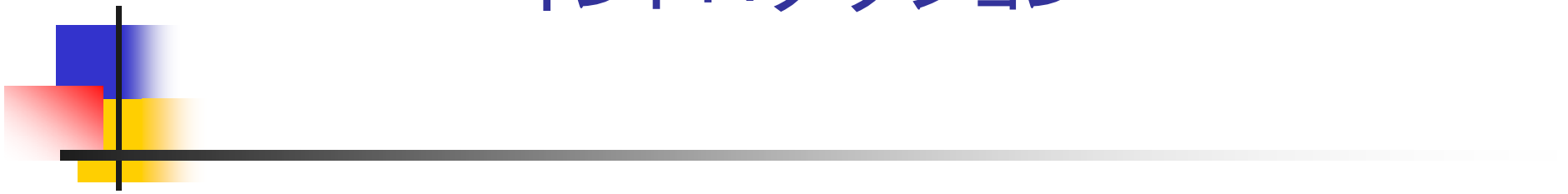


言語・オートマトン
イントロダクション

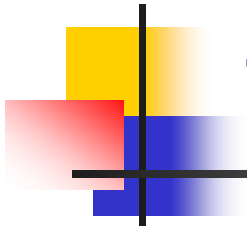
山本章博
情報学研究科 知能情報学専攻
2017.10.2

連絡先: akihiro@i.kyoto-u.ac.jp

2016年度
言語・オートマトン

山本章博

連絡先: akihiro@i.kyoto-u.ac.jp
075-753-5995

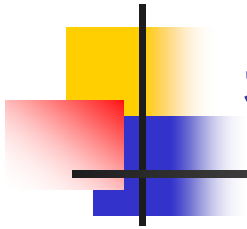


講義計画

内容:本講義では，形式言語理論と計算理論の初歩を扱う.

形式言語理論は，プログラミング言語を設計する基盤になっているほか，マークアップ言語や自然言語処理，生命情報学など現代の情報学の様々な分野で応用されている．また，計算理論は計算機による処理の特徴と限界を明らかにする理論である.

本講義では，有限オートマトンについて述べ，さらに文脈自由言語や チューリング機械，帰納的関数などについて講述する．また，これらの応用についても適宜言及する.



教科書・参考書

参考書

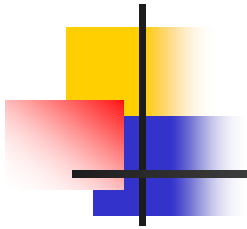
- Hopcropft, Motowani, Ullman: Introduction to Automata Theory, Languages, and Computation: 3rd edition, Peason, 2007
- 富田・横森：オートマトン・言語理論:第2版,森北出版 2013
- 岡留：オートマトンと形式言語入門, 森北出版, 2009
- 有川(監修)西野・石坂(著) 形式言語の理論, 丸善出版, 1999

講義資料

- (ほぼ)毎回, 講義資料をKULASISを用いて電子的に配布して使用する.
  - KULASISが利用できない場合は山本(章)のHP で配布

評価方法

- 小テスト (随時)＋定期試験
  - 小テストを行わない週には，講義の理解度を確認する演習問題を出題（評価に含めない）



形式言語理論

言語を文字列・単語列の集合と抽象化した上で

- 文を生成（構成・定義）するための理論
  - 正しい文字列・単語列を生成（定義）する
- 文を受理するための理論
  - 送られてきた文字列・単語列が正しいかどうかを判定する

- 句構造文法と受理機械

文法の型

- 文脈自由文法(context free) 文法中の全ての生成規則$\alpha \rightarrow \beta$ について，αが非終端記号一つであるもの
  - 生成規則に制限を加えることで，様々な文法の型が導入できる.

0型	句構造文法	Turing Machine
1型	文脈依存文法	線形有界オートマトン
2型	文脈自由文法	非決定性プッシュダウン・オートマトン
3型	正則文法	有限状態オートマトン

