



コンピュータよもやま話

コンピュータの基礎



目次

- ハードウェアの話
- カーネルの話
- 形式言語の話
- 関数型言語とラムダ計算の話



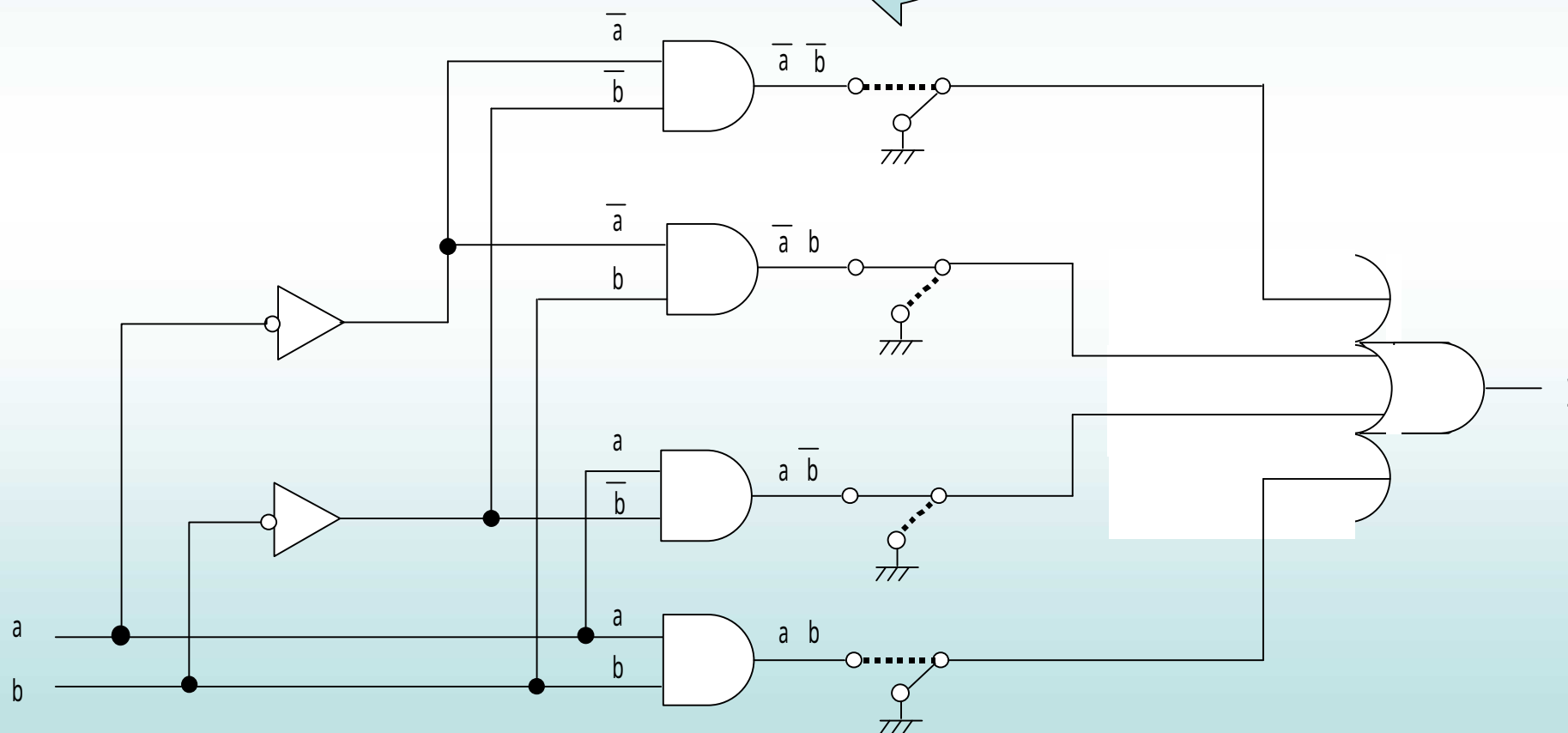
ハードウェアの話

- コンピュータの必須機能を絞り込む
 - フローチャートに従って実行する能力
 - 始点、終点、演算結果の記憶、比較結果に応じた分岐で構成される有向グラフ
- 実現には真理値表から組合せ回路への機械的な変換を利用
 1. NOT、AND、ORから論理演算、四則演算を構成(ALU)
 2. 組合せ回路から記憶回路または順序回路を構成(レジスタ)
 3. プログラムカウンタと条件付きジャンプ(流れの制御)

2入力XORの例

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

真理値表から組合せ回路への
機械的な変換



加算 (算術演算)

a[n] b[n] a[1] b[1] a[0] b[0]
c[n] c[n-1] c[1] c[0] 0

s[n] s[1] s[0]

c[n-1]	a[n]	b[n]		c[n]	s[n]
0	0	0		0	0
0	0	1		0	1
0	1	0		0	1
0	1	1		1	0
1	0	0		0	1
1	0	1		1	0
1	1	0		1	0
1	1	1		1	1

c[n-1] は下位ビットからの桁上がり (carry) を意味する.

