

2006年度夏学期

シケプリ

de

情報

第1章 情報の学び方

ここは授業では扱われませんでした。と、いうわけで、悪いんですがここは各自教科書を読んで勉強してください。自分自身まだちゃんと読んでないんで何とも言えないんですが、まあ授業でやってないんだからそんなに突っ込んで出題されないんじゃないですか。教科書読んで疑問点などあればビシバシメールください。答えられたら答えます。私は植田なにがしとかいう情報機器オタクではないので、「わかりません」って返事が返ってきても殺意とか抱かないでください。とりあえず全力でお答えしようとはしますから。

もちろん第1章以外のところでも疑問点あればビシバシメールください。それでは、次のページから本格的な情報のお勉強に入りましょう。2章あたりはなんだか堅苦しくダラダラといっぱい書いてあってわかりにくいかもしれませんが（この文章は2章、3章を作り終えてから書きました）、だんだんくだけてきます。途中から著作権も肖像権もどうでもよくなってきます。だから極力このシケプリの外部流出は避けたいところです。まあ、こんなことはどうでもいいのでさっさと勉強始めましょう。はい、次のページへどうぞ。

第2章 情報の表現

【情報表現の様々な側面】

◆道案内における情報表現の多様性

相手が日本人 かつ 目的地が近い … 「2つ目の角を左」など自然言語による道案内

相手が外国人 かつ 目的地が近い … 身振り手振りのジェスチャーを駆使

目的地がやや遠い … 地図を書いた説明

さまざまな表現手段を状況に応じて使い分ける必要がある

◆言語による情報表現の違い

自然言語 → 人間が日常的に使用している言語

人工言語 → プログラミング言語のように人工的に作られた言語

◆情報の説明の仕方の違い

手続き的表現 … 時間を追った手順を説明（第5, 6章のアルゴリズムに関連）

Ex. 駒場図書館の位置の説明

「正門を入れて右、左、右と進めば、工事用の白い壁に突き当たります。

右へ曲がり壁沿いに進んで行けば図書館の『開館中』の看板が見えるはずです。」

宣言的表現 … 対象間の関係や対象の属性を説明（第4章のデータモデルに関連）

Ex. 駒場図書館の位置の説明

「コミュニティプラザの南隣にある、工事用の白い壁に囲まれた建物です。」

◆情報の表現のされ方の違い

記号表現 … 与えられた記号の集合と、それらを解釈するための規則体系

Ex. 数学の記号や数式 日本語 英語 楽譜

パターン表現 … 構成要素間の時空間パターン

Ex. 地図（地図記号は記号表現） ジェスチャー 発声の長短・強弱

たとえば日本語という記号表現の規則体系を理解していない外国人でも、

ジェスチャーなどのパターン表現は理解可能。

◆他の側面

デジタル／アナログによる表現の違い（後述）

情報量から見る側面（後述）

【情報表現とモデル】

◆モデル … 単純化／抽象化された 事物／事象／概念

目的によって異なるモデル化が必要

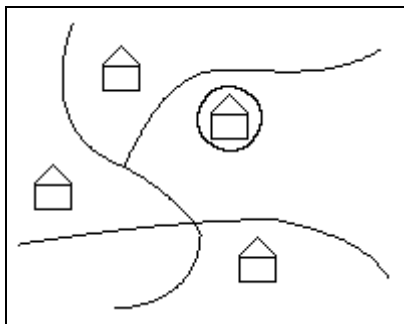


図 2-1

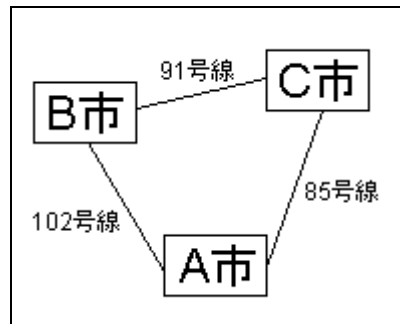


図 2-2

徒歩で移動する場合 → 図 2-1 の方が便利

車で移動する場合 → 図 2-2 の方が便利

◇ジェット旅客機のモデルの場合

コスト削減のためモデルでの実験

風洞実験を行う場合 → 実物と同じ材料でモデルを作るべき

外観を見たい場合 → 加工・修正のしやすい材料を採用すべき

◆モデル表現形式の例

①表 (table)

こみいった事柄を整理 … モデル化の重要な役割の 1 つ

歴史年表／貸借対照表／成績表 等

パソコンなどでの表計算ソフト (Excel 等) の利用も一般的

②図

何らかの目的で描いた 2 次元図形 (絵画・スケッチも含む)

人間の思考・推論を支援／拡張する … ベン図 等

設計図／地図 等

③グラフ ([] 内は図 2-2 との対応を示す)

ノード (node) [市名の書かれた長方形] と

エッジ (edge) [道路を表す線分] から構成される

ラベル [～号線] 付きのエッジもある → ラベル付きグラフ

方向を持つエッジは有向エッジ、弧と呼ばれる

道路ネットワーク／組織図／意味ネットワーク／工場の作業工程 等

【情報表現で考慮すべき事】

◆情報表現の受け手側

日常生活に存在する多くの表現を適切に理解・解釈しなければならない

◆情報表現のデザイン側

多くの表現手段の中から目的・状況に応じて適切なものを選択しなければならない

◇3つの考慮すべき点

- ①表現の**対象** … 物理的実体・抽象的アイデア・思考の方法 等
- ②表現の**目的** … 他者への伝達・自身のアイデアの整理・効率的な問題解決 等
- ③表現の**方法** … 表現に関わるコストや目的に照らして、よりよい方法を選択

【ピクトグラム—記号と意味】

◆記号が表す2側面

意味されるもの—シニフィエ / 意味するもの—シニフィアン

◆サービスエリアのピクトグラム (図 2-3)

抽象化された図形を用いてデザイン

→ 視覚的に認知されやすいように工夫

ドライバーが瞬時に表示内容を認識できるように…

図 2-3



◇記号表現とパターン表現 (前述) の混在

記号表現 … "Parking"の頭文字「P」の利用、距離の尺度に数字を利用

パターン表現 … 「ナイフとフォーク」をモチーフにして表現、矢印で方向を示す

◇3つの考慮すべき点 (前述) を検証

- ①表現の対象 … サービスエリア
- ②表現の目的 … 高速道路を走るドライバーにサービスエリアの所在を知らせる
- ③表現の方法 … ピクトグラムとして図で表現

◇記号が表す2側面 (前述) を検証

意味されるもの … サービスエリア

意味するもの … 「P」や、「ナイフとフォーク」などの図記号

◇図記号の修辭法

提喻 … ある事物を表現する時、それと意味的包含関係のある事物を代用

Ex. 「小町」で「美人」を表す、「花」で「桜」を表す

→ 「サービスエリア」を表現するのに「ナイフとフォーク」の図を使用

隠喩 … ある事物を、「~のようだ」などの形を用いず、直接他のもので表現

Ex. 「時は金なり」、「俺の胃袋は宇宙だ。」

→ "ゴミ箱"アイコン … 「ゴミを捨てる」という行為の暗喩

◆解釈の枠組み

「シニフィアン」から「シニフィエ」へ対応づけるための枠組み（規則）

文脈や状況、文化的背景によって大きく変化

Ex. 車両通行禁止を表す標識—日本と欧州で差がある（図 2-4）

→ 記号の恣意性 … 日本では○=肯定、×=否定と「対応づけられた」にすぎない

図 2-4
「すべての車両通行禁止」



◆日本語文字コード

「文字」と「計算機上の符号（数値）」を対応づけるための枠組み

JIS、シフト JIS、EUC など、異なった日本語文字コードの混在

文字コード	4547	9366	376C	8C8C
JIS	吐	荔	血	・
シフトJIS	(なし)	吐	(なし)	血

図 2-5

文字化け … パソコン上のプログラムの想定と違う文字コードが使用されると起こる

Ex. 「吐血」という文字データの入ったファイルをシフト JIS に基づいて作成

→ パソコンには「9366 8C8C」と記録される（本当はさらに 0 と 1 に直される）

→ そのファイルを読み込むプログラムが JIS に基づいた処理を行った

→ 文字化けの発生 … 「荔・」と表示される

文字コードの標準化・統一化が望ましいが、歴史的経緯、利害関係から困難も多い

【アナログとデジタル】

◆コンピュータでの情報の表現

「0」と「1」の2種類の記号（ビット）の組み合わせ（ビット列）で表現

= 連続量（アナログ）としてではなく、有限個の離散的な値（デジタル）として表現

◇表現できる数値はコンピュータに依る

システムが 16 ビットでは 0~65,535、32 ビットでは 0~4,294,967,295 まで表現可能

◆アナログ表現とデジタル表現

◇アナログ表現

無限の精度を必要とするため、データの複製は元のデータの近似にすぎない

レコード／オーディオテープ／ネガフィルム／ビデオテープ 等

◇デジタル表現

ある情報に対して一定の間隔の尺度を導入し、その尺度の値に近似して表現

Ex. 気温を表現するのに 0.1°C 間隔で尺度を設定

→ 実際には 19.870427...°C の場合も 19.9°C と表現される

精度が有限で近接する量と明確に区別できる → 複製時にデータが劣化しにくい

情報コンテンツの著作権保護への問題をもたらす

◆アナログ→デジタルの変換

情報を離散化する間隔を選択し、表現する必要がある

◇量子化

もともと連続量であった情報をデジタルの形で表現する作業（標本化の次の作業）
用途によって詳細度を定める … 一般に人間の知覚でその差が判別不可であれば十分

▽コンピュータディスプレイ装置

赤（R）、緑（G）、青（B）を混色した RGB 形式を用いている

各々256種類の異なる色で表現（[黒] 暗0 ←-----→ 256 明 [赤、緑、青]）

$256 \times 256 \times 256 = 16,777,216$ 色を表示できる

▽音楽 CD

量子化のために 16 ビット（16 文字）用いている

音の振幅を 65536 の段階に分類している

◇標本化

連続量の情報のある間隔ごとに抽出する作業

標本空間 … 対象の情報が定義される時間や領域

Ex. 1 日の気温変化の情報の標本空間 → 時刻 t

標本化の間隔はデータの利用目的により変化

Ex.

1 時間毎に気温を記録すればその 1 日の気温の推移をとらえられる

1 日毎に気温を記録すればその月の気温の推移をとらえられる

1 ヶ月毎に気温を記録すれば年間の気温の推移をとらえられる

◇標本化定理（ちょっとややこしい。説明下手なんで、わかんなかったらメールを。）

フーリエ解析をご存じですか？

ほとんどあらゆる関数が周期関数の和として表せる、ってことなんですけど。

周期関数っていったら、sin とか cos とか、

ある一定の周期で同じ変化を繰り返す関数ですよ。

あの、波みたいになってるやつです。

で、今から標本化しようとするアナログ量 F が、

時間 t の関数として $F = f(t)$ と定義されることにします。

そうすると、この F も、いろんな周期関数の和で表すことができます。

A っていう周期関数、 B っていう周期関数、 C っていう…

これらの周期関数の波を重ね合わせると F の形になるってわけです。

で、 A とか B とかの周期はそれぞれ違うんですが、そのうち最大の周期を T とします。

このときその周期関数の周波数 W は $W = 1/T$ (Hz) で表されます。

周波数は、1 秒間に何回繰り返すか、という値です。

そして、次に掲げるのがシャノンによって導かれた標本化定理と呼ばれるものです。

標本化を行う際に、
 $1/2W (= T/2)$ 未満の間隔で対象となる情報を標本化すれば、
元のアナログ関数 F を完全に復元できることが保証される。

たとえば、周期が 0.0025 秒の周期関数があるとします。
この周期関数の周波数は 400Hz です。
この関数を完全に復元するためには $1/800 (= 0.00125)$ 秒未満の間隔で標本化、
すなわち、1秒間に 800 回より多くの標本化を行えばいいわけです。

今度は逆に、標本化を行う間隔を t_0 と定めるとします。
このとき、 $1/2t_0$ を「ナイキスト周波数」と呼びます。
上で扱った W の値がこのナイキスト周波数以上であると、
完全な復元を行うことは出来ません。

(内容としては上の標本化定理と同じようなことを言っています。)

では、 $1/2W$ 以上の間隔で標本化する
(=元の関数に、ナイキスト周波数以上の周波数をもつ周期関数が含まれている)
と、どうなるのか。「エイリアシング」と呼ばれる現象が生じます。

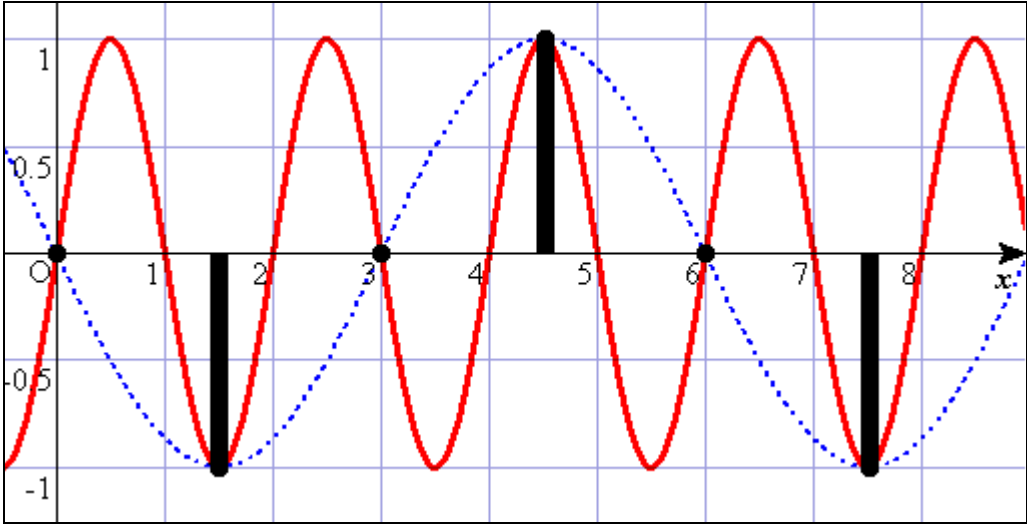


図 2-6

赤の実線が元のデータです。周期は2なので、1未満の間隔で標本化の必要があります。にもかかわらず、1.5という間隔で標本化してしまった場合について考えます。黒で表されるデータが得られます。これを再びアナログ量として再現したいのですが、元の赤のデータ以外にも、青の破線で表されるデータも考えられます。そこで、コンピュータが、青い、誤ったデータを表示してしまう、これが「エイリアシング」です。

