言語プロセッサ2014
-No.2-
平成26年9月29日

東京工科大学
コンピュータサイエンス学部
亀田弘之

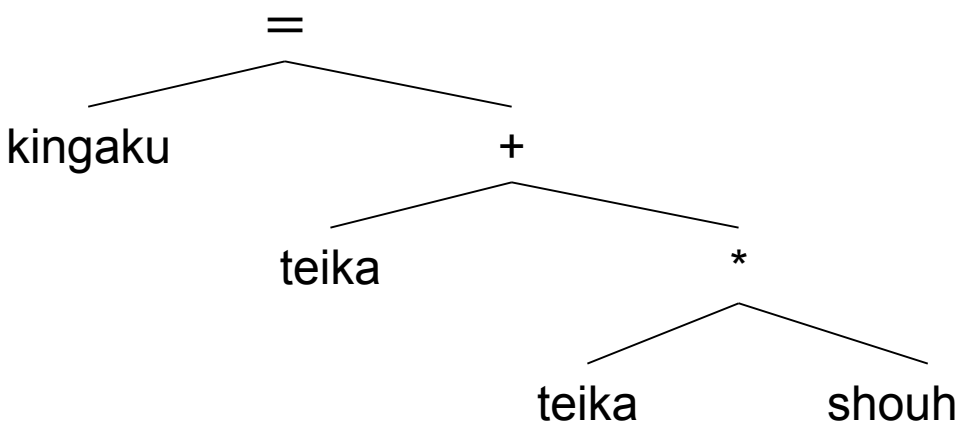
復習から

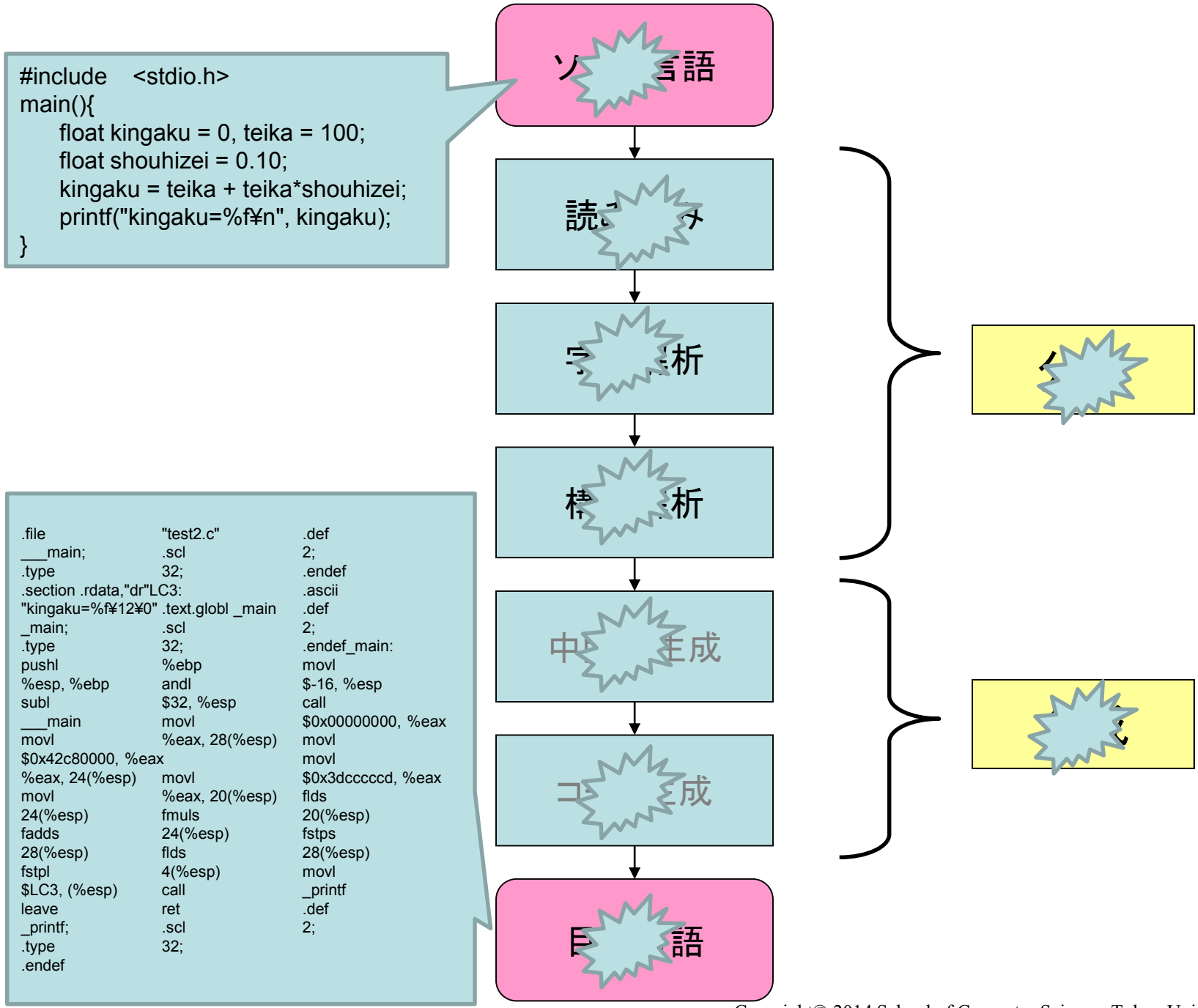
数式の解析

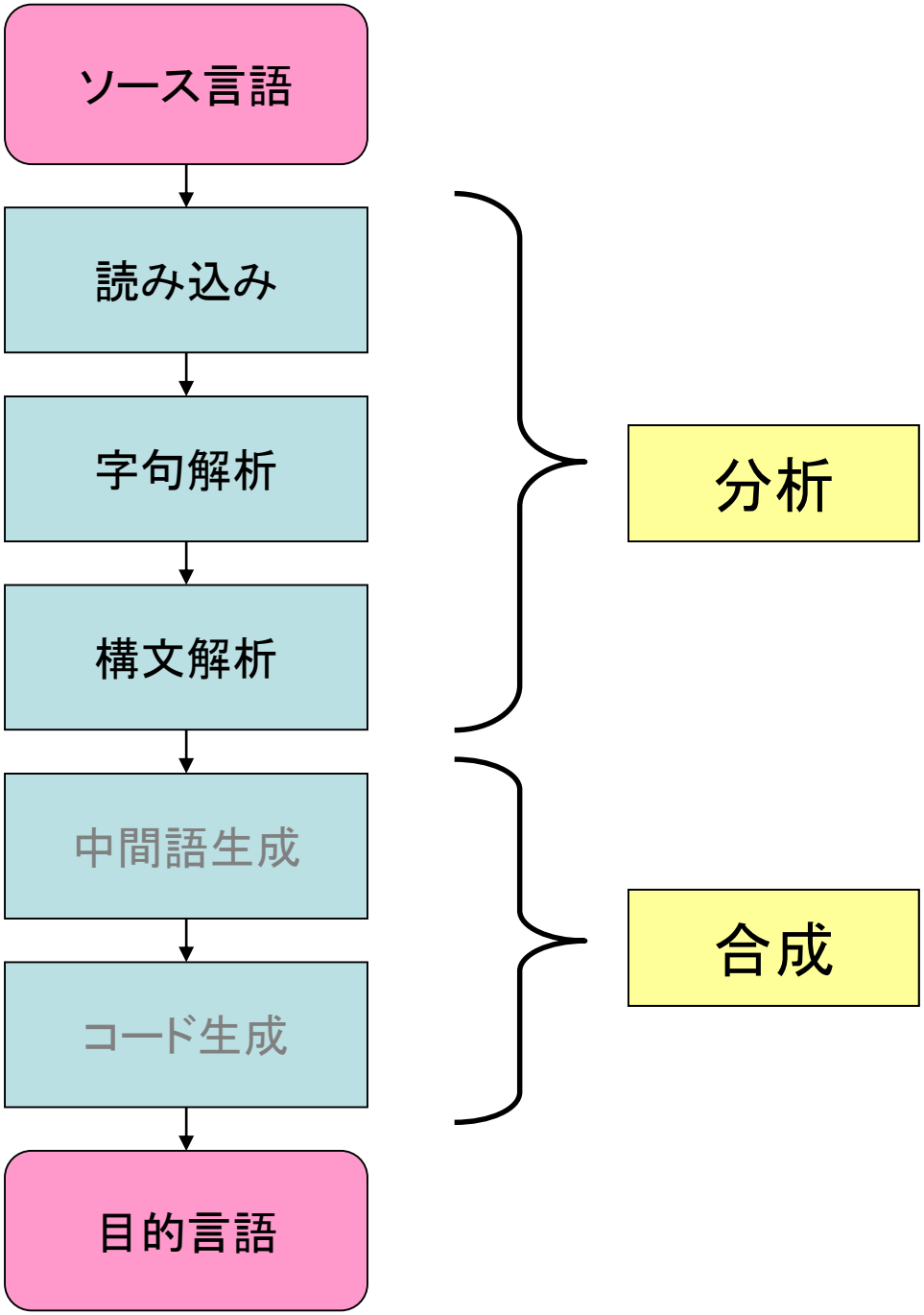
- kingaku = teika + teika * shouhizei

数式の解析

1. 読み込み（文字列として）
   “kingaku = teika + teika * shouhizei”
2. 要素(token)の切り出し
   “kingaku”, “=“, “teika”, “+”, “*”, “shouhizei”
3. 要素の相互関係の分析

