J 言語によるブール代数と論理（入門編）

SHIMURA Masato
jcd02773@nifty.ne.jp

2008 年 12 月 8 日

目次

概要

J 言語の優れたブール演算機能を論理演算に用いる。論理は短い 5 つの関数で簡単に表現できる。命題論理の範囲に限定し内容の多くは矢野に依った。

1 J 言語の論理構造

凡そコンピュータ言語と名乗るものは論理に対応している。J 言語は一歩進め、関数型言語として洗練された機能を有している。

Atom（原始関数) の定義 特定の言語に依存しない数学の思想を採用した *APL* 記号を J はキー

ボード記号に移し替えた。この記号は各国の言葉で自由に再定義できる。　*tasu* =: +

動詞 副詞 接続詞　自然言語のスタイルを採用し関数を分類して定義方法を区別した。通常の関数が名詞、動詞をパワーアップする作用素が副詞。文を繋ぐ接続詞、変数やデータが名詞。形容詞はない。

関数型定義 (Tacit definition)　数式に近いエレガントな記法を組み入れた。従来の後ろからシークエンス通りに実行する明示型定義 (Explicit definition) と併用したり、相互に参照することもできる。

1.1　接続詞　ボンド (&) とアトップ (@)

接続詞は動詞を連結して複数の作用を一度に行う。（複合動詞　持って走る）接続詞には右引数を一個取る単項型と *x u&v y* と動詞の左右に引数を取る両項型がある。

次の Fork と組み合わせれば相当複雑な構文も関数のみで表現できる。

1.1.1　Bond &

Bond(&) の機能の一つは数字を連結できることである。(Atop(@) はエラー)。 `0&{`　`1&{`

単項	両項	
u	*u*	
|	↗	↖
v	*v*	*v*
|	|	|
y	*x*	*y*

(~ *p*) ∧ (~ *q*)

```
x v&u y is (v x) u (v y)
```

1.1.2　Atop

単項は&と同一でありどちらを用いても良い。

単項	両項	
u	*u*	
|	|	
v	*v*	
|	↗	↖
y	*x*	*y*

```
3+:@- 7
_8
```

1.2　フックとフォーク

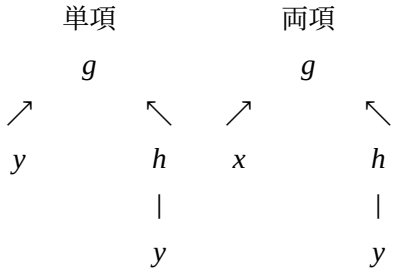
2 の動詞の組み合わせはフック、3 がフォークである。

動詞を連ねると *Train* になる。コンテナの貨物列車である。J は後ろから動詞を 3 組ずつ取っていき、最後が 2 になればその部分はフックになる。

1.2.1　**フック**　Hook

洋服掛けのフックである。右に先に一つの作用をしてから次にかかる。フックは複雑なので余り用いない。

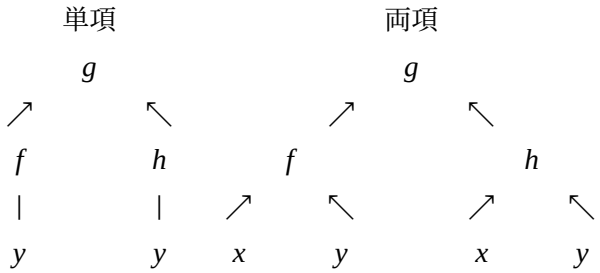
1.2.2　Hook




```
-(+/ % #)    -"1(+/ % #)    NB. 残差を計算する
```

1.2.3　Fork

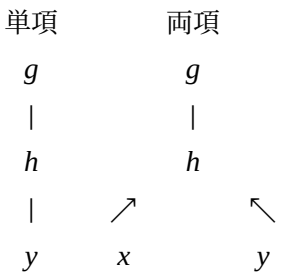
`mean=:+/#%` のように引数を用いない動詞の定義ができる。




```
   (+/%#) >:i.10
5.5
```