計算機科学と言語

情報とメディア

- 五感：発話，手話，表情，…

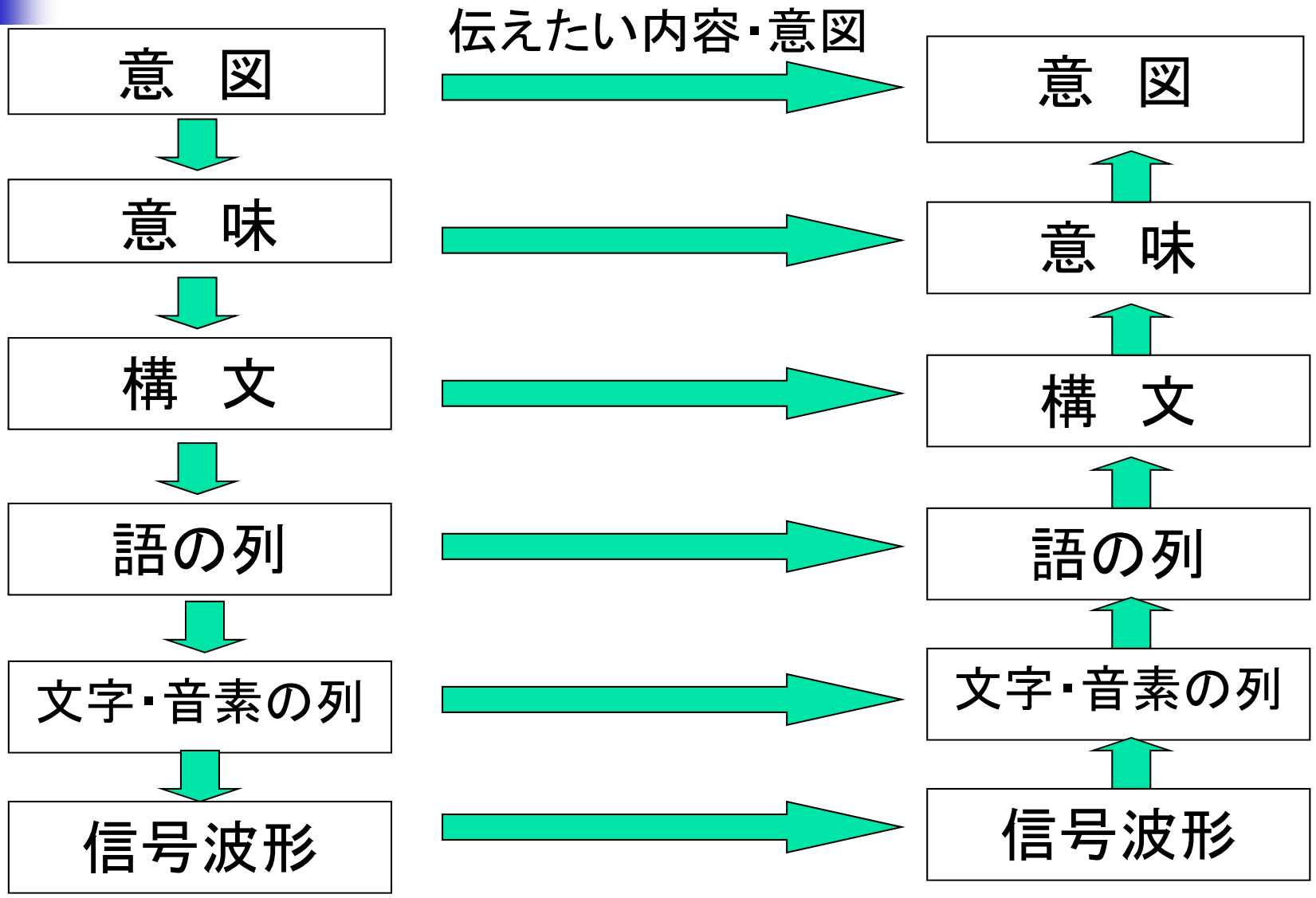
- 媒体(media)：

  手紙，書籍，新聞，電話，入試問題，…

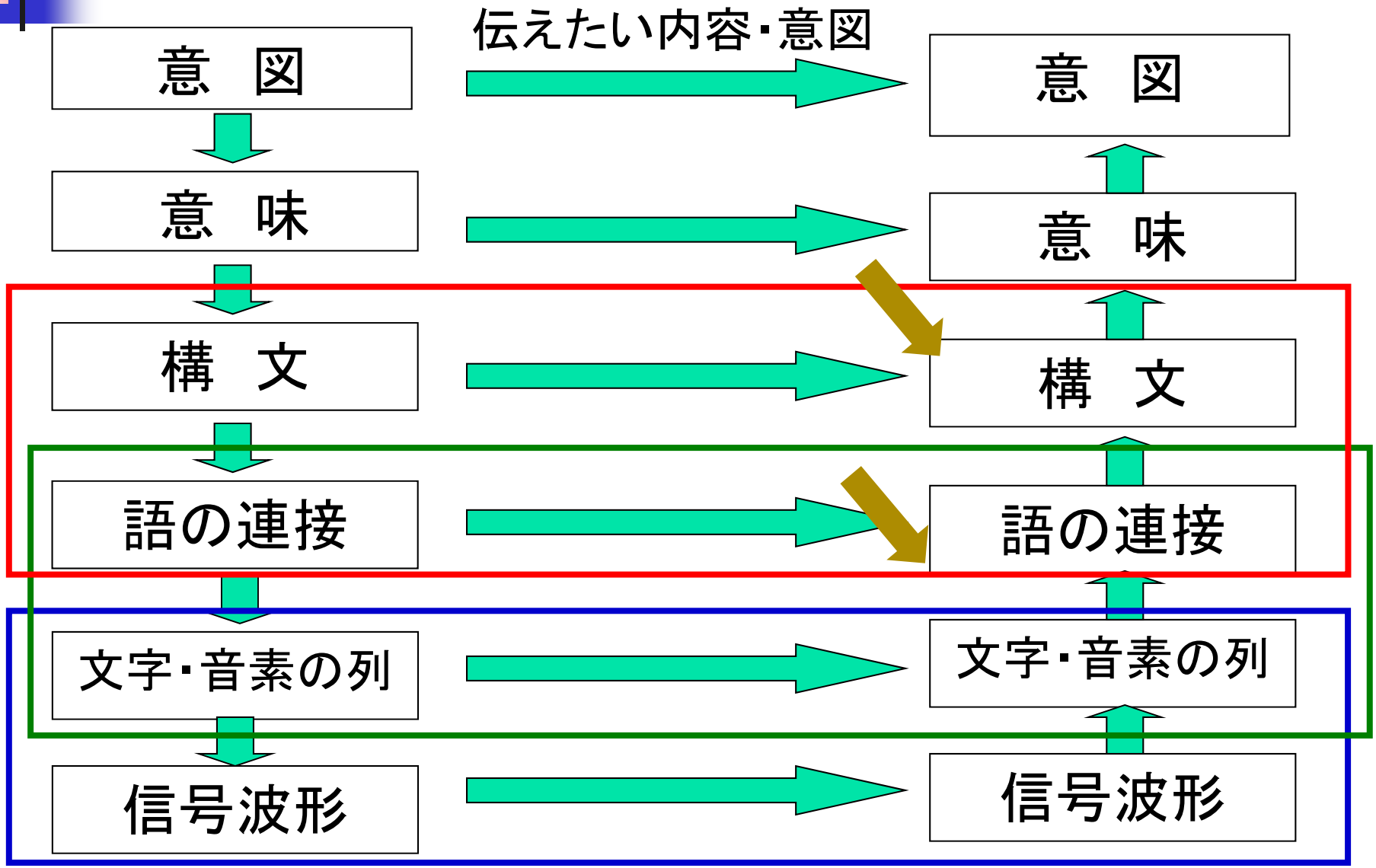
  Telephone，Telefacsimile，Television，…

  Tele：(Gk) far off

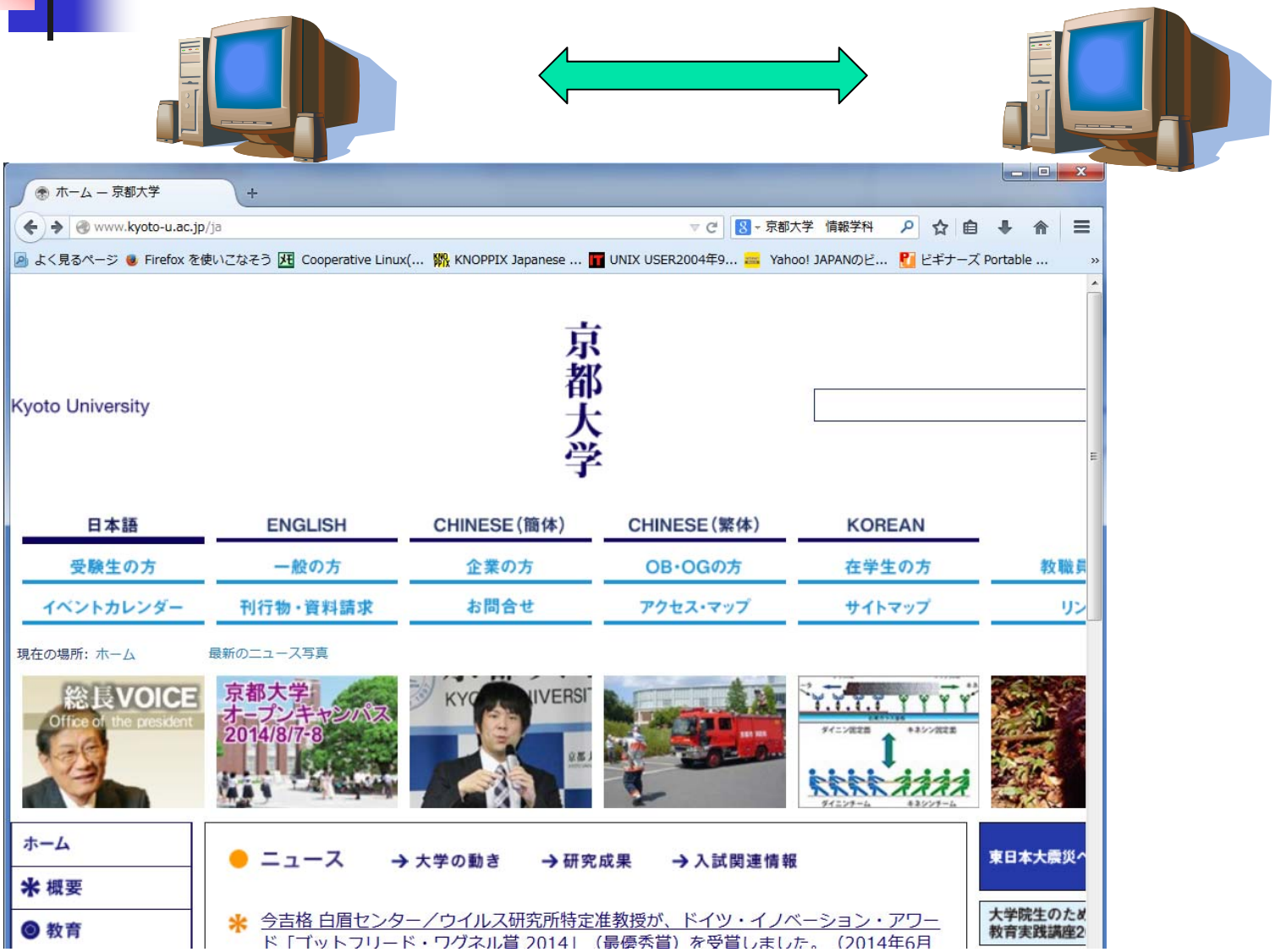
言語の構成要素と伝達



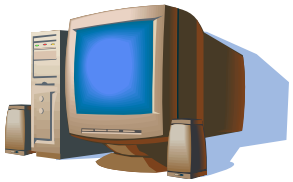
言語の構成要素と伝達



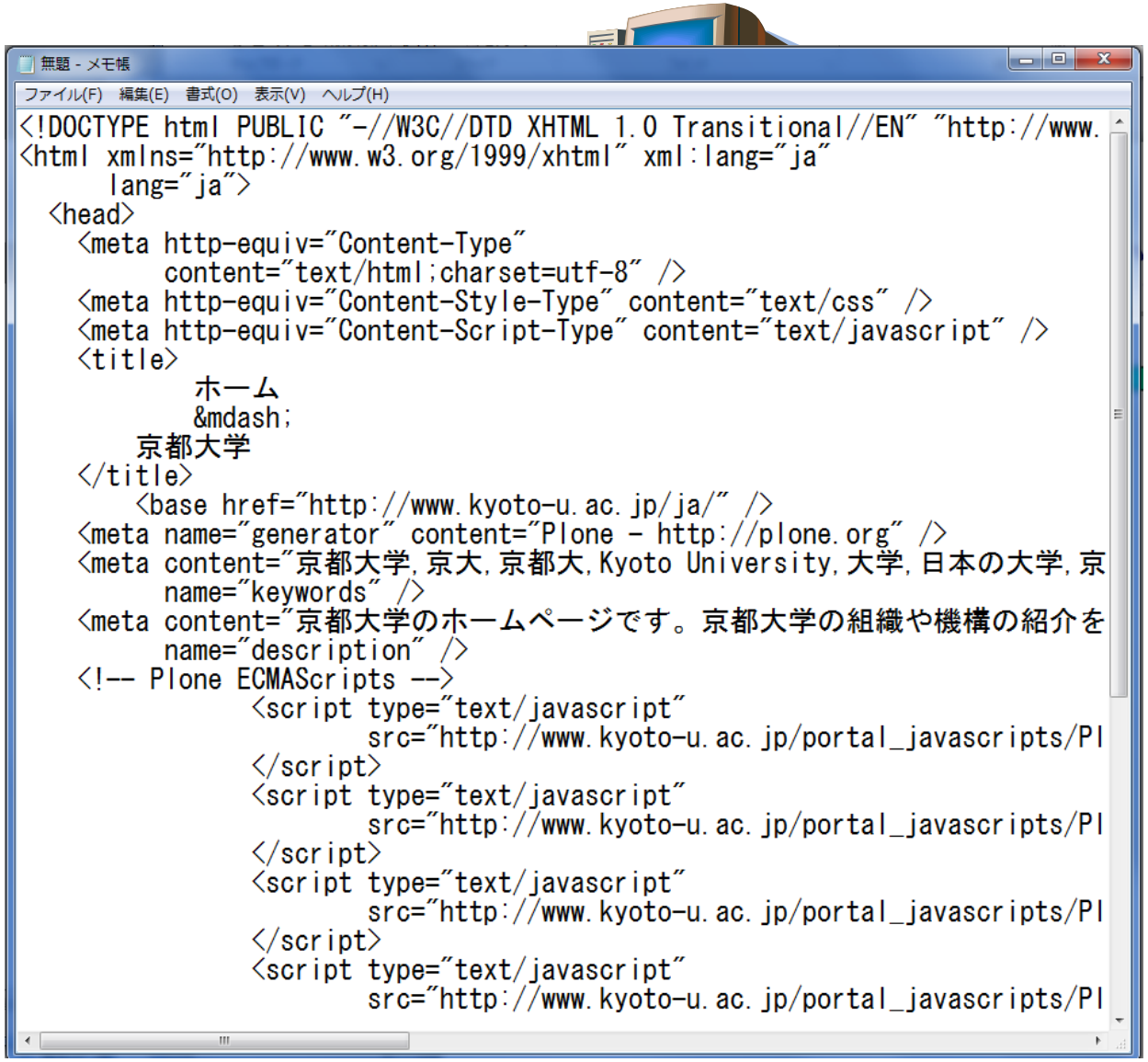
コンピュータ・ネットワーク



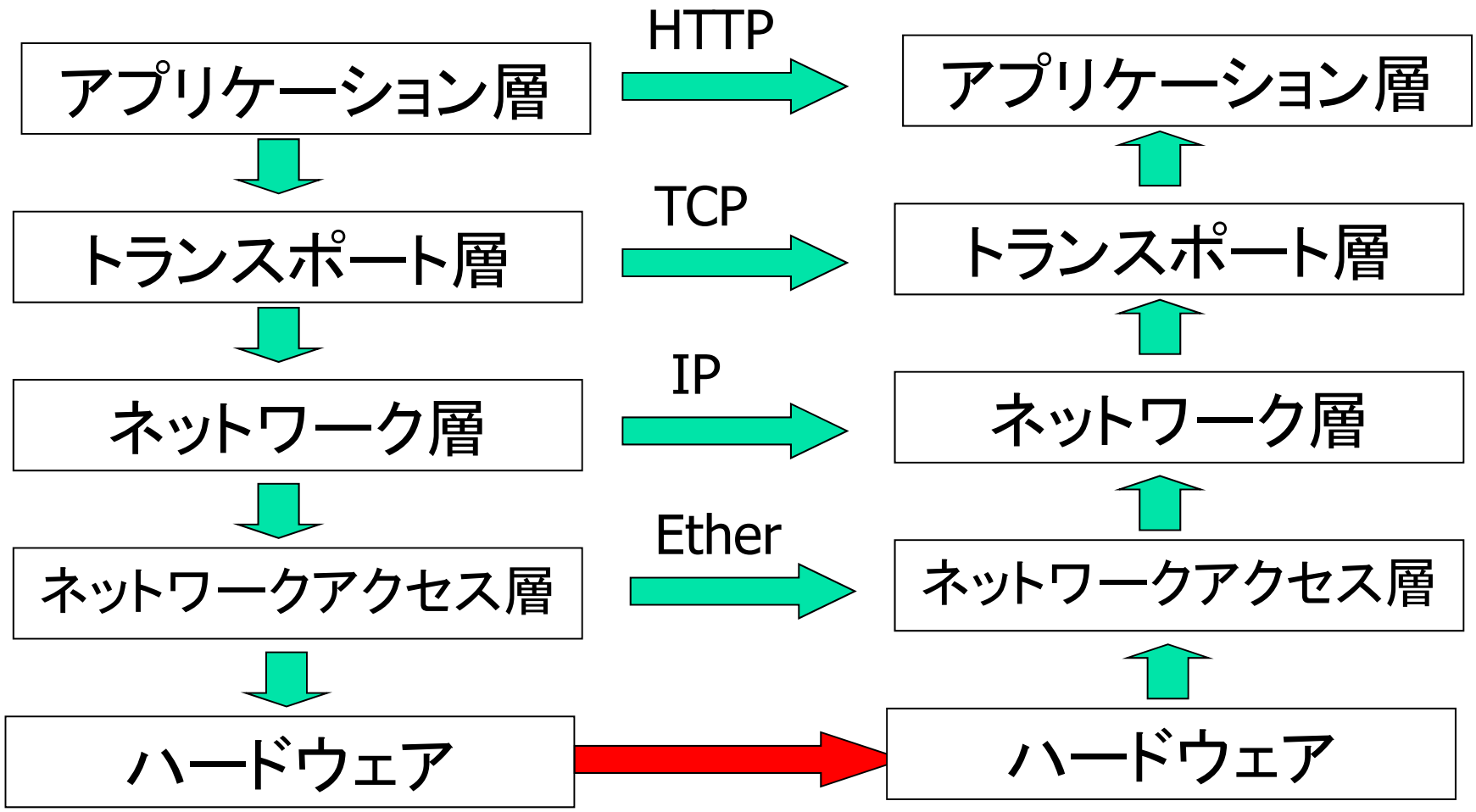
コンピュータ・ネットワーク



- HTML
- 文字列
- ビット列
  (0, 1の列)
- 電気(光)信号



TCP/IP プロトコルの階層



自然言語の“文”の例

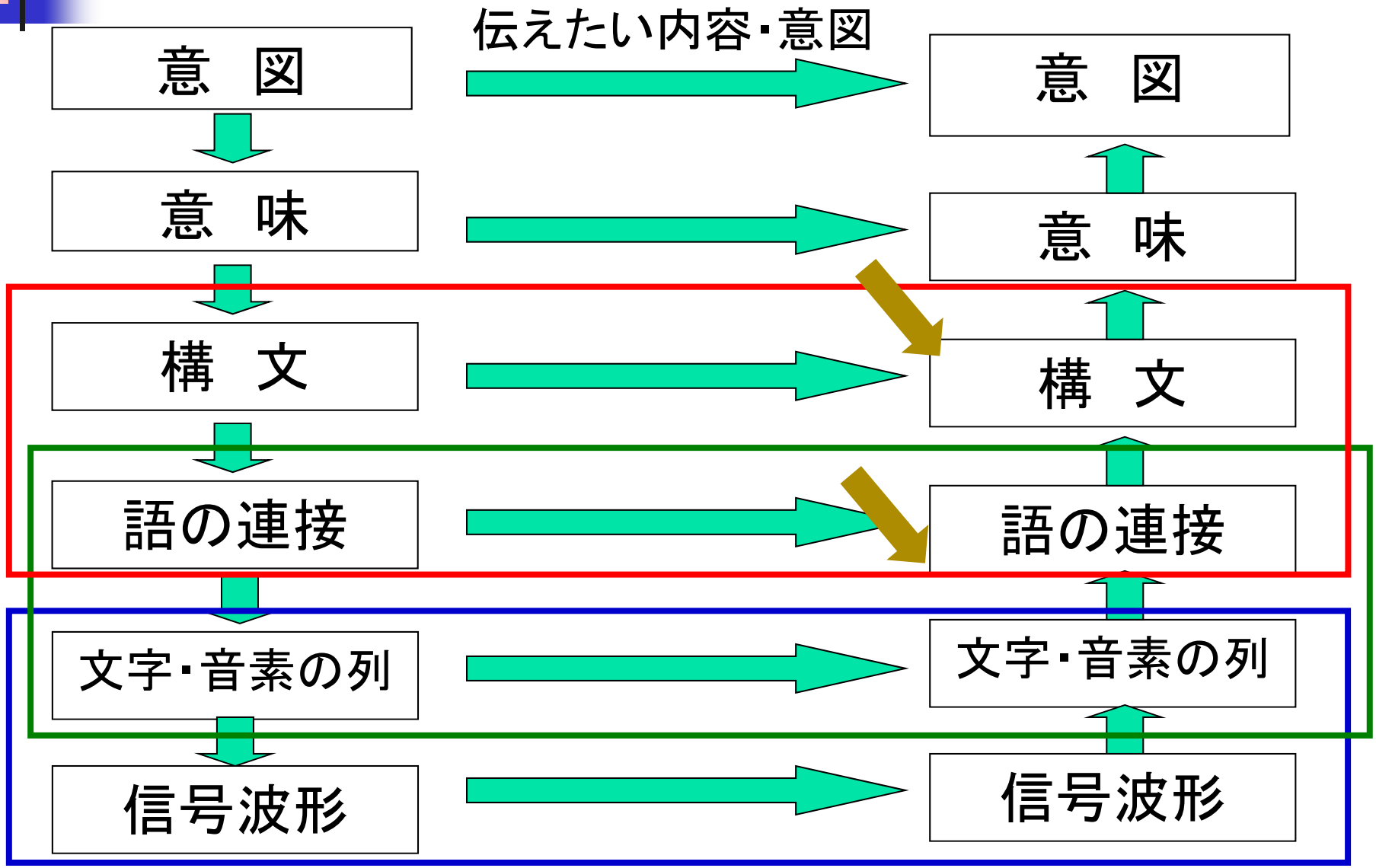
- 私は京都が好きです.
- 京都は私のお気に入りです.
- 京都って, いいよね.

- I like Kyoto.
- Kyoto is one of my favorite cities.