同じことですが、

<http://mars.elcom.nitech.ac.jp/java-cai/signal/sampling2.html>

<http://www.teu.ac.jp/hiha/SignalProcessing/aliasing/aliasing.html>

ここでもエイリアシングについて説明がされていますので、参考までに。

画像においてもエイリアシングは起こり得ます。

そんなに重要じゃないと思いますが、教科書 p.25 参照です。

さて、標本化定理の説明はこんなもんですかね。

もとの書式に戻りたいと思います。

◇音楽 CD における標本化

高い周波数の音も記録するのであれば、かなりの高周波での標本化が必要

→ 人間に聞き取れる範囲の周波数でOK

→ 44.1kHz、つまり $1/44100 = 0.0000227$ 秒の間隔で標本化している

◇標本化定理のまとめ

標本化定理は、 成分となる周期関数の最大周波数に応じた適切な細かい標本間隔を用いれば、 元のアナログ量を完全にデジタル量で扱えることを数理的に保証している。
--

第3章 情報の伝達と通信

【情報の伝達】

◆ AさんがBさんに伝える情報

Aさんが持っている情報をBさんに伝える場合をいくつかの状況にわけて考える

①手紙で送る

事前：Aの手元に手紙があり、Bの手元にはない。

事後：Aの手元に手紙はなく、Bの手元に手紙がある。

②手紙をコピーし、そのコピーを送る

事前：Aの手元に手紙があり、Bの手元にはない。

事後：Aの手元に手紙があり、Bの手元にそのコピーがある。

③電子メールで送る

事前：Aの手元に電子メールが作成されており、Bの手元にはない。

事後：Aの手元にはコピーが残る場合と残らない場合があるが、Bの手元には届いた。

①、②から、情報の伝達は「**受け取り側の状態の変化**」に関係していることがわかる

③を見ても「**送信側の状態の変化**」は本質でないことが確かめられる

また、①②と、③を比べると、伝達の手段（メディア）とも独立であることがわかる

【情報を受け取った効果】

◆ 「情報を受け取った」とは

- ・自分に影響のある、これまで知らなかった事実を知った
- ・何らかの判断の材料にできる事実を知った

※全く関心のない手紙をもらっても、情報を受け取ったとは言えない（迷惑メール等）

◆ 歴史の試験範囲の例

事前に伝えられた出題範囲は日本史、東洋史、西洋史、アメリカ史、のうちから1つ

◇ 「今回は日本史から出題する」というメッセージが与えられた場合

事前：日本史からアメリカ史の4種類すべての試験勉強が必要

事後：日本史の勉強だけでよい

出題に関するあいまいさが4から1に減った → かなり有用な情報

◇ 「今回は世界史（日本史以外）から出題する」というメッセージが与えられた場合

事前：日本史からアメリカ史の4種類すべての試験勉強が必要

事後：東洋史、西洋史、アメリカ史の3種類の勉強だけでよい

出題に関するあいまいさが4から3に減った → それほど有用でない

【情報量】

◆情報量の表し方 I (場合の数に基づいた場合)

◇商の対数(底は2)で表す情報量

定義：情報量 = $\log_2(\text{事前の場合の数} \div \text{事後の場合の数})$

単位：ビット

性質：場合の数が大きく減る程、情報量も大きくなる

二者択一(場合の数 $2 \rightarrow 1$)の時に情報量が 1.000 になる

前頁のメッセージの情報量：日本史 $\rightarrow \log_2(4/1) = 2.000$ / 世界史 $\rightarrow \log_2(4/3) = 0.415$

※情報量を差(事前の場合の数 - 事後の場合の数)で表そうとすると

場合の数が $100 \rightarrow 97$ の場合と $4 \rightarrow 1$ の場合で等価になってしまう

明らかに後者の方が有用で、情報量は大きくあるべき

※情報量を商(事前の場合の数 \div 事後の場合の数)で表そうとすると

情報の加法性を満たさない

◇情報の加法性

ある情報 X を一度に受け取った場合の情報量と、

ある情報 X を何段階かに分けて受け取った場合の情報量は、等しい

Ex. 歴史の試験範囲の例 (図 3-1)

「今回はアメリカ史から出題」というメッセージの情報量

||

「今回は世界史から出題」というメッセージが与えられた上で

「東洋史、西洋史は出題しません」というメッセージが与えられた時の情報量

対数で表された情報量はこの条件(情報の加法性)を満たす

① 「今回は世界史から出題」の情報量： $\log_2(4/3) = 0.415$

② 「(①の後で) 東洋史、西洋史以外から出題」の情報量： $\log_2(3/1) = 1.585$

③ 「今回はアメリカ史から出題」の情報量： $\log_2(4/1) = 2.000$

① $0.415 +$ ② $1.585 =$ ③ 2.000

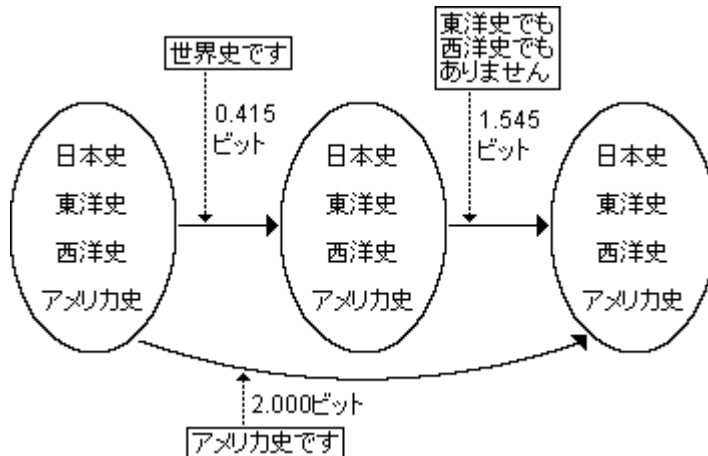


図 3-1

◆情報量の表し方Ⅱ（確率に基づいた場合）

◇確率の逆数の対数（底は2）で表す情報量

定義：情報量 = $-\log_2(\text{確率})$

単位：ビット

性質：低確率の（＝珍しい）ことを伝えるメッセージの方が情報量は大きい

→「犬が人間を噛んだ」に比べて「人間が犬を噛んだ」の情報量の方が大きい

→ニュースになる

すべてが等確率で起こる場合は、場合の数で表しても同じ結果になる

▽白い小球と赤い小球

Aの袋には5つ、Bの袋には2つ、Cの袋には3つ、小球が入っている

10個の小球は9つが白く、1つだけが赤い

中を見ないでいずれかの袋からひとつだけ小球を取り出す

取り出した小球が赤い小球であれば大当たり、ハワイ旅行1年分がもらえる

事前の場合の数は「赤い小球がA [B, C]に入っている」の3通り

では、仮に下の①②③いずれかの情報が入ったとき、どれが最も有用といえるか

①「赤い小球はAの袋に入っている」 → 確率は50%

②「赤い小球はBの袋に入っている」 → 確率は20%

③「赤い小球はCの袋には入っていない」 → 確率は70%

場合の数に基づけば③は3→2で、情報量は0.585ビット

①②は3→1で、情報量は1.585ビットで等しくなる

これでは情報量を正しく表しているとはいえない

そこで、確率を考慮に入れる

①の情報量： $-\log_2(50\%) = -\log_2(1/2) = \log_2(2/1) = 1.000$

②の情報量： $-\log_2(20\%) = -\log_2(1/5) = \log_2(5/1) = 2.322$

③の情報量： $-\log_2(70\%) = -\log_2(7/10) = \log_2(10/7) = 0.515$

情報量の最も大きい②が、最も有用といえる

【情報通信】

◆プロトコル

プロトコルとは、お互いの意図を理解するために定める約束事

「トランシーバで話す時は最後に『どうぞ』という」

「クーショとの会話が仏語でも日本語でも上手くいかなかったら英語で話す」

なども一種のプロトコル

人間なら少々約束違反があっても理解できることがあるが、コンピュータにはできない

→コンピュータ同士の通信ではより厳密にプロトコルが規定

◆暗号化

メッセージを送信するとき、平文のまま送信すると第三者に不法に読み取られる危険

→「鍵」に基づいて意味を持たない文字列に変換する … 暗号化

復号化 … 暗号化された文字列を、「鍵」に基づいて平文に変換し直すこと

◇共通鍵暗号

暗号化と復号化に同じ鍵を用いる

鍵の受け渡しの時点で他者に鍵を知られないように注意

▽植田の陰謀 with 加藤さん (図 3-2)

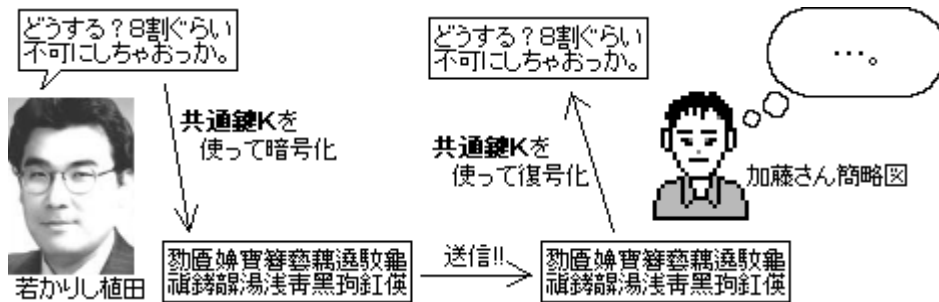


図 3-2

◇公開鍵暗号

暗号化の鍵は誰でも入手可能にしておく … 公開鍵

復号化の鍵は自分だけが持つ … 秘密鍵

▽「兄、妹が心配」の巻 (図 3-3)

公開鍵Kはケンジの持つ秘密鍵K'と対をなす

同様に公開鍵M, P, Aは、それぞれ、ミチコ、ピエール、アニーが持つ秘密鍵M', P', A'と対をなす

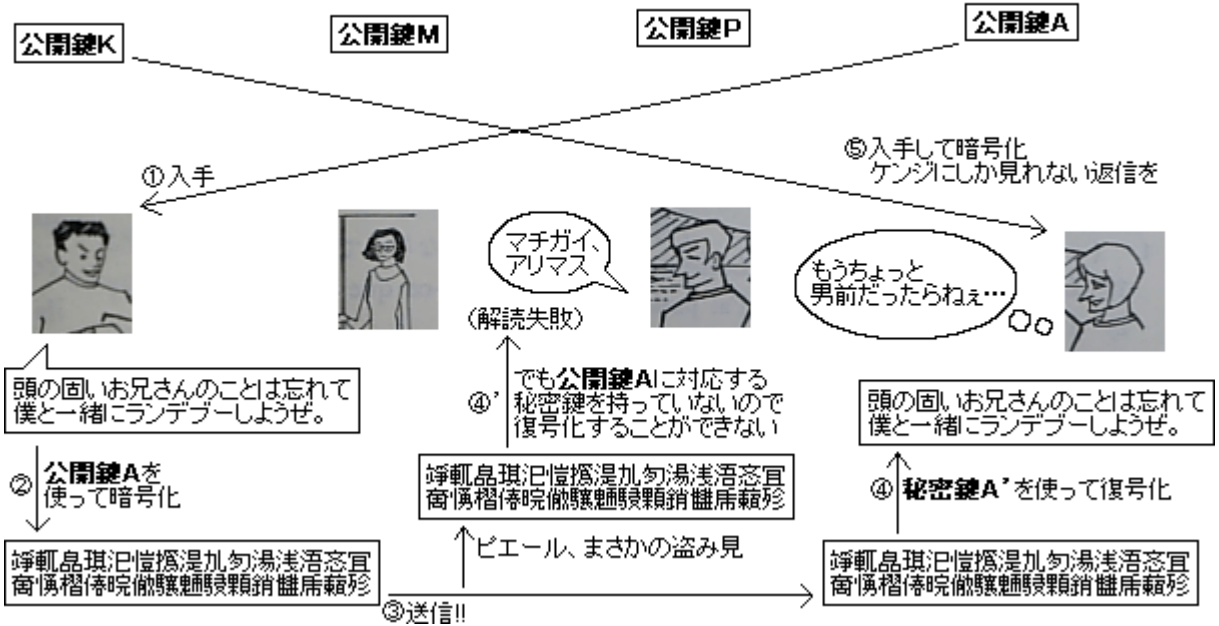


図 3-3

【情報ネットワークの枠組】

◆交換の方式

交換機 … 通信される情報を経路に振り分ける装置

◇回線交換

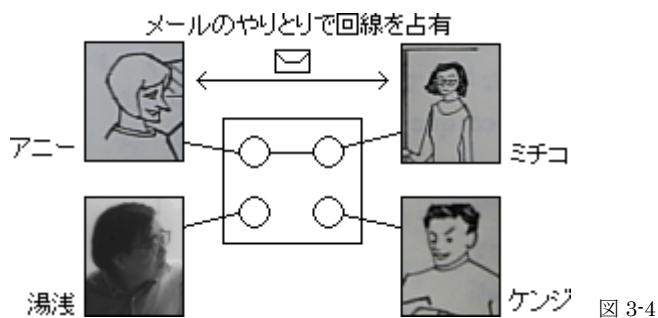
通信する両者の間に仮想的な通信路を確保

つながった両者の間では安定した通信が可能

回線を確保できなければまったく通信できない

特に通信するデータがなくても回線は確保されたまま → 回線の利用効率が低い

▽ケンジのジレンマ (図 3-4)



- ①ケンジはアニーとメールのやりとりをしたい
- ②アニーはミチコとメール中
- ③回線が使われているのでアニーとの通信はできない
- ④じゃあミチコでいいや、と思ったが、同様にミチコとも通信不可
- ⑤ケンジの通信できる相手は湯浅しかいない