1.2.4　Capped fork

`[: u v` は Hook や Fork を掛けないで、通常の後ろから前の定義通りに実行したいときに用いる。



■*George Boole*　(1815-1864) England

イングランド中東部のリンカーンで裕福ではなかった自営業の家に生まれる。ラテン語、ギリシャ語、フランス語、ドイツ語は早くから収得し、16 才から助教師として教え始め、19 才でリーンカーンに自身で学校を開設した。この頃から数学を学び始め、ドゥ･モルガンと交流する。29 才で王立協会会メダルを得て、34 才でアイルランドのコークの *Queens College* の最初の数学教授となり、ここで業績と名声を得て生涯を過ごす。エベレストに名を残す *sir George Everest* の娘マリーに微分を教え始めエベレストの死後結婚し 5 人の娘を授かる。[*1]微分方程式の著作が多いが 1854 年に出版された

An investigation into the Laws of Thought, on Which are founded the Mathematical Theoties of Logic and Probabilities

は、ブール代数として知らる。1938 年にクロード･シャノンがブール代数をリレー回路に応用する方法を示し、スイッチング回路、コンピュータのハード以外にも、データベース検索、グーグルの検索エンジンなどソフト面にも応用されている。

2　ブール代数と J のプリミティブ

■J とブール代数　J のブール代数の機能を大掴みする。

項目　　J	formula	Bool	数値	文字列
AND　　*.	*. | 0 1 0 | 0 0 1 | 0 1 1 is 1 1 only	d=: 0 1 d */ d 0 0 0 1	24 *. 60 120　NB.LCM	⊗

[*1] かの山を測量したときのインド測量局長官

OR +.	+. | 0 1 0 | 0 1 1 | 1 1 1 is 0 0 only	d +./ d 0 1 1 1	24 +. 60 12 NB. GCD	⊗
Not AND *:	*: | 0 1 0 | 1 1 1 | 1 0 reverse AND	d *:/ d 1 1 1 0	⊗	⊗
Not OR +:	+: | 0 1 0 | 1 0 1 | 0 0 reverse OR	d +:/ d 1 0 0 0	⊗	⊗
NOT -.	| −. 0 | 1 1 | 0	-. 0 1 1 0	(i.9)-. 2 3 5 7 0 1 4 6 8 2 3 4 5 -.'ABCDEF' 2 3 4 5	'abcdefg' -. 'aiueo' bcdfg

3 論理

ブール代数をベースに論理関数を作成する。計算は文字列でなく数値演算で行い最後に文字に変換する。表記法は矢野に依る。

		矢野	別の表記法	或 *Software*
そして	*AND*	∧	·	&
或いは	*OR*	∨	∨	|
ない	*NOT*	~	− ¬	!
ならば（条件）		→	⊃	

[*2]

論理演算に用いる関数は AND,OR,NOT の 3 つである。(XOR) は用いない。ここでは J の用法に従い、論理学の日本語の用法は用いない。

		論理関数名	数値演算関数名
AND	∧	*and_logis*	*and_logis*0
OR	∨	*or_logis*	*or_logis*0
NOT	~	*not_logis*	*not_logis*0

```
R0
+-------+-------+
|1 1 0 0|1 0 1 0|
+-------+-------+
 T T F F T F T F
```

項目	J	formula	Bool	論理	Example
AND 論理積 ∧	*.	*. | 0 1 0 | 0 0 1 | 0 1 1 is both 1	d *./ d 0 0 0 1	p q | $p \wedge q$ T T | T T F | F F T | F F F | F 双方が T の場合のみ T	and_logis R0 TTT TFF FTF FFF
OR 論理和 ∨	+.	+. | 0 1 0 | 0 1 1 | 1 1 1 is either 1	d +./ d 0 1 1 1	p q | $p \vee q$ T T | T T F | T F T | T F F | F どちらかが T ならば T	or_logis R0 TTT TFT FTT FFF