- Kyoto, my favorite!

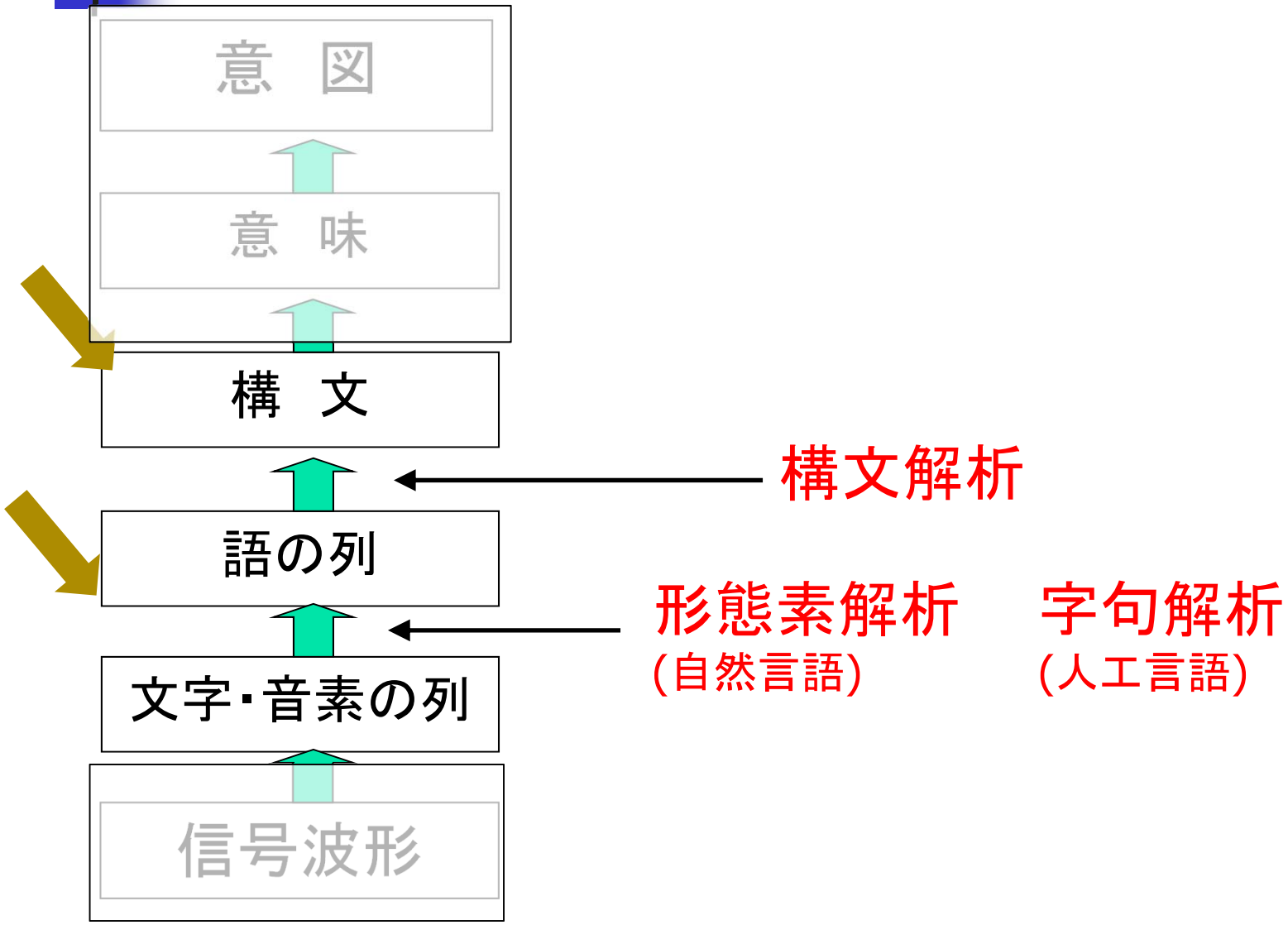
計算機科学と“言語”(1)

- 自然言語: 日本語，英語，ドイツ語，…
  - 人間が人間に意図を伝える
  - 文構造と意味は人間の知的活動中に成立
    文法は後から構成している
    - 国際補助語 (エスペラント…)
- 人工言語(形式言語)
  - プログラミング言語(C, C++, Java, Scheme,…)
    人間がコンピュータに指示する
  - マークアップ言語(HTML, XML,…)
    コンピュータ間のデータ授受
  - 文構造と意味は設計され，定義されている

言語の構成要素と伝達



言語の処理の流れ



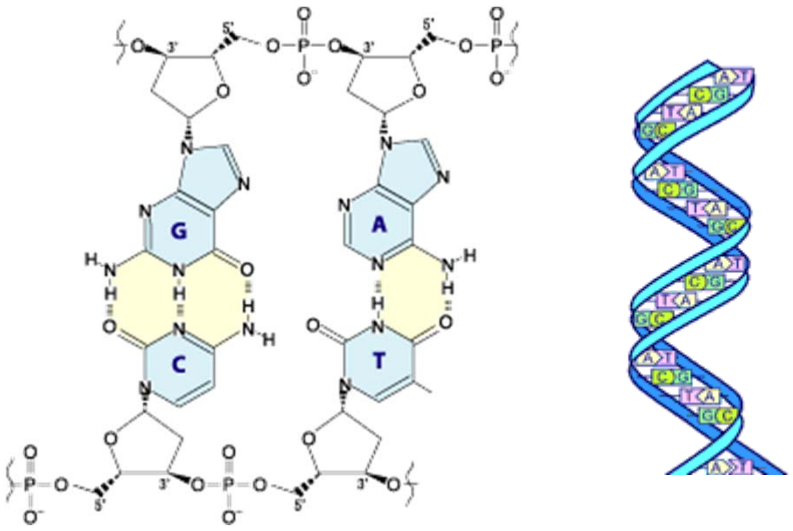
人工言語の“文”の例

```
void main()
{   init  x;   x = x+1;  }
```

```
<TABLE><TBODY>
   <TR vAlign=center align=left>
    <TD colSpan=2>[
    <A  href=“http://www.i.kyoto-
   u.ac.jp/~akihiro/index-e.html”  >English</A> |
      Japanese ]
    </TD></TR>
</TBODY></TABLE>
```