◇パケット交換

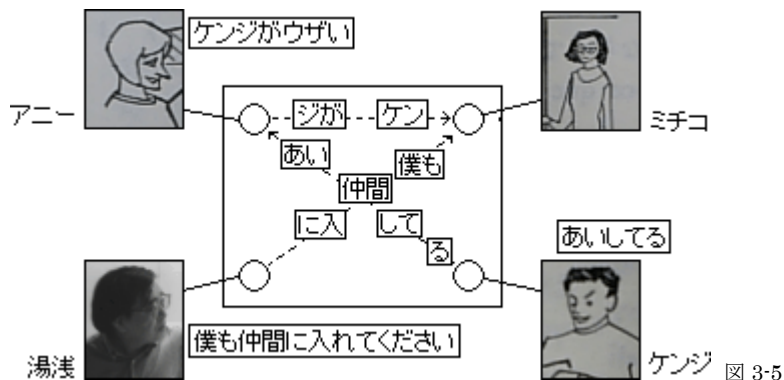
データをパケットに分割 → 次々と交換機に届け、交換機はそれを順次配送

1パケット送るたびに回線の占有は解かれるので、どの両者も通信可能

インターネットではこの方式が主流

通信速度の保証は難しい … パケット管理のための遅延、混雑時の速度低下

▽それぞれの思い (図 3-5)



同時に複数方向へのメールを送ることが出来る

【インターネット】

◆プロトコル

現在のインターネットでは **TCP/IP** と呼ばれるプロトコル群が使われる

◇TCP/IP

4階層から成る階層プロトコル

図 3-6

TCP/IPモデル	主なプロトコル	主な役割
アプリケーション層	HTTP, SMTP	アプリケーション間の通信
トランスポート層	TCP, UDP	信頼性のある1:1の通信
インターネット層	IP	ネットワーク間通信
ネットワークインタフェース層	(イーサネット)	ネットワーク内通信

TCP … データの packets 分割・合成を担当

IP … 分割された packets を宛先に届ける、また、そのために適切なルートを選ぶ

◇カプセル化

上位のプロトコル (図 3-6 において上にある方が上位) に則って作成されたデータ

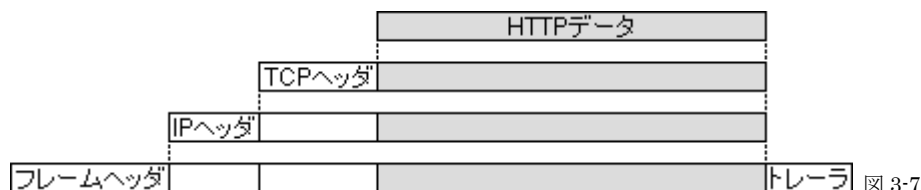
… 下位のプロトコルのための制御データが付加される

先頭に付加される物 … ヘッダ

末尾に付加される物 … トレーラ

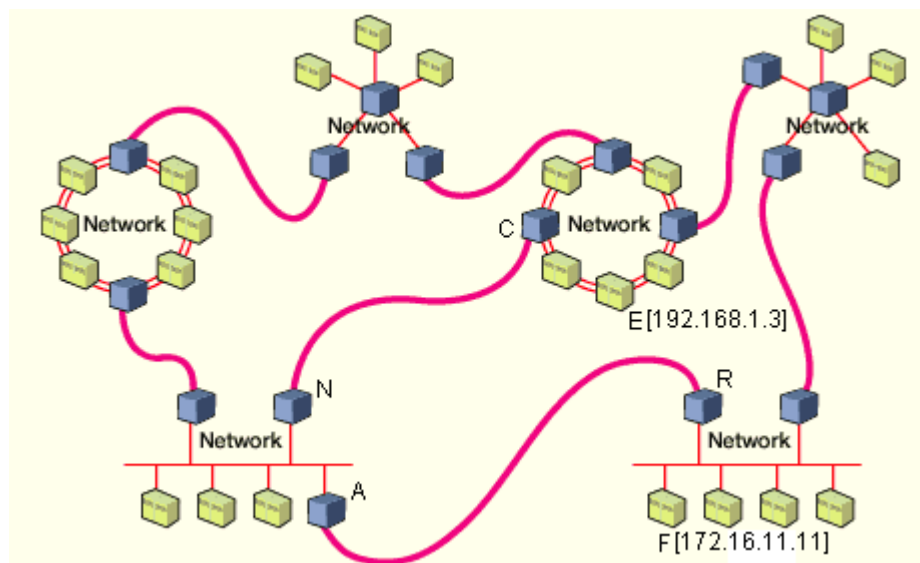
さらに下位のプロトコルによる通信を行うには、

ヘッダを含めた全データに対して再びカプセル化が行われる (図 3-7)



◆ネットワークの集合体と通信

インターネット … 小規模なネットワークが互いに接続された集合体 (図 3-8)



ルータ … ネットワーク同士を接続する中継器機 (図 3-8 でいうと紫のやつ)

◇マシン F のウェブブラウザで、マシン E のサーバにあるウェブサイトを開覧する例

- ①ブラウザがウェブサイトの URL (アドレス) から **IP アドレス** を調べる
IP アドレスはコンピュータの住所のようなもので、32 ビットの数字列で表される
それを人間にわかるように置き換えたのが **URL**
- ②ブラウザが、宛先の IP アドレスのウェブサーバに対して **HTTP** のメッセージを送る
 - (a)送信元マシン F がメッセージをパケットに分割 (TCP プロトコル)
 - (b)分割されたパケットを宛先の IP アドレスに送信する
 - (c)各パケットはまず、宛先の IP アドレス向けの適切なルータである R に届けられる
 - (d)F→R→A→N→C→E の適切な順に転送される (IP プロトコル)
 - (e)受信したマシン E はパケットを元の順番に並べ、**HTTP** メッセージを読みとる
- ③(e)のメッセージを受け取ったウェブサーバがブラウザにメッセージを返す
(②の行程を逆に辿る)

※情報の加法性についての補足

Ex.歴史の試験範囲の例 (図 3-1)

「今回はアメリカ史から出題」というメッセージの情報量

||

「今回は世界史から出題」というメッセージが与えられた上で

「東洋史、西洋史は出題しません」というメッセージが与えられた時の情報量

と、書きましたが、実際我々がこれらを等価と考えられるかというと、必ずしもそうではないことがわかれると思います。「今回は世界史から出題」というメッセージを得た後、アメリカ史以外の勉強をしてしまった人にとっては「東洋史、西洋史は出題しません」というメッセージを聞いたら損した気分になります。この損した気分は、最初から「今回はアメリカ史から出題」というメッセージを得ていたなら味合わずにすんだものですから、やはり2つの情報が等価とは言い難いです。時間の経過が進めば進むほど情報量は失われていくというのが、我々の感じるところです。しかし、ここでは、時間という概念や、勉強を進めてしまうという偶然の事象は考えずに、情報量を扱っています。教科書にも「この加法性は絶対的なものでなく、情報量を扱いやすくするための要請である。」と書かれています。まあ、細かいことは気にするな、ってことです。

第4章 データの扱い

【データモデル】

◆データとデータモデル

データ … コンピュータの処理対象となる符号化された情報

データモデル … データを体系的に扱うためのモデル

【代表的なデータモデルと演算】

◆集合モデル

グループは集合で表すことが多い … 集合Aと集合Bを定義

共通部分(積集合) $[=A \cap B]$ 、和集合 $[=A \cup B]$ 、差集合 $[=A - B]$ の演算をもつ

対象「AかBのメンバーでCのメンバーでない人たち」 → モデル「 $(A \cup B) - C$ 」

Ex. 検索エンジンの AND 検索、OR 検索、NOT 検索

◇ベン図とオイラー図

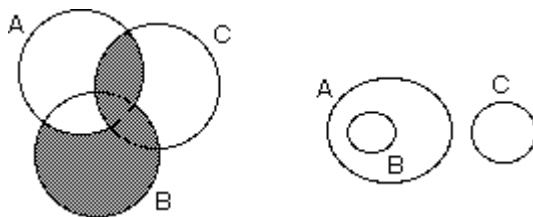


図 4-1

ベン図 … 図 4-1 左図、円の重なった部分で共通部分を表す、網掛け部には要素がない

オイラー図 … 図 4-1 右図、共通部分がなければ別々の円で表す

完全な包含関係の場合は円の中に円を入れる

◆ネットワークモデル

◇ケーニヒスベルグの橋 (図 4-2、図 4-3)

図 4-2

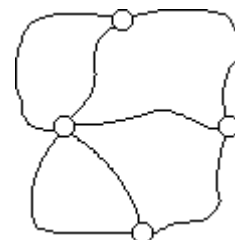
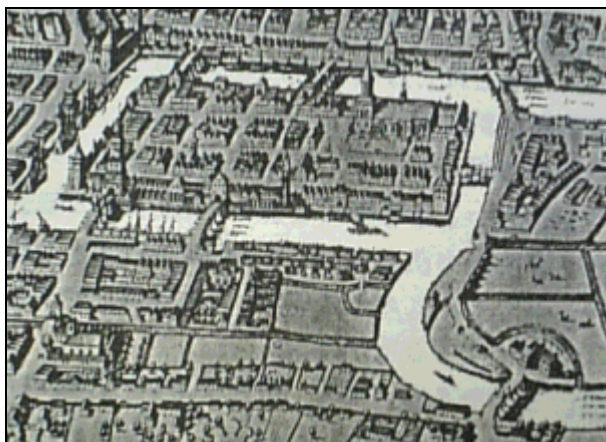


図 4-3

「同じ橋を二度渡らずに、すべての橋を渡ることは可能か」

→この問題を考えるために図をモデル化 (図 4-2 → 図 4-3) … グラフで表す

→「それぞれの線 (エッジ) を一度だけ辿って、すべてのエッジを辿れるか」

オイラー経路 … すべての線を重複なく辿る経路

◇ウェブ

ウェブのそれぞれのページをノード (図 4-3 でいうと○) と考える

→ それぞれのリンクはノードとノードをつなぐ矢印(エッジ)と見ることが出来る

◆階層モデル/木構造

◇階層的ファイルシステム

コンピュータのファイルシステムも木構造になっていることが多い (図 4-4)

▽ファイルシステムを木構造にすることによる利点

- ①フォルダの中にサブフォルダを作って細かい仕分けができる
- ②フォルダを分けておけば、それぞれのサブフォルダが混ざることはない
- ③フォルダをたどることで、すべてのフォルダとファイルを重複なく処理できる
- ④サブフォルダ以下の内容を、そのサブフォルダで代表させることができる

たとえば、階層構造の中でサブフォルダを移動すれば、サブフォルダ以下をまとめて移動させることが出来る

◇住所の階層性

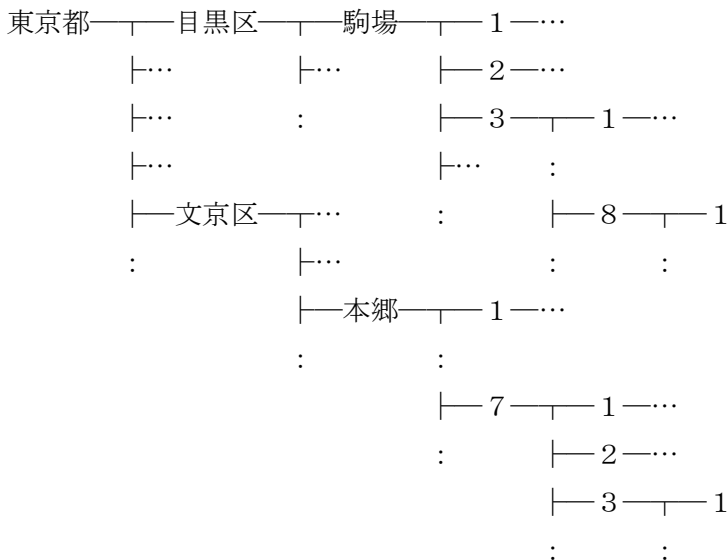
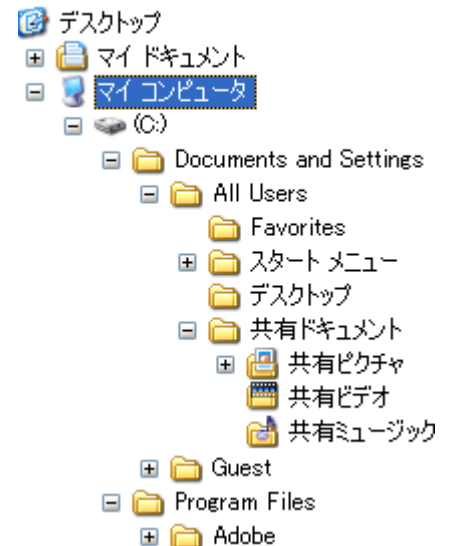


図 4-4



▽ドメイン名も階層構造になっている

u-tokyo.ac.jp … jp (日本) →ac (大学等) →u-tokyo (東京大学) →…

◆関係モデル

「関係」によるモデル化

関係 … 実際に関係のある組を集めたデータ

Ex. 「山口君は東京に住んでおり、電話番号は 03-1987-0427」という事実

→山口、東京、03-1987-0427 の関係

正規化 … 同じデータが何回も出てくるような冗長な関係を、分割して簡潔にすること

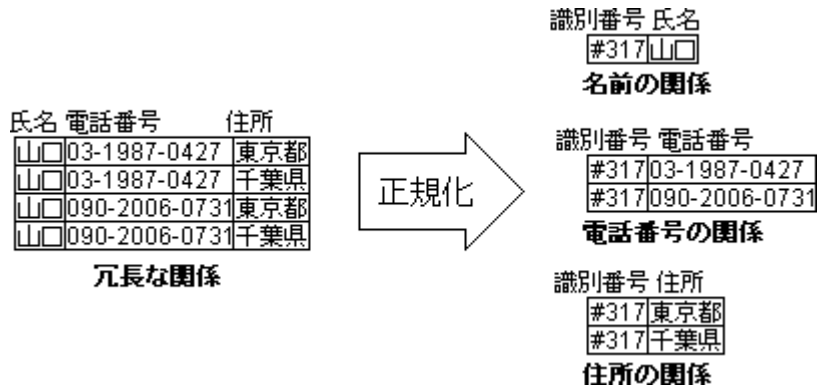


図 4-5

◇関係代数

▽5つの演算から成る

- ①和 … 2つの関係の組を集める
- ②差 … 関係の組の中で他の関係にないものを集める
- ③射影 … 関係の組から不要な項目を除いた組を集める
- ④選択 … 関係の組のなかで指定された条件を満たすものだけ集める
- ⑤直積 … 2つの関係の組を総当たりでつないだ組を集める