[*2] & は省略可能 | pipe

Not Equal (XOR exclusive or) $\underline{\vee}$ ≠ ⊕		0 1 0 0 1 1 1 0	d-.@=/ d 0 1 1 0 reverse equal	p q $p\underline{\vee}q$ T T F T F T F T T F F F	nequal_logis R0 TTF TFT FTT FFF
NOT （論理）否定 ~	-.	−. 0 1 1 0	-. 0 1 1 0	p $\sim p$ T F T F F T F T 補集合となる	not_logis {. R0 TF TF FT FT

3.1 Script

単なる数値演算と TF に戻すタイプの 2 通りを作成した。0 の付く方は論理演算を組み立てるときに便利である。

*and_logis*0 , *and_logis*

```
 and_logis0=: 3 : 0
NB. calc AND
'A0 B0'=: y
A0 *. B0
)

 and_logis=: 3 : 0
NB. calc AND
'A0 B0'=: y
(trans_tf_sub y),.trans_tf_sub A0 *. B0
)
```

```
 or_logis0=: 3 : 0
NB. calc OR
'A0 B0'=: y
A0 +. B0
)

 not_logis0=: 3 : 0
NB. calc Not equal
'A0' =:; y
-. ; y
)

NB. ---------------------
R0=: 1 1 0 0 ; 1 0 1 0
```

not_logis は 1 つの値にかかる。複雑な構文の時の組み込みに応用できる両項関数 picl_not も作成した。

```
 0 1 2 3  pick_pq_not R0
+-------+-------+-------+-------+
|1 1 0 0|1 0 1 0|0 0 1 1|0 1 0 1|
+-------+-------+-------+-------+
  p       q      ~p       ~q
  0       1       2        3
```

4 条件

将を射んとする**ならば**馬を射よ

条件文は 2 つである。これで 5 の用いる関数が出そろった。

	矢野	別の表記法	別名	用いる関数	数値演算用
条件	→	⊃	*IMPLY*	*cond_pq*	*cond_pq*0
双条件	↔	≡	*EQUIV*	*condw_pq*	*condw_pq*0

4.1 条件 →

$p \to q$ は $(\sim p) \vee q$ である

```
  1 1 0 0 ((-.@[) +. ]) 1 0 1 0  NB. use fork
1 0 1 1

   cond_pq0 R0
1 0 1 1

trans_tf_sub 1 0 1 1
TFTT
```

正	逆	裏	対偶
$p \to q$	$q \to p$	$\sim p \to \sim q$	$\sim q \to \sim p$
`cond_pq  R0`	`cond_pq  |. R0`	`cond_pq  -. L:0 R0`	`cond_pq |.-. L:0 R0`
TTT	TTT	TTT	TTT
TFF	TFT	TFT	TFF
FTT	FTF	FTF	FTT
FFT	FFT	FFT	FFT
	rorate	not	not and rotate

```
cond_pq0=: 3 : ' or_logis0 (<-.;{.y),{: y'
```

4.2 双条件 ↔

*3

p と q は等価、p ならば q であり、かつ、q ならば p である。
$p \to q$ は $(p \to q) \wedge (q \to p)$ である。

*3 日本語に馴染みにくい （if and only if）など

↔	=
condw_pq R0 TTT TFF FTF FFT	双条件は *equal*(=) と同じである。 1 1 0 0 = 1 0 1 0 1 0 0 1 condw_pq0 R0 1 0 0 1

```
condw_pq0=: 3 : 0
TMP=: y,-.&.> y  NB. close<-calc<-open
and_logis0 (or_logis0 2 1{ TMP);or_logis0 3 0{TMP
)
```

■J Grammar

演算順序 J は演算順序は後ろから単項または組み合わせ（両項）で行う。記号による優先順位はない。論理では左優先や NOT 優先が多いが TEXT の定義による。紛らわしいときは括弧で明確にする。

副詞 J では関数を動詞と言うが、動詞を引数にする場合は作用素となる。J は作用素を動詞を修飾する副詞として扱う。`1 : 0` と定義する。

並列演算 動詞を連ねて並列演算することもできる。

```
   (+: ; -: ; *: ; %:) 2j_1
+----+------+----+-----------------+
|4j_2|1j_0.5|3j_4|1.45535j_0.343561|
+----+------+----+-----------------+
```