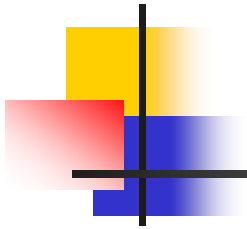
DNA

- DNA, RNA配列の類似性



Wikipediaより

…ATTTCACTTGGGTTTAAAATG
CTGTGACCTTGAGTAAGTTGCC
GTCTCTGAATCTGATCCTTTCG
ATTTCCCATTCTCCAAACTGAG
AACTAGCACTGCTGAGACGTGG
TTATTTCCAATAATAATTTGTA
TATTTTACATAACGCACCACAC
CAACATCTTCACCCAGTTGGAG
CCTACTCCTTTGCTCCCGCTG…



形式言語理論

言語を文字列・単語列の集合と抽象化した上で

- 文を生成（構成・定義）するための理論
  - 正しい文字列・単語列を生成（定義）する
- 文を受理するための理論
  - 送られてきた文字列・単語列が正しいかどうかを判定する

- 句構造文法と受理機械

計算機科学と“言語”(1)

- ある規則性(文法)にしたがって並べられた
  文字または音素からなる語
  語(形態素)からなる文
  の集合

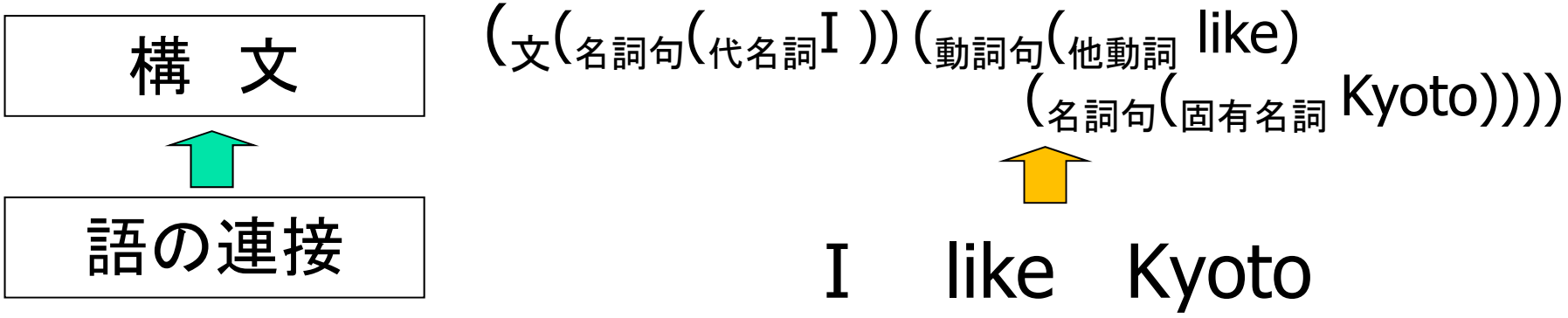
- 意味と語や文を切り離す.
  - 文法の機能に注目

字句解析

英語で記述された文の場合

- 英語で文を記述する際には，単語間に空白を入れる．(分かち書き)

  ➡単語を基本単位として構文を解析すればよい



トークン(token)

- プログラミング言語，マークアップ言語などでは，操作の最小単位をトークンとよぶ

例 C言語の場合

```
void  main ( ) {
        int  par ;
        par = par + 12 ;
}
```

トークン(token)

- マークアップ言語，プログラミング言語などでは，処理の最小単位をトークンとよぶ

例 C言語の場合

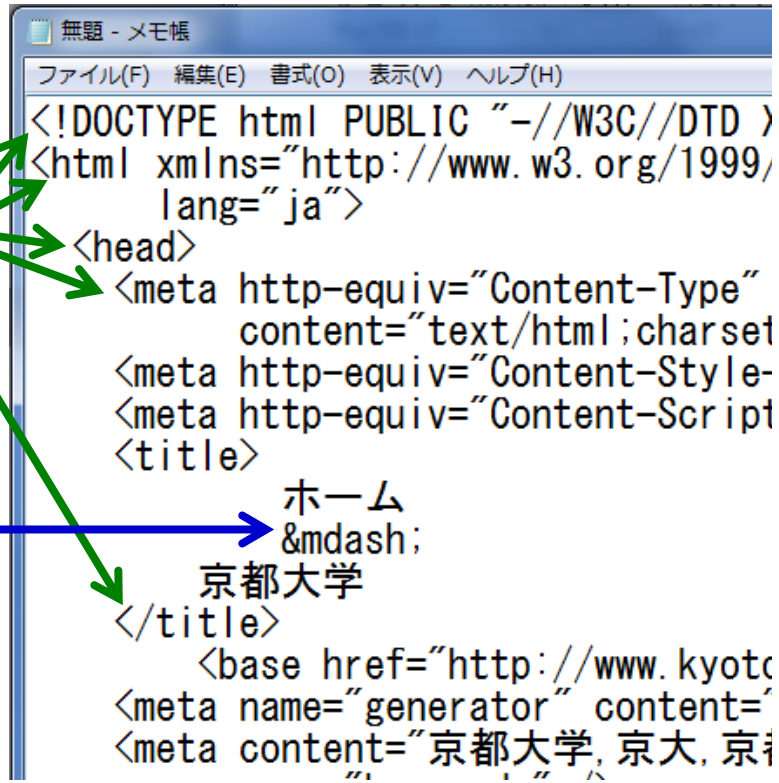
```
void  main ( ) {
        int  par ;
        par = par + 12 ;
}
```

トークン(token)

- プログラミング言語，マークアップ言語などでは，操作の最小単位をトークンとよぶ

例 HTMLの場合

- '<' と '>'で挟まれた文字列
  '</' と '>'で挟まれた文字列
  (タグとよぶ)
- '&'から始まり';'や改行で終わる文字列
- それ以外の部分に現れる一つずつの文字

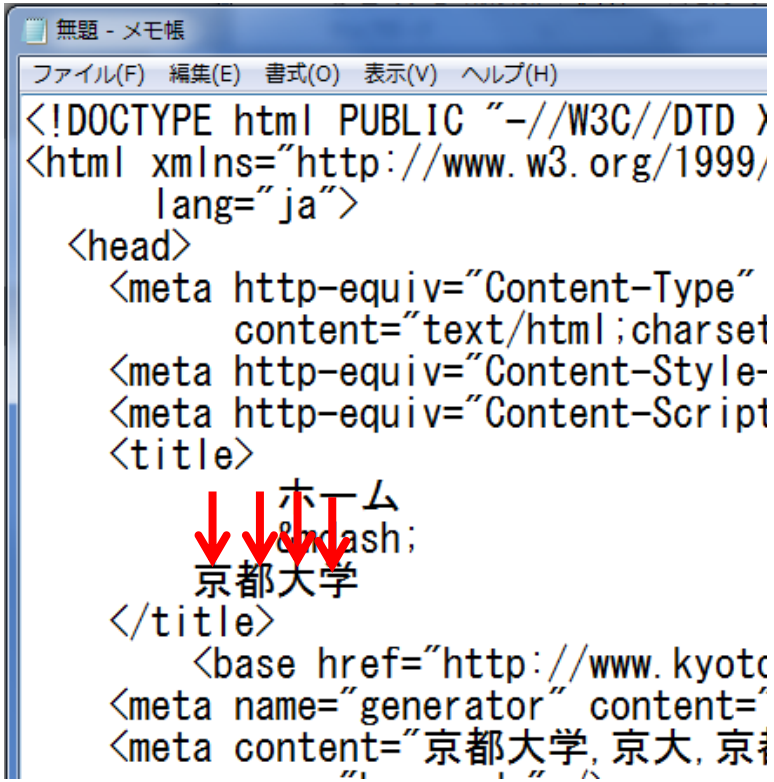


トークン(token)

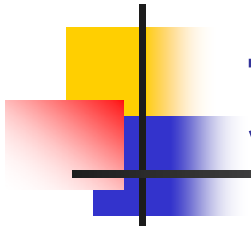
- プログラミング言語，マークアップ言語などでは，操作の最小単位をトークンとよぶ

例 HTMLの場合

- '<' と '>'で挟まれた文字列
  '</' と '>'で挟まれた文字列
  (タグとよぶ)
- '&'から始まり';'や改行で終わる文字列
- それ以外の部分に現れる一つずつの文字



構文解析

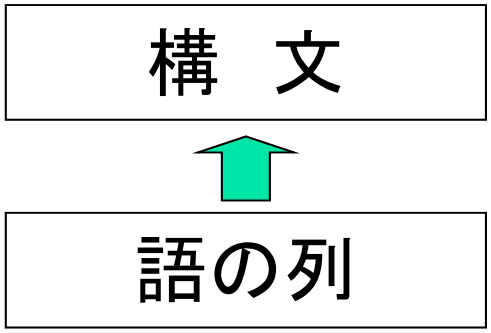


文法と構文

- 通常，ある言語において，文を単語や形態素，トークン(形態素)の列で構成するときには，ある規則に従っている.
- 規則を文法，文法に従った文が持つ構造を構文とよぶことにする

英語で記述された文の場合

- 単語の品詞と文中での位置が構文を決める.
- 構文は 単語⇒句⇒節⇒文　のように階層を成す

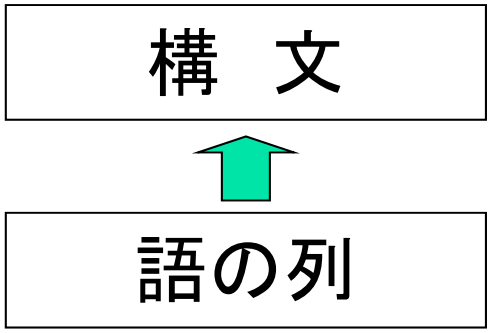


The quick brown fox jumps over the lazy dog.