ラムダ計算で論理値や整数値とその演算を表現し組み合わせていく発想と同じ

条件付きジャンプ(流れの制御)

条件付きジャンプが可能になれば条件分岐と繰り返しが可能になる。(チューリング完全につながる)
 下図にその原理を示す。正確を期すならばタイミングの説明が必要になるがタイミングチャートは省略する。
 ここでは特別なハードウェアが必要ないということを説明したい。

図の説明

PC (Program Counter)

次に実行される命令のアドレスを記憶するレジスタ。自動的に次のアドレスに更新される。
 ジャンプ命令実行中はジャンプ命令の次のアドレスを指している。

PC + offset

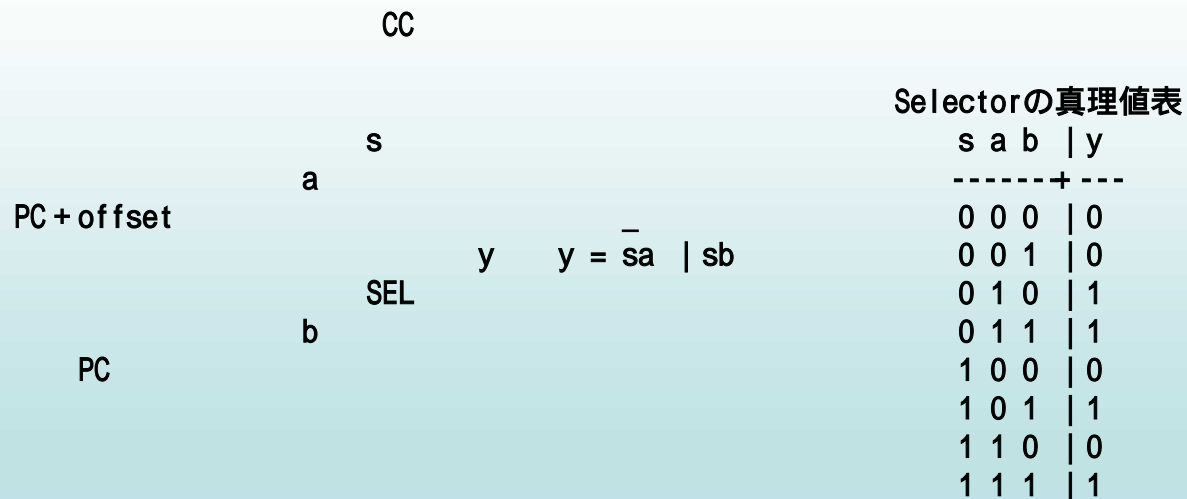
条件成立時にジャンプする命令のアドレス。

CC (Condition Code Register)

比較命令の演算結果を記憶するレジスタ。
 条件付きジャンプ命令の直前には比較命令がありCCの内容はジャンプ先の命令か次の命令かの選択に使われる。

SEL (Selector)

選択信号により入力信号のいずれかを選んで出力するハードウェア。
 図では1個しか描かれていないが32ビットならば32個必要になる。





非ノイマン型コンピュータ

- データフロー型コンピュータ
- 要求駆動型コンピュータ



カーネルの話

- プロセスとスレッド
- ファイルシステム
- メモリ管理（仮想記憶）



形式言語の話

- コンパイラとカーネルは特別なプログラム
- チョムスキー階層

- 句構造文法
- 文脈依存文法
- 文脈自由文法
- 正規文法

コンパイラは文法による語の生成の逆操作を行うプログラムである。

句構造文法 (phrase structure grammar)

生成規則に制約を設けない文法である.