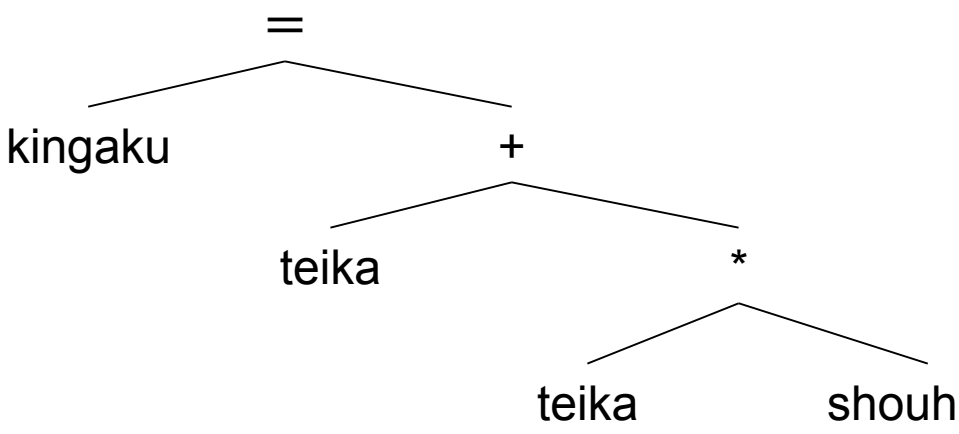






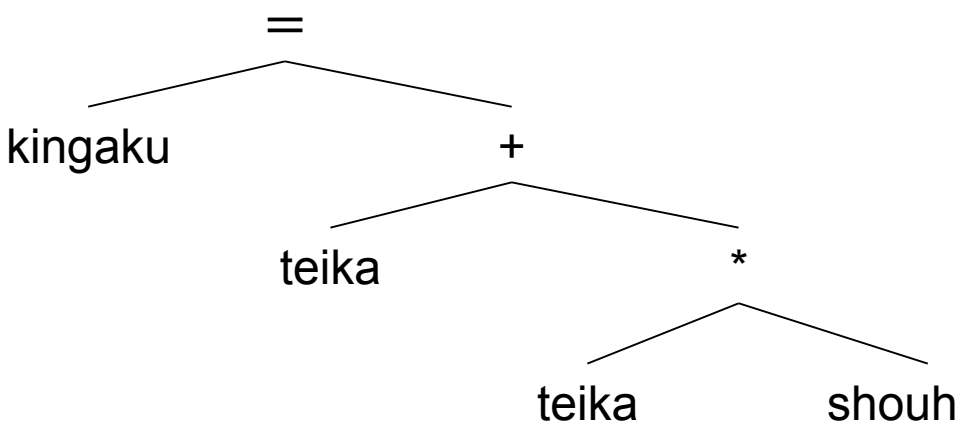
数式の解析

1. 読み込み（文字列として）
   “kingaku = teika + teika * shouhizei”
2. 要素(token)の切り出し
   “kingaku”, “=“, “teika”, “+”, “*”, “shouhizei”
3. 要素の相互関係の分析



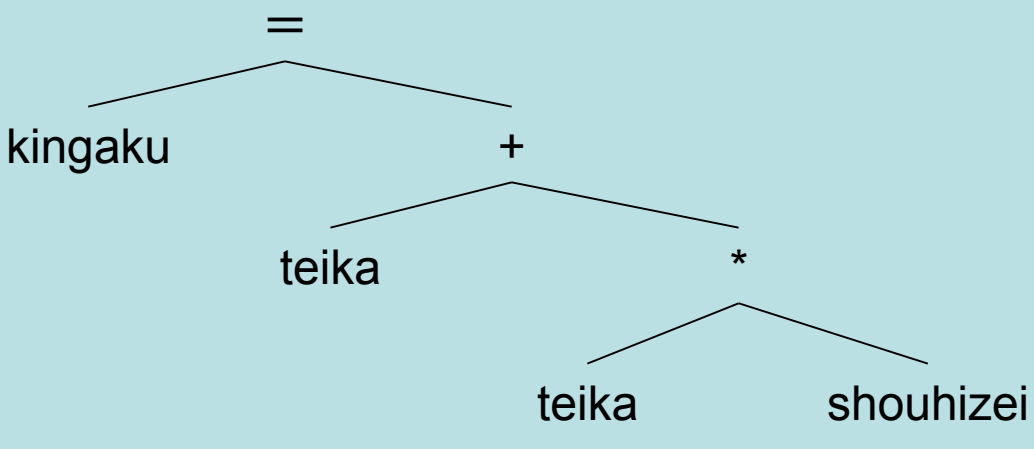
数式の解析

1. 読み込み（文字列として）