冠詞　形容詞　形容詞　名詞　動詞　前置詞　冠詞 形容詞　名詞

英語で記述された文の場合

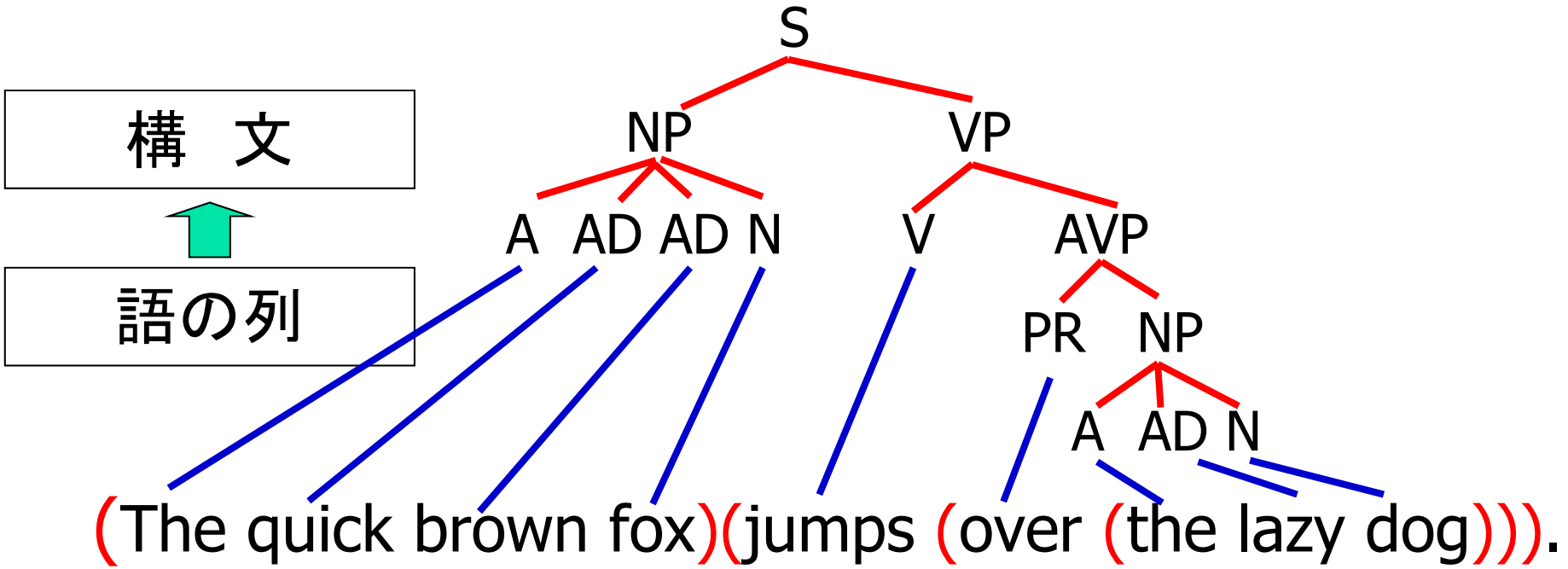
- 単語の品詞と文中での位置が構文を決める.
- 構文は 単語⇒句⇒節⇒文　のように階層を成す



(The quick brown fox)(jumps (over (the lazy dog))).

英語で記述された文の場合

- 単語の品詞と文中での位置が構文を決める.
- 構文は 単語⇒句⇒節⇒文　のように階層を成す



句構造文法(1)

- 文法を“節や句の構造を生成するための規則”ととらえる

例 I like Kyoto.

(文(名詞句(代名詞I))(動詞句(他動詞 like) (名詞句(固有名詞 Kyoto))))

HTML(XML)風に

```
<文>
  <名詞句> <代名詞> I </代名詞> </名詞句>
  <動詞句> <他動詞> like </他動詞>
       <名詞句> <固有名詞> Kyoto</固有名詞> </名詞句>
  </動詞句>
</文>
```

句構造文法(2)

- 生成規則: $\alpha \rightarrow \beta$ という形の規則

例　文 → 名詞句 動詞句

名詞句 → 代名詞　　動詞句 → 動詞

名詞句 → 固有名詞　　動詞句 → 動詞 名詞句

名詞句 → 冠詞 名詞　　…

…

代名詞 → I　代名詞 → you　代名詞 → he　…

固有名詞→ Kyoto　固有名詞→ Tom …

動詞 → like　動詞 → have …

句構造文法(3)

- 生成規則 $\alpha \rightarrow \beta$ のα, βは
  - 非終端記号(構文要素)
    - 文, 名詞句, 動詞句,代名詞,動詞, ...
  - 終端記号(単語・文字)

  の有限列

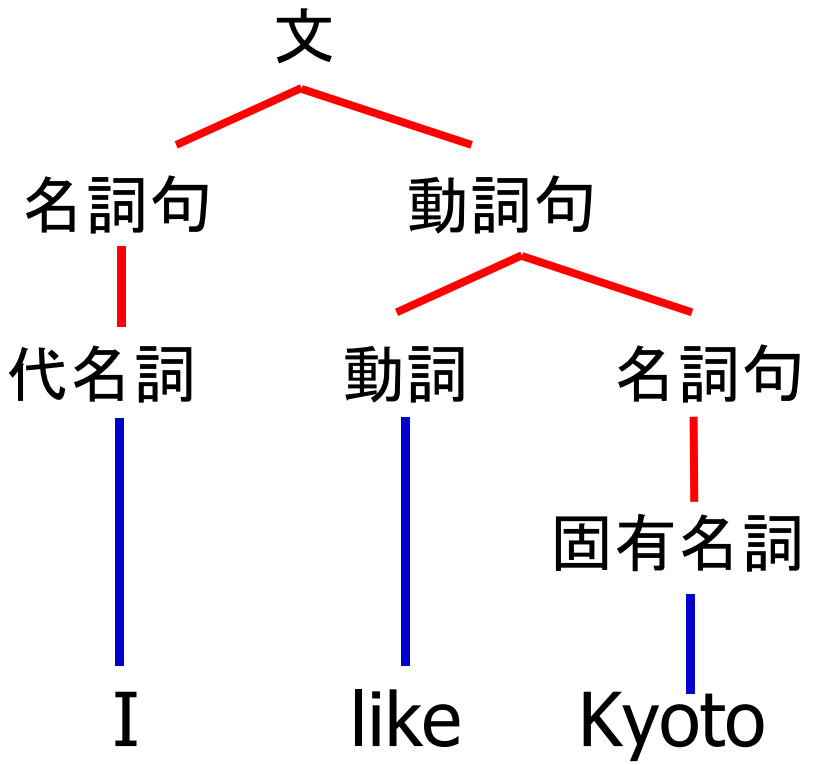
- 文法 G：生成規則の集合

  と非終端記号，終端記号，文全体(開始)を表す非終端記号

- Gによって生成される文: “文” に生成規則を有限回適用して得られる終端記号列

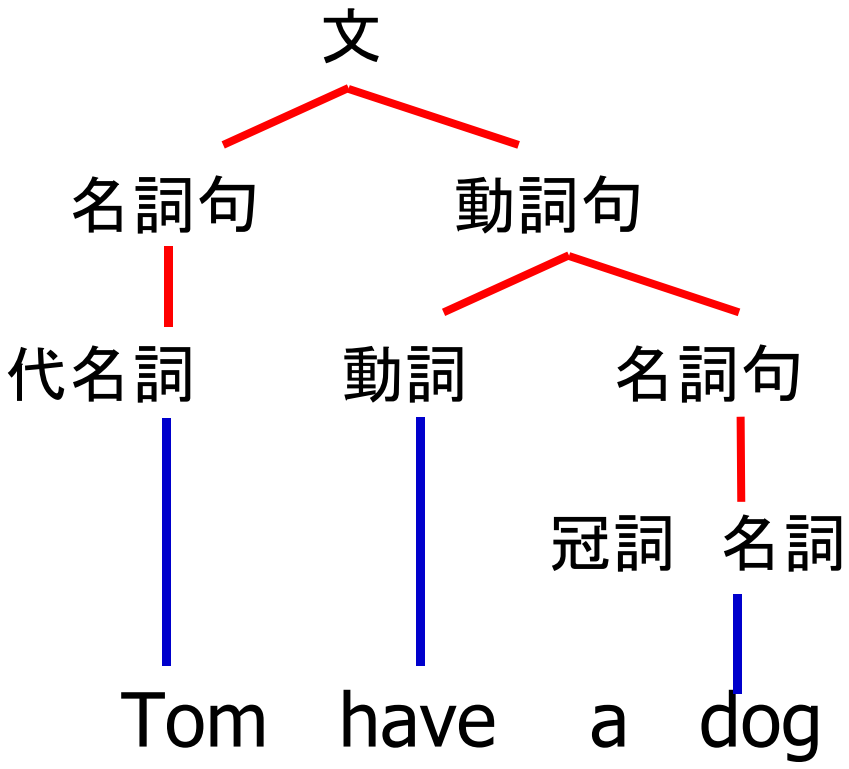
文法に従った導出の例(1)

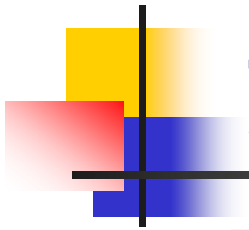
文
- |– 名詞句 動詞句
- |– 代名詞 動詞句
- |– 固有名詞 動詞句
- |– I 動詞句
- |– I 動詞 名詞句
- |– I like 名詞句
- |– I like 固有名詞
- |– I like Kyoto



文法に従った導出の例(2)

文 |– 名詞句 動詞句
|– 代名詞 動詞句
|– 固有名詞 動詞句
|– Tom 動詞句
|– Tom 動詞 名詞句
|– Tom have 名詞句
|– Tom have 冠詞 名詞
|– Tom have a 名詞
|– Tom have a bag





文法の型

- 文脈自由文法(context free): 文法中の全ての生成規則$\alpha \rightarrow \beta$ について，αが非終端記号一つであるもの
  - 三人称や複数の扱いを表現する際には，複雑な生成規則を用いなければならない.
  - 生成規則に条件を変えることで，様々な定義能力を持つ文法の型が導入できる.

構文解析(Parsing)

- 単語(文字)の列σが与えられたとき，その構文を決定すること

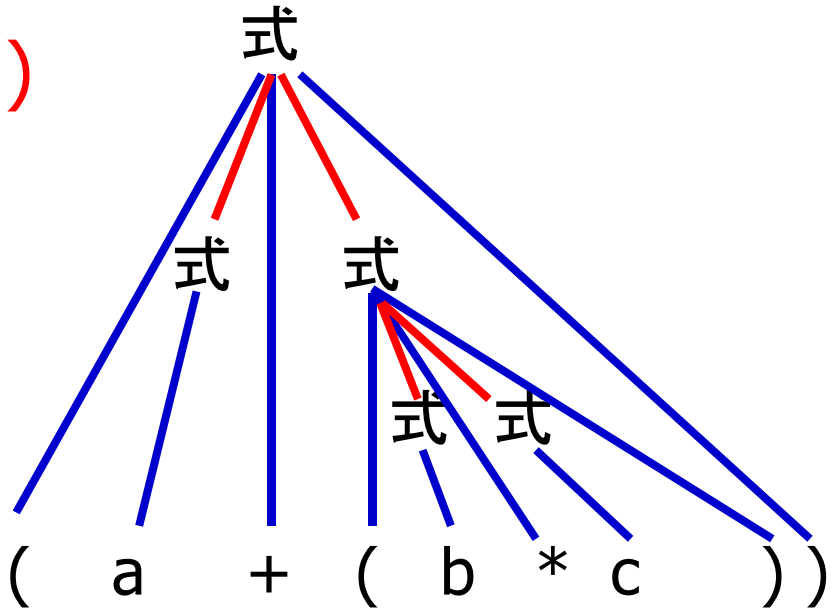
- 単語(文字)の列σが与えられたとき，“文”から始まりσで終わる導出を構成する.