N:非終端記号 T:終端記号 P:生成規則

文法の例 -- a が 2 の巾乗個連結された言語を定義

$N = \{ S, A, B, C, D, E \}$

$T = \{ a \}$

$P = \{$

$S \rightarrow ACaB,$

$Ca \rightarrow aaC,$

$CB \rightarrow DB,$

$CB \rightarrow E,$

$aD \rightarrow Da,$

$AD \rightarrow AC,$

$aE \rightarrow Ea,$

$AE \rightarrow$

$\}$

開始記号 : S

この生成規則は左辺よりも右辺の長さが短い

この生成規則は左辺よりも右辺の長さが短い

生成過程の例

前述の句構造文法による生成過程の例

S			ADaaaaB	[5]	aD	Da
ACaB	[1]	S	ACaaaaB	[6]	AD	AC
AaaCB	[2]	Ca	AaaCaaaB	[2]	Ca	aaC
AaaDB	[3]	CB	AaaaaCaaB	[2]	Ca	aaC
AaDaB	[5]	aD	AaaaaaaCaB	[2]	Ca	aaC
ADaaB	[5]	aD	AaaaaaaaaCB	[2]	Ca	aaC
ACaaB	[6]	AD	AaaaaaaaaE	[4]	CB	E
AaaCaB	[2]	Ca	AaaaaaaaaEa	[7]	aE	Ea
AaaaaCB	[2]	Ca	AaaaaaaaaEaa	[7]	aE	Ea
AaaaaDB	[3]	CB	. . .			
AaaaDaB	[5]	aD	AaEaaaaaaa	[7]	aE	Ea
AaaDaaB	[5]	aD	AEaaaaaaa	[7]	aE	Ea
AaDaaaB	[5]	aD	aaaaaaaa	[8]	AE	

文脈依存文法 (context sensitive grammar)

文脈依存文法とは句構造文法の生成規則に
右辺は左辺以上の長さでなければならない
という制約を設けた文法である。

文法の例 -- $a^{**n} \cdot b^{**n} \cdot c^{**n}$ (a^{**n} は a の n 個連結を表す)

$N = \{ S, B \}$

$T = \{ a, b, c \}$

$P = \{$

$S \rightarrow aBSc,$

$S \rightarrow abc,$

$Ba \rightarrow aB,$

$Bb \rightarrow bb$

$\}$

開始記号 : S

文脈自由文法 (context free grammar)

生成規則に

$A \rightarrow$

という制約を設けた文法である。

ただし A は非終端記号であり、 ϵ は空を除いた非終端記号または終端記号の列である。

左辺は 1 個の非終端記号でなければならないことが文脈自由文法を特徴付けている。

プッシュダウンオートマトン (スタックを持つ有限オートマトン)

文法の例 -- $a^n \cdot b^n$ (a^n は a の n 個連結を表す)

$N = \{ S \}$

$T = \{ a, b \}$

$P = \{ S \rightarrow aSb, S \rightarrow ab \}$

開始記号 : S

正規文法 (regular grammar)

生成規則に

$A \rightarrow a$ または $A \rightarrow aB$ ($A \rightarrow Ba$)

という制約を設けた文法である。

ただし A と B は非終端記号であり, a は終端記号である。

左辺は 1 個の非終端記号でなければならず, 右辺も限定されたパターンのみ許されることが, 正規文法を特徴付けている。

正規言語を受理する仮想機械 (有限オートマトン) には現在の状態以外の記憶がない。

文法の例 -- $a^m \cdot b^n$ (a^n は a の n 個連結を表す)

$N = \{ S, A \}$

$T = \{ a, b \}$

$P = \{ S \rightarrow aS, S \rightarrow aA, A \rightarrow bA, A \rightarrow b \}$

開始記号 : S



関数型言語とラムダ計算の話

手続き型言語

- 抽象化レベルが低い
 - ハードウェアの効率的な利用が可能
 - 記述能力が低い (how)
- 変数の書き換え
- フローチャートで表現できるグラフ

関数型言語

- 抽象化レベルが高い
 - ハードウェアの効率的な利用は困難
 - 記述能力が高い (what)
- 副作用がない
- ラムダ計算



型なしラムダ計算

ラムダ式とは

変数 x, y, \dots | ラムダ抽象 $x.M$ | 関数適用 MN
である。ここで x, y, \dots は変数 M, N, \dots はラムダ式とする。

ラムダ計算とは β -redexを見つけて簡約することである。

ここで β -redexとは

$(\lambda x.M)N$

を指す。

関数型言語でよく使われる無名関数、高階関数、遅延評価などの理解に役立つ。

ハードウェアの話では論理値や整数を基本的な要素と考えるのが自然。しかしラムダ計算では基本的な要素はラムダ式のみ。

ラムダ式の表現能力

ラムダ式がチューリング完全 (Turing-Complete) であることの概要説明

いろいろなデータをラムダ式だけで表現可能

条件式を表現可能

Y-Combinator (不動点演算子)

再帰関数定義可能

繰り返し

論理値

true $tf.t$

false $tf.f$

if $pxy.pxy$

and $pq.pq<false>$

or $pq.p<true>q$

not $p.p<false><true>$

$<false>$ は $tf.f$ に置き換えて計算することを意味する。
一般的に使われる記号ではない。

整数

0 $sz.z$

1 $sz.sz$

2 $sz.s(sz)$

3 $sz.s(s(sz))$

succ $nsz.s(ns)$

plus $mnsz.ms(ns)$

再帰関数

Y $f.(x.f(xx))(x.f(xx))$

YTuring ($xf.f(xxf))(xf.f(xxf))$

ラムダ式の加算

記号 = をラムダ計算とは無関係な文字列定義 <name> を文字列置換と解釈する。

```
plus = mnsz.ms(nsz)
2     = sz.s(sz)
3     = sz.s(s(sz))
5     = sz.s(s(s(s(sz))))
```