■真理表 出そろったところで 5 関数の真理表を作ってみよう

```
 logis=.  not_logis0&{.;and_logis0;or_logis0;cond_pq0;condw_pq0

   logis  R0
+-------+-------+-------+-------+-------+
|0 0 1 1|1 0 0 0|1 1 1 0|1 0 1 1|1 0 0 1|
+-------+-------+-------+-------+-------+
not     and     or      cond    condw
```

```
|:( L:0) 1 0 1 0 0 0 0 <;.1 |:  logis trans_tf0 R0

 pq naocw  NB.
+--+-----+
|TT|FTTTT|
|TF|FFTFF|
|FT|TFTTF|
|FF|TFFTT|
+--+-----+
```

■pick_not　$(\sim p) \rightarrow q$ などの構文は良く出てくる。これを手短く定義する両項関数を作成する。
左引数は 0/1/2/3 である。これで同時処理が可能である。

```
   R0,-.&.> R0
+-------+-------+-------+-------+
|1 1 0 0|1 0 1 0|0 0 1 1|0 1 0 1|
+-------+-------+-------+-------+
    0       1       2       3
    p       q      ~p      ~q


   (2 1&pick_pq_not) R0
+-------+-------+
|0 0 1 1|1 0 1 0|
+-------+-------+

   cond_pq0&(2 1&pick_pq_not) R0
1 1 1 0
```

5　ドウ・モルガンの法則/de Morgan law

5.1　ドゥ・モルガンの法則

$\sim (p \vee q) = (\sim p) \wedge (\sim q)$
$\sim (p \wedge q) = (\sim p) \vee (\sim q)$

p, q の真理集合を P, Q とすれば
$p \vee q \to P \cap Q$
$\sim (p \vee q) \to (P \cap Q)'$
である。

■August De Morgan　1806-1871

父の東インド会社の勤務地のインドで生まれ、7 ヶ月でイングランドに戻る。母は antilogalism で知られる *James Dodson* の係累であった。父は 10 才の時になくなり、母は南西イングランド各地に移り住んだため、数学の才能は 14 才まで見いだされなかった。国教会の熱心な信者であった母は彼が聖職に就くことを望み、*Oxford* で古典を学んだが彼は別の道を探し、16 才からケンブリッジの *trinity* で学ぶ。主に *algebra* と論理学の改革に取り組み、運動は得意でなかったがフルートでも有名であった。国教会派の神学の試験（必須）と彼の論理学に相違があり MA に進めず、法学を学び法廷法律家になろうとロンドンに戻ったが数学を教える方により熱心になった。22 才でロンドン大学の教授に就任。数学の入門講義 *On the study of mathematics* は現在でも U.S.A. で出版されている。1837 年に気のあった同僚の娘 *Sophia Elizabeth* と結婚し 3 人の息子と 4 人の娘を授かった。*London Mathematics Society* を創設し、初代の代表になった。ハミルトンとも親好があった。*Oxford, Cambridge, Royal Society* とは距離を置き、ロンドン塔やウエストミンスター寺院を訪れることはなかった。

アリストテレス以来の論理学にも改革の時が訪れた。

$\sim (p \vee q)$	not_logis0&or_logis0 R0 0 0 0 1 not_logis0&or_logis0 trans_tf0 R0 pq* --- TTF TFF FTF FFT

$(\sim p) \wedge (\sim q)$	1 1 0 0 (*.& -.) 1 0 1 0 0 0 0 1 not_logis0 L:0 R0 +-------+-------+ |0 0 1 1|0 1 0 1| +-------+-------+ and_logis0&(not_logis0(L:0)) R0 0 0 0 1 and_logis0&(not_logis0(L:0)) trans_tf0 R0 pq* ---- TTF TFF FTF FFT