- 構文解析の結果得られる構文を表す木を構文木(parse tree)とよぶ

数式と文法(1)

非終端記号: 式
終端記号：a, b, c, +, *, (,)
開始記号：式

式 → (式 + 式)
式 → (式 * 式)
式 → a, b, c

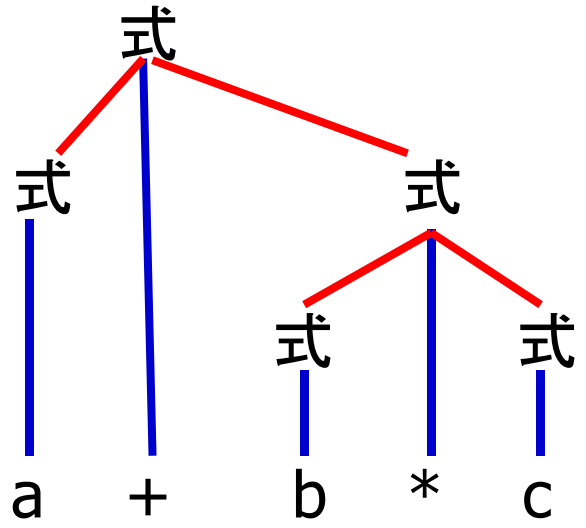


式 |– (式 + 式) |– (a + 式) |– (a + (式 * 式))
|– (a + (b * 式) |– (a + (b * c))

数式と文法(2)

非終端記号: 式
終端記号：a, b, c, +, *
開始記号：式

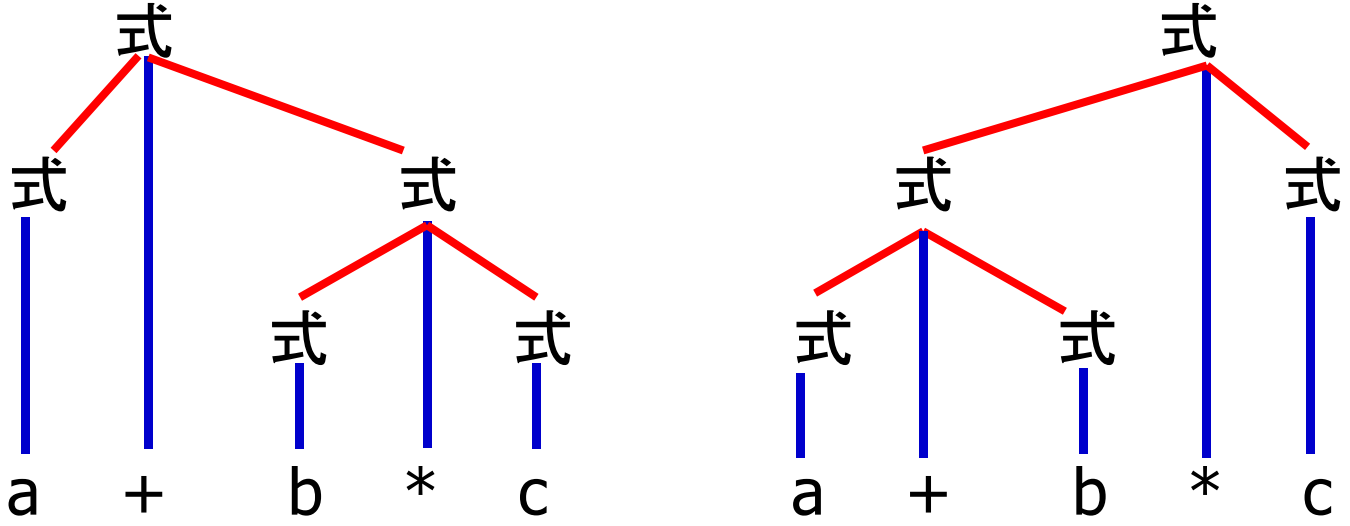
式 → 式 + 式
式 → 式 * 式
式 → a, b, c



式 |– 式 + 式 |– a + 式 |– a +式 * 式
|– a + b * 式 |– a + b * c

曖昧な文法

- ある文法を用いると，一つの単語(文字)の列σに対して異なる2種類以上の導出が構成されることがあるとき，その文法は曖昧である.



数式と文法(3)

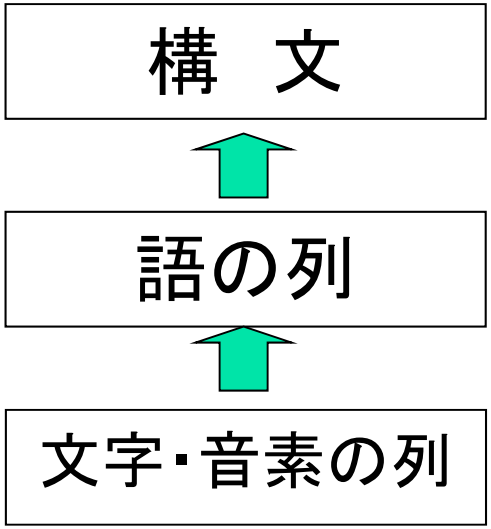
式 → 和
和 → 和 + 和
和 → 積
積 → 積 * 積
積 → 変数
変数 → a, b, c

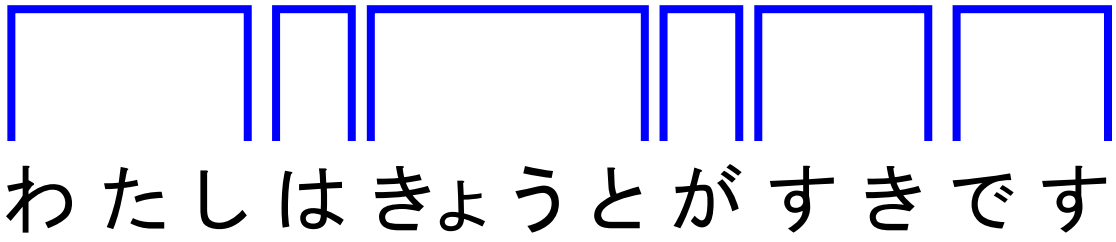
式 |– 和 |– 和 + 和 |– 積 + 和 |–変数 + 和 |– a + 和
|– a + 積 |– a + 積 * 積 |– a + 変数 * 積
|– a + b * 積 |– a + b * 変数 |– a + b * c

日本語で記述された文の場合

- 日本語で文を記述する際には，単語間に空白を入れない．

  ⇒ 単語を推定してから構文を解析しなければならない．





形態素(morpheme)

- 文字列(音素列)において意味を持つ最小単位
  - それ以上分解すると意味をなさなくなる

例　わ た し は きょうと に りょこう する

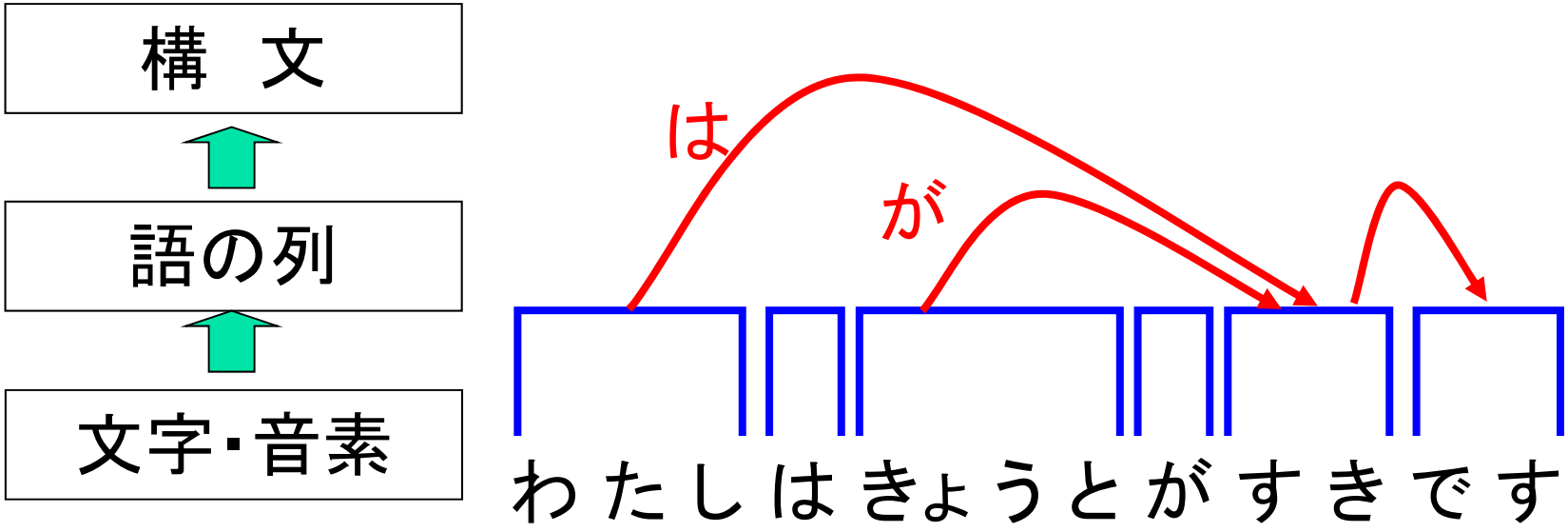
　　すもも も もも も もも の うち

  - 自然言語については，一つの言語に対して，複数の文法が考案されている．

    ⇒ “単語”の定義は文法に依存する
  - 英語でも，接頭辞(un-, non-など)や接尾辞(-tion, -ible)などは一つの形態素として扱う

日本語で記述された文の場合

- 単語や句の修飾･被修飾関係が構文を決めている



プログラミング言語の場合

- 数式と同様の構文規則を用いる
- トークン自体にも構文規則がある

例 正整数の10進表現 → ０以外の数字 数字列

数字列→ 数字 数字列

数字 → ０

０以外の数字→ ０

０以外の数字→ ２

０以外の数字→ ４

数字 → ０以外の数字

０以外の数字→ １

０以外の数字→ ３

…

０以外の数字→ ９

マークアップの場合(1)

