7 に進め		
7. A→Table	ϕ ; A	
8. ϕ＜∧故 6 に進め I. B. E. M. でない		
6. ∧→Table	ϕ ; A ; ∧	
2. B. をよむ		
3. 7 に進め		
7. B→Table	ϕ ; A ; ∧ ; B	
8. ∨＞・故 9 に進め I. B. E. M. である		
9. ∧に関して翻訳	ϕ ; WS 1	A∧B→WS 1
10. ϕ＜・故11に進め		
11. E. M. は・故 翻訳修了		

左端の数字は Fig. 1 のそれと対応している。Table の内容，翻訳結果は書き改めない限り同じ内容が保たれているものとする。

上のようにして A∧B の結果を WS 1 にしまい込むような program が作成された。

例 2　A≡B⊃C.

始めに翻訳結果としてどういうものが得られれば良いかを見ておく。演算

順序の約束に従いこの Boolean Expression は A≡(B⊃C). の意味を持つから B⊃C→WS1；A≡WS1→WS1 のプログラムを作れば良い。

	操作又は判定	Table	翻訳結果
1.	Table Clear	ϕ	
2.	A≡をよむ		
3.	7に進め		
7.	A→Table	ϕ；A	
8.	6に進め		
6.	≡→Table	ϕ；A；≡	
2.	B⊃をよむ		
3.	7に進め		
7.	B→Table	ϕ；A；≡；B	
8.	≡＜⊃故 6に進め		
6.	⊃→Table	ϕ；A；≡；B；⊃	
2.	c. をよむ		
3.	7に進め		
7.	c→Table	ϕ；A；≡；B；⊃；C	
8.	⊃＞・故 9に進め		
9.	⊃に関して翻訳	ϕ；A；≡；WS1	B⊃C→WS1
10.	≡＞・故 9に進め		
9.	≡に関して翻訳	ϕ；WS1	B⊃C→WS1； A≡WS1→WS1
10.	11に進め		
11.	E. M. は・故 翻訳完了		

翻訳完了時には所期の結果が得られている。

例 3　　A∧(¬B∨C).

この例では negation ¬，カッコの翻訳の仕方が分る。

操作又は判定	Table	翻訳結果
1. Table Clear	ϕ	
2. A∧をよむ		
3. 7へ進め		
7. A→Table	ϕ ; A	
8. 6へ進め		
6. ∧→Table	ϕ ; A ; ∧	
2. (をよむ		
3. 4へ進め		
4. 5へ進め		
5. 6へ進め		
6. (→Table	ϕ ; A ; ∧ ; (	
2. ¬をよむ		
3. 4へ進め		
4. 5へ進め		
5. φ[¬]= 8 故 8へ進め		
8. (＜¬ 故 6へ進め		
6. ¬→Table	ϕ ; A ; ∧ ; (; ¬	
2. B∨をよむ		
3. 7へ進め		
7. B→Table	ϕ ; A ; ∧ ; (; ¬ ; B	
8. ¬＞∨故 9へ進め		
9. ¬に関して翻訳	ϕ ; A ; ∨ ; (; WS1	¬B→WS 1

10.	(＜∨故 11に進め		
11.	6に進め		
6.	∨→Table	ϕ；A；∧；(；WS1；∨	
2.	C)をよむ		
3.	7に進め		
7.	C→Table	ϕ；A；∧；(；WS1；∨；C	
8.	∨＞)故 9に進め		
9.[注1]	∨に関して翻訳	ϕ；A；∨；WS1	¬B→WS1 WS1∨C→WS1
10.	2に進め		
2.	・をよむ		
3.	4に進め		
4.	9に進め		
9.	∧に関して翻訳	ϕ	¬B→WS1 WS1∨C→WS1 A∧WS1→WS1
10.	11に進め		
11.	E. M. は・故 翻訳完了		

この flow-chart によれば，同順位の operation は左側から順に結合することになる。例えば

B1∧B2∧B3.

は (B1∧B2)∧B3. と翻訳される。こうしても上記の¬を除く4つの operation に関しては，各々結合法則が成立しているから構わない。

4. object program における作業番地の節約について

例えば A1∨A2∨A3∨A4∨A5. を計算する場合に一時的に答を格納して

注1）この段階では，最後に出た E. M. が) でその前の E. M. が (である。この場合には10で判定して Table の (を一つ取り去り 2. へ進むような別の配慮が必要であった。

おくための記憶容量は1個で間に合う。つまり object program は (A1∨A2)→(WS1), (WS1∨A3)→(WS1), (WS1∨A4)→(WS1), (WS1∨A5)→(WS1) のように作成しておけば良い。同じ問題を (A1∨A2)→(WS1), (WS1∨A3)→(WS2), (WS2∨A4)→WS3, (WS3∨A5)→(WS4) の如く compile することはより簡単であろうが，上の方式に比べて余分の作業番地を3個使用しなくてはならない。

しかし，作業番地はいつでも一個で済むかというと，そうではなく，今迄に述べて来たような翻訳方法を取る場合

$$(A1 \vee B1) \equiv (A2 \vee B2) \supset (A3 \wedge B3 \vee C3) \vee A4 \wedge A5.$$