$\sim (p \wedge q)$	not_logis0&and_logis0 R0 0 1 1 1 not_logis0&and_logis0 trans_tf0 R0 TTF TFT FTT FFT
$(\sim p) \vee (\sim q)$	1 1 0 0 (+.& -.) 1 0 1 0 0 1 1 1 (not_logis0(L:0)) R0 +-------+-------+ |0 0 1 1|0 1 0 1| +-------+-------+ or_logis0&(not_logis0(L:0)) R0 0 1 1 1 or_logis0&(not_logis0(L:0)) trans_tf0 R0 TTF TFT FTT FFT

5.2　トートロジー/tautology

それを含む命題の真偽を問わず T となる命題をブィットゲンシュタイン由来の**恒真性**という。

■金城湯池　多くの教科書では 30 近いトートロジーを ··· の法則と名付けて紹介している。ここではトウトロジーの照査を今までの動詞で行う。

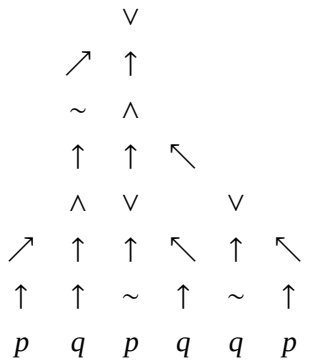
$(p \wedge q) \rightarrow (p \leftrightarrow q)$ は全て T となる。（トートロジー）

$(\sim (p \wedge q)) \vee (((\sim p) \vee q) \wedge (\sim p) \vee p)$ である

$((p \wedge q) \Rightarrow (p \leftrightarrow q)$ と表す)

$(p \wedge q) \Rightarrow (p \leftrightarrow q)$ は

$(\sim (p \wedge q)) \vee (((\sim p) \vee q) \wedge (\sim p) \vee p)$ である



p	q	$p \wedge q$	$p \leftrightarrow q$	$\rightarrow$
T	T	T	T	T
T	F	F	F	T
F	T	F	F	T
F	F	F	T	T

OR は 0,0 以外は 1 なので懐が深い

```
 tt0=.  and_logis0;condw_pq0
   tt0 R0
+-------+-------+
|1 0 0 0|1 0 0 1|
+-------+-------+

 cond_pq0   tt0 R0
1 1 1 1
```

```
 cond_pq0 trans_tf0  tt0 R0
awT
----
TTT
FFT
FFT
FTT
```

双条件タイプのトートロジー
$((\sim p) \to q) \leftrightarrow (p \vee q)$
$((\sim p) \to q) \Leftrightarrow (p \vee q)$　と著す
$((\sim p) \to q) = (p \vee q)$ と同じ

```
condw_pq0 (cond_pq0  (2 1&{  R0,  -.&.> R0));or_logis0 R0
+-------+-------+
|1 1 1 0|1 1 1 0|
+-------+-------+

condw_pq0 (cond_pq0  (2 1&{  R0,  -.&.> R0));or_logis0 R0
1 1 1 1
```

$p \Leftrightarrow q$　の時は p, q は相互に必要十分条件である。

6 推論

6.1 2 文の推論

推論エンジンには今までに作成した小型のスクリプトを組み合わせる。このため *range_suspect* は単にみせ映えのために並べる単機能の副詞型とする。

$p \to q$ が真で、$\sim q$ も真なら、$\sim p$ は真と考える。次の真理表ができる。

p	q	$p \to q$	$\sim q$	$\sim p$
T	T	T	F	F
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

最終行が TTT で通っている。
$p \to q$　犬は四つ足である。
$\sim q$　是は四つ足でない
$\sim p$　是は犬ではない

```
  logis=. cond_pq0;(3 2&pick_pq_not)
   logis  R0
+-------+-------+-------+
|1 0 1 1|0 1 0 1|0 0 1 1|
+-------+-------+-------+

   logis trans_tf0 R0
pqcnn
------
TTTFF
TFFTF
FTTFT
FFTTT   NB. all T
```

6.2 3 段論法

p, q, r の 3 文の推論の例。組み合わせは 2 ビットに戻して得られる。(さすがブール!)
ここでも推論のマイクロエンジンを用いて様々な命題に対応する。

p, q, r の組み合わせ

p, q, r の条件文の例。
$p \to q$ が真で、$q \to r$ も真なら、$p \to r$ は真である。次の真理表ができる。
右側の 3 列が全て T の場合が正となる。