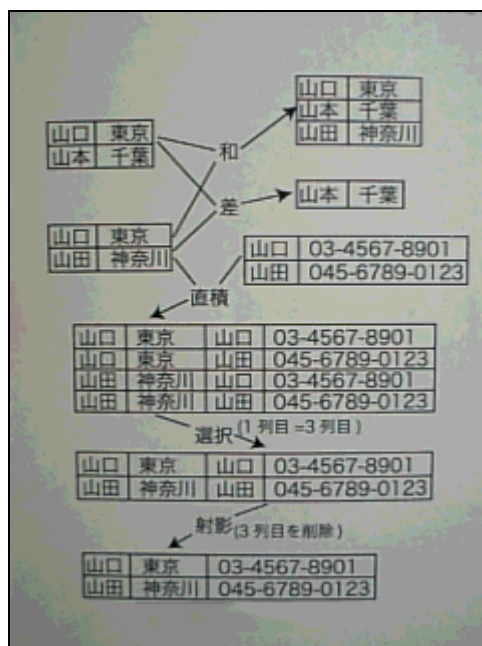


図 4-6

第5章 計算の方法

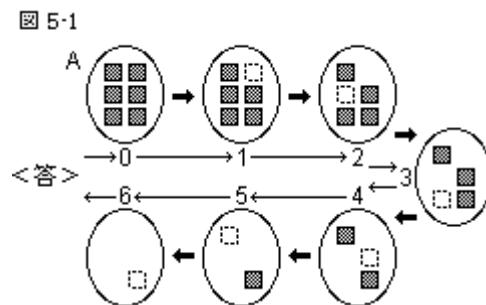
【計算の方法】

◆計数 (counting) の方法

ある集合Aに含まれる要素数を求める

◇取り出し型 (図 5-1)

- ↓<答> (変数) を 0 にする
- ↓A が空でない間、以下の処理を繰り返す
 - ↓要素を 1 つ取り出す
 - ↓<答>の値を 1 増やす
- ↓<答>の出力



◇分割型 (図 5-2)

- ↓A が空なら<答>は 0、要素数が 1 なら<答>は 1
- ↓そうでなければ
 - ↓A を A_B と A_C に分割する (A_B も A_C も空集合ではない)
 - ↓ A_B 、 A_C についても A と同じ処理を行う
 - ↓<答> = A_B の要素数 + A_C の要素数
- ↓<答>の出力

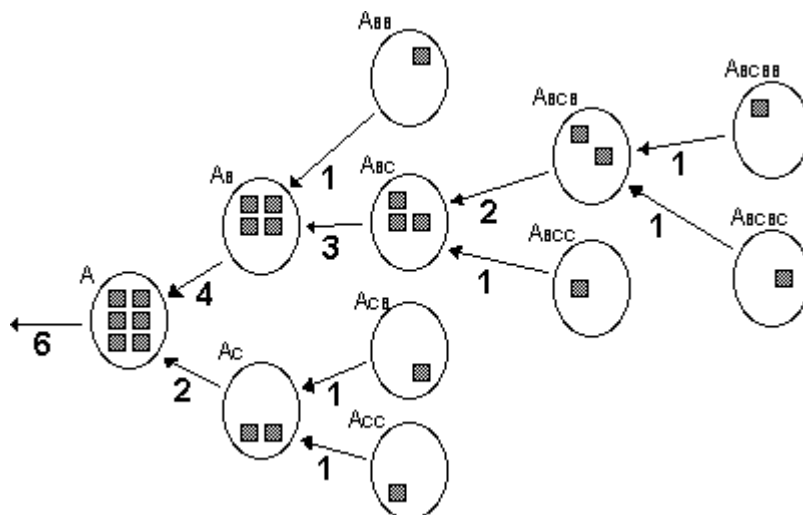


図 5-2

【計算の記述】

◆八十八夜問題

今年の八十八夜は何月何日か。ただし、今年の立春は2月4日で、今年が平年である。

八十八夜 … 立春から数えて88日目の日 (=立春の87日後)

◇解法の手順

2月4日の87日後を求めたい

→ 2月91 (=4 + 87) 日という仮想日付になる

<残り日数>を91にする

91 > 28 (2月の日数) なので

<残り日数>から2月の日数を引き、3月に進む

→ 3月63 (=91 - 28) 日という仮想日付になる

63 > 31 (3月の日数) なので

<残り日数>から3月の日数を引き、4月に進む

→ 4月32 (=63 - 31) 日という仮想日付になる

32 > 30 (4月の日数) なので

<残り日数>から4月の日数を引き、5月に進む

→ 5月2 (=32 - 30) 日という仮想日付になる

2 < 31 (5月の日数) なので、計算終了

◇よりプログラミングっぽい記述

<残り日数> … はじめは91に設定 → 次第に小さい値に変化 … 「変数」と呼ぶ

このとき、「残り日数」という文字列自体を「変数名」と呼ぶ

代入 … 変数に値を設定する操作

「変数名」 ← 「値 or 式」 という形式で表すことにする

逐次処理 … 計算処理の基本の1つ

途中を抜かしたり、複数の処理を同時にやったりせず、

書かれている順番に順序よく処理

条件付き処理 … ～ならば…という処理を行う、という操作

if 条件

then 条件が成立した場合に行う処理

else そうでない場合に行う処理

endif

という形式で表すことにする (else は省略が可能)

計算の記述で使われる「変数」 → 値を書き換えられる

数学で使われる「変数」 → 値と結合する … 「パラメタ」と呼ばれる

◇よりプログラミングっぽい記述を用いて書き直し

```
<残り日数> ← 4+87
if <残り日数> > 28 (2月の日数)
  then <残り日数> ← <残り日数>-2月の日数
    if <残り日数> > 31 (3月の日数)
      then <残り日数> ← <残り日数>-3月の日数
        if <残り日数> > 30 (4月の日数)
          then <残り日数> ← <残り日数>-4月の日数
            if <残り日数> > 31 (5月の日数)
              then <残り日数> ← <残り日数>-5月の日数
                :
                (6月以降の処理)
                :
            else “5月”<残り日数>”日” と表示
          endif
        else “4月”<残り日数>”日” と表示
      endif
    else “3月”<残り日数>”日” と表示
  endif
else “2月”<残り日数>”日” と表示
endif
```

▽字下げ (インデント)

この例では **then** と **else** が、対応する **if** と **endif** よりも右にずらされている

→ まとまりの構造をわかりやすくするため

たとえば、緑の斜体文字で表された部分は、3月の処理を表すひとまとまり

◇反復処理と配列

反復処理 … ある条件が成立している間、指定された処理を繰り返す操作

```
while 条件 do
```

```
  繰り返し実行する処理
```

```
done
```

という形式で表すことにする

配列 … 要素の集合から添字によって値を取り出し、変数として扱うことができる

この要素の集合が、配列

たとえば、**daymonth** という配列には1月~12月の日数が要素として格納

daymonth₁ = 31、**daymonth**₂ = 28 … 1、2が添字、31、28が要素

配列全体をまとめて扱うこともできる

◇反復処理と配列を用いて書き直し (daymonth には既に要素が格納済み)

```
<残り日数> ← 4+87
m ← 2
while <残り日数> > daymonthm do
  <残り日数> ← <残り日数>-daymonthm
  m ← m+1
done
m“月”<残り日数>”日”と表示
```

これがわからない人は、今からコンピュータになったつもりで順に記述を辿ってみましょう。<残り日数>って書くと文字数食うので代わりに「凹」って書きますね。なんか箱っぽいから (変数は値を格納する箱と喩えられます)。それではスタート。

よし、まず1行目。凹に91を入れて、と、2行目、mに2を入れる、と。3行目から6行目までは反復処理だな。じゃあ while の条件を見てみよう。「凹 > daymonth_m」か。これが真のとき、do と done に挟まれてる処理を行えばいいんだな。今凹は91、mは2で、daymonth₂ っていうたら28だから、「91 > 28」だ。これは真だから、中の処理をやっちゃおう。凹に凹-daymonth_mを代入。と、いうことは、凹は91 (凹) -28 (daymonth₂) に書き換わるわけだから、新しい凹の値は63だ。で、mの新しい値は2+1で3だな。よし、中の処理は一通り終わったな。反復処理は条件が真である限りまた中の処理を行うんだから、ここでもう一度条件の確認だ。今凹は63、mは3で、daymonth₃ っていうたら31だから、「凹 > daymonth_m」は「63 > 31」、真だな。じゃあもう一回中の処理を行おう。凹-daymonth_mは、63-31で32、つまり新しい凹の値は32。mは4。done 以上の処理が終わったからもう一度条件の確認。「凹 > daymonth_m」、「32 > 30」、まだ真だ。凹-daymonth_mは、32-30だから凹は2、mは4の次だから5だ。そろそろどうだろう、条件をもう一度確認してみよう。「凹 > daymonth_m」に、凹=2、m=5を当てはめてみると、「2 > 31」、お、これは偽だ、条件が成立していない。じゃあ、これで while~done の処理はおしまいだ。次へ行こう。「m“月”凹”日”と表示」すればいいんだな。今mは5、凹は2だから、表示する文字列は、「5月2日」。これでオッケー。

はい、こんな感じです。わかんねえよ、ばか。って怒っちゃった人はビシバシメールをください。

【プログラム言語処理系】

◆機械語とソフトウェア（用語集的な感じで。これで十分だと思いますが、教科書は p115)

機械語 … コンピュータ（ハードウェア）が理解できる言語（すべて2進数）

バイナリプログラム … 機械語で書かれたプログラムの別名

ハードウェア … コンピュータの、バイナリプログラムを解釈し実行するための部分

ソフトウェア … バイナリプログラムを含む各種のプログラム

アセンブリ言語 … 機械語の機能部分やデータの場所に、人が読める名前をつける言語

アセンブラ … アセンブリ言語でつけた名前を、
数値や”0と1の集まり”（機械語）に変換するもの（図5-3）

言語処理系 … “プログラム言語で書かれたプログラム”を、
“機械語で書かれたプログラム”に変換（翻訳）するシステム（図5-4）

高水準言語 … 人間が理解しやすい形でプログラムを読み書きできる言語
そのままではコンピュータは実行できない

C言語、C++、Java など

コンパイラ … プログラム言語で書かれたプログラムを機械語に変換するソフトウェア
言語処理系の1つ

ソースプログラムに誤りがあれば指摘

→プログラムが誤りを含んだまま実行されてしまう危険が減る

ソースプログラム … コンパイラで変換される前のプログラム

オブジェクトプログラム … コンパイラで変換されたプログラム

ライブラリと結合して最終的なバイナリプログラムとなる

ライブラリ … システムに予め用意されている標準機能を提供するプログラム群
たとえば、データの入出力、コンピュータ内でのデータ管理など

インタプリタ … コンパイラを通さずに実行可能な言語処理系

いちいちコンパイル（機械語に変換）しなくていいので、

プログラムの変更は容易、ただし解釈に時間がかかり実行は遅い

中間言語方式 … ソースプログラムを”ある程度機械語に近い”段階にまで変換

実際の実行は中間言語で記述されたプログラムを、

また別のプログラムで解釈実行することで実現

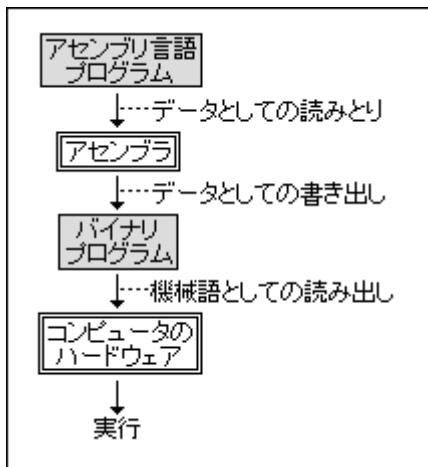


図 5-3

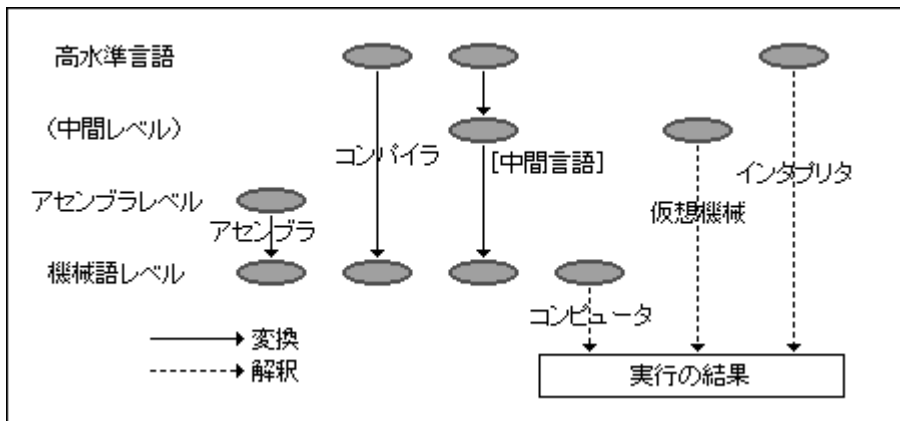


図 5-4

第6章 問題の解き方

レポートをコピーで済ませた皆さんも頑張って理解してください

【アルゴリズム】

◆アルゴリズムの役割

コンピュータによって問題を解く

… 人間が書いたプログラムをコンピュータが実行するということ

アルゴリズム … プログラムになる前の段階の計算手順

◇現実問題からプログラムを作る過程 (図 6-1)