DTD宣言→ <!DOCUTYPE DTD属性列>

文書開始タグ→ <html 文書属性列>

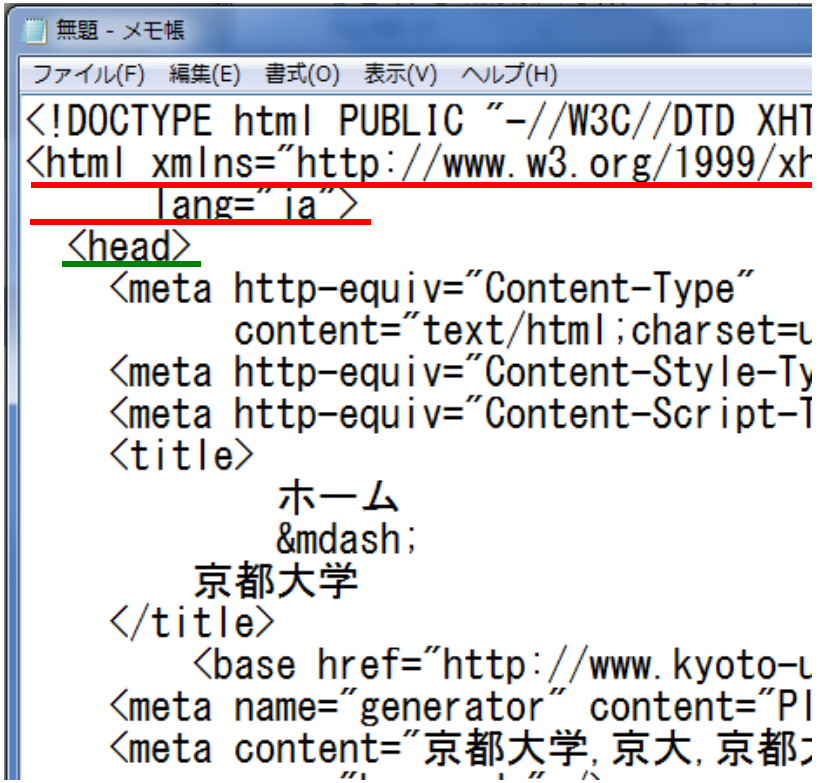
文書終了タグ→ </html>

頭部開始タグ→<head>

頭部終了タグ→</head>

本体開始タグ→<body>

本体終了タグ→</body>



マークアップの場合(2)

文書 → DTD宣言 文書開始タグ 頭部 本体 文書終了タグ

頭部 → 頭部開始タグ タグ列 タイトル部 タグ列
頭部終了タグ

タイトル部→タイトル開始タグ 文字列 タイトル終了タグ

本体 → 本体開始タグ タグと文字列 頭部終了本体

- 直観的には, 数式の括弧 '(' , ')' がそれぞれ開始タグ, 終了タグに対応し, 定数(変数)が文字列に対応する

機械学習と形式言語理論

テキストマイニング

- 人工知能(機械学習)データマイニング技術，統計学を文書に対して適用することにより分析を行う手法を“テキストマイニング”という.
  - 機械学習：元来は，人工知能に持たせる“学習する機能”を意味していたが，現在では統計的データ分析アルゴリズム全般を指している.
  - データマイニング：大量のデータに潜む隠れた規則性を発見すること

単語の統計を利用する

- 最近のニュース記事を集めて，ニュースのジャンル別に出現する単語の頻度を計測する

例

1: 首相 が 党首 討論: 政治

2: 安倍 首相 が 会談: 政治

3: 阪神 が 連勝: スポーツ

4: 日経 平均 が 反落: 経済

5: 首相 が 日銀 総裁 と 会談: 経済

記事番号	首相	が	党首	…	連勝	…	日経	…	クラス
1	1	1	1		0		0		政治
2	1	1	0		0		0		政治
3	0	1	0		1		0		スポーツ
4	0	1	0		0		1		経済
5	1	1	0		0		0		経済

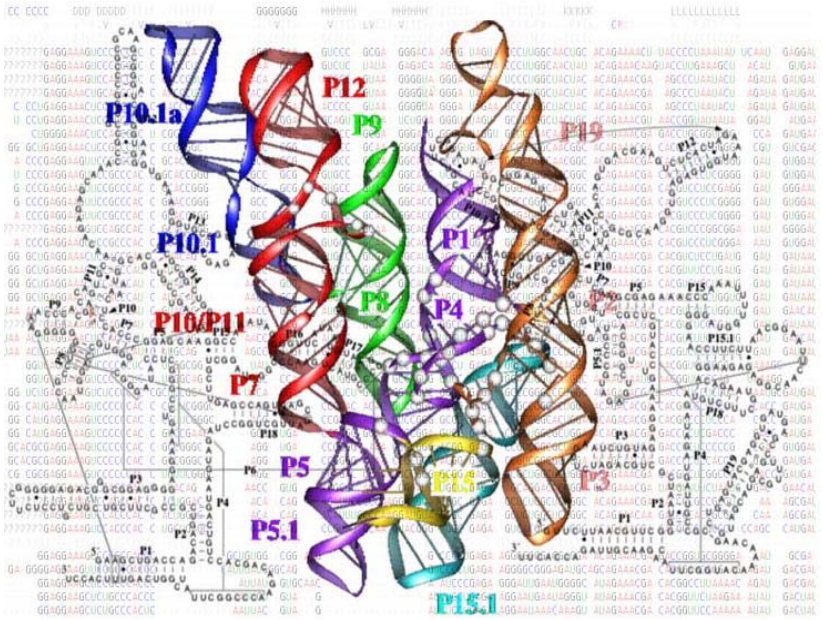
文書の自動分類

- 新しいニュース記事のジャンルを自動で分類
  - “迷惑メールのフィルタリング”で有名になった

例　もし“首相”という単語を含む記事が新たに来たときに，そのジャンルは何か？

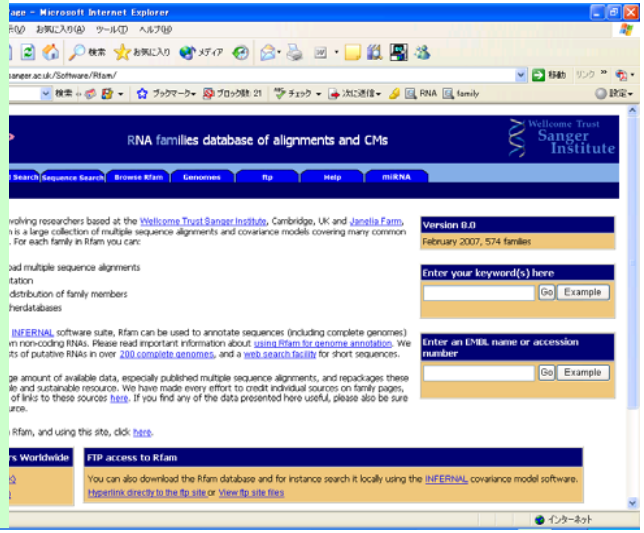
記事番号	首相	が	党首	…	連勝	…	日経	…	クラス
1	1	1	1		0		0		政治
2	1	1	0		0		0		政治
3	0	1	0		1		0		スポーツ
4	0	1	0		0		1		経済
5	1	1	0		0		0		経済
…									

RNA Classification



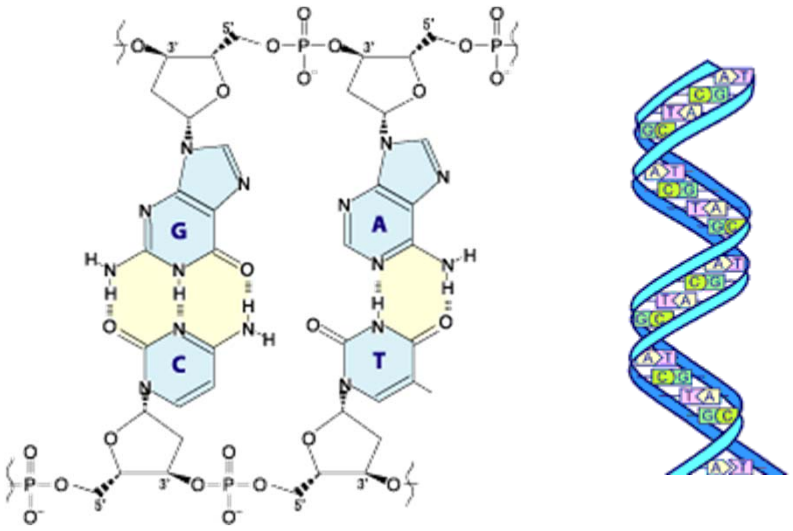
Non-coding RNA (ncRNA) is an important molecule in bioinformatics, and is considered to be a factor of the difference between higher organism and others

RNA sequences are accumulated in **RNA families**, and the members in each family have similar structures and functions. (Rfam database)



DNA

- DNA, RNA配列の類似性



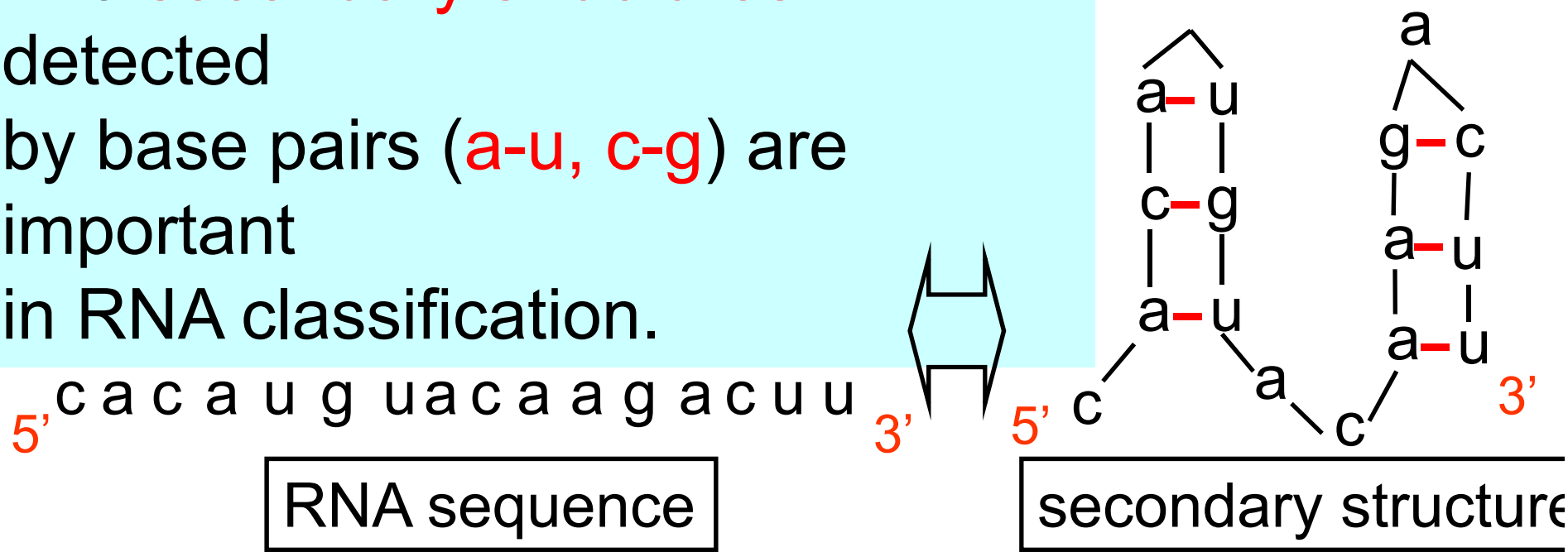
Wikipediaより

…ATTTCACTTGGGTTTAAAATG
CTGTGACCTTGAGTAAGTTGCC
GTCTCTGAATCTGATCCTTTCG
ATTTCCCATTCTCCAAACTGAG
AACTAGCACTGCTGAGACGTGG
TTATTTCCAATAATAATTTGTA
TATTTTACATAACGCACCACAC
CAACATCTTCACCCAGTTGGAG
CCTACTCCTTTGCTCCCGCTG…

データモデリングの例：蛋白質

- 一次構造: アミノ酸の配列
- 二次構造: α－ヘリックス構造、βシート構造
- 三次構造: 立体構造
- 四次構造: タンパク質が数個集まってできる構造

The secondary structures detected
by base pairs (a-u, c-g) are important
in RNA classification.