$p \to q$

$q \to r$

$p \to r$

p	q	r	$p \to q$	$q \to r$	$p \to r$
T	T	T	T	T	T
T	T	F	T	F	F
T	F	T	F	T	T
T	F	F	F	T	F
F	T	T	T	T	T
F	T	F	T	F	T
F	F	T	T	T	T
F	F	F	T	T	T

```
|. 2 2 2 #: i.8
1 1 1
1 1 0
1 0 1
1 0 0
0 1 1
0 1 0
0 0 1
0 0 0

A0=. {@|:@|. 2 2 2 #: i.8
+---------------+---------------+---------------+
|1 1 1 1 0 0 0 0|1 1 0 0 1 1 0 0|1 0 1 0 1 0 1 0|
+---------------+---------------+---------------+

  logis=. cond_pq0&(0 1&{);cond_pq0&(1 2&{);cond_pq0&(0 1&{)
   logis A0
+---------------+---------------+---------------+
|1 1 0 0 1 1 1 1|1 0 1 1 1 0 1 1|1 1 0 0 1 1 1 1|
+---------------+---------------+---------------+
   logis trans_tf0 A0
pqrccc
------
TTTTTT NB. T
```

```
TTFTFT
TFTFTF
TFFFTF
FTTTTT NB. T
FTFTFT
FFTTTT NB. T
FFFTTT NB. T
```

7　公理

ラッセル・ホワイトヘッド	1. $(p \vee p) \to p$ 2. $q \to (p \vee q)$ 3. $(p \vee q) \to (q \vee p)$ 4. $p \vee (q \vee r) \to q \vee (p \vee r)$ 5. $(q \to r) \to ((p \vee q) \to (p \vee r))$	
ヒルベルト・アッカーマン	1. $(p \vee p) \to p$ 2. $p \to (p \vee q)$ 3. $(p \vee q) \to (q \vee p)$ 4. $(p \to q) \to ((r \vee p) \to (p \vee r))$	ラッセル (4) は他の公理から導出できる
ルカジュビッツ	1. $p \to (q \to p)$ 2. $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$ 3. $(-p \to -q) \to (q \to p)$	フレーゲの 6 の公理を簡約したもの

幾つかを確認してみよう

■$(p \vee p) \to p$

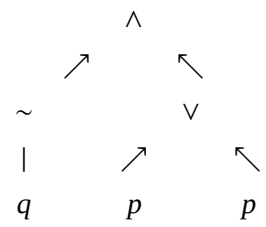
```
r0=. and_logis0&(0 0& pick_pq_not); 0&pick_pq_not
   r0 R0
```

```
+-------+-------+
|1 1 0 0|1 1 0 0|
+-------+-------+

 cond_pq0  r0 R0
1 1 1 1
```

■$q \to (p \vee p)$

```
r1=.([: > 1&pick_pq_not);or_logis0
   r1 R0
+-------+-------+
|1 0 1 0|1 1 1 0|
+-------+-------+

 cond_pq0  r1 R0
1 1 1 1
```

$q \to (p \vee q)$ *is* $(\sim q) \wedge (p \vee q)$



■$(p \vee q) \to (q \vee p)$

```
 h2=.  or_logis0;or_logis0&|.
   h2 R0
+-------+-------+
|1 1 1 0|1 1 1 0|
+-------+-------+

 cond_pq0  h2 R0
1 1 1 1
```

8 Reference

矢野健太郎「新しい数学」 岩波新書 1966
沢田允茂「現代論理学入門」 岩波新書 1962

付録 A 準備

position	a. i. 'TF' 84 70	T は a. で得た文字列の 84 番目
sequential machine	a=. 'T F' > ;: a T F	文字列操作の関数
antibase #:	2 2 #: i.4 0 0 NB. 0 0 1 NB. 1 1 0 NB. 2 1 1 NB. 3	{@|:@ |. 2 2 #: i.4 +-------+-------+ |1 1 0 0|1 0 1 0| +-------+-------+