

第1章 「オートマトン」の授業の概要

この授業の目的

計算の数学的モデルであるオートマトンとチューリングマシンについて学ぶ。
以下のような応用がある。

- プログラミング言語の“ ”を形式的に定める。
 - 理論 … 「オートマトン」（この授業）
 - 実践 … 「コンパイラ」（後期）

プログラミング言語の仕様は主に構文・文法 (syntax) に関する部分と意味 (semantics) に関する部分に分けられる。

構文 … プログラムのかたち・構造に関すること

どのような文字列がプログラム（あるいは、文・式）であるのか、ないのか？

例えば、“while { c++; }” や “1 + return n;” は C のプログラム（の一部）ではない。

正規表現や BNF などで記述する。

意味 … プログラムの実行結果に関すること、例えば、

“int i, n = 0; for (i = 0; i < 10; i++) n += i;”
を実行したあと、n は 45 になる。

表示的意味論・操作的意味論などの手法がある。詳しくは大学院の「プログラミング言語意味論」で扱う。

- コンピューターの理論的な限界を知る。

後期開講の「コンパイラ」と合わせて受講することを強く推奨する。

この授業を受講すべき理由

- オートマトンの直接の応用である ・ は、プログラミング言語処理系やデータマイニングだけではなく、広い範囲のアプリケーションプログラムで必要になる。

（何らかの文法を持つ）テキストの入力 → 構造を持つデータ → （処理） → 別形式の出力

また、字句解析・構文解析は計算機科学の古典であり、叡智の結集である。コンパイラやインタプリタなどのプログラミング言語処理系は大規模な記号処理プログラムの身近な例である。

コンパイラはソースプログラム（人間にとって理解しやすい形、C 言語なら `~.c`）をオブジェクトプログラム（CPU が直接理解できる形、Windows なら `~.exe`）に翻訳するプログラムである。

一方、インタプリタはソースプログラムを翻訳しながら実行する。

- 情報系の技術者にとってコンパイラやインタプリタは日常使用する道具であり、ブラックボックスのままにしておくわけにいけない。例えば、エラーメッセージの意味を理解する必要がある。
- 字句解析に使用する正規表現や、構文解析に使用する文脈自由文法の限界を知る必要がある。さらにはコンピューターに解くことができない問題があることを認識する必要がある。

この授業で学ぶ項目

問題: 「`"12+34*56"` のように式を表す文字列を受け取って、その値を計算するプログラム `calc` を作成せよ。」

例: `calc("12+34*56")` ⇒ 1916

`calc("x*(y+78)")` ⇒ ? (x と y に依存)

問題: 「`"12+34*56"` のように式を表す文字列を受け取って、その値を計算する機械語を生成するプログラム `compile` を作成せよ。」

字句解析

まず、どうやって単語に切り分けるのか ??? というところから始まる。

- `"12+34*56"` ⇒ `"12"`, `"+"`, `"34"`, `"*"`, `"56"`
- `"for(ans=0; ans<max; ans++) ..."`
↓
`"for"`, `"("`, `"ans"`, `"="`, `"0"`, `";"`, `"ans"`, `"<"`, `"max"`, `";"`, `"ans"`, `"++"`, `"")`,

ソースプログラム（や自然言語の文）を単語に切り分ける処理を _____ という。

字句解析のキーワード

字句 (lexeme), トークン (token), _____ (regular expression), 有限オートマトン (finite automaton), _____ (lexer generator), `lex`, `flex`, (正規言語の) 反復補題,

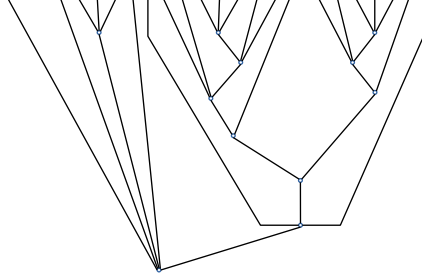
構文解析

- 切り分けた単語を、どうやってまとめた単位（例えば、文や式）にまとめて正しい順序で計算するのか？
- そもそも、式や文って何なのか？

プログラムの構造を木 (_____) などの形に表現することを _____ という。

- $12+34*56 \Rightarrow$ _____
- `while (i > 0) { n = n * i + 1; i = i - 1; }` \Rightarrow

```
while(i>0){n=n*i+1; i=i-1;}
```



構文解析のキーワード

文脈自由文法 (context-free grammar), バックス・ナウア記法 (Backus-Naur Form, BNF), プッシュダウン・オートマトン (pushdown automaton), (文脈自由言語の) 反復補題, 再帰下降構文解析 (recursive descent parsing), LR 構文解析 (LR parsing), 構文解析器生成系 (parser generator), yacc, bison,

※ 灰色の項目は主に「コンパイラ」で扱う。

チューリングマシン

計算可能とはどういうことか? コンピューターにできないことは何か?

チューリングマシンのキーワード

チューリングマシン (Turing machine), チャーチの提唱 (Church's thesis), 停止性問題 (halting problem),