   “kingaku = teika + teika * shouhizei”
2. 要素(token)の切り出し
   “kingaku”, “=“, “teika”, “+”, “*”, “shouhizei”
3. 要素の相互関係の分析



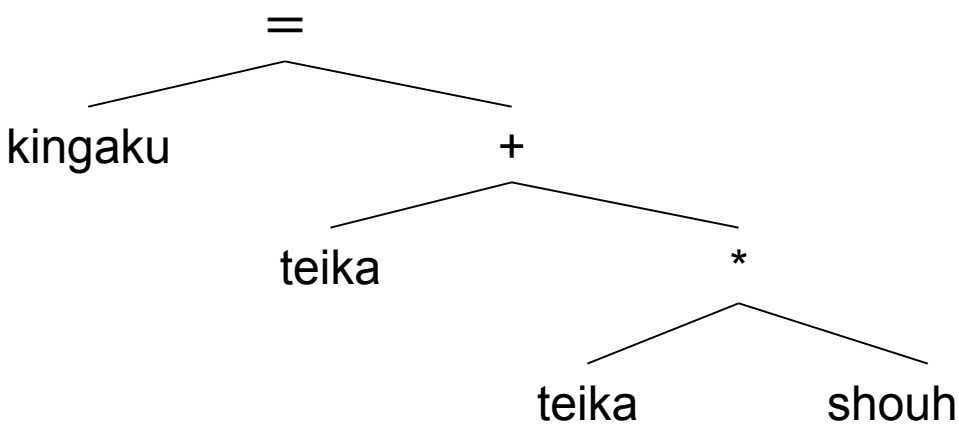
数式の解析

1. 読み込み（文字列として）
   “kingaku = teika + teika * shouhizei”
2. 要素(token)の切り出し
   “kingaku”, “=“, “teika”, “+”, “*”, “shouhizei”
3. 要素の相互関係の分析