- ①問題をモデル化する
- ②モデル化された問題に対して、それを解く計算手順 (= アルゴリズム) を考える
- ③手順通りに計算するプログラムを作る



図 6-1

◇アルゴリズム ≠ プログラム

前章で登場した八十八夜問題の解法はまだプログラムではなく、アルゴリズム段階

```
<残り日数> ← 4 + 87
m ← 2
while <残り日数> > daymonthm do
  <残り日数> ← <残り日数> - daymonthm
  m ← m + 1
done
m" 月"<残り日数>" 日"と表示
```

図 6-2

これを java などのプログラミング言語で書き直すと、

あとは計算を実行するだけでよいプログラムになる

アルゴリズムの選択がプログラムの実行時間を大きく左右する

アルゴリズムとプログラムを

ひっくるめて考える (頭の中で解法を考えながらプログラムを作る) と…

- プログラムを作ってから新しい解法を思いついたら二度手間
- いちいちまたプログラムを打ち直し
- アルゴリズムの時点で比較、選択した方が効率がよい

◆アルゴリズムの実例

◇コンピュータより古いアルゴリズム

ユークリッドの互除法、ドント方式など

◇x の平方根を求める問題

ある正の実数 x が与えられたときに、2乗すると x に近くなる正の実数 y を精度 δ で求める。つまり、 $|\sqrt{x} - y| < \delta$ となるような y を1つ求める。

▽反復法を用いたアルゴリズム

```

y ← 0
while (y + δ)2 < x do
  y ← y + δ
done
return y

```

※ 「return y」は「y を解として計算を終了」の意

▽反復法を用いたアルゴリズムの考え方

- 解の候補を順に検討 → 候補を2乗し、それが x を超えたら1つ前の候補が解
- $\delta = 0.5$ で $x = 8$ の場合
- $y = 0$ としよう
 - … 0 の2乗は0で、 $0 < 8$ → y に0を代入しといて次へ
- $y = 0.5$ としよう
 - … 0.5 の2乗は0.25で、 $0.25 < 8$ → y に0.5を代入しといて次へ
- … → $y = 2.5$ としよう
 - … 2.5 の2乗は6.25で、 $6.25 < 8$ → y に2.5を代入しといて次へ
- $y = 3$ としよう
 - … 3 の2乗は9で、 $9 > 8$ → y に3は代入せず、反復処理を終える
- return y すなわち、2.5 を解として計算を終了

▽反復法を用いたアルゴリズムの性質

2 の平方根を精度 $\delta = 0.0001$ で求めると、14142 回の計算が必要 (図 6-3)

回数	0	1	2	…	14140	14141	14142
候補(y)	0.0000	0.0001	0.0002	…	1.4140	1.4141	1.4142
$(y + \delta)^2$	0.00000	0.00000	0.00000	…	1.99968	1.99996	2.00024

図 6-3

精度に逆比例した必要計算回数

- 1桁よい精度の解を求めるには10倍の計算回数がかかる

▽二分法を用いたアルゴリズム ($x > 1$ を条件とする)

```

a ← 0
b ← x
while b - a > δ do
  c ←  $\frac{a+b}{2}$ 
  if  $c^2 > x$  then b ← c else a ← c endif
done
return a

```

▽二分法を用いたアルゴリズムの考え方

数直線上で、解の含まれる区間をどんどん絞っていく

区間を半分に分け、左と右、解の含まれる方を次の区間にする

上のアルゴリズムではこの区間の左端 (=最小値) が a、

右端 (=最大値) が b という変数で表されている

つまり、a が大きくなり、b が小さくなる程区間は絞られていく

この区間の大きさが精度 δ よりも小さくなったところで計算を終え、

a (=区間の最小値) を解として終了

図 6-4 は $x=2$ のときの例

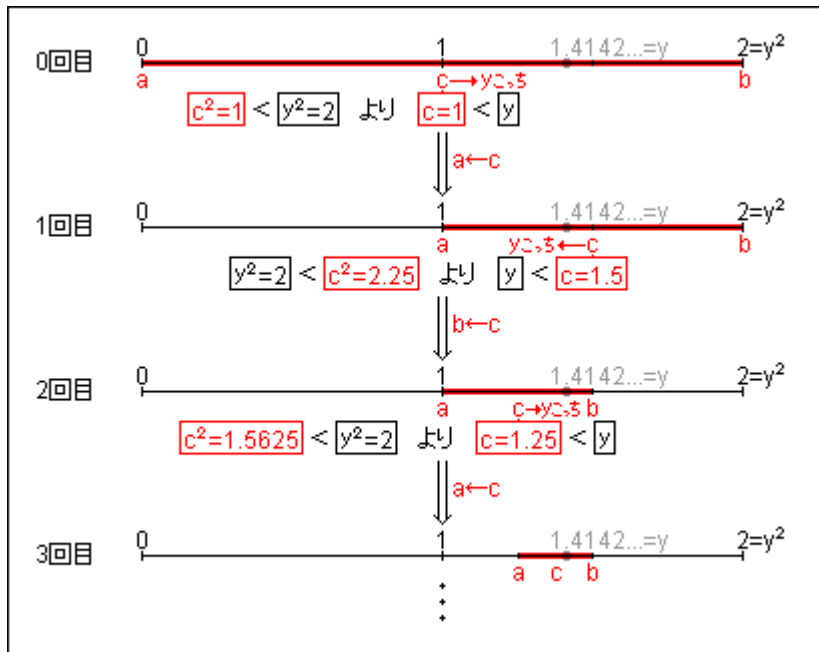


図 6-4

▽二分法を用いたアルゴリズムの性質

2 の平方根を精度 $\delta = 0.0001$ で求めると、15 回の計算が必要

→ 反復法よりも効率的

【計算量】

◆計算量の考え方

計算量 … アルゴリズムをもとにしたプログラムの実行時間を見積もるための指標
明日の天気を知るプログラムがあっても、実行するのに1週間かかっては意味がない
→ 事前に計算量を見積もる
→ 無駄なプログラムを作らなくて済む
より効率的なプログラムが選択できる

◇**計算量のオーダー** … 計算量の見積もりにおいて基準とされる、非常に大まかな尺度
どんな変数に比例するかのみが重要で、**定数は無視**される

▽N個のデータを処理するプログラムAとB

A … 2秒の処理をN回繰り返し、最後に3秒の処理を行う

- プログラムの実行にかかる時間は $2N+3$ 秒
- 計算量のオーダーは N

B … 10秒の処理を $\log_2 N$ 回繰り返す

- プログラムの実行にかかる時間は $10\log_2 N$ 秒
- 計算量のオーダーは $\log_2 N$

▽定数は無視する理由

処理1回当たりの計算時間の違いよりも、
計算回数が何に比例するかの違いの方が、全体の実行時間に決定的な違いを及ぼす
→ $N=10$ のとき A → 23秒 B → 約33秒
→ $N=1000$ のとき A → 2003秒 B → 約100秒
→ $N=10$ 万のとき A → 20万3秒 (約56時間) B → 約166秒 (約3分)
… この大きな差は2と10の定数から生じたものではなく、
Nと $\log_2 N$ という違いから生じたもの
→ Nが大きくなればなるほど違いが大きくなる
また、定数倍はコンピュータの性能などによっても左右されてしまう
→ 10秒かかるはずが9.7秒しかかかってなかったり

◆計算量の例

◇xの平方根を求めるアルゴリズムにおける計算量 (のオーダー)

反復法 → \sqrt{x} / δ

二分法 → $\log_2(x/\delta)$ (= $x/2^n < \delta$ となるような最小のn)

◇フィボナッチ数を求めるアルゴリズムにおける計算量 (のオーダー)

アルゴリズム自体はテスト範囲じゃないので、「こんなのもあるんだ」程度に

再帰アルゴリズム → 2^n

メモ化アルゴリズム → n

◇データの個数 n に対する計算時間の、アルゴリズムによる違い (図 6-5)

1回の処理に 1 ns (ナノ秒) かかる場合

n	1	10	100	1000	1万	10万	100万	1000万
$\log n$	-	1ns	2ns	3ns	4ns	5ns	6ns	7ns
n	1ns	10ns	100ns	1 μ s	10 μ s	100 μ s	1ms	10ms
n^2	1ns	100ns	10 μ s	1ms	0.1s	10s	16分	27時間
n^3	1ns	1 μ s	1ms	1s	16分	11日	31年	3000年
2^n	2ns	1 μ s	40兆年					

図 6-5

第7章 コンピュータの仕組み

前半は訳のわからん用語の暗記物、後半はややこしい演算、頑張りましょう

【計算の実現機構】

◆プログラム内蔵方式

プログラム … コンピュータの実行する計算処理手順に関わる情報

データ … コンピュータが処理する対象の情報

万能チューリング機械 … プログラムとデータをテープに記録する仮想的な計算機械

プログラム内蔵方式 … テープに代わってメモリ上にプログラムとデータを保持し、
プログラムに従って計算を進めるような機構

現実のコンピュータはこの方式をとる

フォンノイマン型コンピュータ … プログラム内蔵方式のコンピュータの別名

◆コンピュータの基本構成 (図 7-1)

演算装置 … データに対して計算処理を施す演算装置

主記憶装置 (メインメモリ) … 複数のデータやプログラムを記憶する装置

制御装置 … 主記憶装置上のプログラムに従って演算装置を駆動する装置
主記憶装置へのデータの読み書きも担当

中央処理装置 (CPU) … 制御装置 + 演算装置 Central Processing Unit

マイクロプロセッサ (MPU) … CPU を1つの半導体集積回路として実現したもの
Micro Processing Unit

アドレス … 主記憶装置上の情報の位置を表す数値

演算レジスタ … 中央処理装置による計算に使用するデータを一時的に保持する装置
初期のコンピュータではこれが1つだけしかなく、
アキュムレータ (AC) と呼ばれていた

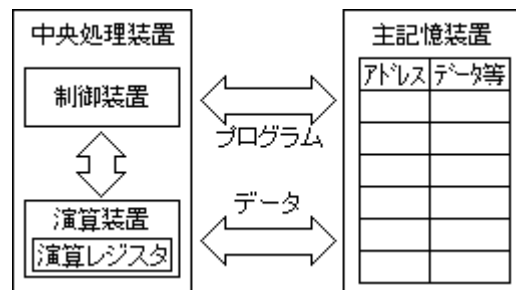


図 7-1

※演算レジスタが十分多く存在すると、計算の際に主記憶装置にほとんどアクセスせずにする。それでは勉強にならないので、この授業では演算レジスタは1つという設定にする。

◆機械語レベルのプログラム例

命令集合 … 個々の CPU で利用できる命令の集まり (図 7-2)

命令コード … 命令の種類を表す符号

オペランド … 命令コードの後につけられる付加情報

種類	内容	意味
データ転送命令	load A	アドレスAのデータを演算レジスタに読み込む
	store A	演算レジスタのデータをアドレスAに書き込む
演算命令	add A	アドレスAのデータを演算レジスタの値に加える
	subtract A	アドレスAのデータを演算レジスタの値から引く
分岐命令	jump A	アドレスAにプログラムの実行を移す
	jumpzero A	演算レジスタの値が0のときのみjump Aと同じ機能
その他	write	演算レジスタのデータを出力する
	halt	プログラムの実行を停止する

図 7-2

データ転送命令 … 主記憶装置上のデータを演算レジスタに読み込み

演算レジスタのデータを主記憶装置に書き込み

演算命令 … 算術演算 (四則演算) や論理演算 (AND や OR など) を実行

結果は演算レジスタに書き込まれる

分岐命令 … 指定したアドレスにプログラムの実行を移す

条件分岐 (ジャンプ) と無条件分岐 (ジャンプ) がある

条件ジャンプ … 演算レジスタの値に応じて選択的に実行を移す (jumpzero)

無条件ジャンプ … 演算レジスタの値にかかわらず実行を移す (jump)

上の命令集合では、すべての命令はオペランドとしてたかだか1つの値をとる

… add A B などの形式はとれない

◇load と store (図 7-3)

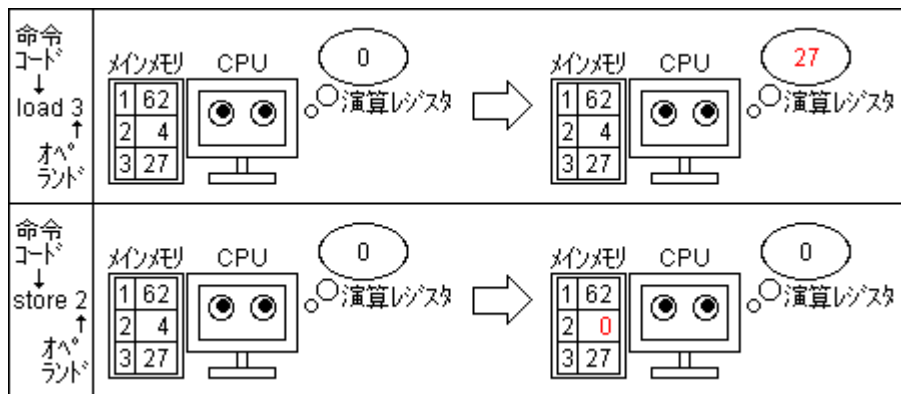


図 7-3

◇add、jumpzero、write (図 7-4)