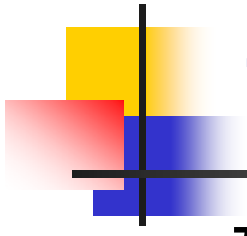
Our purpose

To distinguish between the member sequences in a given RNA family and non-member sequences.

State of art in this field

SVM combined with kernel functions for structured data.
e.g. string kernel
which

記号列の数学



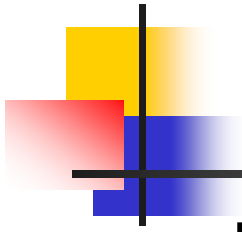
記号列の数学への誘い

- 列は計算機科学の基本をなしている
  - 1と0の列
  - ASCII文字の列
  - ひらがな,漢字の列
  - 単語の列
  - A, T, C, Gの列

- 四則演算のような強力な演算がない
  - 連接: $s \cdot t$　記号列 s の後に t をつなげる

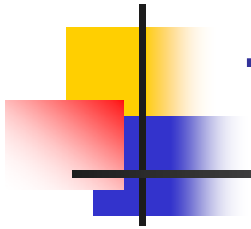
    結合律: $s \cdot (t \cdot u) = (s \cdot t) \cdot u$

    単位元: 空の記号列 $\varepsilon \cdot s = s \cdot \varepsilon = s$
  - 演算をアルゴリズムで定義する



形式言語理論では

- 操作の最小単位を記号または文字と呼ぶ
  - 英単語から構成される文を議論するときには，各英単語を記号もしくは文字として扱う
  - トークンからなる式を議論するときには，トークンを記号もしくは文字として扱う
  - トークンの構成を議論するときには，数字や英文字を各英単語を記号もしくは文字として扱う
- 対象とする記号または文字の有限集合をアルファベット(alphabet)とよぶ
- 有限長の記号からなる列を記号列または語(word)とよぶ



記号列データ

- Σ：アルファベット(alphabet)
- Σ^*：Σ中の記号からなる有限列全体の集合
  - 空列をεで表す
  - $\Sigma^+ = \Sigma^* - \{\varepsilon\}$

Example 1

$\Sigma = \{a, b\}$

$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots, abaabab, \ldots\}$

Example 2

$\Sigma = \{A, T, C, G\}$

$\Sigma^* = \{\varepsilon, A, T, C, G, AA, AT, AC, AG, \ldots, AAA, AAT, \ldots\}$

列と系列

- 列：string　　系列：sequence
- 部分列(部分語)：substring　(subword)
  部分系列　：subsequence

例　w = abaab

ba, baa などはwの部分列(部分語) であり，
部分系列でもある

aaa や bbはwの部分列(部分語) ではないが，
部分系列である

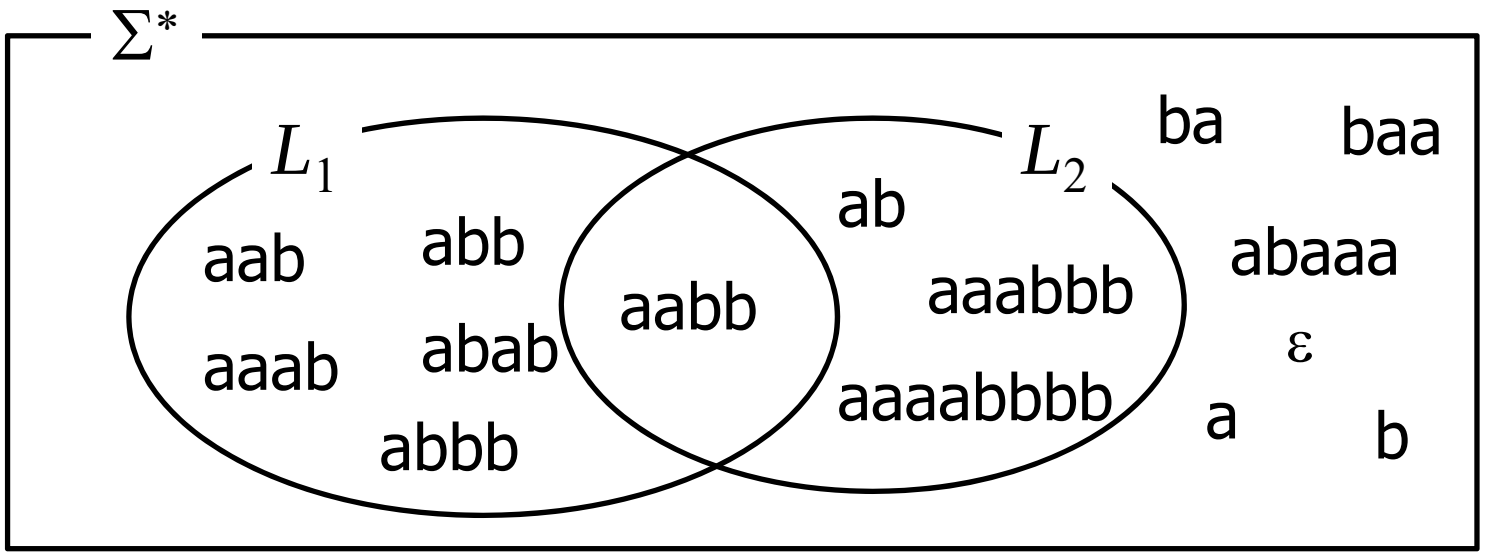
形式言語

- 全体集合をΣ^*とするとき，Σ^*の部分集合を形式言語(あるいは単に言語)とよぶ.
  - 以後，アルファベットΣが文脈から明らかなときは，単に記号列の集合とよぶこともある.

例 $\Sigma = \{a, b\}, \Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

$L_1 = \{aab, abb, aaab, aabb, abab, abbb, \ldots\}$

$L_2 = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$



集合の内包的表現と規則性

集合を表す2種類の方法

- 要素を並べて表す(外延的)

$$\Sigma = \{a, b\}$$

$$L = \{ab, aab, abab, aaab, abaab, aaaabbbb, \ldots\}$$

- 全体集合の要素がその集合に属する条件を用いて表す(内包的)

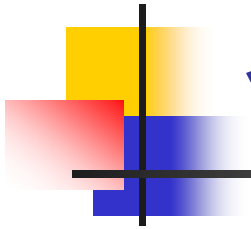
$$L = \{w \in \Sigma^* \mid w \textit{ starts with } a \textit{ and ends with } b\}$$

用語と記号(1)

アルファベット Σ を一つ固定する.

- 記号列 $w = c_1 c_2 \ldots c_n$ の長さを $|w| = n$ と定義する.
  空列εに対しては $|\varepsilon| = 0$ と定義する.
  例 $|abb| = 3, |abaabab| = 7$
- 形式言語 L と自然数 n に対して形式言語$L|_n$ を
  $L|_n = \{w \in \Sigma^* \mid w \in L$ かつ $|w| = n\}$ と定義する.
- 形式言語 L, M に対して形式言語 $L \cdot M$ を
  $L \cdot M = \{ uv \in \Sigma^* \mid u \in L$ かつ $v \in M\}$と定義する.
  - 演算子・はしばしば省略することがある.

  例 $L = \{ab, abb, ba\}$, $M = \{\varepsilon, bb, bab\}$ とするとき
  $L \cdot M = \{ab, abb, ba, abbb, abbbb, babb, ba, babb, babab\}$.



命題と注意

命題　アルファベット Σ を一つ固定する.

形式言語 L, M, N に対して $(L \cdot M) \cdot N = L \cdot (M \cdot N)$.

したがって，$(L \cdot M) \cdot N$ を $L \cdot M \cdot N$ と表記する.

注意　一般には　$L \cdot M \neq M \cdot N$.

用語と記号(2)

- 形式言語 L と非負整数n に対して形式言語L^nを以下のように帰納的に(再帰的に)定義する.
  - $L^0 = \{\varepsilon\}$. $(L^0 \neq \varnothing$に注意$)$
  - $L^1 = L$.
  - $L^{n+1} = L \cdot L^n$.

注意1. 形式言語理論では L^nはLのn回の直積$L \times L \times \cdots \times L$ではない.

注意2. $L \cdot L^n = L^n \cdot L$ が成り立つ.

例 $L = \{$ab, abb, ba$\}$, $M = \{\varepsilon$, bb, bab$\}$ とするとき

$L^2 = \{$abab, ababb, abba, abbab, abbabb, abbba,
baab, baabb, baba$\}$,

$M^3 = \{\varepsilon$, bb, bab, bbbb, bbbab, babbab, babbb, babbab,
bbbbbab, bbbabbab, bbbabbb, bbbabbab
babbbbab, babbabbab, babbabbb, babbabbab$\}$

用語と記号(3)

- 形式言語 L に対して L^* を

$$L^* = L^0 \cup L^1 \cup \ldots \cup L^n \cup \ldots$$

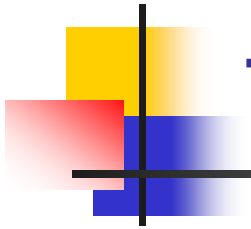
$$= \{w \in \Sigma^* \mid \text{ある非負整数} n \text{ が存在して } w \in L^n\}$$

と定義する.

例 $L = \{\text{ab, abb}\}$ とするとき

$L^* = \{\varepsilon$, ab, abb, abab, ababb, abbab, abbabb,
ababab, abababb, ababbab, ababbabb,
abbabab, abbababb, abbabbab, abbabbabb,
abababab, ababababb, abababbab, abababbabb,
ababbabab, ababbababb, ababbabbab, ababbabbabb
abbababab, abbabababb, abbababbab, abbababbabb,
abbabbabab, abbabbababb, abbabbabbab, abbabbabbabb , …$\}$

受講上の注意



計算機科学 v.s. 数学

- 理論の構成方法は数学の構成方法を規範としている
  - 概念の数学的定義，定理，証明，...
- 論法は数学のそれとは似て非なるもの

論法の違いの例

定理 非負整数m, nには最大公約数dが存在する.

証明1　非負整数k の約数全体の集合を$I(k)$と表すとき，集合 $I(m)\cap I(n)$ はmとnの公約数の集合である. この集合は自然数の集合だから，必ず最小値 d が存在する.

証明2　非負整数mをnで割った商をq, 余りをrとすると,

$$m=nq+r_0 \quad (0 \le r < n)$$

だから, mとnの公約数はrの約数でなければならない.

またnとrの公約数はmの約数でなければならないから, mとnの最大公約数はnとrの最大公約数に一致する. 次に nをqで割った余りr_1 に対しても同様のことが成立する. こえを繰り返してr_0, r_1, r_2,…を次々求めれば, この数列は単調減少列だから必ず有限列になり，その最後の項がmとnの最大公約数である.