数式の解析

1. 読み込み（文字列として） 読み込み
   “kingaku = teika + teika * shouhizei”
2. 要素(token)の切り出し　語句解析
   “kingaku”, “=“, “teika”, “+”, “*”, “shouhizei”
3. 要素の相互関係の分析　構文解析



数式の解析

1. 読み込み
   - ファイルからの入力技法
2. 字句解析
   - 有限オートマトンの理論
   - 正規文法
   - 正規表現
3. 構文解析
   - 線形有界オートマトン理論
   - 文脈自由文法

前提知識

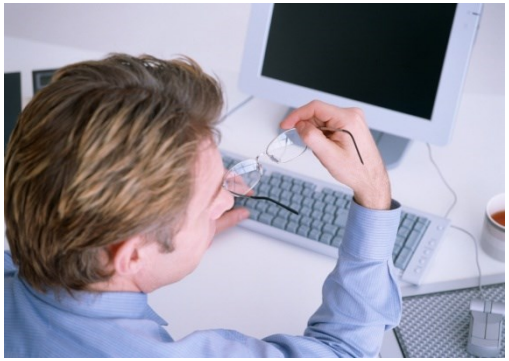
- 言語理論とオートマトン
  - 前期科目「言語理論とオートマトン」
  - 抽象的・論理的な思考への慣れ
    (特に、正規表現と有限オートマトン)
- プログラミング技法
  - 今までいろいろと習ってきましたよね！
  - 基本的な知識があれば一応OK
  - ファイルの入出力が難しい人もいるかも…

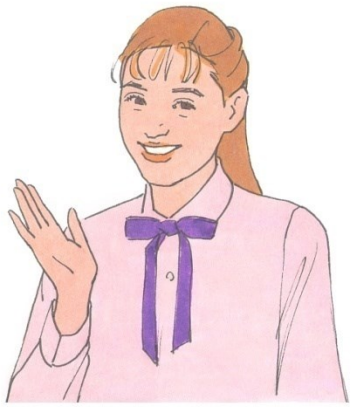
学んで得られるもの

- 言語理論とオートマトン
  - 抽象的・論理的な思考への慣れ
  - ソフトウェア分野における基本的概念
    - 正規表現 etc.
    - プログラミング言語へのより深い理解
- プログラミング技法
  - プログラミング力（知識）アップ
  - 洗練されたアルゴリズムの理解　　などなど

- 言語プロセッサ関連は、コンピュータサイエンスの英知が集積されている！

- この授業を取った人は先見の明がある！





今日の内容

1. 正規表現 (regular expressions; re)
2. オートマトン (automaton, *pl.* automata)
3. 正規表現とオートマトンとの関係
4. オートマトンのシミュレータ
5. デモ

今日の学習目標

1. 正規表現が使える。
2. オートマトンを理解し，作れる。

1. 正規表現

- **正規表現**とは
  - 聞いたことのある人は？　①
  - 使ったことのある人は？　②
  - 使い方説明できる人は？　③

どんな時に正規表現を使うのか？

正規表現とは

- 文字列のパターンを表現するためのもの
  - 例:
    1. { a, aa, aaa, aaaa, aaaaa, . . . }
    2. { 1, 01, 001, 0001, 00001, . . . }
    3. { aab, bab }
    4. { 2, 4, 6, 8, 10, . . . } (= 正の偶数全体の集合)

など

パターンが見つかりましたか？

練習

- いま current directory に以下のような(名前の)ファイルがあるとする。

program62　procedure71　profileV7　prototype52　parser
infile2010　filedummy777　outfile777　file256　fixer8

- 問１ proで始まるファイル名のもの
- 問２ fileを含むもの
- 問３ 末尾に数字がありそれが偶数のもの

正規表現(定義)

アルファベットVの上の正規表現とは以下のものである。

1. εは正規表現
2. a∈Vならば、aは正規表現
3. rとsが正規表現ならば rs は正規表現
4. rとsが正規表現ならば r | s は正規表現
5. rが正規表現ならば r* は正規表現
6. 以上のものだけが正規表現