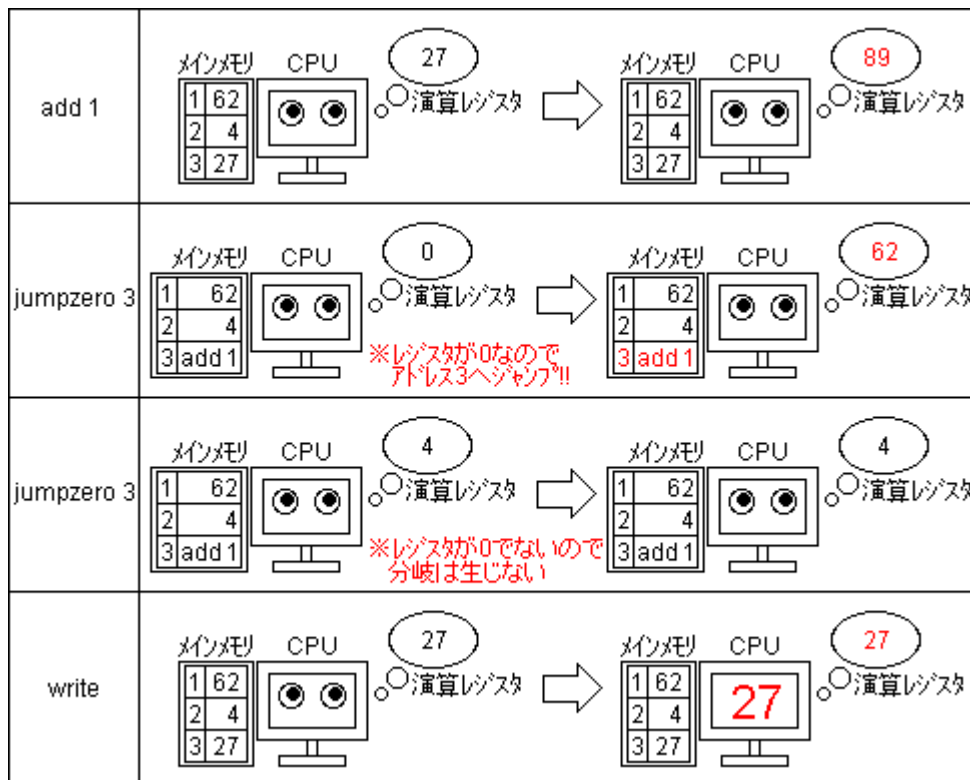


図 7-4

◇1 から 10 までの和を求めるプログラム

アドレス	命令	アドレス	命令
1001	load 2001	1008	jump 1001
1002	add 2002	1009	load 2001
1003	store 2001	1010	write
1004	load 2002	1011	halt
1005	subtract 2003	2001	0
1006	store 2002	2002	10
1007	jumpzero 1009	2003	1

▽各アドレスの意味

1001~1011 には命令 (プログラム) が記憶されている

2001~2001 にはデータが記憶されている

→ プログラムとデータがぐちゃぐちゃになってややこしくならないように
アドレス 2001 … 和を記憶する場所

0→10→19→27→34→…→55 と増えていく

アドレス 2002 … ここの値が 0 になるまで処理が続く

10→9→8→…→0 と減っていく

アドレス 2003 … アドレス 2002 の値を 1 ずつ減らすための「1」を記憶
ここを 2 にすると 2002 の値は 10→8→6→… と減っていく

▽処理の流れ（さっきのプログラムと見比べながら進むと理解しやすいかも）

アドレスの小さい場所の命令から大きい場所の命令へ、順々に処理

以下実際の処理の流れ（[] 内は順に、演算レジスタ, アドレス 2001, 2002 の値）

- 1001 アドレス 2001 の値 0 を演算レジスタに読み込む [0,0,10]
- 1002 アドレス 2002 の値 10 を演算レジスタの値 0 に加算 [10,0,10]
- 1003 演算レジスタの値 10 をアドレス 2001 の値 0 に上書き [10,10,10]
- 1004 アドレス 2002 の値 10 を演算レジスタの値 10 に上書き [10,10,10]
- 1005 演算レジスタの値 10 からアドレス 2003 の値 1 を減算 [9,10,10]
- 1006 演算レジスタの値 9 をアドレス 2002 の値 10 に上書き [9,10,9]
- 1007 演算レジスタの値 (9) が 0 ではないので無視 [9,10,9]
- 1008 アドレス 1001 へ戻る [9,10,9]
- 1001 アドレス 2001 の値 10 を演算レジスタの値 9 に上書き [10,10,9]
- 1002 アドレス 2002 の値 9 を演算レジスタの値 10 に加算 [19,10,9]
- 1003 演算レジスタの値 19 をアドレス 2001 の値 10 に上書き [19,19,9]
- 1004 アドレス 2002 の値 9 を演算レジスタの値 19 に上書き [9,19,9]
- 1005 演算レジスタの値 9 からアドレス 2003 の値 1 を減算 [8,19,9]
- 1006 演算レジスタの値 8 をアドレス 2002 の値 9 に上書き [8,19,8]
- 1007 演算レジスタの値 (8) が 0 ではないので無視 [8,19,8]
- 1008 アドレス 1001 へ戻る [8,19,8]
- :
- 1008 アドレス 1001 へ戻る [1,54,1]
- 1001 アドレス 2001 の値 54 を演算レジスタの値 1 に上書き [54,54,1]
- 1002 アドレス 2002 の値 1 を演算レジスタの値 54 に加算 [55,54,1]
- 1003 演算レジスタの値 55 をアドレス 2001 の値 54 に上書き [55,55,1]
- 1004 アドレス 2002 の値 1 を演算レジスタの値 55 に上書き [1,55,1]
- 1005 演算レジスタの値 1 からアドレス 2003 の値 1 を減算 [0,55,1]
- 1006 演算レジスタの値 0 をアドレス 2002 の値 1 に上書き [0,55,0]
- 1007 演算レジスタの値 (0) が 0 なので、アドレス 1009 へ進む [0,55,0]
- 1009 アドレス 2001 の値 55 を演算レジスタの値 0 に上書き [55,55,0]
- 1010 演算レジスタの値 55 を出力、画面に表示
- 1011 プログラム停止

▽高水準言語で記述すると

```
sum ← 0
i ← 10
while i ≠ 0 do
  sum ← sum+i
  i ← i-1
done
write(sum)
```

※こっちの方が理解しやすい（はず）

◆実際の機械語プログラム

ビットパターン（0と1の列）による記述（…バイナリプログラム）

→ コンピュータ内の物理的表現、電圧の高低、電流の on/off に対応

さっきやった和のプログラムは load など英文字が含まれているので、
機械語のプログラムというよりは

アセンブリ言語（シケプリ前編 p.24 参照）によるプログラムに相当

バイナリプログラムでは図 7-2 の 8 つの命令コードを区別するのに

3 (= $\log_2 8$) ビットのビットパターンが用いられる (000,001,010, …, 111)

アドレス、演算レジスタ上のデータ、主記憶装置上のデータ

→ すべて 2 進数による表現

【論理演算と組み合わせ回路】

ここが一番ややこしいんじゃないでしょうか。もう湯浅とか出てこさせてふざけてる場合じゃないです。ファイト！！

◆組み合わせ回路とは何ぞや？

2 進符号表現の演算を実現するための記憶をもたない回路

n ビットの入力があると、即座かつ一意に m ビットの出力を返す

要は、何か入力したらすぐ（ある法則に従って）何か返してくれるってこと！

この程度の理解でいいと思います。

◆n 入力 1 出力の論理関数

論理関数 … 2 進法に基づく演算

n ビットの入力パターンは 2^n 通り

→ この入力パターンについて、それぞれ 0 か 1 の出力が可能

→ 0 か 1 の 2 通りが 2^n パターン

→ 論理関数の種類は $2 \cdot 2^n$ 種類

完備な組み合わせ … $2 \cdot 2^n$ 通りのすべての論理関数を実現できる演算の組み合わせ

→ {AND, OR, NOT} の組み合わせが有名

→ 証明は教科書 p. 159 にあるけど、わけわからんし覚えなくていいと思います

※論理演算では 0 のことを偽(FALSE)、1 のことを真(TRUE)と呼んだりします

AND(x,y) … 2つの入力 x,y がともに真(1)のときに真(1)と出力

→ $\text{AND}(0,0) \Rightarrow 0$ $\text{AND}(0,1) \Rightarrow 0$ $\text{AND}(1,0) \Rightarrow 0$ $\text{AND}(1,1) \Rightarrow 1$

OR(x,y) … 2つの入力 x,y の少なくとも一方が真(1)のときに真(1)と出力

→ $\text{OR}(0,0) \Rightarrow 0$ $\text{OR}(0,1) \Rightarrow 1$ $\text{OR}(1,0) \Rightarrow 1$ $\text{OR}(1,1) \Rightarrow 1$

NOT(x) … 入力 x が真(1)のときに偽(0)、偽(0)のときに真(1)と出力

つまり真偽が逆転する

→ $\text{NOT}(0) \Rightarrow 1$ $\text{NOT}(1) \Rightarrow 0$

◇たとえば 1 入力の場合

論理関数の種類は 4 種類

▽具体的には

①入力 i が 0 でも 1 でも、出力 o が 0 となる論理関数

②入力 i が 0 なら 0、入力が 1 なら 1 を出力(o)する論理関数

③入力 i が 0 なら 1、入力が 1 なら 0 を出力(o)する論理関数

④入力 i が 0 でも 1 でも、出力 o が 1 となる論理関数

▽ {AND, OR, NOT} を使って表現すると

① $o = \text{AND}(i, \text{NOT}(i))$

② $o = i$

③ $o = \text{NOT}(i)$

④ $o = \text{OR}(i, \text{NOT}(i))$

②と③はわかりやすいと思います。

入力されたものをそのまま出力するだけだから $o=i$

入力されたものの真偽をひっくりかえして出力するだけだから $o=NOT(i)$ です。

①はどんな入力に対しても偽(0)と出力したい。そこで、AND と NOT を利用します。

$x=i, y=NOT(i)$ としましょう。

y は i の真偽を入れ替えたものですから、 x の真偽とも入れ替わってます。

つまり x が真のとき y は偽だし、 x が偽のとき y は真です。絶対に。

じゃあ $AND(x,y)$ は? $AND(x,y)$ では x,y がともに真のときだけ真と出力されます。

ところが上で言ったようにこの場合の x と y はともに真ではありえません。

じゃあもう偽(0)と出力するしかないじゃないですか!

今度は④ $OR(x,y)$ を検証してみましょう。

$OR(x,y)$ は x,y 少なくともどちらか一方が真のとき真と出力されます。

逆に言えば x,y がともに偽のときだけ偽と出力されます。

でも、 x,y がともに偽ではありえませんよね。

ほら、もう真(1)と出力するしかないじゃないですか!

これが①、④の考え方です。

◇今度は2入力

論理関数の種類は 16 種類 (図 7-5)

▽具体的には (x,y を入力する)

①入力が $x=y=0$ でも、 $x=0$ かつ $y=1$ でも、 $x=1$ かつ $y=0$ でも、 $x=y=1$ でも 0 を出力

②入力が $x=y=0$ 、 $x=0$ かつ $y=1$ 、 $x=1$ かつ $y=0$ なら 0、 $x=y=1$ なら 1 を出力

:

⑮入力が $x=y=0$ 、 $x=0$ かつ $y=1$ 、 $x=1$ かつ $y=0$ なら 1、 $x=y=1$ なら 0 を出力

⑯入力が $x=y=0$ でも、 $x=0$ かつ $y=1$ でも、 $x=1$ かつ $y=0$ でも、 $x=y=1$ でも 1 を出力

(x, y)	(0, 0)	(0, 1)	(1, 0)	(1, 1)
①	0	0	0	0
②	0	0	0	1
③	0	0	1	0
④	0	0	1	1
⑤	0	1	0	0
⑥	0	1	0	1
⑦	0	1	1	0
⑧	0	1	1	1
⑨	1	0	0	0
⑩	1	0	0	1
⑪	1	0	1	0
⑫	1	0	1	1
⑬	1	1	0	0
⑭	1	1	0	1
⑮	1	1	1	0
⑯	1	1	1	1

図 7-5

図 7-5 のように、2進演算の入出力の関係を表にしたものを**真理値表**といいます

▽論理関数表現

- ①は x, y がどんな入力になっても 0 を出力するという論理関数
→ **0** と表される (これをさっきは $AND(i, NOT(i))$ の形に変形しました)
- ②は x, y がともに 1 のとき 1 を出力するという論理関数
→ **$AND(x, y)$** と表される
- ④は x の真偽をそのまま出力するという論理関数
→ **x** と表される (上の例で言うと $o=i$)
- ⑥は y の真偽をそのまま出力するという論理関数
→ **y** と表される (上の例で言うと $o=i$)
- ⑧は x, y 少なくとも一方が 1 のとき 1 を出力するという論理関数
→ **$OR(x, y)$** と表される
- ⑪は y の真偽を逆転して出力するという論理関数
→ **$NOT(y)$** と表される
- ⑬は x の真偽を逆転して出力するという論理関数
→ **$NOT(x)$** と表される
- ⑯は x, y がどんな入力になっても 1 を出力するという論理関数
→ **1** と表される (これをさっきは $OR(i, NOT(i))$ の形に変形しました)
これらはさっきやりましたね
ここではさらに論理関数表現が増えます
- ⑦は x, y どちらか一方だけが 1 のとき 1 を出力するという論理関数
→ **$XOR(x, y)$** と表される (eXclusive OR の略 … 排他的論理和)
- ⑨は x, y がともに 0 のとき 1 を出力するという論理関数
→ **$NOR(x, y)$** と表される
(つまり $OR(x, y)$ の否定です、⑧と比べるときれいに真偽が逆転してますよね)
- ⑩は x と y が一致するとき 1 を出力するという論理関数
→ **$EQ(x, y)$** と表される (EQuivalence の略ですが EQual の略って覚えていいです)
… $XOR(x, y)$ と否定の関係にあります
- ⑮は x, y 少なくとも一方が 0 のとき 1 を出力するという論理関数
→ **$NAND(x, y)$** と表される (見ての通り $AND(x, y)$ と否定関係にあります)

▽またまた完備性の話

さっきも言ったように {AND, OR, NOT} は完備な組み合わせです。
なので、⑦⑨⑩⑮の論理関数表現も、この3つの表現だけに置き換え可能です。
これは後でやります。ちなみに NAND、NOR はそれぞれ単独で完備性を備えています。
つまり、すべての表現を NAND だけ、または NOR だけで表現可能なんです。