とすると次のような加算結果が得られる。

<plus><3><2> <5>

<plus><3><2> <xxx> は対応するラムダ式に置換
= (mnsz.ms(nsz)) (sz.s(s(sz))) (sz.s(sz))
(nsz. (sz.s(s(sz))) s(nsz)) (sz.s(sz))
sz. (sz.s(s(sz))) s ((sz.s(sz)) sz)
sz. (z.s(s(sz))) ((sz.s(sz)) sz)
sz.s(s(s((sz.s(sz)) s)))
sz.s(s(s((z.s(sz)) z)))
sz.s(s(s(s(sz))))

部分適用 (高階関数の応用)

ラムダ計算と高階関数の自然な結び付きを示す例として**部分適用**を取り上げる。
部分適用の実例として

$$(mn.m+n)3 \quad n.3+n$$

を示す。ただし + や 3 など略記法を使っていることに注意。

意味は

2変数関数 $mn.m+n$ を引数 3 に**部分適用**すると1変数関数 $n.3+n$ になると表現できる。

$$\begin{aligned} & (mn.m+n)3 \quad \text{略記法} \\ = & \langle \text{plus} \rangle \langle 3 \rangle \quad \langle \text{xxx} \rangle \text{ は対応するラムダ式} \\ = & (mnsz.ms(nsz))(sz.s(s(sz))) \\ & nsz.(sz.s(s(sz)))s(nsz) \\ & nsz.(z.s(s(sz)))(nsz) \\ & nsz.s(s(s(nsz))) \\ = & n.3+n \quad \text{略記法} \end{aligned}$$

この結果が引数に3を加算する関数となっていることを確認するために結果の関数を2に適用して

$$(n.3+n)2 \quad 5$$

となることを示す。

$$\begin{aligned} & (n.3+n)2 \quad \text{略記法} \\ = & (nsz.s(s(s(nsz))))(sz.s(sz)) \\ & sz.s(s(s((sz.s(sz))sz))) \\ & sz.s(s(s((z.s(sz))z))) \\ & sz.s(s(s(s(sz)))) \\ = & 5 \quad \text{略記法} \end{aligned}$$



代入操作

代入操作の定義 (プログラミング言語の基礎理論 大堀淳著 p.18)

形式	条件	結果
(a) $x[x:=N]$		N
(b) $y[x:=N]$	$x \neq y$	y
(c) $(y.M)[x:=N]$	$x = y$	$y.M$
(d) $(y.M)[x:=N]$	$x \neq y \quad y \notin FV(N)$	$y.M[x:=N]$ [注1]
(e) $(y.M)[x:=N]$	上記以外	$z.M[y:=z][x:=N]$ [注2]
(f) $(M1M2)[x:=N]$		$M1[x:=N]M2[x:=N]$

[注1] $y \notin FV(N)$ は y が N の自由変数ではないことを意味する。

[注2] 自動的に変換されることに注意。変数 z はまったく新しい名前が選ばれる。

計算例

$(y.xy)[y:=z]$	$y.xy$	(c)の場合
$(y.xy)[x:=z]$	$y.zy$	(d)の場合
$(y.xy)[x:=yz]$	$x.yzx$	(e)の場合

変換

変換の定義 (計算論 高橋正子著 p.65)
記号 \rightarrow は 変換を意味するものとする.

規則	例
(A) $(x.M)N \rightarrow M[x:=N]$	$(x.x)M \rightarrow M$ $(x.y)M \rightarrow y$ $(x.y.x)z \rightarrow y.z$
(B) $M \rightarrow N$ ならば $x.M \rightarrow x.N$	$x.(y.y)x \rightarrow x.x$
(C) $M \rightarrow N$ ならば $MP \rightarrow NP$	$(a.b)cd \rightarrow bd$
(D) $M \rightarrow N$ ならば $PM \rightarrow PN$	$x((x.y)z) \rightarrow xy$



最左簡約の優先

複数の簡約結果がある場合は最左簡約を採用する.

最左簡約が停止しないラムダ式はどれを選んでも停止しない.
もし正規形を持つならばどれを選んでも同じ正規形となる.

計算例

$$(x.y)((x.xx)(x.xx)) \quad y$$
$$(x.y)((x.xx)(x.xx)) \quad (x.y)((x.xx)(x.xx))$$