正規表現の例

	正規表現	具体的な文字列
1.	a	{ }
2.	b	{ }
3.	ab	{ }
4.	a|b	{ }
5.	a|(a|b)	{ }
6.	a*	{ }
7.	(a|b)*	{ }
8.	a|b*	{ }
9.	ab*	{ }
10.	(ab)*	{ }

有限オートマトンの定義

$FA = (K, \Sigma, \delta, q_0, F)$

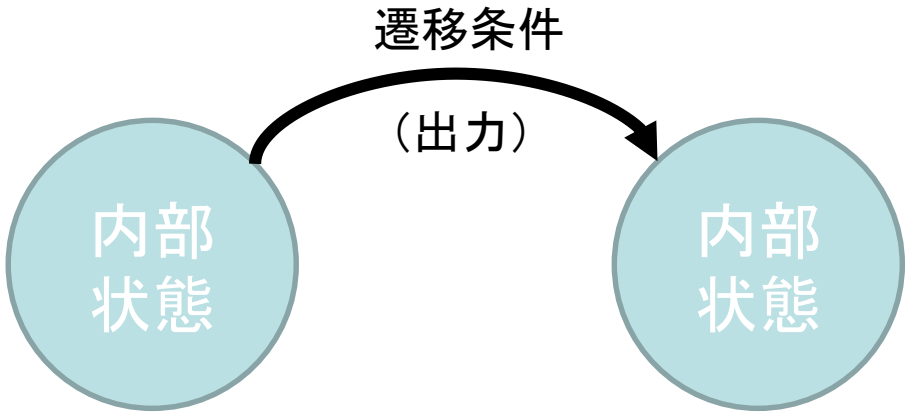
ただし、

- K ： 状態の集合(Kは有限集合)
- Σ ： 入力アルファベット(Σは有限集合)
- δ ： 状態遷移関数

$$\delta : K \times \Sigma \ni (q_i , a) \rightarrow q_j \in K$$

- q_0 ： 初期状態
- F ： 最終状態の集合 ($F \subseteq K$)

オートマトンのイメージ



初期状態
最終状態　など

オートマトンの例

- ゲーム人工知能
  - PackMan
  - RPG (Role Playing Games) など

今日の演習1

Packmanの観察

次のサイトに行って実際にPackmanを体験してみよう。(参考：http://j-game.net/packman.html)

【課題】

1. どのような内部状態が存在するか？
2. 内部状態を書き出す。
3. 内部状態同士の遷移関係を表す矢印を引く。
4. 遷移の条件およびそのときの動作を書き込む。
5. オートマトンとしての解釈を考える。

次の話題

オートマトンと正規表現の関係

- （有限）オートマトンと正規表現とは対応関係にある。
  つまり、(有限)オートマトンは正規表現で書き表せるとともに、正規表現は(有限)オートマトンで書き表せる。

今日の演習2

練習

問　正規表現 b(a|b)a を考える。

1. この正規表現が表わす文字列をすべて求めよ。
2. この正規表現の表わす文字列をすべて、かつ、それらだけを受理するオートマトンを作れ。

もっと練習

問　次の正規表現と等価なオートマトンを作れ。

1. ab
2. a*
3. 01(10|01)*11

オートマトンのシミュレータ

```
s = s0;
c = nextChar( );
while( c != eof ) {
   s = move( s, c );
   c = nextChar( );
}
if(sがFに含まれる) return “yes”;
else return “no”;
```

字句解析プログラムは、当然ながら自力で作成できるが、オートマトン理論を知っていればスマートに作成できる。

Flexを使った例

- 正規表現 a(a|b)*c を受理するオートマトンをシミュレートするプログラムの自動生成が可能。

デモ

- （詳しくは後日）

次回までの予習（準備）

- 自分が普段使っているテキストエディタでは、正規表現を扱えるか確認しなさい。
- 使えない場合は、正規表現が使えるエディタを探し、インストールしなさい。

本授業としては、復習をしっかりやすことをお勧めします。