◆MIL 記法

回路に似せた論理関数表現法

基本的な論理関数を**基本素子** (図 7-6) で表し、その間の結線で論理関数を表現

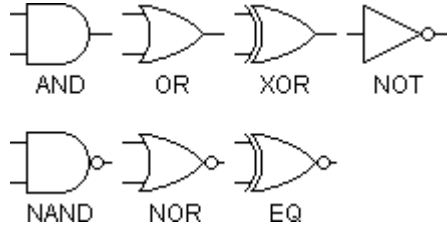


図 7-6

素子の図中の○は、**ビット反転** (0→1、1→0) を表している

◇XOR 回路の表現

では、さっき「後でやる」と言っておいた、

XOR(x,y)の {AND, OR, NOT} での置き換えをやってみましょう。

結論から言うと、XOR(x,y)という論理演算は、

$OR(AND(x, NOT(y)), AND(NOT(x), y))$

という訳のわからん形に置き換えられます。

これを MIL 記法で表してみよう！

まず一番外枠の OR の基本素子を置きます (図 7-7)。



図 7-7

次に、OR への入力は $AND(x, NOT(y))$ と $AND(NOT(x), y)$ 、

これは AND 基本素子から出力される値が OR に入力されるということなので、

OR 基本素子の2つの入力端子に AND 基本素子2つの出力端子を繋ぎます (図 7-8)。

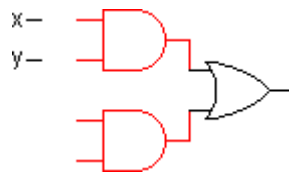


図 7-8

上の AND 基本素子には x と NOT(y)を入力します。

NOT(y)は y をビット反転させたものなので、

入力の直前に○を置いて反転させます (図 7-9)。

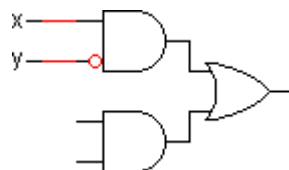


図 7-9

今度は下の AND 基本素子にも入力します。

NOT(x)と y を入力するので、y をそのまま引っ張ってきて、x の方は○をつけてつなぎます (図 7-10)。

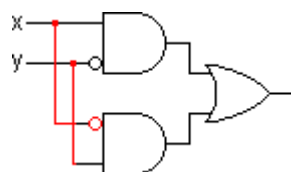


図 7-10

これで完成です (図 7-11)。

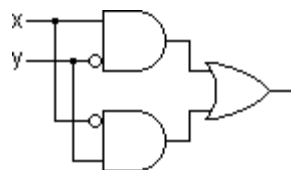


図 7-11

さて、本当に XOR(x,y) としての回路になっているのか確かめてみましょう。次の表を使って説明します (図 7-12)。

x	y	\bar{x}	\bar{y}	上の AND	下の AND	OR
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

図 7-12

まず入力パターンとしては(x,y)=(0,0),(0,1),(1,0),(1,1)の4通りあります。そのそれぞれに対し、(NOT(x),NOT(y))=(1,1),(1,0),(0,1),(0,0)となります。XOR(x,y)はx,yのどちらか一方だけが1のとき1を出力するものですから、(x,y)=(0,0),(0,1),(1,0),(1,1)に対して、それぞれ0,1,1,0と出力されればOKですね。まずは上のAND基本素子から見ていきましょう。

上のAND基本素子にはxとNOT(y)が入力されます。

表の赤くなっている部分です。

AND(x,y)はx,yがともに1のときに1を出力しますから、

(x,NOT(y))=(1,1)のとき、すなわち(x,y)=(1,0)のときだけ1を出力します。

次は表の緑色になっている部分、下のAND基本素子について見てみましょう。

NOT(x)とyが入力されるんです。

上から2段目、(NOT(x),y)=(1,1)のとき、

すなわち(x,y)=(0,1)のときだけ1を出力します。

では、この2つの出力がORに入るとどうなるか。

(x,y)=(0,0)のとき、(上のANDからの出力,下のANDからの出力)=(0,0)

(x,y)=(0,1)のとき、(上のANDからの出力,下のANDからの出力)=(0,1)

(x,y)=(1,0)のとき、(上のANDからの出力,下のANDからの出力)=(1,0)

(x,y)=(1,1)のとき、(上のANDからの出力,下のANDからの出力)=(0,0)です。

ここのORは、上のANDからの出力と下のANDからの出力、

どちらか一方が1のとき、1を出力します。

ですので、最終的には $(x,y)=(0,1),(1,0)$ のとき、1が出力されることになります。

$(x,y)=(0,0),(0,1),(1,0),(1,1)$ に対して、それぞれ0,1,1,0と出力されるわけです。

これってXOR(x,y)と同じですよ。

こんな説明でわかるのかしら…

◇EQ、NOR、NANDの置き換え

これは簡単ですね。

否定の関係にあるものを、NOT()でガバッとくくってしまえばオッケーです。

$NOR(x,y) \rightarrow NOT(OR(x,y))$ みたいだね。

【加算器】

◆2進数の加算の仕方

10進数と同じ原理 … 各桁ごとに加算 → 結果が2以上なら繰り上がり

2進数の下からn桁目の足し算をすると仮定

足し合わせる2つの数をx、y、

n桁目に残る数をs、

n+1桁目に繰り上がる数をcoutとする。

すると図7-13のような関係になる。

x	y	s	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

図7-13

また、n-1桁目からの繰り上がりをcinとすると、

今度は図7-14のような関係になる。

x	y	Cin	s	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

図7-14

ではこの演算を行う回路はどのようになるか。

まずは図7-13の2入力2出力の回路から考えてみましょう。

x,yとsとの関係をよく見てください。

xとy、どちらか一方だけが1のとき、sは1になってます。

これってどっかで見たことないですか？そうです、XORです。

つまり、 $s=XOR(x,y)$ で表現できるんです。

そしてx,yとcoutとの関係もよくよく見てみてください。

ANDの関係が見えます。つまり $cout=AND(x,y)$ です。

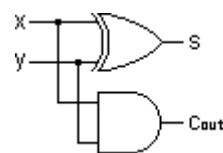


図7-15

これをMIL記法で表すと図7-15のようになり、

この、下からの繰り上がりが考慮されない加算器を(1ビット)半加算器といいます。

下からの繰り上がりが考慮される加算器を(1ビット全加算器)といい、

2つの半加算器とOR基本素子を用いて図7-16のように構成することが出来ます。

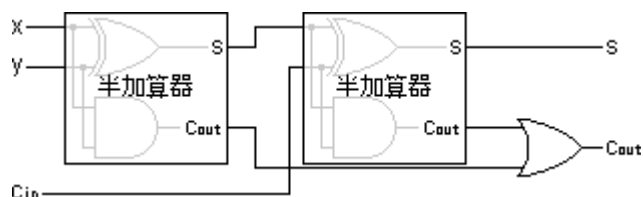


図7-16

◇nビット全加算器

でも、これではまだ $1011+1001=10100$ という計算はできません。
このような n ビット（この例では $n=4$ ）の加算するにはどうするか。
図 7-17 のように、1 ビット全加算器をいっぱい繋げるのです。

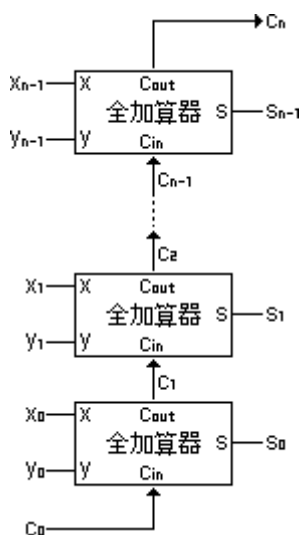


図 7-17

このように、半加算器 2 つを組み合わせることで 1 ビット全加算器を構成し、
1 ビット全加算器を組み合わせることで n ビット全加算器を構成する、
という具合に、簡単な回路を組み合わせることで複雑な回路を階層的に構成する方法を、
モジュール化という。

各段階における中間的な構成要素（この場合は半加算器や 1 ビット全加算器）を
モジュールという。

こうやって簡単かつ確実な技術を組み合わせることで、
間違いの入り込む余地を減らす作戦らしいのです、はい。

第8章

技術的な話からは抜け出したんで多少楽なんじゃないでしょうか。シケプリもいよいよ後編、ゴールはもうすぐです！みんな頑張ろうじゃないか。

【見えにくい情報システム】

◆情報システムとは

広義 → 情報処理機器（コンピュータなど）+情報伝達ネットワーク

多様なサービスや機能を提供するシステム

計測・制御系システム、組み込みシステム、ゲーム機なども含む

狭義 → いわゆるビジネス・アプリケーション

… 企業、政府機関、サービス機関などが、

①外部に提供する情報サービスシステム

②内部で用いる、情報管理、意志決定支援システム

→大学でいえば学生証番号・成績の管理など

◆見えない情報システム

ソフトウェアは抽象的記述にすぎず、物理的実体がない

◇組み込みシステム

特定の機能を実現する目的でコンピュータを組み込んでいる、特定目的のシステム

… 携帯電話や炊飯器、車、デジカメなどなど

→ 身近すぎて情報システムとして認識されにくい

◇パソコン … もっともよく目にするコンピュータ

ネットワークの向こう側で見えないコンピュータが多数稼働している

【情報システムの仕組み】

◆チケット予約システムのサービス

◇チケット予約システムの利用者

顧客 … チケットを予約する

主催者 … 公演の日時、席の区分や値段など予約に必要な情報をシステムに与える

システムの管理者 … システムが問題なく稼働し続けるようにシステムを保守

◇顧客から見えるチケット予約システム—提供される機能

▽一般情報照会

・ジャンル別、地域別公演照会

・チケット販売スケジュール照会

▽公演詳細情報照会（顧客が目的の公演を指定すると見られる）

- ・公演内容（日時、演目）
- ・公演者（名前、プロフィール）
- ・公演会場（会場名称、住所）
- ・チケット情報（座席別料金、座席表）
- ・予約関連情報（予約受付先、予約受付期間、予約状況）
- ・主催、協賛（名前、住所）

▽予約機能

- ・公演の指定
- ・チケット受け取り方法指定
 - 郵送を指定すると、続いて郵送先住所、受取人氏名、電話番号などの入力
- ・予約確定
- ・決済方法指定
 - クレジットカードを指定すると、
続いてカード会社名、カード番号、有効年月、カード所有者名などを入力

▽付随機能

- ・誤入力の防止、誤入力の処理
 - Ex.誤入力防止の工夫
 - 16桁の番号を入力する場合に入力欄を4桁ずつ区切る
 - パスワードの確認入力
- ・キャンセルの処理

◆チケット予約システムの仕組み

ウェブブラウザからの利用を前提としたサービスが一般的

◇クライアント／サーバ型の構成（図 8-1）

サーバ … サービスを提供するプログラム、また、それを搭載したコンピュータ

クライアント … サーバに対してサービスを要求するプログラム

アプリケーション（適用プログラム）

…（この場合はチケット予約に関する）具体的な処理を行うプログラム

Web サーバから呼び出されて処理を行う

データベース … 通常、公演情報や販売状況データが格納される場所

アプリケーションがここにアクセスして情報を得る

▽クライアントーサーバの関係①

Web クライアント（Web ブラウザ）ーWeb サーバ

▽クライアントーサーバの関係②

アプリケーションーデータベース

利用者が直接ウェブサーバにアクセスするのではなく、
 電話やファックスでチケット予約センターに連絡をする場合は、
 チケット予約センターの受付オペレータが、
 利用者に代わってウェブサーバを参照する

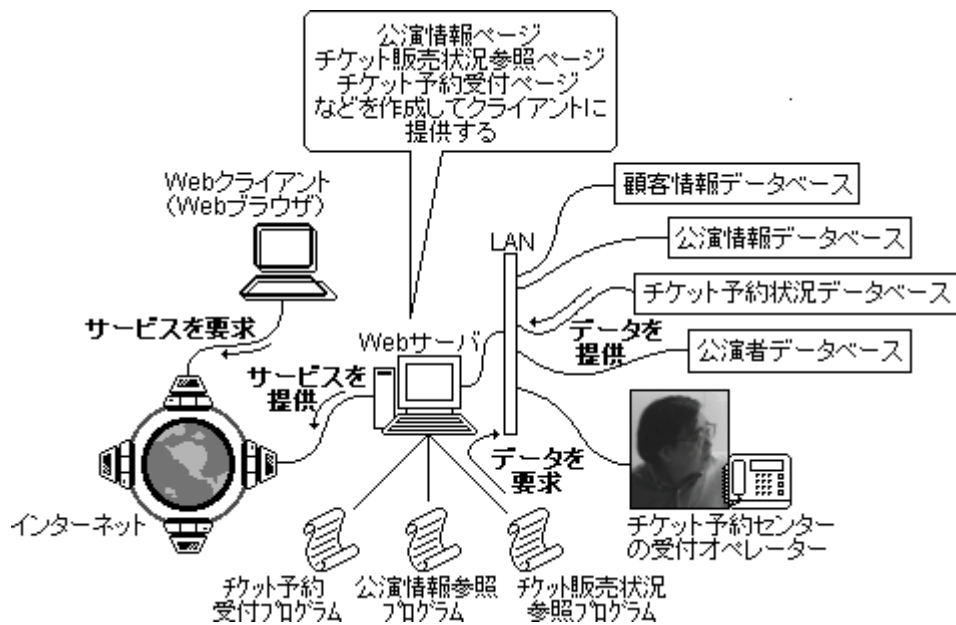


図 8-1

これ全部（写真は除く）自分で書いた！偉くないですか？はい、どうでもいいです。

第9章

世の中、かくも使いにくい物ばかり？と、いうことでね。まあ、情報機器さんたちからしてみたら「世の中こんなに暮らしやすくしてやってなのに文句言うんじゃないか」って感じなのかもしれませんが、使いにくい物は使いにくいです。たぶんこのシケプリも簡潔性がなくて使いにくいと思います。でもあと2章で終わりだよ！頑張ろうじゃないか。

【こんな経験ないですか】

◆暮らしの中の使いにくさ

- ・ ドアを押すのか引くのかわからない
- ・ 電灯とスイッチの位置が無関係
- ・ 携帯電話の機種によって使い方が違う
- ・ ワープロが勝手に入力を変更する（オートコレクト機能）