のような Boolean Expression を object program に直すには (A1∨B1)→WS1, (A2∨B2)→WS2, (A3∧B3)→WS3, (WS3∨C3)→WS3, (A4∧A5)→WS4, (WS3∨WS4)→WS3, (WS2⊃WS3)→WS2, (WS1≡WS2)→WS1 といったやり方が最も作業番地を食わない翻訳方法であろうから，最低4個の作業番地を要する。これをもし作業番地の節約を考慮せずに（つまり一度使った作業番地をもう一度使うということを考えないで）翻訳したとすると，object program は8個の作業番地を要することとなり，丁度倍である。

この節約の方法は次のようにして出来る。

二つの作業番地を使用して二項関係を翻訳した時，且つその時に限り一方の作業番地は以下の演算で自由に用いることが出来るから，翻訳の際に作業番地を WS1, WS2,……の順に使用したものとすれば，WSi と WSj ($i<j$) の間で演算した結果は必ず WSi に格納するようにすれば良い。ここで注意しなくてはならないことは，演算の種類又は翻訳の段階によっては，作業番地が1個又は0個しか関係しない時があるから，作業番地を2個使ったという条件を適切に判定しなくてはならない。

5. 論 理 演 算

ALGOL でとり扱う論理演算は 1. で述べた5種類であるが，それらの計

算機内部での取扱いは Exclusive Or と Logical And との二つの命令があれば次のように簡単に行なうことが出来る。但し情報は True を 1 で，False を 0 で表現することにし，⊕ で Exclusive Or の演算，∩ で Logical And の演算を示すものとする。

¬X　　これは 1⊕X に等しい。

X∧Y　　明らかに X∩Y

X∨Y　　(X⊕Y)⊕(X∩Y) に等しい。

X⊃Y　　(1⊕X)⊕(Y⊕(1⊕X)∩Y)) に等しい。

言いかえれば ¬X⊕Y⊕(¬X∧Y) に等しい。これは真理表を作って見れば容易に確かめうる。

X≡Y は (X⊕Y)⊕1 である。

6. 論理命題の真理表 —— example

真理値は True か False かの二つの値しか取らないから False となる時のみその旨印字し，あらゆる Boolean Variable の組合せに対して True となった場合には Tautology と印字させることにした。

(1)　(P⊃(Q∨R))∧¬(P≡¬R).

P	Q	R
TRUE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	TRUE	TRUE
FALSE	FALSE	TRUE

これは P, Q, R がたとえば True, True, False の値を取った場合に上の Boolean Expression が全体として False になることを示している。そして Expression がこれ以外の組合で False となることはない。

(2)　¬((¬P∧¬Q)∧(P∨R)).

P	Q	R
FALSE	FALSE	TRUE

(3) $(A1 \wedge (A2 \vee (A3 \supset \neg A4))) \wedge (A1 \wedge (\neg A2 \wedge \neg (A3 \supset \neg A4)))$
$\supset (((A5 \equiv A6 \supset A7 \vee A8 \wedge \neg A9) \wedge \neg (A10 \supset A7)) \vee \neg (A10 \vee A9))$.

TAUTOLOGY

(4) $B1 \equiv B2 \supset B3 \vee B4 \wedge B5$.

B1	B2	B3	B4	B5
TRUE	TRUE	FALSE	TRUE	FALSE
TRUE	TRUE	FALSE	FALSE	TRUE
TRUE	TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

この表から上の命題は B1 が True B2 が False の時には常に True となること等が分る。

(5) $(B1 \supset B2) \wedge (B3 \vee (\neg B4 \supset \neg B5) \vee (B5 \supset B4)) \supset B1$.

B1	B2	B3	B4	B5
FALSE	TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE

FALSE	TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE

この表より B1 が True の時にはこの命題は常に成立することが直ちに分る。

7. 数式翻訳との類似性

以上述べた Boolean Expression の翻訳の idea は殆んど並行して数式翻訳の Automatic Coding にも生かせる。普通は四則演算 ＋－×／ と巾乗 ↑の内，↑；×，／（×，／ は同じ order）；＋，－（＋，－ は同じ order）；の順序で演算の優先順位が定まっているから，order を

$$\uparrow > \times = / > + = - > \phi$$

のように入れて考えればよい。二項演算でないマイナス－は↑よりも強い order を入れてやれば良いように思う。

8. ま と め

この翻訳プログラム（Automatic Coding）は一口に言って Boolean Expresslon を頭から読みとっていって，翻訳してよい所が来るとすぐ翻訳して object program に吐き出してしまう逐次翻訳の立場を取るものであり，且つその際に翻訳開始の段階及びどこまで翻訳の程度が及ぶかということを，End Mark 間に定義された order を手掛りに使うものである。

E. M. = End Mark

E. M.	¬	∧	∨	⊃	≡	(	)	•	ϕ
$-\varphi$(E. M.)	8	7	6	5	4	3	2	1	0