◆設計者とユーザの齟齬

Ex. 券売機の硬貨投入口

- コイン詰まりを防ぐための棒が埋め込まれている
- 「500円玉は右と左、どちらに入ればよいのですか？」

インタフェース設計者の工夫（device）が予期せぬ形でユーザに解釈されることがある
インタフェースが設計者や人工物（artifact）とユーザとを結ぶ唯一の接点

【インタフェースの定義と機能】

◆定義

インタフェース … もととは「界面」「接面」など、2つの異なる存在の境界面のこと
本章で扱うのはユーザインタフェース、ヒューマンインタフェース
… コンピュータなどの人工物とユーザ（人間）との間の接面

◆機能

ユーザはインタフェースを通して人工物を操作

- 人工物が機能を最大限に発揮するためには、使いやすいインタフェースが必要

◇ドライバという（古典的な）人工物

力を伝達したり増幅したりする機能をもつ人工物のひとつ

固いねじには握りの太いドライバ、

細いねじや折れやすいねじには握りの細いドライバが適切

人間の道具（ドライバ）への働きかけが、そのまま対象（ねじ）への働きかけとなる

- ◇コンピュータという（複雑な）人工物
- 道具への働きかけが対象への働きかけと等価とは限らない
- ▽エンターキーを押すという操作
 - 目的は力の伝達や増幅ではなく、情報（意志決定）の伝達
 - 伝達対象は状況によって変わる
 - 漢字変換の確定、ミサイルの発射命令、改行 等

【インタフェースの二重界面性】

◆佐伯先生の「第一界面と第二界面」の考え方（図 9-1）

第一界面 … ユーザと人工物との間に存在する直接的なインタフェース
操作インタフェースともいう

第二界面 … 人工物と物理的なタスクとの間に存在する間接的なインタフェース
制御インタフェースともいう

ユーザの目的は第二界面における物理的なタスクの実行

→ 実際に操作できるのは第一界面

ドライバなどの古典的な道具ではほぼ第一界面＝第二界面

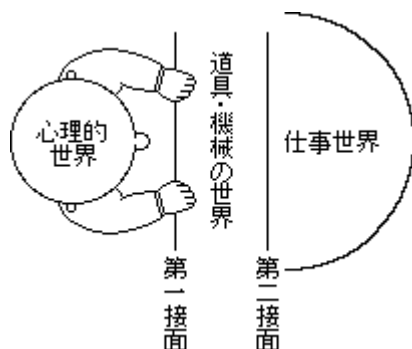


図 9-1

◆コンピュータの汎用性に由来する使いにくさ

◇道具・機械の世界から仕事世界へ（第二界面）の対応づけは多様

- ・ワープロによる文書の作成
- ・データベースソフトによるデータの管理
- ・数値計算プログラムによる建物の構造計算
- ・ブラウザによるホームページの閲覧 などなど

◇心理的世界から道具・機械の世界へ（第一界面）の対応づけは限定されている

- ・キーボード
- ・マウスなどのポインティングデバイス
- ・ディスプレイや GUI (Graphical User Interface)

第二接面での多様な機能を、第一接面の限られたデバイスで操作可能に

→ エンターキーを押した結果が多岐に渡るという事態は、このような理由による
逆向きの対応づけに関しても同じ

→ ソフトウェアの多様な実行結果が、限られた出力デバイスにより提供
(プログラム実行時のエラーメッセージの表示など)

→ ユーザはそれらの情報を元に、自らが操作の善し悪しを解釈する必要

【まとめ】

◆インタフェースのまとめ

◇インタフェースは、システム設計者とユーザとがコミュニケーションできる唯一の場

◇システムのユーザビリティ（使いやすさ）の大部分は、
インタフェースのユーザビリティによって左右される

◇インタフェースの構成には、
ユーザの行動特性を考慮し、ユーザ中心の設計思想が求められる

第 10 章

いよいよ最後の章です。いや一長かった。でもこれさえ終われば後はテストを残すのみ！
頑張ろうじゃないか。

【技術と社会】

◆技術と社会

情報リテラシー … ただ単にパソコンが操作できるという意味のリテラシーではない
マウスでクリックしたときに“裏で何が動いているのか”について
おおまかな想像が及ぶ程度のリテラシーを意味する

◇技術と社会を論じる際の4つの論点

- ①技術の中立性（後述）
- ②技術と民主主義
- ③技術倫理
- ④リテラシー論（後述）

◆情報技術による技術上の変化

◇コンピュータ技術の変遷

▽1960年代と1970年代

巨大データベースの管理と大規模計算

→ 権力の集中、個人のプライバシー保護が論点に

▽1980年代

マイクロコンピュータの誕生

消費者市場の中にコンピュータ技術が参入

ソフトウェアの所有権に関わる問題が出現

→ ソフトウェアは物理的実体がなく、複製も容易である

人工知能への関心

▽1990年代以降

インターネット技術の普及

ICT（情報コミュニケーション技術）に関する論考が増えてきている

プライバシーの問題が再燃 → 著作権、匿名性の問題

◇メディアの発展

インターネット技術を含めた情報技術の影響は幅広い

それまでの社会の体制や権力を支えていた構造をひっくり返す力を持つ

社会を変革させる力を持つ … このような論考を**メディア論**という

活版印刷 → 聖書の普及と宗教改革 映像技術 → 社会主義の崩壊

◇インターネット技術

ホストコンピュータ管理による閉じられた世界から、
村（ネットワーク）中心の開かれた世界へ（図 10-1）

1つ1つのコンピュータが独立にネットワークに繋がることが出来る技術
→ 相対的に、中央集権的なやり方（ホスト対端末）を弱める

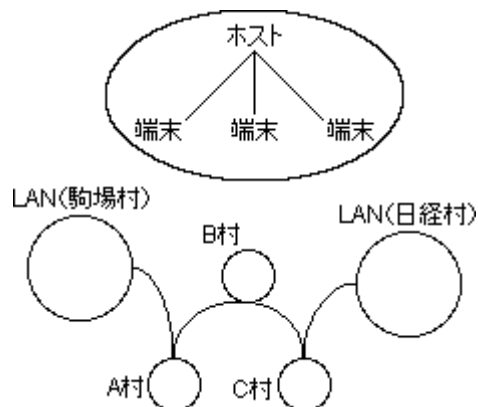


図 10-1

◇4つの変化

- ①場所の制約からの解放
- ②時間の制約からの解放
- ③経路の制約からの解放
- ④輸送コストをほぼ、ゼロとした

Ex. 日本からアメリカへ秒単位でEメールが送れる

→ ①～④すべて当てはまる

▽以前の活版印刷や映像技術との違い

以前 → 4つの変化は「作り手 → 大衆」という経路に関してのみ当てはまった

今回 → 4つの変化は「1対1レベルのコミュニケーション」の経路でも成立

コミュニケーション形態を一変

… 地理的要素に依存しないコミュニケーションが可能

→ 国家や地域共同体のような地理的要素に依拠した社会を相対化

【情報技術に固有な社会との転換】

◆ 2つの転換

①無形性と複製可能性 (by ジョンソンさん)

… ソフトウェアやデジタルコンテンツは従来の“もの”媒体とは異なる性質を持つ

→ 所有と権利に関わる法制度や倫理との間での転換

②インターネットのグローバルな通信射程と匿名性

→ ネット上での自由で規制のない議論空間、公共空間を構築、電子民主主義の促進

→ プライバシーやセキュリティの転換

◇権利と所有概念

従来の“もの媒体” → “電子媒体”

→ 複写する行為や、著作権の侵害に対する利用者の意識の希薄化

▽著作権法

知的財産権 … 知的な創作活動によって何かを作り出した人に対して、
他人に無断で利用されない権利を付与する制度
著作権、工業所有権（このなかに特許権が含まれる）など

著作権法第一条

この法律は、著作物並びに実演、レコード、放送及び有線放送に関し著作
者の権利及びこれに隣接する権利を定め、これらの文化的所産の公正
に留意しつつ、著作者等の権利の保護を図り、もって文化の発展に寄与
することを目的とする。

▽著作権法第十条による著作物の例示

- 1.小説、脚本、論文、講演その他の言語の著作物
- 2.音楽の著作物
- 3.舞踏または無言劇の著作物
- 4.絵画、版画、彫刻その他の美術の著作物
- 5.建築の著作物
- 6.地図又は学術的な性質を有する図面、図表、模型その他の図形の著作物
- 7.映画の著作物
- 8.写真の著作物
- 9.プログラムの著作物

◇コンピュータプログラムの著作権

昭和 60 年の著作権法改正でプログラムの定義が明記された

著作物として保護されるもの

… ソースプログラム

オブジェクトプログラム（ソースプログラムをコンパイルしたもの）

オペレーティングシステム（OS）（Windows とか Mac とか）

アプリケーションプログラム（Word とか Internet Explorer とか）

著作物として保護されないもの

… プログラム言語（これ保護されたら新しいプログラム作れません）

規約

解法

▽著作権侵害の事例

事例：A社はビデオゲームXを開発し、その著作権を所有している。B社は都内で経営する喫茶店にゲームXの無断複製ビデオゲームを設置して、顧客に利用させた。A社は著作権侵害を訴えることができるだろうか？

ビデオゲーム機に取り付けられたROMに収納されているオブジェクトプログラムは、A社の著作物（ソースプログラム）の複製物である。したがって、Bが使用したビデオゲーム機のように、ROMのオブジェクトプログラムを他のROMにコピーして製造した偽造ゲーム機は、Aのソースプログラムの著作権を侵害する。

◇デジタルコンテンツの著作権

使用許諾の概念が曖昧

- … メモリ上の電子情報は市場や流通機構なしに直接個人の手へ渡ることが可能
- **DRM (Digital Rights Management)** が問題となる

▽Winny 開発者の基礎問題

Winny … ファイル交換ソフト WinMX の改良版

ファイル交換が完全に匿名化されていることが特徴

- 匿名性の高さ故に、違法行為を行なってもそのような意識が希薄
- 多くの音楽や映像、ゲームや他のソフトが著作権者に無断でやりとり
- デジタルコンテンツの著作権の侵害を幫助する側面を持つ

アメリカでも似たような事件があった … Napster 裁判

- 開発者が逮捕されたわけではない
- 開発者が逮捕された Winny 事件は世界的にも異例な事件

起訴した側の主張

- … 現行の法律から鑑みて、
- 開発者の行動は著作権侵害を幫助するという意味で罪である

開発者の支援者や技術者らの主張

- … 技術の進歩とともに法律も進化しなくてはならず、
- 現在の技術によって簡単に著作権法違反が発生してしまう現状のほうが問題
- 裁判の結果にかかわらず興味深い事件 … 論点の幅が大きい
- 技術開発と法、倫理のありかた、市場、文化の保護、自由と規制など、
- 多くの論点を含んでいる（もっと詳しい解説は教科書 p.242 脚注）

【情報技術論】

◆技術は中立か

技術本質主義 … 技術は、社会の形態や要望にかかわらず

“独立に” 発展する、という立場

技術の価値中立論 … 社会への良い面も悪い面も技術の“影響”にすぎず、

技術自体に問題はない、という立場

技術の社会構成主義 … 社会は自身にとって必要な技術をそのつど選択してきたのだ、

という立場

→ 社会の成員による選択と技術とが**共進化**

技術は価値中立でなく、社会の価値により取捨選択される

◇社会構成主義的見方

現在もさまざまな情報技術や情報をめぐる制度が

選択淘汰されつつある時代に生きている、という感覚

→ われわれ一人一人の選択が、次世代の技術に影響

→ 技術開発と同時に社会への影響を考え、選択を公に開くことが必要

◆情報リテラシー

使いやすいインタフェースの設計

→ “裏で何が動いているのか理解しないままにマシンを使う” ユーザの増加

◇情報における批判的思考 (クリティカル・シンキング)

自分の操作の裏で何が動いているのかについて、ある程度論理的に考える能力

Ex. 添付ファイル来た → マウスでクリック、開いてみた → うわぁウイルスだ

→ “マウスでカチッの裏側で何が動いているのか” について最低限の理解の必要

また、情報を主体的に選択、収集、活用、編集、発信するうえでの批判的思考も含む

◇他の技術リテラシーとの違い

情報技術の日常生活への浸透度が高い+日々の技術革新の速度が速い

→ 広範囲の人に情報における批判的思考の育成が必要

→ 情報技術の特徴 (多対多の通信射程、匿名性、複製可能性) を押さえたうえでの

批判的思考が必要

【これからの世代の情報】

ラストです。いよいよみなさんとのお別れが近づいてきました。ところで、ここテストに出るのかな？絶対出ないと思います。でもなんか、藤垣裕子助教授のメッセージがこもってそうなので、教科書をそのまま抜粋します。って、本当は要約するのが面倒臭かっただけです、ごめんなさい。それではまた会う日まで。31日かな。See you next time!!!

10.5 これからの世代の情報

これまでの節で概説してきたように、情報技術の普及によって、よい側面（意志決定への直接参加の機会の促進、民主的な公共空間の創出など）とダークサイド（セキュリティとプライバシー、有害情報の規制、既存の社会規範が追いつかないためのさまざまな問題）の双方が同居し、またそれが技術の発展とともに加速されている。同時に、技術が新しい技術にとって代わる、会社が新しい会社にとって代わる、法律が新しい法律にとって代わる、といった社会の新陳代謝も加速されている。

技術と社会規範の新陳代謝が加速されるなかで、これからの若い世代は、面白いものにぶつかるチャンスもあるし、危ないものにぶつかるチャンスも多々あるだろう。その場面場で技術開発者、利用者、法律の専門家や倫理の専門家などの相互の議論によって解決しなくてはならない問題も多く発生するだろう。しかし、技術の社会構成主義のところでもみたように、われわれのいまの選択は、将来世代の情報技術に影響を与えるのである。文明時代の野蛮人になるのではなく、問題が発生するごとにそれらの議論に“参加”し、新しい社会規範の構築に“参加”できるだけの情報リテラシーを身につけて、これからの社会を生き抜いてほしいと思う。